Міністерство освіти і науки України
Державний вищий навчальний заклад
“Ужгородський національний університет”
Математичний факультет
Кафедра системного аналізу і теорії оптимізації

ОСНОВИ ІНФОРМАТИКИ

Методичні матеріали з організації самостійної роботи для студентів математичного факультету з дисципліни “програмування”

Ужгород – 2011

Семйон І.В., Чупов С.В., Брила А.Ю., Апшай Н.І. **Основи інформатики**. Методичні матеріали з організації самостійної роботи для студентів математичного факультету з дисципліни “програмування”.– Ужгород, 2010. – 41 с.

Розглядаються базові поняття інформатики та програмування: інформація, способи представлення та вимірювання інформації, системи числення, інформаційна та обчислювальна системи. Кожна тема ілюструється за допомогою схем та прикладів. При цьому акцентується увага на розташуванні інформації в оперативній пам’яті.

Призначений для студентів різних напрямків підготовки, а також для профільного вивчення інформатики у школі.

Рекомендовано до друку Вченою радою математичного факультету

ДВНЗ “Ужгородський національний університет”

від 10 березня 2011 року, протокол № 3.

1. Інформація. Властивості, характерні риси та вимірювання інформації

Поняття інформації

Термін “інформація” виник від латинського слова “informatio” – роз’яснення, виклад і до середини XX ст. означав відомості, які передаються між людьми. З філософської точки зору інформація є відображенням реального світу за допомогою відомостей (повідомлень). На даний час термін *інформація* часто вживається замість слів *повідомлення, відомості, дані, сигнали*. Поширеними є наступні означення інформації.

Інформація – сукупність відомостей (даних), які сприймаються з навколишнього середовища (вхідна інформація), видаються у навколишнє середовище (вихідна інформація) або зберігаються у певній системі.

Інформація у широкому розумінні – це відображення довколишнього світу за допомогою знаків та сигналів.

Інформація – це повідомлення, відомості, сигнали, а також сукупність даних або сукупність знань і залежностей між ними.

Стосовно інформаційних технологій широко поширеними є означення, яке ґрунтується на понятті *даного*:

даним *називається деякий знак або сигнал, які можуть бути передані, оброблені і зафіксовані на певному носії;*

інформація *– це дані, які несуть у собі певний зміст для того їх отримує.*

Як бачимо, поняття *інформації* настільки багатозначне, що однозначного строго означення інформації не дається. Тому досить поширеною в науковому світі є концепція, згідно з якою поняття інформації належить, як і поняття речовини та енергії, до фундаментальних неозначуваних понять науки. У випадку, коли наука не може дати чіткого визначення певному об’єкту чи явищу, доводиться користуватися поняттями. Поняття відрізняються від означень тим, що різні люди за різних обставин можуть вкладати в них різний зміст (на побутовому рівні інформація – повідомлення, які одержуємо від природи і суспільства, для біологів – генетичний код, в техніці – різного роду сигнали). Отже, у різних наукових дисциплінах та в різних галузях техніки існують різні поняття інформації. Об’єднують усі ці поняття чотири основні *властивості інформації:* її можна створювати, передавати (і відповідно приймати), зберігати та опрацьовувати.

Характерні риси інформації

1. Інформація приносить знання про довколишній світ, яких не було в розглядуваній точці до одержання інформації.
2. Інформація нематеріальна, але передається за допомогою матеріальних носіїв – знаків і сигналів.
3. Знаки і сигнали несуть інформацію тільки для адресата, здатного розпізнавати їх.

4. Під час передачі інформації від джерела до адресата інформація у джерела не зникає (не зменшується).

Отже, інформація є нематеріальною але передається за допомогою повідомлень. *Повідомлення – це форма подання інформації у вигляді мови, тексту, зображення, цифрових даних, таблиць, графіків і т.д.* Засоби подання повідомлень – знаки і сигнали.

Зв'язок між інформацією і повідомленням встановлюється через поняття відображення φ, що є результатом домовленості між відправником (джерелом) і одержувачем повідомлення (адресатом), чи приписом (алгоритмом).

Відображення φ^{-1} повідомлення P на інформацію називають *правилом інтерпретації*.

$$P \xrightarrow{\varphi^{-1}} I$$

Отже, при передачі інформації відправник за допомогою відображення φ перетворює інформацію у повідомлення і передає адресату по лінії зв'язку. Коли адресат одержує повідомлення, він застосовує обернене відображення φ^{-1} для отримання переданої інформації. Схематично це виглядає так

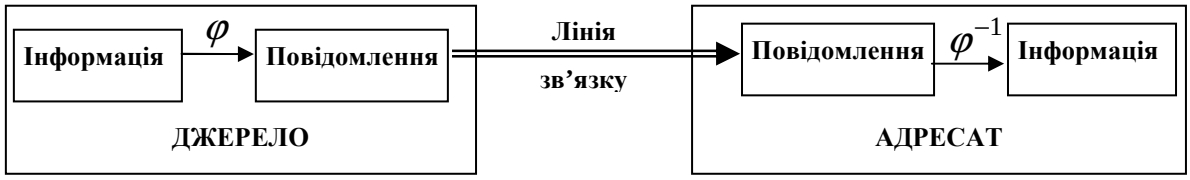
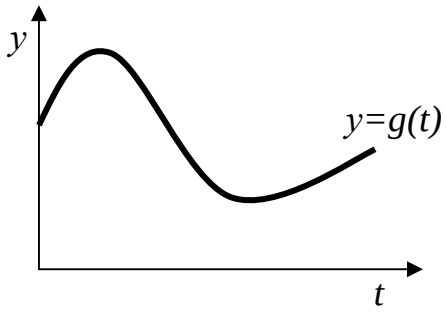


Рис. 1. Схема передачі інформації

Неперервна та дискретна інформація

Сьогодні важко уявити собі життя без різного роду пристроїв, які спрощують виконання багатьох завдань. Але їхня діяльність неможлива без процесів обробки інформації. В технічних пристроях і системах процеси прийому, передачі та обробки інформації здійснюються за допомогою сигналів, які відображають фізичні характеристики досліджуваних об'єктів. *Сигнал* – це фізичний процес, який несе певний зміст для того, хто його отримує.

Сигнали, які використовуються у технічних пристроях, можна поділити на *неперервні (аналогові)* і *дискретні (імпульсні)*. Неперервний сигнал може бути описаний за допомогою неперервної функції $y = g(t)$, яка плавно змінюється у часі (рис.2). Ця функція відображає залежність інтенсивності сигналу y від часу t. Дискретні сигнали можуть бути описані за допомогою функції $y = f(t)$, яка у певні моменти часу змінюється стрибкоподібно (рис.3). Дискретний сигнал, значення якого виражені окремими скінченими числами, називається *цифровим*.



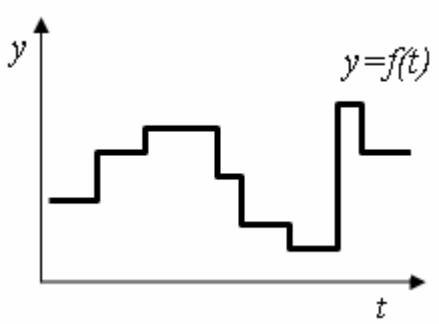


Рис.2. Неперервний сигнал Рис.3. Дискретний сигнал

Відповідно до видів сигналів розрізняють два *способи подання інформації*: аналоговий та цифровий. Інформацію, яка подається за допомогою неперервних сигналів називають *неперервною інформацією*. Якщо ж інформація подається за допомогою дискретних сигналів, то її називають *дискретною (цифровою) інформацією*.

Будь-яку неперервну інформацію можна апроксимувати дискретною інформацією з будь-якою ступінню точності. Цю наближену заміну виконують за допомогою методу *дискретизації*. До його складу входять: квантування в часі і квантування за рівнем.

Метод квантування у часі полягає у тому, що область визначення функції $g(t)$ розбивається на під інтервали рівної довжини Δt, а сама функція замінюється іншою $h(t)$, сталою на кожному інтервалі. В якості значення утвореної функції на кожному інтервалі беруть деяке середнє арифметичне значення функції $g(t)$ (рис. 4).

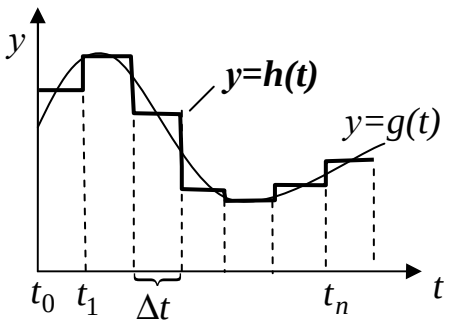


Рис.4. Квантування у часі

Дискретизація дозволяє подати неперервну інформацію дискретно, у вигляді таблиці, яку можна помістити в пам’ять комп’ютера для наступного опрацювання.

Таблиця 1. Табличне подання функції

t	t_0	t_1	…	t_n
$h(t)$	$h(t_0)$	$h(t_1)$	…	$h(t_n)$

Зрозуміло, що зменшуючи інтервал розбиття Δt покращується наближення функції, але при цьому збільшується також об'єм пам'яті, який необхідний для збереження значень цієї функції.

Аналогічно виконується квантування за рівнем. Але зараз на певну кількість однакових чи неоднакових частин (рівнів) Δy розбивається відрізок від найменшого до найбільшого значень функції $g(t)$. Таким чином, для всякого t можна сказати, до якого рівня відноситься значення функції $g(t)$.

Вимірювання інформації

Найбільшу практичну цінність мають алфавітний (об'ємний або технічний) та імовірнісний способи.

Алфавітний спосіб вимірювання інформації

Основою алфавітного (об'ємного) способу вимірювання інформації є слово, що є послідовністю символів деякого алфавіту. Еталоном для підрахунку інформації вважається слово мінімальної довжини, що складається з одного символу. Кількість інформації, що міститься в слові з одного символу, приймають за одиницю вимірювання. Найчастіше використовують двозначний алфавіт $\{0,1\}$. Величину, знатну набувати лише двох значень (0 та 1), називають ***бітом*** (binary digit – двійковий знак). Порівнюючи текст з еталоном, можна встановити обсяг (кількість) інформації за наступною формулою

$$I = k * l$$

де k – кількість символів у повідомленні, а l – кількість бітів в одному символі.

Приклад. Розглянемо повідомлення "інформація – це скарб". У цьому повідомленні є 21 символ (включаючи пробіли). Якщо для кожного символу виділяється 8 біт, то обсяг повідомлення

$$I = 21 * 8 = 168 \text{ біт.}$$

Як правило використовуються похідні від біта, значно більші одиниці вимірювання інформації:

Байт = 8 бітів;
Кілобайт (Кб) = 1024 Байт;
Мегабайт (Мб) = 1024 Кб;
Гігабайт (Гб) = 1024 Мб;
Терабайт (Тб) = 1024 Гб;
Петабайт (Пб) = 1024 Тб.

Імовірнісний спосіб вимірювання інформації

У науковому плані поняття "інформація" пов'язують із імовірністю настання тої чи іншої випадкової події. *Випадкова подія* – це подія, яка може відбутися, а може і не відбутися. *Імовірність* – це число, яке характеризує можливість настання випадкової події.

Ідеї даного способу вимірювання інформації були закладені американським інженером Робертом Хартлі (1927-1928 рр.). Процес одержання інформації розглядається як вибір одного повідомлення із заданої наперед скінченої множини з N рівноможливих повідомлень, а кількість інформації I, що міститься в повідомленні, визначається за формулою

$$I = \log N .$$

Логарифм можна брати за довільною основою, що еквівалентно вибору одиниці виміру інформації. У найпростішому випадку вибору між двома можливими повідомленнями ($N = 2$, на задане питання очікується відповідь "Так" або "Ні") кількість одержаної інформації приймають за 1 біт. Тому прийнято, що логарифм слід брати за основою 2

$$I = \log_2 N .$$

Приклад. Нехай ми маємо колоду з 32 гральних карт. Щоб вибрати одну з них існує 32 можливості. Отже, для визначення карти, яку було вибрано, необхідно $I = \log_2 32 = 5$ біт інформації. Іншими словами, нам необхідно відповісти на 5 питань, причому на кожне із них дається одна із двох відповідей: "Так" або "Ні" ("1" або "0"). Якщо, наприклад, було вибрано даму пік, то питання можуть бути такими:

1. Карта червона? Відповідь : "Ні" – 0.
2. Це трефи (хрест)? Відповідь : "Ні" – 0.
3. Це одна з 4 старших? Відповідь : "Так" – 1.
4. Це одна з 2 старших? Відповідь : "Ні" – 0.
5. Дама? Відповідь : "Так" –1.

Отже, цей вибір можна описати послідовність з 5 двійкових символів: 00101.

Розглянемо зараз випадок, коли процес одержання інформації розглядається як вибір одного повідомлення із заданої наперед скінченої множини з N повідомлень, ймовірність яких може бути різною. Нехай P_i – ймовірність одержання i-го повідомлення. Тоді кількість інформації I, що міститься в повідомленні, визначається за формулою Шенона

$$I = -\sum_{i=1}^{N} P_i \log_2 P_i .$$

Проілюструємо обчислення кількості інформації за формулою Шенона на прикладі.

Приклад. Система складається із 3 червоних, 5 зелених і 2 жовтих лампочок. Визначити, яку кількість інформації дає візуальне повідомлення про спалах однієї лампочки.

