
ОСНОВИ ТЕОРІЇ
ІНФОРМАЦІЇ
ТА КОДУВАННЯ

Міністерство освіти і науки України
Львівський національний університет імені Івана Франка

Основи теорії інформації та кодування

Навчальний посібник

Львів
2023

УДК 519.7:004(075.8)
О 75

Автори:
І. А. Прокопишин, Р. Є. Рикалюк, В. Ф. Чекурін, К. А. Червінка

Рецензенти:
д-р техн. наук, проф. В. Б. Дудикевич
(Національний університет "Львівська політехніка", м. Львів);
д-р фіз.-мат. наук, проф. Т. С. Нагірний
(Інститут механічної інженерії, Зеленогурський Університет,
м. Зелена Гура, Польща);
д-р техн. наук, проф. Р. М. Пасічник
(Західноукраїнський національний університет, м. Тернопіль)

Рекомендовано до друку Вченою радою
Львівського національного університету імені Івана Франка
(протокол № 49/6 від 29.06.2023 року)

О 75 **Основи теорії інформації та кодування** : навч. посібник / [І. А. Прокопишин, Р. Є. Рикалюк, В. Ф. Чекурін, К. А. Червінка]. – Електрон. вид. – Львів : ЛНУ ім. Івана Франка, 2023. – 156 с.

ISBN 978-617-10-0820-5 (електрон. вид.)

Розглянуто головні поняття теорії інформації, ефективного кодування даних, моделювання та кодування графічних зображень, звуку та відео, кодування з захистом від завад та шифрування, а також теоретичні основи побудови електронних схем комп’ютерів.

Для студентів математичних та інших спеціальностей, які вивчають комп’ютерні науки.

УДК 519.7:004(075.8)

ISBN 978-617-10-0820-5 (електрон. вид.)

ЗМІСТ

СПИСОК АБРЕВІАТУР

АІС – автоматизована інформаційна система
АЦП – аналого-цифровий перетворювач
ДКП – дискретне косинус-перетворення
ЕОМ – електронно-обчислювальна машина
ЕПТ – електронно-променева трубка
ІС – інформаційна система
МК – матриця квантування
МКО – міжнародна комісія з освітленості
НСД – найбільший спільний дільник
ОП – оперативна пам’ять
ОС – операційна система
ТВЧ – телебачення високої чіткості
ЦАП – цифро-аналоговий перетворювач
AAC – *Advanced Audio Coding*
AC-3 – *family of Audio Compression technologies developed by Dolby Laboratories*
AES – *Advanced Encryption Standard*
ANSI – *American National Standard Institute*
ASCII – *American Standard Code for Information Interchange*
AT&T – *American Telephone and Telegraph*
AVC – *Advanced Video Coding – video compression standard*
BD – *Blu-ray Disc*
BMP – *Bitmap Picture – bitmap image storage format developed by Microsoft*
CCITT – *Comité Consultatif International Téléphonique et Télégraphique*
CD – *Compact Disc*
CD-DA – *Compact Disc Digital Audio – standard for recording sound on CD*
CD-R – *Compact Disc-Recordable*
CD-RW – *Compact Disc-ReWritable*
CIE – *Commission Internationale de l´Eclairage*
CIELAB – *color space developed by CIE in 1976 based on CIE XYZ*
CIE XYZ – *color space created by the CIE in 1931*
CIF – *Common Interchange Format – video standard with scan of 352x288 pixels*
CMM – *Color Matching Module*

CMS – *Color Management System*

CMY – *subtractive color model, in which the main colors are Cyan, Magenta and Yellow*

CMYK – *subtractive color model, in which the main colors are Cyan, Magenta, Yellow and blacK*

DES – *Data Encryption Standard – symmetric block encryption algorithm*

DjVu – *lossy image compression technology and the corresponding format*

DVD – *Digital Video Disc*

ENIAC – *Electronic Numerical Integrator and Computer*

EPS – *Encapsulated PostScript – simplified version of PostScript*

FPX – *FlashPix – bitmapped computer graphics file format*

GIF – *Graphics Interchange Format – 8-bit bitmap graphics format*

HDTV – *High-Definition Television – high-definition television standard with scan of 1920x1080 pixels*

Full HD – *see HDTV*

HSB – *color intuitive model based on three color characteristics: Hue, Saturation, and Brightness*

HSV – *see HSB*

ICC – *International Color Consortium*

ISO – *International Organization for Standardization*

JPEG – *name of full-color image compression algorithm and corresponding format, developed by the Joint Photographic Expert Group*

LZ – *prefix for family of data compression algorithms (LZ77, LZ78, LZW etc.) – the initials of Abraham Lempel and Jacob Ziv*

MD – *prefix for family of Message-Digest Algorithms (MD1–MD5 etc.)*

MIDI – *Musical Instrument Digital Interface*

MP3 – *coding forma for digital audio*

MPEG – *Moving Pictures Experts Group – line of digital television standards*

NTSC – *National Television System Committee – standard for analog television broadcast*

PAL – *Phase Alternating Line – color encoding system for analogue television*

PCM – *Pulse Code Modulation*

PDF – *Portable Document Format*

PGP – *Pretty Good Privacy – encryption program*

PKE – *Public Key Encryption*

PNG – *Portable Networks Graphic – raster-graphics file format*

PS – *PostScript – page description language and the corresponding format*

QCIF – *Quartet Common Interchange Format – video standard with scan of 176x144 pixel*

RA – *Real Audio – audio data format that uses several audio codecs*

RGB – *additive color model, in which the main colors are Red, Green and Blue*

RIFF – *Resource Interchange File Format – standard for storing any data structures*

RLE – *Run Length Encoding*

RSA – *Rivest–Shamir–Adleman – public-key cryptosystem developed by Ron Rivest, Adi Shamir and Leonard Adleman*

SECAM – *Séquentiel de couleur à mémoire – analog color television system*

SHA – *prefix for family of Secure Hash Algorithm standards, SHA-1– SHA-3*

SI – *Système International d'Unités*

TIFF – *Tagged Image File Format – format for storing raster graphic images*

UHDTV – *Ultra High Definition Television – digital ultra high definition television standards*

VOC – *Voice File – 8-bit audio format*

WAV – *Wave Form Audio File – 16-bit audio format developed by Microsoft*

WCS – *Windows Color System*

WMA – *Widows Media Audio – series of audio codecs and their corresponding audio coding formats developed by Microsoft*

WMF – *Windows Meta File – universal vector graphics file format for Windows applications*

XOR – *eXclusive OR – logical and bitwise operation modulo two addition*

XYZ – *see CIE XYZ*

YCbCr – *color model that uses a luminance component and two color difference components*

YUV – *color model that uses a luminance component and two color difference components*

ПЕРЕДМОВА

Сьогодні важливим чинником у розвитку суспільства є інформаційні технології, що проникають у всі сфери людської діяльності – в державне управління, економіку, науку, освіту, культуру та побут.

Отож доволі строгий, проте доступний виклад основ теорії інформації та кодування, як одних з фундаментальних галузей комп'ютерних наук, є своєчасним і потрібним для підготовки спеціалістів.

Посібник містить дванадцять розділів.

Перший розділ, вступний, окреслює предмет комп'ютерних наук. У ньому означено поняття інформаційної системи та інформаційної технології.

У другому та третьому розділах викладено головні поняття теорії інформації та питання кількісної оцінки інформації дискретних ймовірнісних джерел. Зокрема, розглянуто математичні моделі сигналів, перетворення аналогових сигналів у цифрові, теорему про відліки, способи кодування тексту та чисел у комп'ютерах, ентропію простого та стаціонарного джерела інформації.

У четвертому та п'ятому розділах викладено основи теорії ефективного кодування та її застосування до стиснення даних. Розглянуто побудову префіксних кодів за допомогою бінарних дерев, теорему Шеннона для стаціонарних джерел, методи ефективного кодування Шеннона–Фано, Гаффмана та арифметичного кодування, а також приклади їхнього використання. Описано методи стиснення даних без втрати інформації, зокрема, словникові методи стиснення і метод Лемпела–Зіва–Велча.

У розділах 6, 7 та 8 розглянуто основи моделювання та представлення у комп'ютерах графічних, звукових та відеоданих. Викладено основи колориметрії, колірні моделі та кодування кольорів. Розглянуто базові графічні моделі та формати, методи кодування звуку та відеозображень, а також методи їхнього стиснення.

У розділі 9 вивчатимемо проблеми завадостійкого кодування. На строгому математичному рівні розглянуто загальні питання побудови лінійних систематичних кодів, зокрема, коду Геммінґа.

У десятому та одинадцятому розділах викладено основи криптографії. Детально описано симетричні криптосистеми та криптосистеми з відкритим ключем. Розглянуто стандарти шифрування даних, гешувальні функції, цифровий підпис, викладено алгоритм *RSA* та його надійність.

У дванадцятому розділі розглянуто теоретичні основи побудови електронних схем комп'ютерів, які ґрунтуються на апараті математичної логіки.

Для розуміння головних теоретичних положень, викладених у посібнику, достатньо знань з курсу вищої математики для ЗВО. Посібник містить значну кількість прикладів, які ілюструють теоретичний матеріал. Наприкінці кожного розділу подано запитання для самоконтролю.

Під час підготовки посібника автори використали усі джерела, наведені у списку літератури.

Автори висловлюють глибоку вдячність професорам В. Б. Дудикевичу, Т. С. Нагірному та Р. М. Пасічнику за зауваження та пропозиції, надані ними під час рецензування посібника.

Розділ 1. ПРЕДМЕТ КОМП'ЮТЕРНИХ НАУК

Ми всі є свідками бурхливого розвитку комп'ютерної техніки та щораз ширшого її використання у найрізноманітніших галузях людського життя. Комп'ютери, засоби телекомунікації, інтернет проникають у всі сфери людської діяльності – в економіку, науку, освіту та культуру. Інформаційні процеси стають вирішальним чинником у розвитку суспільства. Отож знання комп'ютерних наук і вміння ефективно використовувати обчислювальну техніку у своїй практичній діяльності – одна з найнеобхідніших професійних якостей сучасного фахівця з вищою освітою.

1.1. Інформатика. Кібернетика. Комп'ютерні науки

Інформатика – це наука, яка вивчає методи збору, нагромадження, зберігання, передавання, обробки, пошуку та відображення інформації з використанням засобів обчислювальної техніки, а також принципи функціонування цих засобів.

Інформатика – термін французького походження, що виник від поєднання двох понять: "інформація" та "автоматика". Він виражає сутність цієї дисципліни як науки про автоматизовану обробку інформації. Термін цей поширений здебільшого у Франції, країнах Східної Європи та колишнього СРСР. Натомість у США та більшості країн Західної Європи у близькому значенні вживають термін *комп'ютерні науки* (*Computer Science*), який позначає комплекс наук, пов'язаних з функціонуванням та застосуванням засобів обчислювальної (комп'ютерної) техніки [16]. Слово *комп'ютер* (з англ. буквально – *обчислювач*) походить від назви першої цифрової електронної обчислювальної машини – *Electronic Numerical Integrator and Computer* (*ENIAC*), створеної під керівництвом Дж. Еккерта та Дж. Моклі (*J. P. Eckert, J. W. Mauchly*) 1945 року в Пенсільванському університеті США [18]. У радянській літературі використовували назву – *електронна обчислювальна машина* (ЕОМ).

Як галузь знань інформатика сформувалася на початку 40-х років XX ст. як теоретична основа функціонування обчислювальних машин і була поєднанням математичної логіки та теорії алгоритмів. Нині теоретична інформатика тісно пов'язана з математикою, значною мірою впливає на розвиток різних її розділів і суттєво розширює сферу її застосувань.

Зміст інформатики як наукової дисципліни поступово змінюється. Наприклад, у 70-х роках XX ст. інформатикою називали "наукову дисципліну, яка вивчає структуру і загальні властивості наукової інформації, а також закономірності всіх процесів наукової комунікації" [4].

У цьому зв'язку варто згадати також про кібернетику. Французький фізик і математик А.-М. Ампер (*A.-M. Ampère*) 1834 року запропонував називати кібернетикою науку про управління людським суспільством, а американський математик Н. Вінер (*N. Wiener*) 1948 року назвав кібернетику наукою "про управління та зв'язок у живому організмі та людині". На початку 70-х років кібернетика остаточно сформувалася як наука фізико-математичного профілю з власним предметом дослідження – кібернетичною системою – складною системою, яку вивчають, застосовуючи методи комп'ютерних наук [4]. Сьогодні предметом її вивчення є принципи побудови і функціонування систем автоматичного керування, а головним завданням – розробка методів прийняття рішень технічними системами.

Аналізуючи літературні джерела (зокрема, [16]), можна виділити такі основні напрями комп'ютерних наук:

1. ***Теорія інформації*** – розділ присвячений математичному описові та аналізу методів кодування, передавання та збереження інформації. Широко використовує методи алгебри, теорії ймовірності, математичної статистики та інших математичних дисциплін.
2. ***Алгоритми та структури даних.*** Цей напрям охоплює декілька розділів: теорію алгоритмів, теорію обчислювальної складності, теорію паралельних обчислень, теорію структур даних, криптографію та ін.
3. ***Мови програмування.*** Розділ комп'ютерних наук, метою якого є розроблення та дослідження систем позначень, методів і засобів для запису у текстовому вигляді алгоритмів та даних з метою їхнього виконання на комп'ютерах. Подані у такому вигляді алгоритми називають програмами.
4. ***Архітектура комп'ютерів.*** Цей розділ комп'ютерних наук вивчає загальні принципи побудови та функціонування комп'ютерів. Ґрунтується на таких науках, як булева алгебра, цифрова логіка, теорія скінченних автоматів, теорія кодування, теорія систем та ін.

5. ***Операційні системи та комп'ютерні мережі.*** Вивчає методи ефективного управління ресурсами окремих комп'ютерів і комп'ютерних мереж та організації їхньої взаємодії з людиною.
6. ***Інженерія програмного забезпечення.*** Вивчає принципи та методи створення й супроводу великих програмних систем, що відповідають заданим системним вимогам та вимогам користувачів.
7. ***Бази даних та інформаційні пошукові системи.*** Розділ, у якому вивчають і розробляють методи та засоби створення комп'ютерних систем, призначених для нагромадження й зберігання великих масивів інформації, а також її пошуку і модифікації за запитами користувачів. Важливою вимогою до таких систем є забезпечення можливості їхньої роботи в режимі колективного доступу до даних. Ґрунтується на реляційному численні, теорії графів, методах паралельної обробки даних тощо.
8. ***Штучний інтелект та робототехніка.*** Цей розділ комп'ютерних наук вивчає моделювання процесів пізнання світу людиною з метою створення компонентів машин, які здатні імітувати її діяльність. Охоплює такі напрями: розпізнавання образів, машинне навчання, методи прийняття рішень, експертні системи, інженерія знань та ін.
9. ***Комп'ютерна графіка.*** Розробляє методи моделювання та візуального відтворення реальних і віртуальних об'єктів та імітації їхнього руху. Використовує методи лінійної алгебри, обчислювальну геометрію та інші математичні науки.
10. ***Обчислювальна математика*** – комплексна наука, що досліджує та розробляє обчислювальні методи. Методи обчислень – сукупність математичних методів, які зводять розв'язування математичної задачі до арифметичних та логічних дій над числами скінченної розрядності.
11. ***Ділова інформатика.*** Вивчає питання обміну інформацією і створення програмних систем, що забезпечують роботу установ і координацію дій їхніх співробітників.
12. ***Біоінформатика.*** Займається дослідженням інформаційних процесів у живих організмах.
13. ***Кібербезпека.*** Вивчає методи захисту інформації та підтримуючої інфраструктури у кіберпросторі – віртуальному просторі телекомунікаційних систем та інформаційних мереж.
14. ***Взаємодія людини і комп'ютера.*** Ця галузь комп'ютерних наук вивчає питання ефективного обміну інформацією між людьми та обчислювальними системами (інтерфейс користувача).

1.2. Інформаційні технології та інформаційні системи

Одним з головних практичних завдань комп’ютерних наук є розробка ефективних методів, алгоритмів та програм обробки інформації, тобто розробка *інформаційних технологій*.

Інформаційна система (ІС) – це система, що реалізує інформаційні технології; призначена для збору, нагромадження, зберігання, передавання, обробки, пошуку та відображення інформації.

Однією з найпростіших інформаційних систем можна вважати книгу (у взаємодії з читачем), оскільки вона систематизовано зберігає текстову інформацію та має засоби для її пошуку: зміст, алфавітний (предметний) покажчик, список позначень, список літератури та ін. Складними інформаційними системами є бібліотеки, облікові системи установ, організацій, підприємств (кадри, бухгалтерія), інтернет тощо. Типова структура *автоматизованої інформаційної системи* (АІС) наведена на рис. 1.1.

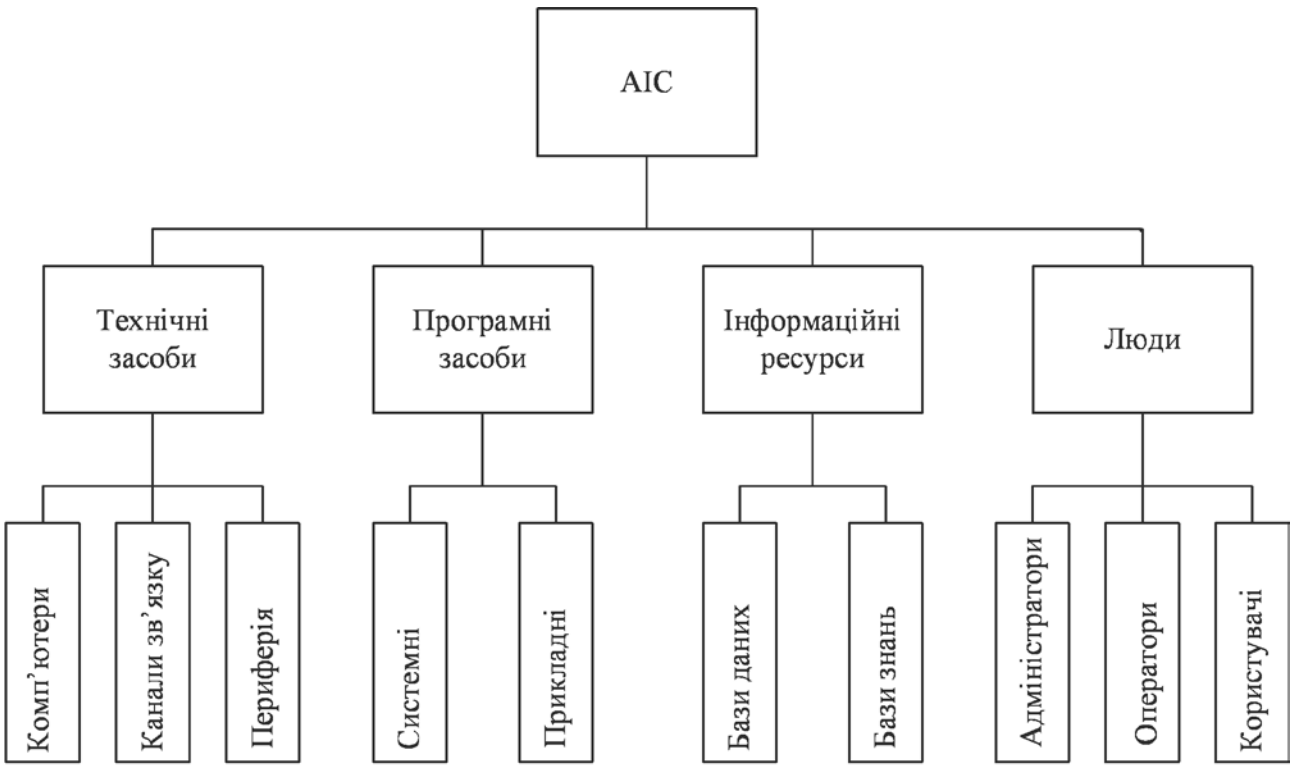


Рис. 1.1. Структура автоматизованої інформаційної системи

Отже, автоматизована інформаційна система – це комплекс технічних, програмних засобів та інформаційних ресурсів, які функціонують під управлінням людини: призначена для збору, нагромадження, зберігання, передавання, обробки та відображення інформації.

Як приклад інформаційної системи можна розглядати, зокрема, будь-який програмно-технічний комплекс, призначений для спеціалізованої обробки інформації – бухгалтерський облік на підприємстві, облік кадрів, матеріальних цінностей, електронні каталоги бібліотек, електронні пошукові системи і т. д. Така система має, зазвичай, чотири головні складові: апаратні та системні засоби, базу даних, блок аналізу, а також інтерфейс користувача.

Запитання та завдання

1. Охарактеризуйте зміст і генезис комп’ютерних наук.
2. Що таке інформатика?
3. Що вивчає кібернетика?
4. Розтлумачте етимологію терміна "комп’ютер".
5. Назвіть основні розділи комп’ютерних наук.
6. Що є предметом теорії інформації?
7. Що вивчає обчислювальна математика?
8. Яка роль математики у комп’ютерних науках?
9. Опишіть структуру автоматизованої інформаційної системи.
10. Наведіть приклади інформаційних систем.

Розділ 2. БАЗОВІ ПОНЯТТЯ ТЕОРІЇ ІНФОРМАЦІЇ

2.1. Інформація

Інформація – фундаментальне поняття філософії, визначається як сукупність знань про об'єкт (явище) і фактичних даних щодо нього та його зв'язків з іншими об'єктами та явищами. Інформація, матерія та енергія утворюють три основні ресурси, які використовує людина у своїй життєдіяльності.

Термін походить від латинського слова *informatio – роз'яснення, виклад, обізнаність*.

У суб'єктивному трактуванні інформація – це індивідуальна чи групова інтерпретація отриманої послідовності сигналів (наприклад, звукових чи оптичних).

Когнітивно-системний[1] підхід розрізняє чотири головні елементи природних і штучних процесів мислення: дані, інформацію, знання та преференції. В інтерпретації цього підходу людське знання перетворює інформацію, внаслідок чого виникає інша інформація чи нове знання. Залежно від індивідуальної концептуальної системи одна і та ж послідовність сигналів (знаків) може бути джерелом різної інформації для різних осіб чи роботів. Якщо ж група людей чи суспільство має в деякій області спільну концептуальну систему (наприклад, теорію, сукупності понять, поглядів, означень, правил тощо), то різні індивідууми однаково сприймають та інтерпретують однакові сигнали.

В інформаційно-логічному розумінні інформація – це зміст повідомлення, переданого за допомогою даних.

Одну і ту ж інформацію (зміст) можна передати, використовуючи різні дані – мову, знаки, креслення, формули тощо. На цій підставі можна говорити про різні види інформації – звукову, вербальну, текстову, графічну, цифрову.

---

[1] Когнітивний (від лат. *cognitio – знання, пізнання*) – пізнавальний, який відповідає пізнанню.

У комп'ютерних науках вирізняють технічний (синтаксичний), семантичний та прагматичний аспекти інформаційного процесу. Технічний аспект пов'язаний з питаннями кодування даних, швидкістю та надійністю їхнього передавання та накопичення на носіях тощо. Семантичний аспект полягає у тому, як передати зміст за допомогою кодів. Прагматичний аспект визначає практичну цінність для адресата отриманої інформації, а, отже, і його реакцію на неї.

Розглянемо простий приклад. Науковець отримав електронною поштою архівований файл conf.zip. Розархівація файла, встановлення кодової таблиці, відтворення тексту на екрані чи папері з використанням текстового редактора – це технічні аспекти інформаційного процесу. Переклад англомовного тексту українською мовою визначає у цьому прикладі семантичний аспект. Натомість усвідомлення змісту отриманої інформації та аналіз конкретних даних, які містить лист (наприклад, термін подання заявки на участь, об'єм та форма матеріалів, які треба подати, або ж сума оргвнеску, яку треба перерахувати в оргкомітет тощо) складають прагматичний аспект інформаційного обміну. Реакція адресата на отримане повідомлення залежить від прагматичного аспекту інформації, яку воно містить. Наприклад, одержувач може негайно розпочати підготовку заявки на участь у конференції, зберегти лист та повернутися до нього згодом або ж видалити його з комп'ютера (якщо надіслане повідомлення не має для нього інтересу).

Розділ комп'ютерних наук, який називають теорією інформації, вивчає передусім технічні проблеми інформаційних процесів. Семантичні проблеми, натомість, вивчає математична лінгвістика.

Теорія інформації – розділ комп'ютерних наук, який займається математичним описом і аналізом методів кодування, передавання та збереження інформації. Широко використовує методи алгебри, теорії ймовірності, математичної статистики та інших математичних дисциплін. Важлива складова теорії інформації – теорія передавання інформації (іноді ці терміни вживають як синоніми).

Основи теорії інформації закладені американським електротехніком і математиком Клодом Шенноном (*Claude Shannon*) у його знаменитій праці "Математична теорія зв'язку" [33]. Сучасні аспекти теорії інформації та кодування розглянуто у працях [6; 13;17; 19; 22; 25; 31].

2.2. Повідомлення, сигнали, дані. Загальна схема передавання інформації

Передавання інформації передбачає наявність двох об'єктів: *джерела інформації* (ДІ) та споживача – *адресата* (А). Первинними джерелами інформації є явища та об'єкти навколишнього світу, зокрема, живі організми, люди та технічні пристрої.

Водночас первинним споживачем інформації також можуть бути живі організми, люди та технічні пристрої. У найпростішому випадку інформація, що надійшла від джерела інформації до адресата, впливає певним чином на його поведінку та далі не передається. Загалом інформацію можуть передати іншому адресату, перетворити та використати, спонукаючи до певних дій. Іноді важко з'ясувати первинне джерело інформації.

Потрібно зауважити, що первинне повідомлення джерела – це продукт взаємодії джерела та спостерігача, який фільтрує певні ознаки явища і подає їх у деякій стандартній формі (мімікою та жестами, у звуковій формі – за допомогою мови чи спеціальних сигналів, або фіксує на носії шляхом зміни його властивостей). Отже, джерело інформації загалом – це явище зовнішнього світу та спостерігач, який породжує повідомлення у деякій стандартній формі.

Якщо повідомлення джерела є неперервним фізичним процесом, то таке джерело називають *неперервним джерелом інформації.* Якщо ж повідомлення джерела можна подати за допомогою літер деякого алфавіту, то йдеться про *дискретну модель джерела*.

Для реалізації технічної можливості передачі та обробки повідомлення його накладають на деякий фізичний процес хвильової природи, який після цього утворює *сигнал*. Розрізняють звукові, електричні, електромагнітні, оптичні та інші сигнали.

Дані – це зареєстровані повідомлення чи сигнали (механічна зміна форми об'єкта, якості його поверхні; зміна оптичних, електричних, магнітних характеристик або ж хімічного складу; зміна стану електричної чи іншої системи). На технічному рівні *дані* є синонімом терміна *інформація*.

Зміст інформації відмінний від самого факту надходження даних. Для отримання власне інформації до даних застосовують адекватні методи. Наприклад, під час перекладу іноземного тексту такими методами є правила граматики та словники.

Інформація надходить від джерела до адресата завдяки деяким матеріальним об’єктам – носіям інформації. Найчастіше це електричні або оптичні *сигнали*, які передають *каналами зв’язку*.

У цифрових каналах зв’язку перед передаванням інформації здійснюють її перетворення у послідовність двійкових цифр. Отримані у такий спосіб числові масиви (дані) накладають на носій шляхом його модуляції та отримують сигнал. При модуляції змінюється певний параметр фізичного процесу (носія даних), наприклад, амплітуда, частота чи фаза змінного електричного струму, амплітуда, тривалість чи частота послідовності імпульсів електричної напруги або світлових імпульсів тощо.

На виході каналу зв’язку здійснюють зворотні перетворення сигналу й отримують дані, дешифрування яких дає інформацію. Оскільки під час передавання та перетворення сигналів на них накладаються завади – зовнішні фізичні поля тієї ж природи, що й носій даних, то дані, що надходять до адресата, можуть дещо відрізнятися від тих, що вийшли з джерела інформації.

Спрощену функціональну схему цифрового зв’язку подано на рис. 2.1.

Рис. 2.1. Загальна схема передавання інформації: ДІ – джерело інформації; К – кодер; М – модулятор; ДМ – демодулятор; ДК – декодер; А – адресат

Детальніше з теорією та практичними аспектами цифрового зв’язку можна ознайомитися за фундаментальними підручниками [29; 32].

2.3. Математичні моделі сигналів

Аналогові та цифрові сигнали. Первинне повідомлення джерела інформації – це деякий фізичний процес, що розвивається у просторі та часі. Наприклад, для звукової інформації – це коливання тиску повітря, які утворюють звукові хвилі, а для візуальної – світлові хвилі, відбиті поверхнею об’єкта.

Використовуючи спеціальні технічні пристрої (мікрофони, відеокамери тощо), інформативні параметри цих процесів (коливання тиску звукової хвилі, інтенсивність та довжину світлової хвилі) перетворюють в електричні сигнали, які є неперервними функціями часу $U(t)$. Такі сигнали називають *аналоговими*.

Для передавання вихідного низькочастотного аналогового сигналу каналами зв'язку його накладають на високочастотний сигнал-носій.

Аналогові телекомунікаційні пристрої історично виникли першими і набули значного поширення. Аналоговими є класичні телефонний та радіозв'язок, телебачення; створені аналогові обчислювальні машини розв'язували деякі класи математичних задач.

Однак з часом виявилось, що використання цифрових (дискретних) сигналів для передавання та обробки даних є значно ефективнішим. Сьогодні ми остаточно увійшли в еру цифрових технологій. Нас оточують цифровий телефонний зв'язок, цифрова аудіо- та відеоапаратура, цифрові комп'ютери тощо.

Головні переваги цифрової форми представлення сигналів такі:

1) цифрові сигнали є стійкішими до випадкових помилок, зумовлених завадами, оскільки: по-перше, імпульси можна відновлювати у проміжних пунктах; по-друге, можна застосувати коди, які виявляють і виправляють помилки;

2) цифрові сигнали можна обробляти за допомогою цифрових обчислювальних пристроїв, використовуючи найскладніші алгоритми;

3) цифрова апаратура значно надійніша від аналогової і дешевша за масового виробництва.

Принцип перетворення аналогових сигналів у цифрові. За аналого-цифрового перетворення відбувається заміна аналогового сигналу $U(t)$, $t \in [0,T]$ послідовністю цілих чисел з множини $M = \{0,1,\ldots,2^n - 1\}$, де параметр n визначає розрядність двійкового цифрового коду.

Цей процес здійснюють у два етапи.

Перший етап – *дискретизація сигналу* (*sampling*), на якому формують дискретну послідовність значень $U_i = U(t_i)$ неперервного сигналу $U(t)$ у фіксовані моменти часу t_i, $i = 0,1,\ldots N_S$. Моменти часу t_i (*точки відліку*) найчастіше обирають на проміжку $[0,T]$ рівномірно з кроком $\Delta t = T / N_S$.

Значення сигналу U_i у точках відліку t_i називають *відліками*. Величину Δt називають *кроком дискретизації* або кроком квантування за часом. *Частота дискретизації* $f_s = 1/\Delta t$ (*sampling rate*) – величина, обернена до кроку дискретизації – показує частоту здійснення відбору значення сигналу $U(t)$ за одиницю часу. Згідно з теоремою про відліки (її подамо нижче), для коректного відтворення аналогового сигналу частота дискретизації повинна бути щонайменше вдвічі більшою від максимальної частоти у спектрі аналогового сигналу.

На другому етапі відбувається *квантування сигналу* (*quantizing*). Для цього діапазон зміни рівня сигналу $[U_1, U_2] \supseteq [U_{\min}, U_{\max}]$ покривають дискретною сіткою дійсних чисел $V = \{V_j, j = 0, 1, \ldots, N_q\}$, найчастіше – рівномірно з кроком $\Delta = (U_2 - U_1)/N_q$. Далі кожен член U_i виділеної з сигналу числової послідовності $\{U_i\}$ заміняють найближчим до нього значенням $V_i \in V$:

$$U_i \to V_i,\ i = 0, 1, \ldots, N_q\ . \tag{2.1}$$

Параметр Δ називають *кроком квантування*, а множину V – *шкалою квантування*. Кількість рівнів квантування N_q визначається максимальною розрядністю двійкових чисел, які застосовують для цифрового кодування відліків U_i. Зокрема, якщо застосовувати кодування двійковими числами довжини n, то $N_q = 2^n - 1$.

Існує взаємно однозначна відповідність між рівнями квантування V_j та цілими числами m_j з послідовності $M = \{0, 1, \ldots, 2^n - 1\}$:

$$V_j = V_0 + m_j \cdot \Delta\,. \tag{2.2}$$

Отже, у результаті встановлюється взаємно однозначна відповідність між відліками U_i та цілими числами $m_i \in M$:

$$U_i \approx V_i = V_0 + m_i \cdot \Delta\,, \tag{2.3}$$

де для кожного відліку U_i число $m_i \in M$ визначають, наприклад, з умови:

$$m_i = \operatorname{argmax}\left\{ \min_{m_j \in M} \left| U_i - V_j \right| \right\}. \tag{2.4}$$

Похибка, яка виникає за аналого-цифрового перетворення, пропорційна до величин кроків квантування та дискретизації: завдяки їхньому зменшенню можна підвищити точність перетворення. Однак у цьому випадку зростатиме об'єм даних. Не доцільно здійснювати квантування з кроком, значно меншим від рівня можливих шумів у вихідному сигналі. До того ж, роздільна здатність органів чуття людини та технічних сенсорів обмежена. Ці чинники є визначальними для встановлення достатньої кількості рівнів квантування та кроку дискретизації щодо конкретних класів сигналів.

Наприклад, цифрова телефонія використовує частоту дискретизації 8 кГц, звуковий стандарт запису на компакт-диски (*CD-DA*) – 44,1 кГц, а цифрові магнітофони – до 48 кГц. Шкала квантування найчастіше є 8-ми, 16-ти або 32-бітною.

Процес перетворення аналогового сигналу в цифровий ілюструє рис. 2.2. У результаті аналоговий сигнал представляють послідовністю цілих чисел {3, 4, 4, 4, 4, 2, 0}, або в двійковій системі числення – {011, 100, 100, 100, 100, 010, 000}.

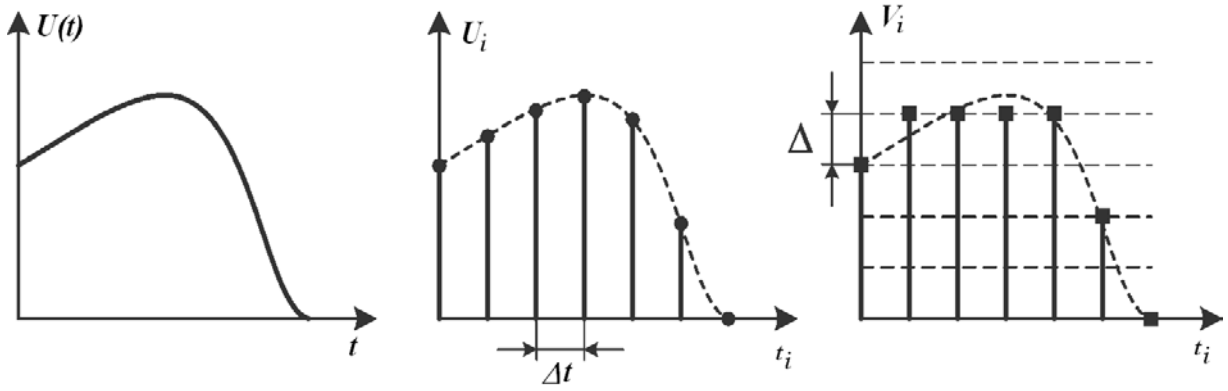


Рис. 2.2. Схема перетворення аналогового сигналу у цифровий

Дискретизацію та квантування здійснюють спеціальні електронні пристрої – *аналого-цифрові перетворювачі* **(**АЦП**)**, у зворотному напрямку – *цифро-аналогові перетворювачі* (ЦАП**)**.

Теорема про відліки. Теорема про відліки є важливою для визначення частоти дискретизації аналогового сигналу, достатньої для правильного його відтворення за дискретним сигналом. Її сформулював і довів 1915 року Е. Т. Віттекер (*E. T. Whittaker*). До цієї теореми у різних варіантах поверталися Г. Найквіст (*H. Nyquist*) 1928 року, В. О. Котельников 1933 року, К. Шеннон 1949 року та ін. Подамо доведення цієї теореми.

Нехай $f(t)$ – деяка функція, яка має на кожному скінченному проміжку скінченну кількість розривів і абсолютно інтегрована на проміжку $(-\infty,\infty)$, тобто

$$\int_{-\infty}^{\infty} |f(x)|dx < \infty. \tag{2.5}$$

Тоді у всіх точках диференційовності цієї функції справедливі рівності [26]:

$$f(t) = \int_{-\infty}^{\infty} c(\nu)\mathrm{e}^{i2\pi\nu t}\,d\nu, \quad c(\nu) = \int_{-\infty}^{\infty} f(t)e^{-i2\pi\nu t}\,dt. \tag{2.6}$$

Співвідношення (2.6) визначають обернене та пряме перетворення Фур'є. Функцію $c(\nu)$ називають трансформантою або образом Фур'є, а функцію $f(t)$ – оригіналом. Якщо t – час, то ν – частота, а функція $c(\nu)$ визначає неперервний спектр функції $f(t)$.

Функція $f(t)$ має обмежений спектр, якщо існує така частота ν_0, що для всіх ν, які задовольняють умову $|\nu| > \nu_0$, має місце рівність $c(\nu) = 0$.

Теорема 2.1. Про відліки

Нехай функція $f(t)$ має спектр, обмежений частотою ν_0. Тоді, за довільного кроку дискретизації $\Delta t < \Delta t_0 = 1/2\nu_0$, цю функцію можна повністю відтворити за множиною її значень у дискретних точках $t_k = \pm k\Delta t,\ k = 0,1,2,\ldots$.

Д о в е д е н н я

Здійснимо періодичне продовження функції-образу $c(\nu)$ на всю числову вісь з періодом $2\nu_0$ і розкладемо її в ряд Фур'є:

$$c(\nu) = \sum_{k=-\infty}^{\infty} c_k e^{i\pi k\nu/\nu_0}, \quad c_k = \frac{1}{2\nu_0}\int_{-\nu_0}^{\nu_0} c(\nu)e^{-i\pi k\nu/\nu_0}, \quad k = 0, \pm1, \pm2, \ldots. \tag{2.7}$$

Враховуючи обмеженість спектра, за формулою оберненого перетворення Фур'є (2.6) отримаємо:

$$f(t) = \int_{-\nu_0}^{\nu_0} c(\nu)\mathrm{e}^{i2\pi\nu t}\,d\nu. \tag{2.8}$$

За цією формулою знайдемо значення функції у точках відліку $t_k = k\Delta t$, покладаючи $\Delta t = \Delta t_0$:

$$f(k\Delta t) = \int_{-\nu_0}^{\nu_0} c(\nu) \mathrm{e}^{ik\pi\nu/\nu_0}\, d\nu . \tag{2.9}$$

Порівнюючи цей вираз з формулою для коефіцієнтів Фур'є (2.7), знайдемо:

$$c_k = \Delta t f(-k\Delta t),\ k = 0, \pm 1, \pm 2, \ldots . \tag{2.10}$$

Відліки функції $f(t)$ у вузлах $t_k = k\Delta t$ $(k = 0, \pm 1, \pm 2, \ldots)$ цілком визначають коефіцієнти розкладу в ряд Фур'є образу $c(\nu)$, а отже – і саму цю функцію

$$c(\nu) = T \sum_{k=-\infty}^{\infty} f(k\Delta t) e^{-i2\pi k\Delta t\, \nu} . \tag{2.11}$$

Підставляючи вираз (2.11) у формулу (2.8), отримаємо:

$$f(t) = T \sum_{k=-\infty}^{\infty} f(k\Delta t) \int_{-\nu_0}^{\nu_0} \mathrm{e}^{i2\pi(t-k\Delta t)\nu}\, d\nu \ . \tag{2.12}$$

Після обчислення інтеграла остаточно матимемо:

$$f(t) = \sum_{k=-\infty}^{\infty} f(k\Delta t) \frac{\sin\pi(t/\Delta t - k)}{\pi(t/\Delta t - k)} . \tag{2.13}$$

Отже, функція зі скінченним спектром повністю відновлюється за своїми значеннями в дискретних точках $t_k = k\Delta t$ $(k = 0, \pm 1, \pm 2, \ldots)$, де $\Delta t = \Delta t_0 = 1/(2\nu_0)$.

Легко довести, що формула відновлення (2.13) справедлива для будь-якого T, меншого T_0.

Мінімальну частоту дискретизації $2\nu_0$, необхідну для відновлення функції за її відліками в дискретних точках, називають *частотою Найквіста*.

Практична реалізація формули відтворення (2.13) здійснюється з деякою похибкою, зумовленою такими чинниками:

- спектр реальних сигналів дуже широкий;
- реальні сигнали обмежені в часі, тому кількість відліків є скінченною;
- технічно неможливо реалізувати ідеальний відновлюючий фільтр.

