КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Россада Т.В., Русіна Н.Г., Федорова М.В.

ОСНОВИ ІНФОРМАТИКИ

навчально-методичний посібник
для студентів спеціалізації
035.10 Філологія (прикладна лінгвістика)

Київ — 2017

УДК 004.+519.76:378.147(075.8)
О74

Россада Т.В. Основи інформатики: навчально-методичний посібник для студентів спеціалізації 035.10 Філологія (прикладна лінгвістика) / Т.В. Россада, Н.Г. Русіна, М.В. Федорова // Київ. — 2017. — 176 с.

Рекомендовано до друку вченою радою факультету комп'ютерних наук та кібернетики (протокол № 1 від 11 вересня 2017 року).

Навчально-методичний посібник містить теоретичні відомості та практично-методичні розробки з курсу «Основи інформатики» для студентів спеціалізації «Прикладна лінгвістика». Дисципліна «Основи інформатики» є базовою та необхідною для подальшого вивчення мов та технологій програмування, а також для розв'язання широкого кола лінгвістичних задач методами комп'ютерної лінгвістики. Курс дозволяє студентам зрозуміти математичні основи роботи комп'ютерної техніки, принципи збору, накопичення, оброблення та пошуку інформації. Посібник містить зразки практичних завдань, змістового та тестового контролю.
Посібник призначено для студентів і аспірантів та для широкого кола фахівців у галузі прикладної (комп'ютерної) лінгвістики.
Методика викладання цього курсу була апробована у навчальному процесі КНУ імені Тараса Шевченка і дала позитивні результати.

Рецензенти:
доктор філологічних наук, професор Дарчук Н.П.,
доктор фізико-математичних наук, професор Дорошенко А.Ю.,
кандидат філологічних наук, доцент Зубань О.М.

Вступ

Сучасний стан розвитку інформаційних технологій диктує свої правила для науковців: ретельне та якісне дослідження у сучасному науковому світі неможливе без застосування можливостей комп'ютерної техніки. Використання комп’ютера потребує вміння застосовувати сучасні технології, інформаційні системи, засоби конструювання та розробки програмних систем.

Навчально-методичний посібник призначений для підготовки студентів спеціалізації 035.10 Філологія (прикладна лінгвістика) відповідно до вимог державних стандартів вищої освіти.

Підготовка фахівців освітньо-кваліфікаційного рівня бакалавр спрямована на здобуття спеціальних умінь, знань та набуття навичок достатніх для виконання професійних завдань, тобто формування компетентностей майбутніх комп’ютерних лінгвістів.

У результаті вивчення курсу студент має *знати*:

- що таке інформація та які властивості вона має;
- із яких операцій складається процес оброблення даних;
- у яких одиницях та якими способами вимірюється кількість та ємність інформації;
- на які рівні поділяється програмне забезпечення, що являє собою операційна система та які її функції;
- для чого слугують інструментальні мови та системи програмування;

- чим відрізняються мови високого рівня від мов низького рівня;
- що таке алгоритм, які його властивості та які існують способи формального опису алгоритму;

та *вміти*:

- працювати із сучасним програмним забезпеченням загального та спеціального призначення;
- оптимально обирати програмне та технічне забезпечення залежно від рівня складності завдань;
- описувати алгоритми за допомогою формальних мов.

По завершенню вивчення дисципліни студент має набути навички роботи з комп'ютером та управління інформацією, вільно володіти основним комп'ютерним інформаційним інструментарієм у подальшому вирішенні лінгвістичних завдань, мати базові знання в галузі інформатики і сучасних інформаційних технологій та бути впевненим користувачем ПК (персонального комп'ютера).

Місце в структурно-логічній схемі навчального плану освітньо-професійної програми освітньо-кваліфікаційного рівня «бакалавр» спеціалізації 035.10 Філологія (прикладна лінгвістика). Нормативна навчальна дисципліна «Основи інформатики» входить до циклу «Обов'язкові навчальні дисципліни» ОПП «Прикладна (комп'ютерна) лінгвістика та англійська мова» і є базою для вивчення низки інших нормативних курсів («Основи програмування», «Об'єктно-орієнтоване програмування», «Бази даних та бази знань», «Автоматичний лінгвістичний аналіз») а також спецкурсів з інформаційних технологій та комп'ютерної лінгвістики циклу «Дисципліни вільного вибору студента».

1. Системи числення

Вступ

Історія *числа* почалася в той момент, коли людина навчилася розрізняти один предмет від багатьох. Аналіз мов первісних племен виявив, що наші предки спочатку не мали абстрактного поняття про число, проте використовували позначення для елементів деякої кількості. Так, були окремі слова для «одна людина», «дві людини», «три людини», і окремі для «одна сокира», «дві сокири» та «три сокири». Такі послідовності були достатньо короткими й завершувались поняттям «багато».

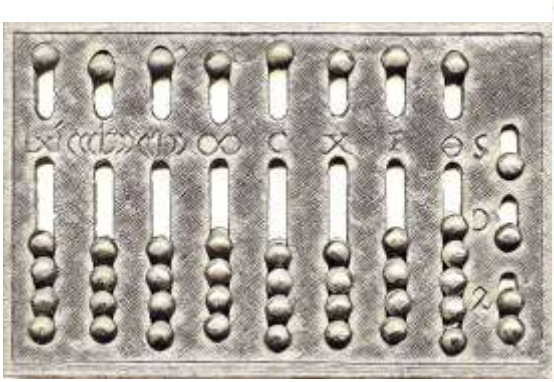
Поняття *числення* з'явилося в той момент, коли *кількість* предметів почала співставлятися з предметами деякої *еталонної сукупності*. У більшості племен *еталоном* виступали пальці рук, що вплинуло на назви чисел у різних мовах, в тому числі в українській [29].

Першими записами числа були риски (або зарубки), кожна з яких позначала число «один». Із появою потреби записувати великі числа, люди почали придумувати та використовувати нові *знаки* (див. табл. 1.1).

Усвідомлення *кількісної характеристики* об'єкту як окремої сутності дало можливість *оперувати* числами: *обчислювати* та *рахувати*. Усвідомлення нескінченності числа дало поштовх для формулювання принципів позначення як завгодно великих чисел, що лягли в основу *позиційних систем числення*. А потреба проводити обчислювальні операції швидко та точно сприяла виникненню перших рахівниць (див. рис. 1.1-1.2).

Таблиця 1.1. Позначення чисел в різних системах числення

Система числення	**Позначення чисел**				
Арабська	0	1	2	3	4
	5	6	10	15	19
Римська		I	II	III	IV
	V	VI	X	XV	XIX
Мая					
					
Вавилонська					
					



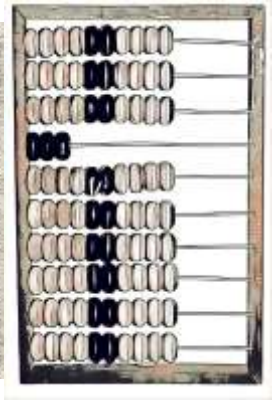


Рисунок 1.1. Абак

Рисунок 1.2. Рахівниця

Застосування *операцій* над числами (*додавання* та *віднімання*, *множення* та *ділення*), виявлення їх властивостей та правил дій та застосування їх для розв'язання широкого кола задач стимулювало розвиток науки про числа — *арифметики*.

Виявлені закономірності дозволяли класифікувати числа на *парні* та *непарні* (в залежності від подільності на 2), *прості* та *складені* (в залежності від кількості множників) тощо, а згодом автоматизувати процес обчислення, що сприяло появі перших *арифмометрів* (див. рис. 1.3) — механічних приладів для виконання арифметичних дій.

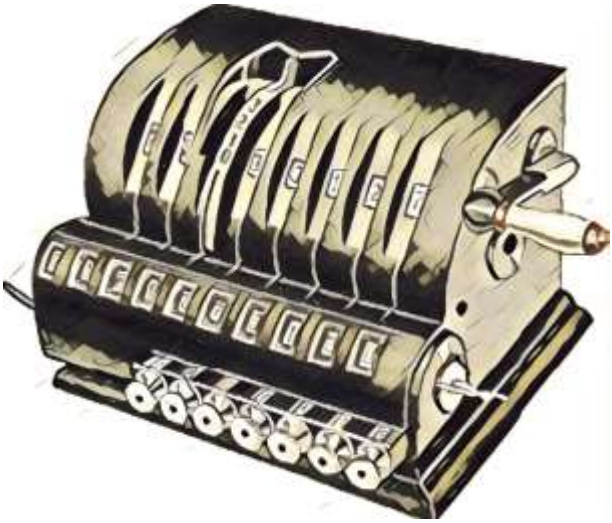


Рисунок 1.3. Перший варіант арифмометра В. Т. Однера

1.1. Історія виникнення чисел

Перші знаки для позначення чисел винайшли шумери — народ, який жив у 3000-2000 рр. до н. е. в Месопотамії (нині в Іраку). Історія свідчить, що на табличках з глини вони видавлювали клиноподібні рисочки, а згодом — спеціальні знаки. Деякі клинописні знаки позначали числа 1, 10, 100, тобто були цифрами, інші числа записувалися за допомогою з'єднання цих знаків. За допомогою цих позначок рахували дні тижня, голови худоби, розміри земельних ділянок, обсяги врожаю.

Вавилоняни, які прийшли в Месопотамію після шумерів, успадкували багато досягнень шумерської цивілізації — збереглися клинописні таблички з перекладом однієї одиниці вимірювання в інші. У Вавилоні система числення була шестидесяткова, в ній вперше застосовувався позиційний принцип, який згодом детальніше розглянемо. Позначення чисел повторюються після п’ятдесяти дев’яти, а їх величина визначається позицією цифри в числі як цілому. Користувалися цифрами і стародавні єгиптяни, про це свідчить математичний папірус Ринда, названий по імені англійського єгиптолога, який придбав його в 1858 р.в єгипетському місті Луксорі. На папірусі записані 84 математичні завдання з розв'язками. Судячи з цього історичного документу, єгиптяни користувалися такою системою цифр, в якій число позначалося сумою значень цифр.

Прообрази сучасних арабських цифр з'явилися в Індії пізніше V ст. Але індійські цифри в X-XIII ст. потрапили до Європи завдяки арабам, звідси і виникла назва — *арабські*. Велика заслуга у поширенні й виникненні індійських цифр в арабському світі належала працям двох

математиків: середньоазіатського вченого Хорезмі (бл. 780-бл. 850) і араба Кінді (бл. 800 – бл. 870). Хорезмі, який жив у Багдаді, написав арифметичний трактат про індійські цифри, який став відомий в Європі в перекладі італійського математика Леонардо Пізанського (Фібоначчі).

Текст Фібоначчі зіграв вирішальну роль у тому, що арабо-індійська система запису чисел вкоренилася на Заході. А арабська назва нуля — сифре — стало словом «*цифра*». Широке поширення в Європі арабські цифри отримали з другої половини XV ст.

Цікаві факти числа

Австралійські аборигени племені гумулгал, спосіб життя яких приблизно такий же, як в неоліті, користувалися системою числення, в якій було всього два слова для чисел: урапон — один, і укасар — два. Всі інші числа утворювалися з цих двох: урапон-укасар — 3, укасар-укасар — 4, укасар-укасар-урапон — 5 і т. д.

У таких країнах, як Китай, Японія і Корея число «4» вважається нещасливим. Тому поверхи з номерами, які закінчуються на «4» відсутні.

У деяких культурах число 3 вважають моторошним і невдалим. Наприклад, у В'єтнамі три людини на фотографії — це погана прикмета, оскільки вважається, що той, хто стоїть всередині, може померти.

Число 7 вважається найбільш щасливим числом. Існує 7 днів у тижні, 7 смертних гріхів і сім чеснот, 7 континентів, 7 кольорів веселки, 7 музичних нот, 7 днів Творіння та багато іншого. У Європі є повір'я, згідно з яким сьомий син 7-го сина володіє магічною силою. Також число 7 найчастіше є улюбленим числом людей у всьому світі.

1.2. Різновиди систем числення

Під *системою числення* (англ. numeral system або system of numeration) зазвичай розуміють сукупність прийомів та правил найменування і позначення чисел.

Формально, система числення — це метод подання (зображення, запису) чисел, що є спеціальною *формальною мовою*, *алфавітом* якої є множина символів, що називаються *цифрами*, а *синтаксисом* — правила, що дають змогу однозначно здійснити запис чисел.

Подання числа в певній системі числення називають *кодом числа* у цій системі числення.

Приклад 1.2.1. Звична нам *десяткова система числення* має алфавіт, що складається із десяти цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. А *римська система числення* має алфавіт, що складається із латинських літер: I, V, X, L, C, D, M. Число одинадцять у десятковій системі числення записується як «11», а в римській — як «XI». Ці два різних записи позначають одне число і вони є його кодом у відповідних системах числення.

За *методом інтерпретації запису числа* системи числення зазвичай поділяються на *непозиційні* (наприклад, унарна система числення), *позиційні* (наприклад, звична нам десяткова система числення) та *змішані* (наприклад, римська система числення). Розглянемо детальніше кожну з цих систем.

1.2.1. Непозиційні системи числення

Непозиційні системи числення виникли історично першими. Вони ґрунтуються на *кількісному підході* до визначення числа, який для кодування тих чи інших кількостей застосовував особливі знаки — цифри. Кожному такому знаку відповідав кількісний еквівалент.

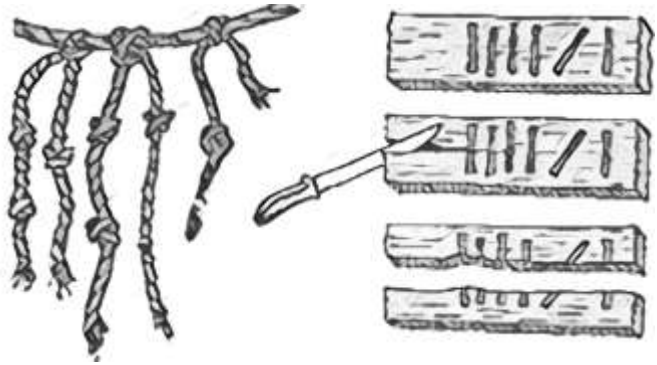
Формально, непозиційною є система числення, для якої значення символу, тобто цифри, не залежить від його положення в числі [14].

Зверніть увагу! Серед знаків більшості непозиційних чисел не було такого, який би відповідав нулю.

Розглянемо приклади непозиційних систем числення.

Унарна система числення

Найпростішим прикладом непозиційної системи числення є *унарна система числення.* В ній будь-яке натуральне число представляється у вигляді послідовності одиничок відповідної кількості. Раніше для запису таких чисел застосовували зарубки на деревах, вузлики на мотузках тощо (див. рис. 1.2.1.1).



а) *б)*

Рисунок 1.2.1.1. Стародавні засоби запису чисел
а) вузликове письмо (Південна Америка, VII ст. до н.е.)
б) дерево із зарубками (Чехія, 30 тис. рр. до н. е.)

На сьогоднішній день унарні числа застосовуються в деяких алгоритмах стискання даних [2]. Це поняття також

є основою для аксіом Пеано, що формалізують арифметику в рамках математичної логіки [4]. Форма унарної нотації також використовується для представлення чисел в лямбда численні [3].

Давньоєгипетська система числення

Давньоєгипетська система числення вживалася в Стародавньому Єгипті аж до початку X ст. У цій системі цифрами були ієрогліфічні символи; вони позначали числа 1, 10, 100 і т. д. до мільйона (див. табл. 1.2.1.1).

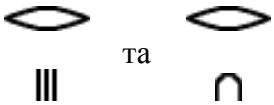
Таблиця 1.2.1.1. Позначення чисел у давньоєгипетській системі числення

Знак	***Число***	***Опис***
	1	Одна риска
	10	П'ята
	100	Петля мотузки
	1 000	Лілія (або лотос)
	10 000	Палець
 або	100 000	Жаба або пуголовок
	1 000 000	Людина з піднятими вгору руками

Для позначення арифметичних операцій в цій системи використовувалися спеціальні ієрогліфи *плюс* та *мінус*:  та  відповідно [1].

Древні єгиптяни також застосовували операції над дробовими числами, використовуючи для позначення дробу ієрогліф  .

Приклад 1.2.1.1. Числа $\frac{1}{3}$ та $\frac{1}{10}$ в давньоєгипетській системі числення позначаються наступним чином:



У древніх єгиптян також були спеціальні символи для дробів $\frac{1}{2}$, $\frac{2}{3}$ і $\frac{3}{4}$, якими можна було записувати також інші дроби, більші ніж $\frac{1}{2}$:

 ,  та  відповідно.

Грецька система числення

Грецька система числення використовує для представлення чисел *літери грецької абетки (див. табл. 1.2.1.2).* Грецькі цифри також відомі під назвами: *іонічні цифри, мілетські цифри, александрійські цифри або алфавітні цифри*. У сучасній Греції вони використовуються і нині на позначення порядкових числівників.

В грецькій системі числення були свої методи проведення операції над числами: додавання, віднімання, множення та ділення (див. рис. 1.2.1.2).

Таблиця 1.2.1.2. Позначення чисел у грецькій системі числення

Знак	Число	Знак	Число	Знак	Число	Знак	Число
α'	1	ι'	10	ρ'	100	͵Α	1000
β'	2	κ'	20	σ'	200	͵Β	2000
γ'	3	λ'	30	τ'	300	͵Γ	3000
δ'	4	μ'	40	υ'	400	͵Δ	4000
ε'	5	ν'	50	φ'	500	͵Ε	5000
ϛ'	6	ξ'	60	χ'	600	͵Ϛ	6000
ζ'	7	ο'	70	ψ'	700	͵Ζ	7000
η'	8	π'	80	ω'	800	͵Η	8000
θ'	9	ϟ'	90	ϡ'	900	͵Θ	9000

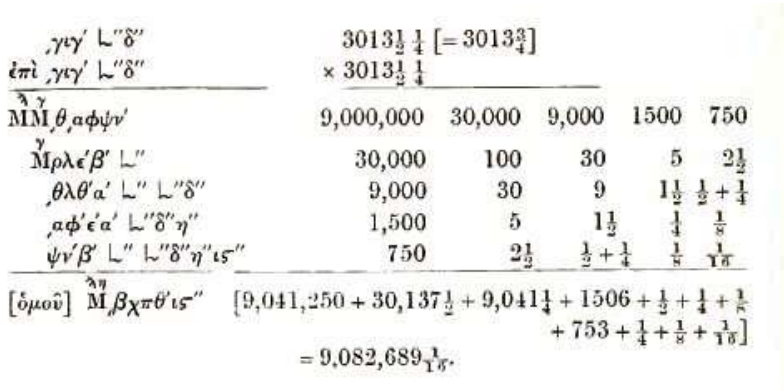


Рисунок 1.2.1.2. Приклад множення за рукописом Евтохія: ліворуч — грецькі цифри, праворуч — арабські

Старослов'янська система числення

У Київській Русі для рахунку і запису використовувалися *слов'янські цифри*. В цій системі числення застосовувалися символи абетки в лексикографічному порядку. Багато в чому вона схожа з грецькою системою написання цифрових символів. *Слов'янські цифри* — це позначення чисел за допомогою букв давніх алфавітів — *кирилиці* і *глаголиці* (рис. 1.2.1.3).

Для того щоб відрізнити букву від цифри, використовувався спеціальний значок — титло. Всі слов'янські цифри мали його над буквою. Символ пишеться зверху і являє собою хвилясту лінію. Цей знак використовується також в інших стародавніх системах рахунку. Він лише трохи змінює свою форму. Спочатку такий вид позначення прийшов від Кирила та Мефодія, оскільки нашу абетку вони розробили на основі грецької. Титло писалося як з більш округлими краями, так і з

гострими. Обидва варіанти вважалися правильними і використовувалися повсюдно.

Позначення цифр на письмі відбувалося зліва направо. Виняток становили числа з «11» до «19». Вони писалися справа наліво. Історично це збереглося в назвах сучасних числівників (один-на-дцять, два-на-дцять і т. д., тобто першою стоїть буква, що позначає одиниці, другий — десятки). Кожна буква алфавіту позначала цифри від 1 до 9, з 10 до 90, з 100 до 900.

Не всі букви слов'янського алфавіту застосовувалися для позначення цифр. Так, «Ж» і «Б» не використовувалися для нумерації. Їх просто не було в грецькому алфавіті, який був прийнятий в якості зразка). Також відлік починався з одиниці, а не зі звичного для нас нуля. Іноді на монетах використовувалася змішана система позначення цифр — з кирилиці і арабських літер. Найчастіше використовувалися тільки рядкові літери.

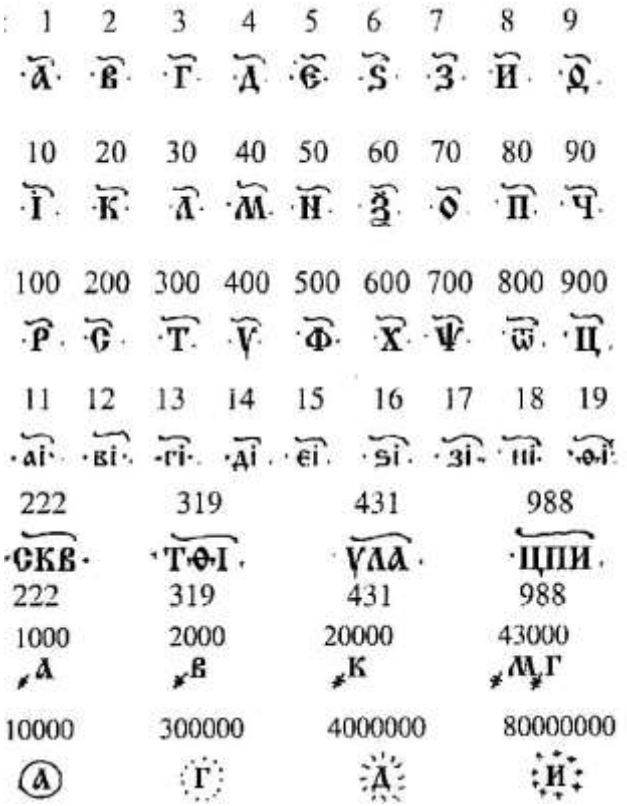


Рисунок 1.2.1.3. Приклади запису чисел кирилицею

Наші предки за допомогою спеціальних позначень писали дати і необхідні числа в літописах, документах, монетах, листах. Складні числа до 999 позначалися кількома літерами поспіль під загальним знаком «титло» (див. рис 1.2.1.3).

Слов'янські цифри, які позначали число більше 1000 писалися зі спеціальним знаком. Його ставили перед

потрібною літерою з титло. Якщо необхідно було написати числівник більше 999 використовувалися спеціальні знаки: до «Аз» ліворуч приписували нахильну рису з двома поперечними рисками — 1000; «Аз» у колі — 10000 (тьма); «Аз» у колі з крапок — 100000 (легіон); «Аз» в колі, що складається з ком — 1000000 (леодр). У ці кола поміщається буква з необхідним цифровим значенням.

Таке позначення можна було зустріти в документації і на давніх монетах. Перші подібні цифри можна побачити на петровських срібних монетах в 1699 році. З таким позначенням вони карбувалися 23 роки. Ці монети зараз відносяться до раритетів і дуже цінуються.

Недоліки непозиційних систем числення

Серед суттєвих недоліків непозиційних систем числення можна виділити наступні. По перше, кількість чисел, яку можна одержати за допомогою непозиційного числення, обмежена алфавітом. По друге, з такими числами важко виконувати арифметичні й логічні операції.

1.2.2. Змішані системи числення

Якщо у непозиційних системах числення значення цифри жодним чином не залежить від її позиції у числі, то у *змішаних системах* є виключення.

Формально, змішаною є система числення, для якої значення символу частково залежить від його положення в числі. Розглянемо приклад змішаної системи числення.

Римська системи числення

Римська система числення використовувалися стародавніми римлянами. Вона базується на застосування особливих знаків (літер латинської абетки) для десяткових розрядів I = 1, X = 10, C = 100, M = 1000 та їх половин V = 5, L = 50, D = 500.

До речі, літери для позначення цифр в римській системі числення вибрані не просто так: I (1) і V (5) схематично позначають 1 та 5 пальців відповідно, аналогічно X (10) це дві долоні (див. рис.1.2.2.1). Цифра L, що позначає число 50, виникла шляхом трансформації половинки «зірочки» яка раніше позначала число 100 (див. рис. 1.2.2.2). Цифри C (100), D (500) та M (1000) виникли від початків їх назв: Centum, Demimille та Mille відповідно.

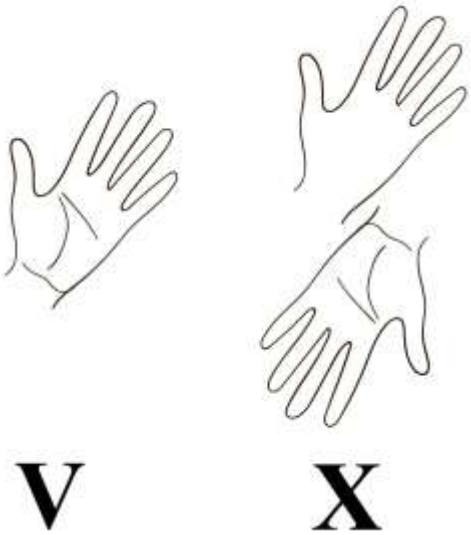


Рисунок 1.2.2.1. Позначення цифр 5 і 10 в римській системі числення

Рисунок 1.2.2.2. Трансформація цифри L в римській системі числення

Для запам’ятовування позначення цифр римської системи числення пропонуємо наступні *мнемонічні правила*:

1000 M — Ми

500 D — Допитливі

100 C — Сучасні

50 L — Лінгвісти

10 X — Хочемо

5 V — Вивчати

1 I — Інформатику

Натуральні числа записуються за допомогою повторів цифр. При цьому, якщо більша цифра стоїть перед меншою, то вони додаються (принцип додавання), якщо ж менша перед більшою, то менша віднімається від більшої (принцип віднімання). Останнє правило застосовується тільки для уникнення чотириразового повторення однієї цифри.

Приклад 1.2.2.1. У римській системі числення знаку X відповідає кількість елементів 10. Запис XI (цифра I стоїть справа від X) позначає число 11, проте запис IX (I стоїть зліва) позначає число 9. Таким чином цифра I позначає як 1, так і -1, в залежності від того, в якому місці запису числа вона стоїть.

Римська система числення на сьогодні майже не застосовується, бо виконання арифметичних дій над

багатозначними числами в цій системі дуже незручне. Тим не менш, її використовують для позначення розділів і частин законів, томів видань, століть, інколи років, днів тижня, місяців у датах (1.V.1975), на циферблатах деяких годинників, порядкових числівників, а також похідних, номер яких більший за три ($у^{IV}$, $у^{V}$), а також з естетичною метою.

Рисунок 1.2.2.3. Римські числа на вітражі інституту філології КНУ імені Тараса Шевченка

Розглянемо приклади переведення із римської системи числення в звичну нам десяткову.

Приклад 1.2.2.2. Переведіть у десяткову систему числення число XXVI.

Розв'язання: $XXVI = 10 + 10 + 5 + 1 = 26$.

Відповідь: 26.

Приклад 1.2.2.3. Переведіть у десяткову систему числення число XXCIX.

Розв'язання: $XXCIX = 100 - 10 - 10 + 10 - 1 = 89$.

Відповідь: 89.

Приклад 1.2.2.4. Переведіть у десяткову систему числення число CXXXI.

Розв'язання: $CXXXI = 100 + 10 + 10 + 10 + 1 = 131$.

Відповідь: 131.

Приклад 1.2.2.5. Переведіть у десяткову систему числення число MXCIX.

Розв'язання: $MXCIX = 1000 + 100 - 10 + 10 - 1 = 1099$.

Відповідь: 1099.

Приклад 1.2.2.6. Переведіть у десяткову систему числення число MDXI.

Розв'язання: $MDXI = 1000 + 500 + 10 + 1 = 1511$.

Відповідь: 1511.

Приклад 1.2.2.7. Переведіть у десяткову систему числення число CDLXXXIIV.

Розв'язання:

$CDLXXXIIV = 500 - 100 + 50 + 10 + 10 + 10 + 5 - 2 = 473$.

