Міністерство освіти і науки України

Житомирський державний університет імені Івана Франка

Кафедра комп'ютерних наук та інформаційних технологій

Конспект лекцій освітньої компоненти «Теоретичні основи інформатики» для здобувачів другого (магістерського) рівня вищої освіти спеціальності 014 Середня освіта за спеціалізацією 014.09 Середня освіта (Інформатика)

Житомир
Вид-во ЖДУ імені Івана Франка
2023

УДК 004.9(075.8)
К 65

Рекомендовано до друку вченою радою Житомирського державного університету імені Івана Франка
(протокол №19 від 27.10.2023 р.)

Рецензенти:

Міца Олександр – доктор технічних наук, завідувач кафедри інформаційних управляючих систем та технологій ДВНЗ «УжНУ»;

Струтинська Оксана – доктор педагогічних наук, професор кафедри інформаційних технологій і програмування факультету математики, інформатики та фізики Українського державного університету імені Михайла Драгоманова;

Горобець Сергій – кандидат педагогічних наук, доцент кафедри комп'ютерних наук та інформаційних технологій Житомирського державного університету імені Івана Франка.

Конспект лекцій освітньої компоненти «Теоретичні основи інформатики» для здобувачів другого (магістерського) рівня вищої освіти спеціальності 014 Середня освіта за спеціалізацією 014.09 Середня освіта (Інформатика) / Уклад.:, С. С. Жуковський, О. М. Кривонос, Ю. І. Мінгальова, П. Г. Шевчук. – Житомир: Вид-во ЖДУ ім. І. Франка, 2023. – 88 с.

Конспект лекцій освітньої компоненти «Теоретичні основи інформатики» укладено для використання здобувачами другого (магістерського) рівня вищої освіти спеціальності 014 Середня освіта за спеціалізацією 014.09 Середня освіта (Інформатика), викладачами закладів вищої освіти. У методичних вказівках визначено мету та завдання освітньої компоненти. У тексті лекцій викладені основні поняття освітньої компоненти. Лекції подані в науковій та логічній послідовності, охоплюють основні принципи, питання курсу.

ЗМІСТ

Вступ

Програма вивчення освітньої компоненти “Теоретичні основи інформатики” для підготовки здобувачів другого (магістерського) рівня вищої освіти відповідає освітньо-професійній (науковій) програмі Інформатика в закладах освіти.

Предмет вивчення освітньої компоненти є виклад понять теоретичних засад інформатики та їх використання під час вирішення фахових завдань.

Міждисциплінарні зв’язки: «Методика навчання інформатики в профільній школі», «Виробнича практика в закладах середньої та передвищої освіти», «Виконання кваліфікаційної роботи».

Програма навчальної дисципліни складається з таких модулів:

Модуль І. Теорія інформації.

Модуль ІІ. Математичні основи інформатики.

Мета, завдання та результати навчання.

Мета вивчення освітньої компоненти: підготовка здобувачів освіти до ефективного застосування сучасної комп'ютерної техніки та інформаційних систем, формування у студентів знань, умінь та навичок роботи з наявним програмним забезпеченням з метою подальшого ефективного застосування сучасних інформаційних систем, баз і банків даних та їх послідовного застосування в різних середовищах, під час організації навчального процесу, здійснення наукового пошуку, обробки результатів експериментальних досліджень, якісного оформлення науково-методичної документації і оптимального використання робочого часу.

Основними завданнями вивчення освітньої компоненти є:

забезпечити засвоєння здобувачами освіти основних компонент теорії сучасного навчання курсу інформатики та навчити їх застосовувати теоретичні знання для вирішення практичних завдань;

ознайомити здобувачів освіти із сучасними тенденціями навчання інформатики;

розкрити суть складових частин і засобів сучасної методики інформатики як науки; спрямувати здобувачів освіти на творчий пошук під час практичної діяльності у школі;

забезпечити ґрунтовне вивчення шкільних програм, підручників, навчальних і методичних посібників з інформатики, способів використання в навчальному процесі шкільної комп'ютерної техніки і відповідного програмного забезпечення, розуміння методичних ідей використання методів і засобів сучасної інформаційної технології;

виховати у майбутніх вчителів творчий підхід до розв'язування проблем викладання інформатики та використання засобів ІКТ у своїй майбутній діяльності, сформувати знання, вміння і навички, необхідні для самостійного аналізу навчального процесу, дослідження різноманітних методичних проблем та психолого-педагогічних ситуацій, розвинути здатність до постійної

самоосвіти, наукового пошуку шляхів удосконалення процесу навчання елементам інформатики.

Компетентності та програмні результати навчання:

Компетентності

Змістовно освітня компонента спрямована на формування здобувачами вищої освіти здатності розв'язувати складні задачі або проблеми в галузі освіти, що передбачає здійснення інновацій та/або проведення педагогічних досліджень і характеризується невизначеністю умов.

ЗК1. Здатність застосовувати знання у практичних ситуаціях.

ФК1. Здатність до поглиблення знань і розуміння предметної області та професійної діяльності.

ФК 8. Здатність формувати в учнів культуру академічної доброчесності та дотримуватися її принципів у власній професійній діяльності.

ПК1. Здатність розуміти концептуальні засади освіти в галузі інформатики та методики її викладання у закладах освіти, тенденції розвитку інформатики й інформатизації суспільства, використовувати теоретичні знання і практичні вміння щодо формування у здобувачів освіти базових і предметних інформатичних компетентностей.

ПК2. Здатність до поглиблення знань і розуміння предметної області та професійної діяльності.

ПК7. Здатність розуміти інноваційні ІКТ-зорієнтовані педагогічні технології та використовувати їх в навчальному процесі.

Програмні результати навчання

РН1. Демонструє вміння застосовувати знання з психології, педагогіки, фундаментальних і прикладних наук (відповідно до предметної спеціальності) у практичних ситуаціях здійснення освітньої діяльності, поглиблює знання з предметної області.

РН9. Демонструє уміння класифікувати, упорядковувати і узагальнювати навчальний матеріал відповідно до умов навчального процесу, потреб формування ключових компетентностей та інтегрованого навчання.

РН14. Демонструє дотримання культури академічної доброчесності у власній діяльності та демонструє вміння формувати її в учнів.

ПРН2. Демонструє теоретичні знання і практичні вміння щодо формування у здобувачів освіти базових і предметних інформатичних компетентностей.

ПРН5. Володіє вміннями розв’язку задач шкільного курсу інформатики різних профілів і вибіркових модулів, вміє аналізувати та оцінювати ефективність їх розв'язку.

Інформаційний обсяг освітньої компоненти.

Модуль І. Теорія інформації.

Тема 1. Інформація та дані як категорії інформатики. Арифметичні основи комп'ютерної обробки інформації.

Інформатика - теоретична та прикладна наука. Поняття інформації, повідомлення, сигналу, даних. Види та властивості інформації. Якість інформації. Адекватність інформації. Синтаксичні заходи інформації. Семантичний захід інформації. Прагматичний захід інформації. Знакове подання інформації. Операції та обмеження для інформаційних одиниць. Концепція інформаційної технології. Склад процедур інформаційної технології. Зміст інформаційних технологій. Інформаційна система: поняття, структура та склад

Тема 2. Комп'ютерне подання інформації. Логічні засади обробки інформації.

Форми подання числової інформації Подання цілого числа Подання речового числа Подання символьної інформації Подання графічної інформації Подання звукової інформації

Алгебра логіки Функції та формули логіки Структурні формули Аналіз та синтез цифрових схем Функціонально повні системи булевих функцій Мінімальний логічний базис Основні поняття логіки предикатів

Тема 3. Виявлення та корекція помилок у переданій інформації.

Причини виникнення та типи помилок Способи захисту від помилок Побудова коригувального коду Методи виявлення помилок Методи корекції помилок

Модуль ІІ. Математичні основи інформатики.

Тема 4. Основи теорії множин. Основні поняття теорії графів.

Множини та операції над ними Відносини на множинах Поняття лінгвістичної змінної та нечіткої множини Види функцій власності Операції над нечіткими множинами Нечіткі висловлювання

Тема 5. Елементи теорії алгоритмів та формальних мов.

Інтуїтивне поняття алгоритму Необхідність формалізації інтуїтивного поняття алгоритму. Поняття алгоритмічної системи Зведення алгоритмів до числових функцій. Поняття обчислюваної функції Поняття про формальні мови та породжувальні граматики Машина Тьюринга Алгоритмічна роздільність

Форми представлення алгоритмів Базові алгоритмічні структури Структурне проектування Об'єктно-орієнтоване проектування Мови програмування Підготовка програми до виконання

Тема 6. Основи інформаційного моделювання предметної галузі.

Категорії предметної галузі Багаторівнева система моделювання предметної галузі Інформаційний опис об'єктів предметної галузі Подання інфологічної моделі предметної області ER-діаграмами.

Модуль І. Теорія інформації.

Тема 1. Інформація та дані як категорії інформатики. Арифметичні основи комп'ютерної обробки інформації.

Мета: надати студентам загальне уявлення про інформаційні процеси, такі як збір, зберігання, обробка та передача інформації, ознайомити студентів із основними поняттями та категоріями інформатики, такими як інформація, дані, знання, інформаційні процеси, розглянути арифметичні основи комп'ютерної обробки інформації, такі як арифметичні операції, арифметичні типи даних, арифметичні коди.

План:

- Інформатика - теоретична та прикладна наука.
- Поняття інформації, повідомлення, сигналу, даних.
- Види та властивості інформації.
- Якість інформації. Адекватність інформації.
- Синтаксичні заходи інформації. Семантичний захід інформації. Прагматичний захід інформації.
- Знакове подання інформації. Операції та обмеження для інформаційних одиниць.
- Концепція інформаційної технології.
- Склад процедур інформаційної технології.
- Зміст інформаційних технологій.
- Інформаційна система: поняття, структура та склад

Поняття інформації та даних

Інформація - це дані, які оброблені, упорядковані, структуровані або представлені таким чином, щоб вони були корисними при використанні в певному контексті. Веб-сайт Computer Hope, наприклад, надає величезну кількість інформації про комп’ютери, яку кожен може прочитати, щоб дізнатися більше про них і про комп’ютерні теми.

Інформатика займається процесом зберігання, пошуку та використання інформації. Чарльз Беббідж винайшов електронний пристрій, який згодом став відомим як комп'ютер. Здатність комп'ютерів обробляти величезні обсяги даних і видавати точні результати вражає. Блоки введення, обробки та виведення даних складають три основні частини комп'ютера. В інформатиці інформація часто зберігається в комп'ютерній системі у вигляді даних.

Інформація визначається як впорядковані дані, які мають певну значущу цінність для користувача. Інформація також є обробленими даними, які використовуються для прийняття рішень і вжиття заходів. Оброблені дані мають відповідати наступним критеріям, щоб мати суттєве значення для прийняття рішень:

Точність: інформація має бути точною.

Повнота: інформація має бути повною.

Своєчасність: інформація має бути доступною, коли вона потрібна [2].

З моменту винаходу комп'ютерів люди використовували термін «дані» для позначення комп'ютерної інформації, і ця інформація або передавалася, або зберігалася. Але це не єдине визначення даних; існують і інші типи даних. Отже, які дані? Даними можуть бути тексти чи числа, написані на папері, або це можуть бути байти та біти в пам'яті електронних пристроїв, або це можуть бути факти, які зберігаються у свідомості людини.

Дані — це набір фактів або цифр, які можуть бути оброблені комп'ютером. Вони можуть бути в будь-якій формі, як-от число, текст, зображення чи звук. Дані можна обробляти кількома способами, наприклад, шляхом організації, класифікації, аналізу та інтерпретації [4].

Дані в лінійному форматі можуть бути представлені цифрами 0-9, символами -,-,* і так далі. Кожен рядок даних містить окремий символ і читається зліва направо. Дати можна форматувати різними способами, зокрема ДД/ММ/РРРР. Можна зрозуміти, як генеруються графіка та звук, використовуючи символи.JPG,.wav та.mp3.

Коли дані вводяться в комп’ютер, вони спочатку перетворюються в числовий формат. У випадку комп’ютера це виконується процесором, який розбиває дані на окремі біти. Комп’ютер аналізує та аналізує дані, які надходять до нього після аналізу тексту. Для того, щоб визначити значення кожної складовлї, вони повинні спочатку їх ідентифікувати. Подібним чином символ «А» може представляти букву «а», як у букву «д». У цьому процесі дуже важливо визначити символи та зрозуміти їх значення. Зображення може бути представлене символом.JPG, тоді як звуковий файл може бути представлено символом.wav.