Отож на практиці крок дискретизації роблять дещо меншим (на 30–50 %), ніж передбачає теорема про відліки.

2.4. Кодування даних

Для створення принципової можливості автоматизованої передачі та обробки даних дуже важливо уніфікувати форму їхнього представлення. Для цього використовують *кодування* – представлення даних одного типу через стандартні дані іншого типу. Природні людські мови – це не що інше, як система кодування понять для вираження думок. Прикладами систем кодування є система запису математичних виразів, телеграфна азбука Морзе, система Брайля для сліпих та ін.

Подамо формальний опис процесу кодування. Нехай задано алфавіти $A=\{a_1,\ldots,a_n\}$ та $B=\{b_1,\ldots,b_m\}$. Під час кодування двійковими числами найчастіше $B=\{0,1\}$.

Позначимо:

A^*, B^* – множини слів, які можна скласти з алфавітів A та B, відповідно;

$S\subset A^*$ – деяка підмножина множини слів A^*.

Методом кодування називають функцію $F: S\to B^*$. Якщо $\alpha\in S$ – повідомлення, тоді $\beta\in B^*$ – код повідомлення. Обернену функцію $F^{-1}: B^*\to S$, якщо вона існує, називають *методом декодування*.

Алфавітним (*політерним*) *кодуванням* називають таке кодування, коли кодують окремі літери вихідного повідомлення. У цьому випадку $S=A$ і множина S є мінімальною. Слово $\beta_i=F(a_i)\in B^*$ називають кодом літери a_i, а множину кодів усіх літер $V=\left\{\beta_i=F(a_i)\in B^*,\ i=\overline{1,n}\right\}$ – множиною елементарних кодів. Алфавітне кодування часто задають за допомогою таблиці (схеми) кодування:

$$\sigma=\left\langle\alpha_1\to\beta_1,\ldots,\alpha_n\to\beta_n\right\rangle,\ \alpha_i\in A,\beta_i\in B^*,\ i=\overline{1,n}. \qquad (2.14)$$

Для аналізу завадостійкості схем кодування вводять поняття кодової відстані – $d(\beta',\beta')$. Детальніше про це йтиме мова у підрозділі 9.3.

Схему кодування називають *подільною* або *дешифровною*, якщо довільне слово, складене з елементарних кодів, однозначно розкладається на елементарні коди. Наприклад, код Морзе не є подільним, однак у ньому використовують спеціальний пропуск між літерами, який неявно є додатковим символом кодуючого алфавіту.

Схему кодування називають *префіксною,* якщо будь-який елементарний код не є початком довшого елементарного коду. Легко довести, що будь-яка префіксна схема кодування є подільною. Обернене твердження – не вірне. Метод побудови двійкових префіксних кодів за допомогою бінарних дерев описано у підрозділі 4.1.

Зрозуміло, що проблема подільності не є принциповою за використання елементарних кодів постійної довжини.

Загальна постановка задачі кодування: для заданих A, B та S знайти такий метод кодування, який має задані властивості (задовольняє певним обмеженням) і є оптимальним у деякому сенсі. Критерієм оптимальності найчастіше є довжина кодованого повідомлення. Опишемо можливі властивості (обмеження) методу кодування:

1. Існування декодування. Трансляція програм – це кодування, яке не вимагає однозначного декодування.

2. Завадостійкість, або здатність до виправляння помилок. Кодування F – завадостійке, якщо виконується умова:

$$\exists p>0 \ \forall \beta' \in B^*, \forall \beta'' \in V \left(d(\beta', \beta'') < p \Rightarrow F^{-1}(\beta') = F^{-1}(\beta'') \right). \qquad (2.15)$$

3. Задана складність (або простота) кодування та декодування. Для криптографії важливо, щоб F обчислювалось просто, а обчислення F^{-1} було складним.

Вирішальне значення для задачі кодування має природа множини S. Для опису цієї множини застосовують ймовірнісні, теоретико-множинні, логіко-комбінаторні та інші підходи.

За ймовірнісного опису $S = A$ і задано ймовірність p_i появи символу a_i у повідомленні, причому $\sum p_i = 1$. Якщо множину S задано словами, породженими формальною граматикою, то говорять про логіко-комбінаторний опис.

2.5. Представлення та кодування даних у комп'ютерах

Одиниці представлення даних та вимірювання кількості даних. Усі дані в комп'ютерах (текст, числа, програми та інше) представляють двійковим кодом (алфавіт $B = \{0,1\}$).

Найменша одиниця представлення даних – двійковий розряд (*binary digit*) має назву біт (*bit*), скорочене позначення – 1 б (1 b). Інші стандартні одиниці представлення даних такі:

1 байт (*byte*) = 8 біт, скорочене позначення – 1 Б (1 B);

1 слово (*word*) = 2 байти = 16 біт;

1 подвійне слово (*long word*) = 4 байти = 32 біти.

Для вимірювання великий масивів даних застосовують масштабніші одиниці:

1 кілобайт = 1024 байти, скорочено – 1 Кбайт або 1 КБ (1 *Kbyte*, 1 KB);

1 мегабайт = 1024 Кбайт, скорочено – 1 Мбайт або 1 МБ (1 *Mbyte*, 1 MB);

1 гігабайт = 1024 Мбайт, скорочено – 1 Гбайт або 1ГБ (1 *Gbyte*, 1 GB);

1 террабайт = 1024 Гбайт, скорочено – 1 Тбайт або 1ТБ (1 *Tbyte*, 1 TB).

Швидкість передавання даних вимірюють у бітах на секунду (*bits per second*), скорочено – біт/с (англійською – 1 bps = 1 b/s), та у відповідних похідних одиницях. Швидкість передавання сигналу – це число дискретних переходів чи елементарних імпульсів за секунду, одиниця вимірювання – бод (*baud*). Якщо біт кодується одним дискретним переходом чи імпульсом, то 1 бод = 1біт/с. В реальних комунікаціях швидкість передавання даних завжди менша від швидкості передавання сигналу.

Кодування текстових даних. У комп'ютерах для кодування текстових даних переважно застосовують алфавітне (політерне) кодування двійковим кодом постійної довжини. Наприклад, 8 двійкових розрядів достатньо, щоб закодувати 256 елементів.

У перші роки розвитку обчислювальної техніки стандартів кодування не існувало. І лише 1968 року *ANSI* (*American National Standard Institute*) ввів у дію систему кодування *ASCII* (*American Standard Code for Information Interchange*) з довжиною коду 8 біт.

Таблицю кодування зручно подати у вигляді квадратної таблиці символів розміром 16×16, де номер стовпчика позначає старшу цифру шіснадцяткового представлення коду, а номер рядка – молодшу цифру цього представлення.

Таблицю кодування *ASCII* (*ANSI*) умовно поділяють на такі дві таблиці: базову – коди 0–127 (00–7F); розширену – коди 128–255 (80–FF). Перші 32 коди базової таблиці кодують керуючі символи, а коди 32–127 – розділові знаки, цифри, великі і малі літери англійського алфавіту. Базова частина стала міжнародним стандартом, а розширену використовують для кодування національних алфавітів.

Проблема полягає в тому, що для одного і того ж національного алфавіту існує декілька стандартів кодування. Наприклад, у Радянському Союзі набув поширення стандарт кодування російського алфавіту КОИ–8, який сьогодні використовують у російському інтернеті. З часом виникли ще дві системи кодування – ГОСТ та ГОСТ-альтернативна, які маловживані. На основі другої з них 1991 року розробили стандарт українського алфавіту – РСТ УРСР–2018–91, більше відомий як *RUSCII* [14]. Фірма "*Microsoft*" створила інший стандарт кодування кириличних символів – *Windows–1251*, відмінний від *RUSCII*. Ще є стандарт Міжнародної організації зі стандартизації (*International Organization for Standardization, ISO*) для кодування символів російського та українського алфавіту.

Створено і впроваджено 16-бітний стандарт кодування – *Unicode*. Цей стандарт охоплює більшість існуючих алфавітів. Початкова таблиця збігається з таблицею *ASCII* (коди 0–00FF), а коди найпоширеніших абеток такі: 037E–03CE – грецький алфавіт; 0401–0491 – кирилиця; 0580–05F4 – іврит; 060C–06F9 – арабський алфавіт.

Нижче подано таблицю 2.1 з кодуванням українських літер, відмінних за написанням від російських, у найпоширеніших стандартах.

Таблиця 2.1. Кодування символів української абетки у різних стандартах

Стандарт / Літера	РСТ УРСР 2018-91	*Win–1251*	*ISO*	*Unicode Hex*
Ґ	242	165	163	0490
ґ	243	180	243	0491
Є	244	170	164	0404
є	245	186	244	0454
І	246	178	166	0406
і	247	179	246	0456
Ї	248	175	167	0407
ї	249	191	247	0457

2.6. Представлення чисел у комп'ютерах

Насамперед потрібно розрізняти стандарти форматів і методів арифметики чисел та їхню реалізацію у процесорах і мовах програмування (трансляторах).

Кодування цілих чисел не викликає особливих труднощів. Широко використовують такі типи цілих чисел:

Ціле число довжиною в слово має розмір одного слова (16 біт). Числа від 0 до 32767 записують безпосередньо в двійковій системі числення, а від'ємні числа від -1 до -32768 записують доповнювальним кодом ($-x \to 2^{16} - x$). Таблиця кодування виглядає так:

Шістнадцядковий код	0000	0001	...	7FFF	8000	8001	...	FFFF
Число	0	1	...	32767	–32768	–32767	...	–1

Ціле довжиною в подвійне слово займає 4 байти і кодується аналогічно. Діапазон його значень від -2^{31} до $2^{31}-1$. *Ціле довжиною в чотири слова* має довжину 8 байт, дає змогу задати цілі числа в діапазоні $[-2^{-63},\ 2^{63}-1] \subset \mathbf{Z}$.

Мови програмування можуть мати типи цілих чисел, які відповідають вказаним, а також додаткові типи. Наприклад, ціле довжиною в слово відповідає типу *Integer* мови *Pascal*, а ціле довжиною у подвійне слово – типу *LongInt.*

Для уніфікації способів представлення дробових чисел у комп'ютерах розроблено міжнародний стандарт ISO/IEC 60559, який є ідентичним стандарту IEEE 754 і визначає формати та методи для арифметики з плаваючою комою в комп'ютерних системах [23; 24].

Для кодування дійсних чисел у цьому стандарті використовують *формат з плаваючою комою* (крапкою), який ґрунтується на представленні чисел у нормалізованій формі.

Число a в *нормалізованій* (*експонентній*) *формі* записують так:

$$a = \pm q^p \sum_{i=1}^{m} a_i q^{-i} = \pm a_0.a_1a_2\ldots a_m \cdot q^p,\ (a_i \in \{0,1\ldots q-1\},\ a_0 \neq 0), \qquad (2.16)$$

де $q \in \{2, 10, 8, 16\}$ – основа системи числення (на машинному рівні $q=2$); $a_0.a_1a_2\ldots a_m$ – *мантиса*; $p \in Z$ – *порядок* (*експонента*).

Зауважимо, що у нормалізованому записі двійкових чисел нульовий розряд завжди рівний одиниці ($a_0 = 1$) і його можна не зберігати (так звана прихована одиниця).

Відповідно до цього стандарту, числа з плаваючою комою мають такий формат:

- перший біт відводять для знаку числа;
- наступні r біт займає порядок числа – ціле зі знаком з діапазону $[-2^{r-1}, 2^{r-1}-1]$;
- останні m біт відводять для дробової частини мантиси (хвоста мантиси).

Основні типи чисел з плаваючою комою, які передбачені у стандарті [24], подано у таблиці 2.2.

Таблиця 2.2. Основні типи чисел з плаваючою комою

Тип	Загальна довжина, біт	Довжина порядку, біт	Довжина хвоста мантиси, біт
Single	32	8	23
Double	64	11	52
Quadruple	128	15	112

Вони забезпечують певні діапазони подання десяткових чисел (табл. 2.3).

Таблиця 2.3. Діапазони подання чисел з плаваючою комою

Тип	Число десяткових знаків	Максимальний модуль десяткового порядку
Single	7-8	~38
Double	15-16	~308
Quadruple	34-35	~4932

Множина чисел, яку можна представити у форматі з плаваючою комою конкретного типу, відповідно до (2.16), є скінченною множиною раціональних чисел:

$$F = \left\{ a \in \mathbf{Q} \,\middle|\, a = \pm 1, a_1 a_2 \ldots a_m \cdot 2^p,\ a_i \in \{0,1\},\ p \in \mathbf{Z},\ p^- \le p \le p^+ \right\}. \qquad (2.17)$$

Ці числа розміщені нерівномірно, відстань між двома найближчими числами з порядком p дорівнює 2^{p-m}.

Відстань між одиницею і найближчим наступним за нею числом дорівнює 2^{-m}. Цю величину називають *машинним іпсилон* і позначають ідентифікатором *macheps*.

Для прикладу розглянемо представлення чисел у 32-розрядних процесорах *Intel*. Цей процесор має декілька типів цілих та дійсних чисел.

Для кодування цілих чисел використано вже згадані цілі довжиною в одне, два та чотири слова.

Для кодування дробових чисел використано три типи чисел з плаваючою комою: одинарної точності – довжиною 4 байти, подвійної точності – довжиною 8 байт і підвищеної точності – довжиною 10 байт. Представлення чисел аналогічне (2.16) з тією різницею, що $a_0 = 0$. Конкретні параметри для зазначених типів чисел з плаваючою комою подано у таблиці 2.4.

Таблиця 2.4. Параметри подання чисел з плаваючою комою у 32-розрядних процесорах *Intel*

Тип	Загальна довжина, біт	Довжина порядку, біт	Довжина мантиси, біт
Single	32	8	23
Double	64	11	52
Extended	80	15	64

Однак насправді реалізовано ширший діапазон порядків, передусім у від'ємній області – за рахунок зменшення кількості значущих цифр. Нижче у таблиці 2.5 наведено допустимі модулі для чисел у форматі з плаваючою комою для 32-розрядних процесорів *Intel*.

Таблиця 2.5. Допустима величина та розрядність чисел з плаваючою комою для 32-розрядних процесорів *Intel*

Тип	Наближений десятковий діапазон модуля	Кількість десяткових знаків мантиси
Single	$1{,}5\cdot10^{-45}$ –$3{,}4\cdot10^{38}$	7–8
Double	$5{,}0\cdot10^{-324}$ – $1{,}7\cdot10^{307}$	15–16
Extended	$3{,}6\cdot10^{-4932}$ – $1{,}1\cdot10^{4932}$	19–20

Формати та методи для арифметики з плаваючою комою для інших процесорів описано у їхній технічній документації. Імплементацію стандарту IEEE 754 у мовах програмування розглянуто у довіднику [23].

Числа, модуль яких менший за допустимий, називають *машинним нулем*, а числа, модуль яких перевищує допустимий – *машинною нескінченністю*. Якщо під час виконання арифметичних операцій модуль числа стає меншим від машинного нуля, то настає помилка "зникнення порядку", а якщо більшим від "машинної нескінченності" – то помилка "переповнення порядку".

Наближення довільного дійсного числа $x\,(x \neq 0)$ числом з плаваючою комою $fl(x) \in F$ має відносну похибку $\delta(x) = \left|x - fl(x)\right| / \left|x\right|$. Якщо наближення здійснюється відкиданням зайвих розрядів без округлення, то максимальна відносна похибка дорівнює машинному іпсилон – *macheps*. Коли застосовується класичне правило округлення, то верхня оцінка похибки зменшується вдвічі $\delta(x) \leq 0{,}5\,macheps$.

Отже, при виконанні на комп'ютері арифметичних операцій над числами у форматі з плаваючою комою виникають похибки обчислень, можливі помилки переповнення та зникнення порядку.

Запитання та завдання

1. Чим відрізняються дані від інформації?
2. Опишіть загальну схему передавання інформації від джерела до адресата.
3. Опишіть принцип перетворення аналогових сигналів у цифрові.
4. Що таке частота дискретизації та глибина квантування?
5. Сформулюйте теорему про відліки.
6. Дайте формальний опис кодування даних.
7. Наведіть приклади кодування даних.
8. Назвіть основні стандарти кодування даних у комп'ютерах.
9. Що таке алфавітне кодування?
10. Яку схему кодування називають префіксною?
11. Назвіть основні одиниці кількості інформації.
12. Опишіть експонентний формат представлення чисел у комп'ютері.

Розділ 3. КІЛЬКІСНА ОЦІНКА ІНФОРМАЦІЇ ДИСКРЕТНИХ ДЖЕРЕЛ

3.1. Кількість інформації випадкової події

Для порівняння між собою різних повідомлень важливо ввести кількісну міру, яка даватиме змогу оцінити кількість інформації у повідомленні. Вперше таку міру 1928 року ввів американський математик Р. Гартлі (*R. Hartley*). Згодом, 1948 року, К. Шеннон ввів поняття ентропії джерела інформації, повне обґрунтування якого дав радянський математик О. Я. Хінчін на основі теорії стаціонарних випадкових процесів.

Виходячи з емпіричних уявлень про природу інформації, можна стверджувати:

1) коли деякий об’єкт може перебувати в одному із n можливих станів, то існує апріорна невизначеність щодо істинного стану об’єкта, а отримання повідомлення про його актуальний стан цю невизначеність усуває;

2) якщо кількість можливих станів об’єкта $n=1$, то повідомлення про його стан не несе жодної інформації;

3) якщо стан об’єкта є випадковою подією, то кількість інформації, яку несе повідомлення про те, що об’єкт перейде у i-й стан ($i=\overline{1,n}$), визначається ймовірністю p_i цього стану, і менш ймовірні події несуть більшу інформацію;

4) кількість інформації є адитивною величиною: якщо подія C є добутком незалежних подій A і B, то кількість інформації, породженої подією C, дорівнює сумі кількостей інформації подій A та B.

Нехай A – випадкова подія, ймовірність якої дорівнює p. Виходячи з зазначених емпіричних міркувань, кількість інформації, яку принесе ця подія, є функцією її ймовірності

$$I(A)=\varphi(p) \qquad (3.1)$$

з такими властивостями:

1) $\varphi(x):[0,1]\to R$ – монотонно спадна;

2) $\varphi(1)=0$;

3) $\varphi(xy)=\varphi(x)+\varphi(y)$.

Умови 1–3 задовольняє лише логарифмічна функція:

$$\varphi(x)=C\log_2\frac{1}{x},\ C>0. \tag{3.2}$$

Це випливає з теореми 3.1.

Теорема 3.1. Про оцінку кількості інформації випадкової події
Розглянемо функцію $f(x):[1,\infty)\to\mathbf{R}$ *з такими властивостями:*
1) $f(x)$ *строго зростає на проміжку* $[1,\infty)$*;*
2) $\forall x,y\in[1,\infty)\ f(xy)=f(x)+f(y)$.
Тоді $f(x)=C\log_2 x,\ C>0$.

Д о в е д е н н я

З властивостей 1–2 очевидним чином випливає, що

$f(1)=0;\ \ 0<f(2)<\infty;\ \ \forall x\in[1,\infty)\ \forall n>0\ \ f(x^n)=nf(x)$.

Нехай $n\in\mathbf{N},\ x\geq 1$. Покладемо $m=\lfloor n\log_2 x\rfloor$. Тоді справджується нерівність

$$2^m\leq x^n<2^{m+1}. \tag{3.3}$$

Це очевидно після її логарифмування:

$$m\leq n\log_2 x<m+1. \tag{3.4}$$

Оскільки функція $f(x)$ строго зростаюча, то з нерівності (3) отримаємо

$$mf(2)\leq nf(x)<(m+1)f(2). \tag{3.5}$$

З двох останніх нерівностей маємо:

$$\left|f(x)-f(2)\log_2 x\right|<\frac{f(2)}{n}. \tag{3.6}$$

Здійснюючи в (3.6) граничний перехід $n\to\infty$, отримаємо:

$$f(x)=f(2)\log_2 x. \tag{3.7}$$

Теорему доведено.

3.2. Ентропія дискретного джерела інформації

Розглянемо ймовірнісну модель дискретного джерела інформації. Припустимо, що джерело інформації має скінченну множину станів $\{A_1, A_2, \ldots, A_n\}$, ймовірність стану A_i дорівнює p_i, $\sum_{i=1}^{n} p_i = 1$. Множину станів джерела можна ототожнити з деяким алфавітом $A = \{a_1, a_2, \ldots, a_n\}$, вважаючи, що подія A_i полягає у появі символу a_i. Тоді джерело інформації повністю задається ансамблем

$$S = \left\{ a_i \to p_i,\ i = \overline{1,n},\ \sum_{i=1}^{n} p_i = 1 \right\}. \tag{3.8}$$

Послідовність символів, породжена джерелом, утворює повідомлення джерела. Якщо поява чергового символу є незалежною подією, то таке джерело називають *джерелом Бернуллі*, а його повідомлення – *простими повідомленнями*. Якщо ймовірність появи чергового символу залежить від попередніх літер повідомлення, то таке джерело називають *джерелом з пам'яттю*, а його повідомлення – *складними повідомленнями*. Дослідження складних повідомлень базується на теорії стаціонарних випадкових послідовностей.

Надходження одного символу джерела інформації називають елементарним повідомленням джерела. За результатами підрозділу 3.1, кількість інформації елементарного повідомлення a_i дорівнює

$$I(a_i) = \log_2 \frac{1}{p_i}. \tag{3.9}$$

Однак ця величина є апостеріорною і ніяк не характеризує джерело інформації загалом. Чи можна дати апріорну оцінку кількості інформації без її фактичного отримання?

Клод Шеннон [33] запропонував вважати апріорною мірою кількості інформації елементарного повідомлення математичне сподівання кількості інформації цього повідомлення:

$$H(S) = \sum_{i=1}^{N} p_i \log_2 \frac{1}{p_i}. \tag{3.10}$$

Цю величину називають *безумовною ентропією джерела*.

Легко встановити найпростіші властивості ентропії:

1. Ентропія залежить не від конкретних значень величин a_i, а лише від їхніх ймовірностей – p_i.

2. Ентропія – невід'ємна величина; вона дорівнює нулю тоді і лише тоді, коли ймовірність одного зі станів рівна 1, а всіх інших – 0.

3. Ентропія максимальна, коли всі стани джерела рівноймовірні:

$$p_i = \frac{1}{n},\ i = \overline{1,n},\ \max H(S) = \log_2 n. \tag{3.11}$$

Це легко встановити методом множників Лагранжа. Власне так означив кількість інформації дискретного джерела з n станами Р. Гартлі.

4. Ентропія об'єднання декількох статистично незалежних джерел інформації дорівнює сумі ентропій вихідних джерел.

Нехай маємо два статистично незалежні джерела інформацій, які задаються ансамблями:

$$U = \left\{ u_i \to p_i,\ i = \overline{1,n},\ \sum_{i=1}^{n} p_i = 1 \right\} \text{ та } V = \left\{ v_k \to q_k,\ k = \overline{1,m},\ \sum_{i=1}^{m} q_k = 1 \right\}.$$

Тоді об'єднане джерело задається ансамблем

$$(U,V) = \left\{ (u_i, v_k) \to p_i q_k,\ i = \overline{1,n},\ k = \overline{1,m},\ \sum_{i=1}^{n}\sum_{k=1}^{m} p_i q_k = 1 \right\}. \tag{3.12}$$

Легко довести, що ентропія об'єднаного джерела дорівнює сумі ентропій окремих джерел

$$H(U,V) = \sum_{i=1}^{n}\sum_{k=1}^{m} p_i q_k \log_2 \frac{1}{p_i q_k} = H(U) + H(V). \tag{3.13}$$

5. Доведемо, що для статистично залежних джерел справджується нерівність

$$H(U,V) \le H(U) + H(V). \tag{3.14}$$

Об'єднане джерело для цього випадку задається ансамблем

$$(U,V) = \left\{ (u_i, v_j) \to p(u_i, v_j),\ i = \overline{1,n},\ j = \overline{1,m},\ \sum_{i=1}^{n}\sum_{j=1}^{m} p(u_i, v_j) = 1 \right\}, \tag{3.15}$$

де $p(u_i, v_j)$ – ймовірність сукупності (u_i, v_j), яка виражається через умовні ймовірності так:

$$p(u_i,v_j)=p(u_i)p(u_i|v_j)=p(v_j)p(v_j|u_i). \tag{3.16}$$

Зауважимо ще деякі очевидні формули для умовних ймовірностей:

$$\sum_{j=1}^{m} p(v_j|u_i)=1,\ i=\overline{1,n}; \tag{3.17}$$

$$p(v_j)=\sum_{i=1}^{n} p(u_i)p(v_j|u_i),\ j=\overline{1,m}. \tag{3.18}$$

З урахуванням формул (3.16) і (3.17) за означенням ентропії знайдемо:

$$\begin{aligned} H(U,V)&=-\sum_{i=1}^{n}\sum_{j=1}^{m} p(u_i,v_j)\log_2 p(u_i,v_j)= \\ &=-\sum_{i=1}^{n}\sum_{j=1}^{m} p(u_i)p(v_j|u_i)\left(\log_2 p(u_i)+\log_2 p(v_j|u_i)\right)= \\ &=H(U)+H(U|V), \end{aligned} \tag{3.19}$$

де $H(V|U)=-\sum_{i=1}^{n} p(u_i)\sum_{j=1}^{m} p(v_j|u_i)\log_2 p(v_j|u_i)$ – повна умовна ентропія джерела V відносно джерела U.

Доведемо, що

$$H(V|U)\le H(V). \tag{3.20}$$

Легко переконатись, що функція $f(x)=-x\log_2 x$ – опукла вгору і для неї справджується нерівність Єнсена [11]:

$$f\left(\sum_{i=1}^{n}\alpha_i x_i\right)\ge\sum_{i=1}^{n}\alpha_i f(x_i),\ \alpha_i>0,\ i=\overline{1,n},\ \sum_{i=1}^{n}\alpha_i=1. \tag{3.21}$$

За її допомогою з використанням формули (3.18) отримаємо:

$$H(V|U) = -\sum_{i=1}^{n} p(u_i) \sum_{j=1}^{m} p(v_j|u_i) \log_2 p(v_j|u_i) =$$

$$= -\sum_{j=1}^{m} \sum_{i=1}^{n} p(u_i) p(v_j|u_i) \log_2 p(v_j|u_i) \le$$

$$\le -\sum_{j=1}^{m} \left(\sum_{i=1}^{n} p(u_i) p(v_j|u_i) \right) \log_2 \left(\sum_{i=1}^{n} p(u_i) p(v_j|u_i) \right) =$$

$$= -\sum_{j=1}^{m} p(v_j) \log_2 p(v_j) = H(V).$$

Зауваження 1. Ентропія повідомлення

Повідомлення джерела S розміром у M символів можна розглядати як елементарне повідомлення сукупного джерела $S^M = (S, S, \ldots, S)$. Для ентропії цього джерела виконується нерівність

$$H(S^M) \le M \cdot H(S), \tag{3.22}$$

яка перетворюється у рівність для простих повідомлень.

Зауваження 2. Одиниця кількості інформації

Яка максимальна ентропія одного біта даних? Максимальну ентропію матиме ансамбль $U = \left\{ 0 \to \frac{1}{2}, 1 \to \frac{1}{2} \right\}$. Його ентропія дорівнює $H(U) = \log_2 2 = 1$. Тому одиницю кількості інформації називають так само, як і міру кількості даних – біт. Це також пояснює вибір константи у формулі (3.9).

Зауваження 3. Ентропія n-розрядного двійкового числа

Двійкове число з n розрядами можна трактувати як джерело з 2^n станами. Його максимальна ентропія дорівнює $H = \sum_{k=1}^{2^n} \frac{1}{2^n} \log_2 2^n = n$. Отож максимальна кількість отриманої інформації дорівнює кількості двійкових розрядів. У цьому сенсі кількість інформації та розрядність двійкового числа еквівалентні.

Зауваження 4. Ентропія стаціонарного джерела

Розглянемо нескінченні (у двох напрямках) послідовності літер алфавіту $A = \{A_1, A_2, \ldots, A_n\}$. Їхня множина утворює повний простір подій $\Omega = A^\infty$.

Нехай $S_i^j \subset A^\infty$ – множина послідовностей, які містять літеру a_i на j-му місці, тоді множини $S^j = \bigcup_{i=1}^{n} S_i^j$ утворюють розбиття A^∞.

Джерело S називають *стаціонарним джерелом*, якщо повні та умовні ймовірності появи символів на деякому місці не залежать від зсувів, тобто:

$$p\left(s_{i_0}^{j_0}\right) = p\left(s_{i_0}^{0}\right),\ p\left(s_{i_0}^{j_0} \middle| s_{i_1}^{j_1} \dots s_{i_k}^{j_k}\right) = p\left(s_{i_0}^{0} \middle| s_{i_1}^{j_1 - j_0} \dots . s_{i_k}^{j_k - j_0}\right). \tag{3.23}$$

Нехай $H_m = \frac{1}{m} H(S^1, S^2, \dots, S^m)$ – середня ентропія однієї літери повідомлення з m символів (m-грами), а $\kappa_m = H\left(S^m \middle| S^1 S^2 \dots S^{m-1}\right)$ – повна умовна ентропія m-го символу відносно попередніх $m-1$ символів. Легко встановити, що $\kappa_m = H(S^1 S^2 \dots S^m) - H(S^1 S^2 \dots S^{m-1})$. Тому

$$H(S^1 S^2 \dots S^m) = \sum_{i=1}^{m} \kappa_i . \tag{3.24}$$

Послідовність κ_m – монотонно спадає та існує границя $\lim\limits_{m\to\infty} \kappa_m \geq 0$. Тому з формули (3.24) випливає існування границі $\lim\limits_{m\to\infty} H_m = H_\infty$:

$$\lim_{m\to\infty} H_m = \lim_{m\to\infty} \kappa_m = H_\infty \geq 0. \tag{3.25}$$

Величину H_∞ називають ентропією стаціонарного джерела. Зауважимо, що для джерела Бернуллі виконується рівність:

$$H = H_1 = H_2 = \dots = H_\infty . \tag{3.26}$$

Зауваження 5. Ентропія української мови

Для випадку людських або синтетичних мов джерело інформації описують граматично, проте можна перейти до його ймовірнісного опису, підраховуючи частоти окремих літер у великих текстах.

Алфавіт української мови має 33 літери; крім того, ще вживають пропуски між словами (␣) , розділові знаки (, . : ; – ! ?) та цифри (0 1 2 3 4 5 6 7 8 9). Загалом маємо 51 символ. Максимальна ентропія такого джерела становить $H = \log_2 51 \approx 5{,}67$ біт/літера. Однак підрахунок ентропії з урахуванням статистичної залежності літер дає величину H_8 порядку 2 біти/літера.

Середню частоту використання найуживаніших символів українського алфавіту та біграм наведено у таблиці 3.1 згідно з джерелом [2].

Таблиця 3.1. Середня частота найуживаніших українських літер та біграм

Окремі символи	Біграми	Біграми з пропуском
␣ – 0,134	ст – 0,017	␣п – 0,016
о – 0,082	на – 0,016	␣в – 0,015
н – 0,070	но – 0,014	и␣ – 0,014
а – 0,070	ан – 0,014	о␣ – 0,013

Зауваження 6. Ентропія – технічна міра інформації

Наприклад: невизначеність дії ліків, які у 90 % дають покращення, а у 10 % – погіршення, та ліків, які у 10 % дають покращення, а у 90 % – погіршення – однакова.

Запитання та завдання

1. Як оцінюють кількість інформації випадкової події?
2. Яке джерело інформації називають джерелом Бернуллі?
3. Що таке ентропія дискретного джерела інформації?
4. Які основні властивості ентропії?
5. Сформулюйте екстремальні властивості ентропії джерела.
6. Чому дорівнює ентропія об’єднання кількох статистично незалежних джерел?
7. Ентропія об’єднання статистично залежних джерел.
8. Що таке умовна ентропія?
9. Назвіть одиниці вимірювання кількості інформації.
10. Чому дорівнює максимальна ентропія *n*-розрядного двійкового числа?

Розділ 4. ЕФЕКТИВНЕ КОДУВАННЯ

4.1. Побудова двійкових префіксних кодів за допомогою бінарних дерев

Префіксні схеми кодування є подільними, тобто забезпечують однозначне дешифрування. Опишемо метод побудови двійкових префіксних кодів за допомогою бінарних дерев [19].

Розглянемо довільне бінарне дерево (рис. 4.1).

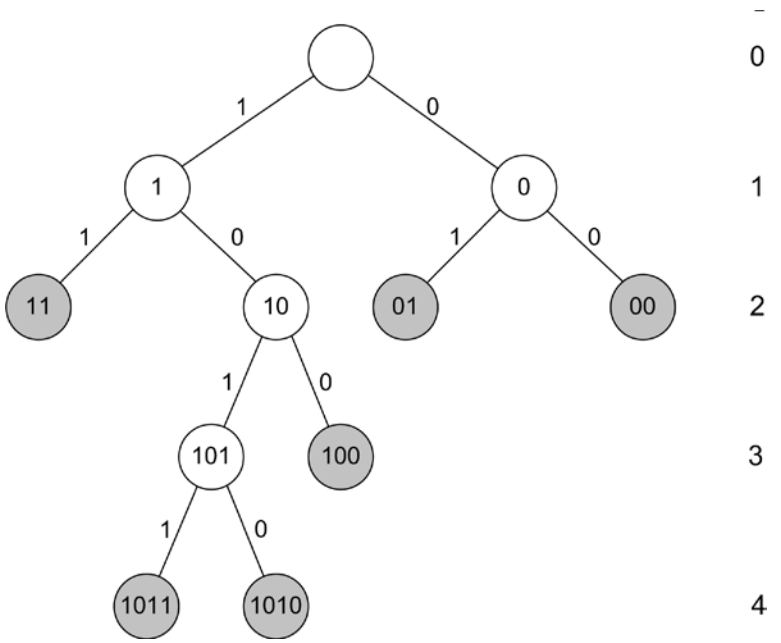


Рис. 4.1. Побудова двійкового префіксного коду
за допомогою бінарного дерева

Припишемо кожному лівому ребру цього дерева цифру "один", а правому – цифру "нуль". Тоді для кожної вершини можна визначити її двійковий код – шлях від кореня до цієї вершини.

Насамперед зауважимо, що отримані таким способом коди – різні, довжина коду вершини k-го рівня дорівнює k, а максимальна кількість вершин цього рівня дорівнює 2^k.

Розглянемо коди кінцевих вершин – листків дерева. Нехай X – код деякого листка. Коди вершин, які розміщені на шляху від кореня до цього листка, утворюють повну множину префіксів X. Код жодного іншого листка не співпадає з кодом X і не може бути його префіксом, отож множина кодів листків є префіксною множиною.

Ми довели теорему 4.1.

Теорема 4.1. Про множину кодів листків бінарного дерева
Множина кодів листків бінарного дерева є префіксною множиною.

Можна довести також обернене твердження.

Теорема 4.2. Про множину префіксних кодів
Будь-яка множина префіксних кодів є деякою підмножиною кодів листків бінарного дерева.

Д о в е д е н н я

Нехай маємо префіксну множину двійкових кодів з довжинами кодів $1 \le \lambda_1 < \lambda_2 < \ldots < \lambda_m$. Розглянемо повне бінарне дерево висоти λ_m. Воно має 2^{λ_m} листків. Можливих двійкових кодів довжини λ_m є також 2^{λ_m}. Тому для будь-якого коду зазначеної довжини можна знайти відповідний листок. На гілках, які відповідають цим листкам, є лише префікси, тому там відсутні коди з вихідної префіксної множини кодів. Відкидаючи парні листки рівня λ_m, коди яких не належать до вихідної множини, утворюємо листки рівня $\lambda_m - 1$. Застосовуючи подібні міркування, переходимо до рівня $\lambda_m - 2$ і т. д., доки не прийдемо до рівня λ_1.

Лема 4.1
Нехай задано довільне бінарне дерево висоти m. Кількість листків рівня k позначимо через ν_k ($\nu_k \ge 0$, $\nu_m > 0$). Тоді виконується рівність

$$\sum_{k=1}^{m} \frac{\nu_k}{2^k} = 1 . \qquad (4.1)$$

Д о в е д е н н я

Припишемо кожному вузлу рівня k бінарного дерева вагу $1/2^k$. Бінарне дерево можна подати як результат росту дерева від кореня до m-го рівня.

Утворення нової пари листків k-го рівня можна трактувати як поділ листка $k-1$-го рівня. При цьому вага листків не змінюється, оскільки два листки k-го рівня мають вагу одного листка $k-1$-го рівня: $\frac{1}{2^k}+\frac{1}{2^k}=\frac{1}{2^{k-1}}$.

Теорема 4.3. Про нерівність Крафта

Нехай $1\le l_1\le l_2\le\ldots\le l_n=m$ *– цілі додатні числа. Тоді для існування префіксного коду потужності* n *з довжинами слів* $l_1,l_2,\ldots,l_n$ *необхідно і достатньо виконання нерівності*

$$\sum_{i=1}^{n}\frac{1}{2^{l_i}}\le 1. \tag{4.2}$$

Д о в е д е н н я

Необхідність. Позначимо κ_k – кількість двійкових кодів довжини k вихідної префіксної множини. За Теоремою 4.2, ця множина є підмножиною кодів листків деякого бінарного дерева, яке має ν_k листків k-го рівня. Тому $\kappa_k\le\nu_k$. Отож на основі леми отримаємо:

$$\sum_{i=1}^{n}\frac{1}{2^{l_i}}=\sum_{k=1}^{m}\frac{\kappa_k}{2^k}\le\sum_{k=1}^{m}\frac{\nu_k}{2^k}=1.$$

Достатність. Нехай κ_k – кількість двійкових кодів довжини k. Подамо нерівність $\sum_{k=1}^{m}\frac{\kappa_k}{2^k}\le 1$ у вигляді $2^m\ge\sum_{k=1}^{m}\kappa_k 2^{m-k}$, яку перепишемо так:

$$\Big(\ldots\Big(\ldots\Big(\big(2-\kappa_1\big)2-\kappa_2\Big)2-\ldots-\kappa_k\Big)2-\ldots-\kappa_{m-1}\Big)2-\kappa_m\ge 0.$$

Ця нерівність відповідає процесу побудови деякого бінарного дерева з κ_k листками k-го рівня до рівня $m-1$. Лише на рівні m це дерево матиме $2^m-\sum_{k=1}^{m-1}\kappa_k 2^{m-k}\ge\kappa_m$ листків. Отож ми отримали бінарне дерево, яке має κ_k листків k-го рівня для $k=\overline{1,m-1}$, і більше, ніж κ_m листків – на рівні m. Отже, ми побудували бінарне дерево, множина кодів листків якого є вихідною множиною кодів.

Нерівність (4.2) встановив 1949 року Л. Г. Крафт (L. G. Kraft) у своїй магістерській роботі. Б. Макміллан (B. McMillan) 1956 року довів, що ця нерівність справедлива для будь-якого дешифровного коду [19].

4.2. Теорема Шеннона про ефективне кодування

Розглянемо дискретне джерело інформації, яке описує модель Бернуллі:

$$S=\left\{a_i \to p_i,\ i=\overline{1,n},\ \sum_{i=1}^{n} p_i=1\right\}. \tag{4.3}$$

Повідомлення цього джерела кодуватимемо двійковим кодом змінної довжини посимвольно (алфавітне кодування), або біграмами, триграмами – загалом блоками довжиною m.

Середня довжина коду – це довжина кодованого повідомлення, розділена на довжину вихідного повідомлення. За алфавітного кодування, коли символ a_i кодується двійковим кодом $\beta_i = f(a_i)\in B^*$ довжиною $l_i = \left| f(a_i)\right|$, середня довжина коду для довгих повідомлень наближено дорівнюватиме

$$L \approx \sum_{i=1}^{n} p_i l_i\,. \tag{4.4}$$

Розглянемо приклад. Нехай вихідне джерело має алфавіт $A=\{a,b,c\}$ з ймовірностями букв $p(a)=\frac{1}{4},\ p(b)=\frac{1}{2},\ p(c)=\frac{1}{4}$, а схему кодування задано таблицею $a\to 11,\ b\to 0,\ c\to 10$. Для довгих повідомлень отримаємо: $L=\frac{1}{4}\cdot 2+\frac{1}{2}\cdot 1+\frac{1}{4}\cdot 2=1\frac{1}{2}.$

Середню довжину коду часто називають *ціною кодування* і позначають літерою C.

Надлишковістю кодування називають величину:

$$R = L - H\,. \tag{4.5}$$

Наведемо формулювання теореми Шеннона про ефективне кодування – першої теореми Шеннона для каналу без перешкод [33].

Теорема 4.4. Теорема Шеннона про кодування для джерела Бернуллі

Дуже довге просте повідомлення завжди можна закодувати префіксним m-блочним двійковим кодом змінної довжини так, що виконуватиметься нерівність:

$$0 \le R \le \frac{1}{m}. \tag{4.6}$$

Це означає, що середня довжина коду буде як завгодно близькою до ентропії джерела, проте не меншою від неї.

