УДК 004.05

С.В. Гринюк, К.Е.Топчевська, П.С. Шолом
Луцький Національний Технічний Університет

КОДУВАННЯ ІНФОРМАЦІЇ ЗА ДОПОМОГОЮ САМОКОРЕГУЮЧОГО ШТРИХОВОГО КОДУ НЕМ-8.

С.В. Гринюк, К.Е.Топчевська, П.С. Шолом. Кодування інформації за допомогою самокорегуючого штрихового коду НЕМ-8. *В статті розглядається кодування інформації за допомогою штрих-кодів. Також перераховані основні принципи побудови саморугулюючого штрихового коду НЕМ-8. Розглянуто задачу по максимізації надійності штрихового коду та створено ефективний код, здатний запобігати неправильному зчитуванню закодованої інформації.*

Ключові слова: *штриховий код, інформація, код, повідомлення, алфавіт, алгоритм Хеммінга.*
Форм.3. Табл.5. Рис.3. Літ.7.

С.В. Гринюк, К.Э.Топчевская, П.С. Шолом. Кодирования информации с помощью самокорректирующее штрихового кода НЕМ-8. *В статье рассматривается кодирования информации с помощью штрих-кодов. Также перечислены основные принципы построения саморугулюючого штрихового кода НЕМ-8. Рассмотрена задача по максимизации надежности штрихового кода и создан эффективный код, способен предотвращать неправильном считыванию закодированной информации.*

Ключевые слова: *штриховой код, информация, код, сообщение, алфавит, алгоритм Хемминга.*
Форм.3. Табл.5. Рис.3. Лит.7.

S.V. Grynyuk, K.E.Topchevska, P.S. Sholom. Encoding information using samokorehuyuchoho barcode HEM-8.. *In the article the encoding information using bar codes. There are the main principles of the samoruhulyuyuchoho barcode Nam-8. The problem to maximize reliability barcode and create efficient code, is able to prevent the wrong read-encoded information.*

Keywords: *barcode, information, code, message, alphabet, Hamming algorithm.*
Form.3. Tab.5. Fig.3. Lit.7.

Вступ. У роботі проведено дослідження про кодування інформації за допомогою штрих-кодів. Сфера застосування штрихових кодів надзвичайно широка і вона весь час розширюється, але не зважаючи на це для більшості пересічних громадян ці чорні та білі смужки залишаються незрозумілими. Широке використання штрихових кодів було зумовлене необхідністю забезпечити автоматизоване введення інформації в комп'ютерні системи управління, що відрізнялося б високою надійністю, простотою і економічністю.

Штриховий код — це не щось особливе, існуюче саме по собі, а передусім елемент системи управління. В відриві від комп'ютерної системи управління, поза зв'язком з її інформаційною базою він не має жодного сенсу. Технологія штрихового кодування застосовується в багатьох сферах людської діяльності, але найбільш широко і ефективно вона використовується в оптовій і роздрібній торгівлі, управлінні матеріальними запасами, управлінні перевезеннями. Штрихові коди характеризуються високою надійністю.

Розглянемо основні принципи та правила, що використовуються при створенні штрихових кодів і які є обов'язковими для будь-якого їх типу. Одразу потрібно зазначити, що інформація яку ми кодуємо представлена в двійковому виді, тобто кодується двома значеннями: '0' та '1'. В штриховому кодуванні існує два способи задання цих значень, першим є спосіб, коли значення '0' та '1' кодуються відповідно двома кольорами - білим та чорним.

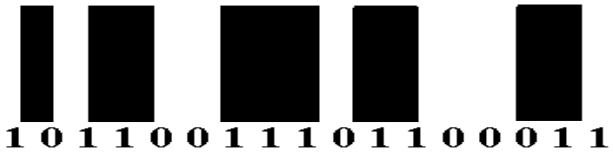

Рис. 1. **Штрих-код**

В цьому способі штрихи що відповідають '0' та '1' мають одинакову ширину. В разі якщо в бінарній послідовності йдуть одне за одним кілька одинакових n значень '0' чи '1' їм буде відповідати білий чи чорний штрих n-кратної ширини.