Відповідь: 473.

1.2.3. Позиційні системи числення

У позиційних системах числення значення цифри цілком і повністю залежить від її *позиції в записі*. Такою є звична для нас *десяткова система числення*.

Формально, позиційною називається система числення, в якій значення кожної цифри залежить від її числового еквіваленту та від її місця (позиції) в числі, тобто один символ (цифра) може приймати різні значення.

Приклад 1.2.3.1. У числі 777 перша 7 означає число 700 (а не 7), друга — 70 (а не 7), а третя — 7. Тобто один і той самий знак має різне значення в залежності від позиції в числі.

Позиційна система числення визначається *основою* та *алфавітом*.

Основа системи числення — постійне для даної системи числення відношення одиниць сусідніх розрядів.

Алфавіт системи числення — знаки для позначення цифр. Наприклад, звична нам десяткова позиційна система числення має основу 10, а її алфавіт складається із десяти цифр: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}.

Основа визначає число 10 для даної системи. Тобто в системі числення з основою n рахуємо наступним чином:

$$1, 2, \ldots, n-1, 10, \ldots$$

Основу числа зазвичай вказують нижнім індексом числа.

Приклад 1.2.3.2. Запис 112_3 означає число 112 у системі числення з основою 3. Рахуючи в трійковій та десятковій відповідним чином бачимо відповідність між числами (див. табл. 1.2.3.1), а саме, що 112_3 це 14_{10}.

Таблиця 1.2.3.1.Відповідність між системами числення з основами 10 та 3

Основа	Послідовні числа													
10	1	2	3	4	5	6	7	8	9	10	11	12	13	14

3	1	2	10	1 1	12	20	2 1	22	10 0	10 1	10 2	11 0	11 1	11 2

Надалі, якщо в записі числа нижній індекс відсутній, будемо вважати число записаним у десятковій системі числення (наприклад, замість 14_{10} будемо писати 14).

У позиційній системі числення з основою x число n з k розрядами подають у вигляді лінійної комбінації степенів числа x:

$$n = a_0 x^{k-1} + a_1 x^{k-1} + .. + a_{k-1} x^0$$

а саме число коротко подається таким чином:

$$a_0 a_1 \ldots a_{k-1}$$

Окрему позицію в зображенні числа прийнято називати *розрядом*, а номер позиції — *номером розряду*. Число розрядів у записі числа називається *розрядністю числа* [40].

Десяткова система числення це не єдина позиційна система числення, що використовується на сьогодні. Наприклад, індіанці юккі в Калифорнії використовують чотиризначну систему числення: вони рахують на проміжках між пальцями. Вавілоняни дуже любили число 60, їх вплив зберігається в нашій 60-хвилинній годині і 60-годинній хвилині. Розглянемо її детальніше.

Вавилонська система числення

Вавилонська система числення виникла приблизно за 2000 р. до н. е. Вона має незвичний для нас алфавіт (рис. 1.2.3.1) та основу 60. На сьогодні ця система числення застосовується для позначення співвідношення між градусом, годиною, хвилиною та секундою.

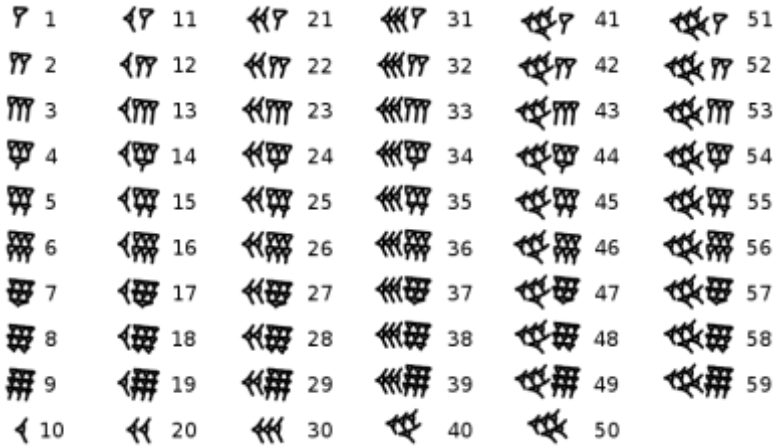


Рисунок 1.2.3.1. Позначення чисел у вавилонській системі числення

Вавилоняни записували всі числа від 1 до 59 у десятковій (але не позиційній) системі за допомогою повторення двох «клинів». Число 60 (одиницю вищого розряду) записували так само, як і 1, але на більшій відстані від інших клинів. Цілі числа, більші за 59, записували в позиційній шістдесятковій системі. Клини, якими записували числа, могли щільно прилягати один до одного. Щоб уникнути неправильного читання, між розрядами залишали помітний проміжок (прогалину).

Знака для зображення нуля у вавилонській системі числення не було, а щоб показати відсутність якого-небудь розряду в записі числа, вавилоняни робили більший проміжок між значущими розрядами, ніби вставляючи цим самим нульовий розряд. Але визначати величини проміжків між цифрами було незручно. Це ще більше посилювалося тим, що «нульового» розряду, якщо він був найменшим у записі числа, ніяк не позначали. Тому

нумерацію математичних клинописних текстів в деяких джерелах називають нумерацією з неозначеною позицією.

1.3. Системи числення в інформатиці

В інформатиці та обчислювальній техніці на сьогоднішній день застосовуються десяткова, двійкова, вісімкова та шістнадцяткова системи числення. Розглянемо їх детальніше.

1.3.1. Десяткова система числення

Десяткова система числення — позиційна система числення з основою 10. Одна з найбільш поширених систем. Алфавіт десяткової системи числення складається із 10 знаків: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0. За припущенням, основа 10 пов'язана із кількістю пальців на руках у людини, що використовувались для рахування.

У десятковій системі числення число *n* подають у вигляді лінійної комбінації степенів числа 10 (рис. 1.3.1.1):

$$n = a_{0}10^{k-1} + a_{1}10^{k-2} + .. + a_{k}10^{0}$$

де $a_0, a_1, \ldots, a_k$ — цифри числа.

Рисунок 1.3.1.1. Представлення десяткового числа

Переведення чисел до десяткової системи числення

Число у позиційній системі x можна звести до десяткової системи числення, підрахувавши значення лінійної комбінації степенів числа *x*. Розглянемо на прикладах.

Приклад 1.3.1.1. Перевести у десяткову систему числення число 8128_{12}.

Розв'язання.

Розкладаємо 8128_{12} на суму степенів 12:

$$\overset{3\,2\,1\,0}{8128}_{12} = 8 \cdot 12^3 + 1 \cdot 12^2 + 2 \cdot 12^1 + 8 \cdot 12^0 = 14000$$

Відповідь: $8128_{12} = 14000$.

Приклад 1.3.1.2. Знайти систему числення в якій виконується рівність $46_x = 58$?

Розв'язання.

Розкладаємо 46_x на суму степенів x:

$$58 = \overset{1\,0}{46}_x = 4 \cdot x^1 + 6 \cdot x^0 = 4x + 6$$

Розв'язуємо рівняння $58 = 4x + 6$:

$$x = (58 - 6) : 4 = 13$$

Відповідь: $x = 13$.

Приклад 1.3.1.3. Знайти основу системи, в якій виконується рівність $16_x + 33_x = 52_x$.

Розв'язання.

Переводимо у десяткову систему:

$$\overset{1\,0}{16}_x = 1 \cdot x^1 + 6 \cdot x^0 = x + 6$$

$$\overset{1\,0}{33}_x = 3 \cdot x^1 + 3 \cdot x^0 = 3x + 3$$

$$\overset{1\,0}{52}_x = 5 \cdot x^1 + 2 \cdot x^0 = 5x + 2$$

Підставляємо у початкову рівність:

$$x + 6 + 3x + 3 = 5x + 2$$

Розв'язуємо отримане рівняння $4x + 9 = 5x + 2$:

$$4x - 5x = 2 - 9$$

$$-x = -7$$

Відповідь: $x = 7$

Приклад 1.3.1.4. Знайти усі системи числення, в яких виконується нерівність $21_x + 32_x > 102_x$.

Розв'язання.

Переводимо числа у десяткову систему:

$$\overset{10}{21_x} = 2 \cdot x^1 + 1 \cdot x^0 = 2x + 1$$

$$\overset{10}{32_x} = 3 \cdot x^1 + 2 \cdot x^0 = 3x + 2$$

$$\overset{2\,10}{102_x} = 1 \cdot x^2 + 0 \cdot x^1 + 2 \cdot x^0 = x^2 + 2$$

Підставляємо у початкову рівність:

$$2x + 1 + 3x + 2 = x^2 + 2$$

Розв'язуємо нерівність $5x + 3 > x^2 + 2$:

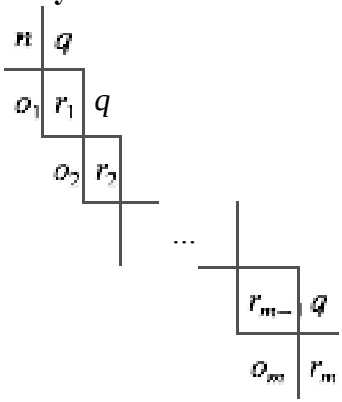
$-x^2 + 5x + 1 > 0$, D = 29

$$\frac{5 - \sqrt{29}}{-2} < x < \frac{5 - \sqrt{29}}{-2}$$

Цілочисельні розв'язки даної нерівності: 0, 1, 2, 3, 4 та 5. Проте в записах чисел є цифра 3, тобто система числення для даної нерівності має бути більшою за 3.

Відповідь: $x = 4$ або $x = 5$.

Переведення чисел з десяткової системи числення в іншу



Для переведення цілих чисел із десяткової в систему числення з основою q потрібно число, записане в десятковій системі числення, послідовно ділити на основу нової системи числення q, виділяючи остачі.

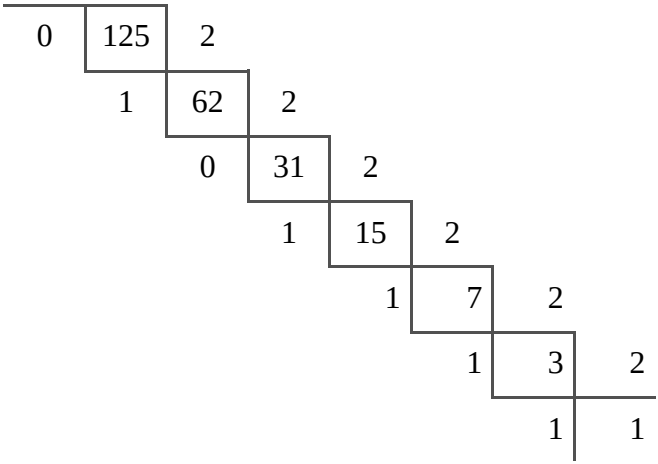
Остачі (тобто a_1, a_2, .. a_m, r_m, де $r_m < q$), записані у зворотному порядку (тобто $\overline{r_m a_m \ldots a_2 a_1}$), будуть являти собою запис числа в системі числення з основою q.

Приклад 1.3.1.5. Перевести число 1000 у систему числення з основою 2.

Розв'язання.

Для переведення числа 1000 у систему числення з основою 2 послідовно ділимо його на основу 2, фіксуючи остачі:

100 0	2	
0	500	2
	0	250	2



В результаті послідовного ділення отримали число 1, яке вже не можемо поділити на 2 і воно є останньою остачею.

Тепер остачі записуємо в оберненому порядку: 1111101000.

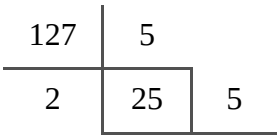
Отже: $1000 = 1111101000_2$.

Зверніть увагу! Система числення з основою 2 також називається *двійковою системою числення*. Детальніше ми розглянемо її в наступному розділі.

Приклад 1.3.1.6. Перевести число 127 у систему числення з основою 5.

Розв'язання.

Для переведення числа 127 у систему числення з основою 5 послідовно ділимо його на 5, фіксуючи остачі:



0	5	5
	0	1

Тепер остачі записуємо в оберненому порядку: 1002.

Отже: $127 = 1002_5$.

Приклад 1.3.1.7. Перевести число 678 у систему числення з основою 3.

Розв'язання.

Для переведення числа 678 у систему числення з основою 3 послідовно ділимо його на 3, фіксуючи остачі:

678	3				
0	226	3			
	1	75	3		
		0	25	3	
			1	8	3
				2	2

Тепер остачі записуємо в оберненому порядку: 221010.

Отже: $678 = 221010_3$.

Зверніть увагу! Для систем числення з основою більше десяти для позначення цифр, що більше дев'яти зазвичай використовують літери латинського алфавіту (10 позначають як A, 11 — як B, 12 — як C тощо). Розглянемо на наступному прикладі.

Приклад 1.3.1.8. Перевести число 2017 у систему числення з основою 13.

Розв'язання. Для переведення числа 2017 у систему числення з основою 13 послідовно ділимо його на основу 13, фіксуючи остачі:

2017	13	
2	155	13
	12	11

Ми бачимо, що серед остач є числа, що не є цифрами 0-9. І ми вже знаємо, що для систем числення з основою більше десяти для позначення цифр, що більше дев'яти використовують літери латинського алфавіту, тобто 11 позначається як B, а 12 — як C.

Отже: $2017 = BC2_{13}$.

Для переведення з однієї позиційної системи числення у будь-яку іншу позиційну систему можна використовувати аналогічні до наведених алгоритмів, маючі на увазі вказану систему числення замість десяткової. Проте арифметичні операції в цих алгоритмах треба виконувати у відповідних системах числення. Тому часто буває простіше перевести із зазначеної системи числення спочатку у десяткову, а потім з десяткової у потрібну систему числення. Розглянемо на прикладі.

Приклад 1.3.1.9. Перевести число 2017_8 із вісімкової у п'ятнадцяткову систему числення.

Розв'язання.

Спочатку переведемо 2017_8 у десяткову систему числення за вже відомим нам алгоритмом:

$$2017_8 = 2 \cdot 8^3 + 0 \cdot 8^2 + 1 \cdot 8^1 + 7 \cdot 8^0 = 1039$$

Тепер десяткове число 1039 переведемо у 15-ву систему числення:

1039	15	
4	69	15
	9	4

Отримали остачі 4, 9 та 4.

Отже: $2017_8 = 494_{15}$.

1.3.2. Двійкова система числення

Алфавіт двійкової системи числення складається із цифр 0 та 1, тобто для її застосування достатньо двох стійких станів фізичних елементів. Ця система є близькою до оптимальної за економічністю, тому вона є *найпоширенішою для подання, зберігання та оброблення даних у пам'яті комп'ютера*: за допомогою двійкових чисел кодується вся інформація на комп'ютері.

Взагалі, проблема вибору системи числення для подання чисел у пам'яті комп'ютера має велике практичне значення. При виборі звичайно враховуються такі вимоги, як надійність подання чисел при використанні фізичних елементів та економічність (використання таких систем числення, в яких кількість елементів для подання чисел із деякого діапазону була б мінімальною). Для подання цілих чисел від 1 до 999 у десятковій системі достатньо трьох розрядів, тобто трьох елементів. Оскільки кожен елемент може перебувати в десятьох станах, то загальна кількість станів — 30. А у двійковій системі числення $999_{10} = 1111100_2$, тобто необхідна кількість

станів — 14 (7 розрядів і кожен може перебувати у двох станах). Проте в такому розумінні є економічніша позиційна система числення — трійкова. Так, для запису цілих чисел від 0 до 999999999 у десятковій системі числення потрібно 90 станів, у двійковій — 60 ($30 \cdot 2$), у трійковій — 57 ($19 \cdot 3$). Але трійкова система числення не дістала поширення внаслідок труднощів фізичної реалізації.

Для переведення із двійкової системи числення у десяткову та навпаки застосовуються алгоритми, які були представлені вище. Проте можна використовувати простіший спосіб, якщо знати напам'ять степені двійки принаймні до десятої, що для програмістів є таким самим природнім, як знання таблиці множення для школяра.

Розглянемо *метод підбору* для переведення числа із десяткової у двійкову систему числення, використовуючи табличку 1.3.2.1 степенів двійки.

Таблиця 1.3.2.1. Степені двійки

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1024	512	256	128	64	32	16	8	4	2	1

Метод підбору полягає у тому, що для заданого числа шукаємо найбільшу степінь двійки, що менша за це число і віднімаємо від числа. З остачею повторюємо ту саму дію. Степені, що були отримані таким чином, визначають позиції (нумерація починається з нуля) одиничок у двійковому записі числа. Розглянемо на прикладі.

Приклад 1.3.2.1. Перевести число 70 у двійкову систему числення за допомогою методу підбору.

Розв'язання.

Ми бачимо з таблички, що найбільша степінь двійки, яка менша за 70, це шоста:

$$70 = 2^6 + 4$$

Ту саму операцію повторимо із остачею 4. Вона співпадає із другою степінню двійки:

$$70 = 2^6 + 2^2$$

Тобто результуючий двійковий запис числа складається із семи цифр, причому із них одиничи стоять на позиціях 6 та 2, а решта цифр — нулі: $\overset{6\,5\,4\,3\,2\,1\,0}{1000100}_2$.

Отже: $70 = 1000100_2$.

Приклад 1.3.2.2. Перевести число 702 у двійкову систему числення за допомогою методу підбору.

Розв'язання.

Ми бачимо з таблички, що найбільша степінь двійки, яка менша за 702, це дев'ята ($2^9 = 512$, а $2^{10} = 1024$):

$$702 = 2^9 + 190$$

Ту саму операцію повторимо із остачею 190. Найближча до цього числа степінь, що менша за нього, це сьома ($2^7 = 128$, а $2^8 = 256$):

$$702 = 2^9 + 2^7 + 62$$

Продовжуючи далі врешті решт отримаємо:

$$702 = 2^9 + 2^7 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1$$

Тобто результуючий двійковий запис числа складається із 10 цифр, причому із них одиничи стоять на позиціях 9, 7, 5, 4, 3, 2 та 1, а решта цифр — нулі: $\overset{9\,8\,7\,6\,5\,4\,3\,2\,1\,0}{1010111110}_2$.

Отже: $702 = 1010111100_2$.

Арифметичні операції у двійковій системі числення

Додавання та віднімання у стовпчик у двійковій системі числення виконується майже так само, як і у десятковій, але треба враховувати особливості цієї системи (див. табл. 1.3.2.2 та 1.3.2.3).

Таблиця 1.3.2.2. Правила додавання у двійковій системі числення

+	**0**	**1**
0	0	1
1	1	10

Таблиця 1.3.2.3. Правила віднімання у двійковій системі числення

–	**0**	**1**
0	0	
1	1	0
10	0	1

Розглянемо на прикладах.

Приклад 1.3.2.3. Знайти суму 101110_2 та 10011101_2, виконавши додавання у стовпчик.

Розв'язання. Виконуємо додавання поетапно.

Крок 1. Виконуємо додавання $0 + 1 = 1$:

\+ 1 0 1 1 1 0

$$\begin{array}{cccccccc} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ \hline & & & & & & & 1 \end{array}$$

Крок 2. Виконуємо додавання $1+0=1$:

$$\begin{array}{cccccccc} & & 1 & 0 & 1 & 1 & 1 & 0 \\ + & & & & & & & \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ \hline & & & & & & 1 & 1 \end{array}$$

Крок 3. Виконуємо додавання $1+1=10$. Оскільки при сумуванні двох одиничок ми отримали 10, то зайвий розряд треба перенести, а в результат записати 0:

$$\begin{array}{cccccccc} & & & & \cdot & & & \\ & & 1 & 0 & 1 & 1 & 1 & 0 \\ + & & & & & & & \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ \hline & & & & & 0 & 1 & 1 \end{array}$$

Крок 4. Виконуємо додавання $\cdot+1+1=1+1+1=11$. Тобто записуємо у результат одиничку і переносимо один розряд:

$$\begin{array}{cccccccc} & & & & \cdot & \cdot & & \\ & & 1 & 0 & 1 & 1 & 1 & 0 \\ + & & & & & & & \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ \hline & & & & 1 & 0 & 1 & 1 \end{array}$$

Крок 5. Виконуємо додавання $\cdot + 0 + 1 = 1 + 0 + 1 = 10$. Тобто записуємо у результат нулик і переносимо розряд:

```
          .  .  .
          1  0  1  1  1  0
+
   1  0  0  1  1  1  0  1
  ------------------------
            0  1  0  1  1
```

Крок 6. Виконуємо додавання $\cdot + 1 + 0 = 1 + 1 + 0 = 10$. Тобто записуємо у результат нулик і переносимо розряд:

```
       .  .  .  .
          1  0  1  1  1  0
+
   1  0  0  1  1  1  0  1
  ------------------------
          0  0  1  0  1  1
```

Крок 7. Виконуємо додавання $\cdot + 0 = 1 + 0 = 1$. Тобто записуємо у результат одиничку:

```
       .  .  .  .
          1  0  1  1  1  0
+
   1  0  0  1  1  1  0  1
  ------------------------
      1  0  0  1  0  1  1
```

Крок 8. Переписуємо у результат одиничку:

```
      .  .  .  .
         1  0  1  1  1  0
+
   1  0  0  1  1  1  0  1
  ------------------------
   1  1  0  0  1  0  1  1
```

Отже: $1011101_2 + 10011101_2 = 11001011_2$.

Приклад 1.3.2.4. Знайти різницю 10010101 та 11000, виконавши віднімання у стовпчик.

Розв'язання. Виконуємо віднімання поетапно.

Крок 1. Виконуємо віднімання $1 - 0 = 1$:

```
   1  0  0  1  0  1  0  1
—
            1  1  0  0  0
  ------------------------
                        1
```

Крок 2. Виконуємо віднімання $0 - 0 = 0$:

```
   1  0  0  1  0  1  0  1
—
            1  1  0  0  0
  ------------------------
                     0  1
```

Крок 3. Виконуємо віднімання $1 - 0 = 1$:

```
   1  0  0  1  0  1  0  1
—
            1  1  0  0  0
  ------------------------
                  1  0  1
```

Крок 4. Щоб виконати наступний крок, треба «позичити» 10_2 зліва та виконати $10 - 1 = 1$:

$$
\begin{array}{cccccccc}
 & & & \cdot & \mathbf{10} & & & \\
 & & & & \mathbf{2} & & & \\
 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
- & & & & 1 & 1 & 0 & 0 & 0 \\
\hline
 & & & & & 1 & 1 & 0 & 1
\end{array}
$$

Крок 5. Позичивши на минулому кроці розряд ми знову змушені позичати. Але зліва знову 0, отже і туди треба позичити. Лише після цього можна виконати операцію $1 + 1 - 1 = 1$:

$$
\begin{array}{cccccccc}
 & & \cdot & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{10_2} & & \\
 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
- & & & & 1 & 1 & 0 & 0 & 0 \\
\hline
 & & & & 1 & 1 & 1 & 0 & 1
\end{array}
$$

Крок 6. Переписуємо у результат $1 + 0 = 1$:

$$
\begin{array}{cccccccc}
 & & \cdot & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{10_2} & & \\
 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
- & & & & 1 & 1 & 0 & 0 & 0 \\
\hline
 & & & 1 & 1 & 1 & 1 & 0 & 1
\end{array}
$$

Крок 7. Переписуємо у результат $1+0=1$:

$$
\begin{array}{cccccccc}
 & \cdot & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{10_2} & & \\
 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
- & & & & 1 & 1 & 0 & 0 & 0 \\
\hline
 & & 1 & 1 & 1 & 1 & 1 & 0 & 1
\end{array}
$$

Крок 8. Одиничку ми позичили, тому замість неї переписуємо нуль:

$$
\begin{array}{cccccccc}
 & \cdot & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{10_2} & & \\
 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
- & & & & 1 & 1 & 0 & 0 & 0 \\
\hline
 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1
\end{array}
$$

Таким чином, $10010101_2 - 11000_2 = 1111101_2$.

Перевіримо результат, перевівши числа у десяткову систему числення:

$$10010101_2 = 2^7 + 2^4 + 2^2 + 2^0 = 149$$

$$\overset{4\,3\,2\,1\,0}{11000_2} = 2^4 + 2^3 = 16 + 8 = 24,$$

$$\overset{6\,5\,4\,3\,2\,1\,0}{1111101_2} = 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 1 = 125.$$

Дійсно, $149 - 125 = 24$.

Отже відповідь: 1111101_2.

Переваги та недоліки двійкової системи числення

Двійкова система числення має ряд недоліків. По перше, двійкові числа мають багато розрядів. Наприклад, якщо для запису числа 2017 у десятковій системі числення достатньо лише чотирьох розрядів, то двійковий запис цього числа буде представлений вже 11 розрядами. По друге, оскільки алфавіт цієї системи числення складається лише з нуликів та одиничок, запис числа в двійковій системі досить однорідний і людині сприймати його досить важко.

З іншої сторони, вказані аспекти двійкової системи числення є перевагами, якщо мова йде про обчислювальні технології. Для реалізації автоматичної системи обрахунку чисел у двійковій системі необхідні технічні пристрої лише з двома стійкими станами, прикладами яких є: є струм/немає струму, намагнічений/не намагнічений тощо. Крім того, алфавіт, що складається лише з двох символів, забезпечує на певному рівні надійність та стійкість до перешкод двійкових кодів. Очевидно, що виконання операцій з двійковими числами, яке досить громіздке для людини, для комп'ютера є простішим, ніж з десятковими.

Є ще кілька систем числення, особливості яких забезпечили їх використання в обчислювальних технологіях. Це вісімкова та шістнадцяткова системи числення. Проте очевидно, що ці системи є похідними від двійкової.

1.3.3. Вісімкова система числення

Вісімкова система числення має основу 8, її алфавіт складається із цифр 0, 1, 2, 3, 4, 5, 6 та 7. Оскільки $8 = 2^3$, вісімкова система числення має особливість: кожна вісімкова цифра може бути записана у вигляді *тріади* — послідовності трьох двійкових цифр (див. табл. 1.3.3.1). За

рахунок цієї особливості переведення із вісімкової в двійкову систему числення можна робити простіше, ніж шляхом переведення спочатку у десяткову, а потім із десяткової у двійкову (рис. 1.3.3.1).

Розглянемо на прикладах.

Таблиця 1.3.3.1. Зв'язок вісімкових і двійкових цифр

7_8	6_8	5_8	4_8	3_8	2_8	1_8	0_8
111_2	110_2	101_2	100_2	011_2	010_2	001_2	000_2

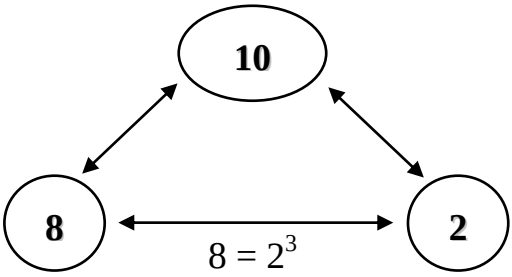


Рисунок 1.3.3.1. Варіанти переведення із вісімкової в двійкову систему числення

Приклад 1.3.3.1. Перевести число 42_8 в двійкову систему числення.

Розв'язання.

Розпишемо цифри числа 42_8 на тріади:

$$42_8 = \underset{4}{\underline{100}}\ \underset{2}{\underline{010}}$$

Отже: $42_8 = 100010_2$.

Приклад 1.3.3.2. Перевести число 2017_8 в двійкову систему числення.

Розв'язання.

Розпишемо цифри числа 2017_8 на тріади:

$$2017_8 = \underset{2}{\underline{010}}\ \underset{0}{\underline{000}}\ \underset{1}{\underline{001}}\ \underset{7}{\underline{111}}$$

Зверніть увагу, що перша цифра: 2, у вигляді тріади представляє собою послідовність 010. В отриманому в результаті двійковому записі числа перший 0 ігноруємо.

Отже: $2017_8 = 10000001111_2$.

Приклад 1.3.3.3. Перевести число 6783_8 в двійкову систему числення.

Розв'язання.

В алфавіті вісімкової системи числення немає цифри 8, отже не існує такого числа, як 6783_8.

Відповідь: задача нерозв'язна, неправильний запис числа.

Для оберненої задачі — переведення із двійкової у вісімкову систему числення — теж можна застосовувати особливість вісімкової системи числення. Для цього достатньо розділити двійкове число на тріади (починаючи з кінця) та перевести їх у вісімкову систему числення.