Теорема 4.5. Теорема Шеннона про кодування для стаціонарного джерела

Дуже довге повідомлення стаціонарного джерела інформації завжди можна закодувати префіксним блочним двійковим кодом змінної довжини так, що середня довжина коду буде як завгодно близькою до ентропії джерела, проте не меншою від неї.

Теореми Шеннона про ефективне кодування не дають алгоритму конструктивної побудови ефективних кодів, тобто таких кодів, які дають найменшу довжину кодованих послідовностей. Однак інтуїтивно зрозуміло, як будувати такі коди: символи вихідного алфавіту, які трапляються частіше, повинні мати меншу довжину коду.

4.3. Алгоритм Шеннона–Фано

Одним з перших алгоритмів ефективного кодування вважають *алгоритм Шеннона–Фано*. Розроблений К. Шенноном у співпраці з Р. Фано (*R. Fano*).

Формування кодової таблиці за цим алгоритмом здійснюють так:

1. У перший стовпчик таблиці записують знаки вихідного алфавіту у порядку спадання їхніх ймовірностей, які записують у стовпчику поряд. Для кодів відводять третій стовпчик таблиці.

2. Знаки розподіляють на дві групи – так, щоб сума ймовірностей у кожній з них була приблизно однаковою. До кодів знаків верхньої групи справа дописують символ "1", а нижньої – "0".

3. Кожну з отриманих груп, своєю чергою, розбивають на дві підгрупи з приблизно однаковою сумою ймовірностей і проводять аналогічне дописування одиниць та нулів до їхніх кодів.

4. Процес завершують, коли всі підгрупи складаються лише з одного знака.

Розглянемо кілька прикладів.

Приклад 4.1. Розглянемо джерело з алфавітом у вісім знаків. Вихідні дані та результати кодування за алгоритмом Шеннона–Фано наведено нижче:

Знак	Ймовірність	Код
z_1	1/2	1
z_2	1/4	01
z_3	1/8	001
z_4	1/16	0001
z_5	1/32	00001
z_6	1/64	000001
z_7	1/128	0000001
z_8	1/128	0000000

Ентропія джерела дорівнює $H(U)=-\sum_{i=1}^{8} p_i \log(p_i)=1\frac{63}{64}$ біт/літера, а середня довжина коду – $L(f,S)=\sum_{i=1}^{8} p_i \left| f(a_i) \right|=1\frac{63}{64}$ біт/літера. Отож надлишковість кодування є нульовою.

Зауважимо, що для цього випадку код постійної довжини матиме щонайменше 3 двійкові знаки, і надлишковість становитиме $R=1\frac{1}{64}$ біт/літера.

Приклад 4.2. Розглянемо випадок, коли середня довжина коду більша від ентропії джерела.

Знак	Ймовірність	Код
z_1	0,22	11
z_2	0,20	101
z_3	0,16	100
z_4	0,16	01
z_5	0,10	001
z_6	0,10	0001
z_7	0,04	00001
z_8	0,02	00000

Для цього випадку ентропія джерела дорівнює $H \approx 2{,}76$ біт/літера, ціна кодування – $L \approx 2{,}84$ біт/літера, а надлишковість становить $R \approx 0{,}08$.

Зауважимо, що у цьому прикладі алгоритм Шеннона–Фано не забезпечує однозначної побудови коду.

З теореми Шеннона випливає, що надлишковість можна зменшити, якщо перейти до кодування блоками. Розглянемо приклад блочного кодування.

Приклад 4.3. Розглянемо джерело Бернуллі, алфавіт якого має дві літери z_1 та z_2 з ймовірностями появи, відповідно, $p(z_1)=0{,}9$ та $p(z_2)=0{,}1$. Ентропія цього джерела – $H(S)=0{,}47$.

За алфавітного кодування для кодування одного знака вихідного алфавіту потрібно один двійковий розряд, тому $L=1$. Надлишковість становить $R=0{,}53$.

Перейдемо до кодування блоками по дві літери – біграмами. Тоді джерело можна описати ансамблем: $S_2=\{z_1z_1-0.81;\ z_1z_2-0.09;\ z_2z_1-0.09;\ z_2z_1-0.01\}$.

Легко бачити, що ентропія цього джерела в розрахунку на одну літеру вихідного алфавіту така ж, як вихідного джерела.

Проведемо кодування методом Шеннона–Фано:

Блоки	Ймовірність	Кодова комбінація
z_1z_1	0,81	1
z_1z_2	0,09	01
z_2z_1	0,09	001
z_2z_1	0,01	000

Отримали, що середнє число двійкових розрядів на блок дорівнює $L_2=1{,}29$ біт/блок , а на літеру – $L_1=0{,}645$ біт/літера.

Кодування блоками з трьох знаків (триграмами) дає ще більший ефект.

Блоки	Ймовірність	Кодова комбінація
$z_1\ z_1\ z_1$	0,729	1
$z_2\ z_1\ z_1$	0,081	011
$z_1\ z_2\ z_1$	0,081	010
$z_1\ z_1\ z_2$	0,081	001
$z_2\ z_2\ z_1$	0,009	00011
$z_2\ z_1\ z_2$	0,009	00010
$z_1\ z_2\ z_2$	0,009	00001
$z_2\ z_2\ z_2$	0,001	00000

У цьому випадку середня довжина коду для триграми та одного символу така: $L_3=1{,}59$ біт/блок; $L_1=(1{,}59)/3=0{,}53$ біт/літера. Отже, середня довжина коду (на символ вихідного алфавіту) значно менша 1.

Зауваження 4.1. Про префіксність коду Шеннона–Фано

Кодування методом Шеннона–Фано можна графічно зобразити за допомогою бінарного дерева. Вихідному алфавіту відповідає коренева вершина дерева. Поділ на групи породжує дві дочірні вершини: ліва відповідає верхній групі, а права – нижній. Лівому ребру приписуємо символ "1", а правому – "0". Процес продовжуємо доти, доки у кожній підгрупі не залишиться по одному символу. Шлях від кореня до вершини, яка відповідає цьому символу, дає його код. Отже, код Шеннона–Фано є префіксним.

Бінарне дерево для прикладу 4.2 зображено на рис. 4.2.

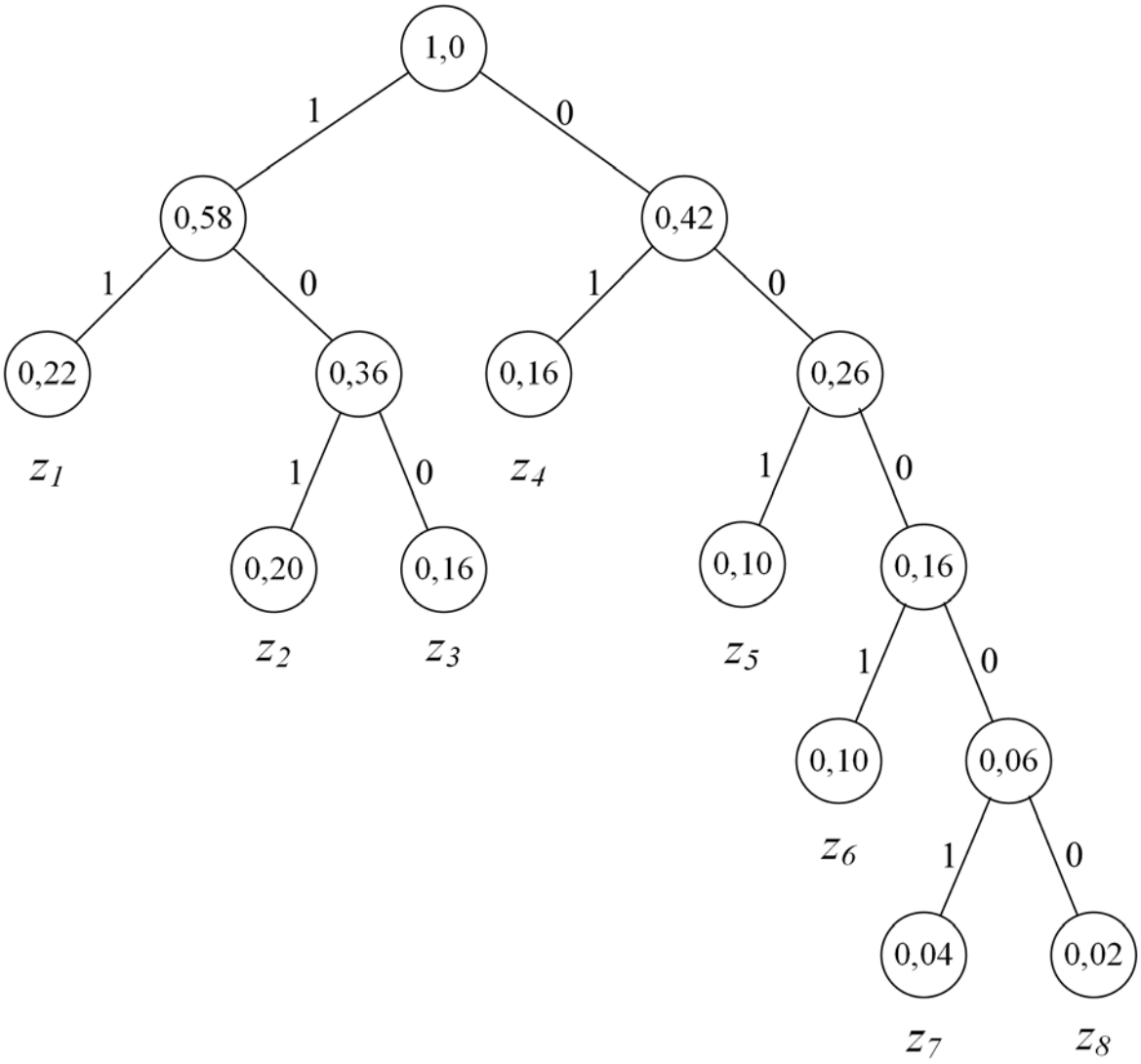


Рис. 4.2. Бінарне дерево для коду Шеннона–Фано (приклад 4.2)

4.4. Алгоритм оптимального кодування Гаффмана

Алгоритм Д. Гаффмана (*D. Huffman*) гарантує однозначну побудову двійкового коду з найменшою середньою довжиною коду [19].

Алгоритм складається з двох етапів.

Перший етап (побудова таблиці) має такі кроки:

1. Перший стовпчик таблиці відводять для кодів літер.

2. Літери алфавіту повідомлень виписують у другий стовпчик у порядку спадання їхніх ймовірностей.

3. Дві останні літери стовпчика об’єднують в одну допоміжну (умовну) літеру, якій приписують сумарну ймовірність.

4. У наступному стовпчику виписують літери, які не брали участі в об’єднанні, та утворену допоміжну літеру в порядку спадання їхніх ймовірностей (можна переписувати лише ймовірності).

5. Процес повторюють, доки не отримують одну допоміжну літеру з ймовірністю один.

Другий етап – побудова кодового дерева.

1. Кодове дерево є бінарним деревом. Вузли дерева відповідають утвореним умовним символам та символам вихідного алфавіту. У вузлах записують ймовірності цих символів.

2. Останній умовний символ з імовірністю 1 є кореневою вершиною дерева. Від неї відходять два ребра до вузлів з відповідними ймовірностями. Ребру, яке йде до вузла з більшою ймовірністю, приписують символ "1", а з меншою – "0".

3. Галуження продовжують доти, доки не дійдуть до окремих літер вихідного алфавіту. Шлях від кореня до окремої літери дає двійковий код літери.

Отож код Гаффмана є префіксним, оскільки його будують за допомогою бінарного дерева.

Приклад 4.4. Розглянемо джерело Бернуллі з алфавітом з восьми символів. Вихідні дані та етапи перетворення відображено нижче у таблиці. Умовні літери та відповідні ймовірності позначено жирним шрифтом. Відповідне бінарне дерево відображено на рис. 4.3.

Код	Ймовірність нової літери після перетворення							
	0	1	2	3	4	5	6	7
01	$z_1 - 0,22$	$z_1 - 0,22$	$z_1 - 0,22$	**$U_5 - 0,26$**	**$U_4 - 0,32$**	**$U_3 - 0,42$**	**$U_2 - 0,58$**	**$U_1 - 1,00$**
00	$z_2 - 0,20$	$z_2 - 0,20$	$z_2 - 0,20$	$z_1 - 0,22$	**$U_5 - 0,26$**	**$U_4 - 0,32$**	**$U_3 - 0,42$**	
111	$z_3 - 0,16$	$z_3 - 0,16$	$z_3 - 0,16$	$z_2 - 0,20$	$z_1 - 0,22$	**$U_5 - 0,26$**		
110	$z_4 - 0,16$	$z_4 - 0,16$	$z_4 - 0,16$	$z_3 - 0,16$	$z_2 - 0,20$			
100	$z_5 - 0,10$	$z_5 - 0,10$	$z_5 - 0,10$	$z_4 - 0,16$				
1011	$z_6 - 0,10$	$z_6 - 0,10$	**$U_6 - 0,16$**					
10101	$z_7 - 0,04$	**$U_7 - 0,06$**						
10100	$z_8 - 0,02$							

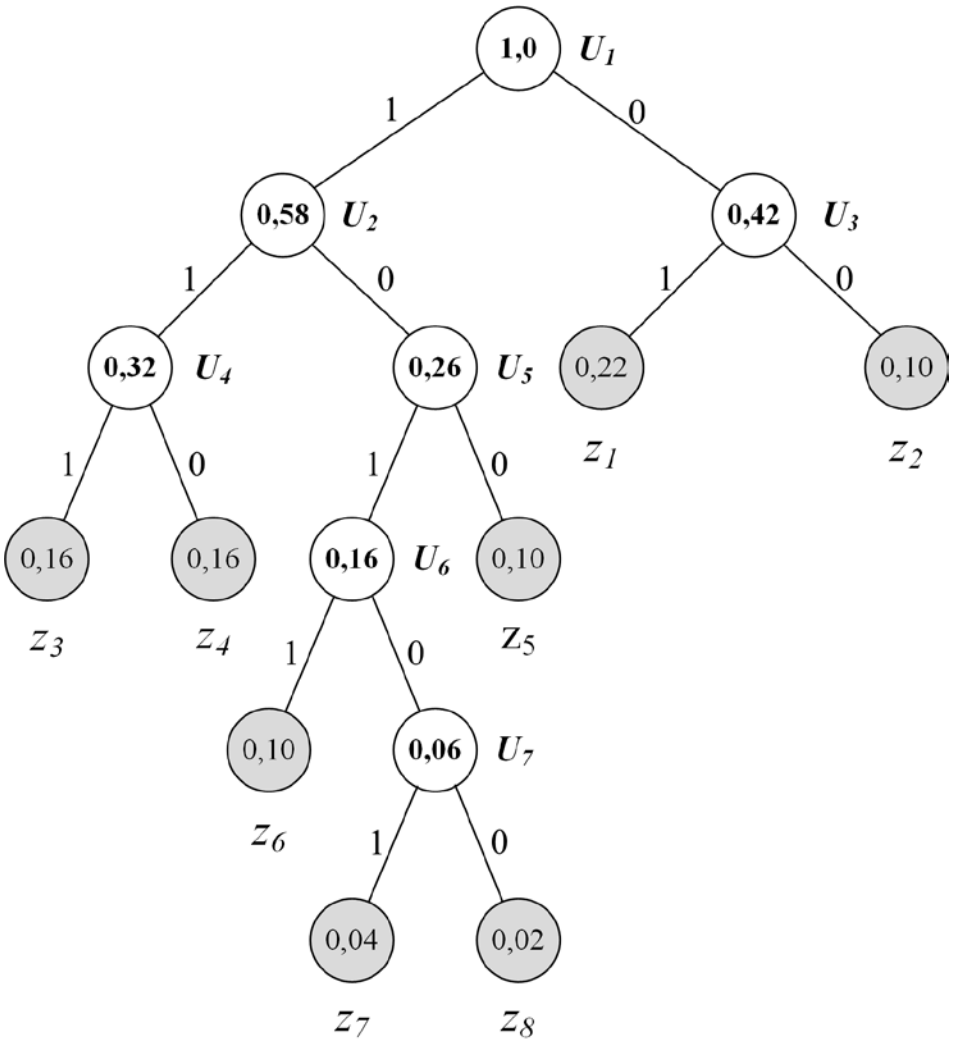


Рис. 4.3. Бінарне дерево алгоритму Гаффмана (приклад 4.3)

Ентропія вихідного ансамблю дорівнює $H = 2{,}75$ біт/літера, середня довжина коду – $L = 2{,}84$ біт/літера. Ефекту досягають завдяки присвоєнню коротких кодових комбінацій літерам з більшою ймовірністю і довших кодових комбінацій – літерам з меншою ймовірністю.

Розглянуті алгоритми ефективного кодування мають певні недоліки:

1. Різна довжина кодових комбінацій. Кодовий пристрій за однакові проміжки часу дає комбінації різної довжини. Виникає потреба у буферних пристроях.

2. Кодування довгими блоками спричиняє затримки.

3. Одинична похибка може перевести кодову комбінацію в іншу, що відрізнятиметься за довжиною. Це спричинить неправильне декодування наступних комбінацій – так званий трек похибки.

4.5. Арифметичне кодування

Арифметичне кодування не є блоковим. Це кодування перетворює повідомлення у деякий двійковий дріб з півінтервалу [0,1), який неможливо розбити на коди окремих символів чи блоків. Таке кодування є оптимальним і досягає теоретичної границі стиснення [19].

Опишемо коротко процедуру кодування. Розглянемо дискретне ймовірнісне джерело з ансамблем:

$$S = \left\{ z_i \to p_i,\ i = \overline{1,n};\ \sum_{i=1}^{n} p_i = 1 \right\}.$$

Кожен символ алфавіту $A = \{z_1, z_2, \ldots, z_n\}$ представимо півінтервалом з проміжку $[0,1)$, довжина якого дорівнює ймовірності появи цього символу, а початок відповідає кінцю півінтервалу попереднього символу алфавіту:

$z_i \to [x_i, y_i) \subset [0,1),\ i = \overline{1,n};$

$x_1 = 0,\ y_1 = p_1,\ x_i = y_{i-1},\ y_i = x_i + p_1,\ i = \overline{2,n}.$

Кодування здійснюють за таким алгоритмом:

1. Першому символу повідомлення відповідає півінтервал відповідно до його місця в алфавіті.

2. Кожен новий символ виділяє з поточного півінтервалу новий півінтервал пропорційно до свого положення в алфавіті та довжини.

3. Кодом повідомлення є півінтервал, виділений після обробки його останнього символу, точніше – число найменшої довжини, яке входить у цей півінтервал.

Кодування послідовності *CBA* наведено на рис. 4.4.

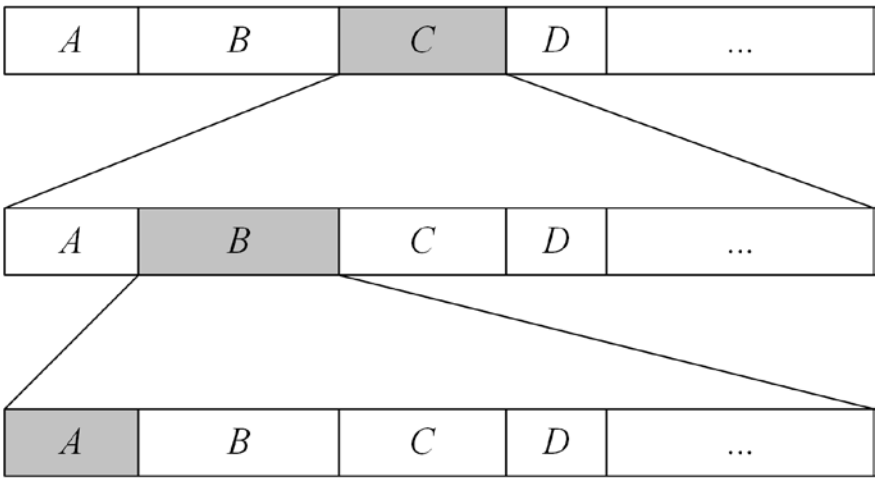


Рис. 4.4. Схема арифметичного кодування послідовності *CBA*

Нехай вихідне повідомлення має m символів – $z_{i_1}, z_{i_2}, \ldots, z_{i_m}$, $z_{i_j} \in A$, $\overline{j=1,n}$.

Позначимо півінтервал, який отримуємо на k-му кроці, через $z_{i_1}, z_{i_2}, \ldots, z_{i_k} \to [a_k, b_k)$. Отримаємо такі рекурентні формули:

$$[a_1, b_1) = [x_{i_1}, y_{i_1});$$

$$\left[a_k, b_k\right) = \left[a_{k-1} + (b_{k-1} - a_{k-1}) x_{i_k},\ a_{k-1} + (b_{k-1} - a_{k-1}) y_{i_k} \right),\ k = \overline{2,m}.$$

Приклад 4.5. Нехай алфавіт складається з двох символів "а" та "б", відповідно, з ймовірностями 3/4 та 1/4. Базовий поділ – $[0,1) = [0,3/4) \cup [3/4,1)$. Побудову коду для послідовності "ааба" наведемо нижче.

Крок	Послідовність	Півінтервал
0	""	[0, 1) = [0, 1)
1	"а"	[0, 3/4) = [0, 0.11)
2	"аа"	[0, 9/16) = [0, 0.1001)
3	"ааб"	[27/64, 36/64) = [0.011011, 0.100100)
4	"ааба"	[108/256, 135/256) = [0.01101100, 0.10000111)

Отож кодом є півінтервал [0.01101100, 0.10000111), однак з нього обирають число з найкоротшим представленням – 0.1 (код – 1).

Декодування здійснюють у зворотному порядку.

Головні недоліки розглянутої схеми арифметичного кодування:

1) необхідна двійкова арифметика з дуже великою кількістю розрядів;
2) результат кодування відомий лише після надходження останнього символу повідомлення.

Існують модифікації алгоритму, які усувають зазначені недоліки.

Запитання та завдання

1. Як побудувати префіксний код за допомогою бінарного дерева?
2. Сформулюйте теорему про множину кодів листків бінарного дерева.
3. Сформулюйте теорему про нерівність Крафта.
4. Що таке середня довжина коду?
5. Що таке надлишковість коду?
6. Сформулюйте теорему Шеннона про ефективне кодування для джерела Бернуллі.
7. Опишіть алгоритм оптимального кодування Шеннона–Фано.
8. Доведіть, що код Шеннона–Фано є префіксним.
9. Опишіть алгоритм оптимального кодування Гаффмана.
10. Опишіть алгоритм арифметичного кодування.

Розділ 5. МЕТОДИ СТИСНЕННЯ ДАНИХ БЕЗ ВТРАТ ІНФОРМАЦІЇ

5.1. Стиснення даних

Стиснення даних – це зменшення об'єму даних шляхом перекодування.

Ми розглядаємо *синтаксичне стиснення даних*, а не *логічне стиснення даних*, яке можливе на семантичному та прагматичному рівнях. Можна викинути всі голосні літери або кожну третю літеру, чи відкинути закінчення слів у тексті, проте зміст тексту буде зрозумілим. Це приклад стиснення даних на семантичному рівні.

Ступінь стиснення вимірюють величинами

$$R = \frac{l}{l_0} \text{ або } r = 1 - R = \frac{l_0 - l}{l_0}, \tag{5.1}$$

де l_0, l – кількість даних, відповідно, до і після стиснення.

Окрім ступеня стиснення, важливими є також інші характеристики методу стиснення:

- складність алгоритму кодування та декодування (кількість операцій, час, пам'ять);
- втрати інформації після декодування;
- універсальність алгоритму;
- стійкість до помилок та ін.

За ступенем відновлення початкових даних методи стиснення поділяють на методи з повним відновленням – без втрат інформації (*lossless*) та методи з частковим відновленням – з втратами інформації (*loose*). Останні знаходять широке застосування для кодування звукових, відео- та графічних даних (певною мірою це семантичні методи стиснення, які розглянемо у розділах 7 та 8).

За сферою застосування розрізняють *універсальні* та *спеціальні* методи стиснення. Спеціальні методи стиснення створені для обробки даних лише певних типів (наприклад, для обробки чорно-білих зображень факсимільного зв'язку – кодування *CCITT – The International Telegraph and Telephone Consultative Committee*). Універсальні методи стиснення призначені для стиснення довільних даних і не використовують жодної апріорної інформації про них.

Для ймовірнісної моделі джерела інформації можливості оптимального кодування визначаються теорією К. Шеннона, а алгоритмом може бути, наприклад, алгоритм Гаффмана або арифметичного кодування. Однак теорія К. Шеннона побудована на ідеалізованих припущеннях про апріорний статистичний опис джерела інформації та нескінченну довжину повідомлення.

Для поширених природних мов оптимальне політерне кодування за алгоритмом Гаффмана даватиме середню довжину коду приблизно 6 бітів, що, порівняно з однобайтовим кодуванням (*ASCII, Win1251*), дає ефект лише на 25 %. Кодування блоками для скінченних повідомлень є доволі громіздким.

Сьогодні розроблено чимало різноманітних способів стиснення даних. Програми, призначені для стиснення даних, називають *кодерами*.

Часто стиснення – це лише один з етапів загального процесу обробки даних у системах телекомунікацій. Методи стиснення також широко використовують універсальні архіватори, системні утиліти для роботи з жорсткими та компакт-дисками, текстові редактори, програми для роботи зі звуковими, відео- та графічними даними тощо.

Для прикладу розглянемо один з найпростіших методів стиснення даних – метод *кодування довжин серій* (*RLE – run length encoding*), який ще називають *методом кодування груп*. Це адаптивний алгоритм стиску, який зменшує фізичний розмір груп однакових символів.

Групи однакових символів кодують 3-ма байтами – <*pref, numb, symb*>, де *pref* – унікальний код, який визначає початок групи, *numb* – кількість символів у групі, *symb* – код символу. Наприклад, кодом повідомлення "abccccddddbbcaaaa" буде рядок "ab#4c#4dbbc#4a".

Цей метод буде ефективним за наявності значної кількості чотирисимвольних і більше повторень. Це характерно для чорно-білих зображень і зображень 1 байт/піксель.

Можливі різні схеми групового кодування:

а) кодування вздовж осі *X* або *Y*, коли зображення має безліч горизонтальних або вертикальних ліній;

б) групове кодування, коли зображення складаються з елементів однакових розмірів (виду шахівниці).

5.2. Словникові методи стиснення

Стиснення даних з побудовою словника. Ці методи створюють словник у безпосередньому вигляді. Вони мають декілька етапів.

Спочатку вихідний текст деяким способом розбивають на групи символів – "слова" (як окремі слова, так і корені, префікси, суфікси, закінчення). Отриману множину слів вважають літерами нового алфавіту. Для цього алфавіту будують таблицю кодування – оптимальну чи рівномірну. Отриману таблицю називають словником. Стиснутий текст включає словник та закодований за його допомогою вхідний текст.

Під час декодування повідомлення відновлюють шляхом заміни кодів словами зі словника.

Передбачають два перегляди тексту – для створення словника і кодування.

Приклад 5.1. Нехай потрібно закодувати український (англійський) текст. Поділимо його на слова, використовуючи роздільники – пропуски та розділові знаки. Припустимо, що текст містить менше, ніж $2^{16} = 65\,536$ різних слів (наприклад, Новий англо-український і українсько-англійський словник [10] містить близько 60 000 слів). Отже, кожному слову відповідатиме двобайтовий код (16 біт). Оскільки середня довжина слова в українській мові дорівнює шість, то таке кодування дає суттєве стиснення тексту – приблизно втричі, порівняно з рівномірним однобайтовим кодуванням літер.

Очевидно, що за такого способу кодування виникають затрати на збереження словника, відносний вклад якого зменшується зі збільшенням вихідного тексту. З іншого боку, метод забезпечує повний словниковий аналіз тексту, що можна використати для пошуку.

Стиснення даних за допомогою адаптивного словника. Це методи стиснення даних словникового типу, які не вимагають збереження словника. Їхня історія розпочалася 1977 року, коли А. Лемпел і Я. Зів (A. Lempel, J. Ziv) створили компресор *LZ*, отож метод стиску було названо *LZ77*.

Існує багато модифікацій цього методу стиску – *LZ78, LZW*. Найпоширеніша схема *LZW* (Лемпела–Зіва–Велча). Алгоритми *LZ7*8 і *LZW* запатентовані у США 1984 р. та 1985 р. Багато фірм, що випускають програмне забезпечення, купило ліцензію на алгоритм *LZW* ("*Adobe"* – 1990 року для шрифтів *Post Script*, "*CompuServe"* – 1994 року для формату *GIF*, хоча створили *GIF* ще 1977 року).

Розглянемо принцип роботи алгоритму *LZ77*. Кодування здійснюють триплетами (групами з трьох елементів) виду <*dist, length, next symb*>, де *dist* – відстань до початку відповідника, *length* – довжина відповідника, *next symb* – наступний символ, за таким алгоритмом:

1. З поточної позиції тексту розглядають підрядки змінної довжини, для яких віднаходять максимально довгий відповідник у попередньому тексті.

2. Якщо такого відповідника не відшукали навіть для однієї літери, то записують триплет з двома нульовими елементами і власне літерою (<0,0,"літера">).

3. Якщо відповідник знайдено, то першим елементом триплета записують відстань у зворотному порядку до початку цього відповідника, другим – довжину відповідника, а третім – наступну за підрядком літеру.

Декодування здійснюють у зворотному порядку. Словник неявно присутній, але зберігати його не потрібно.

Приклад 5.2. Для простоти розглянемо текст, утворений з трилітерного алфавіту $A=\{a,b,c\}$: abbcaacaabbcab.

Закодована послідовність буде такою: <0,0,a>, <0,0,b>, <1,1,c>, <4,1,a>, <3,3,b>, <8,3,b>. Декодування відбувається звичайним шляхом.

Детальніше зі словниковими методами стиснення можна ознайомитися у працях [12; 13].

5.3. Архівація даних

Архівація даних – це каталогізація та стиснення даних з метою їхнього збереження або передавання. Архівацію даних у файлах здійснюють за допомогою спеціальних програм, які називають *архіваторами*.

Значного поширення набули методи архівування, які побудовані на методах стиснення типу *LZ: ACE, ARJ, CAB, RAR, ZIP, 7-ZIP*, та інші.

Сьогодні найпопулярнішими є програми-архіватори *WinRar, WinZip, 7-Zip, PowerArchiver* та інші [3]. Коротко опишемо ці програми.

WinRar (Євген Рошал) – відомий архіватор з власним форматом *RAR*. Підтримує метод неперервного стиску, шифрування, створення багатотомних архівів та архівів автономного розпакування (*SFX*-архівів) тощо. Окрім формату *RAR*, повністю підтримує формат *ZIP*, може розпаковувати багато відомих сьогодні архівів.

WinZip ("*Corel Corporation*") – має зручний інтерфейс, може автоматично розпізнавати типи файлів, підтримує *drag-and-drop* та інтегрується в контекстне меню. Підтримує архіви типу *ZIP, TAR, UUencoded, XXencoded, BinHex, MIME*.

7-Zip (Ігор Павлов) – архіватор з високим ступенем стиску. Підтримує *ZIP*, *7z*, *RAR*, *CAB, GZIP, BZIP2* і *TAR* архіви. Компресія на 2–10 % більша, ніж у *PKZip* та *WinZip*, висока швидкість.

PowerArchiver **("***ConeXware Inc"***)** – універсальний потужний архіватор з підтримкою всіх популярних форматів: *ZIP*, *RAR*, *7-ZIP*, *CAB, LHA (LZH)* , *TAR, TAR, GZ, TAR, BZ2, BH*, *ARJ*, *ARC*, *ACE*, *ZOO*, *GZ*, *BZIP2*, *XXE*, *MIME*, *UUE*, *XPI*, *EAR*, *WAR*, *REP*, *JAR*, *BK, QWK*, *PK3*, а також *CD*-образів *ISO*, *BIN*, *IMG* та *NRG*. Підтримує *SFX*-архіви, шифрування за стандартом *AES*, інтегрується у контекстне меню тощо.

Різновидом архіваторів є програми резервного копіювання (*backup copy*) – створення архівів цілих дисків разом з образом *операційної системи* (ОС), наприклад, *Acronis True Image*, *Hendy Backup* та інші. Служби резервного копіювання вбудовані також у сучасні ОС і доступні з панелі керування.

Існують спеціалізовані архіватори для стиснення звукових файлів. Наприклад, архіватор *SoundSlimmer*, призначений для стиснення файлів mp3, acc, mp4 та wav. Зазначимо також, що сучасні відео-, графічні та деякі текстові формати передбачають автоматичне стиснення файлів під час створення.

Запитання та завдання

1. Що таке стиснення даних?
2. Як вимірюють ступінь стиснення даних?
3. Дайте класифікацію методів стиснення даних.
4. Опишіть метод кодування довжин серій.
5. Опишіть словникові методи кодування.
6. Сформулюйте алгоритм стиснення даних за допомогою адаптивного словника.
7. Що таке архівування даних?
8. Назвіть найпоширеніші архіватори.

Розділ 6. МОДЕЛЮВАННЯ, КОДУВАННЯ ТА ВІДТВОРЕННЯ КОЛЬОРІВ

Представлення текстових і числових даних у комп'ютерах нами розглянуто у підрозділах 2.5 та 2.6. У найпростішому випадку кодування тексту здійснюють безпосередньо за допомогою таблиць кодування. Однак сучасні комп'ютери ефективно працюють з іншими видами даних – графічними, звуковими та відео.

Комп'ютерна графіка вивчає методи моделювання, кодування, обробки та відтворення реальних і віртуальних візуальних об'єктів.

Головними пристроями введення графічної інформації в комп'ютер є сканери і камери, а пристроями виведення – дисплеї та принтери. Інформаційна система повинна забезпечити таке моделювання, перетворення та відтворення графічної інформації, щоб кінцеве зображення відповідало за сприйняттям людини вихідному зображенню.

6.1. Світлові величини

Людина бачить світло, яке випромінюється або відбивається від оточуючих предметів. Світло – видима людиною частина спектра електромагнітних коливань з довжиною хвилі $\lambda = 380-780$ нм (0,38–0,78 мкм).

Світло може бути *монохроматичним* (електромагнітні коливання однієї частоти), мати дискретний або неперервний спектр.

Нехай Φ – *потік електромагнітного випромінювання* через поверхню S – середня кількість променевої енергії, яка проходить через цю поверхню за одиницю часу. *Спектральною густиною потоку випромінювання* називають величину

$$\Phi_\lambda(\lambda) = \lim_{\Delta\lambda \to 0} \frac{\Phi[\lambda, \lambda + \Delta\lambda]}{\Delta\lambda}, \qquad (6.1)$$

де $\Phi[\lambda, \lambda + \Delta\lambda]$ – потік, який припадає на смугу частот $[\lambda, \lambda + \Delta\lambda]$.

Світловий потік Φ_V – це світлова величина, яка оцінює потік випромінювання за викликаними ним світловими відчуттями людини:

$$\Phi_V = K_{\max}\int_0^{\infty}\Phi_\lambda(\lambda)V(\lambda)d\lambda = \int_0^{\infty}\Phi_\lambda(\lambda)K(\lambda)d\lambda, \qquad (6.2)$$

де $K(\lambda)$ – *спектральна світлова ефективність випромінювання*; $K_{\max} = 683$ лм/Вт – максимальне значення спектральної світлової ефективності (досягається при $\lambda \approx 555$ нм); $V(\lambda) = K(\lambda)/K_{\max}$ – *відносна спектральна світлова ефективність* (рис. 6.1). Світловий потік вимірюють у *люменах* (лм).

Величину світлового потоку на одиницю тілесного кута у заданому напрямку називають *силою світла*:

$$I = \frac{d\Phi_V}{d\Omega}. \qquad (6.3)$$

Одиниця вимірювання сили світла – *кандела* (кд), або *свічка*. Це головна одиниця системи *SI*, а одиниця світлового потоку – люмен – є похідною одиницею. З останньої формули знаходимо

$$1 \text{ лм} = 1 \text{ кд·ср},$$

де ср – позначення *стерадіана* – одиниці вимірювання тілесних кутів.

Повна сфера утворює тілесний кут, рівний 4π стерадіан.

Якщо маємо *ізотропне джерело* (з однаковою силою світла для всіх напрямків) зі світловим потоком Φ_V, то його сила світла дорівнюватиме $I = \Phi_V/4\pi$.

Освітленість – це відношення світлового потоку $d\Phi_V$ до площі поверхні dS, на яку він падає:

$$E = \frac{d\Phi_V}{dS}. \qquad (6.4)$$

Одиниця освітленості – *люкс* (лк): 1 лк = 1 кд/м^2.

Освітленість поверхні точковим джерелом світла залежить від її відстані до джерела r та кута ϑ між нормаллю і напрямком на джерело:

$$E = \frac{I}{r^2}\cos\vartheta. \qquad (6.5)$$

Для характеристики здатності поверхні випромінювати (чи відбивати) світло вводять поняття *світності поверхні*. *Світність* рівна повному світловому потоку $d\Phi_V$, який випромінює одиниця площі поверхні dA:

$$R = \frac{d\Phi_V}{dA}. \tag{6.6}$$

За відбивання світла світність виражають через спектральну густину освітленості E_λ і коефіцієнт поглинання $k(\lambda)$

$$R = \int_{380}^{780} E_\lambda k(\lambda) d\lambda. \tag{6.7}$$

Мірою випромінювання в заданому напрямку (під кутом ϑ до нормалі) є *яскравість*:

$$L = \frac{dR}{\cos\vartheta d\Omega}. \tag{6.8}$$

Поверхню, яскравість якої не залежить від напрямку, називають *ідеально розсіюючою*. У цьому випадку

$$L = \frac{dR}{\cos\vartheta d\Omega} = \text{const}. \tag{6.9}$$

Розглянемо півсферу, в центрі якої розміщена випромінююча площадка dA. Приросту $d\vartheta$ відповідає тілесний кут $d\Omega = 2\pi \sin\vartheta d\vartheta$. Тоді:

$$R = \iint_S \frac{dR}{d\Omega} d\Omega = \pi L. \tag{6.10}$$

Наголосимо, що всі світлові величини пропорційні світловому потоку, який визначається світловою потужністю джерела.

6.2. Основи колориметрії

Вимірюванням і кількісним описом кольорів займається спеціальна наука – *колориметрія*.

Колірна модель – це набір правил, які встановлюють відповідність між кольорами та їхнім цифровим представленням. Складність побудови колірних моделей виражається суб'єктивністю поняття кольору – колір не існує поза нашою свідомістю.

Усі колірні моделі базуються на людському сприйнятті кольору. Щодо цього визначальними є простота кодування кольору та його відтворення за кодом, адекватність вихідного кольору та відтвореного.

Монохроматичне випромінювання для кожної довжини хвилі у сприйнятті людини має свій унікальний колір. Такі кольори називають *хроматичними*.

Випромінювання зі сталою спектральною густиною у видимому діапазоні частот людське око сприймає як біле світло. Таке світло називають *ахроматичним*. Ахроматичне джерело світла сприймається білим, а відбите чи переломлене ахроматичне світло (коли коефіцієнт поглинання не залежить від довжини хвилі) – білим, сірим чи чорним. Білими виглядають об’єкти, які відбивають понад 80 % світла білого джерела, а чорними – менше 3 %. *Екраном* називають ідеально розсіюючу поверхню білого кольору.

У світлі людське око здатне розрізняти два види інформації: колір і яскравість. Колір залежить від спектрального складу світла, яскравість визначається сумарною силою світла одиниці площі випромінюючої поверхні.

Для встановлення спектра конкретного світлового пучка використовують призму. Базові кольори спектра в бік зменшення довжини хвилі такі: червоний, оранжевий, жовтий, зелений, голубий, синій, фіолетовий.

Сітківка людського ока покрита світлочутливими елементами – *колбочками* (*R*, *G*, *B* видів) та *паличками*. Палички відповідають за зір при низьких значеннях яскравості і розрізняють світло за яскравістю. Максимальна чутливість *R*-колбочок припадає на червоний діапазон видимого спектра (565 нм), *G*-колбочок – на зелений діапазон (540 нм), а *B*-колбочок – на синій діапазон (440 нм). Графіки відносної спектральної чутливості колбочок та ока загалом подано на рис. 6.1.