Або наприклад, дані дати аналізуються за допомогою комп’ютера за певним алгоритмом. Необхідно з’ясувати, як було введено дату й час, а також країну, часовий пояс і кількість годин різниці між координатами.

Дані можна інтерпретувати різними способами. Дані, наприклад, можуть бути представлені різними способами, такими як графік або таблиця. Тому його значення можна тлумачити по-різному. Наприклад, статистику можна використовувати для представлення даних, тоді як тенденцію можна використовувати для представлення даних [1].

Різниця між інформацією та даними подана в таблиці:

ДАНІ	ІНФОРМАЦІЯ
Дані – це змінні, які допомагають розвивати ідеї/висновки.	Інформація – це значущі дані.
Даними є текстові та числові значення.	Інформація – це уточнена форма фактичних даних.
Дані не залежать від інформації.	Тоді як інформація покладається на дані.
Біти та байти є одиницею вимірювання даних.	Інформація вимірюється в таких значущих одиницях, як час, кількість тощо.

ДАНІ	ІНФОРМАЦІЯ
Дані можна легко структурувати таким чином: 1. Табличні дані 2. Графік 3. Дерево даних	Інформація також може бути структурована так: 1. Мова 2. Ідеї 3. Думки
Дані не мають конкретного призначення	Інформація надана шляхом інтерпретації даних.
Це знання низького рівня.	Це більш високий рівень знань.
Дані безпосередньо не допомагають у прийнятті рішень.	Інформація безпосередньо допомагає у прийнятті рішень.
Дані - це сукупність фактів, які самі по собі не мають значення.	Інформація ставить ці факти в контекст.
Прикладом даних є тестовий бал студента.	Прикладом інформації є середній бал групи, отриманий на основі даних.

Системи управління базами даних та їх переваги та властивості

Системи управління базами даних (Database Management System, DBMS) — це програмне забезпечення, яке дозволяє зберігати, отримувати та аналізувати дані. Системи керування базами даних — це платформа, яка підключає кінцевого користувача до бази даних і дозволяє йому створювати, читати, оновлювати та видаляти дані з бази даних [1].

DBMS має наступні переваги і властивості.

Рішення для управління даними дозволяє впорядковувати, зберігати та отримувати доступ до даних у спосіб, який полегшує їх пошук і використання.

Користувач має повний доступ до бази даних і може їх запитувати.

Цілісність даних забезпечується, якщо дані в базі даних є точними та актуальними.

Безпека даних означає захист бази даних від несанкціонованого доступу.

Обробка даних — це процес аналізу, введення та отримання даних у програмі даних.

Інфраструктура даних будь-якої організації була б неповною без DBMS. Користувачі можуть отримувати доступ до даних і керувати ними таким чином, щоб їх було легше співпрацювати та розуміти, що є величезною перевагою. DBMS може скоротити час, необхідний для обробки та розуміння даних, що призведе до підвищення ефективності бізнесу та результатів за рахунок спрощення управління даними та доступу до них.

Типи даних

Тип даних — це формальна класифікація даних, які зберігаються або обробляються в програмі. Типи даних важливі, оскільки вони визначають операції, які можна виконувати з даними. Наприклад, можна поділити одне число на інше, але не можна поділити слово на інше слово.

У більшості мов програмування всі загальні типи даних вбудовані в мову. Ці типи даних називають базовими або простими типами даних. Існує чотири базові типи даних, з яких можна створити всі інші. Вони відомі як int цілі (цілі числа), float **дійсні** (числа з дробовою частиною), bool **логічні** (істинно/хибно) і char **символьні** (символи). Окрім того, бувають double подвійної точності (дійсний з подвійною точністю), string рядок та void (порожній, не має значення) [4].

Тип даних	опис	приклад
Int *цілі*	Ціле число — це ціле число. Числа можуть бути позитивними або негативними, але вони повинні бути цілими. Це означає, що вони не мають десяткових знаків.	-5, 123, 0
Float *дійсні*	Real також називають float. До дійсних чисел належать числа з десятковими знаками.	1,1, -1,0, 382,0, 12,125
Bool *логічні*	Логічний тип даних базується на логіці та може мати лише одне з двох значень: True або False.	Правда/Неправда
Char *символьні*	Char відноситься до одного символу. Символом може бути будь-яка літера, цифра або символ.	в, A, X, £
String *рядок*	Рядковий тип даних відноситься до тексту. Рядок — це набір символів і включає пробіли,	Привіт, 1$34A, дев'яносто

Тип даних	опис	приклад
	знаки пунктуації, цифри та будь-які інші символи, такі як $, &, £ тощо. Рядки також можуть містити цифри, але вони розглядатимуться як текстові символи, а не числа.	чотири
Void порожні	Нульове значення використовується для позначення «нічого»; відсутність будь-якої цінності, будь-якого типу даних.	Null

Більшість мов також матимуть додаткові вбудовані типи даних, деякі з яких будуть складеними типами даних. Складені типи даних — це набори, що складаються з простих типів, наприклад, масив цілих чисел. Якщо використовується систему управління базами даних, то відбувається доступ до більшої кількості вбудованих типів даних. Наприклад, ви можете вказати, що поле буде містити позначку дати/часу, зазвичай у форматі РРРР-ММ-ДД гг:хх:сс.

Обсяг пам’яті, виділеної для кожного типу даних, буде різним залежно від системи, яка використовується для визначення даних. Різні системи можуть мати різні стандарти; однак, через необхідність передачі даних між системами, існує стандартна специфікація для кожного типу даних [4].

Арифметичні основи комп’ютерної обробки інформації. Збереження даних

Математика є джерелом двох ключових концепцій у розвитку комп’ютера — ідеї, що всю інформацію можна представити у вигляді послідовностей нулів і одиниць, і абстрактного поняття «збережена програма .» У двійковій системі числення числа представлені послідовністю двійкових цифр 0 і 1 так само, як числа у звичній десятковій системі представлені за допомогою цифр від 0 до 9. Відносна легкість, з якою два стани (наприклад, високий і низька напруга) можуть бути реалізовані в електричних і електронних пристроях, які природним чином призводять до двійковий розряд , або біт, став основною одиницею зберігання та передачі даних у комп’ютерній системі .

Комп’ютери представляють дані (наприклад, текст, зображення, звук, відео) у вигляді двійкових значень, які містять два числа: 1 і 0. Найменша одиниця даних називається «біт», і вона представляє одне значення. Крім того,

байт має вісім біт. Пам'ять і зберігання вимірюються в таких одиницях, як мегабайти, гігабайти, терабайти, петабайти та ексабайти. Науковці даних продовжують придумувати все новіші, масштабніші вимірювання даних, оскільки обсяг даних, які генерує наше суспільство, продовжує зростати [2].

У наведеній нижче таблиці показано типову пам’ять, виділену для кожного з типових типів даних.

Тип даних	Розподіл пам'яті	Діапазон
Цілі типи		
Булеві	1 біт (зберігається як 1 байт)	0 до 1
Байт	8 бітів	0 до 255
Слово	2 байти	0 до 65535
Подвійне слово	4 байти	0 до 4,294,967,295
Ціле число	4 байта	−2,147,483,648 до 2,147,483,647
Подвійне ціле	8 байтів	−9,223,372,036,854,775,808 до 9,223,372,036,854,775,807
Дійсні типи		
Дійсне	4 байти	1E-37 до 1E+37 (6 десяткових цифр)
Подвійної точності	8 байтів	1E-307 до 1E+308 (15 десяткових цифр)

Дані можна зберігати у форматах файлів за допомогою систем мейнфреймів, таких як ISAM і VSAM, хоча існують інші формати файлів для перетворення, обробки та зберігання даних, як-от значення, розділені комами. Ці формати даних наразі використовуються в широкому діапазоні типів машин, незважаючи на те, що більш структуровані підходи, орієнтовані на дані, набувають більшого поширення в сучасному світі IT.

Сфера зберігання даних зазнала більшої спеціалізації, оскільки база даних, система управління базами даних і нещодавно технологія реляційної бази даних, що з'явилися нещодавно, надали нові способи організації інформації [6].

Цикл та способи обробки даних

Обробка даних визначається як перевпорядкування або реструктуризація даних людьми або машинами, щоб підвищити їхню корисність і додати цінність для певної функції чи цілей. Стандартна обробка даних складається з трьох основних етапів: введення, обробки та виходу. Разом ці три етапи складають цикл обробки даних.

Вхід: вхідні дані готуються для обробки в зручній формі, що залежить від машини, яка виконує обробку.

Обробка: форма вхідних даних змінюється на більш корисну. Наприклад, інформація з табелів використовується для розрахунку зарплати.

Вихід: на останньому етапі результати обробки збираються як вихідні дані, остаточна форма яких залежить від того, для чого вони використовуються.

Наприклад, вихідні дані стають фактичними зарплатами працівників.

Є два основних способи аналізу даних:

Аналіз даних у якісних дослідженнях

Аналіз даних у кількісних дослідженнях [5].

Аналіз даних у якісних дослідженнях

Аналіз даних і дослідження суб'єктивної інформації працюють дещо краще, ніж числова інформація, оскільки якість інформації складається зі слів, зображень, зображень, об'єктів. Отримання знань із таких заплутаних даних є складним завданням, тому його зазвичай використовують для пошукових досліджень на додаток до аналізу даних. Хоча існує кілька різних способів виявлення шаблонів у друкованих даних, стратегія на основі слів є найбільш широко використовуваним глобальним методом дослідження та аналізу даних. Важливо, що процес аналізу даних у якісних дослідженнях є ручним. Тут фахівці, як правило, читають доступну інформацію і знаходять повторювані або часто вживані слова.

Аналіз даних у кількісних дослідженнях

Основний етап дослідження та аналізу даних полягає в тому, щоб зробити це для експертизи з метою перетворення інформації. Підготовка даних включає в себе наступне.

Перевірка даних

Редагування даних

Кодування даних

Підприємства, які готові працювати в сучасному надконкурентному світі, повинні мати надзвичайну здатність досліджувати складну дослідницьку інформацію, виводити варті уваги фрагменти знань і пристосовуватися до нових потреб ринку.

Інформація та дані є основними категорії інформатики. В інформатиці дані є основою обробки інформації.

Дані та інформація часто використовуються як взаємозамінні, але вони мають різні значення. Дані стосуються необроблених фактів і цифр, таких як числа, вимірювання або спостереження. Вони часто представлені у структурованому або напівструктурованому форматі та не мають внутрішнього значення чи контексту.

З іншого боку, інформація – це дані, які були оброблені, упорядковані та яким надано значення та контекст. Вона відповідає на такі питання, як хто, що, коли, де, чому і як. Інформацію можна використовувати для прийняття рішень або отримання знань.

Тип даних — це формальна класифікація даних, які зберігаються або обробляються в програмі. Загальні типи даних називають базовими або простими типами даних. Це int цілі (цілі числа), float **дійсні** (числа з дробовою частиною), bool **логічні** (істинно/хибно) і char **символьні** (символи), double подвійної точності (дійсний з подвійною точністю), string рядок та void (порожній, не має значення).

Комп’ютери представляють дані, зокрема текст, зображення, звук, відео у вигляді двійкових значень, які містять два числа: 1 і 0. Найменша одиниця даних називається «біт», і вона представляє одне значення. Байт має вісім біт.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Завадський І.О. Основи баз даних: Навч.посіб./. – К.: Видавець І.О. Завадський, 2011. – 192 с.

2. Інформатика. Базовий курс. навчальний посібник у 3-х частинах. Частина 3/Сальнікова І.І., Шестопалов Є.А. – Шепетівка: «Аспект», 2005. – 160 с.

3. Інформатика. Навчальна програма для 10-11 класів загальноосвітніх нав-чальних закладів. Академічний рівень.

4. Інформатика: 11 кл.: підруч. для загальноосвіт. навч. закл.: рівень стандарту / Й.Я. Ривкінд, Т.І. Лисенко, Л.А. Чернікова, В.В. Шакотько; за заг. ред. М.З. Згуровського. – К.: Генеза, 2011. – 304 с.

5. Організація баз даних та знань: конспект лекцій для студентів заочної форми навчання / Укладач А.В. Неня. - Суми: Вид-во СумДУ, 2010. - 109 с.

6. Теорія інформації та кодування / В. С. Василенко, О. Я. Матов; НАН України, Ін-т пробл. реєстрації інформації. — Київ : ІПРІ НАН України, 2014. — 439 с.