Приклад 1.3.3.4. Перевести число 110111_2 у вісімкову систему числення.

Розв'язання.

Розіб'ємо число 110111_2 на тріади:

$$110111_2 = \underset{6}{\underline{110}}\ \underset{7}{\underline{111}} = 67_8$$

Відповідь: 67_8.

Приклад 1.3.3.5. Перевести число 1011100011010_2 у вісімкову систему числення.

Розв'язання.

Якщо ми почнемо розкладати на тріади, то на початку залишиться лише одна цифра:

$$1011100011010_2 = \underset{1}{1}\ \underset{3}{011}\ \underset{4}{100}\ \underset{3}{011}\ \underset{2}{010}$$

Щоб утворити тріаду — допишемо на початку два нулі:

$$1011100011010_2 = \underset{1}{001}\ \underset{3}{011}\ \underset{4}{100}\ \underset{3}{011}\ \underset{2}{010} = 13432_8$$

Відповідь: 13432_8.

Арифметичні операції у вісімковій системі числення

Додавання та віднімання у стовпчик у вісімковій системі числення виконується майже так само, як і у десятковій, але треба враховувати особливості цієї системи (див. табл. 1.3.3.2 та 1.3.3.3). Розглянемо на прикладах.

Таблиця 1.3.3.2. Правила додавання у вісімковій системі числення

+	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

Таблиця 1.3.3.3. Правила віднімання у вісімковій системі числення

-	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0							
1	1	0						
2	2	1	0					
3	3	2	1	0				
4	4	3	2	1	0			
5	5	4	3	2	1	0		
6	6	5	4	3	2	1	0	

7	7	6	5	4	3	2	1	0
10	0	7	6	5	4	3	2	1

Приклад 1.3.3.6. Знайти суму 671_8 та 2107_8, виконавши додавання у стовпчик.

Розв'язання. Виконуємо додавання поетапно.

Крок 1. Виконуємо додавання $1+7=10$, тобто в результат записуємо 0 та переносимо один розряд:

$$\begin{array}{cccc} & & \cdot & \\ & 6 & 7 & 1 \\ + & & & \\ 2 & 1 & 0 & 7 \\ \hline & & & 0 \end{array}$$

Крок 2. Виконуємо додавання $\cdot+7+0=1+7+0=10$, тобто знову в результат записуємо 0 та переносимо один розряд:

$$\begin{array}{cccc} & \cdot & \cdot & \\ & 6 & 7 & 1 \\ + & & & \\ 2 & 1 & 0 & 7 \\ \hline & & 0 & 0 \end{array}$$

Крок 3. Виконуємо додавання $\cdot + 6 + 1 = 1 + 6 + 1 = 10$, тобто знову в результат записуємо 0 та переносимо один розряд:

$$\begin{array}{cccc} & \cdot & \cdot & \cdot \\ & & 6 & 7 & 1 \\ + & & & & \\ & 2 & 1 & 0 & 7 \\ \hline & & 0 & 0 & 0 \end{array}$$

Крок 4. Виконуємо додавання $\cdot + 0 + 2 = 1 + 0 + 2 = 3$, тобто знову в результат записуємо 3:

$$\begin{array}{cccc} & \cdot & \cdot & \cdot \\ & & 6 & 7 & 1 \\ + & & & & \\ & 2 & 1 & 0 & 7 \\ \hline & 3 & 0 & 0 & 0 \end{array}$$

Отже: $671_8 + 2107_8 = 3000_8$.

Приклад 1.3.3.7. Знайти різницю 671_8 та 2107_8, виконавши віднімання у стовпчик.

Розв'язання.

Спочатку знайдемо $2107_8 - 671_8$, а потім до результату додамо знак мінус.

Виконуємо віднімання поетапно.

Крок 1. Виконуємо віднімання $7 - 1 = 6$:

$$\begin{array}{cccc} - & 2 & 1 & 0 & 7 \end{array}$$

$$\begin{array}{r} 6\ 7\ 1 \\ \hline 6 \end{array}$$

Крок 2. Щоб виконати наступну операцію необхідно позичити 10_8. Виконуємо віднімання $10 - 7 = 1$:

$$\begin{array}{r} \cdot \quad 10_8 \\ 2\ 1\ 0\ 7 \\ - \\ 6\ 7\ 1 \\ \hline 1\ 6 \end{array}$$

Крок 3. Щоб виконати наступну операцію необхідно позичити 10_8, оскільки 1-ку вже позичили. Виконуємо операцію $7_8 + 1 - 6 = 8 - 6 = 2$:

$$\begin{array}{r} \cdot \quad 7_8\ 10_8 \\ 2\ 1\ 0\ 7 \\ - \\ 6\ 7\ 1 \\ \hline 2\ 1\ 6 \end{array}$$

Крок 4. Виконуємо операцію $2 - 1 - 0 = 1$:

$$\begin{array}{r} \cdot \quad 7_8\ 10_8 \\ 2\ 1\ 0\ 7 \\ - \\ 6\ 7\ 1 \\ \hline \end{array}$$

1 2 1 6

Отже: $671_8 - 2107_8 = -1216_8$.

Застосування вісімкової системи числення

Раніше вісімкова система широко використовувалася в програмуванні для компактного запису двійкових чисел. Проте з часом її повністю витіснила зручніша для цього система: шістнадцяткова. На сьогоднішній день вісімкова система в інформатиці використовується досить рідко, наприклад, у цій системі вказуються права доступу для команди chmod в Unix-подібних операційних системах.

1.3.4. Шістнадцяткова система числення

Алфавіт шістнадцяткової системи числення складається із 16 цифр: перші 10 позначаються звичним чином: 0, 1, 2, 3, 4, 5, 6, 7, 8 та 9; цифри 10, 11, 12, 13, 14 та 15 позначаються латинськими літерами A, B, C, D, E та F відповідно. Для переведення із шістнадцяткової в десяткову і назад використовуються алгоритми, які ми вже розглянули.

Приклад 1.3.4.1. Перевести число 2738 у шістнадцяткову систему числення.

Розв'язання.

Для переведення із десяткової в шістнадцяткову ділимо число 2738 на 16 поки частка не буде меншою за 16:

2738	16	
2	171	16
	11	10

Отримали число, що складається із цифр 10, 11 та 2, тобто $AB2_{16}$.

Отже: $2738 = AB2_{16}$.

Приклад 1.3.4.2. Перевести число $B0F_{16}$ у десяткову систему числення.

Розв'язання.

$$\overset{2\ 1\ 0}{B0F}_{16} = 11 \cdot 16^2 + 0 \cdot 16^1 + 15 \cdot 16^0 = 2831$$

Відповідь: 2831.

За рахунок того, що $16 = 2^4$, шістнадцяткова система числення має особливість, схожу на ту, що була у вісімкової: кожна шістнадцяткова цифра може бути записана як *тетрада:* чотири двійкових цифри (див. дод. А). За рахунок цього можна переводити із шістнадцяткової у двійкову систему числення без проміжного переведення у десяткову. Розглянемо на прикладах (рис. 1.3.4.1).

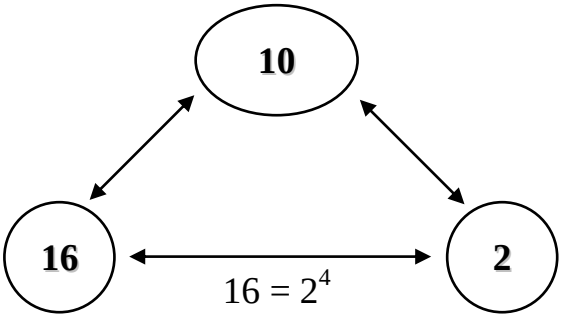


Рисунок 1.3.4.1. Варіанти переведення із шістнадцяткової в двійкову систему числення

Приклад 1.3.4.3. Перевести число $FA01B_{16}$ у двікову систему числення.

Розв'язання.

$$FA01B_{16} = \underset{15}{\underline{1111}}\ \underset{10}{\underline{1010}}\ \underset{0}{\underline{0000}}\ \underset{1}{\underline{0001}}\ \underset{11}{\underline{1011}}$$

Отже: $FA01B_{16} = 1111101000000011011_2$.

Приклад 1.3.4.4. Перевести число 10010110110_2 у шістнадцяткову систему числення.

Розв'язання.

$$10010110110_2 = \underset{4}{\underline{0100}}\ \underset{11}{\underline{1011}}\ \underset{6}{\underline{0110}} = 4B6_{16}$$

Відповідь: $4B6_{16}$.

Цим же способом можна скористатися, щоб перевести число із вісімкової до шістнадцяткової системи числення і навпаки. Для цього достатньо спочатку знайти відповідні тріади чи тетради і звести їх до тетрад чи тріад відповідно. Розглянемо на прикладах.

Приклад 1.3.4.5. Перевести число 567_8 у шістнадцяткову систему числення.

Розв'язання.

Спочатку переведемо число 567_8 у двійкову систему числення:

$$567_8 = \underset{5}{\underline{101}}\ \underset{6}{\underline{110}}\ \underset{7}{\underline{111}} = 101110111_2$$

Тепер 101110111_2 переведемо до шістнадцяткової системи числення:

$$101110111_2 = \underset{1}{\underline{0001}}\ \underset{7}{\underline{0111}}\ \underset{7}{\underline{0111}} = 177_{16}$$

Відповідь: 177_{16}.

Приклад 1.3.4.6. Перевести число $FB0_{16}$ у вісімкову систему числення.

Розв'язання.

Спочатку переведемо число $FB0_{16}$ у двійкову систему числення:

$$FB0_{16} = \underset{15}{\underline{1111}}\ \underset{11}{\underline{1011}}\ \underset{0}{\underline{0000}} = 111110110000_2$$

Тепер 111110110000_2 переведемо до шістнадцяткової системи числення:

$$111110110000_2 = \underset{7}{\underline{111}}\ \underset{6}{\underline{110}}\ \underset{6}{\underline{110}}\ \underset{0}{\underline{000}} = 7660_8$$

Відповідь: 7660_8.

Застосування шістнадцяткової системи числення

Шістнадцяткову систему часто називають також Hex (початкові літери англ. hexadecimal — шістнадцятковий). Для позначення шістнадцяткових констант в комп'ютерних мовах зазвичай використовують префікс «0x»: 42_{16} = 0x42.

Шістнадцяткова система числення широко використовується в комп'ютерних технологіях та в програмуванні. Наприклад, вона застосовується для представлення кольорів при розробці дизайну веб-сайту на мові html, при цьому колір записується як три (у відповідності до трьох кольорив: червогого, зеленого та синього) двозначні шістнадцяткових числа hex від 0 до FF (255) із попереднім знаком # (наприклад рожевий: #FF8080, сірий: #808080, чорний: #000000).

Арифметичні операції у шістнадцятковій системі числення

Додавання та віднімання у стовпчик у шістнадцятковій системі числення виконується майже так само, як і у десятковій. Особливості додавання та віднімання у шістнадцятковій системі числення подані в таблицях 1.3.4.2 та 1.3.4.3.

Таблиця 1.3.4.2. Правила додавання у шістнадцятковій системі числення

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	20	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	20	21	1C	1D	1E

Таблиця 1.3.4.3. Правила віднімання у шістнадцятковій системі числення

-	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0															
1	1	0														
2	2	1	0													
3	3	2	1	0												
4	4	3	2	1	0											
5	5	4	3	2	1	0										
6	6	5	4	3	2	1	C									
7	7	6	5	4	3	2	1	0								
8	8	7	6	5	4	3	2	1	0							
9	9	8	7	6	5	4	3	2	1	0						
A	A	9	8	7	6	5	4	3	2	1	0					
B	B	A	9	8	7	6	5	4	3	2	1	0				
C	C	B	A	9	8	7	6	5	4	3	2	1	0			
D	D	C	B	A	9	8	7	6	5	4	3	2	1	0		

E	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
F	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0

Приклад 1.3.3.6. Знайти суму $B6DA_{16}$ та 42_{16}, виконавши додавання у стовпчик.

Розв'язання. Виконуємо додавання поетапно.

```
   B 6 D A
+
       4 2
-----------
```

Крок 1. Виконуємо додавання $A + 2 = C$:

```
   B 6 D A
+
       4 2
-----------
         C
```

Крок 2. Виконуємо операцію D+4 = 11, отже 1 пишемо та 1 переносимо:

```
     •
   B 6 D A
+
       4 2
-----------
       1 C
```

Крок 3. Виконуємо операцію $\cdot + 6 = 7$:

```
     •
```

```
   B 6 D A
+
       4 2
-------------
   7 1 C
```

Крок 4. Виконуємо операцію $B + 0 = B$:

```
     .
   B 6 D A
+
       4 2
-------------
   B 7 1 C
```

Перевіримо себе.

$B6DA_{16} = 11 \cdot 16^3 + 6 \cdot 16^2 + D \cdot 16^1 + 10 \cdot 16^0 = 46810$

$42_{16} = 4 \cdot 16^1 + 2 \cdot 16^0 = 66$

$B71C_{16} = 11 \cdot 16^3 + 7 \cdot 16^2 + 1 \cdot 16^1 + 12 \cdot 16^0 = 46876$

Дійсно, $46810 + 66 = 46876$.

Отже: $B6DA_{16} + 42_{16} = B71C_{16}$.

Приклад 1.3.3.6. Знайти різницю $B6DA_{16}$та 42_{16}, виконавши віднімання у стовпчик.

Розв'язання.

Виконуємо віднімання поетапно.

```
   B 6 D A
—
       4 2
-------------
```

Крок 1. Виконуємо віднімання $A - 2 = 8$:

```
  B 6 D A
—
      4 2
-------
        8
```

Крок 2. Виконуємо операцію $D - 4 = 9$:

```
  B 6 D A
—
      4 2
-------
      9 8
```

Крок 3. Виконуємо операцію $6 - 0 = 6$:

```
  B 6 D A
—
      4 2
-------
    6 9 8
```

Крок 4. Виконуємо операцію $B - 0 = B$:

.

```
  B 6 D A
—
      4 2
-------
B 6 9 8
```

Перевіримо себе.

$B6DA_{16} = 11 \cdot 16^3 + 6 \cdot 16^2 + D \cdot 16^1 + 10 \cdot 16^0 = 46810$

$42_{16} = 4 \cdot 16^1 + 2 \cdot 16^0 = 66$

$B698_{16} = 11 \cdot 16^3 + 6 \cdot 16^2 + 9 \cdot 16^1 + 8 \cdot 16^0 = 46744$

Дійсно, $46810 - 66 = 46744$.

Отже: $B6DA_{16} - 42_{16} = B698_{16}$.

Питання до розділу 1

1. Передумови виникнення числення
2. Непозиційні та змішані системи числення, їх недоліки та переваги
3. Позиційні системи числення.
4. Особливості подання чисел в позиційних системах числення. Алфавіт та основа позиційної системи числення.
5. Переведення із десяткової до системи числення з основою *x*
6. Переведення із системи числення з основою *x* до десяткової
7. Двійкова та десяткова система числення.
8. Алгоритми переведення із десяткової до двійкової системи числення
9. Двійкова система числення, недоліки та переваги.
10. Арифметичні операції в двійковій системі числення
11. Вісімкова та шістнадцяткова система числення, їх зв’язок із двійковою системою числення.
12. Арифметичні операції у вісімковій та шістнадцятковій системі числення

Завдання до розділу 1

1. Запишіть у дайньоєгипетській системі числення поточний рік.
2. Запишіть у римській системі числення:

   а) поточний рік б) свій рік народження
3. Запишіть у римській системі числення числа:

   а) 123 б) 568 в) 1870

   г) 1849 д) 2026 е) 5555
4. Знайдіть x якщо:

   а) $x_{17} = 2000$ б) $x_{10} = 100_7$

   в) $x_{15} = 345_{17}$ г) $x_8 = 10101_2$

   д) $x_4 = ABCD_{16}$ е) $x_8 = 691$
5. Переведіть у двійкову систему числення:

   а) 133 б) 357 в) 574_8

   г) 59_8 д) $F11_{16}$ е) $88F_{16}$
6. Переведіть у десяткову систему числення:

   а) 101010_2 б) 10110_2 в) 1015_8

   г) 147_8 д) $F21_{16}$ е) $19F_{16}$
7. Додайте у стовпчик:

   а) 11001_2 та 1011101_2 б) 1110101_2 та 1001111_2
8. Відніміть у стовпчик:

   а) 1110101_2 та 1011_2 б) 10101_2 та 1001111_2

9. Переведіть число у двійкову систему числення за допомогою методу підбору:

   а) 100 б) 250 в) 500 г) 2017

10. Переведіть у вісімкову та шістнадцяткову систему числення:

    а) 10100011011_2 б) 10110101111_2

11. Переведіть із вісімкової в шістнадцяткову систему числення 64610_8 та 127760_8

12. Переведіть із шістнадцяткої у вісімкову систему числення $FBA0_{16}$ та $10DD_{16}$

13. В якій системі числення виконується нерівність:

    а) $25_x + 20_x \geq 109_x$ б) $5_x + 5_x \neq 10_x$

    в) $33_x + 44_x \leq 111_x$ г) $24_x + 55_x > 100_x$

    д) $45_x + 42_x \geq 107_x$ е) $45_x + 44_x \leq 111_x$

14. Чи існує система числення x, в якій $2_x + 3_x = 5_x, 3_x \cdot 7_x = 15_x$ та $23_x + 36_x = 59_x$?

15. Для десяткового числа 371 знайти систему числення з основою x, в якій дане число буде представлене тими самими цифрами, але записаними в іншому порядку, тобто $371_{10} = 173_x$.

Приклади теоретичного індивідуального самостійного завдання

1. Підготувати доповідь на тему «Походження українських назв чисел».
2. Підготувати доповідь на тему «Історія виникнення термінів «цифра» та «число».

3. Підготувати доповідь на тему «Історія рахівниці: від абаку до сучасного комп’ютера».

2. Інформація та кодування

Вступ

Із розвитком комп'ютерних технологій поступово змінювалося розуміння таких фундаментальних понять, як інформація, інформаційне середовище, інформаційна сфера тощо. Спробуємо розібратися, що на сьогоднішній день розуміють під цими поняттями. За визначенням тлумачного «Словника української мови» (далі — СУМ), інформація — це відомості про які-небудь події, чиюсь діяльність і т.ін.; повідомлення про щось [39]. В словниках термінів обчислювальних систем можна знайти наступне визначення: сукупність символів, або образів, що несуть змістове навантаження [41]. Академік В.М. Глушков, автор фундаментальним праць у галузі кібернетики, дає наступне визначення інформації: міра неоднорідності розподілу матерії й енергії у просторі й у часі, показник змін, якими супроводжуються всі здійснювані у світі процеси [13].

Отже в загальному розумінні, інформація — це сукупність відомостей (даних), які сприймають із навколишнього середовища (вхідна інформація), видають у навколишнє середовище (вихідна інформація) або зберігають всередині певної системи. Інформація існує у вигляді документів, креслень, рисунків, текстів, звукових та світлових сигналів, електричних та нервових імпульсів тощо. Варто звернути увагу, що значення етимона цієї запозиченої з латини лексеми, īnfōrmātio: виклад, роз'яснення факту, події [26].

2.1. Види та властивості інформації

Український науковець А.О.Білецький у своїй оригінальній лінгвосеміотичній теорії запропонував детальну класифікацію видів інформації, врахувавши різноманітні чинники, умови її творення та різновиди застосування [9] (див. табл. 2.1.1).

Таблиця 2.1.1. Типи інформації за класифікацією А.О.Білецького [9]

Ознака	Тип інформації
Джерело створення	антропогенна — фізіогенна
Форма	кодована (системна) — некодована (одинична)
Зв'язок із ситуацією	ситуаційна — екстраситуаційна
Спосіб оформлення	мовна (вербальна) — немовна (екстравербальна)
Сфера призначення	побутова (узуальна) — спеціальна
Характер змістового навантаження	логічна — естетична

Виходячи з критеріїв визначення інформації, запропонованих А.О.Білецьким, *мовну інформацію* можна визначити як антропогенні (тобто створені людиною) кодовані (тобто закріплені у спеціальній формі) екстраситуаційні (тобто незалежні від конкретної ситуації спілкування) вербальні (тобто представлені у мовній, або

словесній формі) повідомлення про об'єкти та явища дійсності, яким властиві також логічний, естетичний, побутовий, спеціальний та експресивний компоненти [26].

А.О. Білецький також наводить такі функції, виконувані мовною системою як спеціалізованою системою обміну інформацією:

1. *Трансформаційна* (від лат. transformo «перетворюю, змінюю форму») — перетворення інформації;
2. *Консервативна* (від лат. cōnservo «зберігаю») — зберігання інформації;
3. *Транслятивна* (від лат. trānslātio «перенос, переміщення») — передача інформації;
4. *Дименсійна* (від лат. dīmēnsio «вимірювання, розмір») — вимірювання інформації;
   Утилітарна (від лат. ūtilitās «користь, придатність») — використання інформації;
5. *Естетична* (від грецьк. αἰσθητικός «чуттєво сприйманий») — вибір для оформлення та передавання інформації засобів, оптимальних для її сприйняття;

Метасемантична (від грецьк. μετα «зверх, понад» та σημαντικ**ό**ς «означальний») – відображення в оформленні інформації додаткового змісту, який впливає на відчуття людини, стимулює в неї певні реакції на інформацію (пор., наприклад, різні способи оформлення прохання: наказ, уклінне прохання, дружнє спонукання до дії, нейтральний заклик, побажання тощо).

Є й інші способи класифікації інформації. Так, інформацію можна поділити на види за кількома ознаками: за способом сприйняття, за формою подання та за призначенням.

За способом сприйняття людиною інформація поділяється на види залежно від типу рецепторів, що сприймають її:

- візуальна – сприймається органами зору;
- аудіальна – органами слуху;
- тактильна – тактильними рецепторами;
- нюхова – нюховими;
- смакова – смаковими рецепторами.

За формою подання інформація поділяється на такі види:

- текстова, що передається у вигляді символів, призначених позначати лексеми мови;
- числова, у вигляді цифр і знаків, що позначають математичні дії;
- графічна, у вигляді зображень, подій, предметів, графіків;
- звукова, усна або у вигляді запису передачі лексем мови аудіальним шляхом.

За призначенням інформація поділяється на:

- масову (містить тривіальні відомості і оперує набором понять, зрозумілим більшій частині соціуму),
- спеціальну (містить специфічний набір понять, при використанні відбувається передача відомостей, які можуть бути не зрозумілі основній масі соціуму, але необхідні і зрозумілі в рамках вузької соціальної групи, де використовується дана інформація),

- особисту (набір відомостей про яку-небудь особистість, що визначає соціальний стан і типи соціальних взаємодій всередині популяції).

Найбільш важливими *властивостями* інформації є:

- об’єктивність: неупередженість, незалежність інформації від волі та бажань людини;
- повнота: інформацію можна вважати повною, якщо її достатньо для розуміння та прийняття рішення;
- достовірність: інформація достовірна, якщо вона є відображенням дійсності;
- адекватність: визначений рівень достовірності створюваного (за допомогою отриманої інформації) образу реального об’єкта, явища або процесу;
- доступність: міра можливості отримати ту або іншу інформацію;
- актуальність: важливість, істотність у певний момент часу.

2.2. Оброблення інформації

Важливість віднайдення вирізняльних ознак описуваної предметної галузі, вміння оперувати ними для вироблення правильного її сприйняття та використання для розв’язання поставлених завдань, може свідчити такий спогад одного з творців мови програмування Бейсік Джона Кемені. Його родина емігрувала до Америки з Угорщини і в час, про який він згадує, Д.Кемені ще досить посередньо володів англійською мовою. І от йому, сором’язливому 16-річному хлопцеві, довелося 1945 р. складати усний іспит в одній з нью-йоркських шкіл у р-ні Манхеттена. Він так описує цей іспит у своїх спогадах: «Мій словниковий запас

був страшенно обмежений і тому в кожному запитанні я вловлював лише кілька слів. Проте оскільки це був тест, в якому для кожного запитання було запропоновано набір варіантів як відповідь, то й того, що я розумів, було цілком достатньо для побудови моделі та знаходження правильного рішення. Я зумів відкрити цей код і одержав одну з найвищих оцінок у всьому Нью-Йорку» [26].

Складовою частиною інформації є дані, що являють собою зареєстровані сигнали. Під час інформаційного процесу дані перетворюються з одного виду в інший за допомогою методів. Оброблення даних містить в собі множину різних операцій. *Основними операціями* є:

- збір даних — накопичення інформації з метою забезпечення достатньої повноти для прийняття рішення;
- формалізація даних — приведення даних, що надходять із різних джерел до однакової форми;
- фільтрація даних — усунення зайвих даних, які не потрібні для прийняття рішень;
- сортування даних — впорядкування даних за заданою ознакою з метою зручності використання;
- архівація даних — збереження даних у зручній та доступній формі;
- захист даних — комплекс дій, що скеровані на запобігання втрат, відтворення та модифікації даних;
- транспортування даних — прийом та передача даних між віддаленими користувачами інформаційного процесу. Джерело даних прийнято називати сервером, а споживача — клієнтом;

- перетворення даних — перетворення даних з однієї форми в іншу, або з однієї структури в іншу, або зміна типу носія.

Для того, щоб можна було зберігати та працювати з інформацією за допомогою комп'ютера, вона має бути представлена у придатній для оброблення формі. Тут важливим аспектом є форма запису інформації. В цьому сенсі розрізняють:

- *внутрішню формалізацію*, або формалізацію системи мови, створення моделей мовних об'єктів та

- *зовнішню формалізацію*, або формалізацію моделей мовних об'єктів, тобто створення такої форми їхнього представлення, яка була б доступна комп'ютеру та іншим технічним засобам опрацювання інформації.

Внутрішня формалізація полягає у структуруванні інформації, вираженої різними одиницями мови або їхніми сукупностями, виділенні визначальних ознак її будови та правил функціонування. Результатом цього різновиду формалізації мови і стають моделі певних мовних об'єктів або процесів, що з ними відбуваються. Зовнішня формалізація орієнтована на представлення комп'ютеру чи іншому технічному засобу опрацювання інформації уже таких мовних моделей, побудованих за допомогою формально-логічних або математичних методів. Її результат становлять такі моделі мовних об'єктів, які придатні для використання комп'ютера або іншого технічного засобу [26].

Інформація, записана у певній формі, перетворюється на *дані* або *знання*. В основі такого формалізованого запису інформації завжди лежить певна *інформаційна модель* тієї

предметної галузі, якої стосується така інформація. Ступінь деталізації структурування предметної галузі залежить від конкретного завдання, яке ставить перед собою дослідник. Фахівці з інформатики, визначаючи предмет дослідження цієї наукової галузі як «технологію побудови, аналізу та використання людино-комп'ютерного (програмного) знання» [26], основне її завдання і вбачають у розробленні засобів та методів побудови, аналізу та узагальнення інформаційних моделей предметних галузей. *Інформаційна модель* є структурованим і представленим у формалізованому вигляді описом окремого об'єкта або й предметної галузі в цілому, який дозволяє одержати інформацію про них за допомогою комп'ютера. Лінгвістичні моделі становлять різновид інформаційних моделей. Їхня відмінність від інших моделей полягає в універсальному характері мови як спеціалізованої системи обміну інформацією, що обслуговує будь-який різновид розумової діяльності людини.

2.3. Кодування інформації

Для автоматизації роботи з даними, що відносяться до різних типів, важливо уніфікувати їх форму представлення. Для цього використовується *кодування:* представлення даних одного типу через дані іншого типу. Природні мови (українська, англійська тощо) є системами кодування ідей та понять для вираження думок за допомогою мовлення. Іншим прикладом є азбука: система кодування компонентів мови за допомогою графічних символів.

Універсальні засоби кодування успішно втілюються в різноманітних галузях техніки, науки та культури — математичні вирази, телеграфна та морська азбуки, абетка для сліпих тощо (див. рис. 2.3.1).