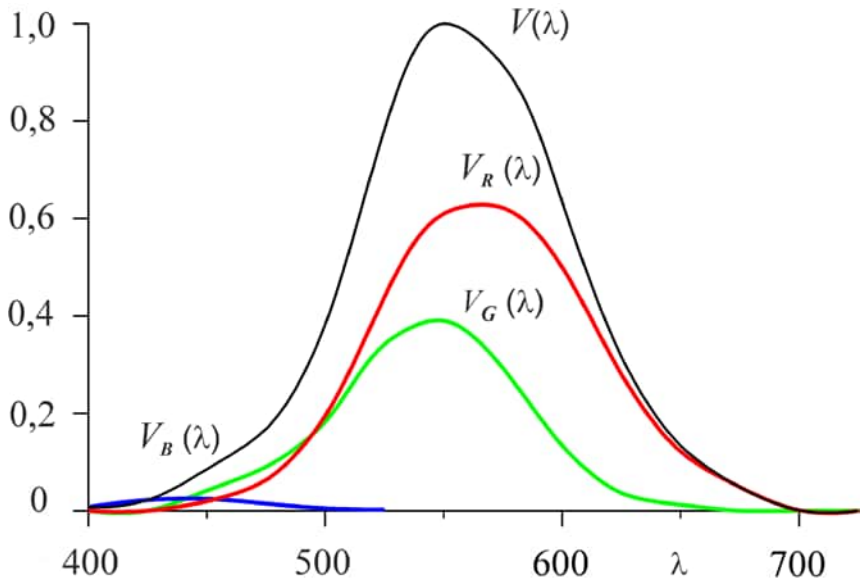


Рис. 6.1. Відносна спектральна чутливість *R*-, *G*-, *B*-колбочок

Ще в середині XIX століття німецький математик Герман Грассман (*Hermann Grassmann*) сформулював чотири закони змішування кольорів:

1) колір визначається трьома величинами;
2) колір змінюється неперервно за неперервної зміни кількості складових;

3) колір не залежить від порядку змішування;
4) додавання кольорів має адитивний характер.

Експериментальні дослідження зі змішування кольорів виконували різні дослідники ще з другої половини XIX століття. Найуспішнішими виявилися досліди, які провели Девід Райт (*David Wright*) та незалежно від нього – Джон Гілд (*John Guild*).

Експериментальна установка складається з білого екрана, розділеного перпендикулярною чорною перегородкою на дві частини (рис. 6.2). На правій частині екрана змішується світло базових монохромних джерел – червоного ($\lambda_R = 700$ нм), зеленого ($\lambda_G = 546{,}1$ нм) та синього ($\lambda_B = 435{,}8$ нм); ліву частину освітлює джерело, колір якого досліджують. Суть експерименту полягає в підборі інтенсивностей r, g, b базових джерел так, щоб кольори суміші і колір досліджуваного джерела співпадали, що формально можна записати як векторну рівність:

$$\boldsymbol{C} = r\boldsymbol{R} + g\boldsymbol{G} + b\boldsymbol{B}. \qquad (6.11)$$

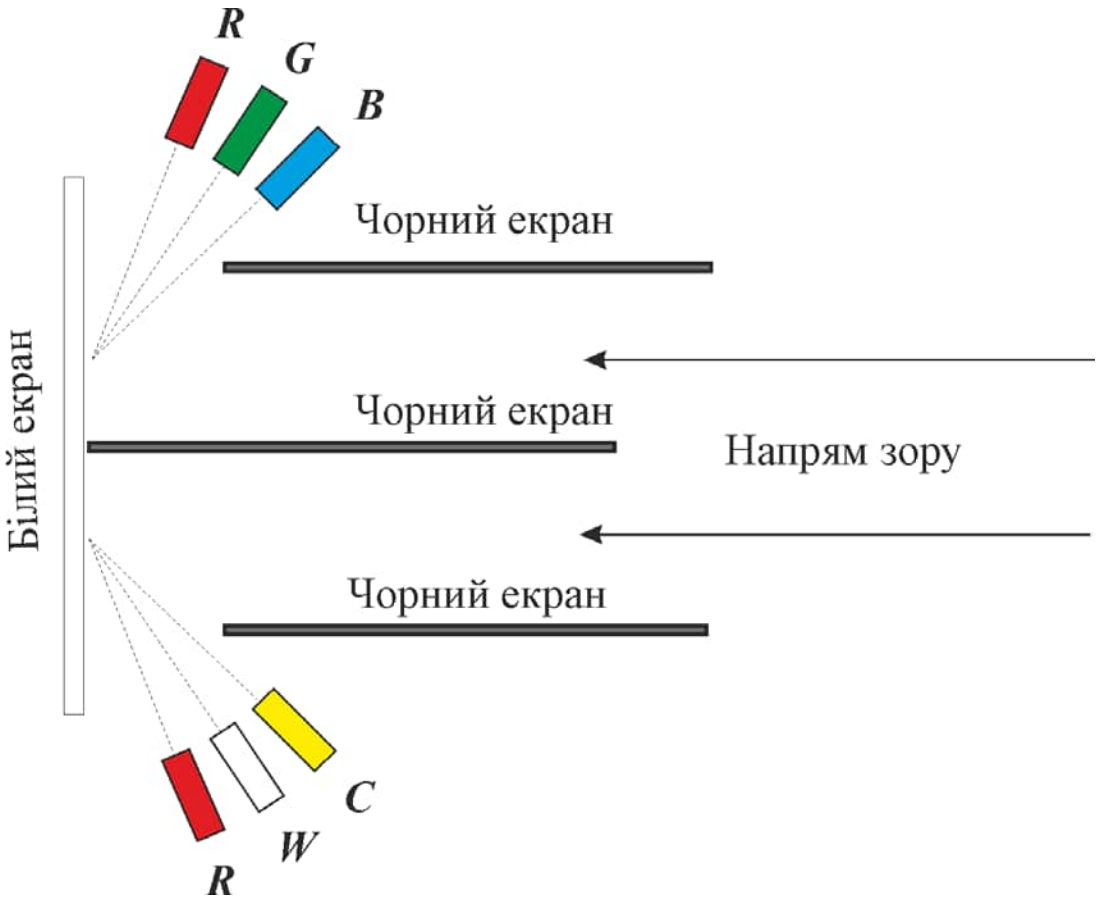


Рис. 6.2. Схема експерименту для дослідження змішування кольорів

Сформулюємо головні результати цих досліджень:

1. Дослідження кольору $\boldsymbol{C}(\lambda)$ монохроматичних джерел з різною довжиною хвилі дозволило отримати вагові функції $\hat{r}(\lambda)$, $\hat{g}(\lambda)$, $\hat{b}(\lambda)$ сумування базових кольорів $\boldsymbol{C}(\lambda)=\hat{r}(\lambda)\boldsymbol{R}+\hat{g}(\lambda)\boldsymbol{G}+\hat{b}(\lambda)\boldsymbol{B}$. Графіки цих функцій зображено на рис. 6.3. Вони засвідчують, що існує область зелено-голубих кольорів спектра, які не відтворює адитивна *RGB* модель, оскільки там потрібно вилучати червоний колір, тобто $\boldsymbol{C}(\lambda_1)-\hat{r}(\lambda_1)\boldsymbol{R}=\hat{g}(\lambda_1)\boldsymbol{G}+\hat{b}(\lambda_1)\boldsymbol{B}$, $\hat{r}(\lambda_1)<0$.

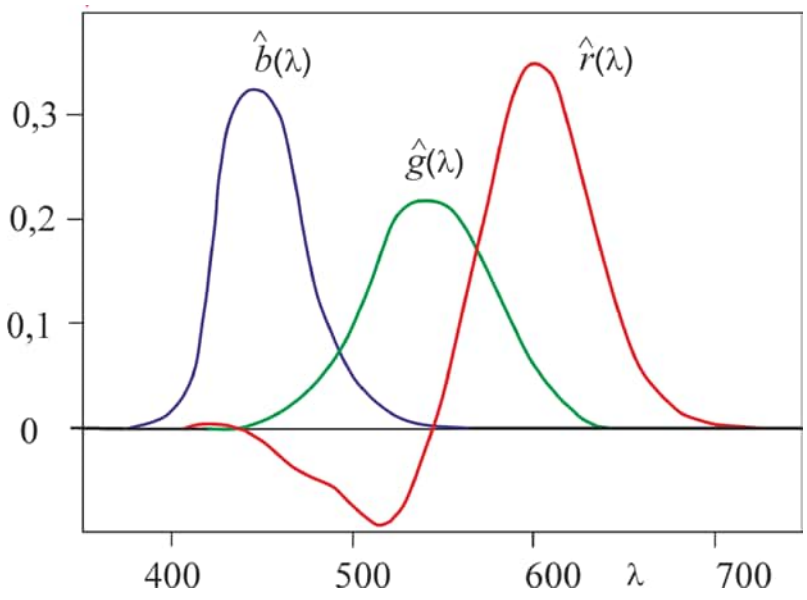


Рис. 6.3. Вагові функції сумування базових кольорів

2. Підтвердився, здебільшого, адитивний характер формування кольору. З рівностей:

$\boldsymbol{C}(\lambda_1)=\hat{r}(\lambda_1)\boldsymbol{R}+\hat{g}(\lambda_1)\boldsymbol{G}+\hat{b}(\lambda_1)\boldsymbol{B}$, $\boldsymbol{C}(\lambda_2)=\hat{r}(\lambda_2)\boldsymbol{R}+\hat{g}(\lambda_2)\boldsymbol{G}+\hat{b}(\lambda_2)\boldsymbol{B}$

випливає, що колір суміші визначається через базові кольори адитивно, тобто

$$\boldsymbol{C}(\lambda_1)+\boldsymbol{C}(\lambda_2)=\left(\hat{r}(\lambda_1)+\hat{r}(\lambda_2)\right)\boldsymbol{R}+\left(\hat{g}(\lambda_1)+\hat{g}(\lambda_2)\right)\boldsymbol{G}+\left(\hat{b}(\lambda_1)+\hat{b}(\lambda_2)\right)\boldsymbol{B}.$$

В інтегральній формі це дає змогу записати формулу для кольору джерела з неперервним спектром L_λ:

$$\boldsymbol{C}(L_\lambda)=r_E\,\boldsymbol{R}+g_E\boldsymbol{G}+b_E\boldsymbol{B}, \tag{6.12}$$

де $r_E=\int_{380}^{780} L_\lambda(\lambda)\hat{r}(\lambda)d\lambda$, $g_E=\int_{380}^{780} L_\lambda(\lambda)\hat{g}(\lambda)d\lambda$, $g_B=\int_{380}^{780} L_\lambda(\lambda)\hat{b}(\lambda)d\lambda$.

Зазначимо, що пропорційне збільшення інтенсивностей базових кольорів збільшує яскравість, але не змінює колір. Тобто власне колір може бути означений лише двома величинами.

3. Відтінки білого отримують за змішування головних кольорів у рівних пропорціях, тобто

$$\boldsymbol{W = R + G + B}. \qquad (6.13)$$

Виявлені недоліки моделі *RGB* привели до пошуків інших моделей, однією з яких була модель *XYZ*, розроблена 1931 року *Міжнародною комісією з освітленості* (МКО). У цій моделі як базові кольори обрано деякі умовні кольори, вагові функції сумування для яких є невід'ємними у всьому діапазоні видимих хвиль. Колірний простір *XYZ* вміщує усі кольори, які сприймає людина.

Міжнародна комісія з освітлення, також відома як *CIE* – від її французької назви *Commission Internationale de l´Eclairage*, є авторитетною професійною організацією з усіх питань науки та мистецтва світла й освітлення, кольору та зору, фотобіології та технології зображення. Ця комісія визнана *ISO* як міжнародний орган стандартизації.

Під егідою *CIE* 1976 року на основі моделі *XYZ* розроблено колірний простір *CIELAB*, зміна кольору в якому близька до лінійної у людському сприйнятті. Цей простір використовує як параметри світлосилу, відношення зеленої складової кольору до червоної та відношення синьої складової до жовтої. Простір *CIELAB* зручний для корекції кольору, тому знайшов широке використання для обробки зображень як проміжний колірний простір, через який відбувається конвертування даних між іншими колірними просторами (наприклад, з *RGB* сканера до *CMYK* друкованого процесу).

6.3. Колірні моделі

Модель RGB. Колір формується адитивно – додаванням трьох базових кольорів різної інтенсивності: червоного – *R*, зеленого – *G* та голубого – *B*. Тобто колір задається трійкою (*r, g, b*) – вектором, координати якого змінюються у певних межах (наприклад, від 0 до 1, часто – цілі числа від 0 до 255).

Простір *RGB* кольорів можна уявити у вигляді одиничного куба в тривимірному просторі. Вершини куба відповідають чорному, червоному, зеленому, синьому, голубому, пурпуровому, жовтому та білому кольорам (рис. 6.4).

На моделі *RGB* ґрунтується принцип дії монітора. Одна точка екрана складається з трьох люмінофорів, які при дії на них електронного променя випромінюють червоне, зелене та синє світло. Колір точки створюється змішуванням трьох базових кольорів з відповідним співвідношенням інтенсивностей променів.

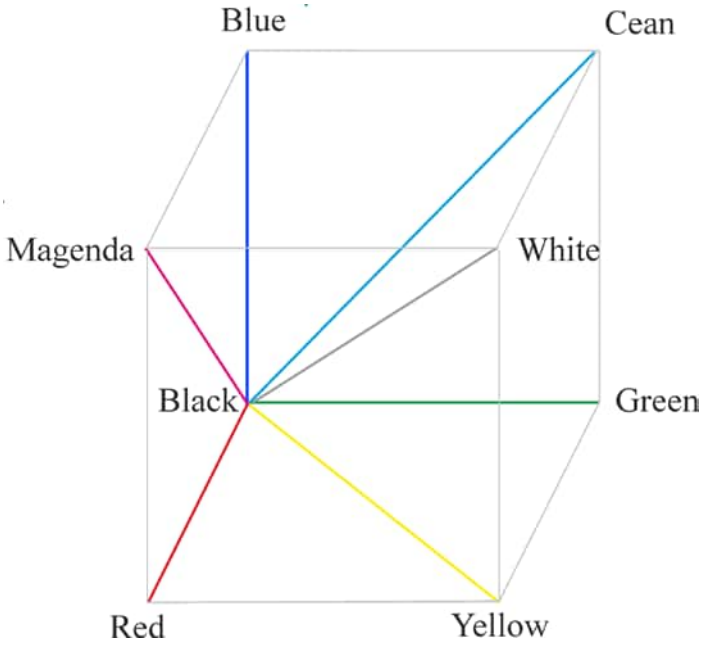


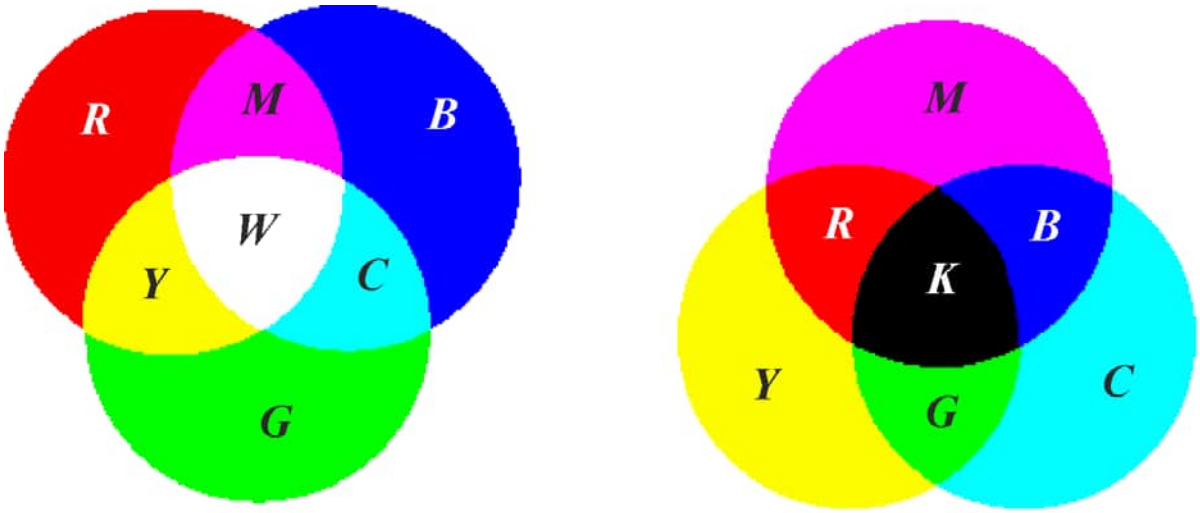
Рис. 6.4. Модель *RGB*

Кодування кольору в сканерах і цифрових камерах також ґрунтується на моделі *RGB*. Вихідний промінь за допомогою фільтрів розділяється на складові: червоний, зелений та синій кольори, інтенсивності яких фіксують світлочутливі елементи, об’єднані в лінійку (для сканера) або в матрицю (для камери).

У природі за моделлю *RGB* формується колір хамелеонів. Шкіра цих тварин вкрита клітинами-пухирцями базових кольорів. Зміна їхніх відносних розмірів приводить до зміни кольору шкіри.

Моделі CMY, CMYK. Розглянемо адитивне змішуванням трьох базових кольорів моделі *RGB*, наприклад, змішування на білому екрані променів від базових джерел (рис. 6.5, *а*). Змішування в рівних кількостях червоного і синього кольору дає пурпуровий колір (*magenta*), червоного і зеленого – жовтий (*yellow*), зеленого і синього – голубий (*cyan*), а змішування трьох базових кольорів – білий колір (*white*):

$$M = R + B,\ Y = R + G,\ C = B + G,\ W = R + G + B.$$



а *б*

Рис. 6.5. Змішування головних кольорів:
а – у адитивній моделі *RGB*; *б* – у субтрактивній моделі *CMY*

Але субтрактивне змішування світла при послідовному проходженні через світлові фільтри, наприклад, проходження відбитого світла при накладанні напівпрозорих фарб, відбувається по-іншому. Біле світло не пройде через послідовні фільтри $R \oplus B$, $R \oplus G$ або $G \oplus B$. Тобто при змішуванні двох напівпрозорих фарб головних кольорів *RGB* на поверхні білого паперу утворяться темні ділянки.

Тому в поліграфії використовують інші базові кольори – *C*, *M*, *Y* :

$C = W - R = B + G$; $M = W - G = B + R$; $Y = W - B = G + R$.

Адитивне змішування цих кольорів (на білому екрані від джерел) відбувається так:

$C + M = W + B = B$; $C + Y = W + G = G$; $M + Y = W + R = R$.

Відбиття білого світла при накладанні фарб відбувається так (див. рис. 6.5, *б*):

$C \oplus M = W - R - G = B$; $C \oplus Y = W - R - B = G$;

$M \oplus Y = W - G - B = R$; $C \oplus M \oplus Y = K$ (*blacK*).

Тобто випадкове змішування фарб не спричинить серйозних наслідків.

Чорний колір утворюється за допомогою трьох фарб – трохи неекономно і він не дуже "чорний". Тому до цієї моделі додають ще чорний колір, утворюючи модель *CMYK*.

Модель *CMYK* широко використовують у поліграфії, кольорових принтерах.

Колірні моделі YUV, YCbCr. Модель *RGB* не економна за зміни лише яскравості зображення. В телевізійних стандартах і деяких графічних форматах (наприклад, *JPEG*) використовують моделі, перша компонента яких визначає яскравість, а дві інші – колір. Ці моделі утворюються з *RGB* лінійним перетворенням.

У колірній моделі *YUV* компоненту яскравості *Y* (*luma*) обчислюють за формулою

$$Y = 0{,}299\ R + 0{,}587\ G + 0{,}114\ B, \tag{6.14}$$

а кольоророзницеві компоненти (*chromaticity*) визначають так:

$$U = R - Y\ ,\ V = B - Y\ . \tag{6.15}$$

Модель *YCbCr* пов'язана з моделлю *RGB* так:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0{,}299 & 0{,}587 & 0{,}114 \\ -0{,}169 & -0{,}331 & 0{,}500 \\ 0{,}500 & -0{,}419 & -0{,}081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}. \tag{6.16}$$

Обернене перетворення має вигляд:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1{,}402 \\ 1 & -0{,}344 & -0{,}714 \\ 1 & -0{,}344 & 0 \end{bmatrix} \left(\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} - \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \right). \tag{6.17}$$

Колірна модель HSB (HSV). Модель *HSB* оперує з кольорами на інтуїтивному рівні. Колір задають трьома величинами:

1) *h* (*hue*) – *колірний тон* (власне колір);
2) *s* (*saturation*) – *насиченість* (наскільки головний колір розбавлено білим);
3) *b* (*brightness*) або v (*value*) – яскравість.

Коди базових кольорів за цією моделлю подано нижче у таблиці 6.1.

6.4. Кодування кольорів

Під час кодування чорно-білих зображень (тонами сірого) застосовують градацію до 256 рівнів; для цього достатньо 8 бітів – 1 байт.

За кодування *RGB* та *CMY* для інтенсивності кожної зі складових максимально використовують 256 градацій, тобто на кодування кольору потрібно затратити 3 байти. Така система забезпечує кодування приблизно 16,5 млн різних кольорів, що близько до чутливості людського ока.

Режим представлення кольорової графіки з використанням 24-х розрядів для кольору називають *повноколірним* **(***True Color***)**. На практиці використовують також 16-бітне кодування кольорів – (1,5,5,5) або (5,6,5). Такий режим називають *High Color*.

При використанні для кодування кольору одного байта можна передати лише 256 кольорів. Зазвичай таке кодування є індексним. Код виражає номер кольору у деякій додатковій таблиці, яку називають *палітрою кольорів*. Палітру додають до графічних даних.

Таблиця 6.1. Кодування базових кольорів у повноколірних моделях

Колір	*RGB*			*CMY*			*HSB*		
Red	255	0	0	0	255	255	0	240	120
Yellow	255	255	0	0	0	255	40	240	120
Green	0	255	0	255	0	255	80	240	120
Cyan	0	255	255	255	0	0	120	240	120
Blue	0	0	255	255	255	0	160	240	120
Magenta	255	0	255	0	255	0	200	240	120
Black	0	0	0	255	255	255	160	0	0
White	255	255	255	0	0	0	160	0	240
Grey	63	63	63	191	191	191	160	0	59

6.5. Управління кольором

Зображення, від його введення в ІС до виведення, проходить складний шлях перетворень зі змінами колірних моделей. Це може спричинити невідповідність початкових і результуючих кольорів. Адекватність відтворення кольорів забезпечує система управління кольором (англ. *Color Management System, CMS*), яка охоплює операційну систему, прикладні програми і пристрої.

Колірний профіль конкретного пристрою показує, як цей пристрій сприймає або відображає стандартні кольори. Він є різним для різних пристроїв і змінюється з часом.

Нехай $\mathbf{c} \in R^n$ – вектор кольору з деякого колірного простору на вході пристрою, а $\tilde{\mathbf{c}} \in R^m$ – вектор кольору на його виході. Зв'язок між цими кольорами функціонально можна подати так:

$$\tilde{\mathbf{c}} = \mathbf{F}(\mathbf{c}), \tag{6.18}$$

де $\mathbf{F}: R^m \to R^m$ – відображення, яке визначає колірний профіль.

Якщо колірний профіль відомий, то можна забезпечити адекватне відтворення кольору пристроєм, здійснивши внутрішнє обернене перетворення:

$$\mathbf{c} = \mathbf{F}^{-1}(\tilde{\mathbf{c}}). \qquad (6.19)$$

Калібрування пристрою – це побудова його колірного профілю з подальшим налаштуванням правильного відтворення кольорів. Для сканера калібрування здійснюють скануванням еталону і використанням спеціальної програми, яка будує профіль і робить його активним. Для побудови профілю монітора використовують *колориметри* – прилади для вимірювання кольорів.

З метою розробки універсальної системи управління кольором, незалежної від операційної системи та прикладних програм, 1993 року створено Міжнародний консорціум по кольору (англ. *International Color Consortium, ICC*). Зусиллями консорціуму розроблено:

1) відкритий стандарт для модуля відповідності кольорів (*Color matching module, CMM*) на рівні ОС;

2) формат колірних профілів для пристроїв, зокрема прямого перетворення профілю одного пристрою на профіль іншого без використання перехідного простору кольорів;

3) робочі колірні простори для маніпулювання кольором.

Останні специфікації можна знайти на сайті консорціуму (https://www.color.org/icc_specs2.xalter).

За стандартом *ICC*, перетворення між двома колірними просторами може відбуватися через перехідний колірний простір (англ. *Profile Connection Space, PCS),* зазвичай *CIE XYZ* або *CIELAB*. Наприклад, перехід від колірної моделі *RGB* до моделі *CMYK* можна здійснити так: *RGB → CIELAB → CMYK*.

Розроблено інші системи управління кольором, зокрема *ColorSync* – для *macOS, Windows Color System (WCS*) – для *Windows* та інші.

Запитання та завдання

1. Опишіть предмет комп’ютерної графіки.
2. Назвіть головні пристрої введення та виведення графічної інформації.
3. Поясніть природу світла та кольору.
4. Що таке монохроматичне світло?
5. Що таке ахроматичне світло?
6. Опишіть предмет колориметрії.
7. Що розуміють під моделлю кольору?

8. Що таке біле світло?
9. Що таке білий екран?
10. У чому суть психофізіологічної природи кольору?
11. Сформулюйте головні закони змішування кольорів (закони Грассмана).
12. Назвіть базові кольори моделі *RGB*.
13. Назвіть базові кольори моделі *CMYK*.
14. Представлення кольорів у моделі *HSB* (*HSV*).
15. Яку модель кольорів використовують у телебаченні?
16. Назвіть графічні пристрої, що використовують моделі *RGB* та *CMYK*?
17. Що таке колірний профіль пристрою?
18. Як здійснюють колірне калібрування пристрою?

Розділ 7. ЦИФРОВІ ГРАФІЧНІ МОДЕЛІ ТА ФОРМАТИ

Графічні зображення – це світловий образ зовнішніх або абстрактних об’єктів, який відображає їхнє розміщення у просторі, колір та інші характеристики. Графічні зображення зовнішніх об’єктів отримують на деякій поверхні, найчастіше плоскій, шляхом пропускання світлових променів через фокусуючий елемент (отвір невеликого діаметра, лінзу), який розділяє світлові промені від різних об’єктів, створюючи образ, який відтворює взаємне розміщення об’єктів та їхні характеристики. Зображення фіксують за допомогою світлочутливих матеріалів або фотоелементів. Після цього його можна зберігати, обробляти та відтворювати різними способами.

Існує два принципово різні методи подання (моделювання) графічних об’єктів. Перший полягає у наближеному поданні графічного зображення за допомогою дискретних елементів зображення – *пікселів*. Одержані в такий спосіб зображення називають *растровими* від слова *растр* (горизонтальна лінія пікселів). Прикладами таких зображень є вишивка або мозаїка.

Другий спосіб полягає у поданні зображення сукупністю геометричних ліній і фігур – *графічних примітивів*, які описують математичними засобами. Подане цим способом зображення називають *векторним* або *об’єктним*. Об’єктні моделі найчастіше застосовують для внутрішнього представлення та обробки зображень, оскільки пристрої введення та виведення зображень зазвичай є растровими.

7.1. Растрові зображення

Найменші логічні елементи растрового зображення називають *логічними пікселями* (від англ. *pixel* – скорочення словосполучення *picture element*). Вони мають координати та колір, але їхні розміри є відносними і можуть не фіксуватися.

Фізичні пікселі – це найменші фізичні елементи зображення, які можна обробляти апаратним і програмним способом. Вони мають реальні розміри.

Представлення зображення безпосереднім відображенням логічних пікселів на фізичні пікселі можливе лише тоді, коли фізичний піксель може передавати колір. Найчастіше логічні пікселі відображаються на растрові точки або їхні групи (розетки).

Растрова точка – елемент зображення певної форми, зокрема квадратний, сформований з окремих фізичних пікселів для створення відтінків головного кольору. Густина фізичних елементів виражає яскравість основного кольору. Растрову точку також часто називають пікселем. Кількість растрових точок на одиницю довжини називають *лінеатурою* (вимірюється в lpi – *lines per inch*).

Роздільність. Якість (чіткість) растрового зображення визначається розміром окремого елемента зображення – пікселя чи растрової точки. Обернену величину до цього розміру називають *роздільністю*.

Роздільність реального растрового зображення – це кількість фізичних пікселів на одиницю довжини (1 дюйм, 1 міліметр), вимірюється в ppi (*points per inch*) або dpi (*dots per inch*), dpm (*dots per millimeter*). Зазначимо, що 100 dpi ≈ 4 dpm.

Розглянемо два окремі відрізки, розміщені на деякій відстані на невидимій прямій лінії. *Кутовою роздільною здатністю людського ока* називають мінімальну кутову відстань між цими відрізками, за якої вони не зливаються в один відрізок. Ця величина приблизно дорівнює одній мінуті ($1' \approx 0{,}3 \cdot 10^{-3}$ рад). Відповідно, *лінійна роздільна здатність ока* з відстані 30 см в середньому рівна 0,1 мм або 10 dpm, тобто 250–300 dpi.

Роздільна здатність растрових пристроїв введення–виведення – це роздільність зображень, яку вони забезпечують.

Офісні сканери зазвичай мають роздільну здатність від 600x600 dpi до 2400x2400 dpi. Вони розділяють вихідні зображення на пікселі і приписують кожному з них три компоненти кольору (найчастіше за моделлю *RGB*).

Принтери використовують колірну модель *CMYK*. Залежно від типу, вони забезпечують різну роздільність зображення: матричні – до 150 dpi, лазерні – 600–1200 dpi, струменеві – 300–1440 dpi.

Електронні дисплеї на основі *електронно-променевих трубок* (ЕПТ) створюють зображення на основі *RGB* моделі. Їхня роздільна здатність обмежена розміром зерна екрану. *Розмір зерна екрана* – характерний розмір трійки люмінофора – відстань між люмінофорами однакових кольорів. Типова величина – 0,18–0,33 мм. Аналогічно працюють *рідкокристалічні дисплеї*, які складаються з рідкокристалічних комірок, що відтворюють базові кольори. Типове геометричне розміщення колірних елементів на екрані дисплея подано на рис. 7.1.

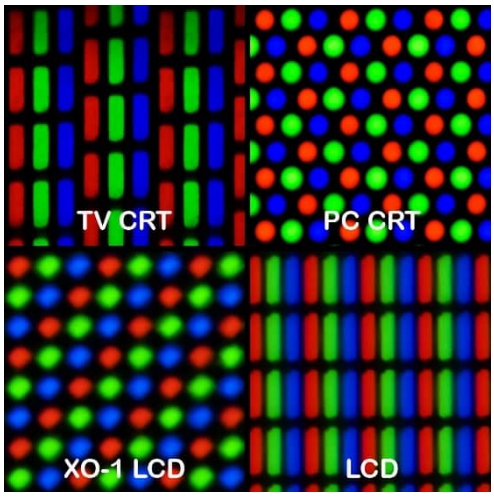


Рис. 7.1. Геометричне розміщення колірних елементів у дисплеях

Роздільність монітора визначається розгорткою (кількістю пікселів) по горизонталі та вертикалі, наприклад, 800x600, 1024x768, 1280x1024, 1600x1200, 1680x1050, 1920x1200 і більше. Сучасні монітори та відеокарти підтримують декілька відеорежимів з різною розгорткою.

Важливими характеристиками моніторів також є:

розмір діагоналі в дюймах (14", 15", 17", 19", 21", 24", 27’’);

формат – відношення ширини до висоти екрана (4:3, 16:10, 16:9 та ін.);

частота кадрів – число кадрів, яке монітор відображає за 1 секунду (для електронних дисплеїв рекомендовано не менше 70–75 Гц).

Виходячи з розміру монітора по діагоналі, розміру зерна та кутової роздільної здатності людського ока, легко встановити оптимальну розгортку.

У цифровій фотографії якість зображення визначають кількістю пікселів, яку вимірюють у мегапікселях, скорочено – Мп (Мрх, Мр).

Детальніше роботу пристроїв введення та виведення графічних даних описано у книгах [7; 15; 30].

Палітра кольорів. Статичне растрове зображення – це двовимірний масив кодів кольорів пікселів. Множину кольорів зображення називають *палітрою*.

Для растрового зображення кількість кольорів палітри k не перевищує розміру зображення у пікселях r і часто набагато менша розміру повноколірної палітри – 2^{24}.

Для кодування k кольорів потрібно не більше $m=\log_2 k+1$ біт. Для збереження r пікселів з колірною глибиною 24 біти потрібно щонайменше $24r$ біт, а з колірною глибиною m потрібно mr біт. Для збереження таблиці кольорів, яка встановлює відповідність між кольорами палітри та повноколірною моделлю, потрібно $24k$ біт. Отож відносний ефект від використання колірної палітри можна оцінити відношенням:

$$\frac{mr+24k}{24r}=\frac{m}{24}+\frac{k}{r}.$$

Побудову m-бітної палітри найчастіше здійснюють *методом медіанних перерізів*, який полягає у квантуванні колірного куба $(256\times256\times256)$ і має такі кроки:

1. Всі пікселі зображення відображають на колірний куб відповідно до вектора кольору пікселя.

2. Краї колірного куба, які не містять жодного пікселя, відсікають площинами, перпендикулярними до осей.

3. Здійснюють поділ отриманого паралелепіпеда на два паралелепіпеди площиною в середній точці найдовшої сторони.

4. Попередні два кроки повторюють для кожного з отриманих паралелепіпедів.

5. Продовжуючи процес поділу m разів, отримують $k=2^m$ паралелепіпедів.

6. Обчислюють геометричні центри паралелепіпедів (центри їхньої маси). Їхні координати дають $k=2^m$ кольорів m-бітної палітри.

Проблеми відтворення зображень під час друку. Під час друку зображень відтворення кольору відбувається не так, як на дисплеї.

Спочатку здійснюють *кольороподіл* – зображення за допомогою світлофільтрів розділяють на базові кольори моделі *CMYK* (або іншої моделі).

Для кожного кольору відбувається *растрування зображення* – поділ зображення на растрові точки – для передачі відтінків кольору. Для кожного кольору беруть різний кут нахилу растру: С –15^{o}, М – 75^{o}, Y – 0 ^{o}, В – 45 ^{o}, що мінімізує накладання кольорів під час друку. Друк здійснюють за допомогою чотирьох матриць, які наносять складову зображення відповідного кольору, точки растра базових кольорів групуються у розетки.

Переваги та недоліки растрових моделей зображень.

Наведемо основні переваги:

1. Простота технічної реалізації. Растрові моделі зображень добре пристосовані до пристроїв введення та виведення комп'ютера. Зокрема, лінійка планшетного сканера створює один ряд растра, матриця цифрової камери – один кадр зображення. Піксель монітора відтворює один логічний піксель зображення, а одна лінія розгортки – один ряд растра.
2. Растрові зображення мають прості формати зберігання.
3. Растрові зображення добре передають реальні зображення.

Основні недоліки растрових зображень такі:

1. Чутливі до геометричних перетворень зображення.
2. Файли для збереження растрових зображень мають значний об'єм.

7.2. Векторні моделі зображень

Векторні зображення складаються з кусково-гладких контурів, побудованих за допомогою параметричних кривих (кривих Без'є). Замкнуті контури можуть мати заливку різних типів (суцільну, градієнтну, з візерунком). Контури можуть мати обвідку (обрис) лінією заданої товщини та кольору.

Криві Без'є визначаються параметрично через координати базисних точок $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n$ за допомогою поліномів Бернштейна:

$$\mathbf{B}(t) = \sum_{k=0}^{n} \mathbf{P}_k b_{kn}(t) \ , \ \ b_{kn}(t) = C_n^k t^k (1-t)^{n-k} \ , \ 0 \le t \le 1, \ C_n^k = \frac{n!}{k!(n-k)!}.$$

Переваги:

1. Допускають геометричні перетворення без зміни якості зображення.
2. Допускають каскадні розміщення об'єктів з різними ефектами просвічування.
3. Малий об'єм файлів для збереження векторних зображень.

Недоліки:

1. Векторні зображення – нереалістичні.

2. Проблема автоматизованого введення векторних зображень, для цього існують спеціальні програми – трассери.

3. Векторні зображення потребують спеціальних програм для створення, обробки та збереження, файли для їхнього збереження – складні.

7.3. Алгоритми стиснення зображень

Викладемо основні алгоритми стиснення зображень на основі праць [20; 31].

Порівняно з текстом растрові зображення вимагають для збереження значно більшого об'єму пам'яті. Наприклад, повноколірна ілюстрація (глибина 24 біт/піксель) розміром 600x800 точок (близько 0,5 Мп) займає близько 1440000 Б – стільки, скільки ж 600 сторінок тексту (40 рядків по 60 знаків у рядку з пробілами на одній сторінці), закодованих кодом *ASCII*. Якщо ж сторінку А4 тексту відсканувати як 24-бітне зображення із роздільною здатністю 600 dpi, розмір скану у форматі *BMP* близько 265 МБ – приблизно 44 тис. сторінок тексту без форматування. Ця особливість растрових зображень визначає актуальність алгоритмів їхнього стиснення.

З іншого боку, людське око вирізняє в зображенні лише контури та переходи кольорів і не чутливе до деталей, що дає змогу використати спеціальні алгоритми стиску.

Насамперед зазначимо, що для стиску растрових зображень можна застосувати спеціалізовані алгоритми без втрат інформації. Зокрема, для стиску факсимільних зображень успішно застосовують *алгоритм CCITT*, запропонований групою зі стандартизації Міжнародного консультативного комітету з телеграфії та телефонії (*Consultative Committee International Telegraph and Telephone*). Цей алгоритм використовує варіант алгоритму *RLE* та алгоритм Гаффмана (див. підрозділ 5.1).

Однак найефективнішими для стиску зображень є спеціалізовані алгоритми з втратами інформації. Розглянемо деякі з них.

Алгоритм JPEG. *JPEG* [’jei’peg] – один з нових і найпоширеніших алгоритмів стиснення, який є стандартом де-факто для повноколірних зображень (стандартизований *ISO* 1991 року). Алгоритм розроблений групою експертів в області фотографії (*Joint Photographic Expert Group*) – підрозділу *ISO*.

Перелічимо кроки алгоритму.

Крок 1. Здійснюють перехід від колірної моделі *RGB* до моделі *YCbCr* та оптимізують палітру методом медіанних перерізів.

Крок 2. Вихідне зображення розбивають на блоки розміром 8x8 пікселів. Для кожного блоку формують три робочі матриці з байтовими елементами – окремо для кожної компоненти кольору. Для великих ступенів стиску здійснюють проріджування кольорів, суть якого полягає в усередненні хроматичних колірних компонент *Cb* та *Cr* для груп з чотирьох пікселів без зміни компонент яскравості *Y*. Це зменшує матриці для колірних компонент у чотири рази.

Крок 3. Двовимірне *дискретне косинус-перетворення* (ДКП) переводить двовимірний масив амплітуд f_{km}, $k,m=\overline{0,N-1}$ у відліки у спектральній області $\tilde{f}_{ij}$, $i,j=\overline{0,N-1}$. Формули прямого та оберненого дискретного косинус-перетворення такі:

$$\tilde{f}_{ij}=\sum_{k=0}^{N-1}\sum_{m=0}^{N-1} f_{km}\lambda_{ij}\cos\frac{i(2k+1)\pi}{2N}\cos\frac{j(2m+1)\pi}{2N},\ i,j=\overline{0,N-1}\ , \tag{7.1}$$

$$f_{km}=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \tilde{f}_{ij}\lambda_{ij}\cos\frac{i(2k+1)\pi}{2N}\cos\frac{j(2m+1)\pi}{2N},\ k,m=\overline{0,N-1}\ , \tag{7.2}$$

де

$$\lambda_{ij}=\begin{cases}1/N & ,\ i=0\wedge j=0\\ \sqrt{2}/N & ,\ (i=0\wedge j>0)\vee(i>0\wedge j=0)\\ 2/N & ,\ i>0\wedge j>0\end{cases}.$$

Застосовуючи ДКП до кожної робочої матриці, отримують матриці коефіцієнтів ДКП. Коефіцієнти у верхньому лівому куті цих матриць відповідають низькочастотним складовим амплітуди колірної компоненти, а в нижньому правому – високочастотним.

Крок 4. Проводять квантування коефіцієнтів ДКП поелементним діленням цих матриць на *матриці квантування* (МК) з подальшим округленням до цілого. На цьому кроці здійснюють управління ступенем стиску, тут відбуваються найбільші втрати.

Крок 5. Матриці коефіцієнтів ДКП розміром 8x8 переводять у 64-елементні вектори методом "зигзаґ-сканування" – у вектор послідовно поміщають елементи матриць з такими індексами (0,0), (0,1), (1,0), (2,0), (1,1), (0,2), … (8,8). Отже, на початку вектора отримують коефіцієнти ДКП, які відповідають низьким частотам, а наприкінці – високим, які можна відкинути.

Крок 6. Вектор згортають за допомогою групового кодування (*RLE*). При цьому отримують пари виду <пропустити, число>, де "пропустити" – лічильник пропущених нулів, а "число" – значення наступної комірки.

Крок 7. Здійснюють кодування методом Гаффмана.

Процес відновлення зображення для зазначеного способу кодування повністю симетричний. Метод дає змогу стискати деякі зображення у 10–15 разів без серйозних втрат.

Переваги алгоритму:

1) вихідне кольорове зображення може мати глибину до 24 біт/піксель;

2) задається ступінь стиску.