7. Чаповська Р.Б. Access’2000 для початківця. Посібник з інформатики. Книга 9. 2004 – 96 с.

Питання для самоконтролю

- Що таке інформація?
- Які основні властивості інформації?
- Які типи інформації існують?
- Що таке дані?
- Які основні типи даних існують?
- Які основні арифметичні операції існують?
- Які основні арифметичні типи даних існують?
- Як кодуються числа в комп'ютері?
- Яка різниця між інформацією та даними?
- Які основні етапи обробки інформації?
- Які основні методи обробки інформації?
- Які основні пристрої для зберігання інформації?
- Які основні пристрої для обробки інформації?

- Які основні пристрої для передачі інформації?

Тема 2. Комп'ютерне подання інформації. Логічні засади обробки інформації.

Мета: сформувати у студентів системні уявлення про інформаційні процеси та арифметичні основи комп'ютерної обробки інформації, розвивати у студентів логічне мислення та вміння застосовувати отримані знання на практиці, прищеплювати студентам інтерес до інформатики та її застосування в різних сферах людської діяльності.

План:

- Форми подання числової інформації
- Подання цілого числа. Подання речового числа. Подання символьної інформації. Подання графічної інформації. Подання звукової інформації.
- Алгебра логіки. Функції та формули логіки.
- Структурні формули. Аналіз та синтез цифрових схем.
- Функціонально повні системи булевих функцій.
- Мінімальний логічний базис. Основні поняття логіки предикатів.

Форми подання числової інформації

Подання інформації на комп’ютерах залежить від різних типів програмного забезпечення, і користувач має вирішити, яке програмне забезпечення найкраще підходить для його потреб і проекту, враховуючи аудиторію, на яку він розрахований. Вимогою до подання інформаціє є чітке представлення у найбільш відповідний спосіб. Макет, дизайн і формат повинні відповідати меті та аудиторії. Можливі такі формати комп’ютерного подання інформації: есе, листівка, плакат, веб-сайт, презентація (слайд-шоу) [5].

До кожного з перерахованих вище можна включити один або всі структурні елементи: текст, зображення та фото, графік або діаграма, аудіо, відео, анімація[5]. Останні три не можна використовувати в друкованому вигляді, але використання цифрових листівок, інформаційних бюлетенів і плакатів (постерів) є звичайною практикою.

У таблиці нижче подано варіанти щодо використання програмного забезпечення для подання інформації залежно від потреби.

Обраний підхід	**Подання інформації**	**Приклади додатків**
Написати твір	Обробка текстів	Microsoft Word; Open Office Writer; Сторінки iWorks
Створіть листівку чи плакат	Настільна публікація	Scribus; Studio Publisher; Microsoft Publisher
Зробіть презентацію або слайд-шоу	Презентація	Works Keynote; Microsoft PowerPoint; Open Office Impress
Створіть графік або діаграму	Електронна таблиця	Microsoft Excel; Open Office Calc;
Створіть веб-сайт	Веб дизайн	Google Sites; Adobe Dreamweaver; Serif WebPlus
Зніміть і змонтуйте відео чи короткий фільм	Редагування відео	Adobe Premiere; Windows Movie Maker
Створіть анімацію	Анімація	Adobe Flash; Microsoft Silverlight; Олівець
Показувати відредаговані зображення або фотографії	Малювати/розфарбовувати	Windows Paint; Adobe Photoshop; Gimp

Характеристика основних типів подання даних

Основними типами комп'ютерного подання інформації є *текстовий, табличний, діаграмний*[4].

Текстовий тип

Подаючи дані текстовим варіантом, ми використовуємо слова для опису зв'язку між інформацією. Текстове представлення дозволяє обмінюватися інформацією, яку неможливо відобразити на графіку. Прикладом даних, які можна представити в текстовому вигляді, є результати дослідження. Коли науковець хоче надати додатковий контекст або пояснення у своїй презентації,

він може вибрати цей формат, оскільки в тексті інформація може виглядати більш зрозумілою.

Текстове подання є звичайним для обміну результатами, думками та представлення нових ідей. Це подання інформації містить лише речення та слова, а не таблиці чи графіки для відображення даних.

Табличний тип

Табличне подання використовує таблицю для обробки великої кількості інформації. Використовуючи цей метод, ми впорядковуємо дані в рядках і стовпцях відповідно до характеристик даних. Табличне представлення корисно для порівняння даних і допомагає візуалізувати інформацію. Дослідники використовують табличний тип подання інформації наступним чином:

Якісна класифікація: якості, включаючи національність, вік, соціальний статус, зовнішній вигляд і риси особистості, можуть подаватися в таблиці для перегляду та порівняння соціологічної та психологічної інформації.

Кількісна класифікація: ця категорія включає елементи, які можна порахувати або пронумерувати.

Просторова класифікація: стосується ситуацій, коли інформація використовує базу розташування, наприклад дані про місто, штат або регіон.

Часова класифікація: час є змінною категорією, тому будь-яка міра часу, включаючи секунди, години, дні або тижні, може допомогти класифікувати дані [2].

Переваги використання таблиці для обміну даними полягають у тому, що вона спрощує дані, роблячи їх легко доступними для глядачів, допомагає порівнювати вибрані змінні та може заощадити місце у презентації, оскільки таблиця систематизує інформацію.

Діаграмний тип

Цей спосіб відображення даних використовує діаграми та зображення. Це найбільш наочний тип представлення даних і забезпечує швидкий огляд статистичних даних. Існує чотири основні типи діаграм, зокрема:

Піктограми: ця діаграма використовує зображення для подання даних. Наприклад, щоб показати кількість книг, проданих протягом першого тижня випуску, ми можемо намалювати п’ять книг, де кожне зображення відповідає 1000 книг, якщо споживачі купили 5000 книг.

Картограми: включає будь-які типи карт, на яких показано місцезнаходження людини, місця чи об’єкта. Наприклад, картограми допомагають орієнтуватися в тематичних парках, щоб ми могли через додатки знайти атракціони, продуктові та сувенірні магазини.

Гістограми: у цьому типі використовуються прямокутники різних розмірів на осях X і Y для представлення різних обсягів у наборі даних. Він зображує числові значення та використовує прямокутники для відображення даних для змінних при аналізі даних.

Кругові діаграми: у діаграмах цього типу дані відображаються у вигляді дробів у колі. Це відображає будь-який тип числових даних, але оптимально працює з меншою кількістю змінних [1].

Оскільки діаграмні більш наочні, ніж інші типи подання даних, вони можуть надати більше інформації про зв’язки між змінними категоріями в наборі даних. Наприклад, гістограма може відображати дані за кольором і розміром прямокутника, а також використовувати більш розширену гістограму для обміну даними з кількох змінних у часі. Схематичне представлення також допомагає швидко читати дані та забезпечує легке порівняння.

Візуалізація даних як перевага комп’ютерного подання інформації

Візуалізація даних полягає в практиці подання даних у вигляді графіків, зображень і таблиць, інфографіки та інших шаблонів [2]. Графічне представлення даних допомагає донести інформацію із контекстом, термінами та перспективами використання представлених даних. Використовуючи технології візуалізації даних, аналітик вводить набір даних у програмне забезпечення та використовує різні інструменти візуалізації для перетворення даних у графіку. Ефективна візуалізація даних може допомогти досягати різних

цілей, а саме відображення змін, розуміння розподілу даних, порівняння значень, спостереження за зв’язками, виокремлення пріоритетного тощо.

Візуалізація даних:

робить дані більш доступними: візуалізація даних може допомогти краще передати важливу інформацію про дані. Ми можемо повідомляти інформацію людям, які мало або зовсім не знайомі зі статистикою та аналізом даних.

уточнює складні дані: знайти тенденції у великих наборах даних може бути складно, але не з візуалізацією. Це допомагає розбити складні дані на просту для розуміння інформацію.

аналізує джерела: візуалізація даних бере інформацію з різних джерел і надає інформацію, яка допомагає аудиторії уникнути нерелевантної інформації, поглинаючи релевантну інформацію. Це допомагає надавати інформацію з різних джерел, відображаючи дані на різних діаграмах, тим самим розвиваючи критичне мислення.

підсилює переконливі аргументи: використання даних може бути корисним, щоб виділити візуально переконливі та важливі аргументи.

покращує процес прийняття рішень: хоча візуалізація даних є трудомісткою технікою, вона може допомогти прийняти важливі рішення та прискорити процес їх прийняття.

дозволяє знаходити помилки: візуалізація даних може допомогти виявити помилки в наборі даних та видалити їх з аналізу.

Види візуалізації даних

Існує декілька основних типів візуалізації даних [1].

Ієрархічна візуалізація даних

Ця техніка візуалізації допомагає організувати дані у групі. Наприклад, можна створювати гілки інформації та кластери, які походять від початкового елемента. Одним із найпоширеніших типів візуалізації ієрархічних даних є система файлів і папок на комп’ютері. Серед поширених типів ієрархічних інструментів виділяють:

дерева рішень

деревоподібні діаграми

блок-схеми

діаграми сонячних променів

Лінійна візуалізація даних

Така візуалізація є одновимірною. Наприклад, дані мають час початку та закінчення, і точки даних можуть збігатися. Широко використовують цю техніку для візуального бухгалтерського звітування про прибутки та збитки. Деякі поширені типи лінійної візуалізації:

стовпчасті діаграми

лінійні діаграми

послідовність часових рядів

діаграми полярних площ

хронологія

діаграми розсіювання

діаграми Ганта

Візуалізація мережевих даних

Візуалізація мережевих даних показує зв’язок між різними вузлами та об’єктами. Однією з основних цілей є визначення найшвидшого шляху між групами та двома елементами. Деякі приклади візуалізації мережевих даних:

діаграми вузол-ланка

хмари слів

матричні діаграми

графік залежності

Візуалізація геопросторових даних

Візуалізація геопросторових або просторових чи площинних даних дозволяє створити тісний зв’язок із зібраними даними та місцевістю. Підприємства та організації можуть використовувати візуалізацію геопросторових даних для відображення інформації про виборців на карті під час політичної кампанії. Крім того, підприємства використовують його для представлення даних про продажі в певних регіонах. Ця технологія дозволяє

накладає карти на точки даних. Деякі приклади візуалізації геопросторових даних:

картограма

теплові карти

карти потоку

карти щільності

Багатовимірна візуалізація даних

Багатовимірні графіки та діаграми – це тривимірні діаграми, які використовують кілька одночасних змінних даних для їх класифікації. Це найбільш яскравий і привабливий тип візуалізації. Деякі приклади візуалізації багатовимірних даних:

діаграми Венна

секторні діаграми

гістограми з накопиченням

діаграми кроків

водоспадні діаграми

Логічні засади обробки інформації

Логічні засади обробки інформації спираються на розділ математичної логіки, який розглядає пропозиції з точки зору їх логічних значень (істинних чи хибних) і логічних операцій над ними.

Алгебра логіки виникла в середині XIX століття з досліджень Г. Буля, а згодом була розвинена К. С. Пірсом, П. С. Порецьким, Б. Расселом, Д. Гільбертом та ін. Розвиток алгебри логіки був спробою вирішення традиційних логічних задач алгебраїчними методами. З народженням теорії множин у 1870-х роках судження і логічні операції над ними стали основним предметом алгебри логіки.

Під пропозиціями розуміють твердження, щодо яких має сенс запитати, чи є вони істинними чи хибними. Наприклад, твердження «кит — тварина» є істинним, а твердження «всі кути прямі» — хибним. Сполучники «і» , «або» , «якщо … то», «еквівалентно», частка «не» тощо, які зазвичай

використовуються в мові логіки, дають змогу побудувати нові, більш «складні», пропозиції з уже встановлених. Таким чином, враховуючи, що "x> 2" і "x≤ 3" , можна отримати, використовуючи сполучник "і" , судження "x> 2 і x≤ 3" ; за допомогою сполучника «або» можна отримати речення «x> 2 або x≤ 3» тощо [6].

Істинність або хибність отриманих таким чином пропозицій залежить від істинності чи хибності початкових пропозицій і від відповідного трактування сполучників як операцій над пропозиціями. Часто істинне символізують цифрою «1», а хибне — цифрою «0». Сполучники «і» , «або» , «якщо … то» , «еквівалентно» позначаються відповідними символами:

& (сполучник),

VV(диз'юнкція),

→→(наслідок),

∼∼(еквівалентність);

¯ ¯ над символом (заперечення) [7].