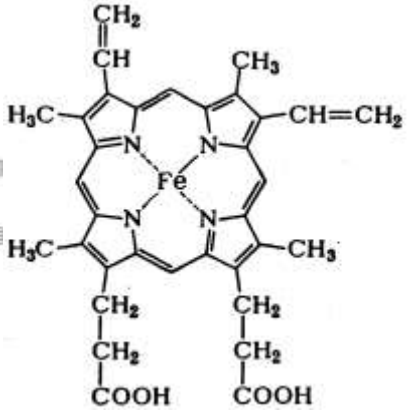
$16 = 10_{16} = 20_8 = 10000_2$

а)

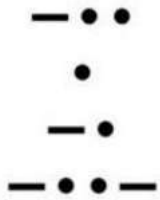
$E = mc^2$

б)

в)



г)



д)

```
while a!=b:
   if a>b:
    a=a-b
    else:
   b = b-a
```

е)

Рисунок 2.3.1. Приклади кодування інформації:
а) кодування формули за допомогою мови математики
б) кодування фізичного закону за допомогою мови фізики
в) кодування мелодії за допомогою музичної нотації
г) кодування хімічного елементу за допомогою структурної формули
д) кодування тексту за допомогою азбуки Морзе
е) кодування алгоритму за допомогою мови програмування

Формально, *кодування* — це запис інформації за допомогою деякої знакової системи — *мови.* Мови бувають *природні* та *формальні*. В основі природніх мов лежать правила, що мають виключення, а в основі формальних — строгі однозначні правила.

Зверніть увагу! Одна і та сама інформація може бути закодована різними способами. Наприклад, число 42 може бути закодоване таким чином: числовий запис: 42; текстовий запис: сорок два; римська система: XLII.

2.3.1. Двійковий код, формула Хартлі

В інформатиці є своя система кодування, що називається двійковим кодом. Ґрунтується вона на представленні даних послідовністю двох знаків: 0 та 1. Ці знаки називають двійковими цифрами або *бітами* (від скорочення англійських слів binary digit). Таким чином біт є одиницею інформації. Слід зауважити, що вся інформація, що зберігається та обробляється засобами обчислювальної техніки, незалежно від її типу (числа, текст, графіка, звук, відео), представлена у двійковому коді. Двійковим кодом можна закодувати інформацію будь-якого типу. Єдиний суттєвий недолік такої системи кодування: вона важко сприймається людиною.

Одним бітом можна виразити два поняття: 0 або 1 (ні або так, хибне або істинне). Якщо кількість бітів збільшити до двох, то тоді можна вже закодувати чотири поняття : 00, 01, 10, 11. Трьома бітами кодують вісім понять: 000, 001, 010, 011, 100, 101, 110, 111. Збільшуючи на одиницю кількість розрядів в системі двійкового кодування, ми збільшуємо в два рази кількість значень, які можуть бути виражені в цій системі кодування, тобто кількість значень вираховується за формулою:

$$N = 2^m$$

де N — кількість незалежних значень, що кодуються, m — розрядність двійкового кодування.

Для підрахунку *кількості інформації*, що міститься в повідомленні довжини n алфавіту потужності m використовується *формула Хартлі:*

$$I = log_2 N = n log_2 m$$

де I — кількість інформації у бітах.

Формула Хартлі була запропонована Ральфом Хартлі у 1928 році як один із наукових підходів оцінки повідомлення.

Проілюструємо суть цієї формули на наступному прикладі. Припустимо, нам треба вгадати число від 1 до 100 задаючи питання типу «Число менше/більше *x*?» та «Це число *x*?», на які можемо отримати відповідь так чи ні. Для мінімізації кількості заданих питань можна використати метод дихотомії — ділення навпіл. Оскільки кожне питання зменшує область пошуку, то для найшвидшого пошуку числа варто, щоб ця область зменшувалась вдвічі. Припустимо, гравець задумав число 42. Варіант гри із мінімальною кількістю питань такий:

1. «Число більше 50?». Відповідь: «Ні» і область пошуку звужується до половини: від 0 до 50 включно.
2. «Число більше 25?». Відповідь: «Так» та область пошуку змінюється на від 26 до 50.
3. «Число більше 38?». Відповідь: «Так» та область пошуку — від 39 до 50.
4. «Число більше 44?». Відповідь: «Ні», область пошуку — від 39 до 44.
5. «Число більше 41?». Відповідь: «Так», область пошуку — від 42 до 44.
6. «Число більше 43?». Відповідь: «Ні», область пошуку — два числа: 42 та 43.
7. «Це число 42?» або «Це число 43?» У першому випадку відповідь: «Так», отже число вгадане. У другому — «Ні», проте число також вгадане методом виключення.

Щоб вгадати число від 1 до 100 нам знадобилося 7 питань. Якби ми питали «Це число 1?», «Це число 2?» нам знадобилося б набагато більше питань. Ділення навпіл — найбільш оптимальний спосіб знаходження числа. Об'єм інформації в одній відповіді так / ні — один біт (0 або 1). Таким чином для відгадування числа від 1 до 100 нам знадобилося сім біт інформації про це число.

Розглянемо застосування формули Хартлі на прикладах.

Приклад 2.3.1.1. В аеропорту стоїть 6 літаків, з них один летить до Києва. Скільки інформації в повідомленні «В Київ летить третій літак»?

Розв'язання.

За формулою Хартлі маємо:

$$I = log_2 6 \approx 3$$

Відповідь: в повідомленні 3 біти.

Приклад 2.3.1.2. Іван живе у дев'ятиповерховому будинку. Скільки інформації в повідомленні «Іван живе на сьомому поверсі»?

Розв'язання.

Скориставшись формулою Хартлі, маємо:

$$I = log_2 9 \approx 4$$

Відповідь: в повідомленні 4 біти.

2.3.2. Одиниці вимірювання інформації

Найменшою одиницею об'єму даних прийнято вважати байт — групу з 8 бітів. Байтом можна закодувати, наприклад, один символ текстової інформації. Наступним одиницями кодування є:

- кілобайт (Кбайт):

  1 Кбайт = 2^{10} байт = 1024 байт;

- мегабайт (Мбайт):

  1 Мбайт = 2^{10} Кбайт = 1024 Кбайт;

- гігабайт (Гбайт):

  1 Гбайт = 2^{10} Мбайт = 1024 Мбайт;

- терабайт (Тбайт):

  1 Тбайт = 2^{10} Гбайт = 1024 Гбайт

- тощо.

Саме в таких одиницях вимірюється *ємність даних в інформатиці*.

Приклад 2.3.2.1. Скільки місця в пам'яті треба виділити для зберігання виразу: «Hello, world!», якщо для зберігання одного символу використовується 1 байт.

Розв'язання.

Спочатку рахуємо всі символи, враховуючі прогалини та розділові знаки, що використані у виразі. Їх 13. Отже кількість необхідних байт: $13 \cdot 1$ байт = 13 байт.

Відповідь: 13 байт.

Приклад 2.3.2.2. Скільки місця треба виділити для зберігання 100 сторінок книги, якщо на кожній сторінці поміщаються 36 рядків по 128 символів в кожній (для зберігання одного символу використовується 1 байт)?

Розв'язання.

На одній сторінці $36 \cdot 128 = 4608$ символів. Отже на 100 сторінках — $4608 \cdot 100 = 460800$ символів. Якщо 1 символ займає 1 байт, то нам треба виділити 460800 байт, що дорівнює 450 Кбайт.

Відповідь: 450 байт.

Приклад 2.3.2.3. Скільки місця в пам'яті треба виділити для зберігання 16-кольорового малюнка розміром 64 на 32 пікселя?

Розв'язання.

Малюнок містить $64 \cdot 32 = 2048$ пікселів. При використанні 16 кольорів на 1 піксель потрібно виділити $log_2 16 = 4$ біти. Отже всього на малюнок треба виділити $2048 \cdot 4 = 8192$ біти, або 1024 байти, або 1 Кбайт.

Відповідь: 1 Кбайт.

Приклад 2.3.2.4. Для зберігання растрового малюнка розміром 32 на 64 пікселя виділили 2 Кбайта пам'яті. Яка максимально можлива кількість кольорів у палітрі?

Розв'язання.

Загальна площа малюнка — $64 \cdot 32 = 2048$ пікселів. В 2 Кбайтах міститься 2048байт. Отже на один піксель виділяється один байт, або 8 біт. Всього варіантів різних кольорів — $2^8 = 256$.

Отже: максимально можлива кількість кольорів в малюнку — 256.

Приклад 2.3.2.5. Обсяг повідомлення, що містить 1024 символів, склав $\frac{1}{512}$ частину Мбайта. Яка потужність алфавіту, за допомогою якого записано повідомлення?

Розв'язання.

Обсяг повідомлення в байтах: $1024 : 512 = 2$ Кбайта, тобто 2048байт. На 1символ доводиться $2048 : 1024 = 2$ байти, тобто 16 біт. Отже потужність алфавіту — 2^{16}, тобто 65536 символи.

Відповідь: 65536 символи.

Приклад 2.3.2.6. В деякій країні автомобільні номери містять 7 символів (використовуються 25 букв і десяткові цифри в будь-якому порядку). Всі символи кодуються однаковою мінімально можливою кількістю біт, а кожен номер — мінімально можливою кількістю байтів. Скільки пам'яті потрібно для зберігання 50 автомобільних номерів

Розв'язання.

Для кодування номерів використовують 35 символів (25 букв + 10цифр). Використаємо формулу Хартлі, щоб

знайти кількість інформації, необхідної для закодування одного символу (округляємо до більшого):

$$I = log_2 35 \approx 6$$

В кожному номері 7 символів, отже, один номер кодується $7 \cdot 6 = 42$ бітами.

За умовою кожен номер кодується мінімально можливою кількістю байтів. Значить, 6 байтами ($42 : 8 = 5.25$, округляємо до більшого).

Отже, щоб зберегти 50 номерів, необхідно $50 \times 6 = 300$ байт.

Відповідь: 300 байт.

2.3.3. Стандарти кодування текстової інформації

Для кодування текстової інформації кожен символ кодується відповідним числом, що визначається домовленністю. Є кілька різних домовленостей про кодування символів, що застосовуються на сьогоднішній день і представляюсь собою *таблиці кодування*.

Таблиця кодування — це таблиця, в якій всім символам комп'ютерного алфавіту поставлено у відповідність порядкові номери (коди).

ASCII та ANSI кодування

Для кодування текстової інформації прийнято міжнародний стандарт ASCII (American Standard Code for Information Interchange) — таблиця кодування, що фіксує лише першу половину кодів, тобто символи з номерами від 0 (00000000) до 127 (0111111). Таким чином кодують літери латиниці (не розширеної), цифри, розділові знаки, дужки і деякі інші службові символи (наприклад, невидимі знаки кінця рядка).

ASCII представляє собою 8-бітну систему кодування для представлення десяткових цифр, латинського та національного алфавітів, розділових знаків і керуючих символів. Спочатку розроблена як 7-бітна, з широким розповсюдженням 8-бітного байта ASCII стала сприйматися як половина 8-бітної. У комп'ютерах зазвичай використовують розширення ASCII з задіяним 8-м бітом і другою половиною кодової таблиці (наприклад ЯКІ-8). Нижню половину кодової таблиці (0-127) займають символи US-ASCII, а верхню (128-255) — різні інші потрібні символи.

За цим стандартом кожен символ займає 1 байт. Всього є 256 символів. Всі символи знаходяться у спеціальній таблиці. Кожен символ має свій внутрішній код, який співпадає з порядковим номером символу в таблиці. Перша половина цієї таблиці (символи з номерами з 0 по 127) стандартна. Вона містить цифри, латинські літери та інші необхідні символи.

ANSI (від American National Standards Institute — Американський національний інститут стандартів) — це таблиця ASCII з розширенням для кодів з номерами від 128 до 255 (залежно від локалізації), яку ще називають сторінкою кодування. В ній останні 128 кодів використано для кодування кирилиці, наприклад: білоруської та української.

У другій половині таблиці (символи з номерами з 128 по 255) знаходяться літери національних алфавітів та додаткові символи. Ця частина таблиці різна для різних країн та різних кодових таблиць, встановлених у операційній системі (див. дод. Б). По таблиці з додатку А видно, що, наприклад, символ f має номер 102, символ Б має номер 129. Символи англійська А та російська А мають однаковий вигляд, але внутрішні коди в них різні —

65 та 128. Символ 0 (нуль) має внутрішній код 48, а символ прогалини — 32.

Зверніть увагу! Великі та маленькі літери мають різні ASCII–коди.

Для виведення на екран символу, якого немає на клавіатурі, потрібно зажати клавішу ALT, набрати ASCII–код символу на цифровій клавіатурі, а потім відпустити клавішу ALT.

Windows-1251

Таблиця кодів символів Windows-1251 є стандартом для кодування літер кирилиці в операційній системі Windows. У ній, наприклад, літері «а» українського алфавіту ставиться у відповідність число 224, літері «і» — число 179, літері «ґ» — число 180 та ін.

Цілих чисел від 0 до 255 вистачає, щоб закодувати символи двох алфавітів — латиниці й кирилиці та деякі інші символи. Але для кодування символів інших алфавітів (грецького чи арабського алфавітів, ієрогліфів тощо) потрібно значно більше значень кодів. Для них розроблено таблицю кодів символів Юнікод (англ. unicode — уніфіковане кодування).

Unicode

Таблиця Юнікод складається з 17 наборів по 65 536 значень кодів у кожному та дає можливість закодувати 1 114 112 різних символів, тобто майже всі символи писемності всіх світових мов. Як і в інших таблицях кодів, у Юнікоді незмінними залишаються перші 128 значень кодів, що відповідають таблиці ASCII. Окремий розділ у таблиці Юнікод містить коди літер кирилиці. Наприклад, літері «а» українського алфавіту ставиться у відповідність код 53 424, літері «і» — код 53 654, літері «ґ» — код 53 905

та ін. Наразі у новітніх операційних системах використовується таблиця кодів Юнікод.

У таблиці Юнікод містяться коди не лише літер та цифр, а й символів, які позначають торговельні марки, грошові одиниці, символи транскрипцій, ідеограми тощо. Наприклад, кодом символу української грошової одиниці гривні є число 8 372, кодом ідеограми чоловік є число 10080, а ідеограми жінка — число 10 081 тощо.

Ідеограма — писемний знак, що передає, на відміну від букви, не звук певної мови, а деяке поняття, ідею.

Юнікод має декілька реалізацій, але найпоширенішими є дві: UTF (Unicode Transformation Format) — Формат Перетворення Юнікоду та UCS (Universal Character Set) — Універсальна Таблиця Символів. Число після UTF визначає кількість бітів, що виділені під один юніт, а число після UCS визначає кількість байтів. Універсальний набір символів задає однозначну відповідність символів кодам — елементам кодового простору, тобто невід'ємним цілим числам. UTF-8 став найпоширенішим для інтернаціональних кодувань.

UTF-8 є системою кодування зі змінною довжиною кодування символів. Це означає, що для кодування символів він використовує від 1 до 4 байт на символ. Так, перший байт UTF-8 можна використовувати для кодування ASCII, що дає повну сумісність з кодами ASCII. Перекодування кодів ASCII у кодах UTF-8 для латинських символів не збільшить розмір даних, бо для цього використовується тільки один байт на символ. Для символів інших мов, де, наприклад, для кодування треба використовувати два байти на символ, це кодування збільшує розмір даних на, приблизно, 50% або більше.

UTF-8 дозволяє працювати в стандартизованому прийнятому на міжнародному рівні багатомовному

середовищі, з порівняно незначним збільшенням обсягу даних. UTF-8 являє собою ідеальний спосіб передачі символів через Інтернет, електронну пошту, чат тощо.

Коди в стандарті Unicode поділені на декілька областей. Область з кодами від U+0000 до U+007F містить символи набору ASCII. Далі розміщені області знаків різних писемностей, знаки пунктуації і технічні символи. Частина кодів зарезервована для використання в майбутньому. Для символів кирилиці виділені коди від U+0400 до U+052F.

Проблеми при кодуванні текстів

Особливості кодування тексту з використанням різних таблиць кодів символів можна побачити також під час перегляду веб-сторінок. Іноді під час відкриття веб-сторінки таблиця кодів символів обирається браузером неправильно. У такому випадку текст веб-сторінки непридатний для розуміння. У кожного веб-браузера є інструменти для вибору таблиці кодів символів користувачами. Для україномовних веб-сторінок найчастіше застосовують таблицю кодів символів Юнікод (UTF-8), але для окремих веб-сторінок може бути використане кодування Кирилиця (Windows-1251) або Кирилиця (KOI8-U).

2.3.4. Кодування графічної інформації

Існує два типи кодування графічної інформації: *растрове* та *векторне*. Растрове кодування кодує зображення у вигляді матриці точок — пікселей. При цьому положення пікселя в матриці визначає позицію пікселя на рисунку, а саме значення пікселя визначає колір рисунку.

Векторне кодування передбачає створення примітивних об'єктів (лінія, крива, крапка, прямокутник, трикутник, коло тощо) з яких складається рисунок. Ці елементи описуються за допомогою математичних формул.

Растрове кодування графічної інформації

Суттєвим недоліком растрового кодування є необхідність *дискретизації зображення*: розбиття на *пікселі* — найдрібніші одиниці цифрового зображення в растровій графіці.

Піксель (від англ. pixel, скорочено від англ. PICture'S ELement — елемент зображення) — неподільний об'єкт прямокутної (зазвичай квадратної) форми, що має певний колір. Будь-яке растрове комп'ютерне зображення складається з пікселів, розташованих по рядках і стовпцях.

Якість зображення залежить від кількості пікселів: чим більше — тим зображення точніше (див. рис. 2.3.4.1). Для оцінки якості зображення використовують *розширення — число пікселей на дюйм* (pixels per inch — ppi).

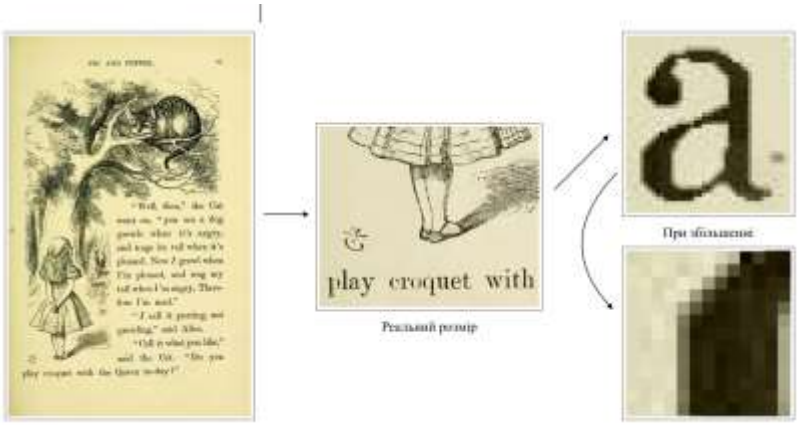


Рисунок 2.3.4.1. При збільшенні растрового зображення можна бачити пікселі, з яких воно складається

Інший показник якості зображення — *кольори*, що застосовуються. Зазвичай кожен колір кодується комбінацією трьох чисел, що визначають насиченість відповідних кольорів. В моделі RGB зазвичай кожен колір: червоний (R, red), зелений (G, green) та синій (B, blue)

кодується числом від 0 до 255 та записується у шістнадцятковому форматі (див. табл. 2.3.4.1), таким чином можна закодувати 256·256·256 = 16777216 кольорів, а для зберігання одного пікселя необхідно 3 байти.

Таблиця 2.3.4.1. Приклади кодування кольорів

Код	***Red***	***Green***	***Blue***	***Колір***
FFFFFF	FF_{16} = 255	FF_{16} = 255	FF_{16} = 255	Білий
000000	00_{16} = 0	00_{16} = 0	00_{16} = 0	Чорний
009933	00_{16} = 0	99_{16} = 147	33_{16} = 51	Темно-зелений
FF0000	FF_{16} = 255	00_{16} = 0	00_{16} = 0	Червоний
99FF66	99_{16} = 147	FF_{16} = 255	66_{16}=102	Фіолетовий

При друці графічні зображення зазвичай кодують за допомогою моделі CMYK (Cyan + Magenta + Yellow + Key Color), тому зображення при друці може трохи відрізнятися від того, що можна було бачити на екрані.

Векторне кодування графічної інформації

Векторні зображення будуються з геометричних фігур: відрізків, ламаних, прямокутників, кіл, еліпсів, дуг тощо. Для кожної фігури рисунку у пам’яті зберігаються (див. табл. 2.3.4.2.):

- її розміри та координати на рисунку,
- колір і стиль границь,
- колір і стиль заливки.

Приклад 2.3.4.1. Зображення з рисунку 2.3.4.2 кодується наступним чином:

```
<svg>
      <rect width="135" height="40"
            x="0" y="10"
            stroke-width="1"
            stroke="rgb(0,0,0)"
            fill="rgb(0,200,255)"/>

      <rect width="135" height="40"
            x="0" y="50"
            stroke-width="1"
            stroke="rgb(0,0,0)"
            fill="rgb(255,255,0)"/>

      <line x1="0" y1="0" x2="0" y2="150"
            stroke-width="15"
            stroke="rgb(0,0,0)"/>
</svg>
```

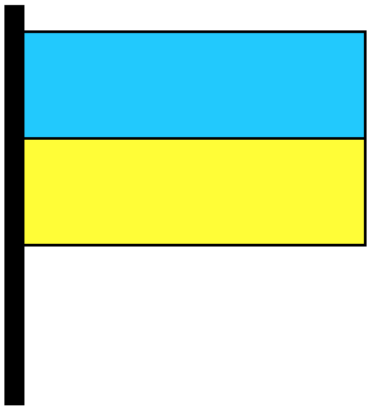


Рисунок 2.3.4.2. Приклад векторного зображення

Ми бачимо, що svg-рисунок складається із трьох фігур: двох прямокутників (rectangle) та однієї лінії (line). Перша фігура має ширину (width) 135 та висоту (height) 30 і

знаходиться за координатами x=0 та y=10. Колір заливки (fill) має rgb-формат: R = 0, G = 200 та B = 255, що визначає відтінок блакитного кольору. Друга фігура має такі самі висоту та ширину і знаходиться за координатами x=0 та y=50 (трохи нижче). Колір заливки має rgb-формат: R = 255, G = 255 та B = 0, що визначає насичений жовтий колір. Третя фігура — це лінія висотою 150 пікселів, що має колір rgb(0,0,0), тобто чорний.

2.3.5. Кодування звуку та відео

Звук є *хвилею з частотою та амплітудою*, які постійно змінюються. Чим більша амплітуда, тим звук голосніший, при рості частоти — вищий тон. Звукові хвилі в нашому довкіллі абсолютно різноманітні. Складні сигнали досить точно можна представити у вигляді простих синусоїдальних коливань. Причому, кожна синусоїда може бути точно задана певним набором чисел — *амплітуда, фаза, частота* — які розглядають код звуку у будь-який момент часу [24].

Звук є неперервним, проте звукова інформація у пам'яті комп'ютера зберігається у дискретному вигляді: послідовності значень сигналу (глибини сигналу) в моменти 0, T, $2T$ і т.д., де T — проміжок часу в секундах.

Якість звуку визначається *частотою дискретизації*. *Частота дискретизації* вимірюється в Гц та визначається наступним чином:

$$f = \frac{1}{T}$$

Зверніть увагу! Звичайна людина здатна сприймати звуки на частоті від 16 Гц до 20 кГц.

Приклад 2.3.5.1. Частота дискретизації аудіозапису становить 22 кГц. Встановіть значення T.

Розв'язання.

$$22\ кГц = 22 \cdot 10^3\ Гц = 22000$$

$$T = \frac{1}{f} = \frac{1}{22000} \approx 0.00005$$

Отже: $T = 0.00005$ *с.*

Інший показник якості — «глибина» сигналу (розрядність звукової карти) — скільки біт виділяється на збереження сигналу (8 біт — 256 рівнів, 16 біт — 65536 рівнів тощо).

Під час оцифрування аудіозапису частина інформації втрачається при дисретизації, частина — через недостатню глибину сигналу (див. рис. 2.4.3.1).

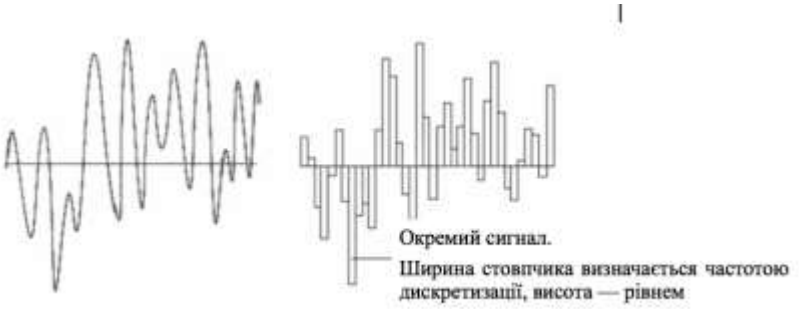


Рисунок 2.3.4.1. Кодування звукової інформації

Оскільки відео являє собою комбінацію зображень та звуку, то і кодування передбачає кодування множини зображень (при цьому на якість відео впливає кількість зображень в секунду) та аудіо.

Питання до розділу

1. Підходи до визначення інформації
2. Типи інформації за класифікацією Білецького А.О.
3. Функції інформації
4. Підходи до класифікації інформації
5. Властивості інформації
6. Оброблення інформації
7. Формалізація інформації.
8. Інформаційна модель
9. Кодування інформації
10. Двійковий код
11. Вимірювання кількості інформації. Формула Хартлі
12. Одиниці вимірювання інформації
13. Особливості кодування текстової інформації
14. ASCII та ANSI кодування
15. Кодування Windows-1251
16. Кодування Unicode
17. Проблеми при кодуванні текстів
18. Кодування графічної інформації: векторне та растрове
19. Кодування звуку та відео

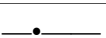
Завдання до розділу

1. Закодуйте своє ім’я за допомогою азбуки Морзе, скориставшись табл. 2.1.

Таблиця 2.1. Азбука Морзе

А	•—	І	••	Т	—
Б	—•••	Ї	•———•	У	••—
В	•——	Й	•———	Ф	••-•

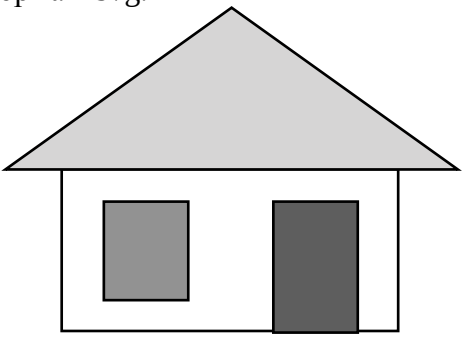
Г	⋅⋅⋅⋅	К	—⋅—	Х	————
Ґ	———⋅	Л		Ц	—⋅—⋅
Д	—⋅⋅	М	———	Ч	————⋅
Е		Н	—⋅	Ш	———⋅—
Є		О	————	Щ	———⋅———

Ж		П		Ь	
З		Р		Ю	
И		С		Я	

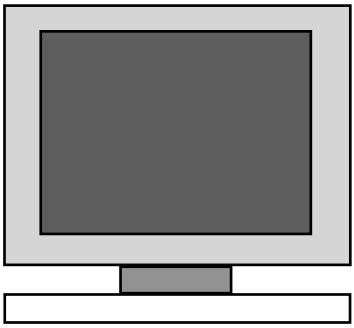
2. Закодуйте повідомлення з використанням таблиці кодів символів ASCII (скористайтеся додатком Б):
   а) Linguistic б) ваше прізвище та ім’я.
3. Визначте колір по коду:
   а) CC66FF б) FFFF00

в) 00FFFF г) FFCCFF

4. Опишіть векторні зображення, подані на рис. 2.1.а та 2.1.б у форматі svg.



а)



б)

Рисунок 2.1. Приклади зображень

5. Порівняйте (поставте знак <, > або =):

   а) 3 байти та 24 біти
   б) 1000 Кбайт та 1 Мбайт
   в) 250 байт та 0,25 Кбайт
   г) 9546 біти та 1 Кбайт

2. У гральній колоді 36 карт. Скільки інформації в реченні: «Фокусник витягнув бубнового туза»?