Недоліки алгоритму:

1) за великих ступенів стиску зображення розпадається на квадрати 8x8 пікселів одного кольору;

2) ореоли на межі різких переходів кольорів, зумовлені ефектом Гіббса для ДКП;

3) розмивання чітких ліній (наприклад, у кресленнях, схемах, картах);

4) можлива незворотна втрата інформації.

Фрактальне стиснення – це пошук самоподібних областей у зображенні та визначення для них параметрів афінних перетворень.

Рекурсивний* (*хвильовий*) *алгоритм. Англійська назва рекурсивного стиску – *wavelet*. Орієнтований на кольорові та чорно-білі зображення з плавними переходами. Ідеальний для зображень типу рентгенівських знімків.

Ідея алгоритму полягає у збереженні у файлі різниці між середніми значеннями сусідніх блоків у зображенні, яка для зображень з плавною зміною кольорів близька до нуля.

Наприклад, два числа a_{2i} та a_{2i+1} завжди можна відновити за їхнім середнім значенням $b_i^1 = (a_{2i} + a_{2i+1})/2$ та усередненою різницею $b_i^2 = (a_{2i} - a_{2i+1})/2$. Аналогічно, послідовність a_j $\left(j = \overline{1,2m} \right)$ можна відновити за послідовностями b_i^1 та b_i^2, $\left(i = \overline{1,m} \right)$.

Розглянемо, наприклад, стиснення стрічки з восьми значень яскравості пікселів (a_j): (220; 211; 212; 218; 217; 214; 210; 202). Отримаємо такі послідовності b_i^1 та b_i^2: (215,5; 215; 215,5; 206) і (4,5; -3; 1,5; 4). Зазначимо, що значення b_i^2 близькі до нуля.

Повторимо операцію для b_i^1, тоді отримаємо: (215,25; 210,75); (0,25, 4,75), (4,5; -3; 1,5; 4). Округлюємо коефіцієнти до цілих і стискаємо алгоритмом Гаффмана з фіксованими таблицями.

У розглянутому прикладі *wavelet*-перетворення здійснено двічі, його можна повторити для довшого рядка 4–6 разів.

Алгоритм для двовимірного випадку реалізується аналогічно.

Розглянемо квадрат з 4-х пікселів і значеннями яскравості $a_{2i,2j}$, $a_{2i+1,2j}$, $a_{2i,2j+1}$, $a_{2i+1,2j+1}$ та еквівалентний масив даних:

$$
\begin{aligned}
b_{i,j}^{11} &= (a_{2i,2j} + a_{2i+1,2j} + a_{2i,2j+1} + a_{2i+1,2j+1})/4;\\
b_{i,j}^{12} &= (a_{2i,2j} + a_{2i+1,2j} - a_{2i,2j+1} - a_{2i+1,2j+1})/4;\\
b_{i,j}^{21} &= (a_{2i,2j} - a_{2i+1,2j} + a_{2i,2j+1} - a_{2i+1,2j+1})/4;\\
b_{i,j}^{22} &= (a_{2i,2j} - a_{2i+1,2j} - a_{2i,2j+1} + a_{2i+1,2j+1})/4.
\end{aligned}
\qquad (7.3)
$$

Використовуючи це перетворення для зображення розміром 256x256 пікселів, отримаємо 4 матриці розміром 128x128 елементи (рис. 7.2). У першій з них зберігатиметься зменшена копія зображення, у другій – усереднені різниці значень яскравості пікселів по горизонталі, у третій – усереднені різниці значень яскравості пікселів по вертикалі, у четвертій – різниця усереднених значень яскравості пікселів по діагоналях.

Рис. 7.2. Один крок рекурсивного алгоритму

Можна продовжити перетворення для першої матриці та отримати чотири матриці розміром 64x64 елементів. Повторивши перетворення 3-й раз, отримаємо чотири матриці розміром 32x32 елементи, три матриці розміром 64x64 елементи і три матриці розміром 128x128 елементів. На практиці четвертою матрицею зазвичай нехтують (вважають її коефіцієнти рівними нулю). Це дає стиснення на третину $\left(\frac{1}{4}+\frac{1}{16}+\frac{1}{64}+\ldots=\frac{1}{3}\right)$.

Далі можна провести квантування коефіцієнтів і здійснити стиск методом Гаффмана.

Відновлення зображення цілком симетричне. Зазначимо, що збереження зменшеного зображення дає змогу, зокрема, поступово проявляти зображення.

Алгоритм JPEG 2000. Алгоритм *JPEG 2000* є вдосконаленням алгоритму *JPEG*. Головні відмінності цього алгоритму від попереднього такі:

1) краща якість зображення за такого ж ступеня стиску;

2) підтримка кодування окремих областей з кращою якістю (наприклад, очей людини на фотографії);

3) стиснення методом ДКП замінено на *wavelet*-стиск, що, зокрема, дає змогу поступово проявляти зображення;

4) метод Гаффмана замінений на ефективніший метод арифметичного стиснення;

5) підтримка стиску без втрат;

6) підтримка стиску двоколірних зображень (глибина – 1біт/піксель);

7) підтримка прозорості на рівні формату.

7.4. Растрові графічні формати

Опишемо головні растрові графічні формати, базуючись на енциклопедії графічних форматів [27].

***Bit Map format* (*BMP*).** *BMP* – стандартний растровий формат ОС *Windows*, створений фірмою "*Microsoft*". Файли цього формату можуть мати розширення .bmp, .dib. У цьому форматі можна зберігати зображення з глибиною кольору 1, 4, 8, 24 біт/піксель з використанням *RGB*-моделі; 1, 4, 8-бітна глибини відповідають індексованому кольоровому зображенню, для яких в інформаційному блоці растра зберігається таблиця колірності.

Файли цього формату мають просту структуру.

Заголовок файла (14 байт): перші 2 байти – символи *ASCII "BM"*, далі – розмір файла у байтах та ін.

Інформаційний блок растра містить заголовок і таблицю кольорів (за потреби).

У заголовку інформаційного блока (40 байт) задають:

- ширину зображення в пікселях;
- висоту зображення в пікселях;
- кількість бітів на піксель (глибина кольору);
- розмір зображення в байтах;
- роздільність на горизонталі (піксель на метр);
- роздільність по вертикалі (піксель на метр);
- кількість кольорів у зображенні та ін.

Таблиця кольорів може мати одну з 4-х структур. Наприклад, якщо колірна глибина рівна 1, то таблиця кольорів містить 2 елементи – кольори переднього плану і фону. Якщо колірна глибина становить 4 біти, то таблиця зберігає 16 елементів і т. п. Якщо глибина кольору 24 біти, то таблиці кольорів не використовують. Довжина кожного елемента таблиці становить 4 байти: 1-й – *G*; 2-й – *B*; 3-й – *R*; 4-й – резерв.

Закодоване растрове зображення – двовимірний масив кодів кольорів пікселів.

Головна перевага – простота формату, а недолік – дані не стиснуті, отож файли займають дуже багато місця. Знаходить широке використання лише в середовищі *Windows*.

***Graphics Interchange Format* (*GIF*).** Розроблений фірмою *CompuServe* (1987, 1988) для передавання растрових зображень у глобальних мережах. *GIF* зберігає індексоване колірне зображення до 256-ти кольорів, колірна модель *RGB* (глибина 8 біт).

Формат *GIF* використовує стиснення даних за алгоритмом *LZW* (Лемпела–Зіва–Велча), що дає змогу значно зменшити розмір графічних файлів. Патентування алгоритму *LZW* 1995 року дещо стримувало застосування цього формату, однак термін його патентного захисту закінчився 2006 року.

GIF використовує чергування рядків (*interlaced*), завдяки чому можна швидше побачити зображення загалом, однак з меншою чіткістю. Цього досягають за рахунок запису і подальшого завантаження спочатку 1-го, 5-го, 9-го рядків, далі – 2-го, 6-го, 10-го рядків і т. д. У форматі *GIF* можна задавати прозорі кольори за допомогою додаткового α-каналу. *GIF*-файл може містити декілька растрових зображень. Послідовне їхнє відображення дає змогу здійснювати анімацію.

Головний недолік цього формату – обмеженість палітри 256-ма кольорами.

Знаходить широке використання в мережі "Інтернет" для збереження рисунків, схем, діаграм.

***Partable Networks Graphic* (*PNG*).** Формат *PNG* (читається [*pin*]) розроблений на зміну формату *GIF* (здебільшого внаслідок патентування 1995 року алгоритму *LZW*) з метою подання зображень в інтернеті. Має кілька вдосконалень, порівняно з *GIF*:

1) таблиця кольору може бути 3-байтовою (*True Color*);

2) використовується двовимірне чергування – для рядків і для стовпчиків;

3) підтримує напівпрозорість пікселів з 256-ма градаціями сірого;

4) засновує алгоритм стиснення без втрат *Deflate*, який використовує алгоритми *LZ*77 та Гаффмана і вільний від ліцензування.

***Joint Photografics Experts Group* (*JPEG*).** Абревіатура *JPEG* позначає ряд графічних форматів (.jpeg, .jpg, .jfif), розроблених на основі методів стиску *JPEG* та *JPEG 2000*, які описано вище.

Формат *JPEG* – поширений формат графічного зображення в інтернеті. Недоліками формату є розмивання чітких ліній, кутів (малопридатний для схем, креслень тощо) та можлива незворотна втрата інформації за стиснення.

***Flash PIX* (*FPX*).** Створений зусиллями відомих комп'ютерних і графічних фірм ("*Apple*", "*Cannon*", "*Hewlett-Packard*", "*IBM*", "*Intel*", "*Kodak*", "*Microsoft*"). Зберігає не тільки високоточне зображення, а й копії з нижчою роздільністю. По замовчуванню в браузер завантажується зображення з низькою роздільністю, яку можна збільшити на вимогу користувача.

***Tagged Image File Format* (*TIFF*).** Розроблений фірмою "*Aldus*" 1987 року, останню модифікацію *TIFF* 6 зроблено 1992 року.

TIFF є відкритим форматом, який дає змогу створювати будь-яку модель зображення.

Стандартизовані такі колірні моделі:

- двоколірні зображення (*bi-level image*);
- монохромне зображення (*grey-scale image*);
- індексоване кольорове зображення (*paletted color image*);
- повноколірне зображення (*full RGB*);
- також моделі *CMYK, YCbCr, CIELAB.*

Формат *TIFF* дає змогу зберігати в одному файлі будь-яку кількість зображень, допускає стиснення алгоритмами *PackBits* (варіант групового кодування), *CCITT*, *LZW*, *JPEG* та ін.

Сьогодні це визнаний стандарт для збереження сканованих зображень.

***Windows Meta File* (*WMF*).** Універсальний формат векторної графіки для ОС *Windows*. Розроблений фірмою "*Microsoft*" і є невід'ємною частиною *Windows*. Зокрема, його використовують під час перенесення інформації через системний буфер (*Clipboard*).

7.5. Векторні та комбіновані формати

До векторних форматів можна зачислити робочі формати програм векторної графіки *CorelDraw* (.cdr), *Adobe Illustrator* (.ai), *Macromedia FreeHand* (.fh7, .fh7), *Adobe Flash* (.fla), *Golden Sofware Grapher* (.grf) та *Surpher* (.srf) та ін.

Багато сучасних графічних форматів є комбінованими: можуть поєднувати растрову та векторну графіку, фрагменти кодів та ін. Найпоширеніші з них – формати *PS, EPS, PDF, DjVu, SVG*.

***PostScript* (*PS*)** – мова опису сторінок. Підготовлені на *PostScript* файли є універсальними програмами для пристроїв виведення. Вони мають розширення .ps або, зрідка, .prn. Такі файли отримують, зокрема, з допомогою функції *Print to File* графічних програм при використанні драйвера *PostScript*-принтера. Такі файли містять у собі текст, зображення (растрові та векторні), шрифти, а також інформацію про кольороподіл, лінеатуру растра, форму растрової точки та ін.

У багатьох прикладних програмах передбачене перетворення документа в *PS*-програму. Цю програму можна послати безпосередньо на принтер з підтримкою *PostScript* або перетворити інтерпретатором *PostScript* в інший формат для принтера (якщо принтер не підтримує *PostScript*) чи дисплея. Існують інтерпретатори мови *PostScript* для різних операційних систем, найпоширеніший – програма *Ghostscript*.

Мова *PostScript* розроблена засновниками компанії "*Adobe Systems*" Джоном Ворноком (*John Warnock*) і Чарльзом Ґешке (*Charles Geschke*) 1984 року, а 1991 року ця компанія випустила наступну версію – *PostScript Level* 2, 1998 року ввела стандарт – *PostScript* 3.

***Encapsulated PostScript* (*EPS*)**. Формат *Encapsulated PostScript* (*EPS*) використовує спрощену версію *PostScript*: не може містити більше однієї сторінки, не зберігає деяких налаштувань для принтера. Файли такого формату можуть створювати більшість програм для роботи з графікою, однак їхнє редагування можна здійснити лише за допомогою програм *Adobe Photoshop* та *Adobe Illustrator*.

Разом з файлом можна зберегти ескіз (*image header*, *preview*). Це копія низької чіткості у форматі *TIFF*, *JPEG* або *WMF*. Для програм, які безпосередньо не працюють з форматом *EPS*, вона дає змогу побачити зображення. Такі програми імпортують ескіз, заміняючи його під час друку на *PostScript*-принтері оригінальною інформацією. На принтерах, які не підтримують *PostScript*, виводиться на друк власне ескіз.

***Portable Document Format* (*PDF*).** Формат *Portable Document Format* (*PDF*) – мобільний формат документів, розроблений компанією "*Adobe Systems*" для електронних видань.

PDF-файл містить *PDF*-публікацію та спеціальні дані. *PDF*-публікація (документ) має одну чи більше сторінок. Кожна сторінка може містити: текст, графіку та ілюстрації, анімацію, відео- та аудіоінформацію в апаратно-незалежному форматі, у вигляді так званого сторінкового опису (*page description*). Вона також може містити інформацію, яка забезпечує гіпертекстову навігацію в електронній публікації.

Формат *PDF* подає текст і графіку, використовуючи модель формування зображень мови *PostScript*. Оператори створення сторінок *PDF* подібні до операторів мови *PostScript*. Однак мова *PDF* – це не мова програмування; вона не містить змінних, процедур та ін.

Для зменшення розміру файла *PDF* використовує різні методи стиску даних: *LZW* – для текстових даних; *JPEG* – для повноколірних ілюстрацій; *CCITT* – для чорно-білих зображень.

Для забезпечення незалежності від шрифтів *PDF*-файл містить опис кожного шрифту, використаного в документі, який включає назву, кегль (розмір) і стиль шрифту. Якщо шрифт документа відсутній в операційній системі, то в режимі перегляду він буде замінений на подібний. Деякі шрифти можуть вбудовувати в документ.

Найбільші можливості для роботи з *PDF*-файлами надає пакет *Adobe Acrobat*. Однак існує низка безкоштовних програм для створення та перегляду цих файлів. Для перегляду та друку найпоширенішими є офіційна програма компанії "*Adobe*" *Acrobat Reader*, а також програма *Foxit Reader*. Для створення *PDF*-файлів можна використати *PrimoPDF*, *PDFWriter*, *PDFTeX*, плагіни для *MS Office* та ін.

У грудні 2007 року формат *PDF* затверджено як стандарт *ISO* 32000.

DjVu. Вимовляється *DjVu* як "дежавю" (з французької "колись вже бачене"). Технологія стискання зображень з втратами розроблена компанією "*AT&T*" 1996 року спеціально для зберігання сканованих документів – книг, журналів, рукописів, де наявна велика кількість рисунків, схем, формул та інших зображень, які роблять розпізнавання такого документа дуже складним.

Файли, підготовлені за цією технологією, мають розширення .djvu та .djv.

У процесі перекодування в *DjVu*-формат використовують технологію розділення вихідного зображення на три шари – передній план, фон та чорно-білу маску. До кожного з цих шарів застосовують окремі алгоритми стиснення:

1) універсальний алгоритм стиснення *ZP* для зображення переднього плану;

2) вейвлет-алгоритм *IW44* для стиснення фону;

3) алгоритм *JB2* (*DjVuBitonal*) для стиснення чорно-білого зображення.

Це дає змогу забезпечити високу ефективність стиснення. Наприклад, для сканованого документа розмір *DjVu*-файла, зазвичай, у 3–10 разів менший, ніж відповідні чорно-білі *TIFF*- чи *PDF*-файли, і в 5–10 менший, ніж кольоровий *JPEG*-файл.

Для перегляду файлів формату *DjVu* використовують безкоштовні програми *WinDjView*, *DjVuSolo*.

Детальнішу інформацію про комп’ютерну геометрію, алгоритми комп’ютерної графіки та цифрової обробки зображень можна почерпнути з книг [8; 28; 30].

Запитання та завдання

1. Опишіть поняття: логічний піксель, фізичний піксель, растрова точка.
2. Що таке растрове зображення?
3. Що називають векторною (об’єктною) моделлю зображення?
4. Чому рівна векторна роздільна здатність людського ока?
5. Що називають лінійною роздільною здатністю графічного пристрою?
6. Назвіть основні переваги та недоліки растрових зображень.
7. Назвіть основні переваги та недоліки векторних моделей зображень.
8. Опишіть принцип роботи кольорового дисплея.
9. Які головні параметри визначають якість зображення монітора?
10. Опишіть основні графічні формати.
11. Назвіть методи стиснення графічних даних без втрати інформації.
12. Назвіть методи стиснення графічних даних з втратою інформації.

Розділ 8. КОДУВАННЯ ЗВУКУ ТА ВІДЕО

8.1. Перетворення звукового сигналу у цифровий

Звук – це явище поширення коливань у пружному середовищі. Людське вухо відчуває зміни тиску, зумовлені звуковими коливаннями у діапазоні частот від 20 Гц до 20 кГц. Найвища чутливість досягається у діапазоні 3,0–3,5 кГц і дорівнює 10^{-5} Па [31].

За допомогою технічних пристроїв, таких як мікрофон, звукові коливання можна перетворити в електричний сигнал, а, отже, – у цифрову форму за методикою, описаною у розділі 2. Цифровані звукові сигнали можна накопичувати на цифрових носіях інформації, обробляти на комп'ютерах, передавати цифровими каналами зв'язку.

Перетворення звукового сигналу у цифровий – це квантоване вимірювання амплітуди звукового сигналу через однакові проміжки часу, яке відбувається у два етапи:

дискретизація – представлення функції сукупністю її значень у деякі моменти часу (*sampling*) – перехід від функції неперервного аргументу до функції дискретного аргументу;

квантування – представлення функції з неперервною шкалою значень функцією з дискретною шкалою значень (*quantizing*).

Частота дискретизації (*sampling rate*) показує кількість вимірювань амплітуди в секунду. Відповідно до теореми Котельникова–Найдквіста, частота дискретизації повинна бути щонайменше вдвічі більшою від максимальної частоти аналогового сигналу (див. підрозділ 2.3).

У цифровій телефонії використовують частоту дискретизації 8 кГц, стандарт цифрового запису на компакт-диск (англ. *Compact Disc Digital Audio, CD-DA*) – 44,1 кГц, формат *AAC* – до 96 *кГц*, а формат *Ogg Vorbis* – до 192 кГц. Шкала квантування є цілочисельною – 8-, 16- або 32-бітною.

Частота дискретизації і шкала квантування визначають *швидкість потоку* або *бітрейт* (*bitrate*) – кількість даних однієї секунди звучання, яку вимірюють у бітах за секунду (*bps*).

Легко підрахувати, що пряме оцифрування однієї секунди стереозвуку (2 канали) за частоти дискретизації 44,1 кГц та 16-бітної глибини потребує $44100 \times 16 \times 2$ біт $= 1378{,}125$ Кбіт $\approx 1{,}346$ Мбіт пам'яті. Отож для зменшення об'єму аудіоданих широко застосовують методи стиску даних – як без втрат інформації, так і з її втратами.

Програми стиснення (кодування та декодування) аудіо та відео називають *кодеками*.

Декілька слів про носій – *лазерний компакт-диск* (*Compact Disc, CD*). Стандартний компакт-диск складається з трьох шарів: основа (полікарбонат), дзеркальний шар (алюміній, срібло, золото), захисний шар (лак). Товщина диска – 1,2 мм, його діаметр – 12 см.

Інформаційний рельєф нанесено на спіральній доріжці, яка починається від центра диску, за допомогою заглибин – "пітів" (*pit* – ямка). Проміжки між "пітами" називають "лендами" (*land* – площадка). Чергування "піта" та "ленда" позначає цифру один, а довжина "піта" або "ленда" визначає серію нулів. Відстань між витками доріжки h дорівнює 1,4–2,0 мкм, мінімальна довжина заглибини d – 0,9 мкм.

Диски з можливістю запису (*Compact Disc-Recordable*, *CD-R*) мають подібну конструкцію, проте між основою і дзеркальним шаром розміщується шар органічної речовини, яка темніє від нагрівання. У початковому стані цей шар прозорий, однак за дії лазерного променя утворюються непрозорі ділянки, еквівалентні переходу від "ленда" до "піта", і навпаки. Подібно сконструйовані диски з можливістю перезапису (*Compact Disc-ReWritable*, *CD-RW*), а також інші типи дисків – *DVD*, *BD* тощо.

Масове виготовлення компакт-дисків здійснюють методом штампування за допомогою матриці, виготовленої електрохімічним способом.

У стандарті *CD-DA* вихідний стереофонічний звуковий сигнал дискретизують з частотою 44,1 кГц і квантують глибиною 16 біт. Кожні шість відліків лівого та правого звукових каналів оформляють у мікрокадри розміром 24 байти і далі кодують кодом Ріда–Соломона, у результаті отримують блок довжиною 32 байти (256 біт). Код Ріда–Соломона має надлишковість 25 %, що дає змогу виявляти до 4-х помилкових байтів і виправляти 2 байти.

Об'єм даних однієї секунди звучання дорівнює 1,346 Мб звукових даних. Стандартний час звучання всього диска *CD-DA* становить 74 хвилини, на ньому записано близько 750 МБ аудіоданих (всього – 996 МБ).

Адресацію даних здійснюють за шкалою хвилина–секунда–сектор (сектор – 1/75 секунди). В одному секторі – 2352 Б аудіоданих. Одношвидкісні аудіодиски здійснюють 200–500 обертів за хвилину.

Відразу після створення диски *CD-DA* пристосували для запису не тільки звукових, а й інших типів даних. Їх назвали *CD-ROM*. Для *CD-ROM* один сектор містить 2048 Б (2 КБ) даних, що дає змогу розмістити на диску 650,39 МБ даних. Швидкість зчитування 150 КБ/с = 1,2 Мб/с.

Розвиток технології виготовлення оптичних дисків (звуження лазерного променя, покращення покриття, збільшення кількості реєструючих шарів та ін.) сприяв створенню місткіших дисків, найбільшого поширення серед яких набули стандарти *DVD* (*Digital Video Disc*) та *BD* (*Blu-ray Disc*). Назва останнього – *синій промінь* – пов'язана з кольором використовуваного лазера з довжиною хвилі 405 нм. Наведемо порівняльні характеристики основних одношарових оптичних носіїв інформації (табл. 8.1).

Таблиця 8.1. Характеристики одношарових оптичних дисків

Параметр	*CD*	*DVD*	*BD*
Об'єм	0,65 ГБ	4,7 ГБ	27 ГБ
Колір і довжина хвилі лазера	Інфрачервоний, 780 нм	Червоний, 650 нм	Фіолетовий, 405 нм
Відстань між витками	1,8 мкм	0,74 мкм	0,34 мкм

8.2. Звукові формати

Найбільшого застосування на практиці набули три головні методи моделювання звуку і збереження звукових даних у файлах для подальшого їхнього відтворення:

1) файли форми сигналу;

2) файли з нотним записом (керуючі файли);

3) дескриптивні (описові) файли.

Файли форми сигналу – це цифрові образи звуку, отримані в результаті аналого-цифрового перетворення – *Pulse Code Modulation* (*PCM*), до яких можуть додавати заголовок. У заголовку зазначають параметри, які характеризують цифрований звук:

- частоту дискретизації;
- кількість рівнів квантування відліків (зазвичай 8 або 16);
- кількість каналів (моно, стерео та ін.);
- тип файла в *ASCII*-коді;
- довжину даних у байтах;
- номер версії формату;
- метод стискання;
- зміщення блоку звукових даних відносно початку файла;
- мітки позицій для синхронізації та ін.

Коротко опишемо основні формати.

PCM (*Pulse Code Modulation*) – безпосередньо цифрований звук без заголовка. Розширення – .pcm. Застарілий формат, який трапляється зрідка.

CD-DA (*Compact Disc Digital Audio*) – стандарт для запису звуку на лазерний компакт-диск. Розширення – .cda. Запропонований 1980 року фірмами "*Sony*" та "*Philips*"; бітрейт – 1,346 Мб/с.

WAV (*Wave Form Audio File*) – 16-бітний формат, розроблений фірмою "*Microsoft*". Розширення – .wav. Допускає частоти дискретизації 11024, 22050 і 44100 Гц. Створений за стандартом *RIFF* (*Resource Interchange File Format*) для зберігання будь-яких структур даних. Використовують в ОС *Windows* для збереження службових звуків, що супроводжують події *Windows*.

VOC (*Voice File*) – 8-бітний формат, розроблений компанією "*Creative Lab*". Розширення – .voc. Підтримує довільну частоту дискретизації. Трапляється у старих програмах.

Файли з нотним записом (*song files, music files*) містять послідовність команд, які визначають ноту, інструмент, момент часу і тривалість відтворення. Формат передбачає одночасну гру декількох інструментів. У цьому випадку говорять про відповідну кількість голосів (каналів). Файл з нотним записом можна назвати програмою для оркестру. Найпоширеніший формат *MIDI* (*Musical Instrument Digital Interface*). Розширення – .mid, .midi, .rmi.

Для відтворення файлів з нотним записом використовують такі види синтезу:

– *FM*-синтез – імітація ноти музичного інструмента шляхом формування обвідної синусоїди з частотою цієї ноти;
– табличний синтез (*Wavetable Synthesis*) – використання цифрованих нот реальних інструментів з пам'яті звукової плати.

Дескриптивні формати. Найпоширеніший формат *MP3* – є частиною відеоформату *MPEG*-1 *Layer* 3. Розроблений німецьким інститутом "*Fraunhofer IIS*" 1994 року з подальшою підтримкою корпорації "*Thompson*". Прийшов на зміну формату *RA* (*Real Audio*). Формат *MP3* за відносно малого розміру забезпечує прийнятну якість звуку.

Цей формат використовує таку ж частоту дискретизації та глибину квантування, як і стандарт *CD-DA* (44,1 кГц; 16 біт); за використання лише алгоритму стиску без втрат інформації має бітрейт 320 Кб/с, тобто зменшує розмір вихідного файла у чотири рази, порівняно зі стандартом *CD-DA*.

Для зменшення ширини потоку цей формат застосовує алгоритми стиснення звукових даних з втратами, що ґрунтуються на психофізіологічному сприйнятті звуку людиною:

1) порогом чутливості високих тонів встановлена частота 16 кГц;

2) відкидаються слабкі звуки з рівнем нижче порогу чутливості людського вуха;

3) використовується психоакустичний ефект маскування одних звуків іншими: чутливість вуха послаблена 5 мс після виникнення сильного звуку і 100 мс після його завершення – вухо тимчасово є заглушеним; вухо налаштовується на певну частоту звучання (якщо йде сильний звук на частоті 1000 Гц, то слабкий звук з частотою 1100 Гц вже не розрізнятиметься).

Відповідно до зазначеного вище, здійснюється адаптивний стиск, який дає змогу зменшити ширину потоку до 64 Кб/с. Наголосимо, що бітрейт 256 Кб/с забезпечує якісне відтворення музичних творів, яке задовольняє людей з музичним слухом. Найпоширеніший бітрейт – 128 Кб/с.

MP3 є потоковим форматом, звукова інформація розбита на рівні за тривалістю частини – *фрейми*, які взаємно незалежні і мають заголовок. Це забезпечує легкий перехід до довільного фрейму та зручність використання у мережі "Інтернет".

На зміну формату *MP3* йдуть нові формати: лінійка форматів *Dolby Digital*, *Advanced Audio Codec* (*AAC*), *Ogg Vorbis*, *Microsoft Widows Media Audio* (*WMA*), *Twin VQ* та ін. Коротко розглянемо основні з них.

Dolby Digital – лінійка технологій стиснення аудіоданих з втратами, розроблена "*Dolby Laboratories*" (*Dolby Digital, Dolby Digital EX, Dolby Digital Live, Dolby Digital Surround EX, Dolby Digital Plus, Dolby TrueHD*), які широко застосовують у кінематографі, відеоіграх та відеофільмах.

Стандарт *Dolby Digital*, або *AC*-3, підтримує 5.1 аудіоканали: 5 – для частот 20 Гц – 20 кГц (правий передній, центральний, лівий передній, лівий задній, правий задній); 1 – низькочастотний (20–120 Гц). Підтримує частоту дискретизації до 48 кГц. Вперше використаний для озвучення фільму 1992 року.

Формат AAC (*Advanced Audio Coding*), офіційно відомий як *ISO/IEC* 13818 або як *MPEG*-2 *Part* 7, вийшов у світ 1997 року. Найновіша версія – *MPEG*-4 *Part* 3. Створений "*Fraunhofer IIS*" як наступник формату *MP3* з покращеною якістю кодування. Підтримує до 48-ми окремих каналів з частотою дискретизації до 96 кГц. Розширення файлів: .m4a – незахищений файл *AAC*; .m4b – файл *AAC*, який підтримує закладки; .m4p – захищений файл *AAC*.

Формат Ogg Vorbis вільний від ліцензування аудіоформат, розроблений 2002 року групою "*Xiphophorus*" для заміни платних аудіоформатів. Є другим після *MP3* за популярністю форматом компресії звуку з втратами. Використовує іншу, якіснішу, ніж *MP3*, психоакустичну модель. Підтримує до 255-ти окремих каналів з частотою дискретизації до 192 кГц. Це перший аудіоформат з глибиною квантування 32 біти. Розширення файлів – .ogg.

8.3. Моделювання відеоданих

Найпоширенішою моделлю рухомого зображення є його подання часовою послідовністю окремих статичних зображень – *кадрів*. За частоти кадрів понад 10 Гц людина сприймає таку послідовність, як неперервне зображення, а після 40 Гц – не зауважує миготіння. Кінематограф використовує частоту кадрів 24 Гц, телевізійні стандарти *PAL* та *SECAM* – 25 Гц, а американський стандарт телебачення *NTSC* – 30 Гц.

Чіткість відеозображення, як і статичних зображень, визначається розгорткою по горизонталі та вертикалі. *Аналогові телевізійні стандарти PAL* та *SECAM* (625 рядків розгортки по вертикалі) наближено відповідають цифровій розгортці 720x576 пікселів, *стандарт NTSC* (520 рядків розгортки по вертикалі) – цифровій розгортці 640x480 пікселів. Зауважимо, що зазначені стандарти використовують черезрядкову (*interlaced*) розгортку, коли спочатку виводяться непарні, а потім парні рядки кадру.

Стандарт цифрового телебачення високої чіткості (*High-Definition Television – HDTV*, або *Full HD*) передбачає розгортку 1920x1080 пікселів за частоти кадрів 60 Гц з пропорційною розгорткою (усі рядки виводяться послідовно). Сьогодні розроблено формат *Ultra HD* з роздільною здатністю 8–32 мегапікселі, зокрема, з розгортками 3840x2160 (**4К**) та 7680x4320 (**8К**) пікселів .

Цифровий стандарти 352x288 пікселів позначають абревіатурою *CIF* (*Common Interchange Format*), а стандарт 176x144 пікселів – *QCIF* (*Quartet Common Interchange Format*).

Головною проблемою моделювання та відтворення рухомих зображень є значні потоки даних. Зокрема, відео "телевізійного формату" 720x526 пікселів та 25 кадрів за секунду в системі *RGB True Color* (24 біт/піксель) вимагає потоку даних близько 240 Мб/с (30 МБ/с або 1,8 ГБ/хв). У цьому випадку традиційні алгоритми стиснення даних без втрат можуть забезпечити стиснення даних максимум на порядок, що не вирішує проблеми.

Алгоритми стиснення відеоданих та відеоформати, окрім ефективного та якісного стиску зображення, повинні забезпечити ще певні додаткові специфічні вимоги:

- синхронне відтворення зображення та звуку;
- можливість відтворення зображення з будь-якого місця;
- швидкий пошук вперед/назад;
- масштабування – можливість реалізації відео у вікні з різною розгорткою;
- незначний час кодування/декодування (актуально для систем реального часу).

8.4. Базові методи стиснення відеоданих

Технології стиснення відеоданих використовують два принципово різні методи стиснення – зменшення надлишковості у часовому вимірі та стиснення статичних зображень [31].

Алгоритми стиснення статичних зображень аналогічні описаним у розділі 7 і дають змогу стиснути відеодані у 10–15 разів. Розглянемо загальні принципи стиснення відеоданих у часовому вимірі, які використовують подібність послідовних кадрів.

Інтерполяція та екстраполяція. Базується на використанні різних типів кадрів. Зокрема, стандарти *MPEG* використовують кадри чотирьох типів:

I-кадри, стиснуті незалежно від інших (*I – intra-coded*);

P-кадри (*P – predicted*);

B-кадри, які використовують інформацію двох сусідніх кадрів (*B – bidirectional*);

DC-кадри (*DC – down components*) – містять лише низькочастотні компоненти ДКП.

I-кадри є базовими інформаційними кадрами і забезпечують доступ до будь-якого місця відео, їх розміщують послідовно через кожні 10–15 кадрів. Інші типи кадрів фіксують лише відмінності між кадрами. *P*-кадри використовують посилання на один *I*-кадр чи *P*-кадр. *B*-кадри використовують посилання на попередній та наступний кадр, однак їх не можна використовувати як посилання. *DC*-кадри слугують лише для швидкого пошуку.

Для стиснення кожен кадр переводять з колірного простору *RGB* у простір *YUV*. Далі кожну колірну площину розділяють на блоки розміром 8x8, для яких, аналогічно алгоритмам *JPEG*, проводять проріджування кольорів (для колірних компонент *U* та *V*). Блоки об’єднують у макроблоки – групи з чотирьох сусідніх блоків у площині яскравості *Y* (16x16 пікселів) та два відповідні блоки з колірних площин після проріджування. Далі здійснюють стиск макроблоків за алгоритмами, аналогічними *JPEG*, чи *JPEG*-2000. Макроблоки *I*-кадрів стискають незалежно; окремі макроблоки *P*-кадрів та *B*-кадрів стискають незалежно, або з урахуванням їхніх зв’язків з іншими кадрами.

Використання векторів зміщень блоків. Найпростішим способом стиснення, який враховує подібність кадрів, є кодування їхньої різниці. Однак ефективнішим виявляється алгоритм, який ґрунтується на визначенні векторів зсуву окремих блоків кадру стосовно попереднього кадру.

Розпаралелювання. Поділ кадрів на макроблоки дає змогу ефективно розпаралелити процес кодування та декодування відеоданих.

Алгоритми стиснення відеоданих постійно вдосконалюються. На зміну ДКП у блоках приходить вейвлет-перетворення, алгоритму Гаффмана – арифметичне кодування; впроваджуються нові алгоритми пошуку векторів зміщень, розвиваються об’єктно-орієнтовані технології та ін.

8.5. Головні відеостандарти

Motion-JPEG – один з найпростіших алгоритмів стиснення відеоданих, який полягає у незалежному стисненні окремих кадрів алгоритмом *JPEG*. Він забезпечує швидкий доступ до окремих кадрів, можливість редагування потоку, є простим для реалізації, однак забезпечує стиснення лише у 5–10 разів. Цей алгоритм стиску використовує розроблений фірмою "*Microsoft*" відеоформат *Audio Video Interlaced* (*AVI*), розширення файлів – .avi.

Лінійка стандартів MPEG. У рамках Міжнародної організації зі стандартизації (*ISO*) 1988 року розпочала роботу група експертів з цифрового відео – *Moving Pictures Experts Group* (*MPEG*), а 1990 року вона опублікувала стандарт кодування відео *MPEG-1*.

Цей стандарт використовує ширину потоку 1,5 Мбіт/с, підтримує розгортку стандарту *CIF* – 352x240x30, 352x288x25. Базові принципи стиснення викладено вище у підрозділі 8.4.

Стандарт *MPEG-2*, розрахований на потоки 3–15 Мбіт/с, вийшов 1995 року. Цей стандарт забезпечує телевізійну якість зображення (720x576 пікселів), його широко використовують у кабельному та супутниковому телебаченні.

Подальшим вдосконаленням стандартів *MPEG-1* та *MPEG-2* став стандарт *MPEG-4*, опублікований 1998 року. Цей стандарт має кілька вдосконалень, порівняно з попередніми версіями. Він підтримує мову віртуальної розмітки (*VRML*) для роботи з тривимірним зображенням, об’єктно-орієнтований підхід, інтерактивні медіазасоби, а також розширений стандарт аудіо *AAC*.

Широко використовується для запису відео на диски та у відеотелефонії.

Стандарт H.264. Стандарт *H.264*, *MPEG-4 Part 10* або *AVC* (*Advanced Video Coding*) розроблений спільно *VCEG* (*Video Coding Experts Group*) та *MPEG* 2003 року. Цей стандарт має чимало нових вдосконалень, насамперед: багатокадрове передбачення (до 32-х кадрів); гнучкі функції черезрядкового стиснення; нові алгоритми стиснення блоків; засоби стійкості від помилок. Використовують цей стандарт у цифровому *телебаченні високої чіткості* (ТВЧ), з ним працюють Міністерством оборони США та фірма "*Apple*".

Запитання та завдання

1. Назвіть два головні етапи перетворення аналогових сигналів у цифрові.
2. Що таке частота дискретизації?
3. Що таке глибина квантування?

4. Що таке АЦП і ЦАП?
5. Назвіть три основні способи цифрового кодування звукових даних.
6. Опишіть формат форми сигналу представлення звукових даних.
7. Назвіть формати звукових даних у формі нотного запису.
8. Опишіть дескриптивний формат представлення звукових даних.
9. Зазначте психофізіологічні особливості сприйняття звуку людиною.
10. Назвіть найпоширеніші формати звукових даних.
11. Опишіть формат *MP3*.
12. Які переваги та недоліки цифрового кодування звукових даних?
13. Охарактеризуйте психофізіологічні особливості сприйняття рухомого зображення людиною.
14. Сформулюйте загальні принципи стиснення відеоданих.
15. Назвіть найпоширеніші формати відеоданих.

Розділ 9. КОДУВАННЯ З ЗАХИСТОМ ВІД ЗАВАД

9.1. Загальні поняття

Завадостійкими називають коди, що дають змогу виявляти чи виявляти і виправляти помилки, зумовлені завадами. Коди, які виправляють помилки, ще називають *коректуючими*. Такі коди є принципово важливими для надійного передавання та зберігання даних. У цьому розділі викладено основи завадостійкого кодування на базі праць [6; 22].

Усі завадостійкі схеми кодування мають дві загальні ознаки:

1) надлишковість – кодові послідовності, поряд з *інформаційними розрядами*, завжди мають додаткові розряди, які називають *перевірними* або *контрольними*;

2) властивість усереднення – перевірні розряди залежать від інформаційних розрядів.

Існує два великих класи завадостійких кодів – *блокові* і *згортальні коди*. За *блокового кодування* вихідне повідомлення розбивається на блоки довжиною k символів. Кожен такий блок кодується кодовою послідовністю у n символів додаванням $r = n - k$ допоміжних розрядів, які називають перевірними (контрольними) розрядами. Контрольні розряди залежать лише від інформаційних розрядів поточного блоку.

Згортальні коди (або *коди з пам'яттю*) також використовують розбиття на блоки і додавання допоміжних розрядів, однак останні залежать від декількох попередніх інформаційних блоків.

На множині $\mathbf{E} = \{0,1\}$ означимо операцію додавання за модулем два ($\oplus$) та операцію множення, тоді $\mathbf{E}$ – поле, а множина двійкових послідовностей довжини n – $\mathbf{E}^n$ утворює лінійний (векторний) простір над цим полем. Отож блочне кодування – це відображення з $\mathbf{E}^k$ у $\mathbf{E}^n$.

Блочний код довжиною n символів називають *лінійним* (n,k)*-кодом*, якщо множина його кодових слів V утворює k-вимірний підпростір простору $\mathbf{E}^n$.

Приклад 9.1. Метод потрійного дублювання. Найпростішим способом завадостійкого кодування є метод потрійного дублювання інформаційних розрядів – $(3k,k)$-код. Декодування здійснюють методом "голосування". Таке кодування не знаходить практичного використання через значну надлишковість.

Приклад 9.2. Коди з перевіркою на парність. Одним з найпростіших лінійних кодів є $(n,n-1)$-код, перші $n-1$ символи якого є інформаційними, а останній символ дорівнює сумі попередніх за модулем два. Перевірний символ доповнює кодову послідовність так, що кількість одиниць є завжди парною. Тому сума за модулем два для такого коду (контрольна сума) завжди рівна нулю. Наприклад, кодове слово для $(4,3)$-коду матиме вигляд $(a_1,a_2,a_3,a_1\oplus a_2\oplus a_3)$. Такий код дає змогу лише виявити непарну кількість помилок.