Спочатку визначимо імовірність спалаху лампочки кожного кольору (***ймовірність =(кількість лампочок певного кольору)/(загальна кількість лампочок)***):

$$P_Ч = 3/10 = 0{,}3; \quad P_З = 5/10 = 0{,}5; \quad P_Ж = 2/10 = 0{,}2 .$$

Згідно з формулою Шенона кількість інформації рівна

$$I = -\left(P_Ч \log_2 P_Ч + P_З \log_2 P_З + P_Ж \log_2 P_Ж\right) \approx 1{,}49\, біт .$$

Ентропійний спосіб вимірювання інформації

К. Шенон є основоположником статистичної теорії інформації, яка ґрунтується на понятті ентропії. *Ентропія* – загальна міра невизначеності, яка існує при одержанні повідомлення. Міра невизначеності для одержувача повідомлення залежить від того, наскільки великою є множина можливих повідомлень. Отже, для одержувача повідомлення, у зв'язку із випадковістю вибору цього повідомлення, існує невизначеність, яка знімається після його одержання. Тому *інформація* – це повідомлення, що зменшує невизначеність, яка існувала до його одержання. Зазначимо, що Шенон практично не робить різниці між ентропією та кількістю інформації. За Шеноном кількість інформації у повідомленні визначається за формулою

$$I = H_0 - H_1,$$

де H_0 – ентропія до одержання повідомлення, а H_1 – ентропія після одержання повідомлення.

Приклад. Розглянемо колоду із 32 карт. Ентропія при виборі однієї карти дорівнює кількості інформації, яка необхідна для визначення цієї карти

$$H_0 = \log_2 32 = 5 \text{ біт}.$$

Нехай нам повідомили, що було вибрано даму. Оскільки є 4 дами, і ймовірність вибору кожної із них є однаковою, то ентропія після одержання цього повідомлення дорівнює кількості інформації, яка необхідна для визначення вибраної дами

$$H_1 = \log_2 4 = 2 \text{ біт}.$$

А отже, кількість інформації, що несе у собі повідомлення "вибрано даму", дорівнює

$$I = H_0 - H_1 = 5 - 2 = 3 \text{ біти}.$$

Інформатика. Інформаційні технології

Оскільки неможливо дати чіткого означення інформації, то неможливо дати і чіткого означення інформатики. Термін інформатика (informatics) введено французькими науковцями на початку 70-х років і означав "науку про перетворення інформації". У 1963р. радянський вчений Ф.Е.Тємніков одночасно із зарубіжними авторами визначає інформатику, як науку про інформацію, яка складається з трьох основних частин: теорії інформаційних елементів, теорії інформаційних процесів і теорії інформаційних систем. У 1978 р. на конференції в Японії дається широке означення інформатики: "Поняття інформатики охоплює області пов'язані з розробкою, створенням, використанням і матеріально-технічним обслуговуванням систем обробки інформації, включаючи машини, обладнання, математичне забезпечення, організаційні аспекти, а також комплекс промислового, комерційного, адміністративного, соціального і політичного впливу".

На даний час під терміном *інформатика* розуміють *сукупність наукових напрямків, які вивчають інформацію, інформаційні процеси в природі, суспільстві, техніці, формалізацію і моделювання як методи*

пізнання, способи представлення, накопичення, обробки і передачі інформації за допомогою технічних засобів.

Якщо розглянути будь-яку науку, то поряд з цим поняттям часто вживається поняття *технологія.* Як відомо, наукою називають область діяльності людини, що пов'язана з здобуттям нових знань про оточуюче середовище і їх систематизацію. А область діяльності людини, що пов'язана з реалізацією цих знань в процесі створення і використання матеріальних і духовних цінностей називають технологією. Поширеними є наступні означення інформаційної технології.

Інформаційною технологією називають людино-машинну технологію накопичення, обробки і передачі інформації.

Інформаційна технологія – це сукупність методів, виробничих процесів і програмно-технічних засобів, об'єднаних у технологічний ланцюг, що забезпечує пошук, збирання, опрацювання, подання, перетворення, зберігання, поширення, відображення і використання інформації з певною метою.

Інформаційна технологія – це процес, який використовує сукупність засобів і методів збору, обробки і передачі даних (первинної інформації) для отримання нової інформації (інформаційного продукту) про стан об'єкта чи процесу.

Метою інформаційної технології є одержання інформації, яка використовується для прийняття рішення стосовно виконання деякої дії.

В інформатиці можна виділити 8 основних напрямів:

- **теоретична інформатика** (до цього класу відносяться дисципліни, що використовують інформацію для прийняття рішень в різноманітних ситуаціях);
- **кібернетика** (наука, що вивчає керування інформацією у живих, неживих та штучних системах);
- **програмування** (формалізація опису алгоритмів та структур даних з метою застосування у комп'ютері);
- **штучний інтелект (**імітація розумової діяльності людини**);**
- **інформаційні системи;**
- **обчислювальна техніка;**
- **інформатика у природі** (аналіз інформації у біологічних системах);
- **інформатика у суспільстві** (дослідження обігу інформації у соціальних системах).

2. Інформаційна та обчислювальна системи

Поняття інформаційної та обчислювальної системи

Важливим поняттям сучасної науки є поняття системи. Під *системою* будемо розуміти сукупність взаємопов'язаних елементів, які утворюють єдине ціле і призначені (слугують) для досягнення єдиної мети. Система, в якій одним із зв'язків є інформаційний, називається *інформаційною системою*. Інформаційний зв'язок здійснює передачу даних або інформації між елементами цієї системи.

Серед інформаційних систем можна виділити спеціальний тип, який будемо називати *обчислювальними системами*. *Обчислювальна система* – це сукупність апаратних і програмних засобів, що забезпечують автоматизацію збору, накопичення, опрацювання, систематизації, зберігання, подання, передачі інформації. Оскільки програмні засоби можна поділити на системні, які будемо об'єднувати терміном операційної системи, та прикладні (або прикладні програми), то обчислювальна система містить три основні елементи: апаратна частина (АЧ), операційна система (ОС) та прикладні програми (ПП). Як правило, системи існують у деякому, зовнішньому по відношенню до них, середовищі, і можуть взаємодіяти з ним. В якості зовнішнього середовища обчислювальної системи може бути людина або інша обчислювальна система. Схематично взаємодію елементів обчислювальної системи та зовнішнього середовища зображено на рис. 5.

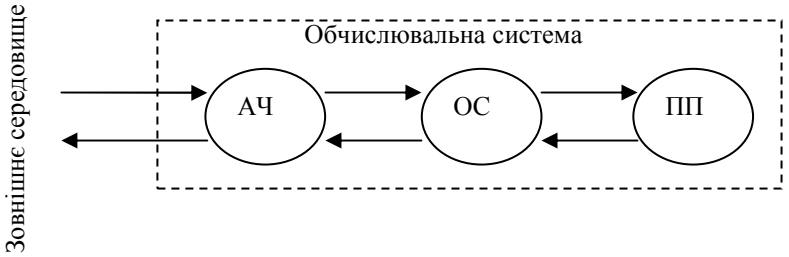


Рис. 5. Схема взаємодії обчислювальної системи і зовнішнього середовища

Проілюструємо процес взаємодії обчислювальної системи та зовнішнього середовища на прикладі.

Приклад. Нехай користувач, як зовнішнє середовище, працює на ЕОМ (обчислювальній системі), з текстовим редактором Microsot Word (прикладною програмою). І нехай користувач, діючи на апаратну частину (клавіатуру), натискає на клавішу "А". Електричний сигнал від клавіатури надходить до процесора, і операційна система генерує повідомлення про натиснення на клавішу "А". Потім операційна система визначає прикладну програму (Microsot Word), яка на даний момент виконується, і надсилає їй згенероване повідомлення. Прикладна програма Microsot Word, обробивши повідомлення, генерує інструкцію (команду) про відображення літери "А" на екрані, і надсилає операційній системі. Операційна система, взаємодіючи з апаратною частиною, відображає символ "А" для моніторі користувача, тобто надсилає користувачу візуальне повідомлення.

Згадаємо, що мова програмування – це система позначень і правил, призначена для опису алгоритмів і структур даних. Оскільки з будь-якою обчислювальною системою пов'язана своя мова, яка призначена для створення інтерфейсу (взаємозв'язку) між обчислювальною системою та зовнішнім середовищем, то можна дати інше означення обчислювальної системи. *Обчислювальна система (машина)* – це інтегрований набір алгоритмів і структур даних, що здатен зберігати та виконувати програми. Цей набір може бути реалізований в апаратній або програмній частинах обчислювальної машини (системи). В першому випадку таку машину будемо

називати *апаратною обчислювальною машиною*, в другому випадку – *програмно-модельованою обчислювальною машиною*. Слід зазначити, що будь-яка сучасна обчислювальна система розроблена як програмно-модельована обчислювальна машина, для якої зовнішнім середовищем є апаратна обчислювальна машина.

Принцип програмного керування роботою обчислювальної машини

У кінці 40-х років американський математик Джон фон Нейман описав абстрактну обчислювальну машину, за допомогою якої можна було автоматично здійснювати керування процесом виконання програми. Ця машина складається з чотирьох основних елементів:

1) пристрою керування (ПК), який організовує покомандне виконання програми та керує ходом її виконанання;

2) арифметико-логічного пристрою (АЛП), який призначений для виконання арифметичних та логічних команд;

3) оперативної пам’яті (ОП), де знаходиться програма і дані під час виконання команд програми;

4) зовнішньої пам’яті та пристроїв вводу/виводу (ЗП).

Між цими елементами встановлено два типи зв’язку: керуюча лінія (→) та лінія зв’язку (=>). Схематично абстрактну обчислювальну машину Джона фон Неймана можна зображено на рис. 6.

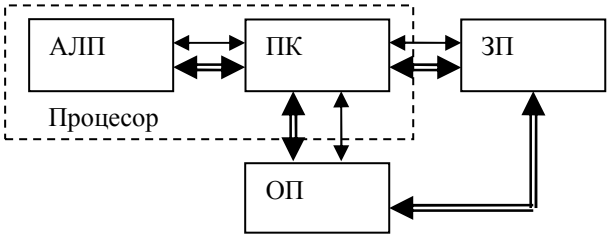


Рис. 6. Абстрактна обчислювальна машина Джона фон Неймана

Описання роботи такої машини назвали принципом програмного керування.

Схематично роботу обчислювальної машини зображено на рис. 7.

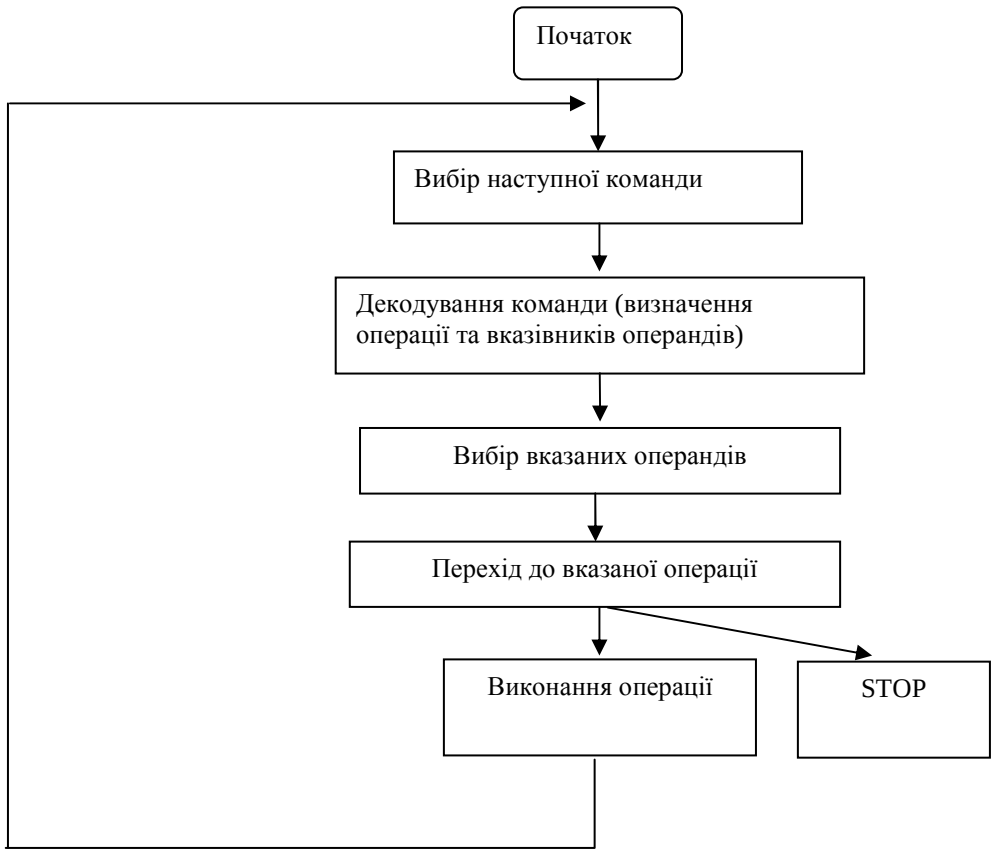


Рис. 7. Загальна схема роботи обчислювальної машини

Типи обчислювальних машин згідно формату машинних команд

Раніше зазначалося, що під час виконання програми обчислювальною машиною, команди та дані розташовуються в оперативній пам'яті. Тому команди програми слід розглядати як окремий тип даних, які при виконанні програми розміщуються в оперативній пам'яті. Таким чином, кожна команда мусить мати своє внутрішнє двійкове представлення в оперативній пам'яті. Команда може нести в собі наступну інформацію: яку операцію має виконати машина, над якими об'єктами (операндами), куди записати отриманий результат, де знаходиться наступна команда. Машинне слово, що містить команду, розбивається на групи розрядів – поля, які використовуються для задавання інформації певного призначення. Одна група розрядів – поле операції – відводиться під код або ж номер операції. Інша група розрядів – адресне поле – може містити адреси операндів, що приймають участь в операції, адресу комірки, в якій треба зберегти результат і, можливо, адресу наступної по порядку команди. Є команди, де в адресному полі вказується безпосереднє значення одного з операндів, а не адреса операнда, як це буває зазвичай. В загальному, машинне слово, що місить команду, можна зобразити так