3. Для гри у шахи використовують різні шахові фігури. Скільки байт інформації несе в собі повідомлення про те, що чорний король стоїть на клітині 4d?
4. Скільки місця потрібно виділити для кодування виразу «Я навчаюсь на прикладній лінгвістиці»?
5. Знаємо, що речення: «Володя живе на другому поверсі» кодується 7 бітами. Знайдіть усі варіанти кількості поверхів, враховуючи, що округлюємо в більшу частину.
6. Алфавіт мови деякого племені складається із звуків «А», «О», та «И». Скільки інформації у слові «Оооаи»?
7. Швидкість передачі даних через ADSL-з'єднання дорівнює 256 000 біт / с. Передача файлу через це з'єднання зайняла 2 хвилини. Визначте розмір файлу в кілобайтах.
8. Андрій зберігає телефонну книгу в блокноті в такому форматі: ім’я (6 символів), прогалина (1 символ), телефон (9 символів) та перехід на новий рядок (1 символ). Вчора Андрій додав до блокноту запис про ще одного друга і розмір файлу перевищив 1 Кбайт. Скільки записів у блокноті Андрія?
9. Катерина хоче завантажити усі фото зі свого instagram на флешку розміром 2 Гбайти. Розмір кожного фото - 1080 х 1080 px, фото 8-бітні. Скільки фотографій зможе завантажити Катя на флешку і чи може завантажити їх всі, якщо в неї їх 2000?
10. Максимальна кількість символів, які можна переслати в одному SMS-повідомленні – 40. На скільки частин потрібно розділити повідомлення в 0,5 Кбайт записане в блокноті у форматі ЮНІКОД?
11. Скільки місця потрібно виділити для зберігання 5 сторінок реферату, якщо на кожній сторінці 8 рядків по 24 символи?

3. Програмне забезпечення

Вступ

В основу роботи комп'ютерів покладено програмний принцип керування, який полягає в тому, що комп'ютер виконує дії за заздалегідь заданою *програмою*. Цей принцип забезпечує універсальність використання комп'ютера: у певний момент часу розв'язується задача відповідно до вибраної програми. Після її завершення у пам'ять завантажується інша програма і т.д.

Програма — це запис алгоритму розв'язання задачі у вигляді послідовності команд або операторів мовою, яку розуміє комп'ютер.

Для нормального розв'язання задач на комп'ютері потрібно, щоб програма була налагоджена, не потребувала дороблень і мала відповідну документацію. Тому стосовно роботи на комп'ютері часто використовують термін програмне забезпечення (software), під яким розуміють сукупність програм, процедур і правил, а також документації, що стосуються функціонування системи оброблення даних.

Програмне та апаратне забезпечення у комп'ютері працюють у нерозривному зв'язку та взаємодії. Склад програмного забезпечення обчислювальної системи називається *програмною конфігурацією*. Між програмами існує взаємозв'язок, тобто багато програм працюють, базуючись на програмах нижчого рівня. *Міжпрограмний інтерфейс* — це розподіл програмного забезпечення на декілька пов'язаних між собою рівнів. Рівні програмного забезпечення являють собою піраміду, де кожен вищій рівень базується на програмному забезпеченні попередніх

рівнів. Схематично структура програмного забезпечення наведена на рис. 3.1.

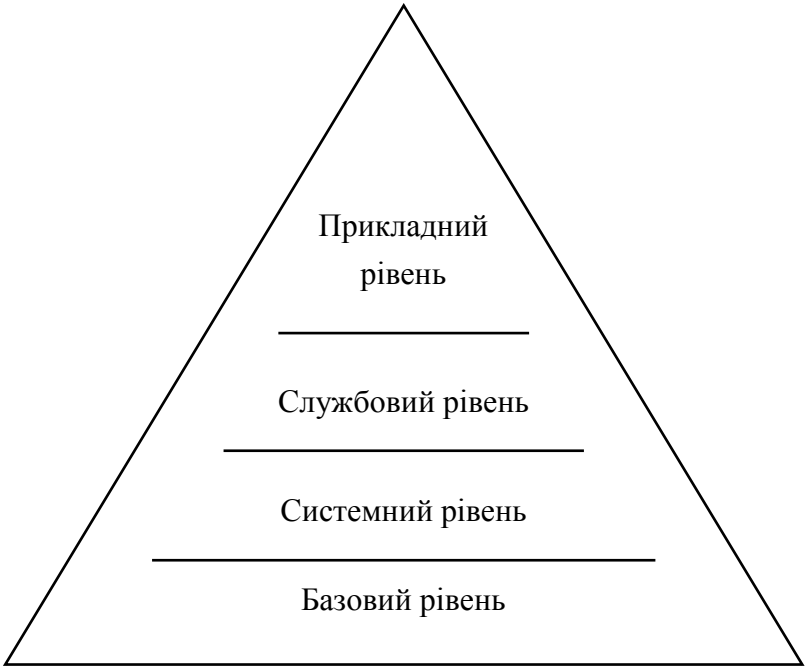


Рисунок 3.1. Структура програмного забезпечення

3.1. Базовий рівень

Базовий рівень є найнижчим рівнем програмного забезпечення. Відповідає за взаємодію з базовими апаратними засобами. Базове програмне забезпечення міститься у складі базового апаратного забезпечення і зберігається у спеціальних мікросхемах постійного запам'ятовуючого пристрою (ПЗП) та напівпостійного програмованого запам’ятовуючого пристрою (НПЗП), утворюючи базову систему введення-виведення BIOS. Програми та дані записуються у ПЗП на етапі виробництва і не можуть бути змінені в процесі експлуатації.

3.2. Системний рівень

Системний рівень — є перехідним. Програми цього рівня забезпечують взаємодію інших програм комп'ютера з програмами базового рівня і безпосередньо з апаратним забезпеченням. Від програм цього рівня залежать експлуатаційні показники всієї обчислювальної системи. При під'єднанні до комп'ютера нового обладнання, на системному рівні повинна бути встановлена програма, що забезпечує для решти програм взаємозв'язок із цим пристроєм. Конкретні програми, призначені для взаємодії з конкретними пристроями, називають драйверами.

Інший клас програм системного рівня відповідає за взаємодію з користувачем. Завдяки йому є можливість вводити дані у обчислювальну систему, керувати її роботою й отримувати результат у зручній формі. Це засоби забезпечення користувацького інтерфейсу, від них залежить зручність та продуктивність роботи з комп'ютером.

Сукупність програмного забезпечення системного рівня утворює ядро операційної системи комп'ютера. Наявність ядра операційної системи — є першою умовою для можливості практичної роботи користувача з обчислювальною системою. Ядро операційної системи виконує такі функції: керування пам'яттю, процесами введення-виведення, файловою системою, організація взаємодії та диспетчеризація процесів, облік використання ресурсів, оброблення команд і т.д.

3.2.1. Операційна система

Операційна система (далі — ОС, англ. *operating system, OS*) — це базовий комплекс програм, що виконує управління апаратною складовою комп'ютера або віртуальної машини; забезпечує керування

обчислювальним процесом і організовує взаємодію з користувачем.

Операційна система звичайно складається з ядра операційної системи та базового набору прикладних програм.

Юнікс-подібні ОС

До юнікс-подібних ОС відноситься велика кількість операційних систем, котрі можна умовно поділити на три категорії — System V, BSD та Лінукс. Сама назва «Юнікс» є торговою маркою, що належить «The Open Group», котра власне й ліцензує кожну конкретну ОС на предмет того, чи відповідає вона стандарту. Тому через ліцензійні чи інші неузгодження деякі ОС, котрі фактично є Юнікс-подібними, не визнані такими офіційно.

Системи Юнікс запускаються на великій кількості процесорних архітектур. Вони широко використовуються як серверні системи у бізнесі, як стільничні системи у академічному та інженерному середовищі. Тут популярні вільні варіанти Юнікс, такі як Лінукс та БСД-системи. Окрім того, деякі з них останнім часом набувають широкого поширення в корпоративному середовищі, особливо це стосується орієнтованих на кінцевого користувача дистрибутивів Лінукс, в першу чергу Ubuntu, Mandriva, Red Hat Enterprise Linux та Suse. Лінукс також є популярною системою на стільницях розробників, системних адміністраторів та інших IT-спеціалістів.

Microsoft Windows

Спочатку родина ОС Microsoft Windows проектувалась як графічна надбудова над старими середовищами DOS. Сучасні версії розроблені на базі нового ядра (англ. *NT - New Technology*, Нова технологія), яке з'явилось в OS/2, запозичене з VMS. Windows запускається на 32- та 64-бітних процесорах Інтел та AMD; попередні версії також

могли запускатись на процесорах DEC Alpha, MIPS, Fairchild (пізніше Intergraph) Clipper та PowerPC. Проводились роботи на портування її на архітектуру SPARC.

Станом на 2006 рік Windows утримувала монопольне становище (близько 94 %) світового ринку настільних систем, дещо втрачаючи позиції через зростання популярності систем з відкритими джерельними кодами. Вона також використовується на малих та середніх серверах мереж та баз даних. Останнім часом Microsoft проводить ряд маркетингових досліджень, котрі мають на меті показати привабливість родини Windows на ринку корпоративних систем, в тому числі надає навчальні ліцензії своїх продуктів та розробляє безкоштовні навчальні матеріали для студентів та школярів.

Mac OS X

Mac OS X — це ряд графічних ОС, що розроблюються, реалізуються та підтримуються компанією Apple. Mac OS X — це наступниця оригінальної MacOS, що її розробляла Apple з 1984 року. На відміну від попередниці, Mac OS X є Юнікс-системою, що розроблена на основі NEXTSTEP, близької до гілки BSD.

3.2.2. Інтерфейс ядра операційної системи

Інтерфейс командного рядка (англ. *command-line interface*, CLI) — різновид текстового інтерфейсу користувача й комп'ютера, в якому інструкції комп'ютеру можна дати тільки введенням із клавіатури текстових рядків (команд). Також відомий під назвою *консоль*. Інтерфейс командного рядка може бути протиставлений системам управління програмою на основі меню чи різних реалізацій графічного інтерфейсу. Формат виводу інформації в інтерфейсі командного рядка не регламентується; звичайно це

простий текстовий вивід, але може бути й графічним, звуковим виводом тощо.

Інтерфейс командного рядка ОС Windows

Щоб викликати стандартний командний рядок в Windows необхідно в нижньому рядку вводу меню Пуск ввести команду `cmd` та натиснути Enter.

Найбільш уживані команди ОС Windows наведені в таблиці 3.2.2.1.

Таблиця 3.2.2.1. Найбільш уживані команди ОС Windows

Команда	**Опис, Формат**	**Приклади**
`help`	Виклик довідки. Формат: `help command_name`	`help cd`
`C:`	Перейти до вказаного диску	`C:` `D:`

Продовження таблиці 3.2.2.1

`cd`	Зміна поточної папки. Ключі:
	`\`: до кореневої папки
	`..`: до батьківської папки
		`cd Films\Comedy`
		`cd \`
		`cd ..`

`dir`	Показати вміст папки. Ключі:
	`/p`: виводить список посторінково, показує список порціями, а не весь зразу.
	`/w`: виводить список без додаткової інформації.
	`/s`: виводить список файлів і папок в усіх підпапках даної папки
	`/AH`: виведе список всіх прихованих файлів
		`dir /p`
		`dir /w`
		`dir /s`
		`dir /AH`

`type`	Вивести вміст файлу	`type data.txt`
`cls`	Очистити екран консолі	
`copy`	Копіювати файл (спочатку вказуємо що, потім вказуємо куди).	`copy d.txt /Dir`
`move`	Перемістити файл	`move d.txt /Dir`
`md`	Створити папку (від «make directory»)	`md New_folder`
`rename`	перейменувати файл або папку (спочатку вказуємо стару назву, потім нову).	`rename d.txt c.txt`
`rd`	Видалити папку (від «remove directory»).	`rd /Dir`

Продовження таблиці 3.2.2.1

`del`	Видалити файл або файли. Видалити можна як вказаний файл, так і видаляти файли по певних критеріях, використовуючи «*» і «?».	`del  data.txt` `del /Dir/*.txt`

Інтерфейс командного рядка Unix-подібних систем

Особливості команд Unix-подібних систем.

1. Регістр має значення.
2. В маршрутних іменах директорії відокремлюються символом «/» (а не «\», як у Windows).
3. Ім'я, що починається з «/» — повне маршрутне ім'я, наприклад: «`/Users/Guest/Pictures`».
4. Ім'я файлу може складатися із будь-яких символів Unicode, максимальна довжина імені — 256 символів, довжина повного маршрутного імені — не більше 32000 символів.
5. Для задання шаблону імені використовуються символи «*» (довільна послідовність символів) та «?» (один довільний символ). Ім'я, що складається з одної точки «.» позначає поточну директорію, ім'я «..» — батьківський каталог.
6. Щоб запустити програму достатньо набрати її ім'я (якщо потрібно — то і аргументи, розділяючи їх прогалинами або табуляторами). Ключі команди виділяються знаком «-«:

<command> -<key$_1$> -<key$_2$> … <argument$_1$> <argument$_2$>…

7. Якщо командний рядок починається знаком «&», то команда запуститься у фоні.
8. Команда має три напрямки дії:

1. стандартне введення:

   `command > filename` (створення файлу filename та запис в нього)

   `command >> filename` (запис в кінець файлу filename)

2. стандартне виведення: `command < filename;`
3. стандартний протокол: `command 2> filename.`

Наприклад, команда запису в файл вмісту поточної директорії: `ls > infdir.`

Найбільш уживані команди UNIX-подібних систем наведені в табл. 3.2.2.2.

Таблиця 3.2.2.2. Найбільш уживані команди Unix-подібних систем

Команда	**Опис, Формат**	**Приклади**
`pwd`	Отримати ім'я поточної директорії	`pwd`
`cd`	Змінити поточну директорію. Формат: `cd [директорія]`	`cd /Users/Guest` `cd ..`
`ls`	Роздрукувати інформацію про файли та директорії. Формат: `ls [ключи] [имена]`	`ls -al`
`cat`	Злити файли в один. Формат: `cat файл1 [файл2...]`	`cat d.txt > c.txt`

Продовження таблиці 3.2.2.2

`cp`	Копіювати файли. Формат: `cp файл1 файл2`	`cp d.txt c.txt`

more, pg	Продивитися вміст файлу. Команди під час перегляду: q — завершити перегляд, ПРОГАЛИНА — наступна сторінка, ЕНТЕР — наступний рядок, b — попередня сторінка, / — пошук, h — список команд	more data.txt pg data.txt less data.txt
mv	Перемістити чи переіменувати файли	mv d.txt c.txt mv d.txt Data
ln	Створити посилання на файл	ln d.txt d.lnk
rm	Видалити файли. Ключі: -i: просити підтвердження на видалення -r: рекурсивно видалити із підкаталогами -f: не просити підтвердження на видалення	rm -i data.txt rm -f Pict
rmdir	Видалити директорію	
mkdir	Створити директорію	mkdir NewDir
chmod	Змінити права доступу до файлу. Ключі: u+x: доступ на відкриття файлу a+w: доступ на редагування файлу	chmod u+x data.txt
echo	Вивести аргументи командного рядка	echo «hello»

Продовження таблиці 3.2.2.2

`ps`	Роздрукувати інформацію про процеси	`ps`
`kill`	Завершити процес (по його коду)	`kill -9 1078`

3.3. Службовий рівень

Програми службового рівня взаємодіють як із програмами базового рівня, так і з програмами системного рівня. Призначення службових програм (утиліт) полягає у автоматизації робіт по перевірці та налаштуванню комп'ютерної системи, а також для покращення функцій системних програм. Деякі службові програми (програми обслуговування) відразу додають до складу операційної системи, доповнюючи її ядро, але більшість є зовнішніми програмами і розширюють функції операційної системи. Тобто, у розробці службових програм відслідковуються два напрямки: інтеграція з операційною системою та автономне функціонування.

Класифікація службових програмних засобів

1. *Диспетчери файлів (файлові менеджери).* За їх допомогою виконується більшість операцій по обслуговуванню файлової структури копіювання, переміщення, перейменування файлів, створення каталогів (папок), знищення об'єктів, пошук файлів та навігація у файловій структурі. Базові програмні засоби містяться у складі програм системного рівня і встановлюються разом з операційною системою.
2. *Засоби стиснення даних (архіватори).*Призначені для створення архівів. Архівні файли мають підвищену щільність запису інформації і відповідно, ефективніше використовують носії інформації.

3. *Засоби діагностики.* Призначені для автоматизації процесів діагностування програмного та апаратного забезпечення. Їх використовують для виправлення помилок і для оптимізації роботи комп'ютерної системи.
4. *Програми інсталяції (встановлення).* Призначені для контролю за додаванням у поточну програмну конфігурацію нового програмного забезпечення. Вони слідкують за станом і зміною оточуючого програмного середовища, відслідковують та протоколюють утворення нових зв'язків, загублені під час знищення певних програм. Прості засоби управління встановленням та знищенням програм містяться у складі операційної системи, але можуть використовуватись і додаткові службові програми.
5. *Засоби комунікації.* Дозволяють встановлювати з'єднання з віддаленими комп'ютерами, передають повідомлення електронної пошти, пересилають факсимільні повідомлення тощо.
6. *Засоби перегляду та відтворення.* Переважно для роботи з файлами, їх необхідно завантажити у «рідну» прикладну систему і внести необхідні виправлення. Але, якщо редагування не потрібно, існують універсальні засоби для перегляду (у випадку тексту) або відтворення (у випадку звука або відео) даних.
7. *Засоби комп'ютерної безпеки.* До них відносяться засоби пасивного та активного захисту даних від пошкодження, несанкціонованого доступу, перегляду та зміни даних. Засоби пасивного захисту - це службові програми, призначені для резервного копіювання. Засоби активного захисту застосовують антивірусне програмне забезпечення. Для захисту даних від несанкціонованого доступу, їх перегляду та зміни

використовують спеціальні системи, базовані на криптографії.

3.3. Прикладний рівень

Програмне забезпечення цього рівня являє собою комплекс прикладних програм, за допомогою яких виконуються конкретні завдання (від виробничих до творчих, розважальних та навчальних). Між прикладним та системним програмним забезпеченням існує тісний взаємозв'язок. Універсальність обчислювальної системи, доступність прикладних програм і широта функціональних можливостей комп'ютера безпосередньо залежать від типу наявної операційної системи, системних засобів, що містяться у її ядрі й взаємодії комплексу людина-програма-обладнання [35].

Класифікація прикладного програмного забезпечення

1. *Текстові редактори.* Основними функціями є введення та редагування текстових даних. Для операцій вводу, виводу та збереження даних текстові редактори використовують системне програмне забезпечення. З цього класу прикладних програм починають знайомство з програмним забезпеченням і на ньому набувають перші навички роботи з комп'ютером.
2. *Текстові процесори.* Дозволяють форматувати, тобто оформлювати текст. Основними засобами текстових процесорів є засоби забезпечення взаємодії тексту, графіки, таблиць та інших об'єктів, що складають готовий документ, а також засоби автоматизації процесів редагування та форматування. Сучасний стиль роботи з документами має два підходи: робота з паперовими документами та робота з електронними документами. Прийоми та методи форматування таких документів різняться між собою, але текстові

процесори спроможні ефективно опрацьовувати обидва види документів.

3. *Графічні редактори.* Широкий клас програм, що призначені для створення та оброблення графічних зображень. Розрізняють три категорії.
    1. У *растрових редакторах* графічний об'єкт представлений у вигляді комбінації точок (растрів), що мають свою яскравість та колір. Такий підхід ефективний, коли графічне зображення має багато кольорів і інформація про колір елементів набагато важливіша за інформацію про їх форму. Це характерно для фотографічних та поліграфічних зображень. Застосовують для оброблення зображень, створення фотоефектів і художніх композицій.
    2. *Векторні редактори* відрізняються способом представлення даних про зображення. Об'єктом є не точка, а лінія. Кожна лінія розглядається, як математична крива III порядку і представлена формулою. Таке представлення компактніше за растрове, дані займають менше місця, побудова об'єкта супроводжується підрахунком параметрів кривої у координати екранного зображення, і відповідно, потребує більш продуктивних обчислювальних систем. Широко застосовуються у рекламі, оформленні обкладинок поліграфічних видань.
    3. *Редактори тривимірної графіки (3-D редактори)* використовують для створення об'ємних композицій. Мають дві особливості: дозволяють керувати властивостями поверхні в залежності від властивостей освітлення, а також дозволяють створювати об'ємну анімацію.

4. *Системи управління базами даних (СУБД)*. Базою даних називають великі масиви даних організовані у табличні структури. Основні функції СУБД: створення структури бази даних; наявність засобів її заповнення або імпорту даних із таблиць іншої бази; можливість доступу до даних, наявність засобів пошуку й фільтрації. У зв'язку з поширенням мережевих технологій, від сучасних СУБД вимагається можливість роботи з віддаленими й розподіленими ресурсами, що знаходяться на серверах Інтернету.
5. *Електронні таблиці* надають комплексні засоби для збереження різних типів даних та їх оброблення. Основний акцент зміщений на перетворення даних, наданий широкий спектр методів для роботи з числовими даними. Основна особливість електронних таблиць полягає у автоматичній зміні вмісту всіх комірок при зміні відношень, заданих математичними або логічними формулами. Широке застосування знаходять у бухгалтерському обліку, аналізі фінансових та торгівельних ринків, засобах оброблення результатів експериментів, тобто у автоматизації регулярно повторюваних обчислень великих об'ємів числових даних.
6. *Системи автоматизованого проектування* (САП-системи або САПР-системи) призначені для автоматизації проектно-конструкторських робіт. Застосовуються у машинобудуванні, приладобудуванні, архітектурі. Окрім графічних робіт дозволяють проводити прості розрахунки та вибір готових конструктивних елементів з існуючої бази даних. Особливість САП-систем полягає у автоматичному забезпеченні на всіх етапах проектування технічних умов, норм та правил. САП є необхідним компонентом для гнучких виробничих

систем (ГВС) та автоматизованих систем управління технологічними процесами (АСУ ТП).

7. *Настільні видавничі системи* автоматизують процес верстання поліграфічних видань. Займає проміжний стан між текстовими процесами та САПР. Видавничі системи відрізняються розширеними засобами управління взаємодії тексту з параметрами сторінки і графічними об'єктами, але мають слабші можливості по автоматизації вводу та редагування тексту. Їх доцільно застосовувати до документів, що попередньо оброблені у текстових процесорах та графічних редакторах.
8. *Редактори HTML (Web-редактори).* Особливий клас редакторів, що об'єднують у собі можливості текстових та графічних редакторів. Призначені для створення і редагування Web-сторінок Інтернету. Програми цього класу можна також використовувати при підготовці електронних документів та мультимедійних видань.
9. *Браузери (засоби перегляду Web-документів).* Програмні засоби призначені для перегляду електронних документів, створених у форматі HTML. Відтворюють окрім тексту та графіки, також музику, людську мову, радіопередачі, відеоконференції і дозволяють працювати з електронною поштою.
10. *Системи автоматизованого перекладу.* Розрізняють електронні словники та програми перекладу мови. Електронні словники — це засоби для перекладу окремих слів у документі. Потрібні для професійних перекладачів, які самостійно перекладають текст. Програми автоматичного перекладу отримують текст на одній мові і видають текст на іншій, тобто автоматизують переклад. При автоматизованому перекладі неможливо отримати якісний вихідний

текст, оскільки все зводиться до перекладу окремих лексичних одиниць. Але, для технічного тексту, цей бар'єр знижений. Програми автоматичного перекладу доцільно використовувати: при абсолютному незнанні іноземної мови; при необхідності швидкого ознайомлення з документом; для перекладу на іноземну мову; для створення чернетки, що потім буде підправлено повноцінним перекладом.

11. *Інтегровані системи діловодства.* Засоби для автоматизації робочого місця керівника. Зокрема, це функції створення, редагування і форматування документів, централізація функцій електронної пошти, факсимільного та телефонного зв'язку, диспетчеризація та моніторинг документообігу підприємства, координація дій підрозділів, оптимізація адміністративно-господарської діяльності й поставка оперативної та довідкової інформації.
12. *Бухгалтерські системи.* Містять у собі функції текстових, табличних редакторів та СУБД. Призначені для автоматизації підготовки початкових бухгалтерських документів підприємства та їх обліку, регулярних звітів по підсумках виробничої, господарської та фінансової діяльності у формі прийнятної для податкових органів, позабюджетних фондів та органів статистичного обліку.
13. *Фінансові аналітичні системи.* Використовують у банківських та біржових структурах. Дозволяють контролювати та прогнозувати ситуацію на фінансових, торгівельних та ринках сировини, виконувати аналіз поточних подій, готувати звіти.
14. *Експертні системи.* Призначені для аналізу даних, що містяться у базах знань і видачі результатів, при запиті користувача. Такі системи використовуються, коли для

прийняття рішення потрібні широкі спеціальні знання. Використовуються у медицині, фармакології, хімії, юриспруденції. З використанням експертних систем пов'язана область науки, що зветься інженерією знань. Інженери знань - це фахівці, які є проміжною ланкою між розробниками експертних систем (програмістами) та провідними фахівцями у конкретних областях науки й техніки (експертами).

15. *Геоінформаційні системи (ГІС).* Призначені для автоматизації картографічних та геодезичних робіт на основі інформації, отриманої топографічним або аерографічними методами.
16. *Системи відеомонтажу.* Призначені для цифрової оброблення відеоматеріалів, монтажу, створення відеоефектів, виправлення дефектів, додавання звуку, титрів та субтитрів. Окремі категорії представляють навчальні, довідкові та розважальні системи й програми. Характерною особливістю є підвищені вимоги до мультимедійної складової.
17. *Інструментальні мови та системи програмування.* Ці засоби служать для розробки нових програм. Комп'ютер «розуміє» і може виконувати програми у машинному коді. Кожна команда при цьому має вигляд послідовності нулів й одиниць. Писати програми машинною мовою дуже незручно, а їх надійність низка. Тому програми розробляють мовою, зрозумілою людині (інструментальна мова або алгоритмічна мова програмування), після чого спеціальною програмою, яка називається транслятором, текст програми перекладається (транслюється) на машинний код. Транслятори бувають двох типів.

1. *Інтерпретатор* читає один оператор програми, аналізує його і відразу виконує, після чого переходить до оброблення наступного оператора.
2. *Компілятор* спочатку читає, аналізує та перекладає на машинний код усю програму і тільки після завершення всієї трансляції ця програма виконується.

   Інструментальні мови поділяються на мови низького рівня (близькі до машинної мови) та мови високого рівня (близькі до мови людини).

Інструментальні мови поділяються на мови низького рівня (близькі до машинної мови) та мови високого рівня (близькі до мови людини). До мов низького рівня належать асемблери, а високого — Pascal, Basic, C/C++, C#, Java, Pyhton, мови баз даних і т.д.

Систему програмування, крім транслятора, складають текстовий редактор, компонувальник, бібліотека стандартних програм, налагоджувач, візуальні засоби автоматизації програмування, IDE. Прикладами таких систем є Delphi, Visual Basic, Visual C++, PyCharm (рис. 3.3.1) та ін.

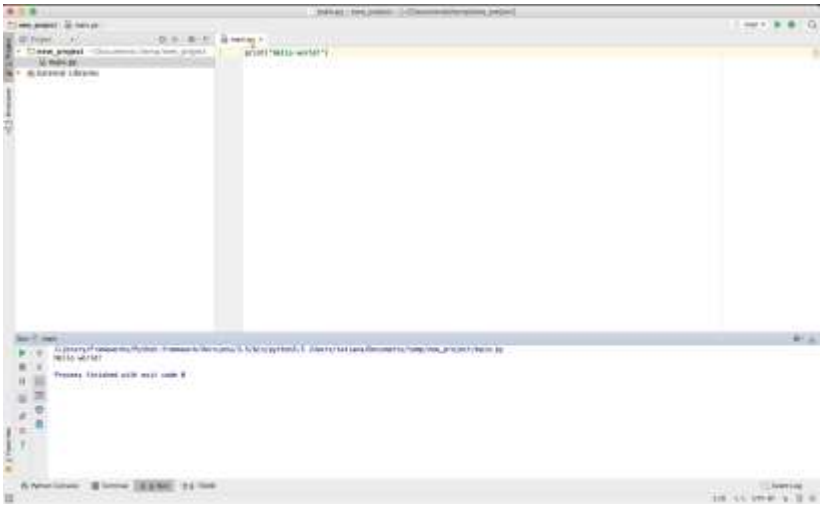


Рисунок 3.3.1. Головне вікно IDE PyCharm

Питання до розділу

1. Комп’ютерна програма
2. Програмне та апаратне забезпечення
3. Програмна конфігурація. Міжпрограмний інтерфейс
4. Структура програмного забезпечення
5. Базовий рівень програмного забезпечення
6. Системний рівень програмного забезпечення
7. Операційна система
8. Інтерфейс командного рядка
9. Класифікація службових програмних засобів
10. Класифікація прикладного програмного забезпечення
11. Інструментальні мови та системи програмування.