Другим способом представлення бітової послідовності в вигляді штрихового коду є спосіб коли '0' та '1' задані не різними кольорами, а різними значеннями ширини штрихів. Тобто маємо чотири атомарні графічні символи два вузькі штриха та два широкі білого та чорного кольорів. В такому штриховому коді білі та чорні штрихи весь час йдуть почергово, а значенням '0' та '1' відповідають відповідно широкі та візькі штрихи. В цьому разі наведена вище бінарна послідовність буде мати вигляд:

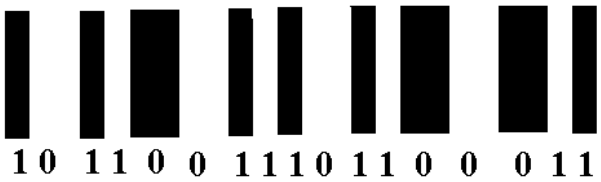

Рис. 2. **Бітова послідовність в вигляді штрихового коду**

В кожному з цих варіантів є як переваги так і недоліки. Так в першому варіанті штриховий код буде коротшим в наслідок того, що всі біти кодуються однаковими по ширині штрихами. По цій самій причині в першому варіанті штриховий код бінарної послідовності зі сталим числом бітів буде мати сталий розмір, в той час як в другому варіанті розмір штрихового коду буде залежати від співвідношення нулів та одиниць. Але недоліком першого варіанту є те, що при великій кількості йдучих один за одним одинакових бітів їх графічне представлення може неправильно тлумачитися. Так, наприклад буде важко розрізнити штрихкоди для 100001 та для 1000001.

Розглянемо інші особливості побудови штрихових кодів, які також використовуються для класифікації штрихових кодів. Однією з таких особливостей є наявність чи відсутність контрольних штрихів(бітів). Вони використовуються в разі потреби стабілізації швидкості зчитування нашого коду від початку до кінця. В випадку відсутності контрольних штрихів, при нерівномірній швидкості зчитування штрих-коду, цей код можливо буде інтерпретовано неправильно. Щоб цьому запобігти, на початку та в кінці нашого коду розміщується набір з принаймні двох контрольних штрихів. Після зчитування ЕОМ цього коду, обчислювальна машина може судити про зміну швидкості сканування штрихового коду і відповідно корегувати процес декодування. Прикладом застосування контрольних штрихав може бути штриховий код типу EAN-13. В ньому контрольні штрихи наявні не тільки на початку та в кінці, а і в середині коду.

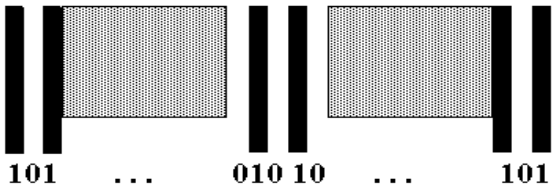

Рис. 3. **Штриховий код типу EAN-13**

Ще однією особливістю при побудові штрихового коду є наявність чи відсутність контрольної суми. Для гарантування правильності декодування штрихового коду в деяких типах штрихового коду до інформації, що кодіється додається деяка контрольна сума яка функціонально залежить від кодованої інформації. Ця контрольна сума кодується в штриховий код разом з основною інформацією, а при декодуванні ЕОМ знову вираховує контрольну суму цього коду і порівнює з заданим. Зрозуміло що в разі неспівпадання цих двох контрольних сум штриховий код був не правильно зіскановано.

Кожне повідомлення, кожна інформація про той або інший факті має немов би дві сторони: конкретний зміст даного повідомлення, даного факту, і статистичні (ймовірносні) властивості, що дозволяють порівнювати цілком разнорідні повідомлення по тій різноманітності станів, з якими ці повідомлення зв'язані.