Код операції	**Адреси операндів**	**Адреса результату**	**Адреса наступної команди**
Поле операції	Поле адрес		
Машинне слово			

Дешифруючи команду, пристрій керування визначає функціональне призначення окремих її розрядів, розглядаючи певні сукупності розрядів, як поле операції або поле адрес. Обчислювальну машину, в якій команда містить код операції та чотири адреси: дві для операндів, одну для вказівки адреси результату і одну для визначення місцезнаходження в пам'яті наступної по порядку команди, називають *чотирьохадресною обчислювальною машиною*. Отже, команда в чотирьохадресній обчислювальній машині має вигляд

$$qA_1A_2A_3A_4,$$

де q – код операції, A_1 – адреса першого операнда, A_2 – адреса другого операнда, A_3 –адреса результата, A_4 – адреса наступної команди.

q	A_1	A_2	A_3	A_4
Код операції	**Адреси операндів**		**Адреса результату**	**Адреса наступної команди**
Поле операції	**Поле адрес**			
Машинне слово				

Приклад. Розглянемо програму, команди якої записані у форматі чотирьохадресної обчислювальної машини. Ця програма буде вводити значення змінних a,b,c та обчислювати значення змінної $r = a + b * c$. Будемо вважати, що для збереження кожної змінної виділено один байт, і розміщені вони, починаючи з адреси 100.

a	b	c	r	← **Змінні**
100	**101**	**102**	**103**	← **Адреси змінних**

Нехай розглядувані операції кодуються наступним чином

00	**01**	**02**	**03**	**05**	← **Код операції**
кінець	**введення**	**виведення**	**додавання**	**множення**	← **Операція**

Код команди займає 1 байт, а адреса – 2 байти, тому вся машинна команда займає 9 байтів. Будемо вважати, що програма розташована у пам'яті, починаючи з адреси 500. Якщо в команді якась із адрес не використовується, то вона дорівнюватиме 0.

Наведемо програму, що обчислює значення r за схемою, яка відповідає послідовності виконання операцій згідно їх пріоритету:

$$r = a + b * c \;\Rightarrow \begin{cases} 1)\ r := b * c; \\ 2)\ r := a + r. \end{cases}$$

Адреса команди **(у 16-ій сист. числення)**	q **(код оп.)**	A_1 **(перший операнд)**	A_2 **(другий операнд)**	A_3 **(результат)**	A_4 **(наступна команда)**	Інтерпретація
500	**01**	**0**	**0**	**100**	**509**	**Введення** a
509	**01**	**0**	**0**	**101**	**512**	**Введення** b
512	**01**	**0**	**0**	**102**	**51B**	**Введення** c
51B	**05**	**101**	**102**	**103**	**524**	$r := b * c$
524	**03**	**100**	**103**	**103**	**52D**	$r := a + r$
52D	**02**	**103**	**0**	**0**	**536**	**Виведення** r
536	**00**	**0**	**0**	**0**	**0**	**Кінець програми**

Було виявлено, що в більшості програм машинні команди ідуть послідовно одна за одною, тобто розташовані лінійно. Тому було запропоновано *трьохадресну машинну команду*, для якої за замовчуванням вважається, що команда, яка має виконуватись наступною, іде одразу за тою, що виконується. Формат такої команди такий $qA_1A_2A_3$. Для зміни лінійної послідовність виконання команд, в систему команд машини було включено команди умовних та безумовних переходів. Трьохадресна машинна команда займає менше пам'яті, а отже, при її використанні в оперативній пам'яті можна розташувати більше даних і команд, що прискорює виконання програми.

Приклад. Розглянемо задачу із попереднього прикладу. Зараз команда займає 7 байтів, тому програма буде наступною.

Адреса команди **(у 16-ій сист. числення)**	q **(код оп.)**	A_1 **(перший операнд)**	A_2 **(другий операнд)**	A_3 **(результат)**	Інтерпретація
500	**01**	**0**	**0**	**100**	**Введення** a
506	**01**	**0**	**0**	**101**	**Введення** b
50C	**01**	**0**	**0**	**102**	**Введення** c
512	**05**	**101**	**102**	**103**	$r := b * c$
518	**03**	**100**	**103**	**103**	$r := a + r$
51E	**02**	**103**	**0**	**0**	**Виведення** r
524	**00**	**0**	**0**	**0**	**Кінець програми**

В багатьох випадках, після виконання команди результат зберігається або в одному із внутрішніх регістрів, або за адресою одного з операндів команди. Виходячи з цього, було запропоновано *двохадресні машинні команди* з форматом: qA_1A_2. Зараз команда займає 5 байтів.

Приклад. Будемо вважати, що результат виконання бінарної операції (з двома операндами) зберігається за адресою першого операнда. Тоді обчислення, які розглядались у попередньому прикладі можна виконати за наступною схемою

$$a + b * c \;\Rightarrow \begin{cases} 1)\ b := b * c; \\ 2)\ a := a + b. \end{cases}$$

Таким чином, результат буде міститися у a.

Адреса команди **(у 16-ій сист. числення)**	q **(код оп.)**	A_1 **(перший операнд)**	A_2 **(другий операнд)**	Інтерпретація
500	**01**	**100**	**0**	**Введення** a
505	**01**	**101**	**0**	**Введення** b
50A	**01**	**102**	**0**	**Введення** c
50F	**05**	**101**	**102**	$b := b * c$
514	**03**	**100**	**101**	$a := a + b$
519	**02**	**100**	**0**	**Виведення** a
51E	**00**	**0**	**0**	**Кінець програми**

Ще більш економною є, *одноадресна машинна команда*, яка має формат qA_1, і займає всього 3 байти. Поле адрес для неї розшифровується так: A_1 –

адреса першого операнда, другий операнд завжди повинен розміщуватися у внутрішньому регістрі, який називають регістр акумулятором (***RA***). Регістр – це невелика ділянка пам'яті, яка розташована в модулі процесора. Після виконання команди результат записується також у внутрішній регістр акумулятор. Зараз команда займає всього 3 байти.

Приклад. Позначимо через **07 –** код операції переміщення (копіювання) значення із регістра RA в комірку оперативної пам'яті, а через **08** – код операції завантаження (копіювання) значення комірки оперативної пам'яті в регістр RA.

Обчислення проведемо за наступною схемою

$$a + b * c \Rightarrow \begin{cases} 1)\ RA := b; \\ 2)\ RA := RA * c; \\ 3)\ RA := RA + a. \end{cases}$$

Адреса команди **(у 16-ій сист. числення)**	q **(код оп.)**	A_1 **(перший операнд)**	Інтерпретація
500	**01**	**0**	**Введення значення змінної a в регістр RA**
503	**07**	**100**	$a := RA$
506	**01**	**0**	**Введення значення змінної b в регістр RA**
509	**07**	**101**	$b := RA$
50C	**01**	**0**	**Введення значення змінної c в регістр RA**
50F	**07**	**102**	$c := RA$
512	**08**	**101**	$RA := b$
515	**05**	**102**	$RA := RA * c$
518	**03**	**100**	$RA := RA + a$
51B	**02**	**0**	**Виведення a**
51E	**00**	**0**	**Кінець програми**

У залежності від того, який формат команд використовується в обчислювальних машинах, вони поділяються на чотири "чистих" *типи обчислювальних машин*: чотирьохадресні, трьохадресні, двохадресні та одноадресні обчислювальні машини.

3. Кодування інформації в пам'яті ЕОМ

Як було зазначено, інформація по своїй природі є нематеріальною, але може бути представлена за допомогою повідомлень. Для автоматизації роботи з ними важливо уніфікувати їх форму представлення. Для цього використовуються прийом *кодування.*

Кодування – це відображення, яке перетворює повідомлення у комбінації з дискретних сигналів (чи набори певних знаків). Правило, яке описує таке перетворення називається *кодом*. Кодом також називають множину всіх можливих комбінацій із дискретних сигналів, які можуть бути

одержані при кодуванні. Саме код є універсальним способом відображення інформації під час її передавання, опрацювання та збереження як системи дискретних сигналів чи символів.

При кодуванні можуть ставитися різні цілі: ***економність, надійність, зручність фізичної реалізації, зручність сприйняття, захист від стороннього доступу.*** Цілі часто суперечать одна одній. Наприклад, економні повідомлення можуть виявитися ненадійними. Звичайний запис числа цифрами набагато економніший ніж запис числа прописом, але пошкодження або знищення однієї цифри змінює величину числа. В той же час запис числа прописом є значно надійнішим.

До пошкодження	**Після пошкодження**
125	**15**
сто двадцять п'ять	**сто адцять п'ять**

При символьному кодуванні повідомлень кожному з них ставиться у відповідність послідовність символів деякого алфавіту. Найпростішим алфавітом, достатнім для кодування довільного повідомлення, є *двійковий* алфавіт, який складається з двох символів 0 і 1. Код, який ґрунтується на двійковому алфавіті називають *двійковим* або *бінарним кодом*. Саме двійковий код використовують для кодування повідомлень в системах передачі та обробки інформації. Вибір двійкового коду зумовлений зручністю фізичної реалізації більшості складових елементів таких систем, які характеризуються двома можливими станами. Так комірка оперативної пам'яті комп'ютера характеризується двома значеннями напруги (висока і низька), комірка магнітного носія даних може бути намагніченою у одному з двох можливих напрямів, комірка оптичного носія даних може поглинати або відбивати світловий промінь і т.д. Таким різним станам ставлять у відповідність символи 1 і 0. Саме тому комірку пам'яті, яка може знаходитися у двох різних станах, також називають *бітом*.

Кодування числової інформації. Системи числення

Система числення — це спосіб зображення чисел і відповідні йому правила дій над числами. Умовні знаки, вживані для позначення чисел, називаються *цифрами*.

Розрізняють *позиційну* та *непозиційну* системи числення.

Історично першими системами числення були *непозиційні системи* – це такі системи, в яких значення кожної цифри не залежить від того місця (позиції), на якому вона знаходиться у записі числа. До таких систем належать, наприклад, старослов'янська система запису чисел за допомогою кириличної абетки та римська система числення.

У римській системі числення використовується сім цифр: **I(1), V(5), X(10), L (50), C (100), D (500), M(1000)**. Якщо в числі цифра меншого номіналу зустрічається раніше ніж цифра більшого, то меншу цифру потрібно відняти від більшої (наприклад, IX означає 10 - 1=9), інакше цифри додають (наприклад, запис MMMX означає 1000 + 1000 + 1000 + 10=3010). Оскільки непозиційні системи числення мають дуже громіздкі правила

виконання арифметичних операцій, то в інформаційних процесах вони не використовуються.

У *позиційній системі числення* значення кожної цифри залежить від позиції (місця), яку дана цифра займає в записі числа. Кількість різних цифр, які використовуються для запису чисел у позиційній системі числення, називають її *основою (потужністю).* Основою системи числення називають також число, яке означає, у скільки разів одиниця наступного розрядку більше за одиницю попереднього. Якщо в системі числення використовується Q різних цифр (основа дорівнює Q)

$$\{0,1,2,...,Q-1\},$$

то таку систему числення називають Q-ковою. Для зручності при записі числа в якості нижнього індексу в дужках будемо записувати основу системи числення, в якій записано це число (наприклад, $23_{(10)}$ – число в десятковій системі числення, $43_{(8)}$ – число в вісімковій системі числення). Найбільш поширеними є десяткова, двійкова, вісімкова та шістнадцяткова системи числення.