Завдання до розділу

1. За допомогою командного рядка:

   а) створити текстовий файл data.txt та записати в нього «hello»;

   б) скопіювати утворений файл в copy.txt;

   в) подивитися вміст утвореного файлу copy.txt;

   г) видалити файл copy.txt.

2. За допомогою командного рядка знайти в поточній директорії:

   а) усі текстові файли

   б) усі малюнки

   в) усі підкаталоги

4. Алгоритми та блок-схеми

Вступ

Слово алгоритм походить від імені перського вченого, астронома та математика Аль-Хорезмі. Приблизно 825 до н. е. він написав трактат, в якому описав придуману в Індії позиційну десяткову систему числення. В першій половині XII століття книжка потрапила до Європи в перекладі латинською мовою під назвою Algoritmi de numero Indorum. Вважається, що перше слово в перекладі відповідає невдалій латинізації імені Аль-Хорезмі, а назва перекладу звучить як «Алгоритмі про індійську лічбу». Через невірне тлумачення слова Algoritmi як іменника в множині ним стали називати метод обчислення.

Поняття алгоритму належить до первісних, основних, базисних понять математики, таких, як множина чи натуральне число. Обчислювальні процеси алгоритмічного характеру (арифметичні дії над цілими числами, знаходження найбільшого спільного дільника двох чисел тощо) відомі людству з глибокої давнини. Проте, в явному вигляді поняття алгоритму сформувалося лише на початку XX століття.

Неформально, алгоритм — це список добре визначених інструкцій для розв'язання задачі. Починаючи з початкового стану, інструкції алгоритму описують процес обчислення, які відбуваються через послідовність станів, які, зрештою, завершуються кінцевим станом.

Перший алгоритм, призначений для виконання на автоматичному обчислювальному пристрої (комп'ютері), описала Ада Лавлейс (рис. 4.1) у 1843 році. Алгоритм мав обчислювати числа Бернуллі й працювати на аналітичній

машині Беббіджа. Цей алгоритм вважається першою комп'ютерною програмою, а його розробниця, Ада Лавлейс — першим програмістом.

Рисунок 4.1. Зображення Ади Лавлейс біля логотипу ACM

З 1930-тих років починається бурхливий розвиток галузі дослідження алгоритмів та становлення інформатики в її сучасному вигляді. В 1935 році Стівен Кліні розробив перше формулювання теорії рекурсії та показав еквівалентність розробленої системи частково рекурсивним функціям. В 1936 році незалежно були опубліковані роботи Алана Тюрінга та Еміля Поста, в яких наведено схожі системи для визначення поняття алгоритму. А вже в 1937 році було доведено еквівалентність різних визначень.

Машина Тюрінга пропонувала конструкцію, придатну для автоматичної інтерпретації. Побудована під впливом Алана Тюрінга у Великобританії обчислювальна система

ACE (завершена в 1950 році), та системи, побудовані за архітектурою фон Неймана в США та в інших країнах (наприклад, МЕЛМ — Мала Електронна Лічильна Машина, розроблена в 1950 році в Інституті електротехніки АН України групою вчених під керівництвом Сергія Лебедєва), були першими універсальними обчислювальними пристроями, які повністю задовольняли вимоги обчислювальності.

Протягом 1935–1960 років було розроблено численні ідеї та технології, які лягли в основу сучасної інформатики.

Формальні засоби для представлення алгоритмів мали не лише теоретичну значущість. Розробка алгоритмічних мов для програмування обчислювальних пристроїв розпочалась в 1951 році з публікації німецького інженера Ганса Рутісхаузера (нім. Heinz Rutishauser). Спочатку важливою здавалась проблема нотації, її досліджував Олексій Ляпунов, але поступово вона відійшла на другий план.

В середині 1950-тих було розроблено мову програмування Фортран Джоном Бекусом з IBM. В кінці 1950-тих було розроблено Алгол та Кобол; для навчальних цілей Бейсік (1963) та Паскаль (початок 1970-тих). Для системного програмування в Bell Laboratories було розроблено C.

Лябмда-числення дало поштовх до створення в кінці 1950-тих функціональної мови програмування Лісп. На основі ідей Ліспу було створено Scheme. Мова ML була розроблена Робіном Мілнером в кінці 1970-тих. В кінці 1980-тих була створена мова Haskell.

Під впливом досліджень в галузі штучного інетелекту розроблено парадигму логічного програмування. На відміну від імперативної та функціональної парадигм, логічне програмування не вимагає від розробника описувати спосіб розв'язання

поставленої задачі. Planner — перша мова логічного програмування, була розроблена в 1969 році. На початку 1970-тих був розроблений Пролог, який згодом став найпоширенішою мовою логічного програмування зі стандартом ISO, затвердженим у 1995 році.

4.1. Алгоритм та його властивості

Формально, *алгоритм* (латинізов. Algorithmi, від імені перського математика IX ст. аль-Хорезмі)— послідовність, система, набір систематизованих правил виконання обчислювального процесу, що обов'язково приводить до розв'язання певного класу задач після скінченного числа операцій. При написанні комп'ютерних програм алгоритм описує логічну послідовність операцій. Для візуального зображення алгоритмів часто використовують блок-схеми.

В якості прикладу можна навести алгоритм Евкліда — ефективний метод обчислення найбільшого спільного дільника (НСД). Названий на честь грецького математика Евкліда, один з найдавніших алгоритмів, що досі використовують. Описаний в Началах Евкліда (приблизно 300 до н. е.), а саме, в книгах VII та X. У сьомій книзі алгоритм описано для цілих чисел, а в десятій — для довжин відрізків.

Існує декілька варіантів алгоритму, розглянемо один із них.

Поки A не дорівнює B виконуємо наступне: якщо $A > B$, то від A віднімаємо B, інакше від B віднімаємо A. Повертаємо результат A. Наприклад, для чисел $A = 9$ та $B = 27$ цей алгоритм працює наступним чином: A не дорівнює B, тому виконуємо кроки. $B > A$, тому в B записуємо $B - A$, тобто 18. Знову порівнюємо A та B. Оскільки вони не рівні — продовжуємо виконувати кроки алгоритму: оскільки

$B > A$, то від B віднімаємо A, тобто B тепер дорівнює $18 - 9 = 9$. Знову порівнюємо A та B. Вони рівні, отже алгоритм завершений, результат — 9.

Формально цей алгоритм можна записати наступним чином:

ПОКИ $A \neq B$:

ЯКЩО $A > B$:

$A := A - B$

ІНАКШЕ:

$B := B - A$

ПОВЕРНУТИ A

Такий спосіб запису називається *псевдокод.*

Кожен алгоритм передбачає існування початкових (вхідних) даних та в результаті роботи призводить до отримання певного результату. Робота кожного алгоритму відбувається шляхом виконання послідовності деяких елементарних дій. Ці дії називають кроками, а процес їхнього виконання називають алгоритмічним процесом.

Алгоритми мають ряд важливих *властивостей*.

1. *Скінченність* — алгоритм має завжди завершуватись після виконання скінченної кількості кроків. Процедуру, яка має решту характеристик алгоритму, без, можливо, скінченності, називають методом обчислень.
2. *Дискретність* — процес, що визначається алгоритмом, можна розчленувати (розділити) на окремі елементарні етапи (кроки), кожен з яких називається кроком алгоритмічного процесу чи алгоритму [30].
3. *Визначеність* — кожен крок алгоритму має бути точно визначений. Дії, які необхідно здійснити, повинні бути

чітко та недвозначно визначені для кожного можливого випадку.

4. *Вхідні дані* — алгоритм має деяку кількість (можливо, нульову) вхідних даних, тобто, величин, заданих до початку його роботи або значення яких визначають під час роботи алгоритму.
5. *Вихідні дані* — алгоритм має одне або декілька вихідних даних, тобто, величин, що мають досить визначений зв'язок із вхідними даними.
6. *Ефективність* — алгоритм вважають ефективним, якщо всі його оператори досить прості для того, аби їх можна було точно виконати за скінченний проміжок часу з допомогою олівця та аркушу паперу.
7. *Масовість* — властивість алгоритму, яка полягає в тому, що алгоритм повинен забезпечувати розв'язання будь-якої задачі з класу однотипних задач за будь-якими вхідними даними, що належать до області застосування алгоритму.

Поширеним критерієм оцінки алгоритмів є *час роботи* та *порядок зростання тривалості роботи* в залежності від обсягу вхідних даних. Кожній конкретній задачі зіставляють деяке число, яке називають її розміром. Наприклад, розміром задачі обчислення добутку матриць може бути найбільший розмір матриць-множників, для задач на графах розміром може бути кількість ребер графа. Час, який витрачає алгоритм як функція від розміру задачі n, називають часовою складністю цього алгоритму T(n). Асимптотику поведінки цієї функції при збільшенні розміру задачі називають асимптотичною часовою складністю, а для її позначення використовують нотацію Ландау (велике O).

Саме асимптотична складність визначає розмір задач, які алгоритм здатен обробити. Наприклад, якщо алгоритм

обробляє вхідні дані розміром n за час cn^2, де c — деяка стала, то кажуть, що часова складність такого алгоритму $O(n^2)$.

Часто, під час розробки алгоритму намагаються зменшити асимптотичну часову складність для найгірших випадків. На практиці ж, трапляються випадки, коли достатнім є алгоритм, який «зазвичай» працює швидко. Грубо кажучи, аналіз середньої асимптотичної часової складності можна поділити на два типи: аналітичний та статистичний. Аналітичний метод дає точніші результати, але складний у використанні на практиці. Натомість статистичний метод дозволяє швидше здійснювати аналіз складних задач.

В додатку В наведено поширені асимптотичні складності з коментарями.

4.2. Представлення алгоритмів у вигляді блок-схем

У процесі розробки алгоритму можуть використовуватись різні способи його опису, які відрізняються за простотою, наочністю, компактністю, мірою формалізації, орієнтації на машинну реалізацію тощо.

Для опису алгоритмів людина часто користується природною мовою, але для запису багатьох алгоритмів природна мова виявилась незручною, тому виникла необхідність у створенні штучних мов, наприклад мови математичних формул, хімічних процесів тощо. Існує спеціальна навчальна алгоритмічна мова, яка була створена для запису алгоритмів на папері; вона використовує слова природної мови, але має більш жорстку структуру. Найбільше поширення для запису логічної структури алгоритмів отримали графічні (структурні) блок-схеми, які спрощують складання та

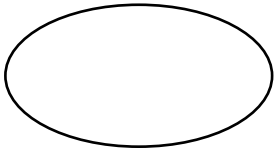
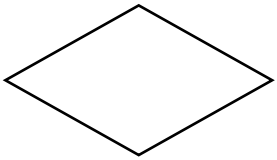
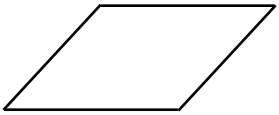
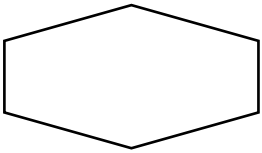
аналіз алгоритму, полегшують перехід від запису алгоритму до написання програми.

Блок-схема алгоритму — це графічне зображення алгоритму у вигляді спеціальних блоків з необхідними словесними поясненнями (табл. 4.2.1). Кожний етап алгоритму представляється у вигляді геометричної фігури (блоку), що має певну форму в залежності від характеру операції. Блоки на схемі з'єднуються стрілками (лініями зв'язку), які визначають послідовність виконання операцій та утворюють логічну структуру алгоритму.

Важливою особливістю базових структур алгоритмів є те, що вони мають один вхід і один вихід, що дозволяє при відносній незалежності конструювати окремі блоки алгоритмів, а потім окремо розроблені структури з'єднувати між собою (вихід однієї базової структури сполучається із входом іншої). Весь алгоритм представляє лінійну послідовність базових структур.

Розглянемо представлення базових структур — композицій — алгоритмів у вигляді блок-схем.

Таблиця 4.2.1. Елементи блок-схеми алгоритму

Елемент	Опис
	Початок і кінець алгоритму
	Арифметичний блок, в який записується операція. В таких блоках знак «=» це знак присвоєння (іноді записують «:=»)
	Логічний блок, в який записується умова
	Блок, в який записуються дані, що вводяться чи повертаються
	Запис умови циклу

Існують три основні (базові) композиції: послідовне виконання, розгалуження та цикл. Вони застосовуються у,

відповідно, лінійних, алгоритмах з розгалуженням та циклічних алгоритмах.

4.2.1. Послідовне виконання та лінійні алгоритми

Лінійний алгоритм (послідовне виконання, структура слідування) — це алгоритм, який забезпечує отримання результату шляхом одноразового виконання послідовності дій, незалежно від вхідних даних і проміжних результатів. Дії в таких алгоритмах виконуються послідовно, одна за однією, тобто лінійно (див. рис. 4.2.1.1).

Рисунок 4.2.1.1. Алгоритм визначення будови слова

Проаналізуємо блок-схему лінійного алгоритму, подану на рисунку 4.2.1.1. Початок і кінець будь-якого алгоритму позначається овальними блоками; у прямокутних блоках вказується крок алгоритму; стрілочка вказує, який крок буде виконуватися наступним.

Приклад 4.2.1.1. Скласти блок-схему алгоритму обчислення значення виразу $20 \cdot x - (40 + y) : 8$

Блок-схема для даного виразу подана на рис. 4.2.1.2.

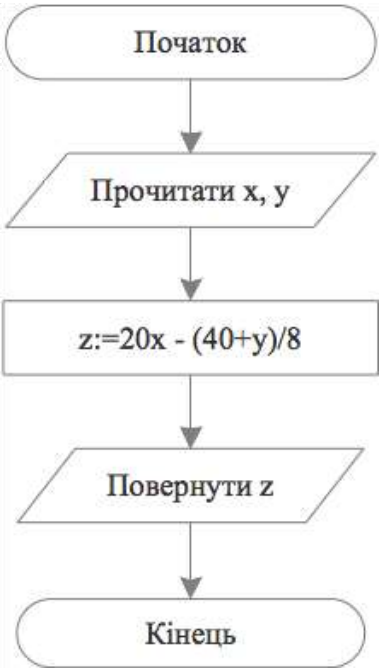


Рисунок 4.2.1.2. Алгоритм обчислення значення виразу $20 \cdot x - (40 + y) : 8$

Операторний блок з рисунку 4.2.1.2 містить операцію присвоєння :=. Елемент зліва від присвоєння (z) називають *змінною*, елемент справа (20x-(40+y)/8) — *виразом*. Присвоєння дозволяє зберегти значення у змінній щоб в

подальшому його використовувати, в тому числі в інших виразах.

Приклад 4.2.1.2. Створити блок-схему алгоритму обчислення функції $X \cdot sin(X/2)$ по введеному значенню аргументу X.

В цій задачі треба послідовно виконати ряд кроків, однакових для всіх вхідних даних, отже це буде лінійний алгоритм (рис.4.2.1.3) .

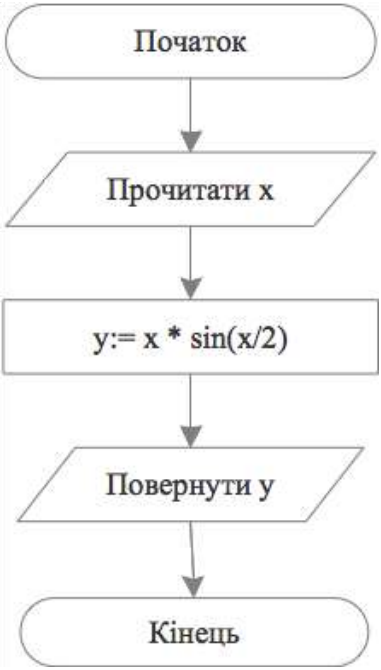


Рисунок 4.2.1.3. Блок-схема алгоритму обчислення функції $X \cdot sin(X/2)$

Приклад 4.2.1.3. Створити блок-схему алгоритму знаходження значення виразу $z = x - 5 \cdot y + 10$ де $y = 3 \cdot x + 4$.

Блок-схема лінійного алгоритму представлена на рис. 4.2.1.4 .

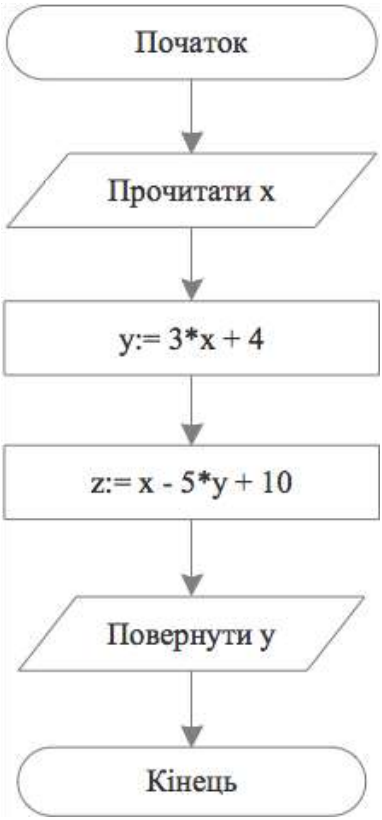


Рисунок 4.2.1.4. Блок-схема алгоритму обчислення значення виразу $z = x - 5 \cdot y + 10$ *де* $y = 3 \cdot x + 4$

4.2.2. Розгалуження та алгоритми з розгалуженням

Висловлювання та їх застосування в алгоритмах

Для знайомства з іншими композиціями необхідно спочатку дати визначення поняттю «висловлювання».

Висловлювання — речення, що виражає судження. Якщо судження, що становить зміст (сенс) деякого висловлювання, є істинним, то і про це висловлювання кажуть, що воно істинне. Подібним чином помилковим називають таке висловлювання, яке є вираженням помилкового судження. Істинність та хибність називаються логічними, або істинними, значеннями висловлювань.

Висловлювання зазвичай протиставляються наказовим, питальним та будь-яким іншими реченням, оцінка істинності чи хибності яких неможлива. Приклад елементарного

Приклад 4.2.2.1. Вираз «Число 5 менше за число 7» є висловлюванням, про яке ми можемо говорити, що воно є істинним. А про вираз «Відкрий, будь-ласка, вікно!» ми не можемо сказати, істинний він, чи хибний.

Розділ математики, який вивчає логічні висловлювання, відноситься до *математичної логіки*. Математична логіка здебільшого не цікавиться, чому те чи інше висловлювання є вірним чи хибним. Натомість математична логіка вивчає, як з одних висловлювань можна конструювати інші («складені») в такий спосіб, щоб значення нового висловлювання повністю визначалося значеннями висловлювань, з яких воно утворене. Для цього використовуються логічні сполучники, найважливіші з них: заперечення (не), кон'юнкція (і) та диз'юнкція (або).
Приклад 4.2.2.2. Приклад складеного логічного

висловлювання: «Число 5 менше за число 7 і 5 є парним числом». Це висловлювання складається із двох простих: «Число 5 менше за число 7» та «5 є парним числом», — поєднаних сполучником «і». Якщо перше висловлювання є істинним, а друге — хибним, то можемо говорити, що складене з них логічне висловлювання є хибним.

Найважливіші логічні сполучники

Значення істинності для логічних операцій, зазвичай задається за допомогою таблиць істинності. В таблиці істинності за допомогою 1 позначаємо істину (True), а за допомогою 0 — хибу (False).

Розглянемо таблиці істинності для операцій, які задають найважливіші логічні сполучники: «не», «і» та «або».

Заперечення (¬, «не») — це одномісний (унарний) сполучник з приєднаною функцією, поданою в табл. 4.2.2.1.

Таблиця 4.2.2.1. Таблиця істинності для операції заперечення

A	$\neg A$
1	0
0	1

З таблиці 4.2.2.1 видно, що якщо висловлювання А є істиним, то складене висловлювання $\neg A$ є хибою.

Приклад 4.2.2.3. Нехай A позначає висловлювання «Йде дощ». Тоді $\neg A$ — «Не йде дощ», що має протилежну істинісну оцінку.

Кон'юнкція (&, «і») — це двомісний (бінарний) сполучник з приєднаною функцією, поданою в табл. 4.2.2.2.

Таблиця 4.2.2.2. Таблиця істинності для операції кон'юнкції

A	B	$A \& B$
1	1	1
1	0	0
0	1	0
0	0	0

З таблиці 4.2.2.2 видно, що висловлювання $A \& B$ буде істинним в одному єдиному випадку: коли і A, і B є одночасно істинними висловлюваннями.

Диз'юнкція ($\vee$, «або») — це двомісний (бінарний) сполучник з приєднаною функцією, поданою в табл. 4.2.2.3.

Таблиця 4.2.2.3. Таблиця істинності для операції диз'юнкції

A	B	$A \vee B$
1	1	1
1	0	1
0	1	1
0	0	0

З таблиці 4.2.2.3 видно, що висловлювання $A \vee B$ буде існиним коли або A, або B є істиним висловлюванням.

Як і звичайні арифметичні операції, логічні сполучники можна комбінувати, утворюючи нові висловлювання. Порядок їх застосування найчастіше визначається дужками.

Як і в арифметиці, щоб зменшити кількість дужок та зробити складені висловлювання більш виразними, використовують домовленість про порядок дій:

1. першим завжди застосовують заперечення,
2. після заперечення застосовують кон 'юнкцію
3. далі — диз ‘юнкцію.

Приклад 4.2.2.4. При яких значеннях A та B вираз $A \vee \neg B$ буде істиним.

Визначимо порядок операцій:

Спочатку треба знайти $X = \neg B$, а далі — $A \vee X$.

Побудуємо табличку для всіх варіантів значень A та B (всього чотири різних комбінації), в якій кожний стовпчик буде являти собою результат обчислення на відповідному кроці:

A	B	X $\neg B$	Результат $A \vee X$
1	1	0	1
1	0	1	1
0	1	0	0

0	0	1	1

Відповідь: вираз $A \vee \neg B$ буде істиним, якщо $A \neq 0$ та $B \neq 1$ одночасно.

Приклад 4.2.2.5. При яких значеннях A, B та C вираз$A \vee B \vee \neg(A \,\&\, \neg(C \vee D))$ буде істинним.

Визначимо порядок операцій. Спочатку підрахуємо $X_1 = C \vee B$, потім $X_2 = \neg X_1$, далі $X_3 = A \,\&\, X_2$, $X_4 = \neg X_3$, $X_5 = B \vee X_4$, і результат обчислимо, як $A \vee X_5$.

Побудуємо табличку для всіх варіантів значень A, B та C (всього вісім різних комбінацій):

A	B	C	X_1 $C \vee B$	X_2 $\neq x_1$	X_3 $A \,\&\, X_2$	X_4 $\neg X_3$	X_5 $B \vee X_4$	Результат $A \vee X_5$
1	1	1	1	0	0	1	1	1
1	1	0	1	0	0	1	1	1
1	0	1	1	0	0	1	1	1
1	0	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
0	1	0	1	0	0	1	1	1
0	0	1	1	0	0	1	1	1
0	0	0	0	1	0	1	1	1

Розгалуження (умова, структура вибору) — у класичному варіанті ця структура розглядається як вибір дій у разі

виконання або невиконання заданої умови. Розгалуження бувають повними і неповними.

Приклад 4.2.2.6. Алгоритм поведінки на пішохідному переході визначається сигналом світлофору. Якщо виконується умова «горить зелене світло» — переходимо дорогу, інакше — чекаємо. Це алгоритм з розгалуженням.

Повне розгалуження — це розгалуження, в якому певні дії визначені й у разі виконання, і в разі невиконання умови. Неповне розалуження — це розгалуження, в якому дії визначені тільки у разі виконання (або у разі невиконання) умови.

Приклад 4.2.2.7. Прикладом алгоритму з неповним розгалуженням може бути: якщо виконується умова «дзвонить телефон», то взяти трубку.

Умова переходу записується у ромбі, з якого виходить не одна, а дві стрілочки. Перехід по одній із них відбувається при істинності умови, по іншій — у випадку хиби. При цьому перша стрілочка позначається відповідно «Так» (або «+», «True» тощо), а друга — «Ні» (або «-», «False» тощо),

Розглянемо розгалуження на прикладі алгоритму, що визначає написання префіксів с- або з-. Префікс слід обирати залежно від літери, з якої починається корінь слова. Якщо корінь слова починається з літер к, п, т, ф, х, то пишеться префікс с-. В іншому разі пишеться префікс з-. В даному випадку умовою є «перша літера кореня одна з: к, п, т, ф, х».

Приклад 4.2.2.8. Побудувати блок-схему алгоритму написання префіксів с- / з-.

Блок-схема алгоритму подана на рис. 4.2.2.1.

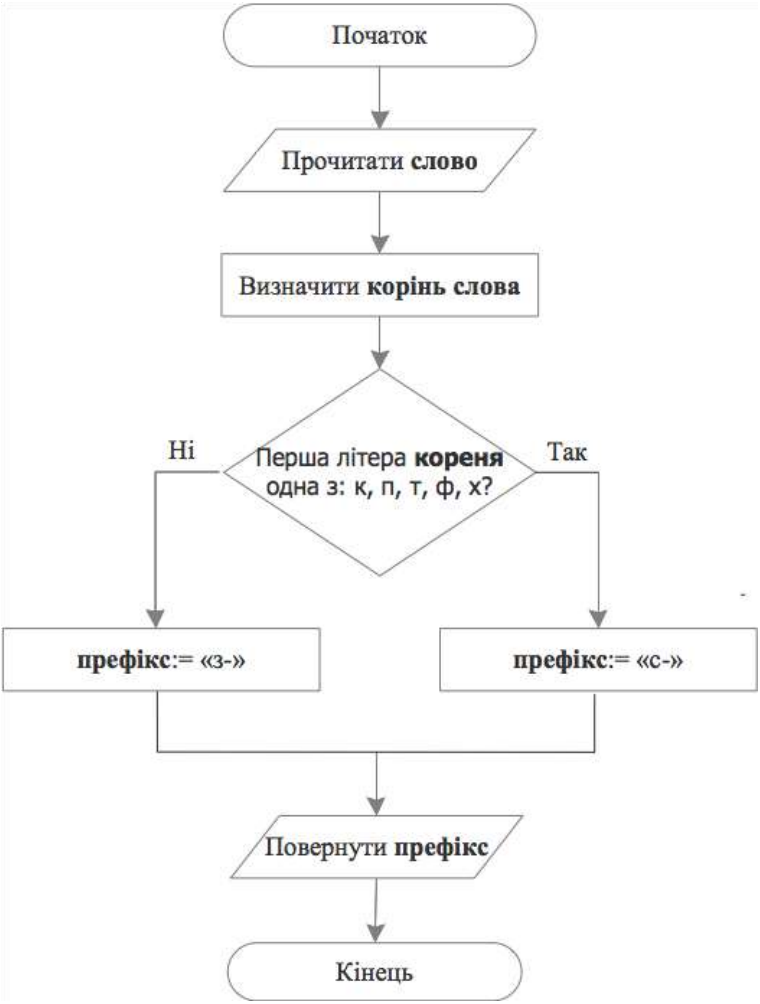


Рисунок 4.2.2.1. Алгоритм правила написання префіксів

Приклад 4.2.2.9. Описати блок-схему алгоритму знаходження максимуму з двох чисел.

Блок-схема алгоритму подана на рис. 4.2.2.2.