Аналіз останніх досліджень і публікацій. Задача, яка буде описана тут, була сформульована під час введення на деякому великому підприємстві комп'ютеризованої системи пропусків [1]. Кожен

працівник мав свій 6-ти значний ідентифікаційний код, котрий разом з фотографією та іменем працівника був нанесений на пластикову картку-перепустку. Але для автоматизації контролю за переміщенням працівників необхідно було забезпечити можливість зчитувати та аналізувати інформацію з цих карток-перепусток і для ЕОМ. Таку можливість можна забезпечити наявністю на перепустці магнітної стрічки або ж штрихового коду. Магнітна стрічка в даній ситуації не є оптимальним вибором, в зв'язку з більшою вартістю як виготовлення такої картки так і її обслуговування. Головною перевагою магнітної картки є можливість збереження більшої інформації про власника, ніж при застосуванні штрихового коду, але в нашому випадку такої потреби немає - достатньо зберігати інформацію про ідентифікаційний код робітника, та можливо ще яку-небуть (рівень доступу, час роботи, і т.п.) [5].

Постановка проблеми. Таким чином оптимальним рішенням цієї задачі є застосування штрихового коду, який здатний кодувати 6-8 числових символів. Але використання якого-небуть із загальновживаних типів штрихових кодів виявилося ускладненим спицифікою використання карток-перепусток - через необхідність весь час носити їх на одязі, картки поступово забруднювалися, на їх поверхні з'являлися подряпини. Все це не впливало на можливість читання інформації людиною, але при використанні сканерів штрихових кодів кількість помилок з часом дуже швидко збільшувалась, виникала потреба в заміні картки-перепустки.

Всі ці проблеми в кінці-кінців привели до формулювання слідуючої постановки задачі: необхідно створити новий тип штрихового коду, основними вимогами до якого є простота, не вибагливість до скануючого пристрою, можливість кодування 6-8 числових символів та висока стійкість до помилок.

Самокорегуючий штриховий код НЕМ-8. Алфавітом штрих-коду назвемо множину символів з яких складається інформація закодована в цьому штриховому коді.

Бітовий ряд - це послідовність скінченого числа "0" та "1" (в графічному представленні - білих та чорних штрихів), яка задає один символ алфавіту штрихкоду.

Основні вимоги до штрихового коду:

Вимога 1. Мінімальна кількість символів, яку повинен містити штрихкод - 6, але можливо це число варто збільшити до 7 або 8 символів вводячи резервне або контрольне число.

Вимога 2. Алфавіт нашого штрихового коду складається тільки з 10 цифр 0,1, ...,9. Одже на кодування одного символа нам знадобиться min 4 біти $2^4=16$ ($2^3<10$). Насправді ця кількість в нас буде дещо більшою, за рахунок введення контрольних бітів та накладення деяких умов. Однією з таких умов буде наступна.

Вимога 3. Для запобігання злиття двох сусідніх бітових рядів, необхідно щоб кожен з таких рядів розпочиновся з 0 та закінчувався 1-цею.

Вимога 4. Велика кількість підряд йдучих однакових символів збільшує імовірніс помилки. Тому слід накласти ще одну умову: в бітовому ряді не повинно бути підряд біль ніж 3 нулів чи одиничок.

Вимога 5. Як зазначалося в вимозі 2, в наслідок виимог 3-4, на кодування одного символа потрібно 6 біт.

Наш штриховий код буде забезпечено властивістю самовиправлення помилок, яка буде базуватися на методі Хеммінга. Для цього нам необхідно вияснити яку кількість бітів ми відведемо під перевірочні. Виходячи з відомої формули $2^k>n+1$, де n - загальна кількість бітів, k - кількість провірочних бітів, m=n-k - кількість інформаційних бітів, отримуємо наступну таблицю:

Таблиця 1. *Кількість інформаційних бітів*

k	n	m=k-n
1	1	0
2	3	1
3	7	4
4	15	11
5	31	26
6	63	57
7	127	120
8	255	247

Мінімальне число бітів в бітовому ряді 6, якщо під інформаційні ми відведемо 26 біт, то ми зможемо кодувати 26/6=4 символи. Така кількість є недостатньою для нашої задачі, тому виділимо під інформаційні 57 бітів, так ми зможемо кодувати до 57/6=9 символів, що є навіть надлишково для нас.