Приклад 9.3. Ітеративні коди. Розглянемо на прикладі. Припустимо, що нам потрібно передати 9 інформаційних символів. Розмістимо їх у вигляді квадратної матриці і порахуємо суми за модулем два за стовпцями та рядками. Очевидно, що такий спосіб дає змогу виявити і виправити одиничні у рядках і стовпчиках помилки. В узагальненому випадку для m^2 інформаційних розрядів необхідно $2m$ контрольних розрядів.

9.2. Теоретико-множинний аналіз завадостійкого кодування

Виконаємо загальний теоретико-множинний аналіз завадостійкого кодування.

Нехай $A=\{a_1,a_2,\ldots,a_l\}$, $B=\{b_1,b_2,\ldots,b_m\}$ – деякі алфавіти; A^*, B^* – множини слів, які можна скласти з алфавітів A та B, відповідно; $S\subset A^*$ – деяка скінченна підмножина множини A^* (для алфавітного кодування $S=A^*$).

Припустимо, що кодування та декодування відбувається без помилок, помилки виникають лише під час передавання даних каналом з перешкодами.

Тоді передавання даних з корекцією помилок можна подати такою схемою:

$$M\xrightarrow{F}C\xrightarrow{N}C'\xrightarrow{\tilde{F}^{-1}}M,$$

де M – вихідне повідомлення; C – його код; C' – код з помилками;

$F: S \to B^*$ – взаємно однозначна функція кодування;

$N: B^* \to B^*$ – функція завад (каналу);

$\tilde{F}^{-1}$ – функція декодування.

Кодування F називають завадостійким для функції завад N, якщо існує однозначна функція (алгоритм) декодування $\tilde{F}^{-1}: B^* \to S$, яка задовольняє умову

$$\forall s \in S \ \tilde{F}^{-1}(N(F(s))) = s. \tag{9.1}$$

За двійкового кодування, коли $B = E = \{0,1\}$, можливі такі типи помилок:

- помилки заміни розряду – $0 \to 1 \vee 1 \to 0$;
- помилки випадання розряду – $0 \to \varnothing \vee 1 \to \varnothing$;
- помилки додавання розряду – $\varnothing \to 0 \vee \varnothing \to 1$.

У цьому випадку функцію перешкод N можна описати за допомогою вектора $\boldsymbol{\delta}_n = (k_r, k_d, k_i)$ верхніх оцінок помилок кожного типу на повідомлення довжини n.

Помилки заміни розрядів моделюють додаванням до вихідного коду за модулем два вектора помилок $\boldsymbol{\varepsilon} \in \mathbf{E}^n$, який містить k_r ненульових розрядів, тобто $C' = C \oplus \boldsymbol{\varepsilon}$.

Позначимо через $F_\delta(s) \subset B^*$ множину всіх слів, які можна отримати з коду $F(s) \in B^*$ в результаті дії всіх можливих завад у каналі $F_\delta(s) = \{\beta \in B^* \,|\, \beta = N_\delta(F(s))\}$. Припустимо, що ця множина – скінченна.

Теорема 9.1. Загальна теорема про завадостійке кодування

Кодування F – завадостійке для перешкод N_δ тоді і лише тоді, коли виконується умова

$$\forall s_1, s_2 \in S \subset A^*: s_1 \neq s_2 \Rightarrow F_\delta(s_1) \cap F_\delta(s_2) = \varnothing. \tag{9.2}$$

Доведення

Необхідність. Припустимо протилежне $\exists s_1, s_2 \in S$ $(s_1 \neq s_2) \wedge (F_\delta(s_1) \cap F_\delta(s_2) \neq \varnothing)$, тобто $\exists \beta \in F_\delta(s_1) \cap F_\delta(s_2)$, тоді функція декодування – неоднозначна.

Достатність. У цьому випадку є взаємно однозначна відповідність між s та множиною $F_\delta(s)$, що дає змогу побудувати однозначну функцію декодування $\tilde{F}^{-1}$.

Наведену теорему ілюструє рис. 9.1.

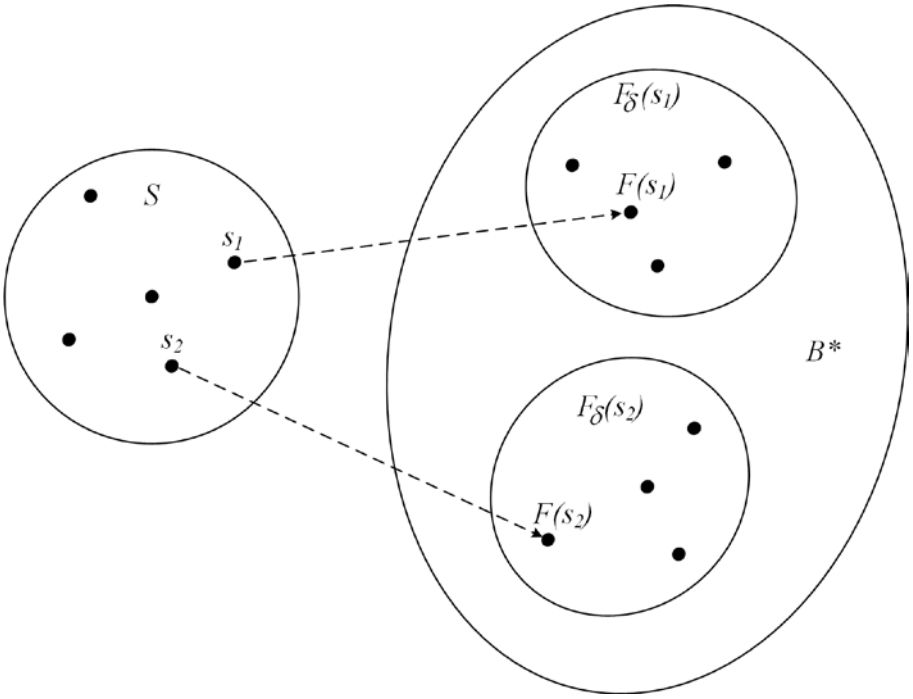


Рис. 9.1. Кодування за наявності завад

З теореми 9.1 маємо метод завадостійкого кодування. За виконання умови (9.2) кодом $s \in S$ є множина $F_\delta(s)$, тобто задається код $F(s)$ та коди-супутники $F_\delta(s) \setminus F(s)$. Таку схему застосовують зрідка через її громіздкість.

9.3. Кодова відстань

Перевірку умови (9.2) легко здійснити, якщо ввести метрику на множині B^*. Таку метрику для двійкових кодів постійної довжини запропонував американський математик Річард Веслі Геммінґ (*Richard Wesley Hamming*) 1950 року.

Позначимо множину кодів через $V \subset \mathbf{E}^n$. Окремим кодом буде вектор-рядок з V.

Відстанню Геммінґа між двома кодами $\beta', \beta'' \in V$ називають кількість розрядів, якими відрізняються ці коди:

$$d(\beta', \beta'') = \sum_{i=1}^{n} b_i' \oplus b_i''. \qquad (9.3)$$

Легко перевірити виконання аксіом відстані:

1) $d(\beta', \beta'') \geq 0;\ d(\beta', \beta'') = 0 \Leftrightarrow \beta' = \beta''$;

2) $d(\beta', \beta'') = d(\beta'', \beta')$;

3) $d(\beta',\beta'')+d(\beta'',\beta''')\geq d(\beta',\beta''')$.

Вагою Геммінга кодової послідовності $\beta\in V\subset\mathbf{E}^n$ називають кількість її ненульових елементів:

$$w(\beta)=\sum_{i=1}^{n}b_i\,. \tag{9.4}$$

Виконуються очевидні рівності:

$$d(\mathbf{0},\beta)=w(\beta)\,; \tag{9.5}$$

$$d(\beta',\beta'')=w(\beta'\oplus\beta'')\,; \tag{9.6}$$

$$d(\beta'\oplus\beta''',\beta''\oplus\beta''')=w(\beta'\oplus\beta'')=d(\beta',\beta'')\,. \tag{9.7}$$

Кодовою відстанню для коду $V\subset\mathbf{E}^n$ називають величину

$$d(V)=\min_{\beta',\beta''\in V}d(\beta',\beta'')\,. \tag{9.8}$$

Приклад 9.4. Розрахунок кодової відстані. Нехай $V=\{\beta_1=00101,\beta_2=10111,\beta_3=10001\}\subset\mathbf{E}^5$. Тоді $w(\beta_1)=2$, $w(\beta_2)=4$, $w(\beta_3)=2$, $\beta_1\oplus\beta_2=10010$, $d(\beta_1,\beta_2)=2$, $d(\beta_2,\beta_3)=2$, $d(\beta_3,\beta_1)=2$. Отож $d(V)=2$.

Теорема 9.2. Про кодову відстань для завадостійких кодів

Схема кодування двійковим кодом постійної довжини $V\subset\mathbf{E}^n$ є завадостійкою до k помилок заміни розрядів $(\delta_n=(k,0,0))$ тоді і лише тоді, коли кодова відстань удвічі більша від k:

$$d(V)>2k\,. \tag{9.9}$$

Д о в е д е н н я

Нехай $\boldsymbol{\varepsilon}$ – довільний двійковий вектор, який має не більше k одиниць. Тоді виконується нерівність:

$$\forall\beta\in\mathbf{E}^n\ d(\beta,\beta\oplus\boldsymbol{\varepsilon})=w(\beta\oplus\beta\oplus\boldsymbol{\varepsilon})=w(\boldsymbol{\varepsilon})\leq k\,.$$

Перенумеруємо всі коди, тоді $V=\left\{\beta_i\in\mathbf{E}^n,\ i=\overline{1,m}\right\}$, і означимо їхні k-околи: $F_i=\left\{\beta\in\mathbf{E}^n\mid d(\beta_i,\beta)\leq k\right\}$, $i=\overline{1,m}$.

Необхідність. Оскільки схема завадостійка, то за теоремою 9.1 $\forall 1 \le i, j \le m \; i \ne j \Rightarrow F_i \cap F_j = \varnothing$. Потрібно довести $d(V) > 2k$. Припустимо протилежне – $d(V) \le 2k$. Тоді існують $\beta_p, \beta_q \in V$, $p \ne q$ такі, що $d(\beta_p, \beta_q) = k + l$, $0 < l \le k$.

Розглянемо $\varepsilon = \beta_p \oplus \beta_q$. Очевидно, що $w(\varepsilon) = k + l \le 2k$. Подамо ε у вигляді суми $\varepsilon = \varepsilon_1 \oplus \varepsilon_2$ так, щоб $w(\varepsilon_1) = k$, а $w(\varepsilon_2) = l$. Це легко зробити, наприклад, залишаючи в ε_1 k перших одиниць з ε. Легко бачити, що $\beta_p \oplus \varepsilon_1 = \beta_q \oplus \varepsilon_2 \equiv x$. З іншого боку $d(\beta_p, x) = k$, а $d(\beta_q, x) = l$, тобто $x \in F_p \cap F_q$. Отримали протиріччя.

Достатність. Доведемо, що з умови $d(V) > 2k$ випливає, що $\forall 1 \le i, j \le m \; i \ne j \Rightarrow F_i \cap F_j = \varnothing$. Припустимо протилежне: $\exists \alpha \in \mathbf{E}^n \; \alpha \in F_i \cap F_j$. Тоді $d(\beta_i, \beta_j) \le d(\beta_i, \alpha) + d(\alpha, \beta_j) \le 2k < d(V)$. Але завжди $d(\beta_i, \beta_j) \ge d(V)$. Отримали протиріччя.

9.4. Лінійні систематичні коди

Блочний код довжиною n символів називають *лінійним* (n,k)*-кодом*, якщо множина його кодових слів V утворює k-вимірний підпростір простору $\mathbf{E}^n$. Найпростіші приклади лінійних кодів розглянуто у підрозділі 9.1.

Для лінійних кодів справджується теорема.

Теорема 9.3. Про кодову відстань для лінійного коду

Кодова відстань для лінійного коду дорівнює мінімальній вазі його ненульових векторів.

Доведення є очевидним, оскільки $\forall u', u'' \in V \; \exists u \in V \; u = u' \oplus u''$ і тому $\min\limits_{u', u''} d(u', u'') = \min\limits_{u', u''} w(u' \oplus u'') = \min\limits_{u} w(u)$.

Для елементів простору $\mathbf{E}^n$ введемо скалярний добуток за формулою

$$uv^T = \bigoplus_{i=1}^{n} u_i v_i . \qquad (9.10)$$

На основі цього легко означити добуток двійкової матриці на вектор і добуток двійкових матриць.

Як описати лінійний (n,k)-код? За означенням, будь-який вектор з V можна подати як лінійну комбінацію k лінійно незалежних векторів. Отож цей код можна описати матрицею $G = [g_{ij}] \in M_{k,n}(\mathbf{E})$, рядками якої є базисні вектори підпростору V:

$$G=\begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{bmatrix}. \tag{9.11}$$

Таку матрицю називають *твірною матрицею*.

Розглянемо простір V' розмірності $r=n-k$, який є ортогональним доповненням до підпростору V в $\mathbf{E}^n$. Запишемо матрицю $H=[h_{ij}]\in M_{r,n}(\mathbf{E})$, рядками якої є базис підпростору V':

$$H=\begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ h_{21} & h_{22} & \cdots & h_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ h_{r1} & h_{r2} & \cdots & h_{rn} \end{bmatrix}. \tag{9.12}$$

Тоді очевидними є еквівалентні рівності:

$$\forall \mathbf{v}\in V\ \mathbf{H}\mathbf{v}^T=\mathbf{0}\in\mathbf{E}^r, \tag{9.13}$$

$$\bigoplus_{j=1}^{n} h_{ij}v_j=0,\ i=\overline{1,r}, \tag{9.14}$$

$$\bigoplus_{j=1}^{n} [h_{\cdot j}]v_j=\mathbf{0}\in\mathbf{E}^r, \tag{9.15}$$

де $[h_{\cdot j}]$ – j-й стовпчик матриці $\mathbf{H}$.

Отже, компоненти вектора $\mathbf{v}\in V$ повинні задовольняти $r=n-k$ незалежних рівнянь. Ці рівняння називають *узагальненою перевіркою на парність*. Відповідно до рівності (9.13), підпростір V ще *називають нуль-підпростором матриці* $\mathbf{H}$, а матрицю $\mathbf{H}$ – *перевірною матрицею*.

Зауваження 9.1. Співвідношення (9.13) виконуються для довільного вектора з V, а, отже, і для базових векторів, отож:

$$\mathbf{H}\mathbf{G}^T=[\mathbf{0}]\in M_{r,k}(\mathbf{E}). \tag{9.16}$$

Очевидно, що підпростори V та V' однозначно визначаються матрицею $\mathbf{G}$ або матрицею $\mathbf{H}$, проте рівності (9.16) однозначно не визначають матрицю $\mathbf{G}$ за матрицею $\mathbf{H}$, і навпаки.

Зауваження 9.2. Чому матрицю $\mathbf{G}$ називають твірною?

Розглянемо вектор $\mathbf{a}=(a_1,a_2,\ldots,a_k)\in\mathbf{E}^k$, тоді $\mathbf{v}=\mathbf{a}\mathbf{G}\in\mathbf{E}^n$. Знайдемо $\mathbf{H}\mathbf{v}^T$: $\mathbf{H}\mathbf{v}^T=\mathbf{H}\mathbf{G}^T\mathbf{a}^T=\mathbf{0}\in\mathbf{E}^r$. Отож $\mathbf{v}$ є правильним кодом з V!

Розглянемо деякі теореми та загальні наслідки.

Теорема 9.4. Про нульовий підпростір перевірної матриці

Нехай V – лінійний (n,k)-код, який є нульовим підпростором матриці $\mathbf{H}$. Тоді кожному слову з V ваги w відповідає відношення лінійної залежності, яке пов'язує w стовпців матриці $\mathbf{H}$. І навпаки, кожному відношенню лінійної залежності між w стовпчиками відповідає кодове слово ваги w.

Теорема є очевидним наслідком формули (9.15). З неї випливає наслідок.

Наслідок 9.1. *Якщо будь-які s стовпців матриці $\mathbf{H}$ є лінійно незалежні, то кодова відстань для її нуль-підпростору V буде не меншою $s+1$.*

Теорема 9.5. Про перевірну матрицю лінійного (n,k)-коду

Якщо V – лінійний (n,k)-код з твірною матрицею $\mathbf{G}=[\mathbf{I}_k\mathbf{P}]\in M_{k,n}(\mathbf{E})$, де $\mathbf{I}_k=[\delta_{ij}]_k\in M_k(\mathbf{E})$, $\mathbf{P}\in M_{k,r}(\mathbf{E})$, то перевірну матрицю можна задати так:

$$\mathbf{H}=[\mathbf{P}^T\mathbf{I}_r]\in M_{r,n}(\mathbf{E}),\ \mathbf{I}_r=[\delta_{ij}]_r\in M_r(\mathbf{E}). \tag{9.17}$$

У розгорнутій формі згадані матриці мають вигляд:

$$\mathbf{G}=\begin{bmatrix}1&0&\cdots&0&p_{11}&p_{12}&\cdots&p_{1r}\\0&1&\cdots&0&p_{21}&p_{22}&\cdots&p_{2r}\\\cdots&\cdots&\cdots&\cdots&\cdots&\cdots&\cdots&\cdots\\0&0&\cdots&1&p_{k1}&p_{k2}&\cdots&p_{kr}\end{bmatrix}; \tag{9.18}$$

$$\mathbf{H}=\begin{bmatrix}p_{11}&p_{21}&\cdots&p_{k1}&1&0&\cdots&0\\p_{12}&p_{22}&\cdots&p_{k2}&0&1&\cdots&0\\\cdots&\cdots&\cdots&\cdots&\cdots&\cdots&\cdots&\cdots\\p_{1r}&p_{2r}&\cdots&p_{kr}&0&0&\cdots&1\end{bmatrix}. \tag{9.19}$$

Д о в е д е н н я

Насамперед зауважимо, що $rank(\mathbf{G})=k$, а $rank(\mathbf{H})=r$. Далі безпосередньо перевіряємо виконання рівностей (9.16):

$$\bigoplus_{l=1}^{n}h_{il}g_{jl}=\bigoplus_{l=1}^{k}h_{il}g_{jl}\oplus\bigoplus_{l=k}^{n}h_{il}g_{jl}=\bigoplus_{l=1}^{k}p_{li}\delta_{jl}\oplus\bigoplus_{l=k}^{n}\delta_{il}p_{jl}=$$
$$=p_{ji}\oplus p_{ji}=0,\ i=\overline{1,r},\ j=\overline{1,k}.$$

Блоковий лінійний (n,k)-код називають систематичним, якщо:

1) кодова послідовність має однозначне розміщення інформаційних та перевірних розрядів (наприклад, $\mathbf{u}=(a_1,a_2,\ldots,a_k,b_1,b_2,\ldots,b_r)$, $r=n-k$, де $a_1,a_2,\ldots,a_k$ – інформаційні розряди, $b_1,b_2,\ldots,b_r$ – перевірні);

2) перевірні елементи є сумою за модулем два обумовлених інформаційних елементів, тобто

$$b_j=\bigoplus_{l=1}^{k} a_l g_{lk+j}=\bigoplus_{l=1}^{k} a_l p_{lj},\ j=\overline{1,r},\ j=\overline{1,n-k}. \tag{9.20}$$

Розглянемо деякий код $\mathbf{u}\in\mathbf{E}^n$. Вектор $\mathbf{s}=\mathbf{H}\mathbf{u}^T\in\mathbf{E}^r$ називають *синдромом помилки*. Якщо $\mathbf{u}\in V$, то $\mathbf{s}=\mathbf{0}$. Покладемо $\mathbf{u}=\mathbf{v}^T\oplus\mathbf{e}_j^T$, де $v\in V$, а $\mathbf{e}_j\in\mathbf{E}^n$ має одиничний розряд лише у *j*-й позиції. Тоді синдром помилки співпадатиме з *j*-м стовпчиком матриці **H**:

$$\mathbf{s}=\mathbf{H}\left(\mathbf{v}^T\oplus\mathbf{e}_j^T\right)=\mathbf{H}\mathbf{v}^T\oplus\mathbf{H}\mathbf{e}_j^T=[h_{\cdot j}]\in\mathbf{E}^r. \tag{9.21}$$

Нехай ми отримали код $\mathbf{u}$ і підрахували синдром помилки $\mathbf{s}=\mathbf{H}\mathbf{u}^T$. Якщо $\mathbf{s}=\mathbf{0}$, то помилки немає. Коли $\mathbf{s}\neq\mathbf{0}$ і співпадає з *j*-м стовпчиком матриці **H**, то допущена помилка інверсії у *j*-му розряді.

Отже, якщо є однозначна відповідність між синдромами і матрицею **H**, то помилку можна виявити (і виправити).

9.5. Код Геммінґа

Розглянемо побудову коду Геммінґа, який забезпечує виправлення однієї помилки [22]. За теоремою 9.2, необхідною і достатньою умовою для цього є нерівність $d(V)\geq 3$.

Помилка заміщення може виникнути у будь-якому з n розрядів коду. Варто також врахувати ситуацію, коли її немає. Тому кількість контрольних розрядів r повинна задовольняти умову:

$$2^r\geq n+1. \tag{9.22}$$

Для випадку рівності отримаємо:

$$n=2^r-1,\ r=2,3,\ldots\ . \tag{9.23}$$

Нагадаємо, що 1956 року Р. В. Геммінґ запропонував розглядати матрицю **H** з $n = 2^r - 1$ стовпчиків висоти r, у яких розміщені двійкові числа від 1 до n. Очевидно, що сума за модулем два будь-яких двох стовпчиків цієї матриці не дорівнює нулю. Тоді, згідно з наслідком 9.1, нульовий простір цієї матриці має вагу Геммінґа рівну 3 і є кодом, який дає змогу виправляти всі одиничні помилки заміщення.

Він також запропонував розмістити перевірні розряди в місцях, які відповідають степеням двійки ($1 = 2^0, 2 = 2^1, 4 = 2^2, \ldots$).

Двійкові представлення 2^i, i = 0,1,… містять одиницю лише в i-му розряді.

Це дає змогу однозначно визначити перевірні розряди за допомогою співвідношення (9.14), оскільки в кожне рівняння входитиме лише один перевірний розряд.

Приклад 9.9. Розглянемо конкретний приклад для $n = 7,\ k = 4,\ r = 3$. Перевірну матрицю подамо у вигляді

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Загальне представлення кодової послідовності: $v = (b_1, b_2, a_1, b_3, a_2, a_3, a_4)$.

Відповідно до формули (9.14), $\bigoplus_{j=1}^{7} h_{ij} v_j = 0$, $i = \overline{1,3}$, звідки отримуємо:

$b_1 = a_1 \oplus a_2 \oplus a_4$, $b_2 = a_1 \oplus a_3 \oplus a_4$, $b_3 = a_2 \oplus a_3 \oplus a_4$.

Закодуємо послідовність $a = 1011$. Тоді $b_1 = 0$, $b_2 = 1$, $b_3 = 0$, в результаті матимемо $\mathbf{v} = 0110011$. Припустимо, що надійшло повідомлення $\mathbf{u} = 1110011$ з помилкою заміщення у першому розряді. Синдром помилки дорівнює $\mathbf{s} = \mathbf{H}\mathbf{v}^T = (0,0,1)^T$ – номер першого розряду, де і є помилка.

Вправи

1. Доведіть, що множина $\mathbf{E} = \{0,1\}$, на якій задані операції додавання за модулем два ($\oplus$) та множення, утворює поле.
2. Доведіть, що множина двійкових послідовностей довжини n – $\mathbf{E}^n$ утворює лінійний (векторний) простір над полем $\mathbf{E} = \{0,1\}$.

3. Розглянемо двійкові вектори $\mathbf{a,b,x,y} \in \mathbf{E}^n$. Доведіть, що рівності $\mathbf{a} \oplus \mathbf{b} = \mathbf{x} \oplus \mathbf{y}$ та $\mathbf{a} \oplus \mathbf{x} = \mathbf{b} \oplus \mathbf{y}$ – еквівалентні.
4. Перевірте загальні аксіоми відстані для відстані Геммінґа (9.3).
5. Доведіть властивості (9.5)–(9.7) ваги Геммінґа.
6. Знайдіть кодову відстань для кодової множини $V = \{\beta_1 = 000111, \beta_2 = 101000, \beta_3 = 111111\} \subset \mathbf{E}^6$.
7. Доведіть, що множина векторів $\mathbf{v} \in \mathbf{E}^n$, які задовольняють рівність (9.13), утворює підпростір простору $\mathbf{E}^n$.
8. Доведіть еквівалентність рівностей (9.13)– (9.15).
9. Виправте помилку у повідомленні $\mathbf{v} = 0100011$, закодованому кодом Геммінґа для $n = 7,\ k = 4,\ r = 3$.
10. Побудуйте код Геммінґа для $n = 15,\ k = 11,\ r = 4$.

Запитання та завдання

1. Назвіть основні типи помилок для двійкового кодування.
2. Які коди називають завадостійкими?
3. У чому полягає різниця між блочними і загортальними кодами?
4. Що таке лінійний (n,k)-код?
5. Що дає змогу виявити кодування з перевіркою на парність?
6. Сформулюйте означення відстані Геммінґа між двома кодами.
7. Що таке вага Геммінґа?
8. Що називають кодовою відстанню для коду?
9. Сформулюйте теорему про кодову відстань для коректуючих кодів.
10. Сформулюйте теорему про кодову відстань для лінійних кодів.
11. Сформулюйте теорему про твірну матрицю.
12. Який блоковий лінійний код називають систематичним?
13. Як розраховують синдром помилки?

Розділ 10. ОСНОВИ КРИПТОГРАФІЇ

10.1. Методи захисту інформації. Криптологія

Захист комп'ютерних даних від несанкціонованого доступу, спотворення та знищення є актуальною економічною, соціальною та технічною проблемою. Для її розв'язання використовують такі головні методи:

1. Фізична ізоляція комп'ютерної техніки та каналів передавання даних. Метод є надійним, однак не завжди можливим. Окрім того, він може істотно ускладнити роботу легальних користувачів.

2. Логічний бар'єр під час доступу до комп'ютерів і мереж, програм, даних. Полягає у використанні систем паролів, у розмежуванні прав доступу користувачів та ін. Однак цей метод не захищає дані на етапі їхнього передавання.

3. Шифрування даних – перетворення даних з метою їхнього захисту від несанкціонованого доступу.

4. *Стеганографія*, у перекладі з грецької – тайнопис. Мета стеганографії – приховати факт існування повідомлення. Головним методом комп'ютерної стеганографії є заміна шумових компонент інформаційними у графічних, звукових і відеофайлах.

Загалом інформаційну безпеку комп'ютерних систем та телекомунікаційних мереж називають *кібербезпекою*.

Зупинимось детальніше на найпоширенішому методі захисту даних – шифруванні, використовуючи праці [2; 5; 21; 34; 35].

Шифрування – це кодування даних з метою їхнього захисту від несанкціонованого доступу. Процес кодування у цьому випадку називають шифруванням або *криптуванням*. У результаті отримують *шифрограму* (*криптограму*, *криптотекст*). Обернену процедуру отримання повідомлення за криптотекстом називають *дешифруванням*. Алгоритми шифрування та дешифрування разом утворюють *криптосистему*, а простіше – *шифр*.

Схематично передавання шифрованого повідомлення подано на рис. 10.1.

Рис. 10.1. Передавання шифрованого повідомлення: M – повідомлення; C – криптотекст; K – ключ для шифрування; K' – ключ для дешифрування

Шифрування та дешифрування можна означити функціонально так:

$$C = E_K(M), \quad M = D_{K'}(C). \tag{10.1}$$

Класичні системи шифрування є *симетричними*, у них ключі для шифрування і дешифрування однакові ($K = K'$), або ключ K' легко визнається за ключем K.

Наголосимо, що головна мета шифрування – це забезпечення конфіденційності та автентичності даних (інформації). Під *конфіденційністю* розуміємо неможливість отримання вихідного тексту за криптотекстом без знання додаткової інформації – *ключа*. *Автентичність* полягає в істинності авторства і цілісності даних.

Створенням шифрів займається *криптографія*, а їхнім розкриттям – *криптоаналіз*. Це дві складові науки *криптології* (від грецьких слів κρυπτοζ – таємний і λογοξ – наука). Криптографія вивчає математичні методи створення шифрів, а криптоаналіз – методи розкриття шифрів без знання ключів.

Сучасна криптографія налічує чотири основні розділи:

1) симетричні криптосистеми;

2) криптосистеми з відкритим ключем;

3) системи електронного підпису;

4) методи управління ключами і криптографічні протоколи.

Як уже зазначено, у симетричних системах для шифрування і дешифрування використовують один і той самий ключ. У системах з відкритим ключем, натомість, використовують два ключі – відкритий і закритий (секретний).

Цифровий (*електронний*) підпис – це криптографічне перетворення тексту, яке приєднується до нього з метою підтвердження автентичності.

Управління ключами (адміністрування ключами) охоплює проблеми генерації ключів, цільового використання ключів, зберігання ключів, погодження ключів та обміну ключами між сторонами.

Зазначимо, що процедуру обміну повідомленнями між сторонами зазвичай регламентують протоколом. *Протокол* – це послідовність узгоджених приписів, яких дотримуються сторони задля досягнення певної мети. *Криптографічні протоколи* – окремий розділ криптографії, який вивчає протоколи обміну повідомленнями, протоколи погодження ключів та обміну ключами, протоколи цифрового підпису, протоколи ідентифікації сторін тощо [2; 35]. *Криптоаналіз* вивчає математичні методи порушення конфіденційності та автентичності даних без знання ключів. Окрім абонентів, які обмінюються інформацією, важливою стороною у криптоаналізі є зловмисник або *криптоаналітик* – особа чи група осіб, метою яких є розкриття криптограм чи визначення ключів. Спробу розкриття зашифрованого повідомлення, його підробки чи визначення ключа методами криптоаналізу називають *криптоатакою* або *атакою на шифр*. Вдалу криптоатаку називають *зломом*.

Спробу розкриття шифру шляхом перебору всіх ключів називають *брутальною атакою на шифр*.

Криптостійкість – це характеристика шифру, яка виражає його стійкість до криптоаналізу. Найуживаніші показники крипостійкості такі:

1. Ймовірність підбору ключа за вказаний час. Визначається кількістю можливих ключів. Наприклад, для шифру довжиною m біт потрібно перевірити щонайбільше $m!$ комбінацій.

2. Кількість операцій чи середній час, необхідний для злому шифру з заданою ймовірністю.

3. Вартість розкриття вихідного тексту чи ключа.

10.2. Короткий історичний огляд

Проблема захисту інформації шляхом її перетворення, яка виключає її розуміння іншою особою, хвилювала людство з давніх-давен. Криптографія – ровесниця писемності. Початково писемність сама по собі була криптографічною системою, оскільки нею володіли лише обрані.

Давньоєврейський *шифр атбаш* створений у V столітті до н. е. Це шифр простої заміни, коли літеру замінюють літерою з тим же номером, однак з кінця алфавіту. Римський імператор Юлій Цезар (100–44 р. до н. е.) використовував у своєму листуванні шифр зсуву, який полягає у заміні літери повідомлення іншою, що знаходиться в алфавіті через три позиції.

Криптоаналіз започаткували арабські дослідники у XV столітті, відкривши частотний аналіз текстів (докладніше далі).

В епоху Відродження в Європі створили чимало стійких до ручного криптоаналізу шифрів. Італійський архітектор і письменник Леон Батиста Альберті запропонував шифр поліалфавітної підстановки, варіант якого згодом назвали іменем дипломата XVI століття Блеза де Віженера. Перу Леона Альберті належить перша друкована праця з криптографії – "Трактат про шифр" (1466).

На початку XIX століття відкрили простий і стійкий до ручного криптоаналізу шифр "чотирьох квадратів", який використовує кодування біграмами. Цей шифр з успіхом застосовували до Першої світової війни.

Інженери телеграфної компанії "*AT&T*" Г. Вернам та М. Моборн (*G. Vernam, J. Mauborgne*) 1917 року винайшли шифр "одноразового блокнота", який, по-суті, є двійковою модифікацією шифру Віженера.

Для автоматизації криптування ще наприкінці XVIII століття винайшли роторні машини, які отримали значне поширення на початку XX століття. Найвідомішою з них була німецька "*Enigma*". Цю машину широко застосовували у німецькій армії протягом Другої світової війни. Проблеми криптоаналізу шифру цієї машини стимулювали англійських учених до розробки електронно-обчислювальних машин.

Становлення криптології як науки відбулося після виходу 1949 року праці Клода Шеннона "Теорія зв'язку секретних систем" [34]. У 40–50-ті роки остаточно сформувались розділи математики, що стали основою криптології – загальна алгебра, теорія ймовірностей і математична статистика, теорія чисел; набули активного розвитку теорія інформації, теорія алгоритмів, кібернетика.

Створення комп'ютерів докорінно змінило критерії ефективності та надійності шифрів. ***Шифр вважають надійним в обчислювальному сенсі, якщо час для його розкриття на комп'ютері перевищує час актуальності зашифрованої інформації.*** Прикладом надійного в обчислювальному сенсі шифру став прийнятий 1976 року в США стандарт шифрування *DES*.

Відкриття 1976 року американськими вченими В. Діффі та М. Гелманом (*W. Diffie, M. Hellman*) і незалежно від них Р. Мерклом (*R. Merkle*) криптування з відкритим ключем спровокувало справжню революцію у криптографії. Асиметрична криптографія відразу відкрила кілька прикладних напрямів, зокрема систему електронного цифрового підпису та електронних грошей.

10.3. Симетричні криптографічні системи

Симетричні криптографічні системи використовують такі головні методи:

- моно- і поліалфавітні підстановки;
- перестановки;
- блокове шифрування.

Найпростішим є метод шифрування підстановкою, який полягає у зміні літер вихідного тексту іншими у тому ж або іншому алфавіті за деяким правилом. *Шифри простої заміни* полягають у заміні кожної літери однією іншою. Ключем є таблиця відповідності літер. Наприклад, *шифр Юлія Цезаря* полягає у заміні літери іншою, яка знаходиться в алфавіті через три позиції.

Очевидно, що шифр простої заміни з n-символьним алфавітом має $n!$ ключів. У цьому випадку можливий *частотний криптоаналіз*. Пояснимо його суть.

Частота літери у тексті – це відношення кількості входжень цієї літери у текст до загальної кількості його літер. Емпірично з'ясовано, що для достатньо довгих загальних текстів частота окремих літер алфавіту не залежить від конкретного тексту, а вектор частот однозначно визначає алфавіт. Отож для шифрів простої заміни частотний аналіз криптотексту дає змогу відтворити вихідний текст. Частотний криптоаналіз, як уже зазначено, відкрили арабські дослідники у XV столітті.

У *голофонних* шифрах кожна літера вихідного повідомлення замінюється не єдиним символом, а довільним символом з деякої множини. Такі шифри вимагають складного частотного аналізу.

Поліграмні шифри використовують кодування декількох літер – поліграм (біграм, триграм і т. п.). Прикладом поліграмного шифру є *шифр "чотирьох квадратів"*, відкритий на початку XIX століття. Для латинського алфавіту залишають 25 літер, відкидаючи найменш уживану – "*j*". Ними заповнюють у різних порядках чотири квадрати розміром 5x5, які розміщують у кутах більшого квадрата (рис. 10.2).

Перед початком шифрування з повідомлення вилучають усі розділові знаки, пропуски і літеру "*j*". Першу літеру біграми, яку шифрують, відшукують у верхньому лівому квадраті, другу – у правому нижньому. Ці дві літери утворюють діагональ прямокутника, у двох інших вершинах якого розташовані літери шифру (рис. 10.2).

Криптоаналіз поліграмних шифрів проводять шляхом аналізу частот поліграм.

q	w	e	r	t		m	n	v	o	y
y	u	i	o	p		b	c	i	r	p
a	s	d	f	g		x	u	e	a	h
h	k	l	z	x		t	w	s	g	l
c	v	b	n	m		q	d	f	k	z

t	p	m	c	n		z	n	f	q	p
o	r	a	k	v		x	m	g	w	o
h	i	e	s	l		c	a	h	e	i
x	g	u	w	d		v	s	l	r	u
b	z	f	y	q		b	d	k	t	y

Рис. 10.2. Шифр "чотирьох квадратів"

Поліалфавітні шифри – це шифри заміни, в яких позиція літери у вихідному тексті визначає правило її заміни. Для прикладу розглянемо *шифр Віженера*, розроблений італійським архітектором і письменником Леоном Альберті у XV столітті.

Нехай алфавіт A має n літер, пронумерованих від 0 до $n-1$: $A=\{a_i,\ i=0,1,\ldots,n-1\}$. Сумою двох літер алфавіту вважають літеру, номер якої в алфавіті k дорівнює сумі номерів цих літер за модулем n:

$$a_i \oplus_n a_j = a_k,\ k = i \oplus_n j. \qquad (10.2)$$

Аналогічно означують різницю.

Вихідний текст розбивають на блоки довжиною m. Ключем є фраза у тому ж алфавіті довжини m. Шифрування полягає у додаванні за модулем m літер ключа до літер блоків тексту, а дешифрування – у відніманні ключа.

Шифрування і дешифрування зручно здійснювати за допомогою таблиці Віженера, яка містить усі можливі суми літер алфавіту. Фрагмент такої таблиці для української абетки показано на рис. 10.3.

Для прикладу, закодуємо фразу "зустрічсуботу", використовуючи як ключ слово "ключ":

```
  зустрічсуботу
⊕ ключключключк
  удпмбцхпдмммд
```

	а	б	в	г	ґ	д	е	є	...
а	а	б	в	г	ґ	д	е	є	...
б	б	в	г	ґ	д	е	є	ж	...
в	в	г	ґ	д	е	є	ж	з	...
г	г	ґ	д	е	є	ж	з	и	...
ґ	ґ	д	е	є	ж	з	и	і	...
д	д	е	є	ж	з	и	і	ї	...
е	е	є	ж	з	и	і	ї	й	...
є	є	ж	з	и	і	ї	й	к	...
...	...	...	...	...	...	...	...	...	...

Рис. 10.3. Таблиця Віженера

Прикладом реалізації шифрів поліалфавітної підстановки є *роторні криптувальні машини*, найвідоміша з них – німецька "*Enigma*" (рис.10.4).

Пояснимо принцип їхньої роботи узагальнено, опускаючи деталі конкретної реалізації.

Базовим елементом роторної машини є диск. З обох боків диска симетрично розміщені 25 електричних контактів, які відповідають літерам латинського алфавіту. Кожен контакт з одного боку диска внутрішньо з'єднаний з одним контактом по інший бік, тобто один диск фактично реалізує просту підстановку.

Диски (від трьох до п'яти) механічно з'єднані з різними передавальними числами з осьовим ротором і мають 25 фіксованих положень, коли контакти суміжних дисків з'єднані. Крайні лівий і правий диски контактують з 25-ма контактами, закріпленими на корпусі.

Рис. 10.4. Шифрувальна машина "*Enigma*"

Окрім ротора з дисками, машина має клавіатуру, подібну до друкарської машинки, панель з висвітлюваними літерами, акумулятор та інші додаткові механізми. Ключем є початкове положення дисків один відносно одного та передавальні числа дисків.

Після натиснення на клавішу з деякою літерою здійснюється поворот ротора, який зумовлює повороти дисків відповідно до їхніх передавальних чисел. Далі до нерухомого контакту з правого боку корпусу, який відповідає натиснутій літері, підводиться струм. Він проходить лише одним шляхом через замкнені контакти дисків і засвічує лампочку, яка відповідає коду літери. Код літери залежить від її порядку у повідомленні, тобто така машина реалізує шифр поліалфавітної підстановки. Дешифрування здійснюють аналогічно у зворотному напрямку.

Шифр Віженера для невеликих довжин блоків m допускає частотний криптоаналіз. Якщо m збігається з довжиною вихідного тексту, то такий шифр стає дуже стійким. Його називають *шифром одноразового блокнота* (*one-time pad*). Для дуже великих m такий шифр вважають абсолютно надійним у теоретико-інформаційному сенсі.

Кодування шифром Віженера зручно здійснювати у двійковій системі числення, з використанням операції додавання за модулем два (⊕, або *XOR – eXclusive OR* – виключне АБО). Шифрування полягає у додаванні за модулем два вихідного повідомлення у двійковій формі і ключа. Дешифрування здійснюють додаванням за модулем два того ж ключа. У такій формі цей шифр винайшли американські інженери Г. Вернам та Д. Моборн 1917 р.

Недоліками шифру одноразового блокнота є необхідність генерування великих послідовностей випадкових чисел та забезпечення надійного каналу передавання ключа.