Основні логічні операції

Який би складний не був логічний зв’язок поміж логічною функцією та її аргументами, його завжди можна представити набором елементарних логічних операцій. Основними логічними операціями є заперечення (операція НЕ, інверсія), диз’юнкція (операція АБО (OR), логічне додавання) і кон’юнкція (операція І (AND), логічне множення) [6].

Запереченням (інверсією, операцією НЕ) називається такий зв’язок між вхідною логічною змінною X і вихідною логічною змінною Y, при якому Y правдиве тільки тоді, коли X хибне, і, навпаки, Y хибне тоді, коли X правдиве. За допомогою логіко-математичної символіки логічна функція Y записується як і читається “Y не є X”. Y=X

Диз’юнкцією (логічним додаванням, операцією АБО) декількох змінних називається така функція, яка хибна тільки тоді, коли одночасно хибні усі аргументи (доданки, вхідні змінні). Операція логічного додавання позначається

знаком + або символом ∨. Наприклад, операція АБО між двома змінними X1 і X2 записується Y=X1 ∨ X2 чи X1+X2 і читається: “Y є X1 або X2”.

Кон’юнкцією (логічне множення, операція І) декількох змінних називається така функція, яка справедлива тільки тоді, коли одночасно справедливі усі вхідні змінні (аргументи). Операція логічного множення (І) позначається знаком математичного множення, тобто крапкою, яку можна не писати, або символом . Наприклад, операція І між двома змінними X1 і X2 записується або Y = X1·X2 = X1X2 і читається: “Y є X1 і X2”. ∧YXX12=^

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Бабич М.П., Жуков І.А. Комп’ютерна схемотехніка: Навчальний посібник. К.: “МК-Прес”, 2004. 412 с.
2. Буйницька О. П. Інформаційні технології та технічні засоби навчання: Навчальний посібник. К: Центр учбової літератури, 2012, 240с.
3. Вітюк О. В., Гуралюк А. Г. , Москалькова Н. М., Шикова О. М. Основи інформатики. К.: МАУП, 2005. Дибкова Л.М. Інформатика і комп"ютерна техніка: Навч. пос. К.: Академія, 2005.416с.
4. Інформатика та комп'ютерна техніка/ М.В.Макарова, Г.В.Карнаухова, С.В.Запара. Суми: Університетська книга, 2008.
5. Інформатика: 11 кл.: підруч. для загальноосвіт. навч. закл.: рівень стандарту / Й.Я. Ривкінд, Т.І. Лисенко, Л.А. Чернікова, В.В. Шакотько; за заг. ред. М.З. Згуровського. К.: Генеза, 2011. 304 с.
6. Спінул Л.Ю., Святненко В.А. Основи цифрової електроніки курс лекцій: Навчальний посібник. К.:КПІ ім. Ігоря Сікорського, 2022, с.118
7. Теорія інформації та кодування / В. С. Василенко, О. Я. Матов; НАН України, Ін-т пробл. реєстрації інформації. К. : ІПРІ НАН України, 2014. 439 с.

Питання для самоконтролю

- Які основні типи кодування інформації?
- Яка різниця між аналоговим і цифровим кодуванням інформації?
- Які основні логічні операції існують?

- Які основні логічні функції існують?
- Як будуються логічні схеми?
- Яка різниця між аналоговим і цифровим поданням інформації?
- Які переваги цифрового подання інформації?
- Які основні поняття логіки?
- Які основні закони логіки?
- Як будуються логічні таблиці?

Тема 3. Виявлення та корекція помилок у переданій інформації.

Мета: надати студентам загальне уявлення про помилки у переданій інформації, їх причини та наслідки, розглянути методи виявлення та корекції помилок, які використовуються в комп'ютерних системах.

План:

- Причини виникнення та типи помилок.
- Способи захисту від помилок.
- Побудова коригувального коду.
- Методи виявлення помилок.
- Методи корекції помилок.

Помилка - це стан, коли інформація одержувача не збігається з інформацією відправника. Помилки виникають в двійкових даних, що передаються від відправника до одержувача, через шум під час передачі. Помилка може бути однобітною, багатобітовою або пакетною. Методи виявлення помилок використовуються, щоб перевірити, чи приймач отримав правильні дані чи пошкоджені. А корекція помилок використовується для виправлення виявлених помилок під час передачі даних від відправника до одержувача [3].

Під час передачі в двійкові дані, надіслані від відправника до одержувача, вносяться помилки через шум під час передачі. Це означає, що біт зі значенням 0 може змінитися на 1 , а біт зі значенням 1 може змінитися на 0 .

Типи помилок, які виникають під час передачі даних, наведено нижче.

Однобітна помилка

Як правило, лише один біт отриманого пошкоджений, і пошкоджений біт може бути розташований у будь-якому місці кадру[3].

На рисунку 1 представлена однобітна помилка.

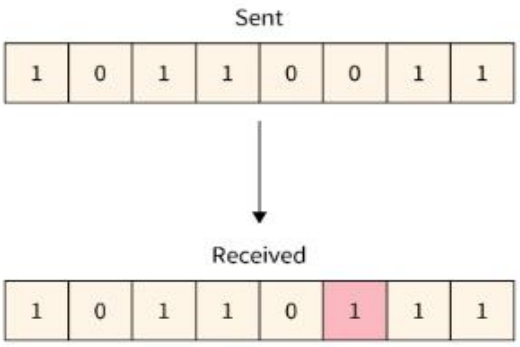


Рисунок 1. Однобітна помилка

Багатобітна помилка

Якщо більш ніж один біт, що отриманий , виявлено пошкодженим, це вважається багатобітною помилкою [3].

На рисунку 2 зображено багатобітну помилку

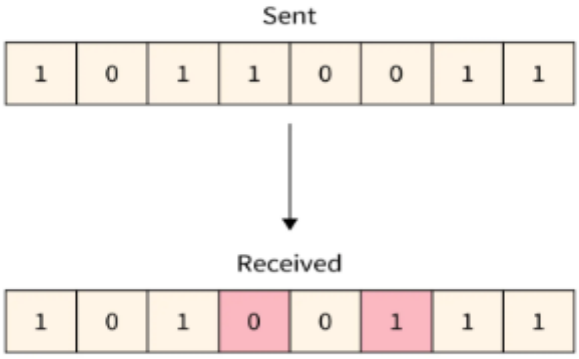


Рисунок 2. Багатобітна помилка

Помилка вибуху

Якщо більш ніж один послідовний біт пошкоджено в отриманому, це вважається помилкою вибуху [3].

На рисунку 3 представлено помилку пакетного біта

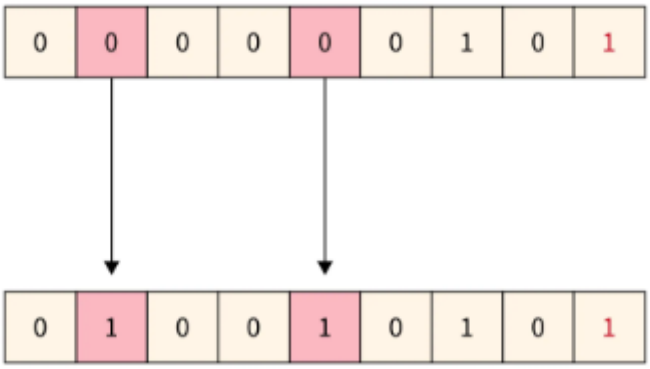
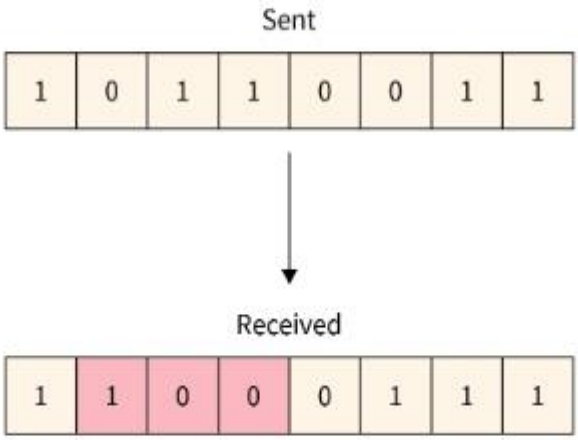


Рисунок 3. Помилка вибуху



Виявлення помилок

Для виявлення помилок використовуються такі методи, як *проста перевірка парності, перевірка двовимірної парності, метод контрольної суми, перевірка циклічної надлишковості* [5].

Проста перевірка парності

Дані, надіслані відправником, проходять перевірку парності, а саме:

1 додається як біт парності до блоку даних, якщо блок даних має непарну кількість одиниць.

0 додається як біт парності до блоку даних, якщо блок даних містить парну кількість одиниць.

Ця процедура використовується для парного числа одиниць. Така перевірка отримала назву перевірки парності [1].

Метод простої перевірки парності зображено на рисунку 4

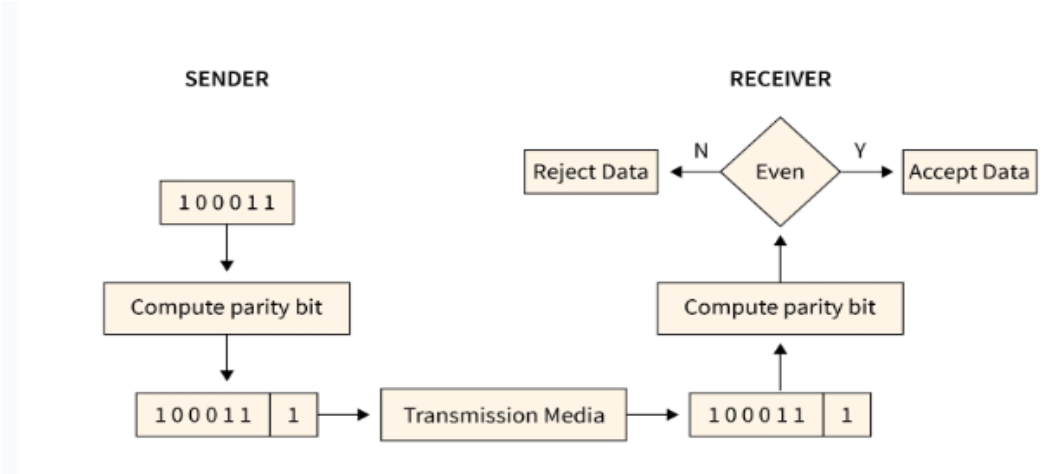


Рисунок 4. Метод простої перевірки

Недоліком цього методу є те, що він виявляє лише однобітну помилку, і йому не вдається виявити багатобітну помилку. А також він не може виявити помилку у випадку помилки в двох бітах.

На рисунку 5 графічно зображено недолік методу простої перевірки парності.

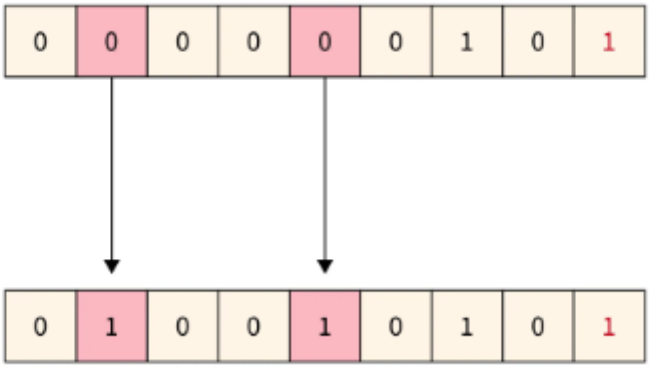


Рисунок 5. Недоліки методу простої перевірки парності

Перевірка двовимірної парності

Для кожного рядка та стовпця біта перевірки парності обчислюються методом перевірки парності. Парність як для рядків, так і для стовпців передається з даними, надісланими від відправника до одержувача. На стороні одержувача біти парності порівнюються з обчисленою парністю отриманих даних [1].

Недоліками цього методу є наступні:

якщо 2 біти пошкоджено в 1 блоці даних, а інший блок даних у тій самій позиції пошкоджено, цей метод не може виявити помилку;

цей метод не використовується для виявлення 4-бітних помилок або помилок, які перевищують 4-біт.

На рисунку 6 зображено метод двовимірної перевірки парності.