В наступній таблиці записано всі можливі 6-ти значні бітові ряди, що задовольняють нашим умовам.

Таблиця 2.

1	000101
2	000111
3	001001
4	001011
5	001101
6	010001
7	010011
8	010101
9	010111
10	011001
11	011011
12	011101

Цієї кількості бітових рядів достатньо, щоб кожному символові нашого алфавіту штрихового коду поставити у відповідність один з них.

В разі виникнення однієї помилки в нашому штриховому коді, вона буде виправлена за допомогою алгоритму Хеммінга. В разі виникнення двох помилок метод Хеммінга перестає бути ефективним, в такій ситуації принаймні повинно бути виявлено наявність помилки. Ми забезпечуємо перевірку на наявність подвійних помилок, ввівши восьмий контрольний символ. Він функціонально залежить від інших символів, і ця залежність вибирається так щоб при зміні одного з перших семи символів, восьмий контрольний також обов'язково змінився. Якщо при виникненні двох помилок і вони знаходяться в одному бітовому ряді, то наявність такої помилки буде виявлено контрольним символом, але якщо ці помилки з'являться в різних бітових рядах, тобто будуть змінені два числа в нашому коді, то можливі такі комбінації символів коли помилка не буде виявлена взагалі. Для запобігання таких помилок набір бітових рядів потрібно підібрати так, щоб матриця кодових відстаней цього набору не містила одиниць. Тоді при виникненні двох помилок в різних бітових рядах, ці новоутворені бітові ряди не будуть належати нашому наборові, і таким чином буде виявлено наявність помилок.

Але тут виникає проблема: для того щоб вивести з матриці кодових відстаней одиниці, потрібно скоротити набір бінарних рядів, але тоді цього набору буде замало щоб поставити один бітовий ряд у відповідність одному симолу нашого алфавіту. Ми приходимо до потреби збільшення довжини наших бітових рядів з 6-ти до 7-ми символів.

57/7=8. Це також задовольняє нашим вимогам. Ми зможемо кодувати до восьми символів. В наступній таблиці записано всі можливі бітові ряди, що задовольняють нашим умовам. Нагадаємо, що алфавіт нашого штрихового коду складається з 10 символів, а оскільки допустимих бінарних рядів 22 то в нас тепер є змога розділивши їх на дві групи, кодувати один з наших символів не бітовим рядом, а належністю бітових рядів інших символів до тієї чи іншої групи.

Таблиця 3.

1	0001001		12	0100101
2	0001011		13	0100111
3	0001101		14	0101001
4	0010001		15	0101011
5	0010011		16	0101101
6	0010101		17	0110001
7	0010111		18	0110011
8	0011001		19	0110101
9	0011011		20	0110111
10	0011101		21	0111001
11	0100011		22	0111011

За рахунок цього можна або додати до нашого коду дев'ятий символ без збільшення кількості бітів в штриховому коді(тим самим зменшивши надлишковість нашого коду), або, залишивши наш код 8-ми символьним, зменшити кількість біт, що будуть кодувати цей код (тим самим також зменшивши надлишковість).

Але в нашій задачі основною вимогою до штрихового коду є не мінімізація надлишковості, а збільшення стійкості коду до помилок. Тому нам доведеться відмовитися від запропонованого вище варіанту розподілу бітових послідовностей на дві групи, з метою запобігання тих самих помилок, що були описані вище.

Виберемо з 22-х бітових рядів (таб. 3) десять таких, щоб їх матриця кодових відстаней не містила одиниць.

Таблиця 4.