Шифри перестановки зберігають усі літери вихідного тексту, але розміщують їх у іншому порядку.

Простий приклад шифру перестановки – *шифр частоколу*. Зашифруємо цим шифром слово "криптографія":

к^{и}п$_{т}$о^{г}р$_{а}$ф^{і}я – ігірпорфякта.

Ключем у цьому шифрі є висота частоколу.

Ще один приклад – *матричний шифр*. Літери повідомлення записують у певному порядку в комірки прямокутної таблиці (матриці). Криптотекст отримують зчитуванням літер з таблиці у зміненому порядку. Наприклад, повідомлення записують у рядках, а криптотекст зчитують зі стовпчиків. Послідовність, у якій зчитують стовпчики, визначають літери ключа (рис. 10.5)

Ш	И	Ф	Р	У	Й
6	1	5	3	4	2
Н	Е	Б	У	Д	І
Т	Ь	С	О	Б	А
К	У	Я	К	И	Й
С	П	И	Т	Ь	Ю

Рис. 10.5. Матричний шифр

У наведеному прикладі ключем є слово ШИФРУЙ, явним текстом – фраза "не будіть собаку який спить", а шифротекстом – послідовність ЕЬУПІАЙЮУОКТДБИЬБСЯИНТКС. Дешифрування за наявності ключа відбувається у зворотному порядку – послідовність записують у стовпчики, а зчитують з рядків.

Блокові симетричні шифри – це сім'я оборотних перетворень блоків – частин фіксованої довжини повідомлення. Фактично блоковий шифр – це система підстановок на алфавіті блоків.

Сьогодні блочні симетричні шифри доволі поширені на практиці. Перші американський і радянський стандарти шифрування (п.10.5) належать до цього класу шифрів.

Одним з способів реалізації блокових шифрів є використання мереж Файстеля, значний вклад у розробку яких зробив американський криптограф Горст Файстель (*Horst Feistel*).

Нехай n – довжина блоку – парне число, а k – довжина ключа K. Означимо нелінійну необоротну функцію $F:\mathbf{E}^{n/2}\times\mathbf{E}^{k}\to\mathbf{E}^{n/2}$, де $\mathbf{E}=\{0,1\}$.

Подамо вихідний блок даних X у вигляді двох підблоків однакової довжини $X=(A,B),\ A\in\mathbf{E}^{n/2},\ B\in\mathbf{E}^{n/2}$ і припишемо йому нульовий індекс $X^{(0)}=(A^{(0)},B^{(0)})$. Тоді один цикл мережі Файстеля визначається ітерацією:

$$X^{(i+1)}=(A^{(i+1)},B^{(i+1)}), \tag{10.3}$$

$$A^{(i+1)}=B^{(i)},\ \ B^{(i+1)}=A^{(i)}\oplus F(B^{(i)},K),\ \ \ i=0,1,\ldots,m-1. \tag{10.4}$$

Мережа Файстеля має фіксовану кількість циклів m, яку обирають з міркувань стійкості.

Найбільшою перевагою мережі Файстеля є той факт, що процедура шифрування та дешифрування співпадає, лише базову інформацію використовують у зворотному порядку, оскільки виконуються очевидні співвідношення:

$$B^{(i)}=A^{(i+1)},\ A^{(i)}=F(B^{(i)},K)\oplus B^{(i+1)},\ i=m-1,m-2,\ldots,0. \tag{10.5}$$

10.4. Криптосистеми з відкритим ключем

Як уже зазначено вище, ідею *криптування з відкритим ключем* (*Public Key Encryption, PKE*) висунули 1976 року американські вчені В. Діффі та М. Гелман і незалежно від них – Р. Меркл. За такого підходу процедура шифрування є загальнодоступною, а дешифрування – секретним.

У криптосистемах з відкритим ключем шифрування здійснюють за відомим алгоритмом з використанням відкритого ключа – K, а для дешифрування використовують секретний ключ K':

$$C=E_K(M),\ \ M=D_{K'}(C). \tag{10.6}$$

Такі криптосистеми називають *несиметричними* (*асиметричними*), на противагу класичним, у яких шифруючий та дешифруючий ключі – однакові ($K = K'$) або дешифруючий ключ легко визначити за шифруючим.

Надійність несиметричних криптосистем ґрунтується на тому факті, що задачі визначення повідомлення M або секретного ключа K' за відомим криптотекстом C та відкритим ключем K є дуже складними.

Зрозуміло, що відображення $C = E_K(M)$ є бієктивним та існує однозначне обернене відображення $E_K^{-1}(C)$, однак його знаходження в несиметричних криптосистемах – дуже складна задача в обчислювальному сенсі.

Протокол секретного спілкування між особами ***A*** та ***B*** буде таким:

Особа ***A*** продукує два ключі: секретний K'_A та відкритий K_A, який відкрито публікує. Аналогічно вчиняє особа ***B***.

Алгоритми шифрування з відкритим ключем та дешифрування з секретним ключем загальновідомі. Особа ***A*** шифрує своє повідомлення M_A, використовуючи відкритий ключ особи ***B*** – K_B, і передає отриманий криптотекст $C_A = E_{K_B}(M_A)$ відкритими каналами до особи ***B***. Дешифрувати це повідомлення може лише особа ***B***, яка має відповідний секретний ключ.

Передавання повідомлення особою ***B*** відбувається аналогічно.

Порівняно з класичним способом криптування, *PKE*-криптування має такі переваги:

1. Зникає проблема передавання і збереження відкритого ключа *K*. Цей ключ передають відкрито.

2. Спрощення секретного спілкування між *n* особами. За класичної схеми потрібно $n(n+1)/2$ ключів , а за *PKE*-схеми – лише *2n* ключів.

3. Можливість організації електронного підпису, якщо схема має додаткову властивість комутування ключів, коли шифруючий і дешифруючий ключі взаємозамінні: $M = D_{K'}(E_K(M)) = D_K(E_{K'}(M))$. Це додатково дає змогу криптувати повідомлення секретним ключем, а декриптувати – відкритим. У найпростішому варіанті цифровим підписом є послідовність M, $E_{K'}(M)$. Застосування відкритого ключа до другої частини повідомлення дає дубльоване повідомлення *M,* що стверджує автентичність документа.

Найпоширенішою системою криптування з відкритим ключем є запропонована 1977 року *система RSA* (*R. Rivest, A. Shamir, L. Adleman*). Ця схема базується на алгоритмах теорії чисел, а її стійкість ґрунтується на складності розв'язування задачі розкладу натуральних чисел на прості множники. Детальний виклад цієї криптосистеми подано у розділі 11.

Криптосистема Рабіна (*Michael Rabin,* 1979), як і криптосистема *RSA,* ґрунтується на складності задачі факторизації цілих чисел. Криптосистема Ель-Гамаля (*Taher Elgamal,* 1985) використовує складність задачі дискретного логарифмування у скінченному полі. Асиметричну криптосистему на основі еліптичних кривих незалежно запропонували 1985 року Н. Кобліц (*N. Koblitz*) і В. Міллер (*V. Miller*). Детальніше з цими та іншими асиметричними криптосистемами можна ознайомитися у працях [2; 5; 21; 35].

10.5. Стандарти шифрування даних

Стандарт шифрування даних *DES* (*Data Encryption Standard*) розроблений фірмою "*IBM*" близько 1970 р., а 1976 року його прийняли у США як стандарт для захисту комерційної та урядової інформації, не пов'язаної з національною безпекою. Це комбінований блоковий шифр, який використовує метод замін і перестановок – усього 16 циклів. Шифрування відбувається блоками по 64 біти. Довжина ключа також 64 біти, проте 56 з них – інформаційні, а 8 – контрольні. Реалізація *DES* доволі швидка (приблизно 1 Гбайт/с), що зумовило його широке використання на практиці.

Комп'ютер, збудований 1998 року, за три доби зміг розкрити ключ *DES* за парою "*текст – криптограма*". Це сприяло створенню надійніших шифрів – *FEAL, RC5, RC6* та інших [5].

З 1991 року функціонує міжнародний стандарт *International Data Encryption Standard* (*IDES*), який використовує 64-бітні блоки і 128-бітний ключ. Він є складовою частиною пакета захисту даних *Pretty Good Privacy* (*PGP*). А 2001 р. в США впроваджено новий стандарт *Advanced Encryption Standard* (*AES*). Цей стандарт використовує блоки і ключ довжиною 128, 196 і 256 біт.

Радянський стандарт шифрування регламентований ГОСТ 28147-89. Цей блоковий 64-бітний шифр з 256-бітним ключем побудований з використанням мереж Файстеля. Стандарт також застосовують в Україні. Його перевидали 2009 року під назвою ДСТУ ГОСТ 28147:2009, однак сьогодні він уже не повністю відповідає сучасним вимогам до швидкодії та криптостійкості.

З 1 липня 2015 року в Україні введено в дію новий стандарт шифрування ДСТУ 7624:2014 "Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного блокового перетворення", який використовує шифр "Калина". Його розроблено для заміни ДСТУ ГОСТ 28147:2009. А новий національний стандарт шифрування коротких повідомлень (до 616 біт) ДСТУ 9041:2020 "Інформаційні технології. Криптографічний захист інформації. Алгоритм шифрування коротких повідомлень, що ґрунтується на скручених еліптичних кривих Едвардса" запроваджено 2020 року.

Перелік актуальних стандартів і технічних специфікацій, дозволених для реалізації в засобах криптографічного захисту інформації, регламентовано наказом Адміністрації Державної служби спеціального зв'язку та захисту інформації України № 687 від 27 жовтня 2020 року (https://zakon.rada.gov.ua/laws/show/z1272-20#Text)

10.6. Гешувальні функції

Гешувальною (*вкорочувальною*) *функцією* (*hash-function*) називають відображення, яке перетворює послідовність даних M довільної довжини у послідовність $H(M)$ сталої довжини і має спеціальні властивості.

Нехай $A=\{a_1,a_2,\ldots,a_n\}$ – алфавіт повідомлення, $B=\{b_1,b_2,\ldots,b_m\}$ – алфавіт кодування, найчастіше $A=B=\{0,1\}$. Позначимо через A_L^* множину всіх слів, які можна скласти з алфавіту A і довжина яких не перевищує L, а через B_l^0 – множину всіх слів з алфавіту B довжиною l, $l \ll L$, зокрема $l=128,160,192,256,384,512$.

Геш-функцією називають відображення $H: A_L^* \to B_l^0$, яке має такі властивості:

1. Функція $H(M)$ визначена для повідомлень довільної довжини (порядку мегабайта і більше) і швидко обчислюється на комп'ютері.

2. Задача побудови оберненого відображення є складною, тобто для будь-якого $Y \in B_l^0$ важко знайти таке $X \in A_L^*$, щоб виконувалась рівність $Y=H(X)$.

3. Відображення $H: A_L^* \to B_l^0$ стійке щодо виникнення колізій, тобто для будь-якого $X \in A_L^*$ важко знайти таке $X' \in A_L^*$, щоб виконувалася рівність $H(X)=H(X')$.

З умови 2 випливає, що відновити вихідне повідомлення за його геш-функцією можна лише шляхом перебору всіх можливих повідомлень, що практично неможливо, а з умови 3 – що колізія, коли різним повідомленням відповідає одна геш-функція, малоймовірна.

Гешувальну функцію ще називають згорткою повідомлення. Можна сказати, що ця функція так характеризує вихідний текст, як людину відбитки пальців чи сітківка ока.

Геш-функції використовують у криптографії для зберігання паролів, засвідчення документів, організації цифрових підписів.

На відміну від пароля, значення геш-функції пароля можна зберігати відкрито, оскільки вона – необоротна. Тому протокол доступу до операційної системи чи мережі можна організувати так:

1. У системі зберігають геш-функції паролів користувачів $H(X_i)$.

2. Користувач $\boldsymbol{A}_i$ задає своє ім’я і пароль X_i.

3. Система обчислює $H(X_i)$ і порівнює зі зразком для користувача $\boldsymbol{A}_i$. Саме значення пароля X_i видаляється з пам’яті.

4. Якщо значення геш-функції співпадає зі зразком, то відкривається сесія для користувача $\boldsymbol{A}_i$.

З метою "нотаріального" засвідчення власності на документ X потрібно обчислити значення деякої відомої геш-функції $H(X)$ та опублікувати його або розмістити у нотаріуса. Якщо необхідно довести власність на документ, достатньо навести X, і зазначити місце, де було раніше поміщено $H(X)$. Підтвердження власності на документ полягає в обчисленні значення $H(X)$ для тексту X і порівнянні його з раніше опублікованим або депонованим у нотаріуса.

Гешувальний алгоритм легко отримати за допомогою будь-якого стандарту шифрування (наприклад, *RSA*). Вихідне повідомлення X розбивають на блоки прийнятої для цього стандарту довжини $X = X_1 X_2 \ldots X_k$. Далі здійснюють рекурентне шифрування за схемою:

$$H_1 = RSA(X_1),\ H_2 = RSA(H_1 \oplus X_2),\ \ldots,\ H = H_k = RSA(H_{k-1} \oplus X_k).$$

Поширеними алгоритмами гешування є *MD2, MD4, MD5* (*Message Digest Algorithm*) з вихідною довжиною 128 біт. Стандарт США 1992 року *SHA-1* (*Secure Hash Algorithm*) дає вихідну послідовність довжиною 160 біт. У Європі поширений стандарт гешування *RIPEMD-160*. У російському стандарті гешування ГОСТ Р 34.11-94 довжина вихідної послідовності дорівнює 256 біт.

З 1 квітня 2015 року в Україні введено стандарт ДСТУ 7564:2014 "Інформаційні технології. Криптографічний захист інформації. Функція гешування" з рекомендованими довжинами послідовності 256, 384 та 512 біт.

Детальний опис стандартів гешування можна відшукати у мережі "Інтернет".

10.7. Цифровий підпис

Цифровий підпис є електронним відповідником традиційного підпису. Він стверджує автентичність документа і повинен задовольняти такі властивості:

1. Лише особа ***A*** може створити підпис особи ***A*** (підробка підпису – неможлива).

2. Можна однозначно стверджувати, що підпис зроблено саме під оригінальним документом (спотворення документа або копіювання підпису з одного документа на інший неможливі).

Найпростіше електронний підпис реалізувати за допомогою несиметричних схем криптування (наприклад, *RSA*).

Нехай особа ***A*** має відкритий (публічний) ключ K_A і секретний ключ K'_A, а схема шифрування має властивість комутування ключів, коли шифруючий і дешифруючий ключі взаємозамінні: $M = D_{K'}(E_K(M)) = D_K(E_{K'}(M))$.

Електронний підпис особи ***A*** полягає у приєднанні до повідомлення M криптограми $E_{K'_A}(M)$: $M, E_{K'_A}(M)$; тут кома позначає операцію конкатенації. Застосування відкритого ключа K_A до другої частини повідомлення дасть вихідне повідомлення. Це ствердить автентичність повідомлення M:

1) підпис належить особі ***A***, оскільки лише вона має ключ K'_A;

2) підпис поставлений власне під повідомленням *M*, оскільки $D_{K_A}(E_{K'_A}(M))$ співпадає з *M*, що неможливо у випадку заміни повідомлення M іншим.

Наведений приклад є прикладом відкритого підписаного повідомлення. Якщо підписане повідомлення призначене лише для адресата ***B***, то необхідне додаткове криптування відкритим ключем особи ***B***: $E_{K_B}\left(M, E_{K'_A}(M)\right)$.

Якщо повідомлення M довге, то такий спосіб організації електронного підпису подвоює кількість даних. У цьому випадку можна використати геш-функцію, тоді підписане повідомлення матиме вигляд: $M, E_{K'_A}(H(M))$.

Поширені стандарти цифрового підпису описано в [5]. В Україні чинний стандарт ДСТУ 4145-2002 "Інформаційні технології. Криптографічний захист інформації. Електронний цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевіряння".

10.8. Адміністрування ключами

Симетричні криптосистеми потребують надійного каналу для обміну таємними ключами. В криптосистемах з відкритим ключем проблема передавання ключів відсутня, оскільки таємні ключі є власністю сторін, а відкриті ключі – загальнодоступні. Однак з появою цих систем симетричні криптосистеми не вийшли з ужитку, оскільки вони значно швидші. Окрім того, асиметричні системи відкрили нові можливості для передавання ключів відкритими каналами.

Головні проблеми адміністрування ключами:

- генерування ключів;
- розподіл ключів (які ключі використовувати і з якою метою);
- безпечне зберігання ключів;
- узгодження і передавання ключів партнерам.

Для адміністрування ключами найчастіше використовують спеціальний криптографічний пристрій. *Криптографічний пристрій* – це електронний пристрій, наприклад, спеціальна карта чи мікросхема. Він закритий у спеціальній скриньці, яка забезпечує миттєве знищення інформації за умови фізичного проникнення, захищена від проникнення електромагнітних хвиль та внутрішнього випромінювання, надійно зберігає головний ключ, який вводять лише один раз і який неможливо зчитати.

Вирізняють такі категорії ключів.

Головний ключ. Не змінюваний, його неможливо зчитати. Зберігається у криптографічному пристрої.

Первинні ключі – ключі для забезпечення зв'язку та зберігання даних. Вони генеруються за допомогою головного і часто змінюються, оскільки збільшення кількості даних, шифрованих одним ключем, полегшує криптоаналіз. Генерування ключів не є складною проблемою для симетричних ключів. Воно вимагає застосування генератора випадкових чисел.

Вторинні ключі. Необхідні з технічного погляду. Їх використовують у різних протоколах, наприклад, для узгодження первинних ключів перед початком сеансу обміну даними.

Сесійні ключі – це первинні ключі, призначені для окремої сесії зв'язку. Часта зміна ключів зменшує кількість даних, шифрованих одним ключем, а також обмежує час для розкриття шифру.

Найпростіший *протокол обміну ключем* між сторонами формулюють так:

1) сторона ***A*** обирає випадкову послідовність R_A і передає її безпечним шляхом стороні ***B***;

2) своєю чергою, сторона ***B*** обирає випадкову послідовність R_B і передає її безпечним шляхом стороні ***A***;

3) сторони незалежно обчислюють геш-функцію $H(R_A, R_B)$, яка слугує ключем.

Обидві сторони однаково відповідальні за ключ, однак цей протокол вимагає безпечного каналу передавання.

Розглянемо протокол обміну ключем X між сторонами ***A*** та ***B***, який використовує асиметричне криптування. Цей протокол полягає у надсиланні ключа як повідомлення, криптованого асиметричним способом:

1) сторона ***B*** відкрито надсилає стороні ***A*** свій публічний ключ K_B;

2) сторона ***A*** продукує сесійний ключ X;

3) сторона ***A*** шифрує ключ X за допомогою ключа K_B і надсилає повідомлення $E_{K_B}(X)$ стороні ***B***, сторона ***A*** може підписати це повідомлення, попередньо оголосивши свій відкритий ключ;

4) сторона ***B*** дешифрує повідомлення своїм закритим ключем K'_B й отримує ключ X.

Цей протокол допускає атаку "людина всередині" ("*man in the middle*"), якщо сторона А у п. 3 протоколу не підпише повідомлення.

Суть цієї атаки полягає у підміні зловмисником публічного ключа сторони ***B***. Нехай ***C*** – зловмисник, який перебуває між сторонами ***A*** та ***B*** і може перехоплювати їхні повідомлення. Опишемо сценарій отримання ключа X зловмисником:

1) зловмисник перехоплює публічний ключ K_B, замінює його на K_C і пересилає стороні ***A***;

2) сторона ***A*** шифрує ключ X за допомогою ключа K_C і надсилає повідомлення $E_{K_C}(X)$ стороні ***B***;

3) зловмисник ***C*** перехоплює повідомлення $E_{K_C}(X)$, дешифрує його за допомогою свого секретного ключа K'_C та отримує ключ X;

4) за допомогою ключа K_B зловмисник шифрує повідомлення $E_{K_B}(X)$ і надсилає його стороні ***B***.

У результаті зловмисник ***C*** отримує сесійний ключ X, однак сторони ***A*** та ***B*** цього не зауважують. Підпис шифрованого ключа таємним ключем сторони ***A*** у п. 3 протоколу не дозволить стороні ***C*** підписати повідомлення після підміни ключа.

Протоколи адміністрування ключами – це лише один з видів криптографічних протоколів, які вивчає окремий розділ криптографії [2].

10.9. Криптоаналіз. Теоретична та практична стійкість шифру

Як уже зазначено, криптоаналіз вивчає математичні методи порушення конфіденційності та автентичності даних без знання ключів.

Криптостійкість – це характеристика шифру, яка виражає його стійкість до криптоаналізу. Це поняття є центральним для криптології. Хоча якісно зрозуміти його доволі легко, проте отримання строгих доказових оцінок стійкості для кожного конкретного шифру – невирішена проблема.

Важливим моментом для визначення стійкості шифру є принцип Керкгоффса (*Auguste Kerckhoffs*, 1883), який формулюють так [21; 35]: "Надійність шифрування має залежати лише від секретності ключа і не залежати від алгоритмів шифрування та дешифрування". Для визнання цього принципу є серйозні підстави:

1. Алгоритми криптування є складними, вимагають громіздкої програмної та апаратної реалізації. Один і той самий алгоритм використовують тривалий час (їх важко зберігати у секреті).

2. Одним алгоритмом користується багато користувачів, що забезпечує його додаткове тестування.

Відповідно до цього принципу, засекречують лише деяку частину параметрів алгоритму, яку називають ключем шифру.

Використання принципу Керкгоффса дає змогу отримати деякі переваги у побудові шифрів:

- розголошення конкретного шифру (алгоритму та ключа) не означає необхідності повної заміни реалізації алгоритму: достатньо замінити лише скомпрометований ключ;
- ключі зберігають окремо від реалізації алгоритму у надійнішому місці і завантажують, за необхідності, на час виконання шифрування, що значно підвищує надійність системи загалом;
- невизначеність алгоритму шифрування повністю визначається ентропією використовуваного ключа – $H(E_K) = H(K)$.

Питання теоретичної стійкості шифрів уперше сформульоване Клодом Шенноном у його знаменитій праці "Теорія зв'язку у секретних системах" [34]: "Наскільки надійна криптосистема, якщо криптоаналітик супротивника має в своєму розпорядженні необмежений час і всі необхідні засоби для аналізу криптограм?" З цим питанням тісно пов'язане ще одне питання: "чи існують шифри, які не міг би розкрити криптоаналітик, що має в своєму розпорядженні як завгодно велику криптограму і необмежені обчислювальні ресурси?"

Теоретично існує абсолютно стійкий шифр. Клод Шеннон за допомогою розробленого ним теоретико-інформаційного методу дослідження стійкості шифрів отримав необхідну умову абсолютної стійкості шифру: для того, щоб шифр був абсолютно стійким, необхідно, щоб невизначеність алгоритму шифрування була не меншою від невизначеності шифрованого повідомлення:

$$H(E_K) \geq H(T). \tag{10.6}$$

Максимально можливої невизначеності блока даних фіксованого розміру досягають, коли всі можливі значення цього блока рівноймовірні – в цьому випадку вона дорівнює розміру блока в бітах (див. підрозділ 3.2). Отже, невизначеність ключа K не перевищує його довжини $|K|$:

$$|K| \geq H(K). \tag{10.7}$$

Приходимо до необхідної умови абсолютної стійкості для шифрів, що задовольняють принципу Керкгоффса:

$$|K| \geq H(K) = H(E_K) = H(E) \geq \max H(T) = |T|. \tag{10.8}$$

Отож побудований за принципом Керкгоффса шифр буде абсолютно стійким, якщо розмір використаного для шифрування ключа буде не меншим, ніж розмір шифрованих даних, тобто

$$|K| \geq |T|. \tag{10.9}$$

Прикладом абсолютно стійкого шифру може слугувати одноразова гама Вернама – накладення на відкриті дані *T* ключа *K* такого ж розміру, складеного зі статистично незалежних бітів, що приймають можливі значення з однаковою ймовірністю, за допомогою деякої бінарної операції °:

$$T' = T \circ K . \qquad (10.10)$$

З зазначеного вище випливає загальний висновок: для абсолютної стійкості істотною є кожна з таких вимог:

- цілковита випадковість (рівноймовірність) ключа (це, зокрема, означає, що ключ не можна виробляти за допомогою будь-якого детермінованого пристрою);
- рівність довжини ключа і довжини відкритого тексту (або ключ більший від довжини відкритого тексту);
- одноразовість використання ключа.

У разі порушення хоча б однієї з цих умов шифр перестає бути абсолютно стійким і з'являються принципові можливості для його розкриття (хоча вони можуть бути такими, що важко реалізовуються).

Отож практичну стійкість конкретного шифру оцінюють шляхом можливих спроб його розкриття; вона залежить від кваліфікації криптоаналітиків, що атакують шифр. Останню процедуру називають перевіркою стійкості.

Найуживаніші практичні показники криптостійкості такі:

1. Ймовірність підбору ключа за вказаний час. Визначається кількістю можливих ключів. Наприклад, для шифру довжиною m біт потрібно перевірити щонайбільше 2^m комбінацій.

2. Кількість операцій чи середній час, необхідний для злому шифру з заданою ймовірністю.

3. Вартість розкриття вихідного тексту чи ключа.

Важливим підготовчим етапом для перевірки стійкості шифру є моделювання різних можливостей і методів, за допомогою яких зловмисник може атакувати шифр.

Зазвичай вважають, що зловмиснику відомий шифр і він має змогу його попередньо вивчати. Зловмисник також знає деякі характеристики відкритих текстів (інформації, яку захищають), наприклад: загальну тематику повідомлень, їхній стиль, деякі стандарти, формати і т. д.

З дещо специфічніших наведемо ще три приклади можливостей зловмисника і відповідні види атак [21]:

1) зловмисник може перехоплювати шифровані повідомлення, проте не має їхніх відповідних відкритих текстів (лише закритий текст);

2) зловмисник може перехоплювати шифровані повідомлення і здобувати деякі їхні відповідні відкриті тексти (відомий відкритий текст);

3) зловмисник може зашифровувати будь-який відкритий текст (обраний відкритий текст).

Загалом для блокових шифрів криптостійкість визначається розміром ключа. Для сучасного рівня безпеки рекомендують ключ довжиною 256 біт і більше [21].

Запитання та завдання

1. Назвіть основні методи захисту інформації.
2. Вкажіть основні ідеї шифрування.
3. Що таке стеганографія?
4. Назвіть основні складові криптосистеми.
5. Опишіть схему передавання шифрованого повідомлення.
6. Охарактеризуйте симетричні та несиметричні криптосистеми.
7. Яке значення має термін "конфіденційність повідомлення"?
8. Яке значення має термін "автентичність повідомлення"?
9. Охарактеризуйте предмет криптографії та криптоаналізу.
10. Що вивчає криптологія?
11. Назвіть основні розділи криптографії.
12. Опишіть схему організації цифрового підпису.
13. Що таке брутальна атака на шифр?
14. Назвіть найуживаніші оцінки криптостійкості.
15. Яка ідея давньоєврейського шифру "атбаш"?
16. Опишіть шифр Юлія Цезаря.
17. Назвіть приклади шифру простої підстановки.
18. До якого класу шифрів належить матричний шифр?
19. Яка ідея частотного аналізу криптотексту?
20. Що таке поліграмні шифри?
21. Як реалізований шифр Віженера?
22. Вкажіть переваги і недоліки шифру одноразового блокнота.
23. Опишіть принцип дії роторних криптографічних машин.
24. Як здійснюють реалізацію блокових шифрів на основі мереж Файстеля?
25. Опишіть принцип криптосистем з відкритим ключем.

26. Назвіть найпоширеніші стандарти шифрування.
27. Що таке геш-функції?
28. Як використовують геш-функції для збереження паролів?
29. Як здійснюють нотаріальне засвідчення документів з використанням геш-функцій?
30. Опишіть алгоритм гешування за допомогою алгоритму *RSA*.
31. Як організують цифровий підпис з використанням криптосистеми *RSA*?
32. Опишіть протокол цифрового підпису відкритого повідомлення.
33. Опишіть протокол цифрового підпису закритого повідомлення.
34. Які основні задачі адміністрування ключами?
35. Вкажіть основні типи ключів.
36. Сформулюйте найпростіший протокол обміну ключами між сторонами.
37. Опишіть протокол обміну ключем з використанням асиметричних криптографічних алгоритмів.
38. У чому полягає криптоатака "*man in the middle*"?

Розділ 11. КРИПТОСИСТЕМА *RSA*

Широковживаною системою криптування з відкритим ключем є система *RSA*, яку запропонували 1977 року Рональд Рівест (*Ronald Rivest*), Аді Шамір (*Adi Shamir*) та Леонард Адлеман (*Leonard Adleman*). Ця схема базується на алгоритмах теорії чисел, а її стійкість ґрунтується на складності розв'язування задачі розкладу натуральних чисел на прості множники. Викладемо алгоритм *RSA* на основі праць [2; 5; 21; 35] .

11.1. Подільність чисел. Алгоритм Евкліда

Щодо подільності цілих чисел очевидним є таке твердження [1]:

Теорема 11.1. Про подільність цілих чисел

$\forall a \in \mathbf{Z}\ \forall b \in \mathbf{N}\ \exists! q \in \mathbf{Z}\ \exists! (r \in \mathbf{N}^+, r < b)\ a = qb + r,$

де a *– ділене;* b *– дільник;* q *– частка від ділення;* r *– остача від ділення.*

Введемо деякі позначення.

Означення 11.1. *Остачу від ділення цілого числа* a *на натуральне* b *називають (зведеним) лишком числа* a *за модулем* b *і позначають* $r = a \bmod b$.

Якщо a ділиться без остачі на b ($r = 0$), то скорочено записують a⋮b (a ділиться на b), або $b \mid a$ (b ділить a).

Означення 11.2. *Найбільшим спільним дільником* (НСД) *двох натуральних чисел називають число* $\text{НСД}(a,b) = \max\{c \in \mathbf{N} \mid c \mid a \wedge c \mid b\}$.

Означення 11.3. *Два натуральні числа називають взаємно простими, якщо їхній найбільший спільний дільник дорівнює одиниці. В цьому випадку вживають позначення* $a \perp b$.

Означення 11.4. *Натуральне число називають простим, якщо його дільниками є лише одиниця і воно саме (має лише тривіальні дільники). Множину всіх простих чисел позначають* $\mathbf{\Pi}$.

Ще давньогрецький математик Евклід (3 ст. до н. е.) запропонував алгоритм знаходження НСД двох натуральних чисел, який ґрунтується на двох очевидних рівностях:

$$\forall a,b \in \mathbf{N},\ a \geq b \quad \Rightarrow \text{НСД}(a,b) = \text{НСД}(b, a \bmod b); \tag{11.1}$$

$$\forall c \in \mathbf{N} \quad \text{НСД}(c,0) = c. \tag{11.2}$$

Запишемо цей алгоритм:

$i := 0\ ;\ r_0 := a;$

$i := 1\ ;\ r_1 := b;$

repeat

$i := i+1;\ r_i := r_{i-2} \bmod r_{i-1};\ \{\ r_{i-2} = q_i r_{i-1} + r_i,\ q_i \in N\ \}$ (11.3)

until $(r_i = 0);$

$\text{НСД}(a,b) := r_{i-1}$.

Очевидним є виконання нерівностей:

$a \geq b > r_1 > r_2 > \ldots > r_{i-1} > r_i \geq 0, i \geq 2,$

звідки отримаємо, що алгоритм збігається за скінченну кількість кроків $m \leq a$.

Цю оцінку легко посилити і показати, що

$$m \leq \min\left\{2\lceil \log_2 a \rceil, 2\lceil \log_2 b \rceil + 1\right\}. \tag{11.4}$$

Д о в е д е н н я ґрунтується на нерівності

$$r_i < \frac{1}{2} r_{i-2}, \tag{11.5}$$

яка випливає з загальної нерівності

$$a \geq b \Rightarrow r = a \bmod b < \frac{1}{2} a. \tag{11.6}$$

Послідовно використовуючи нерівність (11.6), отримаємо:

$$r_0 = a,\ r_1 = b;$$

$$r_2 < \frac{1}{2} a,\ r_3 < \frac{1}{2} b;$$

...

$$r_{2k} < \frac{1}{2^k} a,\ r_{2k+1} < \frac{1}{2^k} b.$$

Достатня умова для завершення алгоритму $\left(r_m = r_{2k} < a/2^k \le 1\right) \vee \left(r_m = r_{2k+1} < b/2^k \le 1\right)$ дає оцінку (11.4).

Приклад 11.1. НСД(211,79) = НСД(79,53) = НСД(53,26) = НСД(26,1) = = НСД(1,0).

З алгоритму Евкліда випливає такий наслідок:

Наслідок 11.1. *Для довільних взаємно простих натуральних чисел* $a \in N$ *і* $b \in N$ *існують такі цілі числа* $u, v \in Z$ *, що є розв'язками рівняння*

$$ua + vb = 1. \tag{11.7}$$

Д о в е д е н н я

Оскільки a, b – взаємно прості, то $\text{НСД}(a,b) = 1$. Отже, на передостанньому кроці алгоритму Евкліда ми отримаємо $r_{m-1} = r_{m-3} \bmod r_{m-2} = 1$, або

$$1 = r_{m-3} - q_{m-1} r_{m-2},\ q_{m-1} \in N. \tag{11.8}$$

Далі, послідовно підставляючи в праву частину рівності (11.8) замість лишку з більшим індексом його вираз через лишки з меншими індексами (11.3), прийдемо до виразу (11.7).

Отже, алгоритм Евкліда легко узагальнюється для знаходження розв'язків діофантового рівняння (11.7). Таку його модифікацію називають розширеним алгоритмом Евкліда. Одне з поширених застосувань – знаходження оберненого елемента за деяким модулем.

11.2. Обчислювальна складність алгоритмів

Обчислювальна складність алгоритму [2; 5; 9] визначається певними параметрами, серед яких найважливішими є час роботи та затрати ресурсів, насамперед *оперативної пам'яті* (ОП). Для алгоритмів реального часу час роботи є визначальним показником ефективності, а, отже, – складності.

З іншого боку, час роботи алгоритму пропорційний кількості операцій: арифметичних, логічних, обміну та ін. Швидкість виконання операцій різних типів неоднакова.

У багатьох алгоритмах переважають однотипні операції, виконання яких займає основний час роботи. Їх приймають за елементарні при оцінці складності алгоритму. Наприклад, складність алгоритмів розв'язування систем лінійних алгебричних рівнянь визначають за кількістю операцій множення та ділення з плаваючою крапкою.

Якщо всі операції зводяться до послідовності елементарних операцій (наприклад, побітових операцій над двійковими числами заданої довжини, яка дорівнює розрядності процесора), то показником складності алгоритму може бути загальна кількість цих елементарних операцій.

Кількість елементарних операцій алгоритму може залежати від багатьох вхідних параметрів, у найпростішому випадку – одного. Наприклад, у багатьох алгоритмах теорії чисел кількість операцій визначається одним вхідним параметром – натуральним числом n (більше з двох чисел, для яких шукають НСД; число, яке факторизують або перевіряють на простоту тощо).

Довжина коду числа n у двійковій системі числення дорівнює $k = \lfloor \log_2 n \rfloor + 1$. Вона є визначальною за оцінки кількості операцій алгоритмів теорії чисел. Наприклад, коли розмірність суматора дорівнює s, то додавання двох великих цілих чисел $a \in \mathbf{N}$ та $b \in \mathbf{N}$ $(a > b)$ вимагає $(\lfloor \log_2 a \rfloor + 1)/s$ елементарних операцій додавання.

Легко бачити, що операція множення (ділення) двійкових чисел a та b довжини k вимагає k операцій додавання чисел довжини k.

Означення 11.5. *Алгоритм має поліноміальну складність на вході довжиною* $k = \lfloor \log_2 n \rfloor + 1$*, якщо кількість елементарних операцій, необхідних для його виконання, обмежена величиною* Ck^d*, де* $C > 0, d > 1$ *– константи.*

Означення 11.6*. Алгоритм має експонентну складність, якщо кількість елементарних операцій перевищує величину* $C\exp(k^{\alpha}), C > 0, \alpha > 0$.

Алгоритм Евкліда знаходження НСД(a,b) ($a \geq b$) використовує менше ніж $O(\ln a)$ циклів. У кожному з них виконується ділення цілих чисел, яке вимагає $O(\ln a)$ додавань великих чисел, кожне з яких зводиться до $O(\ln a)$ елементарних додавань. Це дає загальну оцінку кількості елементарних операцій алгоритму Евкліда для великих чисел – $O(\ln a)^3$. Отже, алгоритм Евкліда має поліноміальну складність.

Алгоритми, які мають експонентну складність, вважають дуже повільними. Такими є відомі сьогодні алгоритми факторизації натуральних чисел. На межі можливостей сучасних комп'ютерів є факторизація чисел порядку 10^{150}.

11.3. Кільце зведених лишків за модулем n

Означення 11.7. *Цілі числа* x *та* y *називають рівними, або конгруентними за модулем* $n \in \mathbf{N}$*, якщо* $x \bmod n = y \bmod n$*. Це записують так:* $x \equiv y \pmod n$.

Наступні три умови є еквівалентними:

1) $x \equiv y(\mathrm{mod} n)$;

2) $\exists k \in \mathbf{Z} \;\; x = y + kn$;

3) $n \mid (x - y)$.

Головні властивості конгруенцій:

1) відношення конгруенції є відношенням еквівалентності – рефлексивне, симетричне і транзитивне;

2) конгруенції можна почленно додавати і почленно множити:

$$x_1 \equiv y_1(\mathrm{mod} n),\; x_2 \equiv y_2(\mathrm{mod} n) \;\Rightarrow$$

$$\Rightarrow\; x_1 + x_2 \equiv (y_1 + y_2)\mathrm{mod} n,\; x_1 \times x_2 \equiv (y_1 \times y_2)\mathrm{mod} n.$$

На основі цього вводять операції додавання та множення за модулем n:

$$x \oplus_n y = (x \mathrm{mod} n + y \mathrm{mod} n)\mathrm{mod} n = (x + y)\mathrm{mod} n; \qquad (11.9)$$

$$x \otimes_n y = (x \mathrm{mod} n \times y \mathrm{mod})\mathrm{mod} n = (x \times y)\mathrm{mod} n. \qquad (11.10)$$

Відношення конгруенції за модулем $n \in \mathbf{N}$ розбиває всі цілі числа на n класів, які можна ототожнити з множиною $Z_n = \{0,1,\ldots,n-1\}$.

Легко показати, що $\langle Z_n, \oplus_n \rangle$ утворює комутативну (абелеву) групу, а $\langle Z_n, \oplus_n, \otimes_n \rangle$ – комутативне кільце з одиницею.

Означення 11.8. *Комутативне кільце з одиницею* $\langle Z_n, \oplus_n, \otimes_n \rangle$ *називають кільцем зведених лишків за модулем* n.

Позначимо через Z_n^* множину елементів Z_n, які мають обернені в Z_n:

$$Z_n^* = \left\{ z \in Z_n \,\middle|\, \exists z' \in Z_n,\; z' \cdot z \equiv 1(\mathrm{mod} n) \right\}.$$

Легко бачити, що Z_n^* утворює комутативну групу відносно множення за модулем n.

Теорема 11.2. Про структуру кільця зведених лишків

Комутативна група Z_n^* *складається з елементів* $z \in Z_n$ *взаємно простих з* n *і лише з них:* $Z_n^* = \{ z \in Z_n \mid \mathrm{НСД}(z,n) = 1 \}$.

Д о в е д е н н я

Необхідність. Оскільки $\text{НСД}(z,n)=1$, то $\exists u,v\in Z\,(uz+vn=1)$, тобто $uz\equiv 1(\text{mod}n)$. Отже, $z'=u\,\text{mod}n$ є оберненим до z.

Достатність. Якщо для деякого $z\in Z_n$ $\exists z'\in Z_n\ z'\cdot z\equiv 1$, то $\exists k\in Z\ z'\cdot z=kn+1$, звідки слідує, що z та n – взаємно прості.

Ефективним методом знаходження оберненого елемента для $z\in Z_n^*$ є розширений алгоритм Евкліда, оскільки n та z – взаємно прості. Цей алгоритм встановлює коефіцієнти розкладу $vn+uz=1$, звідки отримуємо, що $uz\equiv 1(\text{mod}n)$, тобто u є шуканим числом, оберненим до z за модулем n.

Очевидним є такий наслідок.

Наслідок 11.2. *Якщо p – просте, то $Z_p=Z_p^*\cup\{0\}$ утворює поле.*

Приклад 11.2. $Z_9^*=\{1,2,4,5,7,8\}$, $2^{-1}(\text{mod}9)=5$, $4^{-1}(\text{mod}9)=7$.