Таблиця 2. Цифри різних систем числення

Система числення	**Цифри системи числення**
Десяткова	**0,1,2,3,4,5,6,7,8,9**
Двійкова	**0,1**
Вісімкова	**0,1,2,3,4,5,6,7**
Шістнадцяткова	**0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F**

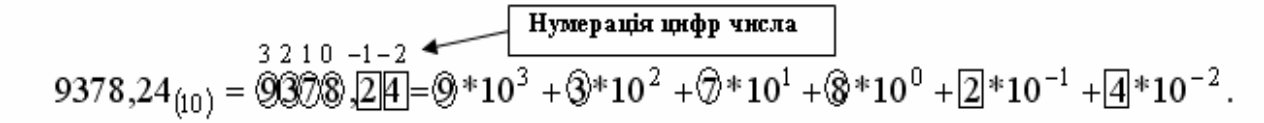
Таблиця 3. Запис шістнадцяткових цифр в різних системах числення

Шістнадцяткова	**Двійкова**	**Вісімкова**	**Десяткова**
0	**0**	**0**	**0**
1	**1**	**1**	**1**
2	**10**	**2**	**2**
3	**11**	**3**	**3**
4	**100**	**4**	**4**
5	**101**	**5**	**5**
6	**110**	**6**	**6**
7	**111**	**7**	**7**
8	**1000**	**10**	**8**
9	**1001**	**11**	**9**
A	**1010**	**12**	**10**
B	**1011**	**13**	**11**
C	**1100**	**14**	**12**
D	**1101**	**15**	**13**
E	**1110**	**16**	**14**
F	**1111**	**17**	**15**

Будь-яке число, записане у Q-ковій позиційній системі числення, можна подати в розгорнутій формі у вигляді розкладу за степенями основи цієї системи числення.

$$q_n\, q_{n-1}....q_1\, q_0, q_{-1}q_{-2}...q_{-m} = q_nQ^n + q_{n-1}Q^{n-1} + ... + q_1Q^1 + q_0Q^0 + q_{-1}Q^{-1} + ... + q_{-m}Q^{-m}.$$

Загальноприйнятою в сучасному світі є десяткова позиційна система числення, яка з Індії через арабські країни прийшла в Європу. Основою цієї системи є число десять. Тому, наприклад, розклад числа $9378{,}24_{(10)}$, записаного у десятковій системі числення, за степенями основи 10 буде наступним



Тут показник степеня основи 10 – це номер позиції цифри в записі числа (до коми нумерація ведеться зліва на право, починаючи з нуля, а після коми нумерація ведеться зліва направо, починаючи з -1).

У позиційних системах числення арифметичні операції над числами зводяться до дій над їх окремими цифрами. Ці правила відомі нам з десяткової арифметики і мають загальний характер, який не залежить від самої системи числення.

Алгоритми переведення чисел в різні системи числення

Алгоритм переведення довільного числа в десяткову систему числення

1) записати розклад числа за степенями даної основи системи числення Q;
2) якщо основа $Q>10$, то в одержаному розкладі замінити цифри числа відповідними десятковими числами;
3) виконати вказані в розкладі арифметичні операції у десятковій арифметиці.

Приклад 1. Перевести двійкове число $11101{,}1_{(10)}$ в десяткову систему числення.

1) запишемо розклад за степенями основи системи числення

$$11101{,}1_{(2)} = \overset{4\ 3\ 2\ 1\ 0\ -1}{1\,1\,1\,0\,1{,}\,1}_{(2)} = 1*2^4 + 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 + 1*2^{-1};$$

2) другий крок алгоритму опускаємо, оскільки цифри числа в двійковій системі числення аналогічно записуються і у десятковій системі числення;

3) виконуємо арифметичні операції

$$11101{,}1_{(2)} = 1*2^4 + 1*2^3 + 1*2^2 + 0*2^1 + 1*2^0 + 1*2^{-1} = 29{,}5_{(10)}.$$

Приклад 2. Перевести шістнадцяткове число $1A5_{(16)}$ в десяткову систему числення.

1) запишемо розклад за степенями основи системи числення

$$1A5_{(16)} = \overset{2\ 1\ 0}{1A\,5}_{(16)} = 1*16^2 + A*16^1 + 5*16^0;$$

2) заміняємо цифри числа на відповідні десяткові числа (у даному випадку потрібно А замінити на відповідне йому десяткове число 10)

$$1A5_{(16)} = 1*16^2 + A*16^1 + 5*16^0 = 1*16^2 + 10*16^1 + 5*16^0;$$

3) виконуємо арифметичні операції

$$1A5_{(16)} = 1*16^2 + A*16^1 + 5*16^0 = 1*16^2 + 10*16^1 + 5*16^0 = 421_{(10)}.$$

Приклад 3. Перевести четвіркове число $1301{,}21_{(4)}$ в десяткову систему числення.

1) запишемо розклад за степенями основи системи числення

$$1301{,}21_{(4)} = \overset{3\ 2\ 1\ 0\ -1-2}{1\,3\,0\,1,\,2\,1}_{(4)} = 1*4^3 + 3*4^2 + 0*4^1 + 1*4^0 + 2*4^{-1} + 1*4^{-2};$$

2) другий крок алгоритму опускаємо, оскільки цифри числа в четвірковій системі числення аналогічно записуються і у десятковій системі числення;

3) виконуємо арифметичні операції

$$1301{,}21_{(4)} = 1*4^3 + 3*4^2 + 0*4^1 + 1*4^0 + 2*4^{-1} + 1*4^{-2} = 113{,}5625_{(10)}.$$

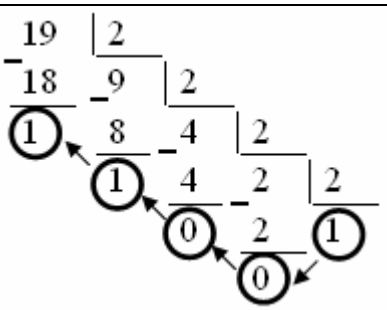
Алгоритм переведення цілого числа з десяткової системи числення у Q-кову систему числення

1) послідовно поділити дане ціле число на основу Q в десятковій системі числення (спочатку ділиться саме число, а далі ділиться отримана частка, поки частка не стане меншою за дільник Q);

2) виписати всі остачі від ділення, починаючи з останньої частки;

3) якщо $Q > 10$, то замінити кожну остачу (десяткове число) та останню частку відповідною Q-ковою цифрою.

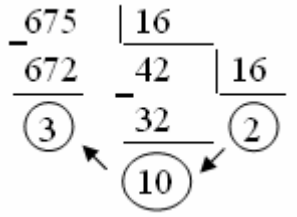
Приклад 1. Перевести ціле десяткове число $19_{(10)}$ у 2-кову систему числення.

1) Послідовно ділимо число 19 на 2 у десятковій арифметиці	
2) Виписуємо послідовність остач, починаючи з останньої частки.	**1 0 0 1 1**

3) Третій крок опускаємо, оскільки одержані числа є цифрами двійкової системи числення.

Отже, $19_{(10)} = 10011_{(2)}$.

Приклад 2. Перевести ціле десяткове число $675_{(10)}$ у 16-кову систему числення.

1) Послідовно ділимо число 675 на 16 у десятковій арифметиці:	
2)Виписуємо послідовність остач, починаючи з останньої частки.	**2 10 3**

3)Замінюємо кожне десяткове число відповідною 16-ковою цифрою.	**2 A 3**

Отже, $\mathbf{675_{(10)} = 2A3_{(16)}}$.

Інший метод переведення числа з десяткової системи числення у Q-кову є так званий *метод вичерпування.* Суть його полягає в тому, що від даного десяткового числа послідовно віднімають певні (максимально можливі) степені числа Q. Тоді десяткове число можна подати як суму цих елементів, а кожен такий степінь визначає відповідну розрядну одиницю в записі двійкового числа.

Приклад. Перевести число $235_{(10)}$ у двійкову систему числення.

$\mathbf{235_{(10)}} = 128+64+32+8+2+1 = 2^7+2^6+2^5+2^3+2^1+2^0 =$

$= \boxed{1}*2^7+\boxed{1}*2^6+\boxed{1}*2^5+\boxed{0}*2^4+\boxed{1}*2^3+\boxed{0}*2^2+\boxed{1}*2^1+\boxed{1}*2^0 =$

$= \mathbf{1\,1\,1\,0\,1\,0\,1\,1_{(2)}}$.

Алгоритм переведення десяткових дробів з десяткової системи числення у Q-кову систему числення

При переведенні десяткового дробу з десяткової системи числення у деяку Q-кову, окремо переводиться ціла частина цього числа і окремо дробова. Алгоритм переведення цілої частини розглянуто раніше, тому розглянемо алгоритм переведення дробової частини правильного десяткового дробу:

1) послідовно множити дробову частину на основу Q в десятковій арифметиці (ціла частина при кожному наступному множенні відкидається). Послідовне множення проводити до тих пір, поки в дробовій частині не одержимо нуль (точне переведення), або до забезпечення необхідної точності (тобто одержання потрібної кількості знаків після коми);

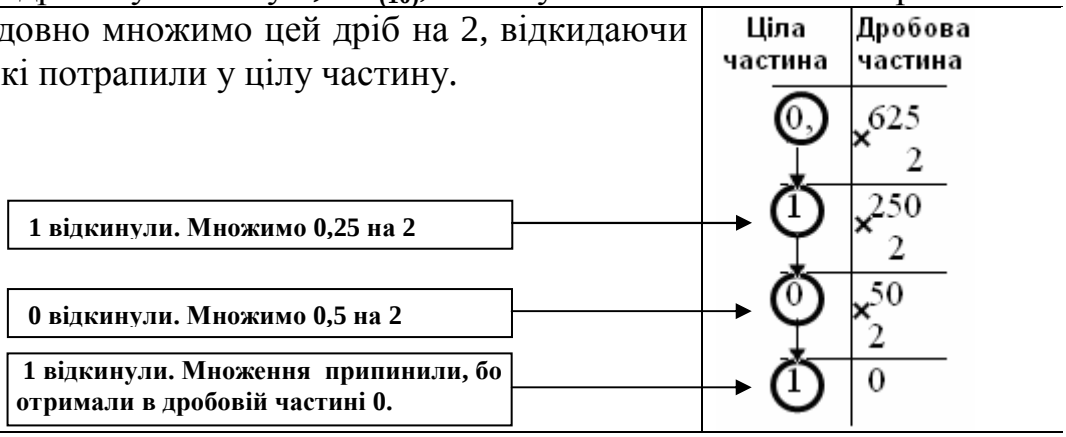
2) виписати всі цілі частини;

3) якщо $Q>10$, то замінити одержані цілі числа відповідними цифрами в Q-ковій системі числення.

Приклад 1. Перевести десятковий дріб $19{,}625_{(10)}$ у двійкову систему числення.

Переведемо окремо цілу і окремо дробову частини. У попередньому прикладі показано, що $\mathbf{19_{(10)} = 10011_{(2)}}$. Переведемо у двійкову систему числення дробову частину $\mathbf{0{,}625_{(10)}}$, застосувавши описаний алгоритм.

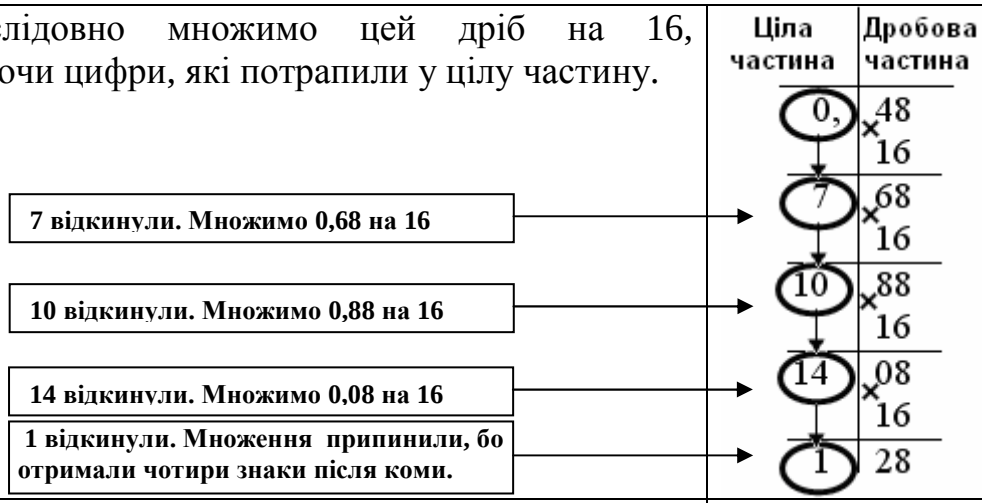
1) Послідовно множимо цей дріб на 2, відкидаючи цифри, які потрапили у цілу частину.



2) Виписуємо послідовність цілих частин, починаючи з нуль цілих.	**0 , 1 0 1**
3) Третій крок опускаємо, оскільки одержані числа є цифрами в двійковій системі числення.	

Отже, $\mathbf{0{,}625_{(10)} = 0{,}101_{(2)}}$. Тому, враховуючи перевід цілої частини, одержимо $\mathbf{19{,}625_{(10)}=10011{,}101_{(2)}}$.

Приклад 2. Перевести десятковий дріб $0{,}48_{(10)}$ у 16-кову систему числення з точністю 4 знаки після коми.

1) Послідовно множимо цей дріб на 16, відкидаючи цифри, які потрапили у цілу частину.	
2) Виписуємо послідовність цілих частин, починаючи з нуль цілих.	**0 , 7 10 14 1**
3) Замінюємо одержані цілі числа відповідними цифрами 16-кової системи числення.	**0, 7 A E 1**

Отже, $\mathbf{0{,}48_{(10)} \approx 0{,}7AE1_{(16)}}$.

Взаємозв'язок двійкової, вісімкової та шістнадцяткової систем числення

Переведення числа з вісімкової в двійкову систему числення

Для переведення числа, записаного у вісімковій системі числення, в двійкову систему числення необхідно кожну цифру цього числа перевести у двійкову систему і записати у формі тріади (групи з трьох двійкових цифр). Якщо число починається з нулів, або є нулі в кінці дробової частини, то ці нулі можна відкинути.

Приклад. Перевести число $3257{,}24_{(8)}$ у двійкову систему числення.