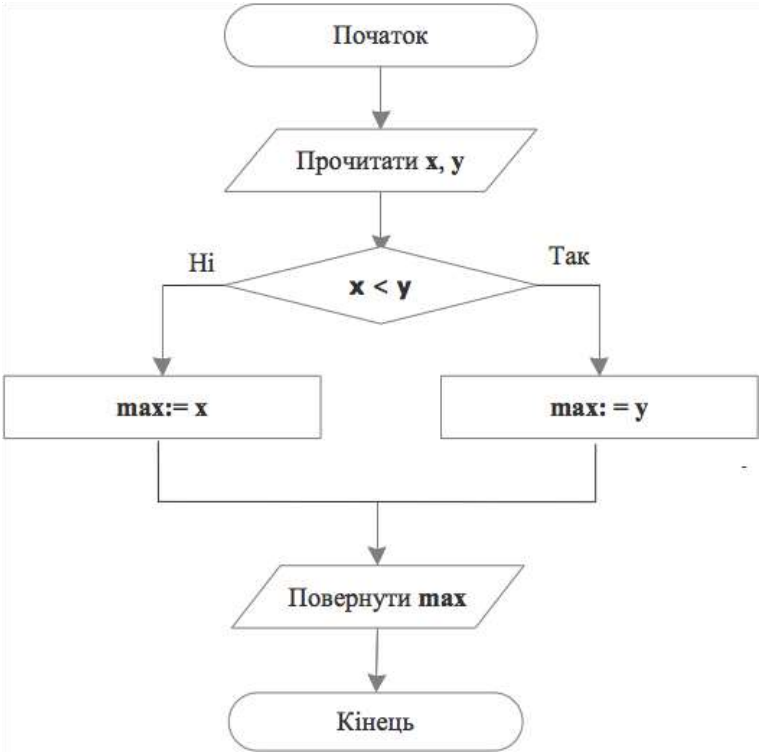


Рисунок 4.2.2.2. Алгоритм знаходження максимуму двох чисел

Розглянемо неповне розгалуження на прикладі алгоритму знаходження максимуму двох чисел без використання третьої змінної (рис 4.2.2.2).

Приклад 4.2.2.10. Описати блок-схему алгоритму знаходження максимуму з двох чисел без використання третьої змінної.

Блок-схема алгоритму подана на рис. 4.2.2.3.

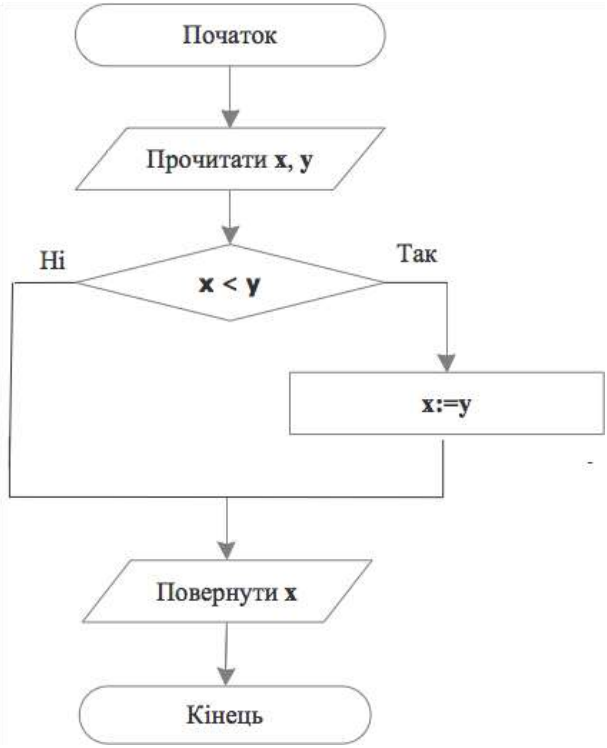


Рисунок 4.2.2.3. Алгоритм знаходження максимуму з двох чисел без використання третьої змінної

Як бачимо, іноді перевірка умови необхідна для того, щоб визначити, чи треба виконувати певну дію.

Приклад 4.2.2.11. Знайти корені квадратного рівняння $a\dot{x}^2 + b \cdot x + c = 0$, де $a, b, c > 0$.

Блок-схема алгоритму подана на рис. 4.2.2.4.

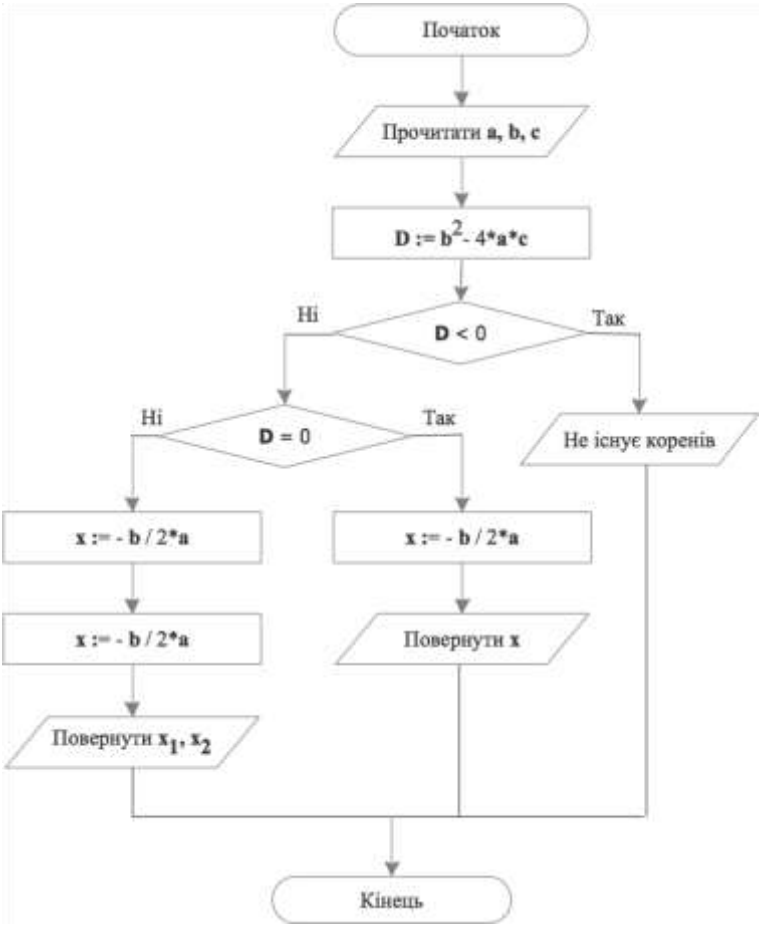


Рисунок 4.2.2.4. Блок-схема алгоритму знаходження коренів квадратного рівняння

4.2.3. Цикл та циклічні алгоритми

Циклічними називають процеси, в яких дії повторюються в одній і тій самій послідовності.

Циклічні процеси ми можемо спостерігати у природі. Планета Земля кожного року проходить один і той самий шлях навколо Сонця. Кожного року відбувається зміна пір року: зима, весна, літо, осінь, потім знову зима. Кожну добу день змінюється ніччю, а ніч — днем.

Циклічно змінюються фази Місяця: спочатку Місяць молодий, потім він росте, згодом досягає повного Місяця, відтоді починає зменшуватися, і знову все починається спочатку.

Циклічний алгоритм (цикл, структура повторення) — це алгоритм, у якому передбачено повторення деякої серії команд. За допомогою цієї структури описуються однотипні дії, що повторюються декілька разів. Такі алгоритми забезпечують виконання довгої послідовності дій, записаних порівняно короткою послідовністю команд. Саме використання циклів дозволяє у повній мірі реалізувати швидкодію комп'ютерів.

Розрізняють два типи циклів: з умовою та із змінною. Цикл з умовою передбачає виконання певних дій поки виконується вказана умова. Умова циклу так само, як і в розгалуженні, поміщається в ромбі. Єдина суттєва відмінність в тому, що в циклічних алгоритмах за допомогою стрілочки ми можемо повернутися назад.

Цикл зі змінною передбачає виконання певного набору дій для змінної, що «пробігає» вказаний набір значень (наприклад, «для всіх x від 1 до 10» передбачає виконання десяти кроків для $x = 1$, $x = 2$, …, $x = 10$). В цьому випадку змінна та значення, які вона «пробігає» вказується в шестикутнику.

Розглянемо цикл на прикладі алгоритму прибирання своїх речей з парти після уроків.

Приклад 4.2.3.1. Описати блок-схему алгоритму прибирання своїх речей з парти після уроків.

Блок-схема алгоритму подана на рис. 4.2.3.1.

Рисунок 4.2.3.1. Алгоритм прибирання своїх речей з парти після уроків

Приклад 4.2.3.2. Знайти значення функції $y = x^2$ для x від -5 до 5 з кроком 0,5.

Блок-схема алгоритму подана на рис. 4.2.3.2.

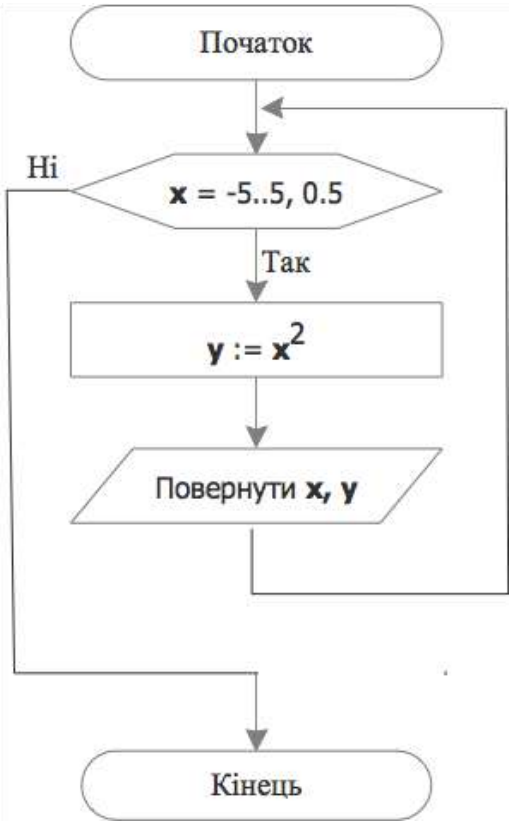


Рисунок 4.2.3.5. Блок-схема алгоритму обчислення функції $y = x^2$ для x від -5 до 5 з кроком 0,5

4.3. Алгоритми автоматичного оброблення текстів

На сьогоднішній день існує ряд підходів до автоматичного оброблення текстів (далі — АОТ). Їх можна класифікувати за задачами, які вони дають можливість розв'язувати, та за методами, які вони використовують для оброблення тексту.

Задачі АОТ можна розбити умовно на дві основні категорії. Перша — задачі, з якими кожного дня стикається користувач: перевірка орфографії, фільтрація спаму, автоматичний переклад тощо (на сьогоднішній день ці задачі практично розв'язані). Особливий інтерес представляє друга категорія, що вимагає оброблення великих об'ємів текстів для таких задач, як знаходження відповідей на питання, оцінка суджень тощо. Особливістю цих задач є відсутність формалізації, у зв'язку з чим для них не існує повноцінного набору рішень, а застосовуються допоміжні методи.

Теоретичну основу АОТ складає комп'ютерна лінгвістика, основні методи якої — машинне навчання, статистичний аналіз, моделі Маркова, логічні моделі і модифікація цих методів з урахуванням Великих даних (big data).

На сьогоднішній день існує два підходи у сфері оброблення текстів: на основі моделей мови та правил, складених експертами та на базі машинного навчання. Перший підхід дає результат кращий, проте укладання правил є процесом трудомістким у порівнянні з другим.

Серед задач АОТ можна виділити наступні.

1. *Аналіз та градація суджень.* Співвіднесення тексту, написаного від першої особи, з дискретною шкалою оцінок: погано, добре, дуже добре і т.д.

Використовується для аналізу відгуків в інтернет-магазинах і висловлювань в соціальних мережах.

2. *Аналіз тональності суджень.* Аналіз тональності висловлювань. Виявлення позитивного або негативного ставлення до обговорюваного предмету. Використовується для аналізу відгуків, генерації діалогу і т. Д.
3. *Класифікація текстів за темами.* Визначення тематики тексту. Використовується в багатьох додатках, зокрема, в рекомендаційних системах, для рубрикації текстів в онлайн-бібліотеках і для організації новинних потоків.
4. *Генерація мови.* Використовується в робототехніці, смартфонах, навігаторах.
5. *Ведення діалогу.* Аналіз реплік співрозмовника і формування на їх основі відповідей. Використовується в робототехніці, експертних системах - наприклад, Королівський банк Шотландії частково замінив контакт-центри роботами, що підтримують діалог з користувачем.
6. *Перевірка правопису.* Використовується в текстових редакторах, пошукових системах.
7. *Витяг смислу з тексту.* Виділення ключових слів і словосполучень, трендів, суммаризація. Застосовується в новинних системах для агрегування серії новинних повідомлень, базах знань для організації зберігання знань і виведення нових фактів.
8. *Пошук відповідей на питання*. Підбірка з питання і, можливо, контексту найбільш релевантного відповіді. Застосовується в пошукових і експертних системах.
9. Машинний переклад.

Корпуси — невід'ємна частина багатьох систем оброблення текстів. Кожне слово в корпусах забезпечено вичерпними граматичними характеристиками: до якої частини мови воно належить, в якій формі воно знаходиться, яка його синтаксична роль. Корпуси служать вхідними даними для навчання в задачах класифікації текстів за темами та жанрами, для навчання синтаксичних парсерів і програм, які використовуються для зняття омонімії та дозволу анафори. Паралельні корпуси, що складаються з однакових текстів на різних мовах, використовують для навчання машинних перекладачів. Як правило, корпуси збираються десятиліттями, і в їх створенні беруть участь великі дослідницькі групи.

Важливий тип вхідних даних будь-якої системи АОТ — *морфологічні словники*. *Тезауруси* (або семантичні мережі) — інший тип широко затребуваних вхідних даних. Мабуть, найвідоміший тезаурус — це WordNet, що представляє собою ресурс, в якому слова пов'язані з допомогою так званих семантичних відносин: синонімії, гиперонімії (приватне — узагальнення), гіпонімії (узагальнення — приватне), меронімії (частина — ціле) та ін. WordNet корисний в задачах машинного перекладу, генерації текстів, класифікації текстів.

Розв’язання практично будь-якої задачі АОТ так чи інакше включає в себе проведення аналізу тексту на декількох рівнях представлення.

1. *Графематичний аналіз.* Виділення з масиву даних пропозицій і слів (токенів).
2. *Морфологічний аналіз.* Виділення граматичної основи слова, визначення частин мови, приведення слова до словникової формі.

3. *Синтаксичний аналіз.* Виявлення синтаксичних зв'язків між словами в реченні, побудова синтаксичної структури пропозиції.
4. *Семантичний аналіз.* Виявлення семантичних зв'язків між словами і синтаксичними групами, витяг семантичних відносин.

Кожен такий аналіз — самостійна задача, яка не має власного практичного застосування, але активно використовується для вирішення більш загальних задач.

На сьогоднішній день існує ряд систем, призначених для вирішення саме допоміжних задач. Такі системи застосовуються або для апробації методів і проведення обчислювальних експериментів, або в якості складових частин (або бібліотек) для систем, які вирішують ту чи іншу прикладну задачу. Прикладом таких систем можуть служити засоби NLTK для графематичного аналізу та токенізації.

Універсалізм в АОТ має на увазі наявність в системі набору взаємопов'язаних методів і підходів. Існує два класи таких систем. До перших відносяться системи, що розробляються дослідницькими департаментами великих компаній: IBM, Intel, SAS, ABBYY, Microsoft, Xerox і т. Д. В якості прикладів систем, призначених для оброблення текстів англійською мовою, можна назвати IBM Content Analytics, SAS Text Miner і IBM Watson. До другого класу відносяться відкриті інтегровані програмні пакети, створені в університетах і які становлять безліч методів і моделей, побудованих на єдиній програмній і математичній платформі. Для англійської мови можна назвати системи Apache OpenNLP, StanfordNLP, NLTK, GATE, spaCy (див. рис. 4.3.1).

Деякі системи АОТ спрямовані на аналіз текстів певних жанрів або тематики. Наприклад, система Watson

застосовується в медицині для діагностування і полегшення процедури прийняття лікарями рішень. Рекомендаційна система новинних повідомлень News360 являє собою додаток для мобільних пристроїв, за допомогою якого користувач може читати і вибирати найбільш цікаву для нього інформацію. На основі уподобань користувача система пропонує нові статті, зібрані з різних новинних порталів і відповідають конкретній тематиці. У деяких випадках ці системи вміють визначати тональність новинного повідомлення — наприклад, користувач може переглядати тільки хороші новини і виключити зі своєї стрічки всі погані.

Рисунок 4.3.1. Візуалізатор синтаксичної структури речення displaCy (https://demos.explosion.ai/displacy/)

Питання до розділу

1. Алгоритм. Властивості алгоритму
2. Критерії оцінки якості алгоритму
3. Асимптотична складність алгоритму
4. Блок-схема алгоритму
5. Елементи блок-схем
6. Послідовне виконання та лінійні алгоритми
7. Розгалуження та алгоритми з розгалуженням
8. Цикл та циклічні алгоритми
9. Задачі автоматичного оброблення текстів
10. Основні підходи до автоматичного оброблення текстів
11. Корпуси та їх застосування
12. Морфологічні словники та їх застосування
13. Семантичний аналіз

Завдання до розділу

1. На рис. 4.1. подано деякий алгоритм у вигляді блок-схеми. Яким буде результат на виході блок-схеми, якщо:

   а) a = 0 б) a = – 3 в) a = 3

2. На рис. 4.2. подано деякий алгоритм у вигляді блок-схеми. Яким буде результат на виході блок-схеми, якщо:

   а) a = 0, b = 0;
   б) a = 5, b = 10;
   в) a = 0, b = 5;
   г) a = – 10, b = 20?

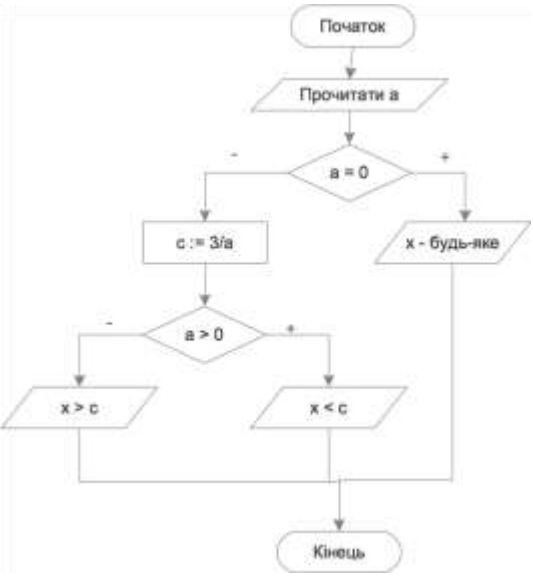


Рисунок 4.1. Блок-схема алгоритму

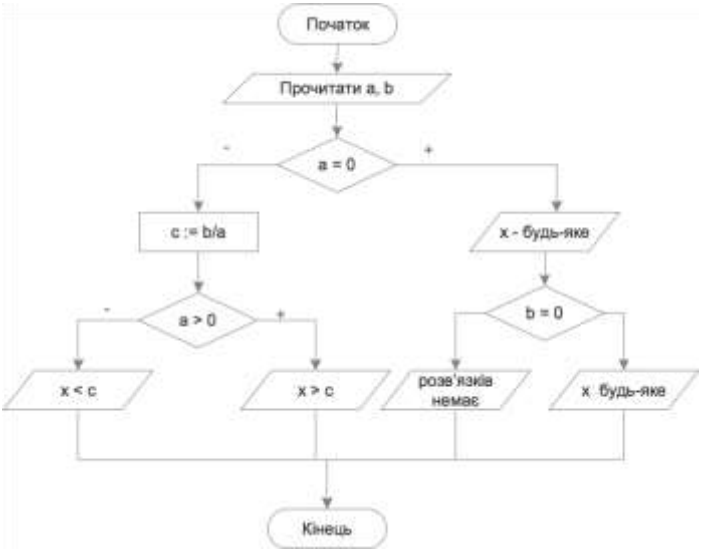


Рисунок 4.2. Блок-схема алгоритму

3. Побудувати блок-схему алгоритму визначення роду відмінків.
4. На вхід подається натуральне число x. Необхідно побудувати правильну конструкцію типу «Мені x років» або «Мені x роки» в залежності від x. Побудувати блок-схему відповідного алгоритму.
5. На вхід подається натуральне число x. Необхідно вивести назву місяця з таким номером, або повідомлення, що такого місяця не існує, якщо $x > 12$. Побудувати блок-схему відповідного алгоритму.
6. Побудувати блок-схему алгоритму обчислення значення функції $y = 5x + 7$.
7. Побудувати блок-схему алгоритму обчислення периметру квадрату за його стороною.
8. Побудувати блок-схему алгоритму обчислення площі кола за його радіусом.
9. Побудувати блок-схему алгоритму обчислення гіпотенузи прямокутного трикутника за його сторонами.
10. Побудувати блок-схему алгоритму обчислення середнього арифметичного двох чисел.
11. Побудувати блок-схему алгоритму знаходження мінімального та максимального значення із двох заданих чисел.
12. Побудувати блок-схему алгоритму знаходження мінімального та максимального значення із трьох заданих чисел.
13. Побудувати блок-схему алгоритму знаходження довшого шляху із двох, якщо перший заданий у кілометрах, а другий — у футах.

14. Побудувати блок-схему алгоритму знаходження більшого за розміром файлу, якщо подано розмір першого файлу у бітах та другого у кілобайтах.
15. Побудувати блок-схему алгоритму переведення десяткового числа у двійкову систему числення
16. Дано ціле число $m > 0$. Побудувати блок-схему алгоритму знаходження найменшого цілого k такого, що $2^k > m$
17. Дано натуральне число n. Побудувати блок-схему алгоритму обчислення:

$$P(n) = (1 + \frac{1}{1^2}) \cdot (1 + \frac{1}{2^2}) \cdot \ldots \cdot (1 + \frac{1}{n^2})$$

18. Дано натуральне число n та дійсне число x. Побудувати блок-схему алгоритму обчислення:

$$P(x, n) = x \cdot (x + 1) \cdot (x + 2) \cdot \ldots \cdot (x + n - 1)$$

19. Дано натуральне число n. Побудувати блок-схему алгоритму обчислення: $P(n) = 1 + \frac{1}{2^2} + \frac{1}{3^2} + \ldots + \frac{1}{n^2}$
20. Дано два цілих числа A та B ($A < B$). Побудувати блок-схему алгоритму знаходження всіх чисел, що розташовані між даними, у порядку зростання.

Зразки тестових завдань

Зразок тестового завдання до розділу «Системи числення»

1. Першим записом числа були:
    а) Кружечки
    б) Зарубки
    в) Трикутники
    г) Овали
2. Сукупність прийомів та правил найменування і позначення чисел називають:
    а) Арифметикою
    б) Лінгвістикою
    г) Системою числення
    д) Базою числення
3. Для системи числення множина символів, що називається цифрами, це:
    а) Код
    б) Алфавіт
    в) Синтаксис
    г) Правило
4. Місце, яке займає цифра в даному числі називається:
    а) Розряд
    б) Степінь
    в) Позиція
    г) Основа
5. Приклад непозиційної системи числення:
    а) Вавилонська
    б) Давньоєгипетська
    в) Фібоначчі
    г) Дуодецимальна

Зразок тестового завдання до розділу «Інформація та мова»

1. Інформацію, яку сприймають із навколишнього середовища називають:

а) Вхідна
б) Вихідна
в) Похідна
г) Позитивна

2. За способом сприйняття для людини інформація може бути (декілька відповідей):

а) Тактильна
б) Смакова
в) Візуальна
г) Аудіальна

3. Набір відомостей про яку-небудь особистість, що визначає соціальний стан і типи соціальних взаємодій всередині популяції називають:

а) Масовою інформацією
б) Спеціальною інформацією
в) Особистою інформацією
г) Адекватною інформацією

4. Приведення даних, що надходять із різних джерел до однакової форми це:

а) Фільтрація даних
б) Сортування даних
в) Перетворення даних
г) Формалізація даних

5. Найпоширеніша система кодування в інформатиці називається

а) Двійковим кодом
б) Вісімковим кодом
в) Десятковим кодом
г) Шістнадцятковим кодом

6. Для кодування текстової інформації прийнято міжнародний стандарт, який називається:

а) American Standard Code for Information Interchange

б) American Code Standard for Information Interchange

в) International Organization for Standardization

г) American Society for Testing and Materials

7. Кількість стандартних кодувань букв латинського алфавіту складає:

а) Одне

б) два (MS-DOS, Windows)

в) три (MS-DOS, Windows, Macintosh)

г) п’ять (MS-DOS, Windows. Macintosh, КОИ-8, ISO)

Зразок тестового завдання до розділу «Програмне забезпечення»

1. Не існує такої операційної системи?

а) Mac OS X

б) Unix

в) Word

г) Windows XP

2. Яку кількість біт містить один байт?

а) 1

б) 2

в) 4

г) 8

3. Яка з нижченаведених програм відноситься до групи програм «Стандартні» ОС Windows?

а) Word

б) Paint

в) Excel

г) ProLing Office

4. До складу операційної системи належить:

а) файлова система
б) система управління базами даних
в) система опрацювання текстів
г) навчаюча система

5. До програмного забезпечення базового рівня відноситься:
а) системи опрацювання текстів
б) BIOS
в) мультимедійні системи
г) утиліти

6. Інтерфейс користувача (поняття) – це
а) взаємодія різних програм між собою
б) взаємодія людини з комп'ютером
в) взаємодія комп'ютера з комп'ютером
г) взаємодія людини з всесвітом

7. Запис алгоритму розв'язання задачі у вигляді послідовності команд або операторів мовою, яку розуміє комп'ютер називають:
а) Апаратурою
б) Програмою
в) Правилами
г) Системою

8. Інструментальні мови та системи програмування відносяться до:
а) Прикладного рівня
б) Службового рівня
в) Системного рівня
г) Базового рівня

Зразок тестового завдання до розділу «Алгоритми та блок-схеми»

1. Алгоритм – це
а) Набір систематизованих правил виконання обчислювального процесу.

б) Набір відомостей про яку-небудь особистість, що визначає соціальний стан і типи соціальних взаємодій.
в) Набір правил до приведення даних, що надходять із різних джерел, до однакової форми.
г) Набір правил для окремих категорій інформаційних систем

2. Першим програмістом вважається:
а) Стів Джобс
б) Ада Лавлейс
в) Алан Тьюрінг
г) Еміль Пост

3. Мову програмування Pascal розробили в:
а) 1948-1950
б) 1951-1953
в) 1961-1963
г) 1968-1970

4. Важливою властивістю алгоритму є можливість:
а) застосування тільки до однакових вхідних даних
б) інтерпретації вихідних даних
в) застосування до різних вхідних даних
г) застосування до вихідних даних

5. Процес повторення дій називають:
а) Розгалуженням
б) Блок-схемою
в) Циклом
г) Зростанням

Зразки модульних контрольних робіт

Зразок завдання на модульну контрольну з тем «Системи числення» та «Інформація та мова»

1. Перевести у двійкову систему числення:
   а) 139 б) 564_8 д) $A10_{16}$
2. Перевести у десяткову систему числення:
   а) 100110_2 б) 765_8 в) $AB0_{16}$
3. В якій системі числення *x* виконується нерівність $45_x + 42_x \geq 107_x$?
4. В залі деякого кінотеатру є чотири ряди по 8 стільців в кожному. В квитку вказане місце «3-й ряд, 5-е місце». Скільки інформації в квитку?
5. Скільки місця потрібно виділити для зберігання 5 сторінок реферату, якщо на кожній сторінці 8 рядків по 24 символи?

Зразок завдання на модульну контрольну роботу з тем «Програмне забезпечення» та «Алгоритми та блок-схеми»

1. Службовий рівень програмного забезпечення
2. Властивості алгоритму
3. Опишіть блок-схему алгоритму підрахунку виразу $(24 \cdot x + 6) : 5 - 17 \cdot y$ для вхідних x та y.
4. Опишіть блок-схему алгоритму перевірки введеного числа на парність.
5. Опишіть блок схему алгоритму знаходження $n!$

Фрагмент робочої програми «Основи інформатики»

Змістовий модуль 1. Основні поняття інформатики

Тема 1. Системи числення

Основні поняття програмування. Системи числення. Позиційні та непозиційні системи числення, їх використання. Двійкова, вісімкова та шістнадцяткова системи числення. Арифметичні операції.