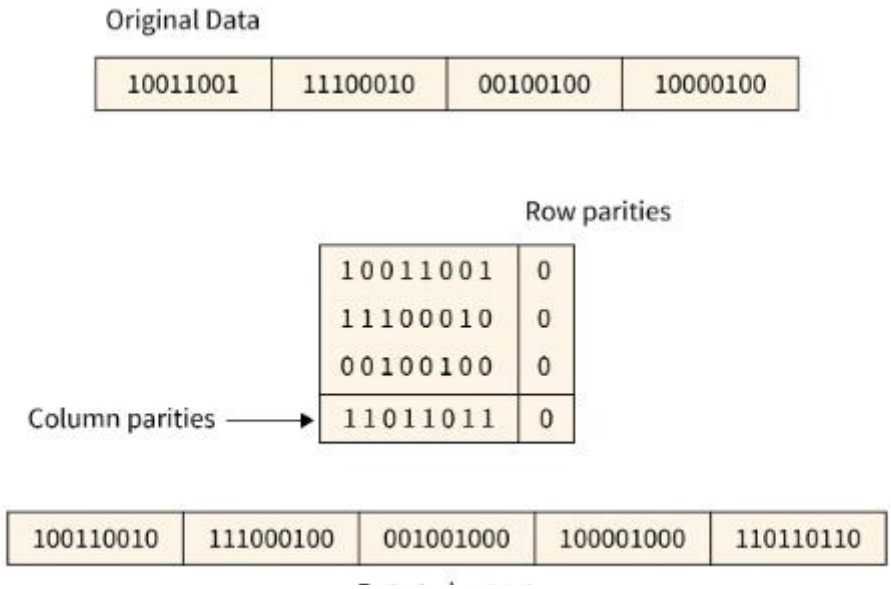


Рисунок 6. Метод двовимірної перевірки парності

Метод контрольної суми

Контрольна сума — це метод виявлення помилок, що виявляє помилку шляхом поділу даних на сегменти однакового розміру, а потім використання доповнення до 1 для знаходження суми сегментів, згодом сума передається разом з даними одержувачу, і той самий процес виконує одержувач. А на стороні приймача всі нулі в сумі вказують на правильність даних [1].

Алгоритм застосування контрольної суми наступний

Спочатку всі дані діляться на k сегментів у схемі виявлення помилок контрольної суми, і кожен сегмент має m біт.

Для визначення суми на стороні відправника всі сегменти додаються за допомогою арифметики з доповненням до 1 . А для визначення контрольної суми доповнюємо суму.

Разом із сегментами даних також передаються сегменти контрольної суми.

Усі сегменти, отримані на стороні приймача, додаються за допомогою арифметики доповнення 1S для визначення суми. Потім доповніть суму.

Отримані дані приймаються лише за умови, що результат буде 0 . І якщо результат не дорівнює 0, він буде відкинутий [2].

Недолік методу виявлення помилки контрольною сумою полягає в наступному: якщо один суб-блок даних має один або більше пошкоджених бітів, а відповідні біти протилежного значення також пошкоджені в іншому суб-блоці. У цій ситуації помилка не виявляється, оскільки в цьому випадку на суму стовпців не впливають пошкоджені біти.

Метод контрольної суми зображено на рисунку 7 нижче.

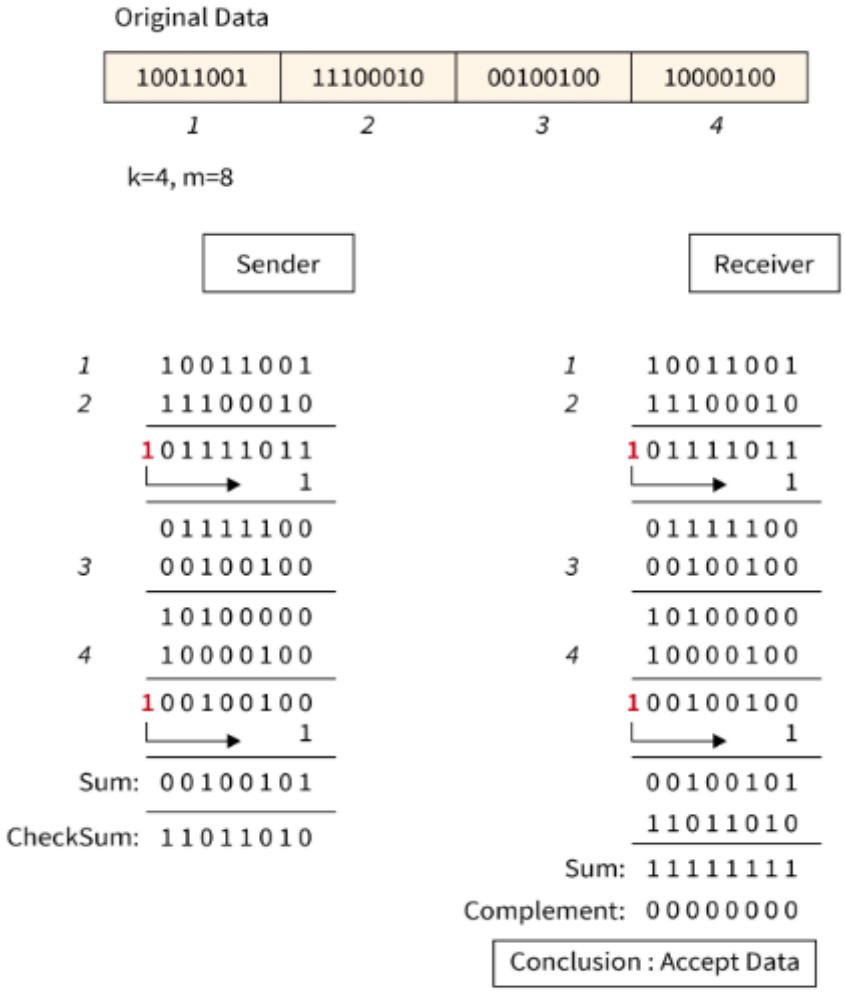


Рисунок 7. Метод контрольної суми

Перевірка циклічної надлишковості

Перевірка циклічної надлишковості має наступні особливості:

Схема контрольної суми використовує метод додавання, але цикличний надлишковий код (CRC) використовує двійковий ділення.

Послідовність бітів, широко відома як перевірка циклічної надлишковості, додається до кінця бітів CRC. Це робиться для того, щоб отримана одиниця даних ділилася на друге заздалегідь визначене двійкове число.

Приймаючі блоки даних потрібно розділити їх на однакове число. Ці одиниці даних приймаються та визнаються правильними лише за умови, що залишок цього ділення дорівнює нулю. Залишок показує, що дані не вірні. Отже, їх потрібно викинути [6].

Недоліками циклічної перевірки надмірності є наступні:

Накладні витрати:

для виявлення помилок потрібні додаткові ресурси та потужність обробки, що може призвести до збільшення накладних витрат на мережу. Це може призвести до зниження продуктивності мережі та збільшення затримки і до переповнення даних.

Помилкові спрацьовування:

механізми виявлення помилок іноді можуть генерувати хибні спрацьовування, що може призвести до непотрібної повторної передачі даних. Це може ще більше збільшити накладні витрати на мережу.

Обмежене виправлення помилок:

виявлення помилок може лише ідентифікувати помилки, але не може їх виправити. Це означає, що одержувач повинен покладатися на відправника для повторної передачі даних, що може призвести до подальших затримок і збільшення витрат на мережу.

Сутність перевірки циклічної надлишковості зображено на рисунку 8 нижче.

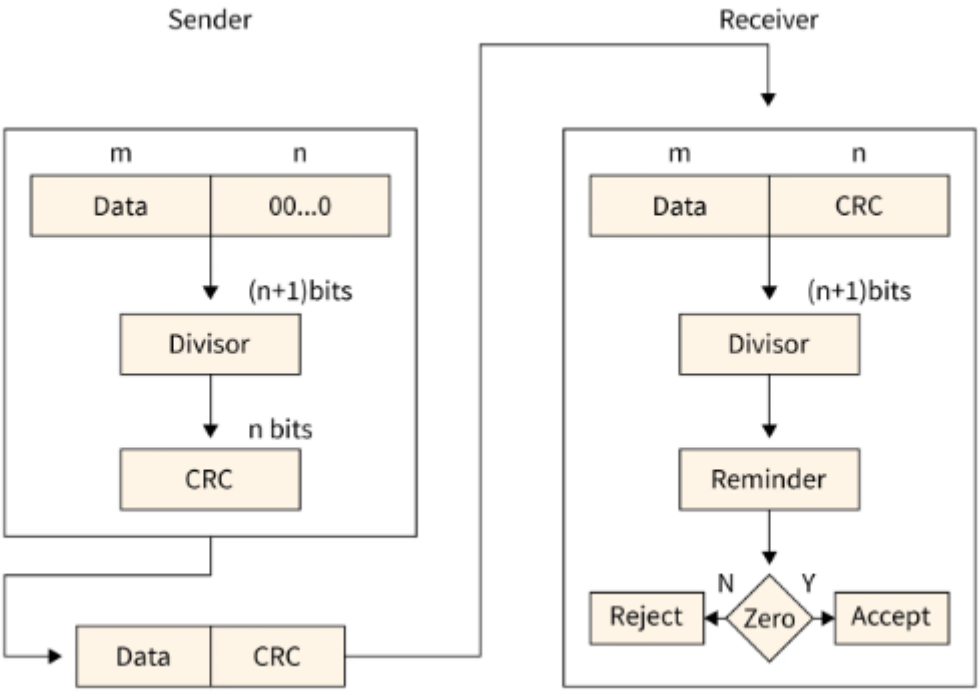


Рисунок 8. Перевірка циклічної надлишковості

Виправлення помилок

Коли дані надсилаються від відправника до одержувача і виникають помилки, їх потрібно виявити та виправити. Існує два найбільш розповсюджених способи виправлення помилок:

Зворотне виправлення помилок

Коли будь-яка помилка знайдена в даних на стороні одержувача, той надсилає запит на повторну відправку всього блоку даних.

Попереднє виправлення помилок

У цьому способі одержувач використовує код виправлення помилок, який автоматично виправляє їх [3].

Ми можемо виявити помилку, використовуючи один додатковий біт, але ми не можемо використовувати цей біт для виправлення. Важливо знати точне місце розташування помилки, якщо ми хочемо її виправити. Наприклад, для виявлення однобітної помилки код виявлення помилки перевіряє, що помилка насправді в одному із семи бітів. Припустимо, що d представляє кількість бітів

даних, а r - кількість надлишкових бітів. Наведена нижче формула використовується для знаходження r кількості зайвих бітів:

2r> =d+r+1

Наведена вище формула використовується для визначення значення r . Наприклад, припустимо, що 4 буде значенням d, тоді 3 буде єдиним і найменшим значенням, яке задовольняє цьому конкретному відношенню.

Код Хеммінга

Код Хеммінга — це техніка, розроблена Р. В. Хеммінгом для визначення положення біта помилки. Код Хеммінга заснований на зв'язку між надлишковими бітами та блоками даних, і його головна перевага полягає в тому, що його можна застосовувати до блоків даних будь-якої довжини.

Біти парності: біти парності — це особливий тип бітів, які додаються до вихідних даних двійкових бітів, щоб зробити загальну одиницю парною або непарною [3].

Парність: для перевірки парності використовується така концепція: значення біта парності буде 0 , якщо загальна кількість одиниць є парною, а значення біта парності може дорівнювати 1, якщо загальна кількість одиниць дорівнює непарній.

Непарна парність: для перевірки парної парності використовується такий порядок: значення біта парності буде 1, якщо загальна кількість одиниць парна, а значення біта парності може бути 0, якщо загальна кількість одиниць є непарною. .

Алгоритм коду Хеммінга наступний:

Додайте інформацію в бітах 'd' до надлишкових бітів 'r' , щоб зробити дані як d+r .

Десяткове значення буде присвоєно позиції кожної (d+r) цифри.

У позиціях 1,2…2k будуть розміщені біти « r» .

Біти парності знову обчислюються на стороні приймача. Позиція помилки визначає десяткове значення біта парності.

Співвідношення позиції помилки та двійкового числа представлено в таблиці.

Помилка позиції	Двійкове число
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Розглянемо приклад для детального пояснення концепції коду Хеммінга.

1010 має бути вихідними даними, які потрібно передати.

'd' загальна кількість бітів даних= 4

Кількість зайвих бітів= 2r>=d+r+1

2r > =4+r+1

Отже, наведене вище співвідношення повністю виконується, коли 3 буде значенням r.

Визначення позиції надлишкових бітів: 3 – кількість зайвих бітів. r1, r2 і r4 використовуються для відображення цих трьох бітів. Ми можемо знайти положення цих бітів для підвищеного ступеня 2. Отже, відповідно до цього, їх положення будуть1,21,22.