1) Переводимо кожну цифру у двійкову систему числення	**3 2 5 7 , 2 4** ↓ ↓ ↓ ↓ ↓ ↓ 011 010 101 111 , 010 100
2) Відкидаємо нулі на початку числа та нулі в кінці дробової частини	11 010 101 111 , 010 1

Отже, $\mathbf{3257{,}24_{(8)} = 11\ 010\ 101\ 111,\ 010\ 1_{(2)}}$.

Переведення числа з шістнадцяткової в двійкову систему числення

Для переведення числа, записаного в шістнадцятковій системі числення, в двійкову систему числення необхідно кожну цифру цього числа перевести у двійкову систему і записати у формі тетради (групи з чотирьох

двійкових цифр). Якщо число починається з нулів, або є нулі в кінці дробової частини, то ці нулі можна відкинути.

Приклад. Перевести число $3257,24_{(16)}$ у двійкову систему числення.

1) Переводимо кожну цифру у двійкову систему числення	**3 2 5 7 , 2 4** ↓ ↓ ↓ ↓ ↓ ↓ 0011 0010 0101 0111 , 0010 0100
2) Відкидаємо нулі на початку числа та нулі в кінці дробової частини	11 0010 0101 0111 , 0010 01

Отже, **$3257,24_{(16)}$= 11 0010 0101 0111, 0010 $01_{(2)}$.**

Переведення числа з двійкової у вісімкову систему числення

Для переведення числа з двійкової у вісімкову систему числення необхідно це число розбити на тріади і кожну з них перевести у вісімкову систему числення. Якщо потрібно, на початок числа і в кінець дробової частини додається необхідна кількість нулів.

Приклад. Перевести число $10011101011,1101_{(2)}$ у вісімкову систему числення.

1) Розбиваємо число на тріади	10 011 101 011 , 110 1
2) Додаємо нулі на початок та в кінець дробової частини, щоб отримати тріади	**0**10 011 101 011 , 110 1**00** ↓ ↓ ↓ ↓ ↓ ↓
3) Переводимо тріади у вісімкову систему числення	2 3 5 3 , 6 4

Отже, **$10011101011,1101_{(2)}$ = 2353, $64_{(8)}$.**

Переведення числа з двійкової у шістнадцяткову систему числення

Для переведення числа з двійкової у шістнадцяткову систему числення необхідно це число розбити на тетради і кожну з них перевести у шістнадцяткову систему числення. Якщо потрібно, на початок числа і в кінець дробової частини додається необхідна кількість нулів.

Приклад. Перевести число $10011101011,1101_{(2)}$ у шістнадцяткову систему числення.

1) Розбиваємо число на тетради.	100 1110 1011 , 1101
2) Додаємо нуль на початок, щоб отримати тетради.	**0**100 1110 1011 , 1101 ↓ ↓ ↓ ↓
3) Переводимо тріади у вісімкову систему числення.	4 E B , D

Отже, **$10011101011,1101_{(2)}$** = $4EB,D_{(16)}$.

Розглянуті алгоритми дають зручний спосіб переведення чисел з вісімкової системи числення у шістнадцяткову та з шістнадцяткової у вісімкову, використовуючи запис часла у двійковій системі числення.

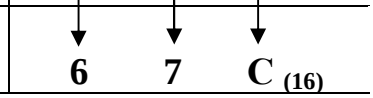
Переведення числа з вісімкової у шістнадцяткову систему числення

Для переведення числа з вісімкової системи числення у шістнадцяткову можна спочатку перевести його у двійкову систему числення, а потім число, записане у двійковій, перевести у шістнадцяткове.

Приклад. Перевести число $3174_{(8)}$ у шістнадцяткову систему числення.

1) Переводимо число у двійкову ситему числення

2) Переводимо у шістнадцяткову систему числення



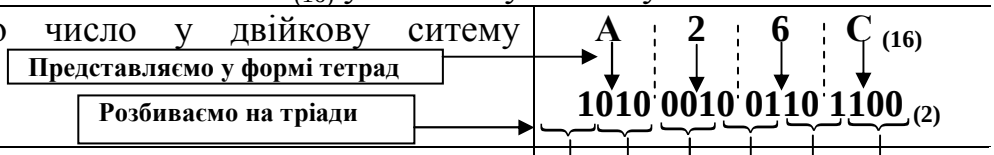
Отже, $\mathbf{3174_{(8)}=67C_{(16)}}$.

Переведення числа з шістнадцяткової у вісімкову систему числення

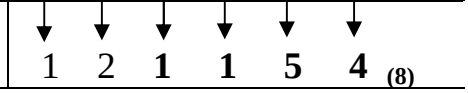
Для переведення числа з шістнадцякової системи числення у вісімкову можна спочатку перевести його у двійкову систему числення, а потім число, записане у двійковій, перевести у вісімкове.

Приклад. Перевести число A 26C $_{(16)}$ у вісімкову систему числення.

1) Переводимо число у двійкову ситему числення



2) Переводимо у вісімкову систему числення



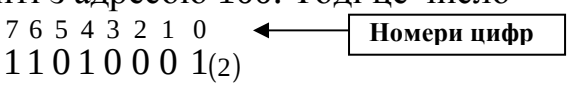
Отже, $\mathbf{A\ 26C_{(16)} = 121154_{(8)}}$.

4. Представлення інформації в пам'яті ЕОМ

Пам'ять ЕОМ можна розглядати як лінійну послідовність бітів – елементів пам'яті, які можуть знаходитись у двох різних станах, у відповідність яким ставимо символи двійкового алфавіту 1 і 0. При збереженні та зчитуванні інформації з елементів пам'яті використовують їх порядкові номери. Біти об'єднують у групи по 8, тобто байти, і здійснюють нумерацію байтів, починаючи з 0. Номер байту і є його *адресою*, яка використовується при записі та зчитуванні даних. Отже, байт можна розглядати як мінімальний елемент пам'яті, а пам'ять ЕОМ можна розглядати як лінійну нумеровану послідовність байтів. Діапазон зміни номерів байтів називають *адресним простором*. Адреси часто записують як числа у шістнадцятковій системі числення. В межах кожного байта біти умовно нумерують починаючи від 0 до 7. Умовно, бо ми не маємо можливості здійснити доступ до кожного окремого біта.

0								1								2								…	Адреси байтів
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	…	Номери бітів
																								…	Біти

Нехай, наприклад, двійкове число $11010001_{(2)}$, для збереження якого виділено 1 байт, міститься в байті з адресою 100. Тоді це число



в пам'яті ЕОМ буде представлено наступним чином.

Мікропроцесори типу I80×86 підтримують дані наступних розмірностей: байт, слово, подвійне слово. Одиницю даних довжиною у два байти називають *словом* (Word). Адреса слова – це адреса першого (“молодшого”) його байта. Зрозуміло, що адреса слова завжди кратна 2. Одиницю даних довжиною у чотири байти називають *подвійним словом* (DWord). Аналогічно до слова, адреса подвійного слова – це адреса першого його байта, і вона завжди кратна 4.

Байт	Байт	Байт	Байт	Байт	Байт	Байт	Байт	…
Слово		Слово		Слово		Слово		…
Подвійне слово				Подвійне слово				…

0				4				8				C				10		···	← Адреси подв. слів
0		2		4		6		8		A		C		E		10		···	← Адреси слів
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	···	← Адреси байтів
																			← Байти

У мікропроцесорах фірми Intel правило запису багатобайтних даних наступне – “молодший” байт знаходиться за “молодшою” адресою, “старший” байт знаходиться за “старшою” адресою.

Приклад. Розглянемо спосіб збереження двохбайтного двійкового числа (одного слова) за адресою 6.

$1110110100010010_{(2)}$

Старший байт — **Молодший байт**

	6																…	← **Адреса слова**
…	6								7								…	← **Адреси байтів**
…	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	…	← **Номери бітів**
…	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**1**	…	← **Біти**

Послідовність байтів, яку процесор може опрацьовувати за один раз називають *машинним словом*. Довжина машинного слова може бути різною: 1, 2, 4, 8 байтів і т.д. Вона визначається типом процесора та операційної системи.

Згідно з принципом програмного керування абстрактною обчислювальною машиною Джона фон Неймана, команди та дані під час виконання програми зберігаються в оперативній пам’яті. А отже, кожна команда програми або кожне дане характеризуються адресою розташування в пам’яті і об’ємом оперативної пам’яті, що використовується для їх зберігання. Байт або групу байтів оперативної пам’яті, що мають фіксовану адресу і фіксований розмір називають *елементом даних*. Оскільки будь-яка інформація кодується у двійковому алфавіті, то кожен елемент даних повинен мати чітку структуру, яка визначає, як інтерпретувати кожен його біт. Така внутрішня структура називається *форматом* елемента даних. Формат елемента даних визначає те, що собою являють дані (чи це є ціле число, чи це є дійсне число, чи це є символ і т.д.).

У програмуванні сукупність правил, які визначають розмір та формат елемента даних називають *типом даних*. Тип даних визначає діапазон значень, яких можуть набувати ці дані, а також сукупність операцій, які можна виконувати над ними.

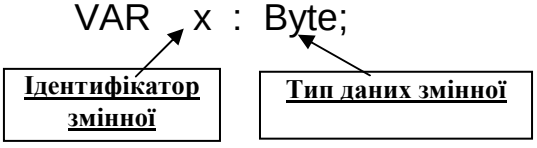
У кожній мові програмування з кожним типом даних пов'язано своє унікальне ім'я (ідентифікатор), яке є синонімом певного описання елемента даних відповідного типу. Наприклад, у мові Turbo Pascal ідентифікатор **Byte** є синонімом опису: 8 послідовних розрядів (1 байт) містить ціле число без знаку в діапазоні від 0 до 255, допустимими є всі операції, які можна виконати над цілими числами.

Отже, будь-які дані в програмі (їх ще називають *величинами*) характеризується адресою розташування в оперативній пам'яті та типом даних. Носіями даних у програмі є *константи*, *змінні*, значення яких зберігається в оперативній пам'яті, та *файли*, які зберігаються на зовнішніх носіях інформації. *Константи* – це величини, значення яких у процесі виконання програми не змінюється. *Змінні* – це величини, значення яких у процесі виконання програми можуть змінюватися.

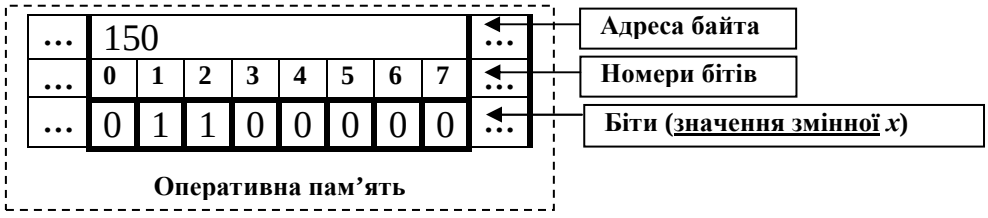
Оскільки безпосередньо через вказівку адреси працювати з елементами даних незручно, то кожному елементу даних ставиться у відповідність ідентифікатор (ім'я змінної чи константи), користуючись яким можна здійснювати доступ до відповідного елемента даних. Отже, у програмі змінна величина характеризується такими ознаками: *ідентифікатором*, *типом* і *значенням*. *Значення змінної* – це елемент даних, з якими ця змінна пов'язана.

У зв'язку з цим можна дати інше означення константи та змінної. Якщо елемент даних не може змінювати свого значення, тобто завжди містить одне і те ж саме значення, то відповідний ідентифікатор називається *константою даного типу*. Якщо елемент даних певного типу може змінювати своє значення під час виконання програми, то ідентифікатор, що зв'язаний з цим елементом даних, називається *змінною відповідного типу*.

Приклад. Нехай у мові програмування Turbo Pascal описано величину *x* типу Byte, і нехай вона зберігається в оперативній пам'яті за адресою 150 та містить значення $6_{(10)}=110_{(2)}$. Наведемо описання змінної *x*.



Тоді змінній *x* буде відповідати наступний елемент даних, розташований за адресою 150.



Представлення цілих чисел

В обчислювальних системах розрізняють два види цілих чисел: цілі числа без знаку (числа тільки більші або рівні за нуль) та цілі числа із знаком (довільні цілі числа).

Представлення цілих чисел без знака

Ціле число без знаку представляються в пам'яті ЕОМ за допомогою двійкового коду, тобто як число, записане у двійковій системі числення. Діапазон зміни чисел залежить від кількості байт, які виділяються для їх зберігання. Слід зазначити, що всі виділені біти використовуються для збереження модуля числа. Отже, якщо виділено n біт пам'яті, то в них можна зберегти число в діапазоні від 0 до $(2^n - 1)$. Якщо число у двійковому представленні містить меншу кількість цифр, ніж виділено біт пам'яті, то всі незайняті біти заповнюються нулями.

Розглянемо цілі типи без знаку у мові програмування Turbo Pascal.

Ідентифікатор типу	Діапазон		Обсяг пам'яті
Byte	0..255	$255=(2^8-1)$	1байт =8 біт
Word	0..65535	$65535=(2^{16}-1)$	2байти =16 біт

Приклад. Нехай у програмі описано змінні цілого типу без знаку x і y

```
VAR x : Word;
    y: Byte;
```

які розташовані в оперативній пам'яті починаючи з адреси 150, і в програмі змінним присвоїли наступні значення

x:=300; y:= 25;

Розглянемо елементи даних, які будуть відповідати цим змінним.