11.4. Функція Ейлера та її властивості

Означення 11.9. *Порядок групи Z_n^* (кількість її елементів) називають функцією Ейлера і позначають $\varphi(n)$. Іншими словами, $\varphi(n)$ – це кількість натуральних чисел, менших від n і взаємно простих з n.*

Головні властивості функції Ейлера:

1) очевидно, що $\varphi(n)<n$;

2) якщо $p\in\mathbf{\Pi}$ і $p>1$, то $\varphi(p)=p-1$;

3) для попарно взаємно простих $n_1,\ldots,n_k$, $k\in N$ справедлива рівність:

$$\varphi(n_1\cdot\ldots\cdot n_k)=\varphi(n_1)\cdot\ldots\cdot\varphi(n_k).$$

Теорема 11.3. Теорема Ейлера

Для взаємно простих цілого x та натурального $n>1$ справедлива конгруенція $x^{\varphi(n)}\equiv 1(\text{mod}n)$.

Теорема 11.4. Мала теорема Ферма

Нехай $p\in\mathbf{\Pi}$ і $p>1$, тоді $\forall a\in Z_p^*$ $a^{p-1}\equiv 1(\text{mod}\,p)$.

Випливає як наслідок з теореми Ейлера.

11.5. Алгоритм *RSA*

Повідомлення записують у цифровій формі. Шифрування здійснюють блоками однакової довжини: кожен блок позначає деяке число *M*, яке не перевищує *n*.

Алгоритм складається з таких етапів: генерація ключів, криптування, дешифрування.

Генерація ключів відбувається у такій послідовності:

1. Обирають два великі прості числа p і q приблизно однакової довжини, порядку $10^{100} \approx 2^{332}$.

Для отримання достатньо великого простого числа насамперед використовують теоретичні результати про розподіл простих чисел.

Сьогодні доведена наявність простого числа між довільним натуральним n і $n+n^{107/200}$ [2].

Для генерації простого числа заданого розміру l обирають випадкове ціле число n з інтервалу $(10^{l-1}, 10^{l})$. Далі послідовно перебирають числа $n, n+1, n+2, \ldots$ і перевіряють їх тестом простоти.

Тести простоти (для перевірки, чи число $z \in \mathbf{N}$ є простим) є двох типів: детерміновані і ймовірнісні.

Детерміновані тести, які ґрунтуються на алгоритмі типу сита Ератосфена, мають квазіполіноміальну складність $O\left((\ln z)^{C \ln\ln\ln z}\right)$, є ефективними для $z \sim 10^{100}$.

Ймовірнісні тести (наприклад, тест Соловея–Штрассена) мають складність $k \cdot O(\ln z)^3$, де k – кількість повторень тесту. Вони розпізнають простоту з імовірністю 2^{-k} і є ефективними для $z \sim 10^{300}$.

Індійські математики Маніндра Агравал, Нірадж Каял та Нітін Саксена (*Manindra Agrawal, Neeraj Kayal, Nitin Saxena*) 2002 року запропонували детермінований поліноміальний алгоритм доведення простоти (*AKS*-тест) зі складністю $O(\ln z)^{12+\varepsilon}$, де ε – мале дійсне число. У подальшому цей тест вдосконалено до часової границі $O(\ln z)^6$ іншими авторами (*H. W. Lenstra jr., Carl Pomerance,* 2005).

2. Обчислюють $n = p \cdot q$ та функцію Ейлера $\varphi(n) = (p-1)(q-1)$.

3. Обирають натуральне число e, менше $\varphi(n)$ і взаємно просте з ним $(e \in Z^*_{\varphi(n)})$.

4. Знаходять число d – обернене до e за модулем $\varphi(n)$, тобто $d \cdot e \equiv 1(\mathrm{mod}\varphi(n))$, наприклад, за допомогою розширеного алгоритму Евкліда.

Ключі отримано: відкритий ключ – пара (n,e)***, таємний ключ – число*** d**.**

Криптування відбувається так:

1. Виділяють блок повідомлення – число $M < n$.

2. Обчислюють $C = M^e (\mathrm{mod} n)$ – криптотекст.

Ефективним методом піднесення до степеня e за модулем n є бінарний алгоритм, який вимагає менше, ніж $2\log_2 e$ множень. Коротко опишемо один з його варіантів.

Число n довжиною $k = \lfloor \log_2 n \rfloor + 1$ двійкових розрядів має такий цифровий запис: $n = n_0 + n_1 2 + n_2 2^2 + ... + n_{k-1} 2^{k-1}$. Отож можна подати

$$x^n = x^{n_0 + n_1 2 + n_2 2^2 + ... + n_{k-1} 2^{k-1}} = \left(\left(...\left(\left(x^{n_{k-1}}\right)^2 \cdot x^{n_{k-2}}\right)^2 \cdot ...\right)^2 \cdot x^{n_1}\right)^2 x^{n_0}. \qquad (11.11)$$

Розрахунок за формулою (11.11) потребує максимум $k-1$ піднесень до степеня і $k-1$ множень, тобто $2\lfloor \log_2 n \rfloor$ операцій множення. Отже, приведений алгоритм має поліноміальну складність.

Дешифрування. Для отримання повідомлення M з криптотексту c обчислюють $C^d (\mathrm{mod} n) = M$.

Приклад 11.3. Вихідне повідомлення – 030102.

Генерація ключів:

1. p = 3, q = 11;
2. $n = pq$ = 33, $\varphi(n) = (p-1)(q-1)$ = 20;
3. e = 7;
4. $d \cdot 7 = 1 (\mathrm{mod}\ 20)$, d = 3.

Криптування:

1. Виділяємо блоки повідомлення M_1 = 3, M_2 = 1, M_3 = 2;
2. Обчислюємо $C_1 = 3^7 (\mathrm{mod} 33) = 2187 \mathrm{mod} 33 = 9$,
   $C_2 = 1^7 (\mathrm{mod} 33) = 1$, $C_3 = 2^7 (\mathrm{mod} 33) = 29$.

Отримали криптотекст – 090129.

Дешифрування:

$P_1 = 9^3 (\text{mod} 33) = 729 \text{mod} 33 = 3;$

$P_2 = 1^3 (\text{mod} 33) = 1;$

$P_3 = 29^3 (\text{mod} 33) = 2.$

Дешифроване повідомлення співпадає з вихідним.

Теорема 11.5. Про дешифрування криптограми RSA
Криптотекст, отриманий алгоритмом RSA, дешифрується правильно.

Доведення для випадку, коли $M \neq 0 (\text{mod} p)$, $M \neq 0 (\text{mod} q)$.

За п. 4 етапу генерації ключів, $d \cdot e \equiv 1 (\text{mod} \varphi(n))$, тому $\exists k \in \mathbf{N} \;\; d \cdot e = k\varphi(n) + 1$.

Тоді:

$$C^d (\text{mod} n) \equiv \left(M^e (\text{mod} n) \right)^d (\text{mod} n) \equiv M^{k\varphi(n)+1} (\text{mod} n) \equiv M \cdot M^{k\varphi(n)} (\text{mod} n) = M,$$

оскільки $M^{\varphi(n)} (\text{mod} n) \equiv 1$ – за теоремою Ейлера.

Доведення загальне.

За п. 4 етапу генерації ключів $d \cdot e = 1 + k(p-1)(q-1)$. Якщо $\text{M} \neq 0 (\text{mod} p)$, тоді $\text{M}^{d \cdot e} \equiv \text{M} \cdot \left(\text{M}^{k(p-1)} \right)^{q-1} (\text{mod} p) \equiv \text{M} (\text{mod} p)$, оскільки $\varphi(p) = p - 1$.

Якщо $\text{M} \equiv 0 (\text{mod} p)$, тоді $\text{M}^{d \cdot e} (\text{mod} p) \equiv \text{M} (\text{mod} p)$. Аналогічно маємо $\text{M}^{d \cdot e} \equiv \text{M} (\text{mod} q)$. Звідси випливає, що $\text{M}^{d \cdot e} \equiv \text{M} (\text{mod} n)$.

11.7. Надійність алгоритму криптування *RSA*

Для надійності криптосистеми *RSA* необхідно, щоб такі задачі були важкими:

1) визначення повідомлення за криптотекстом і відкритим ключем;

2) визначення секретного ключа за відкритим.

У першому випадку задано n, e, C. Потрібно знайти M таке, що $M^e \equiv C (\text{mod} n)$. Ця задача зводиться до знаходження кореня e-го степеня за модулем n з деякого цілого числа. Сьогодні не знайдено ефективного алгоритму її розв'язування (хоча й не доведено, що його не існує).

У другому випадку потрібно знайти таке d, що $d \cdot e \equiv 1 \left(\mathrm{mod} \varphi(n)\right)$. Ця задача еквівалентна знаходженню функції Ейлера $\varphi(n)$, яка, своєю чергою, еквівалентна задачі розкладу n на прості множники. Остання має складність $\mathrm{T}(n) \sim O\left(\exp\left\{c \cdot \sqrt{\ln n \cdot \ln\ln n}\right\}\right)$; її неможливо розв'язати сьогодні за реальний час для $n \sim 10^{200}$.

Запитання та завдання

1. Доведіть, що число x, протилежне за модулем n до числа a, визначається рівністю $x = (n - a \bmod n) \bmod n$.
2. Доведіть нерівність (11.6): $\forall a,b \in \mathbf{N}\ a \geq b \Rightarrow r = a \bmod b < a/2$.
3. Знайдіть за допомогою алгоритму Евкліда НСД таких пар чисел: (211; 79), (34785; 2345), (40902; 24140).
4. Доведіть, що відношення конгруентності за модулем n є відношенням еквівалентності.
5. Доведіть, що $\langle Z_n, \oplus_n \rangle$ утворює комутативну групу.
6. Доведіть, що $\langle Z_n, \oplus_n, \otimes_n \rangle$ утворює комутативне кільце з одиницею.
7. Доведіть, що $\left\langle Z_n^*, \otimes_n \right\rangle$ утворює комутативну групу.
8. Доведіть ***наслідок 11.2***: Якщо p – просте, то $Z_p = Z_p^* \cup \{0\}$ утворює поле.
9. Знайдіть обернені для всіх елементів груп Z_{10}^*, Z_{11}^*, Z_{12}^*.
10. Доведіть мультиплікативність функції Ейлера для випадку $k = 2$.
11. Доведіть імплікацію: $a \equiv b (\mathrm{mod}\, p) \wedge a \equiv b (\mathrm{mod}\, q) \Rightarrow a \equiv b (\mathrm{mod}\, (pq))$.
12. Знайдіть розмір блока в символах *ASCII* для $M \sim 10^{200}$ за *RSA*-шифрування.
13. Виконайте ***приклад 11.3*** при $p = 5,\ q = 7$.

Розділ 12. ТЕОРЕТИЧНІ ОСНОВИ ПОБУДОВИ КОМП'ЮТЕРІВ

12.1. Архітектура універсального комп'ютера

Комп'ютер – це електронний пристрій, призначений для автоматизованої обробки даних відповідно до заданої програми.

Архітектура комп'ютера – це його концептуальна структура, яка визначає головні компоненти та принципи роботи, зокрема, методи обробки даних і принципи взаємодії технічних засобів та програмного забезпечення.

Сьогодні найпоширенішими є принстонська (Неймана) та гарвардська архітектури. Обидві мають два головні вузли комп'ютера: центральний процесор і пам'ять. Різниця полягає у структурі пам'яті: в принстонській архітектурі програми і дані зберігаються в одному масиві пам'яті і надходять у процесор по одному каналу, тоді як гарвардська архітектура передбачає роздільне збереження та передавання даних і програм.

Детальніше опис архітектури конкретного комп'ютера містить: структурну схему та способи доступу до її елементів, організацію та розрядність інтерфейсів, набір і формат машинних команд процесора, способи подання та формати даних, організацію пам'яті та способи її адресації, правила обробки переривань тощо.

Зокрема, за розрядністю інтерфейсів та реєстрів вирізняють 8-, 16-, 32-, 64-розрядні архітектури; за особливостями набору регістрів, формату команд і даних процесора – *Complex Instruction Set Computer* (*CISC*), R*educed Instruction Set Computer* (RISC) та *Very Long Instruction Word* (*VLIW*) архітектури; за кількістю центральних процесорів – однопроцесорні, суперскалярні, багатопроцесорні.

Головні принципи побудови та функціонування універсального комп'ютера вперше сформулював 1945 року американський математик угорського походження Джон фон Нейман (*John von Neumann*) [18]. Він 1944 року приєднався до групи розробників першого електронного комп'ютера *Electronic Numerical Integrator and Computer* (*ENIAC*) у Пенсільванському університеті, якою керували Дж. Еккерт (*J. P. Eckert*) та Дж. Моклі (*J. W. Mauchly*).

За Нейманом, основними компонентами універсального комп'ютера є:

- арифметико-логічний пристрій, який виконує арифметичні та логічні операції;
- пристрій керування – для організації процесу виконання програми;
- оперативна пам'ять для збереження програм і даних;
- пристрій введення;
- пристрій виведення.

Ці пристрої взаємодіють між собою за такою схемою (рис. 12.1).

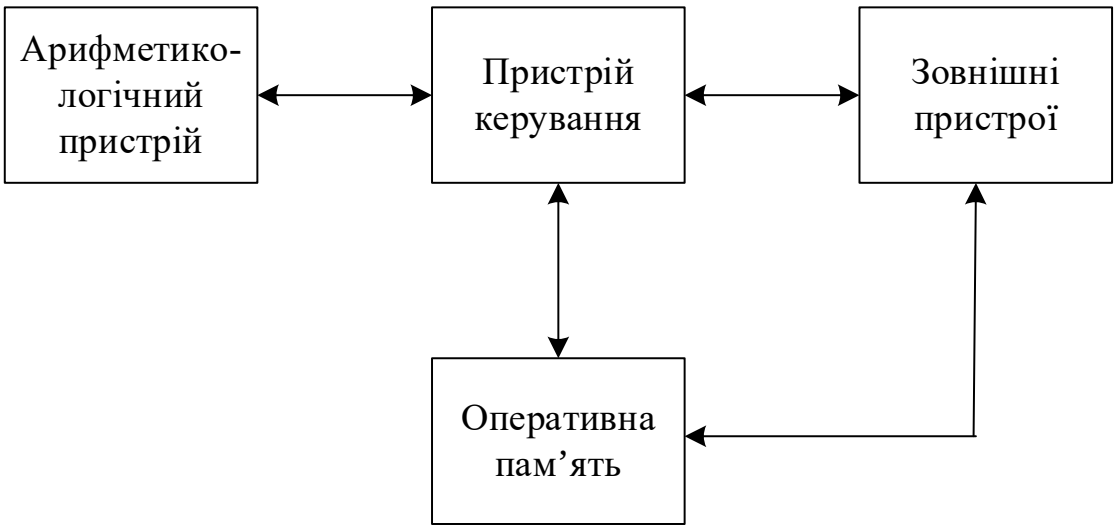


Рис. 12.1. Схема універсального комп'ютера за фон Нейманом

Перелічимо головні принципи роботи універсального комп'ютера:

1. Принцип програмного керування – послідовність дій комп'ютера визначається програмою. Цей принцип запропонував ще 1833 року Чарльз Беббідж (*Charles Babbage*) – винахідник аналітичної машини (першої обчислювальної машини).

2. Програма, як і дані, зберігається в пам'яті комп'ютера. Пам'ять є двох рівнів: оперативна і зовнішня.

3. Принцип універсальності системи команд, достатньої для реалізації будь-якого алгоритму. Наявність команд умовного переходу, які змінюють послідовність виконання команд.

4. Двійкове представлення даних і команд у комп’ютері.

Наприкінці 30-х років у Гарвардському університеті Говард Ейкен (*Howard Aiken*) розробив архітектуру обчислювальної машини з роздільним збереженням та передаванням даних і команд. В електромеханічній обчислювальній машині "Марк I", виготовленій 1944 року під його керівництвом, для зберігання команд використовували перфоровану стрічку, а для роботи з даними – електромеханічні регістри. Це дало змогу одночасно пересилати та обробляти команди й дані, що значно підвищувало загальну швидкодію. Цю архітектуру назвали гарвардською архітектурою.

Ґрунтовно ознайомитись з архітектурою сучасних комп’ютерів можна за посібниками [7; 15].

12.2. Теоретичні основи побудови цифрових електронних схем

Цифрові автомати. Двійкові цифри відображаються квантованими за двома рівнями сигналами. На фізичному рівні такі сигнали можна представити одним з трьох головних способів: потенціальним, імпульсним або динамічним. За потенціального способу низький рівень напруги відповідає цифрі "0", а деякий вищий рівень – цифрі "1". За імпульсного способу цифрам "0" та "1" відповідають імпульси різної полярності або наявність чи відсутність імпульсу. За динамічного способу цифрам "0" та "1" відповідають визначені серії імпульсів. В електронних схемах комп’ютерів застосовують потенціальний спосіб представлення сигналів; інформацію між окремими вузлами та комп’ютерами передають імпульсним чи динамічним способами.

Загальною моделлю цифрової електронної схеми комп’ютера є *багатополюсник* – схема з n входами, m виходами та s параметрами стану (рис. 12.2).

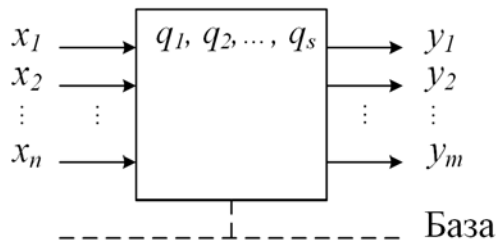


Рис.12.2. Багатополюсник

Для математичного опису роботи цієї схеми використовують теорію булевих функцій [1; 9].

Нехай $X \subset \mathbf{E}_2^n$ – множина вхідних сигналів; $Y \subset \mathbf{E}_2^m$ – множина вихідних сигналів; $Q \subset \mathbf{E}_2^s$ – множина станів, де $\mathbf{E}_2 = \{0,1\}$. У дискретний момент часу $t \in \mathbf{N} \cup \{0\} = \mathbf{Z}^+$ стан багатополюсника описується двійковим вектором $\mathbf{q}(t) \in Q \subset \mathbf{E}_2^s$, на його вхід подається двійковий вектор $\mathbf{x}(t) \in X \subset \mathbf{E}_2^n$, а на виході отримується вектор $\mathbf{y}(t) \in Y \subset \mathbf{E}_2^m$.

Математична модель багатополюсника – це функціональний зв'язок між вхідним сигналом, параметрами стану та вихідним сигналом, який задають: функцією виходів

$$\mathbf{y}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{q}(t)); \tag{12.1}$$

функцією переходів

$$\mathbf{q}(t+1) = \varphi(\mathbf{q}(t), \mathbf{x}(t)); \tag{12.2}$$

початковим станом системи

$$\mathbf{q}(0) = \mathbf{q}^0 \in Q. \tag{12.3}$$

Системи, які описуються булевими функціями (12.1), (12.2) і початковими умовами (12.3), називають *скінченними* або *цифровими автоматами*. Цифровий автомат можна означити як шестірку

$$A = \{Q, X, Y, \mathbf{f}, \varphi, \mathbf{q}^0\}. \tag{12.4}$$

Такий загальний автомат ще називають *автоматом з виходом* або *автоматом Мілі*. Його описав Джордж Мілі (*George H. Mealy*) у своїй статті 1955 року.

Якщо вихідний сигнал безпосередньо не залежить від входу, то рівняння (12.1) спрощується:

$$\mathbf{y}(t) = \mathbf{f}(\mathbf{q}(t)). \tag{12.5}$$

Такий автомат називають *автоматом без виходу*, або *автоматом Мура*. Названий на честь Едварда Мура (*Edward F. Moore*). Прикладом елементарного автомата Мура є елемент пам'яті *тригер*.

В іншому частковому випадку, коли вихідний сигнал не залежить від стану, схему називають *комбінаційною*. Тоді функціональний зв'язок (12.1) має вигляд

$$\mathbf{y}(t) = \mathbf{f}(\mathbf{x}(t)). \tag{12.6}$$

Приклад 12.3. Розглянемо скінченний автомат для порозрядного додавання двох двійкових чисел – *багатотактовий суматор*. Числа додають порозрядно, починаючи з нульового розряду. Моменту часу t відповідає номер розряду:

$$\begin{array}{r} \ldots a_3 a_2 a_1 a_0 \\ + \quad \ldots b_3 b_2 b_1 b_0 \\ \hline \ldots c_3 c_2 c_1 c_0 \end{array} \qquad (12.7)$$

Схему відповідного автомата наведено на рис. 12.3. Автомат має один параметр стану – q_1 ($q_1 = 0$ – розряд переносу відсутній, $q_1 = 1$ – розряд переносу наявний), два входи (x_1, x_2) та один вихід (y_1).

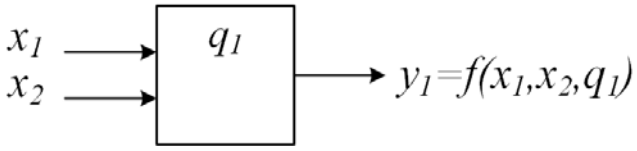


Рис. 12.3. Схема багатотактового суматора

Запишемо таблицю, яка визначає роботу цього автомата:

$x_1(t)$	$x_2(t)$	$q_1(t)$	$y_1(t)$	$q_1(t+1)$
0	0	0	0	0
1	0	0	1	0
0	1	0	1	0
1	1	0	0	1
0	0	1	1	0
1	0	1	0	1
0	1	1	0	1
1	1	1	1	1

За таблицею легко записати функції виходу та переходу. Наприклад, запишемо функцію виходів $y_1 = f(x_1, x_2, q_1)$ як диз'юнктивну нормальну форму:

$$f = x_1 \cdot \overline{x}_2 \cdot \overline{q}_1 \vee \overline{x}_1 \cdot x_2 \cdot \overline{q}_1 \vee \overline{x}_1 \cdot \overline{x}_2 \cdot q_1 \vee x_1 \cdot x_2 \cdot q_1 . \qquad (12.8)$$

Найпростіші логічні елементи. Будь-яку логічну функцію можна реалізувати за допомогою трьох логічних операцій [1; 9]: кон'юнкції – $\wedge$(*AND*, *&* або $\cdot$), диз'юнкції – $\vee$ (*OR*) та логічного заперечення – $\neg$ (*NOT* або верхня риска $\overline{\ }$).

Електронні схеми, які реалізують відповідні логічні операції, називають *логічними елементами* або *логічними вентилями*. Логічні елементи можна реалізувати за допомогою різних пристроїв: шестерень, реле, лампових тріодів чи транзисторів.

Транзистор (від англ. *transfer* – переносити; *resistor* – опір) – триполюсний напівпровідниковий електронний прилад, що змінює свій опір за прикладення напруги на керуючий електрод. Завдяки цьому транзистор застосовують для підсилення, перетворення та комутації електричних сигналів. Під час конструювання логічних схем транзистор використовують як перемикач.

Сучасні інтегральні схеми містять на одному кристалі кремнію (арсеніду галію) мільйони транзисторів. Схематично реалізацію транзистора на кристалі подано на рис. 12.4.

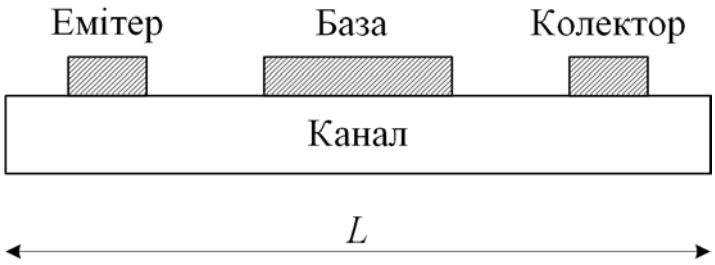


Рис. 12.4. Схема реалізації транзистора на кристалі

Базовий розмір транзистора – довжина каналу *L* становила: у 60-ті роки – 20 мкм; у процесорах *Intel* 80286 (1984) – 1,5 мкм; у процесорах *Pentium* (1993) – 800 нм, *AMD-K7* (1999) – 130 нм; у процесорах *Pentium IV* та його модифікаціях (з 2002 р.) – 130 нм, 90 нм, 65 нм; у процесорах *Intel Core 2*, *Core i3*, *i5*, *i7* (з 2009 р.) – 45 нм, 22 нм, 14 нм.

Сьогодні деякі виробники процесорів анонсували перехід на технологічний процес 2 нм до 2025 року.

Розглянемо транзисторну схему, яка реалізує логічне заперечення. Скористаємось біполярним транзистором типу *n-p-n* (рис. 12.5).

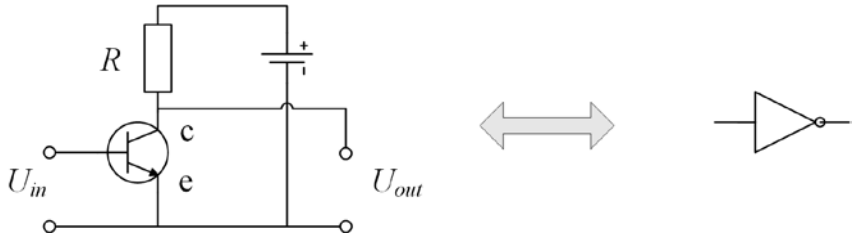


Рис.12.5. Транзисторна схема інвертора та його умовне позначення

Коли на вході напруга відсутня ($U_{in}=0$), транзистор закритий – опір кола емітер – колектор R_T є значним, тому на виході маємо напругу деякого рівня

$$U_{out}=\frac{R_T}{R+R_T}U_0. \qquad (12.9)$$

Якщо на базу транзистора подати позитивну напругу деякого рівня, то опір кола емітер – колектор падає і напруга U_{out} стає незначною.

Електронну схему, яка реалізує операцію логічного заперечення, називають *інвертором*.

Нижче наведені транзисторні схеми, які реалізують логічні операції кон'юнкції та диз'юнкції (рис. 12.6; 12.7).

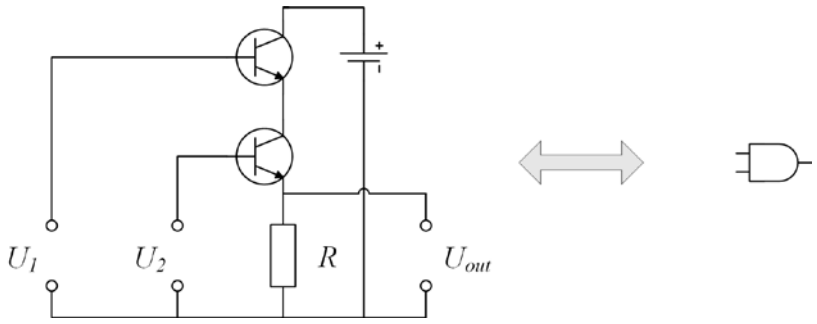


Рис. 12.6. Транзисторна схема логічного "І" та її умовне позначення

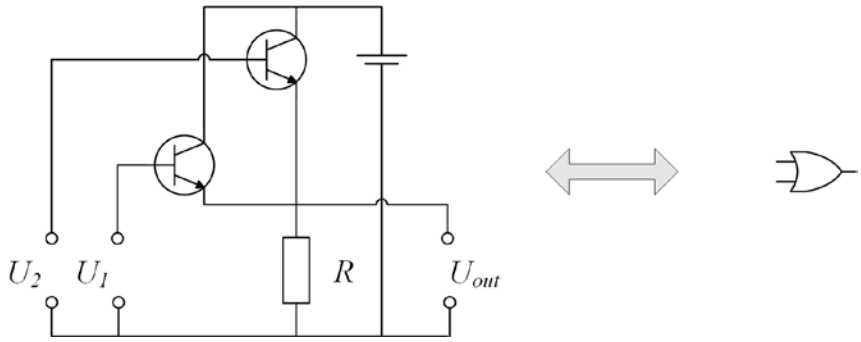


Рис. 12.7. Транзисторна схема логічного "АБО" та її умовне позначення

Схема багаторозрядного суматора. Спершу розглянемо *логічну схему для додавання за модулем два*. Запишемо таблицю істинності для операції додавання за модулем два (*XOR* або $\oplus$):

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	0

На її основі легко записати диз'юнктивну нормальну форму:

$$y = x_1 \oplus x_2 = x_1\overline{x}_2 \vee \overline{x}_1 x_2 . \qquad (12.10)$$

З використанням розглянутих логічних елементів схему для реалізації операції XOR подано на рис. 12.8.

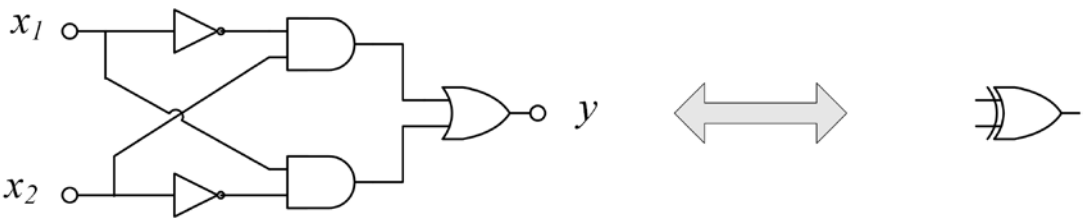


Рис. 12.8. Логічна схема для додавання за модулем два

Логічна схема для перенесення одиниці в наступний розряд при додаванні. Така схема має три входи (x_0 – "0" або "1" з попереднього розряду; x_1 – перший доданок; x_2 – другий доданок), та один вихід – y. Якщо серед аргументів є принаймні дві цифри "1", то потрібно перенести одиницю у наступний розряд – вихід дорівнює "1", у протилежному випадку – "0". Функцію виходу легко записати інтуїтивно:

$$y = R(x_1, x_2, x_3) = x_0 x_1 \vee x_1 x_2 \vee x_2 x_0 . \qquad (12.11)$$

За нею просто зобразити відповідну логічну схему (рис. 12.9).

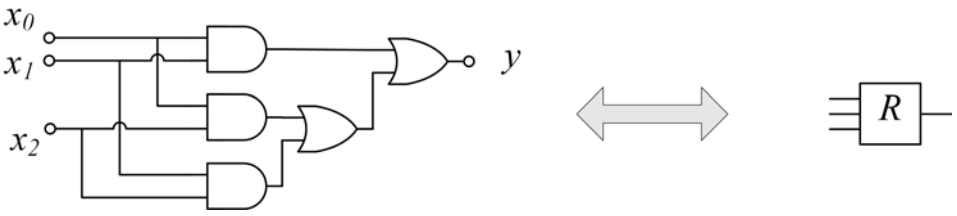


Рис. 12.9. Логічна схема для фіксації розряду перенесення

За допомогою розглянутих логічних елементів додавання за модулем два та перенесення розряду можна реалізувати додавання багаторозрядних двійкових чисел (12.7). Відповідну схему наведено на рис. 12.10.

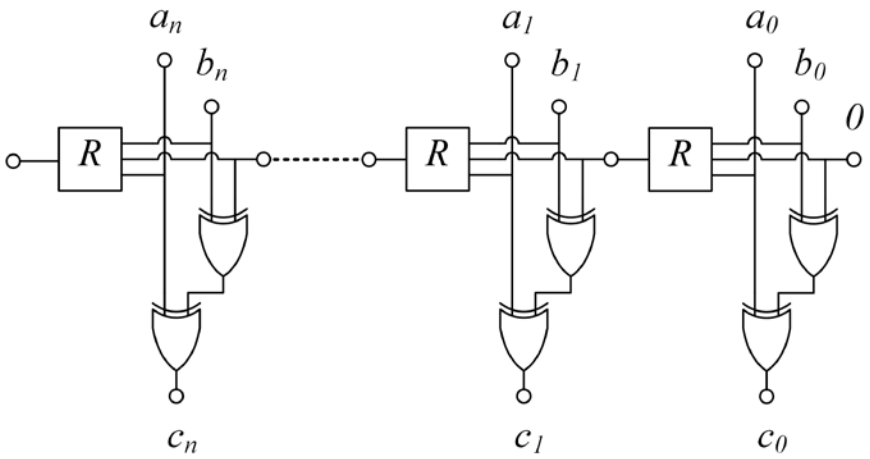


Рис. 12.10. Логічна схема багаторозрядного суматора

Елемент пам'яті – тригер. *Тригер* – це електронна схема, яка постійно видає на виході значення"0" або "1", яке не змінюється доти, доки схему не переведуть у протилежний стан.

Тригер реалізується за допомогою автомата Мура з однією змінною стану, двома вхідними змінними та одним виходом з такими функціями виходу та переходу:

$$y_1 = q_1,\ q_1 = x_1 \vee (q_1 \wedge \overline{x}_2). \qquad (12.11)$$

Відповідну схему зображено на рис. 12.11.

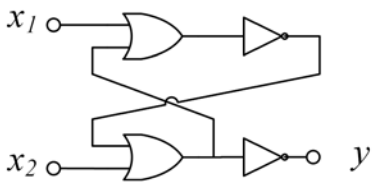


Рис. 12.11 Логічна схема тригера

Запис у пам’ять "1" здійснюють надсиланням на вхід сигналу $x_1 = 1,\ x_2 = 0$, а запис "0" – $x_1 = 0,\ x_2 = 1$.

Отже, електронні схеми комп’ютера – складні багаторівневі системи, головними елементами яких є логічні елементи, реалізовані на транзисторах.

Запитання та завдання

1. Назвіть головні компоненти універсального комп’ютера за фон Нейманом.
2. Які головні принципи роботи універсального комп’ютера?
3. Опишіть математичну модель багатополюсника.
4. Що таке автомат Мілі або автомат з виходом?
5. Опишіть автомат Мура або автомат без виходу.
6. Що таке транзистор?
7. Опишіть транзисторну схему інвертора.
8. Опишіть транзисторну схему логічного "АБО".
9. Опишіть транзисторну схему логічного "І".
10. Опишіть логічну схему багаторозрядного суматора.
11. Що таке тригер? Опишіть логічну схему тригера.

СПИСОК ЛІТЕРАТУРИ

1. Андрійчук В. І. Вступ до дискретної математики / В. І. Андрійчук, М. Я. Комарницький, Ю. Б. Іщук. – Львів : Видавничий центр ЛНУ імені Івана Франка, 2003. – 254 с.
2. Вербіцький О. В. Вступ до криптології / О. В. Вербіцький. – Львів : ВНТЛ, 1998. – 247 с.
3. Глинський Я. М. Інформатика. Практикум з інформаційних технологій / Я. М. Глинський. – Тернопіль : Підручники і посібники, 2014. – 304 с.
4. Енциклопедія кібернетики / Відпов. ред. Глушков В. М. – Т.1 (А–Л). – Київ : Головна редакція Української радянської енциклопедії, 1973. – 584 с.
5. Ємець В. Сучасна криптографія. Основні поняття / В. Ємець, А. Мельник, Р. Попович. – Львів : БАК, 2003. – 144 с.
6. Жураховський Ю. П. Теорія інформації та кодування / Ю. П. Жураховський, В. П. Полторак. – Київ : Вища шк., 2001. – 255 с.
7. Злобін Г. Г. Архітектура та апаратне забезпечення ПЕОМ / Г. Г. Злобін, Р. Є. Рикалюк. – Київ : Каравела, 2007. – 304 с.
8. Коссак О. Комп’ютерна графіка / О. Коссак, М. Мітрулі, Н. Челакос. – Львів : Видавничий центр ЛНУ ім. Івана Франка, 2010. – 205 с.
9. Нікольський Ю. В. Дискретна математика / Ю. В. Нікольський, В. В. Пасічник, Ю. М. Щербина. – Київ : Видавнича група BHV, 2007. – 368 с.
10. Новий англо-український і українсько-англійський словник / [укл. В. Ф. Малишев, О. Ю. Петраковський]. – Київ : Промінь, 2010. – 576 с.
11. Підкуйко С. І. Математичний аналіз / С. І. Підкуйко. – Львів : Галицька видавнича спілка, 2004. – Т.1. – 544 с.
12. Подлевський Б. М. Теорія інформації в задачах : підручник / Б. М. Подлевський, Р. Є. Рикалюк. – Київ : Центр учбової літератури, 2017. – 271 с.
13. Подлевський Б. М. Теорія інформації : підручник / Б. М. Подлевський, Р. Є. Рикалюк. – Львів : Видавничий центр ЛНУ ім. І. Франка, 2018. – 342 с.

14. РСТ УРСР 2018-91. Кодування символів української абетки 8-бітними кодами. – Київ : Міністерство економіки України, 1991. – 15 с.
15. Рикалюк Р. Є. Архітектура комп'ютерів / Р. Є. Рикалюк. – Львів : Видавничий центр ЛНУ ім. Івана Франка, 2002. – 158 с.
16. Brookshear J. G. Computer Science: An Overview / J. G. Brookshear, D. Brylow ; 13th ed. – Pearson, 2020. – 737 p.
17. Burgin Mark. Theory of Information: Fundamentality, Diversity and Unification / Mark Burgin. – World Scientific, 2010. – 672 p.
18. Ceruzzi P. E. A history of modern computing / P. E. Ceruzzi. – MIT Press, 2003. – 459 p.
19. Cover T. M. Elements of Information Theory / T. M. Cover, J. A. Thomas ; 2nd ed. – John Wiley & Sons, Inc., 2006. – 748 p.
20. Eck D. J. Introduction to Computer Graphics / D. J. Eck. – Hobart and William Smith Colleges, 2021. – 456 p.
21. Ferguson Niels. Practical Cryptography / Niels Ferguson, Bruce Schneier. – John Wiley & Sons, 2003. – 432 p.
22. Hamming Richard Wesley. Coding and Information Theory / Richard Wesley Hamming ; 2nd ed. – Englewood Cliffs, New Jersey : Prentice Hall, 1986. – 259 p.
23. Handbook of Floating-Point Arithmetic / Muller J.-M. et al. ; 2nd ed. – Basel : Birkhäuser, 2018. – 638 p.
24. ISO/IEC 60559:2020. Information technology. Microprocessor Systems. Floating-Point arithmetic.
25. Kahre J. The Mathematical Theory of Information / J. Kahre. – Springer, 2002. – 516 p.
26. Korn G. A. Mathematical handbook for scientists and engineers: definitions, theorems, and formulas for reference and review / G. A. Korn, T. M. Korn ; 2nd ed. – Mineola, New York : Dover Publications, 2000. – 1151 p.
27. Murray J. D. Encyclopedia of graphics file formats / J. D. Murray, W. vanRyper ; 2nd ed. – Springer, 1996. – 1116 p.
28. Pratt W. K. Introduction to Digital Image Processing / Pratt W. K. – Hoboken : CRC Press, 2013. – 750 p.

29. Proakis John. Digital Communications / John Proakis, Masoud Salehi ; 5th ed. – McGraw-Hill Education, 2007. – 1150 p.
30. Rogers David F. Mathematical elements for computer graphics / David F. Rogers, J. Alan Adams ; 2nd ed. – New York : McGraw-Hill Publishing Company, 1989. – 512 p.
31. Salomon David. A Guide to Data Compression Methods / David Salomon. – Springer, 2002. – 307 p.
32. Sklar B. Digital Communications. Fundamentals and Applications / B. Sklar, F. Harris ; 3rd ed. – Pearson Education, Inc. 2021. – 1105 p.
33. Shannon C. E. A Mathematical Theory of Communication / C. E. Shannon // Bell System Technical Journal. – 1948. – Vol. 27. – N 4. – P. 379–423, 623–656.
34. Shannon C. E. Communication Theory of Secrecy Systems / C. E. Shannon // Bell System Technical Journal. – 1949. – Vol. 28. – N 4. – P. 656–715.
35. Schneier Bruce. Applied Cryptography: Protocols, Algorithms and Source Code in C / Bruce Schneier ; 2nd ed. – John Wiley & Sons, 1996. – 666 p.

ПРЕДМЕТНИЙ ПОКАЖЧИК

П

С

Т

Ц

Ч

Ш

Електронне навчальне видання

ПРОКОПИШИН Іван Анатолійович
РИКАЛЮК Роман Євстахович
ЧЕКУРІН Василь Феодосійович
ЧЕРВІНКА Костянтин Андрійович

Основи теорії інформації та кодування

Навчальний посібник

Редактор
Ірина Лоїк

Комп'ютерне верстання
Іван Прокопишин

Дизайн обкладинки
Василь Роган

Формат 60x84/8. Умовн. друк. арк. 18,1. Зам. № 8Е

Видавець і виготовлювач:
Львівський національний університет імені Івана Франка,
вул. Університетська, 1, м. Львів, 79000 Львів.

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру видавців,
виготівників і розповсюджувачів видавничої продукції:
Серія ДК № 3059 від 13.12.2007.

О 75 **Основи теорії інформації та кодування** : навч. посібник / [І. А. Прокопишин, Р. Є. Рикалюк, В. Ф. Чекурін, К. А. Червінка]. – Електрон. вид. – Львів : ЛНУ ім. Івана Франка, 2023. – 156 с.

ISBN 978-617-10-0820-5 (електрон. вид.)

Розглянуто головні поняття теорії інформації, ефективного кодування даних, моделювання та кодування графічних зображень, звуку та відео, кодування з захистом від завад та шифрування, а також теоретичні основи побудови електронних схем комп'ютерів.

Для студентів математичних та інших спеціальностей, які вивчають комп'ютерні науки.

УДК 519.7:004(075.8)

ISBN 978-617-10-0820-5