позиція r1 дорівнює 1

позиція r2 дорівнює 2

позиція r4 дорівнює 4

На рисунку 9 нижче зображені дані із зайвими бітами

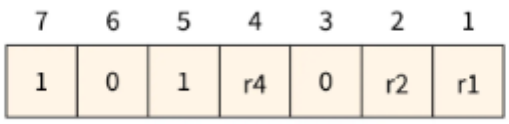


Рисунок 9. Дані із зайвими бітами

Визначення бітів парності

Значення біта r1 обчислюється на основі перевірки парності, виконаної для бітів, доступних у позиції, двійкове перетворення якої містить 1 у першій позиції. На рисунку 10 представлено, як знайти значення біта r1.

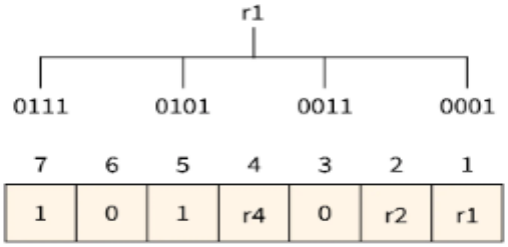


Рисунок 10. Як знайти значення біта r1

Використовуючи наведений вище рисунок, ми можемо знайти позицію, яка має 1, оскільки її перший біт у двійковому представленні дорівнює 1,3,5,6 . Тепер до цих бітових позицій застосовується метод перевірки парності. Кількість одиниць у цих бітових позиціях 2, що є парним числом, тому 0 є значенням біта r1.

Значення біта r2 обчислюється на основі перевірки парності, що виконується для бітів, доступних у позиції, двійкове перетворення якої містить 1 у другій позиції . На рисунку 11 представлено, як знайти значення біта r2.

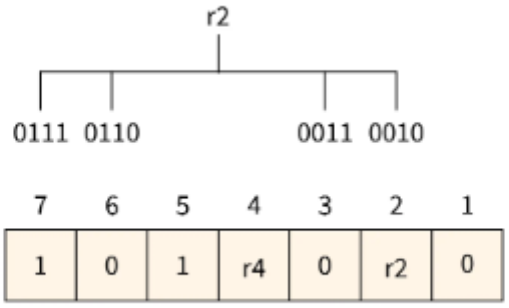


Рисунок 11. Як знайти значення біта r2

Користуючись наведеним вище малюнком, ми можемо знайти позицію, яка має 1 , оскільки її другий біт у двійковому представленні дорівнює 2, 3, 6, 7 . Тепер до цих бітових позицій застосовується метод перевірки парності. Кількість одиниць у цих бітових позиціях дорівнює 1, що є непарним числом, тому 1 є значенням біта r2 .

Значення біта r4 обчислюється на основі перевірки парності, виконаної для бітів, доступних у позиції, двійкове перетворення якої містить 1 у третій позиції. На рисунку 12 представлено, як знайти значення біта r4.

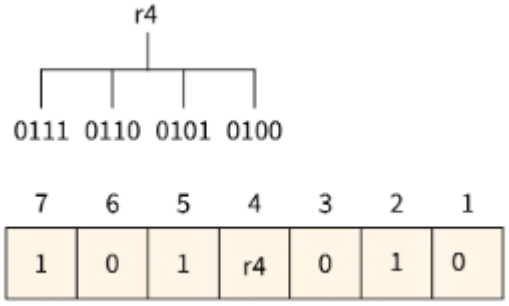


Рисунок 12. Як знайти значення біта r4

Користуючись рисунком, ми можемо знайти позицію, яка має 1, оскільки її третій біт у двійковому представленні дорівнює 4, 5, 6, 7. Тепер до цих бітових позицій застосовується метод перевірки парності. Кількість одиниць у цих бітових позиціях 2, що є парним числом, тому 0 є значенням біта r4 .

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Безруков, В.В. Теорія інформації . Д.: ДДТУЗТ, 2001. 110 с.
2. Жураковсъкий Ю. П. Теорія інформації та кодування : підручник. К.: Вища школа, 2001. 255 с.
3. Кожевников В.Л. Основи збирання, обробки і передачі інформації. Теоретичні основи Д.: НГУ, 2007. 108 с.
4. Локазюк В.М., Савченко Ю.Г. Надійність, контроль, діагностика і модернізація ПК.Навчальний посібник. К.: Видавничий центр «Академія», 2004, 375 с.
5. Романюк М.І., Савченко Ю. Г. Основи теорії інформації та кодування. Конспект лекцій: навч. посіб. для студ. К: КПІ ім. Ігоря Сікорського, 2019. 70 с.
6. Подлевський Б.М., Рикалюк Р.Є. Теорія інформації, Львів: ЛНУ імені Івана Франка, 2016, 339 с.

Питання для самоконтролю

- Що таке помилка у переданій інформації?

- Які основні типи помилок у переданій інформації існують?
- Які основні методи виявлення помилок у переданій інформації існують?
- Які основні методи корекції помилок у переданій інформації існують?
- Яка різниця між однозначними та неоднозначними помилками?
- Яка різниця між випадковими та систематичними помилками?
- Як працюють контрольні суми?
- Як працюють циклічні контрольні суми?
- Як працюють коди Хемминга?
- Як працюють коди Ріда-Соломона?

Модуль ІІ. Математичні основи інформатики.

Тема 4. Основи теорії множин. Основні поняття теорії графів.

Мета: надати студентам загальне уявлення про теорію множин та теорію графів, їх основні поняття та визначення. Розглянути основні операції та властивості множин. Ознайомити студентів із основними типами графів та їх властивостями.

План:

- Множини та операції над ними.
- Відносини на множинах.
- Поняття лінгвістичної змінної та нечіткої множини.
- Види функцій власності.
- Операції над нечіткими множинами. Нечіткі висловлювання.

Теорія множин — це розділ математичної логіки, де вивчаються множини та їхні властивості. Множина — це сукупність об'єктів або груп об'єктів. Ці об'єкти часто називають елементами множини. Наприклад, група гравців у команді з футболу є набором. Оскільки кількість гравців у команді з футболу може бути лише 11 одночасно, можна сказати, що ця множина є скінченною. Іншим прикладом множини є набір українських голосних або приголосних. Але є багато множин, які мають нескінченні члени, такі як множина натуральних чисел, множина цілих чисел, множина дійсних чисел, множина уявних чисел тощо.

Георг Кантор (1845-1918), німецький математик, започаткував концепцію «теорії множин». Працюючи над «Задачами на тригонометричні ряди», він зіткнувся з множинами, які стали одними з найфундаментальніших понять в математиці [4]. « «Множина – це багато, що мислиться як єдине ціле-», зауважив математик. Без розуміння множин буде важко пояснити такі поняття, як функції, послідовності, ймовірність тощо.

Множина — це чітко визначена сукупність об’єктів або людей [1]. Множини можна пов’язати з багатьма прикладами з реального життя, такими як кількість річок в Індії, кількість кольорів у веселці тощо. Щоб зрозуміти множини, розглянемо практичний сценарій. Йдучи з дому до університету, Сергій вирішив записати назви ресторанів, які стоять між ними. Список ресторанів у тому порядку, в якому вони надходили, був таким:

Список 1: Р-А Р-Б Р-В Р-Г Р-Д

Вищезазначений список є набором об'єктів. Крім того, він чітко визначений. Під чітко визначеним мається на увазі, що кожен повинен мати можливість визначити, чи належить об’єкт до певної категорії чи ні. Наприклад, стаціонарний магазин не відноситься до категорії ресторанів. Якщо сукупність об’єктів чітко визначена, вона називається множиною.

Об’єкти в множині називаються елементами множини. Множина може мати скінченні або нескінченні елементи. Повертаючись з університету, Сергій хотів підтвердити раніше складений список. Цього разу він знову написав перелік у тому порядку, у якому приходили ресторани. Новий список був таким:

Список 2:Р-Д Р-Г Р-В Р-Б Р-А

Тепер це інший список. Але чи інша множина? Відповідь - ні. Порядок елементів не має значення в множинах, тому це все та сама множина.

Множини позначають великими літерами латиниці, рідше кирилиці. Множина задана (визначена), якщо про будь-який об’єкт можна сказати, чи належить він цій множині, чи ні. Об’єкти, що утворюють множину, називаються елементами множини і позначаються відповідними малими літерами. Якщо елемент а належить множині А, це позначається так: а $\in$ А, в іншому випадку пишуть а $\notin$ А[3].

Способи подання множин

Множини можна представити двома способами:

Форма реєстру(списку) або таблична форма

Форма конструктора

Форма реєстру

У формі списку всі елементи множини перераховані, розділені комами та взяті у фігурні дужки { }.

Наприклад: якщо набір представляє всі високосні роки між 1995 і 2023 роками, тоді він буде описаний за допомогою форми списку як:

A ={1996,2000,2004,2008,2012, 2016,2020}

Тепер елементи в фігурних дужках записуються в порядку зростання. Це може бути порядок спадання або будь-який випадковий порядок. Як обговорювалося раніше, порядок не має значення для набору, представленого у формі реєстру.

Крім того, множинність ігнорується під час представлення множин [2]. Наприклад, якщо L представляє набір, який містить усі літери в слові АДРЕСА, правильним представленням форми Реєстру буде

L ={А,Д,Р,Е,С,А }= {С,А,Р,Е,А,Д}

L≠ { А,Д,Д,Р,Е,С,А }

Форма конструктора

У формі конструктора множини усі елементи мають спільну властивість. Ця властивість не застосовується до об'єктів, які не належать до множини [2].

Наприклад: якщо набір S містить усі елементи, які є парними простими числами, він представлений у вигляді:

S={ x: x парне просте число}

де 'x' — це символ, який використовується для опису елемента.

":" означає "такий, що"

"{}" означає "набір усіх"

Таким чином, S = {x:x — парне просте число} читається як «множина всіх x, для яких x — парне просте число». Форма списку для цієї множини S буде

S = 2. Ця множина містить лише один елемент. Такі множини називаються одиночними/одиничними множинами.

Інший приклад:

F = {p: p — набір двозначних ідеальних квадратних чисел}

F = {16, 25, 36, 49, 64, 81}

Ми бачимо, що у наведеному вище прикладі 16 — квадрат 4, 25 — квадрат 5, 36 — квадрат 6, 49 — квадрат 7, 64 — квадрат 8 і 81 — квадрат 9.

Незважаючи на те, що 4, 9, 121 тощо також є повними квадратами, вони не є елементами множини F, оскільки вона обмежена лише двозначним повним квадратом.

Типи множин, їх символи

Множини класифікуються на різні типи на основі елементів або типів елементів [5]. Це:

Скінченна множина: кількість елементів скінченна

Нескінченна множина: кількість елементів нескінченна

Порожня множина: не має елементів, позначається ∅.

Набір Singleton: має лише один елемент

Рівна множина: дві множини є рівними, якщо вони мають однакові елементи

Еквівалентна множина: дві множини еквівалентні, якщо вони мають однакову кількість елементів

Підмножина: коли всі елементи множини A належать множині B, тоді A є підмножиною B

Набір потужностей: набір усіх можливих підмножин.

Універсальний набір: будь-який набір, який містить усі попередні набори.

Існують символи, прийняті для загальних множин. Деякі множини мають загальновизнані позначення, наприклад, N – множина натуральних чисел, Z – множина цілих чисел, Q – множина раціональних чисел, R – множина дійсних чисел, C – множина комплексних чисел [7]. Вони наведені в таблиці нижче:

Таблиця 1: Символи, що позначають загальні набори

символ	Відповідний набір
H	Представляє набір усіх натуральних чисел, тобто всіх додатних цілих чисел. Це також може бути представлено Z + . Приклади: 9, 13, 906, 607 тощо.

З	Представляє набір усіх цілих чисел Символ походить від німецького слова Zahl , що означає число. Цілі додатні та від'ємні числа позначаються відповідно Z + і Z – . Приклади: -12, 0, 23045 тощо.
Q	Представляє набір раціональних чисел Символ походить від слова Quotient . Визначається як частка двох цілих чисел (з відмінним від нуля знаменником) Додатні і від'ємні раціональні числа позначаються Q + і Q – відповідно. Приклади: 13/9. -6/7, 14/3 тощо.
Р	Представляє дійсні числа, тобто всі числа, розташовані на числовій прямій. Додатні та від'ємні дійсні числа позначаються R + і R – відповідно. Приклади: 4,3, π, 4√ 3 тощо.
С	Представляє набір комплексних чисел. Приклади: 4 + 3i, і тощо.