$x=300_{(10)}=100101100_{(2)}$; $y=25_{(10)}=11001_{(2)}$.

Змінна x типу Word, отже, для її зберігання виділяється 16 біт. У числі 300, записаному у двійковій системі числення, 9 цифр, тому на початок числа дописуємо 16 - 9=7 нулів. Аналогічно для змінної y – до числа дописуємо 8 - 5=3 нулі.

$x=300_{(10)}=$**0000000**$100101100_{(2)}$; $y=25_{(10)}=$**000**$11001_{(2)}$.

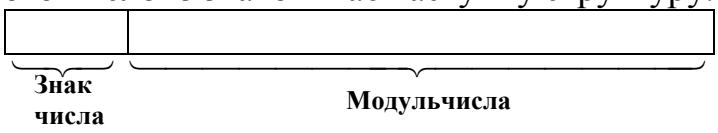
…	150								151								152								…	**Адреси байтів**
…	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	…	**Номери бітів**
…	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	…	**Біти**

x y

Таким чином, змінній x відповідає адреса 150 і елемент даних довжиною 2 байти, починаючи з адреси 150; змінній y відповідає адреса 152 і елемент даних довжиною 1 байт за адресою 152.

Представлення цілих чисел із знаком. Доповняльний код числа

При представленні цілих чисел із знаком старший розряд (біт) ділянки пам'яті виділяється для збереження знаку (0 – додатне число, 1 – від'ємне число). Тобто число із знаком має наступну структуру.



Для подання таких чисел у пам'яті ЕОМ використовують спеціальні двійкові коди: *прямий, обернений* та *доповняльний.*

Прямий код n-розрядного числа. Для його одержання треба на початок даного числа, записаного у двійковій системі числення, додати ще один розряд, який представляє знак цього числа (0 – якщо число додатне, 1 – якщо число від'ємне). Для зручності, щоб умовно відділити цей розряд від інших, після нього ставлять крапку.

	Додатне число	**Від'ємне число**
Число в двійковій системі числення	$x = +\ x_{n-1}x_{n-2}.....x_0$	$x\ = -\ x_{n-1}x_{n-2}.....x_0$
Прямий код числа	$[x]_{np} = 0.\ x_{n-1}x_{n-2}.....x_0$	$[x]_{np} = 1.\ \ x_{n-1}x_{n-2}.....x_0$

Приклад.

	Додатне число	**Від'ємне число**
Число в десятковій системі числення	$+\ 125_{(10)}$	$-\ 125_{(10)}$
Число в двійковій системі числення	$+\ 1111101_{(2)}$	$-\ 1111101_{(2)}$
Прямий код числа	$\mathbf{0.}\ 1111101_{(2)}$	$\mathbf{1.}\ 1111101_{(2)}$

Обернений код n-розрядного числа. Для додатних чисел обернений код співпадає з прямим. Для цілого від'ємного числа обернений код визначається за співвідношенням:

$$[x]_{ob} = 1.\bar{x}_{n-1}\bar{x}_{n-2}.....\bar{x}_0 \text{ де } \bar{x}_i = \begin{cases} 1, \textit{ якщо } x_i = 0, \\ 0, \textit{ якщо } x_i = 1. \end{cases}$$

Простіше кажучи, в модулі від'ємного числа нулі заміняємо одиницями, а одиниці нулями. Знаковий розряд при цьому не змінюється.

Приклад.

	Додатне число	**Від'ємне число**
Число в десятковій системі числення	$+\ 125_{(10)}$	$-\ 125_{(10)}$
Число в двійковій системі числення	$+\ 1111101_{(2)}$	$-\ 1111101_{(2)}$
Прямий код числа	$\mathbf{0.}\ 1111101_{(2)}$	$\mathbf{1.}\ 1111101_{(2)}$
Обернений код числа		$\mathbf{1.}\ 0000010_{(2)}$

Доповняльний код n-розрядного числа. Для додатних чисел доповняльний код співпадає з прямим. Для від'ємного числа доповняльний код визначається за співвідношенням:

$$[x]_{don} = [x]_{об} + 1$$

Додавання одиниці виконується за правилами двійкової арифметики: 0+0=0;

1+0 = 0+1=1; 1+1=10. Слід зазначити, що під час додавання знаковий розряд розглядається як звичайна цифра $(n+1)$-розрядного числа.

Приклад.

	Додатне число	**Від'ємне число**
Число в десятковій системі числення	$+\ 125_{(10)}$	$-\ 125_{(10)}$
Число в двійковій системі числення	$+\ 1111101_{(2)}$	$-\ 1111101_{(2)}$
Прямий код числа	**0.** $1111101_{(2)}$	**1.** $1111101_{(2)}$
Обернений код числа		**1.** $0000010_{(2)}$
Доповняльний код числа		**1.** $0000011_{(2)}$

Таким чином доповняльний код цілого від'ємного числа можна одержати за таким алгоритмом:

1) у знаковий розряд записати 1 (прямий код);

2) у цифрових розрядах одиниці замінити нулями, а нулі – одиницями (обернений код);

3) до оберненого коду додати 1 (доповняльний).

Якщо доповняльний код у знаковому розряді містить "0", то це додатне число, і для його одержання достатньо знаковий розряд замінити на "+".

Для одержання від'ємного двійкового числа, на основі його доповняльного коду, потрібно виконати зворотні дії:

1) від доповняльного коду відняти 1 (одержимо обернений код);

2) у цифрових розрядах одиниці замінити нулями, а нулі – одиницями (одержимо прямий код);

3) знаковий розряд "1" замінити знаком "-".

Використання доповняльного коду дає можливість замінити операцію віднімання операцією додавання чисел, які записані у двійкових кодах.

Приклад 1. Нехай потрібно виконати операцію віднімання

$$x - y,$$

де $x = 111011_{(2)}$, $y = 1101_{(2)}$.

Замінимо операцію віднімання операцією додавання

$$x - y = x + (-y).$$

Будемо вважати, що для збереження одного числа виділено 1байт (1 знаковий розряд і 7 розрядів для збереження модуля числа), тобто маємо числа

$$x = +0111011_{(2)}, \qquad -y = -\ 0001101_{(2)}.$$

Знайдемо доповняльний код числа x. Оскільки прямий, обернений і доповняльний коди для додатних чисел співпадають то

$$[x]_{доп} = 0.0111011_{(2)}.$$

Знайдемо доповняльний код числа $(-y)$.

Число в двійковій системі числення	$-\ 0001101_{(2)}$
Прямий код числа **Замінили "-" на "1."**	**1.** $0001101_{(2)}$

Обернений код числа **Замінили у модулі числа “1” на “0”, а “0” на “1”**	**1.** $1110010_{(2)}$
Доповняльний код числа **Додали 1**	**1.** $1110011_{(2)}$

Отже, $[y]_{доп}$=**1.**$1110011_{(2)}$.

Виконаємо операцію додавання, при цьому знакові розряди додаються як і всі інші, не звертаючи уваги на крапку. Якщо при додаванні у знаковому розряді одержимо двоцифрове число, то старшу цифру відкидають.

$$\begin{array}{r} 0.0111011 \\ +\ 1.1110011 \\ \hline 10.0101110 \end{array}$$

Старшу цифру відкидаємо

Відповідь:

$$x - y = x + (-y) = [x]_{доп} + [y]_{доп} = 0.0101110 = +0101110_{(2)}.$$

Приклад 2. Знайдемо різницю

$$y - x,$$

де $x = 111011_{(2)}$, $y = 1101_{(2)}$.

Замінимо операцію віднімання операцією додавання

$$y - x = y + (-x).$$

Будемо вважати, що для збереження одного числа виділено 1байт (1 знаковий розряд і 7 розрядів для збереження модуля числа), тобто маємо числа

$$y = +\ 0001101_{(2)}, \quad -x = -\ 0111011_{(2)}.$$

Знайдемо доповняльний код числа y. Оскільки прямий, обернений і доповняльний коди для додатних чисел співпадають то

$$[y]_{доп} = 0.\ 0001101_{(2)}.$$

Знайдемо доповняльний код числа $(-x)$.

Число в двійковій системі числення	$-\ 0111011_{(2)}$
Прямий код числа **Замінили “-” на “1.”**	**1.** $0111011_{(2)}$
Обернений код числа **Замінили у модулі числа “1” на “0”, а “0” на “1”**	**1.** $1000100_{(2)}$
Доповняльний код числа **Додали 1**	**1.** $1000101_{(2)}$

Отже, $[y]_{доп}$=**1.** $1000101_{(2)}$.

Виконаємо операцію додавання

$$\begin{array}{r} 0.0001101 \\ +\ 1.1000101 \\ \hline 1.1010010 \end{array}$$

У знаковому розряді одержали “1”, отже це доповняльний код від’ємного числа. Знайдемо це число.

Доповняльний код числа	**1.** $1010010_{(2)}$
Обернений код числа **Відняли 1**	**1.** $1010001_{(2)}$
Прямий код числа **Замінили у модулі числа “1” на “0”, а “0” на “1”**	**1.** $0101110_{(2)}$
Число в двійковій системі числення **Замінили “1.” на “ - ”**	**-** $0101110_{(2)}$

Відповідь: $y - x = -0101110_{(2)}$.

Розглянемо цілі числа із знаком у мові програмування Turbo Pascal.

Ідентифікатор типу	Діапазон		Обсяг пам’яті
ShortInt	-128..127	$(128=2^7)$	1байт =8 біт
Integer	-32768..32767	$(32768=2^{15})$	2байти =16 біт
LongInt	$(-2^{31})..(2^{31}-1)$		4байти=32 біти

Приклад. Нехай у програмі описано наступні змінні

```
VAR  x : Integer;
     y : ShortInt;
```

які розташовані в оперативній пам’яті починаючи з адреси 100, і в програмі змінним присвоїли наступні значення

x:= – 300; y:= +25;

Розглянемо елементи даних, які будуть відповідати цим змінним. Оскільки Типи *Integer, ShortInt* є типами цілих чисел із знаками, тому в пам’яті ЕОМ значення змінних буде зберігатися в доповняльному коді.

Знайдемо доповняльний код значення змінної y, для якої виділяється 1байт (1 біт на знак і 7 бітів на модуль числа).

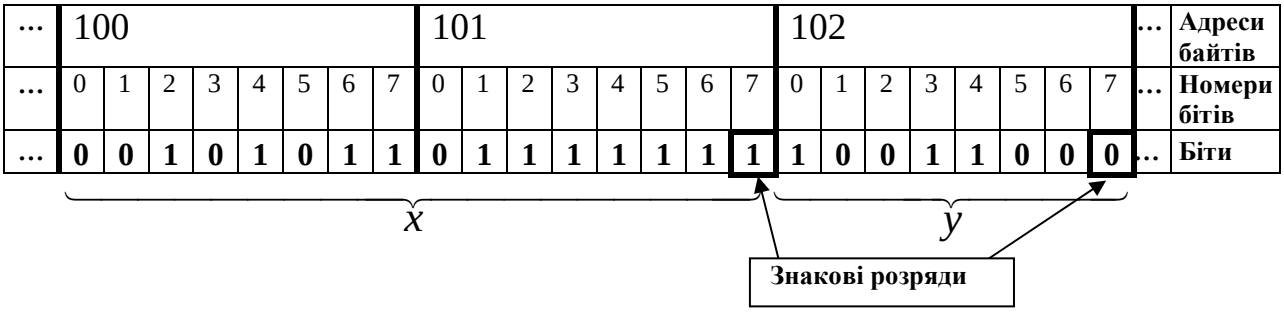
Число в десятковій системі числення	$+\ 25_{(10)}$
Число в двійковій системі числення	$+\ 0011001_{(2)}$
Прямий код числа	**0.** $0011001_{(2)}$
Обернений код числа	
Доповняльний код числа	

Знайдемо доповняльний код значення змінної x, для якої виділяється 2байти (1 біт на знак і 15 бітів на модуль числа).

Число в десятковій системі числення	$-\ 300_{(10)}$

Число в двійковій системі числення	– 000000100101100 (2)
Прямий код числа	**1.** 000000100101100 (2)
Обернений код числа	**1.** 111111011010011 (2)
Доповняльний код числа	**1.** 111111011010100

Розглянемо представлення змінних в оперативній пам'яті.



Представлення дійсних чисел

Дійсні числа в пам'яті ЕОМ можуть представлятися у двох формах: з фіксованою та плаваючою комою.

При використанні *формату з фіксованою комою* окремо зберігаються ціла та дробова частини, як звичайні цілі числа. Таке представлення не має широкого застосування у зв'язку з малим діапазоном допустимих значень.

Значно кращим є *формат з плаваючою комою* або *експоненціальний формат*, який запропоновував у 1937р. німецький вчений Кондад Цузе. При використанні цього формату довільне дійсне число представляється у вигляді:

$$x = a \cdot Q^m,$$

де a – мантиса, m – порядок, Q – основа системи числення, в якій записане число. Мантиса задає значущі цифри числа, а порядок вказує місцезнаходження крапки.

Наведемо приклади чисел, записаних у форматі з плаваючою комою.