0	0001001
1	0010001
2	0010111
3	0011011
4	0011101
5	0100111
6	0101011
7	0101101
8	0110011
9	0110101

Тоді матриця кодових відстаней буде наступна :

Таблиця 5. *Матриця кодових відстаней*

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0	2	4	2	2	4	2	2	4	4
1	2	0	2	2	2	4	4	4	2	2
2	4	2	0	2	2	2	4	4	2	2
3	2	2	2	0	2	4	2	4	2	4
4	2	2	2	2	0	4	4	2	4	2
5	4	4	2	4	4	0	2	2	2	2
6	2	4	4	2	4	2	0	2	2	4
7	2	4	4	4	2	2	2	0	4	2
8	4	2	2	2	4	2	2	4	0	2
9	4	2	2	4	2	2	4	2	2	0

Наш штриховий код буде мати слідуючу структуру:

0xxxxx10xxxxx10xxxxx10xxxxx10xxxxx10xxxxx10xxxxx10xxxxx1nnnnnn

Вона містить один невеликий недолік, останні шість бітів можуть неправильно зчитуватися. Наприклад: а) вище ми накладали обмеження на кількість однакових підряд йдучих символів "0" чи "1" , на комбінацію nnnnnn в данному випадку таке обмеження не поширюється; б) в разі, якщо остнній байт чи байти будуть рівні "1", то при графічному представленні цього коду вони будуть втрачені, оскільки зіллються з "тихою зоною". Щоб запабігти цим небажанним явищам, введемо додаткові нульові біти в кінець нашого коду. Отримаємо код наступної структури:

0xxxxx10xxxxx10xxxxx10xxxxx10xxxxx10xxxxx10xxxxx10xxxxx101nn01nn01nn010

В цьому коді всі вище перераховані недоліки усунуто.

Потрібно також для нашого коду вибрати алгоритм знаходження контрольного числа. Запозичимо алгоритм з коду типу EAN. Тоді вагами будуть "3 1 3 1 3 1 3", а контрольне число буде рівне mod10 від зваженої суми.

Таким чином створивши штриховий код(назвемо його НЕМ-8), було розвязано поставлену перед нами задачу. Нам залишилося тільки перевірити єфективність цього коду. Для початку підрахуємо ентропію нашого коду. Не важко пересвідчитися що НЕМ-8 складається з 71 біта, кількість комбінацій чисел які можна утворити 10 мільйонів (10^7), оскільки восьме число залежить тільки від перших семи, ми його не враховуємо. Кількість інформації шо несе в собі штрихкод НЕМ-8 рівна $I(a)=\log(10^7)$. Вище була наведена формула для знаходження величини надлишковості коду, використаємо її тепер.

$$r = 1 - \frac{I(a)}{n \cdot \log L} \tag{1}$$

в нашому випадку алфавіт двозначний L=2, кількість бітів n=71 тому

$$r = 1 - \frac{\log 10^7}{71 \cdot \log 2} = 0.67 \tag{2}$$

Як бачимо коефіцієнт надлишковості досить великий, для порівняння надлишковість коду EAN-13 рівна

$$r = 1 - \frac{\log 10^{12}}{95 \cdot \log 2} = 0.58 \tag{3}$$

що також не мало. Але завдяки цій надлишковості ці коди стають більш стійкими до помилок, що для нас більш важливо ніж надмірність інформації. Стійкість до помилок нам ще необхідно буде перевірити. Для цього необхідно провести статистичні дослідження по виправленню та виявленню помилок. Для цих досліджень необхідно створити прикладну програму, яка б генерувала для заданого коду певну кількість помилок і намагалася б їх виявити та виправити.

1. Жураковський Ю.П., Полторак В.П. Теорія інформації та кодування: Підручник. — К.: Вища школа, 2011. — 255 с.
2. Берлекемп Э. Алгебраическая теория кодирования. — М., Мир, 1971 — 480 с.
3. Добрушина Р.Л.,Лупанова О.Б. -М.: Издательство иностранной литературы, 1963. 830 с.
4. Котов П.А. Повышение достоверности передачи цифровой информации. -М.: Связь, 1966. 184 с.
5. Словарь по кибернетике / Справочное издание под редакцией Михалевича В.С. -К.: Главная редакция Украинской Советской Энциклопедии имени М.П.Бажана, 1989. 752 с.
6. Шеннон К. Работы по теории информации и кибернетике / Пер. с англ. под ред. Котов П.А. Повышение достоверности передачи цифровой информации. -М.: Связь, 1966. 184 с.
7. Шульгин В.И. Основы теории передачи информации, Харьков, 2002. — 160 с.