Таблиця 2: Інші символи в множинах

символ	Назва символу
{}	встановити
A ∪ B	А союз Б
A ∩ B	А перехрестя В
A ⊆ B	А є підмножиною В
A ⊄ B	А не є підмножиною В
A ⊂ B	правильна підмножина / сувора підмножина
A ⊃ Б	правильна супермножина / сувора супермножина
A ⊇ B	супермножина
A ⊅ Б	не надмножина
Ø	порожня множина
П (С)	множина потужності
A = Б	рівна множина
A c	Доповнення А
a∈B	елемент В
x∉A	х не є елементом А

Формули та операції теорії множин

Формули множин наступні

n(A ∪ B) = n(A) +n(B) – n (A ∩ B)

$n(A \cup B)=n(A)+n(B)$ {коли A і B є непересічними множинами}

$n(U)=n(A)+n(B)–n(A \cap B)+n((A \cup B)\ c\ )$

$n(A \cup B)=n(A-B)+n(B-A)+n(A \cap B)$

$n(A-B)=n(A \cap B)-n(B)$

$n(A-B)=n(A)-n(A \cap B)$

$n(A\ c\ )=n(U)-n(A)$

$n(PUQUR)=n(P)+n(Q)+n(R)–n(P \cap Q)–n(Q \cap R)–n(R \cap P)+n(P \cap Q \cap R)$

У теорії множин використовуються різні типи операцій із множинами. Такі як перетин множин, різниця множин, доповнення множин і об’єднання множин [8]. Дії над множинами мають наочний вигляд за допомогою діаграм, на яких множини зображені у вигляді кола і ті області, де розташовані потрібні елементи, виділені кольором. Ці діаграми мають назву діаграми Ейлера-Венна[8].

Об'єднання двох або більше множин - це множина, що містить усі елементи даних множин. Об’єднання множин можна записати за допомогою символу «∪». Припустимо, що об’єднання двох множин X і Y можна представити як X ∪ Y. Іноді плутають з об'єднанням і універсальним набором. Об’єднання двох або більше множин — це операція, що виконується над ними, результатом якої є набір елементів, присутніх в обох множинах, тоді як універсальна множина сама по собі є множиною, яка містить усі елементи інших множин, включаючи власні елементи [9].

Перетин і різниця двох множин є двома різними операціями. Дуже легко відрізнити операції перетину та об’єднання. Але яка різниця між перетинами та різницею множин? Перетин двох множин, A і B, які є підмножинами універсальної множини U, є множиною, яка складається з усіх елементів, спільних як для A, так і для B. Він позначається символом «∩». Усі ті елементи, які належать як A, так і B, являють собою перетин A і B. Таким чином, ми можемо сказати, що,$A \cap B = \{x : x \in A$ і $x \in B\}$ Для n множин A1, A2, A3, …, An, де всі ці множини є підмножиною універсальної множини U, перетином є множина всіх елементів, які є спільними для всіх цих n множин. Зображуючи

це наочно, заштрихована частина діаграми Венна, наведена нижче, представляє перетин двох множин A і B.Перетин (добуток) A і B – це множина C, яка містить лише елементи, що входять до A і B одночасно. Позначення: C = A ∩ B [9].

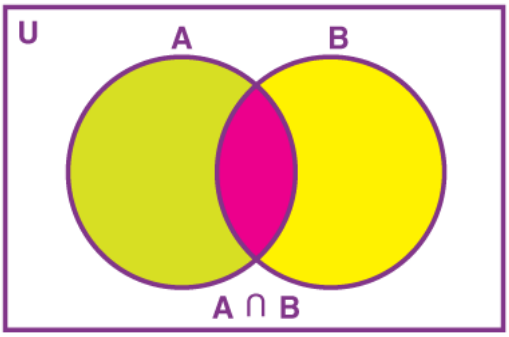


Рисунок 1

Дефініція терміну «граф» та історія виникнення теорії графів

Граф – це наочне графічне представлення взаємозв'язку елементів деякої множини об'єктів. Теорія графів — це дослідження взаємозв'язків, яке є корисним інструментом для кількісного визначення та спрощення рухомих частин динамічної системи. Це дозволяє дослідникам взяти набір вузлів і з'єднань, які можуть абстрагувати будь-що, від планів міста до комп'ютерних даних, і аналізувати оптимальні маршрути. Він використовується для підключення до соціальних мереж, ранжування гіперпосилань у пошукових системах, карт GPS для пошуку найкоротшого маршруту додому тощо.

Теорію графів вперше запровадив у XVIII столітті швейцарський математик Леонгард Ейлер (1707-1783). Його робота над назвою «Проблеми семи мостів Кенігсберга» вважається початком теорії графів. Місто Кенігсберг у Пруссії (сучасний Калінінград, Росія) було розташоване по обидва боки річки Прегель і включало два великі острови — Кнайпхоф і Ломзе — які були з'єднані один з одним через дві материкові частини міста сімома мостами. Питання полягало в тому, щоб придумати прогулянку містом, яка б перетинала кожен із цих мостів лише один раз [4]. Ейлер, визнаючи, що відповідними обмеженнями є чотири ділянки землі та сім мостів, намалював перше відоме візуальне представлення сучасного графа. Це абстрагування від конкретної

проблеми, що стосується міста та мостів, до графіка робить проблему математично обробленою, оскільки це абстрактне представлення включає лише інформацію, важливу для вирішення проблеми. Ейлер фактично довів, що ця конкретна проблема не має розв'язку. Однак завдання, з яким він зіткнувся, полягало в тому, щоб розробити відповідну техніку аналізу, а подальші тести підтвердили це твердження з математичною суворістю. Відтоді розділ математики, відомий як теорія графів, мало застосовувався. Однак у наш час його застосування неймовірно зростає [6].

Формально графова модель складається з двох множин - множини V об'єктів (вершин вузлів) і множини P зв'язків (ребер). Вершина графа – це компоненти системи, дуги – це направленні лінії (стрілки), що зв'язують компоненти між собою, ребра – ненаправлені лінії, що зв'язують компоненти між собою.

Рисунок 2. Сучасний граф, представлений набором точок, відомих як вершини або вузли, з'єднаних набором ліній, відомих як ребра.

Використання теорії графів

Оскільки теорія графів вивчає зв'язки, вона надає корисний інструмент для кількісного визначення та спрощення багатьох рухомих частин динамічних систем. Вивчення графів дає відповіді на багато проблем упорядкування, створення мереж, оптимізації, узгодження роботи [9].

Графи можна використовувати для моделювання багатьох типів відносин і процесів у фізичних, біологічних, соціальних та інформаційних системах, і вони мають широкий спектр корисних застосувань, таких як:

Пошук спільнот у мережах, таких як соціальні мережі (рекомендації друзів/зв’язків), або для можливого поширення COVID-19 у спільноті через контакти.

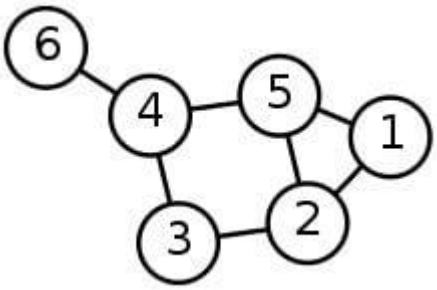


Рисунок 3. Простий приклад графа з шістьма вузлами.

Ранжування гіперпосилань у пошукових системах .

GPS на картах Google, щоб знайти найкоротший шлях додому.

Вивчення молекул і атомів в хімії.

Секвенування ДНК.

Безпека комп'ютерної мережі

Рисунок 4. Складний граф - соціальні мережі. Зображення: Мартін Гранджан/Вікімедіа

Основні типи графів

Існує багато різних видів представлень графів, основними з них є наступні.

Неорієнтовані графи: усі шляхи між кожним вузлом є двонаправленими.

Орієнтовані графи (орграфи) : шляхи між вузлами мають визначені напрямки [7].

Наприклад, у неорієнтованому графі кожен вузол мережі представляє будинки в різних місцях міста, а околиці як дороги між ними. Як випливає з назви, між вузлами графіка немає визначеного напрямку. Це означає, що край (дорога), що з’єднує вузли (будинки) 1-до-2, буде ідентичним краю від 2-до-1. У цьому прикладі ми припустили, що всі дороги є двонаправленими, і нам не потрібно турбуватися про вулиці з одностороннім рухом.

Коли ми маємо справу з *орієнтованими графами*, нам також потрібно вказати напрямок між різними вузлами. Це означає, що якщо у нас є ребро від

вузла 1-до-2, це не обов’язково означає, що ми також можемо рухатися в протилежному напрямку від 2-до-1.

Якщо ми дотримуємося попереднього прикладу вузлів, що представляють будинки в різних місцях міста, це означає, що якщо у нас є спрямоване ребро від 1 до 2, ми маємо вулицю з одностороннім рухом між цими будинками. Тоді ми можемо їхати від будинку 1 до 2, але їхати в протилежному напрямку від 2 до 1 буде заборонено. Нам потрібно було б вибрати маршрут 2-до-3-до-1. Однак між будинками 2 і 4 ми можемо безпечно їхати в будь-якому напрямку, як вказано подвійними стрілками.

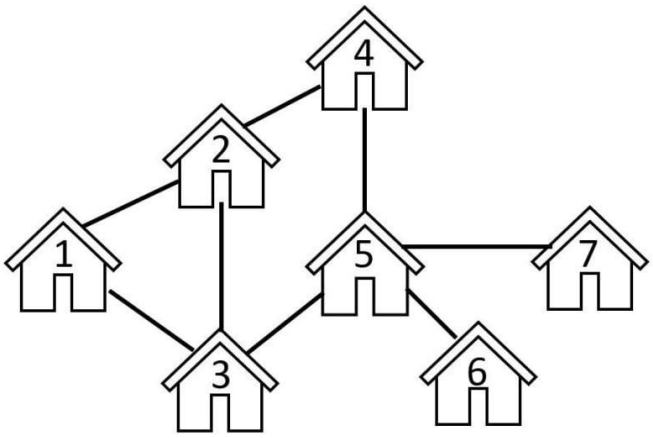


Рисунок 5. Шлях будинків у неорієнтованому графі.

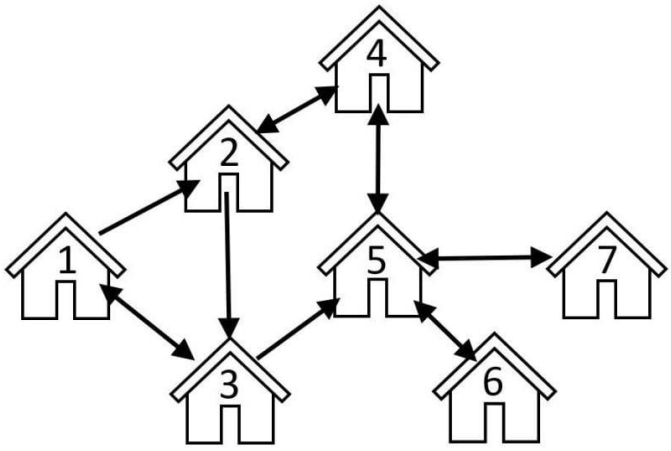


Рисунок 6. Шлях будинків у орієнтованому графі.

Існують й інші типи графів

Нульовий граф: граф, який не має ребер.

Простий граф: граф, який не орієнтований і не має петель або кількох ребер.

Мультиграф: граф із кількома ребрами між одним набором вершин. На ньому утворилися петлі.

З’єднаний граф: граф, де будь-які дві вершини з’єднані шляхом.

Роз’єднаний граф: граф, де будь-які дві вершини або вузли роз’єднані шляхом.

Граф циклу: граф, який завершує цикл.

Повний граф: якщо кожна пара вершин з’єднана ребром, такий граф називається повним графом.

Плаский граф: якщо жодні два ребра графа не перетинаються і всі вершини та ребра намальовані в одній площині, тоді граф називається плоским графом. [1].

Існують певні терміни, які використовуються в представленні графів, наприклад ступінь, дерева, цикл тощо [5].

Дерево в графі - це зв'язок між неорієнтованими мережами, які мають лише один шлях між будь-якими двома вершинами. Він був представлений британським математиком Артуром Кейлі в 1857 році. Дерева графів мають лише прямі лінії між вузлами в будь-якому конкретному напрямку, але не мають жодних циклів чи петель. Тому дерева є орієнтованим графом.

Ступінем у графі називається кількість ребер, з’єднаних із вершиною. Він позначається deg(v), де v — вершина графа.