Тема 2. Інформація та мова

Поняття та види інформації. Властивості інформації. Кодування інформації. Мова і кодування. Двійкове кодування.

Тема 3. Стандарти кодування текстової інформації

Кодування символів та чисел. Стандарти кодування текстової інформації.

Змістовий модуль 2. Алгоритми та блок-схеми

Тема 4. Програмне забезпечення

Програмне забезпечення. Прикладні та системні програми. Системи програмування. Охорона програм та даних. Архіватори. Комп'ютерні віруси та антивіруси.

Тема 5. Алгоритми

Поняття алгоритму та програми, мови програмування. Властивості алгоритму. Представлення алгоритму.

Тема 6. Блок-схеми алгоритмів

Блок-схема алгоритму. Послідовне виконання. Алгоритми з розгалуженням. Умовний оператор. Складені умови. Цикли.

Критерії оцінювання теоретичних завдань

Контроль рівня засвоєння теоретичного матеріалу здійснюється в усній та письмовій формі. При цьому завдання включає:

- теоретичні питання;
- тести.

Варіанти завдань складаються таким чином, що вони охоплюють увесь програмний матеріал дисципліни.

Правильність виконання теоретичних завдань оцінюється за шкалою: 0, 1, 2 балів.

При перевірці відповіді використовуються такі критерії:

- 0 балів — студент не дав жодної відповіді на питання. Відповідь свідчить, що студент не має уявлення про сутність питання.
- 1 бал — студент надав неповну відповідь. Допущені помилки. Виникають труднощі при аналізі матеріалу.
- 2 бали — студент надав повну відповідь на теоретичну питання. Обґрунтував свою думку.

Правильність виконання тестових завдань оцінюється за шкалою: 0, 1, 2, 3, 4, 5 балів.

При перевірці відповідей, які складають сумарно 100 відсотків, використовуються такі критерії:

- 0 балів — студент не дав жодної правильної відповіді не на одне питання - 0%. Студент не вивчав тему.
- 1 бал — студент надав відповіді на питання у відсотковому значенні до 15%. Студент досить поверхнево вивчав програмний матеріал. Отриманих

попередніх знань виявилося недостатньо при проходженні тестування.

- 2 бали — студент надав відповіді на питання у відсотковому значенні до 30% включно. Студент недостатньо був ознайомлений з програмним матеріалом. Матеріал був засвоєний тільки на рівні визначень чи понять.
- 3 бали — студент надав відповіді на питання у відсотковому значенні до 50% включно. Студент знає основну частину програмного матеріалу. Виникають труднощі при аналізі.
- 4 бали — студент надав відповіді на питання у відсотковому значенні до 80% включно. Студент повно ознайомився з теоретичним матеріалом. Допущені окремі помилки під час вибору варіантів відповідей.
- 5 балів — студент надав відповіді на питання у відсотковому значенні до 100% включно. Висновок: студент досить глибоко та повно ознайомився в потрібною темою. Володіє термінологією та елементами аналізу. Вміє співставити питання з варіантами відповідей.

Критерії оцінювання практичних завдань

Максимально практичне завдання може бути оцінено в 4 бали, якщо має місце:

- правильне виконання поставленої задачі;
- правильне використання методів при виконанні практичного завдання;
- правильно вибрані способи для виконання практичного завдання.

3 бали виставляється, якщо має місце:

- правильна послідовність виконання практичного завдання;
- правильне пояснення способів чи методів виконання завдання;
- завдання виконане загалом, але є помилки.

2 бали виставляється, якщо має місце:

- завдання виконано наполовину;
- правильне пояснення способів чи методів виконання завдання, але не вміння на практиці їх застосувати.

1 бал виставляється, якщо має місце:

- часткове виконання практичного завдання;
- не вміння чи не розуміння як застосовувати різні способи та методи для виконання.

0 балів виставляється, якщо практичне завдання не виконане.

За результатами оцінювання засвоєння студентами теоретичного матеріалу та виконання практичних завдань визначається рівень засвоєння програмного матеріалу та виставляється підсумкова кількість балів.

Схематичне представлення отримання балів студентом за два змістові модулі

Оцінка роботи студента протягом семестру:

	Змістовий модуль 1		Змістовий модуль 2	
	Мін. 18 балів	**Макс. 30 балів**	**Мін. 18 балів**	**Макс. 30 балів**
Лабораторні роботи	10	20	10	20
Модульні контрольні роботи (МКР)	8	10	8	10

Для студентів, які в семестрі набрали сумарно меншу кількість балів ніж критично-розрахунковий мінімум (36 балів) обов'язково перескладання МКР.

У випадку відсутності студента з поважних причин відпрацювання та перездачі МКР здійснюються у відповідності до «Положення про порядок оцінювання знань студентів при кредитно-модульній системі організації навчального процесу» від 1 жовтня 2010 року.

Завершальною формою контролю є іспит (40 балів). При простому розрахунку маємо:

	ЗМ1	**ЗМ2**	**Іспит**	**Підсумкова оцінка**
Мінімум	18	18	24	60
Максимум	30	30	40	100

При цьому, кількість балів:

- 1-59 відповідає оцінці «незадовільно» з можливістю повторного складання;
- 60-74 відповідає оцінці «задовільно»;
- 75 - 89 відповідає оцінці «добре»;
- 90 - 100 відповідає оцінці «відмінно».

Правила техніки безпеки і поведінки в комп’ютерному класі

Загальні положення:

- до роботи в комп'ютерному класі допускаються особи, ознайомлені з даною інструкцією з техніки безпеки і правил поведінки, інструкцією №12/2-2001;
- під час перерв між парами проводиться обов'язкове провітрювання комп'ютерного класу з обов'язковим виходом студентів з класу;
- робота студентів в комп'ютерному класі дозволяється лише у присутності викладача/методиста;
- під час занять сторонні особи можуть знаходитися в класі лише з дозволу викладача;
- кожен студент несе відповідальність за стан свого робочого місця і збереження розміщеного на ньому комп’ютерного обладнання.

Суворо *забороняється*:

- перебувати під час перерви в комп'ютерному класі без дозволу викладача/методиста;
- перебувати в комп'ютерному класі у верхньому одязі;
- заносити в комп’ютерний клас їжу чи напої.
- включати без дозволу комп’ютерне обладнання;

- доторкатись до роз’ємів сполучних кабелів та проводів електромережі (можливе враження електричним струмом);
- доторкатися до LCD-екрана монітора та його тильної сторони, пересувати маніпулятори та системний блок;
- вмикати та вимикати апаратуру без вказівки викладача;
- працювати за комп’ютером у верхньому одязі та вологими руками;
- стрибати, бігати, шуміти;
- класти диски, книги, зошити та інші предмети на монітор, клавіатуру чи системний блок;
- інсталювати або копіювати програмне забезпечення з дисків, флеш-носіїв чи інших накопичувачів на комп'ютер, попередньо не перевіривши їхнім антивірусом та без згоди викладача/методиста;

Під час роботи:

- строго виконувати всі зазначені вище правила, а також поточні вказівки викладача/методиста;
- бережно ставтесь до комп’ютерного обладнання;
- натискайте на клавіші клавіатури легко й швидко, не допускаючи різких ударів;
- не користуйтеся клавіатурою та мишею, якщо не включений комп'ютер;
- працюйте на клавіатурі чистими та сухими руками;

- не намагайтеся самостійно усунути несправність у роботі апаратури чи програмного забезпечення;
- не відволікайтесь та не вставайте зі своїх робочих місць, коли в кабінет входять сторонні відвідувачі;
- стежите за станом апаратури та програмного забезпечення та негайно повідомляйте викладача/методиста при виявленні несправності;
- з появою пожежі, диму, іскріння чи запаху гару негайно припинити роботу, вимкнути постачання енергії та сповістити про це викладачеві/методиста.

Всім студентам інституту філології перед початком роботи в комп'ютерному класі обов'язково самостійно ознайомитися з інструкцією «З безпечної роботи на ЕОМ» № 12/2-2001 Київського національного університету імені Тараса Шевченка.

Рекомендовані ресурси

1. Пошукова система:
   http://www.google.com
2. SVG-конвертер:
   http://www.rapidtables.com/web/tools/svg-viewer-editor.htm
3. Інтерактивний інструмент створення блок-схем алгоритмів:
   http://brunorb.github.io/flow-chart.js/dist/index.html
4. Навчальні та ігрові ресурси для майбутніх програмістів:
   1. https://scratch.mit.edu/
   2. https://blockly-games.appspot.com/
   3. https://checkio.org/
   4. https://www.codingame.com/start
   5. https://codecombat.com/
   6. https://codefights.com/
   7. http://theaigames.com/
   8. https://code.org/
   9. https://www.playcodemonkey.com/
   10. https://tomorrowcorporation.com/
   11. https://www.kodugamelab.com/
   12. https://www.gethopscotch.com/
   13. http://www.robozzle.com/
   14. https://www.kodable.com/
   15. https://www.tynker.com/
5. Опис класичних алгоритмів:
   https://en.wikipedia.org/wiki/List_of_algorithms
6. Візуалізації різноманітних класичних алгоритмів:
   1. https://workshape.github.io/visual-graph-algorithms/
   2. http://www.algomation.com
   3. http://algo-visualizer.jasonpark.me

4. http://www.geeksforgeeks.org/fundamentals-of-algorithms/
5. http://rosettacode.org/wiki/Rosetta_Code
6. https://www.toptal.com/developers/sorting-algorithms/
7. https://visualgo.net/en

7. Популярні ресурси для онлайн-освіти:
   1. https://www.coursera.org
   2. https://prometheus.org.ua
   3. https://www.codecademy.com/ru
8. Простий та зрозумілий онлайн-курс «Вступ до програмування:
   https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00sc-introduction-to-computer-science-and-programming-spring-2011/index.htm
9. Курс по алгоритмам на Python:
   https://www.udacity.com/course/intro-to-algorithms--cs215
10. Задачі та головоломки:
    1. https://tproger.ru/articles/problems/
    2. http://www.pythonchallenge.com/

Цей список не є вичерпним!

Використані скорочення

ASCII	American Standard Code for Information Interchange
ANSI	American National Standards Institute , Американський національний інститут стандартів
CMYK	Cyan, Magenta, Yellow, Key Color
HTML	HyperText Markup Language
NLTK	Natural Language Toolkit
UCS	Universal Character Set
UTF	Unicode Transformation Format
АСУ ТП	Автоматизована система управління технологічними процесами
АН	Академія Наук
АОТ	Автоматичне Оброблення Текстів
бл.	близько
ГІС	Геоінформаційні Системи
ГВС	Гнучкі Виробничі Системи
див.	дивись
МЕЛМ	Мала Електронна Лічильна Машина
н. е.	нашої ери

нім.	німецька
НПЗП	Напівпостійний Програмований Запам’ятовуючий Пристрій
НСД	Найбільший Спільний Дільник
ОС	Операційна Система
ПЗП	Постійний Запам'ятовуючий Пристрій
ПК	Персональний Комп’ютер
р.	рік
рис.	рисунок
рр. до н. е.	років до нашої ери
САП	Система Автоматизованого Проектування
ст.	сторіччя
СУБД	Системи Управління Базами Даних
СУМ	Словник Української Мови
табл.	таблиця
т.д.	так далі

Список використаних джерел

1. Allen J.P. Middle Egyptian: An Introduction to the Language and Culture of Hieroglyphs / Allen James Paul // Cambridge: Cambridge University Press. Numerals discussed. — 2000.
2. Golomb S.W. Run-length encodings. IEEE Transactions on Information Theory // IT-12 (3). — 1966. — P. 399–401.
3. Jansen J.M. Programming in the λ-calculus: from Church to Scott and back. The Beauty of Functional Code: Essays Dedicated to Rinus Plasmeijer on the Occasion of His 61st Birthday. / Jansen, Jan Martin // Lecture Notes in Computer Science 8106. Springer-Verlag. — 2013. — P. 168–180.
4. Magaud N. Changing data structures in type theory: a study of natural numbers. / Magaud Nicolas, Bertot Yves // Lecture Notes in Comput. Sci. 2277. Springer, Berlin. — 2002. — P. 181–196.
5. Алгоритмічні мови та програмне забезпечення [Текст] : лабораторний практикум: Навч. посіб. для студ. енергетич. спец. / П. Д. Лежнюк [та ін.] ; Вінницький держ. технічний ун-т. — Вінниця : ВДТУ. —2002. — 121 с.
6. Англійсько-український словник з програмування і математики [Текст] : понад 22 000 термінів / Інститут фундаментальних досліджень Української наукової асоціації ; укл. М. І. Кратко [та ін]. — Луцьк : Надстир'я, 1998. — 640 с.
7. Анисимов А.В. Информатика. Творчество. Рекурсия. // Ответственный редактор А.Г. Ивахненко. Научно-популярное издание. — Киев: Издательство 'Наукова

думка'. Редакция научно-популярной литературы, 1988.

8. Анисимов А.В. Компьютерная лингвистика для всех: Мифы. Алгоритмы. Язык // Для широкого круга читателей, интересующихся современными достижениями информатики, лингвистики и искусственного интеллекта. — Киев: Издательство: 'Наукова думка'. — 1991.
9. Белецкий А.А. Семиотический аспект языковой системы // Структурная и математическая лингвистика. — К., 1979. — вып.7. — С.11-18.
10. Білак Ю.Ю. Системи числення: методичні рекомендації з базової теми дисципліни «Інформатика» / Ю.Ю. Білак, Л.Я. Данько-Товтин. — Ужгород: ДВНЗ «УжНУ», 2015. — 24 с.
11. Большакова Е.И. Автоматическая обработка текстов на естественном языке и компьютерная лингвистика : учеб. пособие / Большакова Е.И., Клышинский Э.С., Ландэ Д.В., Носков А.А., Пескова О.В., Ягунова Е.В. // М.: МИЭМ, 2011. — 272 с.
12. Войтюшенко Н.М. Інформатика і комп'ютерна техніка: навч.посіб. [для студ.вищ.навч.закл.]/Н.М.Войтюшенко, А.І.Остапець. – [2-ге вид.]. – К.: Центр учбової літератури, 2009. – 564 с.
13. Глушков В.М. Гносеологические основы математизации науки // Диалектика и логика научного познания. // М., 1966. — С. 406.
14. Говорущенко Т.О. Курс лекцій «Комп'ютерна логіка» // 2015 р.

15. Жалдак М.І. Інформаційні технології. Навчально-методичний посібник / Жалдак М.І., Хомік О.А., Володько І.В., Снігур О.М. // К.: 2003. – 194 с.
16. Закон України «Про авторське право і суміжні права» // Відомості Верховної Ради України (ВВР), 1994, N 13, ст.64, зі змінами та доповненнями.
17. Закон України «Про електронні документи та електронний документообіг» // Відомості Верховної Ради (ВВР). — 2003. — №36. — ст.275, зі змінами та доповненнями.
18. Закон України «Про інформацію» //Відомості Верховної Ради України (ВВР), 1992, N 48, ст.650, зі змінами та доповненнями.
19. Закон України «Про національну програму інформатизації» // Відомості Верховної Ради (ВВР).- 1998.- №27-28. – ст.181, зі змінами та доповненнями.
20. Закон України «Про телекомунікації» // Відомості Верховної Ради (ВВР).- 2004.- №12. – ст.155, зі змінами та доповненнями.
21. Ильвовский Д. Системы автоматической обработки текстов / Ильвовский Д., Черняк Е. // Открытые системы. Вып. 1. — 2014. — С. 51-53. // Режим доступу: https://www.hse.ru/mirror/pubs/lib/data/access/ram/ticket/87/150893690733881d6140b2c0d8e8380cd9e4d90686/51-53.pdf
22. Інформатика: Комп'ютерна техніка. Комп'ютерні технології: Підручник. – К.: Каравела, 2003. – 464 с.
23. Інформатика: Комп'ютерна техніка. Комп'ютерні технології: Посіб. / За ред. О.І. Пушкаря. – К.: Видавничий центр «Академія», 2001. – 696с.

24. Інформатика. Повні уроки // Електронний ресурс. Режим доступу: http://edufuture.biz/index.php?title=Інформатика_11_клас._Повні_уроки
25. Інформаційні технології. Мови програмування, їх середовище і системний інтерфейс. Незалежні від мов типи даних [Текст]. — Введ. 2001.01.01. — Офіц. вид. — К. : Держстандарт України, 2000. — VI, 106с., VI, 110 с.
26. Карпіловська Є.А. Вступ до прикладної лінгвістики: комп'ютерна лінгвістика: Підручник. //Донецьк: ТОВ «Юго-Восток, Лтд», 2006.— 188 с.
27. Комп'ютерні технології та програмування [Текст] / С. М. Москвіна, Т. В. Грищук ; Вінниц. нац. техн. ун-т. - Вінниця : ВНТУ, 2014 .Ч. 1 : Алгоритмічні основи програмування : лаб. практикум. — 2014. — 115 с.
28. Кормен Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест // М. : МЦНМО, 2002. – 955 с.
29. Красусский М. Древность малороссийского языка // Михаил Красусский // Одесса. — 1880.
30. Матеріали з інформатики та програмування Полякова К.Ю. // Електронний ресурс. Режим доступу: http://kpolyakov.spb.ru/
31. Методика викладання інформатики: Навчально-методичний посібник / Укл.: Ленюк О.М., Лучко В.М., Тупкало І.С. – Чернівці, Рута, 2005. – 72 с.
32. Методичні розробки Костюк М.О. // Електронний ресурс. Режим доступу: http://kostiuk2014.blogspot.com/
33. Морзе Н.В. Методика навчання інформатики: Навч. посіб.: у 4 ч./ За ред. акад. М.І.Жалдака // К.:

Навч.книга, 2003. — Ч.ІІ: Методика навчання інформаційних технологій. – 287 с.

34. Основи інформатики та обчислювальної техніки [Текст] : навч. посіб./ С. В. Кунцев, В. В. Яценко; Державний вищий навчальний заклад «Українська академія банківської справи Національного банку України». − Суми : ДВНЗ «УАБС НБУ», 2011. − 104 с.
35. Основи інформаційних технологій // Електронний ресурс. Режим доступу: http://www.victoria.lviv.ua/html/oit/index.htm
36. Основи програмування та алгоритмічні мови [Текст] : лабораторний практикум для студ. напряму 0804 «Комп'ютерні науки» / уклад. М. О. Сидоров [та ін.]. — К. : НАУ. — 2003. — 72 с.
37. Програмування числових методів мовою Python [Текст] : навч. посіб. для студентів ВНЗ / [А. Ю. Дорошенко та ін.] ; за ред. чл.-кор. НАН України А. В. Анісімова ; Київ. нац. ун-т ім. Тараса Шевченка. — Київ : Київський університет, 2013. — 463 с.
38. Селезнев К. Лингвистика и обработка текстов / Константин Селезнев, Александр Владимиров // Открытые системы, № 04. — 2013. — С. 46–49. Режим доступу: https://www.osp.ru/os/2013/04/13035562/
39. Словник української мови // Електронний ресурс. Режим доступу: http://sum.in.ua
40. Тиш Є.В. Методичні вказівки розроблені у відповідності з навчальним планом напрямку 6.050102 «Комп’ютерна інженерія» // Тернопіль. — 2015.
41. Тлумачний словник з інформатики / Г.Г. Півняк, Б.С. Бусигін, М.М. Дівізінюк та ін. – Д., Нац. гірнич. ун-т, 2010. – 600 с.

42. Шеховцов В. А. Операційні системи // К.: Видавнича група BHV. — 2005.
43. Методичні розробки з інформатики та програмування, що знаходяться у вільному доступі в мережі Інтернет.

Додаток А. Зв’язок між числами десяткової, шістнадцяткової, вісімкової та двійкової систем числення

Десяткова	Двійкова	Вісімкова	Шістнадцяткова
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
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	100	100
17	10001	101	101
18	10010	102	102

19	10011	103	103
20	10100	104	104

Додаток Б. Таблиця ASCII символів

Код	Символ	Код	Символ	Код	Символ	Код	Символ	Код	Символ	Код	Символ
0		43	+	86	V	128	А	171	л	214	╓
1	☺	44	,	87	W	129	Б	172	м	215	╫
2	☻	45	-	88	X	130	В	173	н	216	╪
3	♥	46	.	89	Y	131	Г	174	о	217	┘
4	♦	47	/	90	Z	132	Д	175	п	218	┌
5	♣	48	0	91	[	133	Е	176	▒	219	▉
6	♠	49	1	92	\	134	Ж	177	▒	220	▄
7	•	50	2	93	]	135	З	178	▓	221	▌
8	◘	51	3	94	^	136	И	179	│	222	▐
9		52	4	95	_	137	И	180	┤	223	▀
10		52	5	96	`	138	К	181	╡	224	р
11	♂	54	6	97	a	139	Л	182	╢	225	с
12	♀	55	7	98	b	140	М	183	╖	226	т
13		56	8	99	c	141	Н	184	╕	227	у
14	♫	57	9	100	d	142	О	185	╣	228	ф
15	☼	58	:	101	e	143	П	186	║	229	х
16	►	59	;	102	f	144	Р	187	╗	230	ц
17	◄	60	<	103	g	145	С	188	╝	231	ч
18	↕	61	=	104	h	146	Т	189	╜	232	ш
19	‼	62	>	105	i	147	У	190	╛	233	щ
20	¶	63	?	106	j	148	Ф	191	┐	234	ъ
21	§	64	@	107	k	149	Х	192	└	235	ы
22	▬	65	A	108	l	150	Ц	193	┴	236	ь
23	↨	66	B	109	m	151	Ч	194	┬	237	э
24	↑	67	C	110	n	152	Ш	195	├	238	ю
25	↓	68	D	111	o	153	Щ	196	─	239	я
26	→	69	E	112	p	154	Ъ	197	┼	240	Ё
27	←	70	F	113	q	155	Ы	198	╞	241	ё
28	∟	71	G	114	r	156	Ь	199	╟	242	Є
29	↔	72	H	115	s	157	Э	200	╚	243	є
30	▲	73	I	116	t	158	Ю	201	╔	244	І
31	▼	74	J	117	u	159	Я	202	╩	245	і
32		75	K	118	v	160	а	203	╦	246	ў
33	!	76	L	119	w	161	б	204	╠	247	Ў
34	"	77	M	120	x	162	в	205	═	248	°
35	#	78	N	121	y	163	г	206	╬	249	·
36	$	79	O	122	z	164	д	207	╧	250	·
37	%	80	P	123	{	165	е	208	╨	251	√
38	&	81	Q	124	|	166	ж	209	╤	252	№
39	'	82	R	125	}	167	з	210	╥	253	¤
40	(	83	S	126	~	168	и	211	╙	254	■
41	)	84	T	127	⌂	169	й	212	╘	255	
42	*	85	U			170	к	213	╒		

Додаток В. Поширені асимптотичні складності алгоритмів

Складність	Коментар	Приклади
$O(1)$	Сталий час роботи не залежно від розміру задачі	Очікуваний час пошуку в хеші
$O(\log n)$	Логарифмічне зростання — подвоєння розміру задачі збільшує час роботи на сталу величину	Двійковий пошук серед n елементів
$O(n)$	Лінійне зростання — подвоєння розміру задачі подвоїть і необхідний час	Лінійний пошукв серед n елементів
$O(n \log n)$	Лінеаритмічне зростання — подвоєння розміру задачі збільшить необхідний час трохи більше ніж вдвічі	Сортування злиттям або купою n елементів
$O(n^2)$	Квадратичне зростання — подвоєння розміру задачі вчетверо збільшує необхідний час	Елементарні алгор итми сортування

$O(n^3)$	Кубічне зростання — подвоєння розміру задачі збільшує необхідний час у вісім разів	Звичайне множення матриць

Зміст

Вступ 3

1. Системи числення 5

Вступ 5

1.1. Історія виникнення чисел 8

Цікаві факти числа

1.2. Різновиди систем числення 10

1.2.1. Непозиційні системи числення 10

Унарна система числення

Давньоєгипетська система числення

Грецька система числення

Старослов'янська система числення

Недоліки непозиційних систем числення

1.2.2. Змішані системи числення 19

Римська системи числення

1.2.3. Позиційні системи числення 23

Вавилонська система числення

1.3. Системи числення в інформатиці 27

1.3.1. Десяткова система числення 27

Переведення чисел з десяткової системи числення в іншу

1.3.2. Двійкова система числення 34

Арифметичні операції у двійковій системі числення

Переваги та недоліки двійкової системи числення

1.3.3. Вісімкова система числення 43

Арифметичні операції у вісімковій системі числення

Застосування вісімкової системи числення

1.3.4. Шістнадцяткова система числення 51

Застосування шістнадцяткової системи числення

Арифметичні операції у шістнадцятковій системі числення
Питання до розділу 1 63
Завдання до розділу 1 64
Приклади теоретичного індивідуального самостійного завдання 65

2. Інформація та кодування 67
Вступ 67
2.1. Види та властивості інформації 68
2.2. Оброблення інформації 71
2.3. Кодування інформації 74
2.3.1. Двійковий код, формула Хартлі 76
2.3.2. Одиниці вимірювання інформації 79
2.3.3. Стандарти кодування текстової інформації 82
Windows-1251
Unicode
Проблеми при кодуванні текстів
2.3.4. Кодування графічної інформації 86
Растрове кодування графічної інформації
Векторне кодування графічної інформації
2.3.5. Кодування звуку та відео 90
Питання до розділу 92
Завдання до розділу 93

3. Програмне забезпечення 98
Вступ 98
3.1. Базовий рівень 99
3.2. Системний рівень 100
3.2.1. Операційна система 100
Юнікс-подібні ОС
Microsoft Windows
Mac OS X
3.2.2. Інтерфейс ядра операційної системи 102
Інтерфейс командного рядка ОС Windows
Інтерфейс командного рядка Unix-подібних систем

3.3. Службовий рівень 108
Класифікація службових програмних засобів
3.3. Прикладний рівень 110
Класифікація прикладного програмного забезпечення
Питання до розділу 118
Завдання до розділу 118

4. Алгоритми та блок-схеми 119
Вступ 119
4.1. Алгоритм та його властивості 122
4.2. Представлення алгоритмів у вигляді блок-схем .. 125
4.2.1. Послідовне виконання та лінійні алгоритми 128
4.2.2. Розгалуження та алгоритми з розгалуженням 132
Висловлювання та їх застосування в алгоритмах
Найважливіші логічні сполучники
4.2.3. Цикл та циклічні алгоритми 142
4.3. Алгоритми автоматичного оброблення текстів ... 145
Питання до розділу 150
Завдання до розділу 150

Зразки тестових завдань 154
Зразок тестового завдання до розділу «Системи числення» 154
Зразок тестового завдання до розділу «Інформація та мова» 155
Зразок тестового завдання до розділу «Програмне забезпечення» 156
Зразок тестового завдання до розділу «Алгоритми та блок-схеми» 157

Зразки модульних контрольних робіт 159
Зразок завдання на модульну контрольну з тем «Системи числення» та «Інформація та мова» 159

Зразок завдання на модульну контрольну роботу з тем «Програмне забезпечення» та «Алгоритми та блок-схеми» 159

Фрагмент робочої програми «Основи інформатики» 160

Критерії оцінювання теоретичних завдань 161

Критерії оцінювання практичних завдань 163

Схематичне представлення отримання балів студентом за два змістові модулі 164

Правила техніки безпеки і поведінки в комп'ютерному класі 166

Рекомендовані ресурси..... 169

Використані скорочення..... 171

Список використаних джерел 173

Додаток А. Зв'язок між числами десяткової, шістнадцяткової, вісімкової та двійкової систем числення..... 179

Додаток Б. Таблиця ASCII символів..... 181

Додаток В. Поширені асимптотичні складності алгоритмів 182

Навчальне видання

РОССАДА Тетяна Володимирівна
РУСІНА Наталія Геннадіївна
ФЕДОРОВА Марія Вікторівна

ОСНОВИ ІНФОРМАТИКИ

навчально-методичний посібник
для студентів спеціалізації
035.10 Філологія (прикладна лінгвістика)

Редактор ???

Формат 60x84$^{1/16}$. Ум. друк. арк. 14,0. Наклад 300.
Гарнітура Times New Roman. Папір офсетний. Друк офсетний.
Підписано до друку 11.09.2017

Видавець і виготовлювач ???