Формат з фіксованою комою	Формат з плаваючою комою
$175_{(10)}$	$0{,}175 \times 10^3$
$18{,}75_{(10)}$	$0{,}1875 \times 10^2$
$10010{,}11_{(2)}$	$0{,}1001011 \times 10^{101}$

У програмуванні використовують нормалізоване представлення числа, тобто числа, у якого мантиса задовольняє умовам

$$\frac{1}{Q} \le |a| < 1.$$

Отже, для збереження дійсного числа необхідно зберегти мантису і порядок. Оскільки обсяг пам'яті для збереження мантиси є скінченим, то можна говорити про кількість цифр числа, які можуть бути збережені. Їх ще називають *значущими цифрами числа.*

У мові програмування Turbo Pascal є ряд дійсних типів даних, які відрізняються кількістю значущих цифр числа, кількістю байт, яке воно займає у пам'яті комп'ютера, та інтервалом точності значень.

Тип	Діапазон	Число значущих цифр	Розмір у байтах
Real	$10^{-39} .. 10^{+38}$	11-12	6
Single	$10^{-45} .. 10^{38}$	7-8	4
Double	$10^{-324} .. 10^{308}$	15-16	8
Extended	$10^{-4932} .. 10^{4932}$	19-20	10

Представлення символьної інформації

При збереженні довільного тексту в пам’яті ЕОМ, його розглядають як послідовність символів. Кожному символу ставиться у відповідність деякий фіксований двійковий код, довжина якого може бути різною (8біт, 7біт, 16біт і ін.). Оскільки такий код однозначно визначає ціле число, то можна скати, що кожному символу ставиться у відповідність деяке ціле число. Відповідність між символами і кодами символів встановлюється за допомогою *кодових таблиць*. З розвитком інформаційних технологій було розроблено багато таких таблиць, які відрізняються кількістю біт та кодами, які ставляться у відповідність символам. Тому при передачі текстової інформації вимагається, щоб у одержувача текстової інформації при декодуванні використовувалась та ж сама кодова таблиця, яку використовував відправник при її кодуванні. В іншому випадку, декодування призводить до спотворення інформації. У зв’язку з цим, Інститут стандартизації США (ANSI – American National Standard Institute) ввів у дію систему кодування ASCII (American Standart Code for Information Interchange – американський стандартний код для обміну інформацією). В цій системі кожному символу ставиться у відповідність двійковий код довжиною 8біт=1байт, тобто кількість різних символів, яка може бути закодована рівна $2^8 = 256$. Система кодування ASCII містить дві таблиці – *базова* та *розширена*. Базова таблиця встановлює відповідність між символами та кодами від 0 до 127, а розширена – від 128 до 255.

Коди від 0 до 31 базової таблиці відведено для *керуючих кодів*, які використовуються для керування виводом текстової інформації. Починаючи з коду 32 розміщено символи англійського алфавіту, розділові знаки, цифри та ін.

Таблиця 4. Базова таблиця кодування ASCII

32	пробіл	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	

Так слово “Bit” за допомогою даної таблиці може бути закодоване наступним чином.

B	**i**	**t**	← Символ
66	105	116	← Числовий код
01000010	01101001	01110100	← Двійковий код

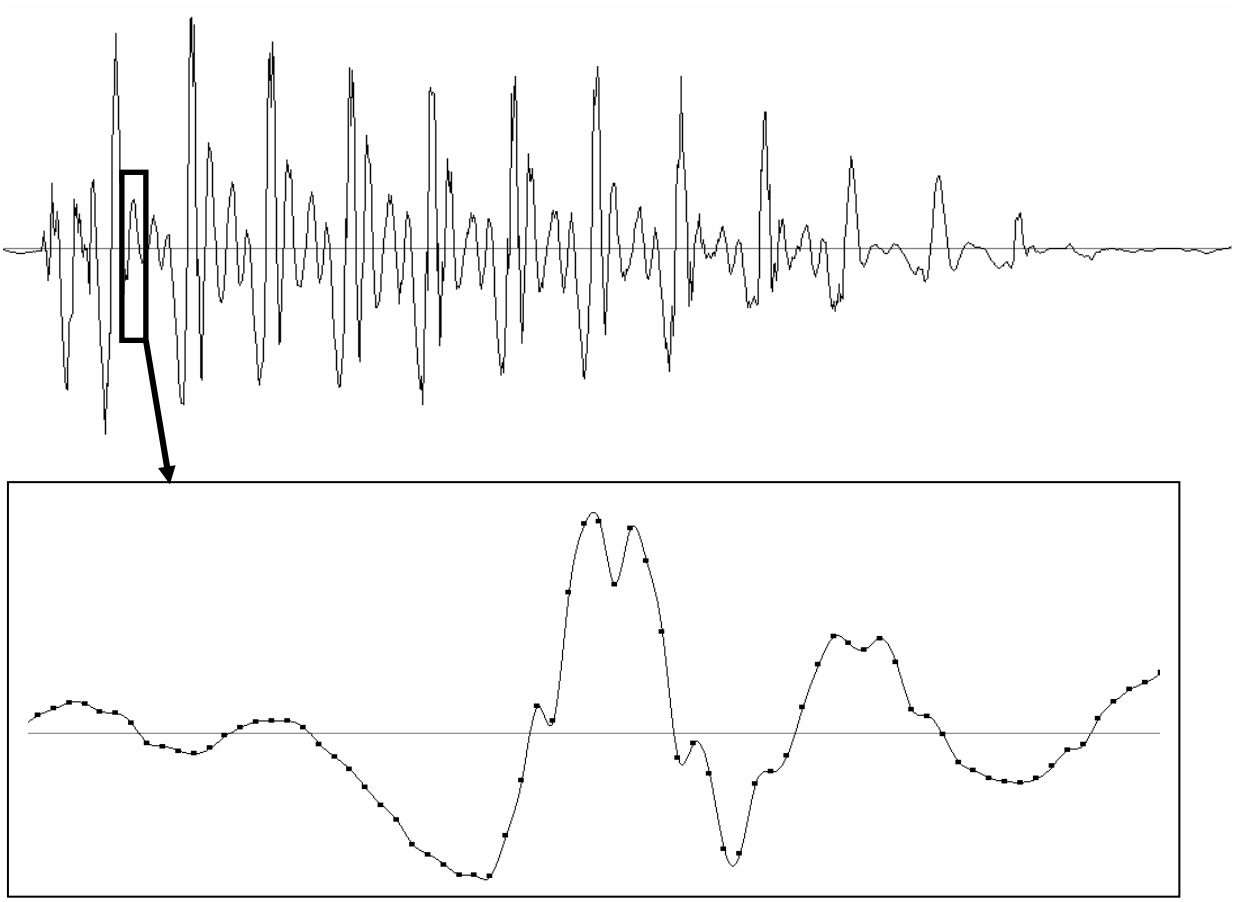
Отже, текстова інформація “Bit” кодується наступною послідовністю бітів

01000010 01101001 01110100.

Однією і з причин розробки інших таблиць кодування є обмеженість таблиці кодів ASCII (всього 256 символів). Тому широко використовуваною є кодова таблиця *UNICODE*, у якій кожному символу ставиться у відповідність 2 байти=16 біт. Шістнадцять розрядів дозволяють закодувати 2^{16}=65536 символів, що дозволяє розмістити багато національних алфавітів. Саме завдяки цьому на даний момент таблиця кодів *UNICODE* набуває все більшої популярності.

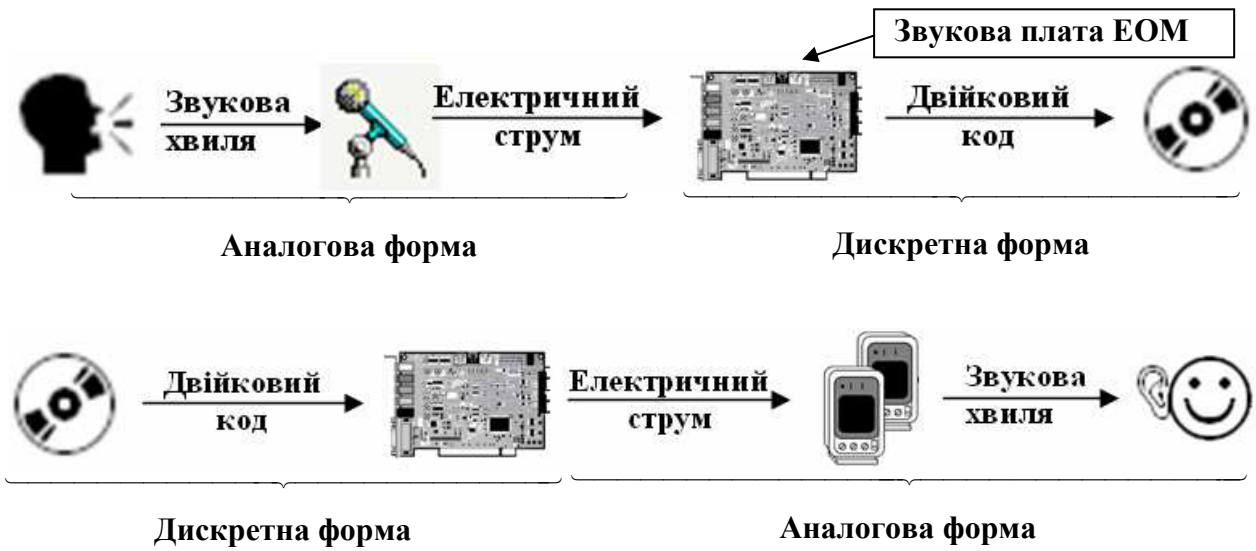
Представлення звукової інформації

З фізичної точки зору звук є неперервними коливаннями (неперервною звуковою хвилею), що можуть поширюватися в повітрі чи іншому пружному середовищі. Одиницею виміру звуку є децибели. Розглянемо графічне зображення звукової хвилі, яка утворюється при вимовлянні слова “Біт”.



Звукова хвиля є неперервною (аналоговою), тому для її збереження застосовують розглянутий раніше метод дискретизації у часі. Кількість вимірів рівня звуку за одну секунду називають *частотою дискретизації*, і вимірюють у герцах (поширеними є частоти 22КГц та 44,1КГц). Зрозуміло, що чим більша частота дискретизації, тим більша якість збереженого звуку, але і більший об'єм пам'яті необхідний для його збереження. Якість звуку залежить також від *розрядності перетворення* – кількості біт, які виділяються для збереження одного виміру (поширеними величинами є 8, 16 та 24 біти).

Загалом кажучи, процеси перетворення звуку у дискретну (цифрову) форму, та відтворення звуку відбувається за такими схемами



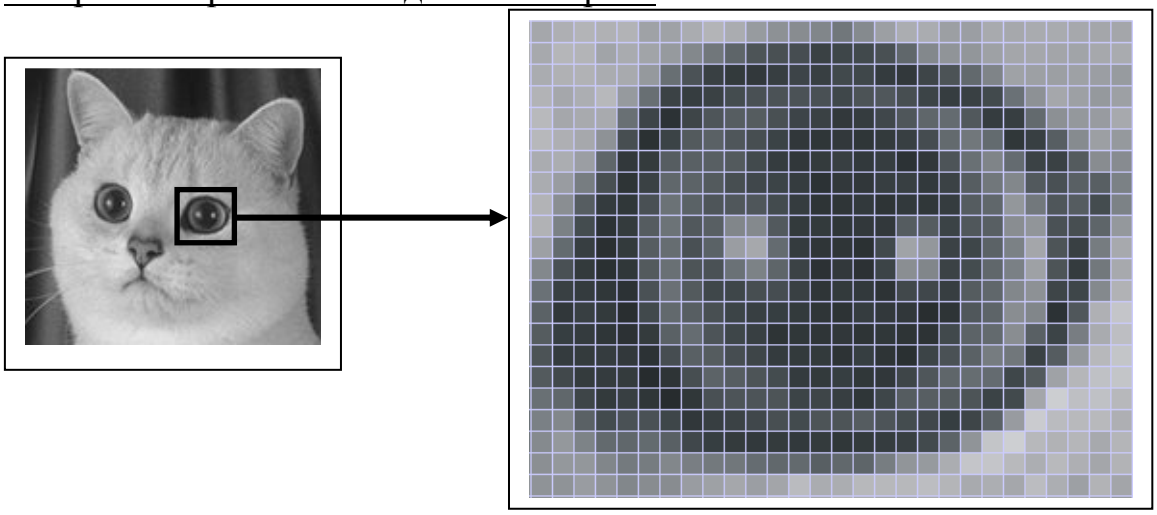
Файли, у яких зберігається звукова інформація, називають *аудіофайлами.* В залежності від способу збереження дискретної звукової інформації, розрізняють різні формати аудіофайлів:WAV, MIDI, MP3 та інші.

Представлення графічної інформації

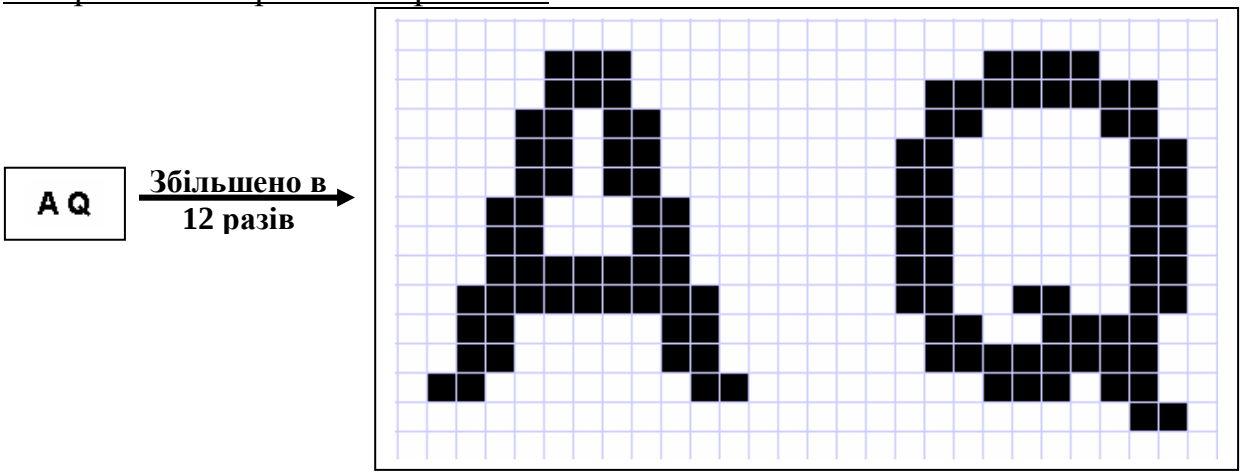
Графічна інформація може подаватися за допомогою графічних зображень: малюнків, креслень, фотографій, зображень на екрані монітора, телевізора і т.д. По своїй структурі графічні зображення можуть бути *растровими* та *векторними***.**

Розглянемо приклад растрових та векторних зображень.

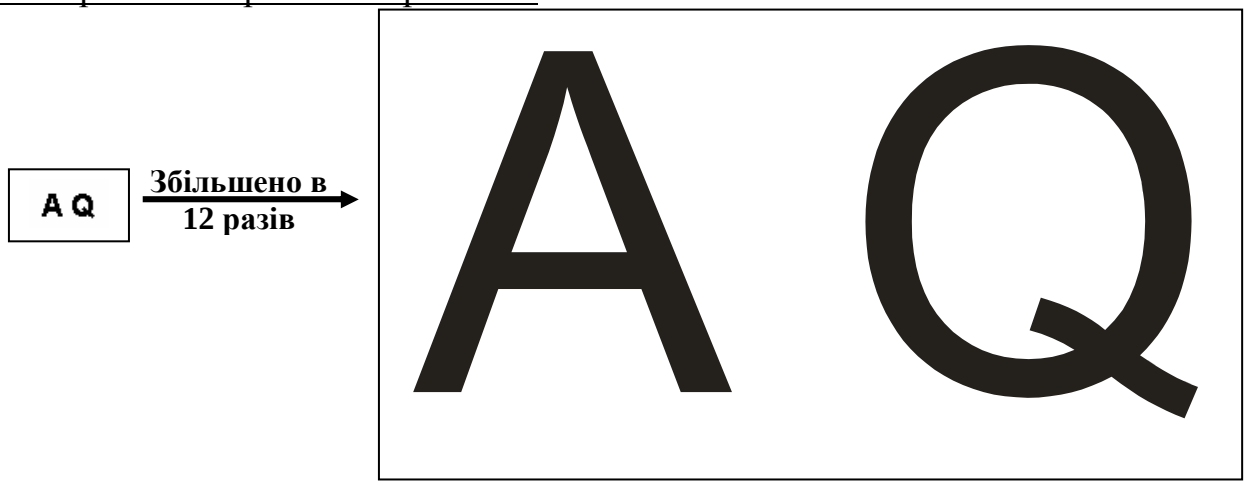
Растрове зображення з відтінками сірого



Растрове монохромне зображення



Векторне монохромне зображення



Як бачимо на збільшеному растровому зображенні, воно складається з маленьких точок – пікселів (PICSEL–**PIC**ture'**S** **El**ement – елемент зображення). Піксель є мінімальним елементом зображення одного кольору. Весь масив пікселів називають *растром* (від латинського "rastrum"–граблі). Кількість пікселів, які припадають на одиницю довжини, характеризують *роздільну здатність растру*. Значення роздільної здатності записують в одиницях *dpi* (doter per inch – кількість точок на одному дюймі (2,54см). Масив пікселів розбито на рядки і стовпці. Таким чином, кожен піксель має координати (*x*,*y*), де *x* – номер рядка, *y* – номер стовпця, на перетині яких він знаходиться. Отже, для збереження растрового графічного зображення необхідно зберегти інформацію про колір кожного пікселя. При цьому кожному кольору ставиться у відповідність деяке число. Кількість біт, яку виділяють для збереження кольору одного пікселя, називають *глибиною кольору*. Глибина кольору визначає скільки різних кольорів може бути збережено. Так кількість різних кольорів при глибині кольору n бітів дорівнює 2^n, як кількість різних комбінацій з n символів двійкового алфавіту.

Приклад. Розглянемо вміст растрового графічного файлу при збереженні монохромного зображення (рис. 8). Таке зображення характерне тим, що для збереження кольору одного пікселя необхідно всього один біт (наприклад, 0 – білий колір, 1 – чорний).

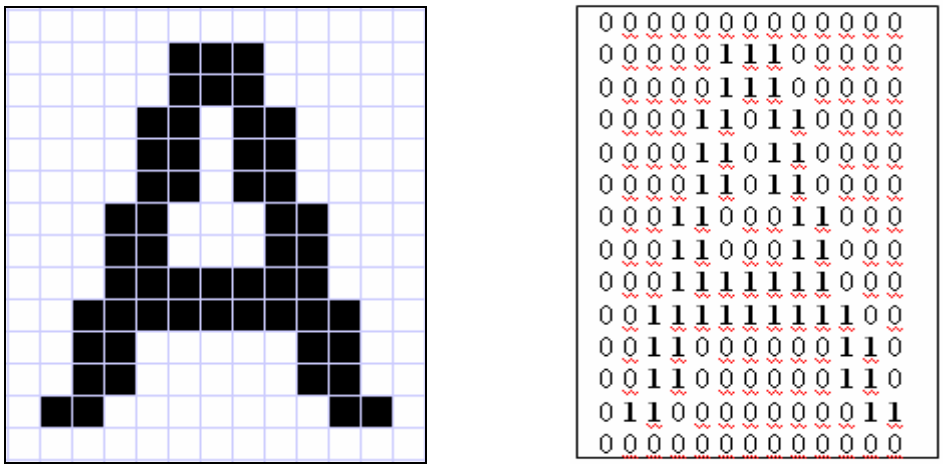
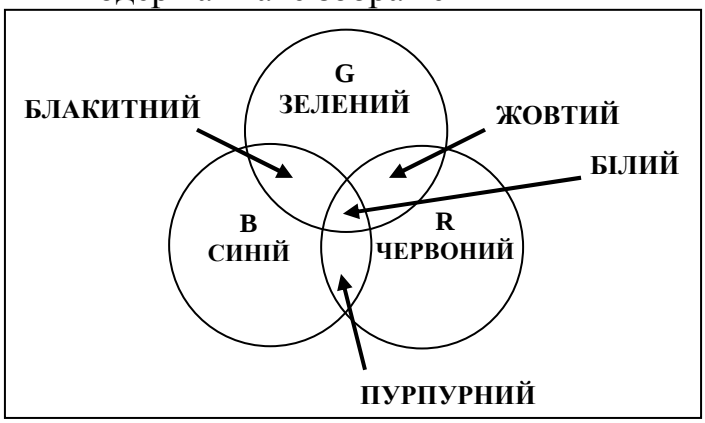


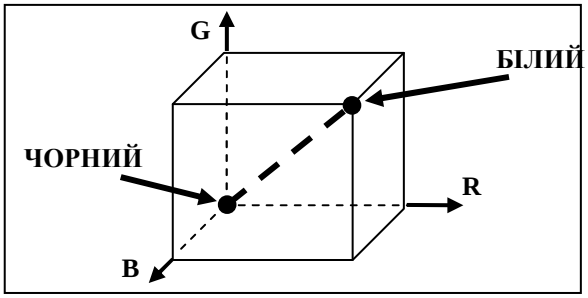
Рис. 8. Приклад монохромного зображення

При збереженні кольорового зображення, для представлення кольору використовується *адитивна кольорова модель RGB*. Ця модель використовується для опису кольорів, які можуть бути отримані за допомогою пристроїв, що ґрунтуються на принципі випромінювання (наприклад, монітори) та змішування фарб (принтери). В якості основних кольорів вибрано червоний (**R**ed), зелений (**G**reen) та синій (**B**lue). Інші кольори та відтінки можуть бути отримані змішуванням певної кількості кожного з основних кольорів.

Історія системи RGB почалася з дослідів Томаса Юнга. Взявши три ліхтарі з світловими фільтрами червоного, зеленого і синього кольору, на білому екрані він одержав таке зображення



Колір, який утворюється змішуванням трьох основних компонентів, можна представити вектором у тривимірній системі координат R, G і B, де координати виражають інтенсивності кожного з кольорів.



Чорному кольору відповідає центр координат – точка (0, 0, 0), білий колір виражено максимальним значенням компонентів. Нехай максимальне значення уздовж кожної осі дорівнює 1, тоді білий колір – це вектор (1, 1, 1). Точки, що лежать на діагоналі куба від чорного до білого, мають однакові значення інтенсивності кожного з кольорів. Це градації сірого – їх можна вважати білим кольором різної яскравості. Загалом кажучи, якщо колір, тобто всі компоненти відповідного вектора (*r*, *g*, *b*), помножити на однаковий коефіцієнт k, то колір (*kr*, *kg*, *kb*) зберігається, змінюється тільки яскравість.

Дослідження показали, що не всі насичені кольори можуть бути представлені сумішшю зазначених трьох компонентів. Але не дивлячись на це, система RGB широко використовується на даний час в кольорових телевізорах і дисплеях комп'ютерів. Отже, при збереженні кольорового

зображення, потрібно зберегти колір кожного окремого пікселя, як інтенсивності складових кольорів.

Растрові зображення мають значний недолік, пов’язаний із якістю зображення. Як показано на рисунку растрового зображення, при його збільшенні зображення спотворюється. Звичайно, покращити якість зображення можна збільшивши кількість пікселів, але це призведе до збільшення об’єму пам’яті під час його збереження. Цього недоліку позбавлені векторні зображення, у яких розмір будь-якого елемента може бути змінений аж “до нескінченності”. Але таке зображення неможливо отримати шляхом сканування, оскільки кожний її елемент будується за допомогою геометричних об’єктів або примітивів, в якості яких можуть виступати лінії, дуги, кола і т.д. Саме завдяки тому, що зображення зберігається у вигляді набору математичних формул, які описують його елементи, векторна графіка дозволяє легко змінювати розміри статичних зображень без втрати чіткості їхніх меж. Векторна графіка створюється за допомогою спеціальних програмних засобів типу Adobe Photoshop, CorelDRAW та ін.

Файли, в яких зберігають графічні зображення, називають *графічними файлами*. В залежності від типу графічного зображення, кількості присутніх у ньому кольорів, способу збереження (із стисненням, чи без) і т.д., розрізняють різні формати графічних файлів: BMP, JPEG, PNG, TIF, GIF, CDR, PSD і інші.

Література

1. Вильямс Р., Маклин К. Компьютеры в школе: Пер. с англ. – К.: Рад. шк., 1988 г.
2. Глинський Я.М. Практикум з інформатики: Навчальний посібник. 2-е вид. – Львів: “Підприємство Деол”, 1999 р.
3. Информатика. Базовый курс. 2-е издание /Под ред. С. В. Симоновича. – СПб.: Питер, 2005. — 640 с
4. Комп'ютерна техніка та програмування : Підручник / В.М. Сидоренко. К.: КДЕУ, 1994.
5. Лесничая И.Г., Миссинг И.В., Романова Ю.Д., Шестаков В.И. Информатика и информационные технологии. Учебное пособие/Под ред. Романовой Ю.Д. – М.: Изд-во Эксмо, 2005 г.
6. Майстренко А. В. Информатика: Учеб. пособие. Тамбов: Изд-во Тамб. гос. тех. ун-та, 2002. Ч. I. 96 с.
7. Мюллер Скотт Модернизация и ремонт ПК, 16_е издание. : Пер. с англ. – М.: Издательский дом “Вильямс”, 2006. – 1328 с
8. Нортон П., Гудман Дж. Персональный компьютер: аппартно-программная организация: пер. с англ. – СПб.: BHV – Санкт-Петербург, 1999. – 848с.
9. Следзінський І.Ф., Василенко Я.П. Основи інформатики. Посібник для студентів. – Тернопіль: Навчальна книга – Богдан, 2003.
10. Степанов А.Н. Информатики: Учебник для вузов. 4-е изд. –СПб.:Питер, 2005. – 684 с.
11. Шафрин Ю.А. Информационные технологии: В 2ч. Ч.2: Офисная технология и информационные системы. – М: Лаборатория Базовых Знаний, 1999.

ЗМІСТ

Відповідальний за випуск: завідувач кафедрою системного аналізу і теорії оптимізації к. ф.-м. н., доц. Кузка О.І.

Автори: к. т. н., доц. Семйон І.В.,
к. ф.-м. н., доц. Чупов С.В.,
к. ф.-м. н., Брила А.Ю.,
к. пед. н., Апшай Н.І.

Рецензенти: д.т.н., проф. Головач Й.І.,
к.т.н., доц. Маляр М.М.

ОСНОВИ ІНФОРМАТИКИ

Методичні матеріали з організації самостійної роботи для студентів математичного факультету з дисципліни “програмування”