Цикл— це замкнутий шлях у графі, який утворює цикл. Коли початкова і кінцева точки однакові в графі, який містить набір вершин, формується цикл графа. Коли в замкнутому контурі немає повторення вершини, то цикл є простим циклом. Цикловий граф позначається C n . Цикл, який має парну кількість ребер або вершин, називається парним циклом. Цикл, який має непарну кількість ребер або вершин, називається непарним циклом.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Андрійчук В. І., Комарницький М. Я., Іщук Ю. Б. Вступ до дискретної математики. — К.: Центр навчальної літератури, 2004. — 254 с.

2. Бардачов Ю. М., Соколова Н. А., Ходаков В. Є. Дискретна математика. — К.: Вища школа, 2002. — 287 с.

3. Бондаренко М.Ф., Білоус Н.В., Руткас А.Г. Комп’ютерна дискретна математика. — Харків: "Компанія Сміт", 2004. — 480 с.

4. Вітенько І. В. Математична логіка: Курс лекцій. — Ужгород: УжДУ, 1971. — 224 с.

5. Капітонова Ю. В., Кривий С. Л., Летичевський О. А., Луцький Г. М.

Основи дискретної математики. — К.: Наукова думка, 2002. — 580 с.

6. Коцовський В. М. Дискретна математика та теорія алгоритмів: Методичні матеріали до практичних робіт / В. М. Коцовський. — Ужгород: Видавництво УжНУ "Говерла", 2019. — 35 с

7. Кривий С.Л. Дискретна математика: підручник для студентів вищ. навч. закл.– Чернівці-Київ: Видавничий дім «Букрек», 2014. – 568 с.

8. Нікольський Ю. В., Пасічник В. В., Щербина Ю. М. Дискретна математика. — К.: Видавнича група BHV, 2007. — 368 с.

9. Ядренко М. Й., Оленко А. Я. Дискретна математика. навчальнометодичний посібник. — К.: Київський університет ім. Т. Шевченка, 1995. — 83 с.

Питання для самоконтролю

- Що таке множина?
- Які основні операції над множинами існують?
- Які основні властивості множин існують?
- Що таке граф?
- Які основні типи графів існують?
- Які основні властивості графів існують?
- Яка різниця між об'єднанням і перетином множин?
- Яка різниця між різницею і доповненням множин?
- Які закони де Моргана?
- Який закон додавання?
- Який закон виключення третього?
- Яка різниця між орієнтованим і неорієнтованим графом?
- Яка різниця між зв'язним і незв'язним графом?
- Яка різниця між деревоподібним і недеревовидним графом?
- Що таке цикл у графі?
- Що таке шлях у графі?

Тема 5. Елементи теорії алгоритмів та формальних мов.

Мета: надати студентам загальне уявлення про теорію алгоритмів та формальні мови, їх основні поняття та визначення, розглянути основні класи алгоритмів та їх властивості, ознайомити студентів із основними типами формальних мов та їх властивостями..

План:

- Інтуїтивне поняття алгоритму. Необхідність формалізації інтуїтивного поняття алгоритму.
- Поняття алгоритмічної системи. Зведення алгоритмів до числових функцій.
- Поняття обчислюваної функції.
- Поняття про формальні мови та породжувальні граматики.
- Машина Тьюринга. Алгоритмічна роздільність.
- Форми представлення алгоритмів.
- Базові алгоритмічні структури.
- Структурне проектування.
- Об'єктно-орієнтоване проектування.
- Мови програмування. Підготовка програми до виконання.

Дефініція терміну «алгоритм» та його застосування

Слово «алгоритм» означає набір фіксованих правил або інструкцій, яких слід дотримуватися під час обчислень або інших операцій вирішення проблем або процедуру для розв’язання математичної задачі за фіксовану кількість кроків, яка часто включає рекурсивні операції, тобто такі, значення яких для даного аргументу розраховується за допомогою значень для попередніх аргументів. Таким чином, алгоритм відноситься до послідовності фіксованих кроків для вирішення конкретної проблеми [3].

Алгоритми відіграють вирішальну роль у різних сферах і мають багато застосувань. Деякі з ключових областей, де використовуються алгоритми, включають:

Комп'ютерні науки. Алгоритми складають основу комп'ютерного програмування та використовуються для вирішення різноманітних завдань, починаючи від простого сортування та пошуку й закінчуючи складними завданнями, такими як штучний інтелект і машинне навчання.

Математика. Алгоритми зустрічаються в усіх розділах математики, в яких існують алгоритмічні проблеми. Такі проблеми можуть виникати в математичній логіці та в теорії моделей. Для кожної теорії проблема, яка виникає, полягає в тому, щоб вирішити множину всіх істинних або доказових тверджень цієї теорії щодо множини всіх її тверджень. Теорія алгоритмів тісно пов'язана з математичною логікою, оскільки поняття алгоритму є основою одного з центральних понять математичної логіки — поняття числення, внаслідок чого теорема Геделя про неповноту формальних систем може бути розв'язана з теорем теорії алгоритмів. Нарешті, теорія алгоритмів тісно пов'язана з основами математики, де однією з ключових проблем є співвідношення між конструктивним і неконструктивним. Зокрема, теорія алгоритмів дає апарат для розвитку конструктивного напряму в математиці.

Дослідження операцій. Алгоритми використовуються для оптимізації та прийняття рішень у таких сферах, як транспорт, логістика та розподіл ресурсів.

Штучний інтелект. Алгоритми є основою штучного інтелекту та машинного навчання та використовуються для розробки інтелектуальних систем, які можуть виконувати такі завдання, як розпізнавання зображень, обробка природної мови та прийняття рішень.

Наука про дані. Алгоритми використовуються для аналізу, обробки та отримання інформації з великих обсягів даних у таких сферах, як маркетинг, фінанси та охорона здоров'я [1].

Це лише кілька прикладів із багатьох застосувань алгоритмів. Використання алгоритмів постійно розширюється з появою нових технологій і галузей, що робить їх життєво важливим компонентом сучасного суспільства.

Перевагами алгоритмів є наступні позиції

доступність застосування.

поетапне представлення рішення заданої задачі.

проблема розбита на менші частини або кроки, отже, програмісту легше перетворити її на реальну програму.

До недоліків алгоритмів відносяться:

Написання алгоритму займає багато часу, тому це займає багато часу.

Розуміння логіки за допомогою алгоритмів може бути дуже ускладненим.

Розгалуження та цикл важко відобразити в алгоритмах [4].

Відомості з історії теорії алгоритмів

Попередні уявлення про алгоритми відігравали певну роль у математиці протягом всієї історії. Однак сама концепція алгоритму була сформульована лише в XX столітті і стала об'єктом самостійного дослідження, хоча спочатку лише в досить нечітко визначеній формі, у 1920-х роках інтуїтивістської школи Л. Дж. Брауера та Г. Вейля [1].

Систематичні дослідження розпочалися в 1936 році, коли А. Черч опублікував першу формалізацію поняття обчислюваної функції. Він запропонував ототожнити поняття визначеної обчислюваної функції з природними аргументами та природними значеннями з поняттям загальної рекурсивної функції. Черч довів у 1936 році, що проблема вирішення набору всіх істинних припущень логіки предикатів була нерозв'язною. Інші важливі результати з цього питання належать А. Тарському, А. І. Мальцеву та ін. А. М. Тьюрінг , та Е. Л. Пост дали першу формалізацію концепції алгоритму в комп’ютерних термінах - машина Тьюрінга [5].

Перші приклади напівгруп з нерозв'язною текстовою проблемою були знайдені незалежно один від одного Марковим і Постом у 1947 р., а приклад групи з нерозв'язною текстовою проблемою був знайдений П. С. Новіковим у 1952 р. Марков у 1958 р. довів, що проблема гомеоморфності в топології була нерозв'язною для важливого класу випадків. Окрім того, Марков уточнив поняття алгоритму, ввівши поняття нормального алгоритму .

У теорії чисел у 1970 році Матіясевич довів, що проблема розв'язності діофантових рівнянь нерозв'язна.

Найбільш загальний підхід до поняття запропонував А. Н. Колмогоров. Запропонована Колмогоровим у 1965 році теорія алгоритмів була використана як основа теорії інформації. Теорія алгоритмів є теоретичною основою ряду проблем обчислювальної математики і тісно пов'язана з кібернетикою, в якій важливим предметом є вивчення алгоритмів керування [1].

Загальні положення теорії алгоритмів

Оскільки алгоритми мають справу виключно з так званими конструктивними об’єктами, ідеї та методи теорії алгоритмів можна застосовувати до неконструктивних об’єктів, лише якщо вони закодовані або перейменовані як конструктивні об’єкти. Вивчення загальних властивостей таких кодувань, переважно ситуацій, у яких такі кодування або імена є натуральними числами, є предметом теорії обчислювання, яка є важливою частиною теорії алгоритмів.

Область застосовності алгоритму — це сукупність об’єктів, до яких він застосовний. Кажуть, що алгоритм А 1) «обчислює функцію f», якщо її область визначення збігається з областю визначення f, і якщо А перетворює будь-який x зі свого домену в f(x); 2) «розв’язує множину А відносно множини X», якщо вона застосовна до будь-якої x від X і перетворює всі x від X∩ A до слова "так" , і все x від X\ A до слова «ні» ; і 3) «підраховує (нумерує) множину B», якщо її областю є послідовність натуральних чисел, а множиною результатів є B. [3].

Функція називається обчислюваною, якщо існує алгоритм, який обчислює функцію. Набір називається розв’язаним відносно X якщо існує алгоритм, який розв’язує його відносно X. Набір називається перелічуваним, якщо він порожній або якщо існує алгоритм, який його перераховує.

Детальний аналіз поняття алгоритму показує, що: 1) область можливих вихідних даних і область застосовності алгоритму є перелічуваними множинами. У свою чергу, 2) для будь-якої пари перелічуваних множин, одна з яких входить до іншої, можна знайти алгоритм, у якому більша множина є діапазоном можливих входів, тоді як менша множина є діапазоном виходів. Справедливі такі основні теореми: 3) функція f обчислювана тоді і тільки тоді,

коли її граф, тобто множина всіх пар виду ⟨ x , f(x) , перелічується; 4) підмножина A перелічуваної множини X розв’язується відносно X якщо і тільки якщо A і X\ A перелічуються; 5) якщо A і Б перелічуються, A ∪ B і A ∩ B також перелічуються; 6) у кожній нескінченній перелічуваній множині X існує перелічувана підмножина з неперелічуваним доповненням (з огляду на 4 пункт) ця перелічувана підмножина не буде розв’язною щодо X); 7) для кожної нескінченної перелічуваної множини X існує обчислювана функція, визначена на підмножині цієї множини, яка не може бути розширена до обчислювальної функції, визначеної на всіх X. Твердження 6) і 2) у поєднанні дають приклад алгоритму з нерозв’язною областю застосовності [4].

Розв’язні та перераховані множини являють собою найпростіші і найважливіші приклади множин, структура яких визначається певними алгоритмічними процедурами. Систематичне вивчення множин конструктивних об'єктів з точки зору властивостей цих множин, пов'язаних з наявністю деяких алгоритмів, становить так звану алгоритмічну теорію множин; окремі поняття, методи та результати цієї теорії знаходять аналоги в алгоритмічній теорії множин .

Поняття алгоритмічних задач

Проблема побудови алгоритму з певними заданими властивостями відома як алгоритмічна задача. Як правило, властивість алгоритму, яку потрібно знайти, формулюється в термінах відповідності, яка має зберігатися між входами та результатами алгоритму.

Прикладами алгоритмічних задач є:

проблема обчислення заданої функції, тобто пошук алгоритму, який обчислює функцію;

задача розв'язування заданої множини, тобто знаходження алгоритму, який розв'язує цю множину відносно деякої іншої множини;

задача перерахування заданої множини, тобто знайти алгоритм перерахування заданої множини [8].

Нерозв'язаність алгоритмічної задачі означає відсутність відповідного алгоритму. Теореми, які встановлюють нерозв'язаність таких задач, належать до найважливіших теорем теорії алгоритмів. Наприклад, якщо алгоритм має нерозв’язну область, алгоритмічна задача розв’язання цієї області щодо множини всіх можливих вхідних даних є нерозв’язаною. Зводячи цю проблему до нерозв’язаності, більшість інших проблем розв’язаності виявилися нерозв’язаними. Питання про те, чи можна таким чином встановити нерозв'язність будь-якої нерозв'язної проблеми, відноситься до проблеми алгоритмічної зведення .

Метрична та описова теорії алгоритмів

Теорія алгоритмів включає описову (якісну) і метричну (кількісну) теорії. Перша стосується алгоритмів з точки зору відповідності, яку вони встановлюють між початковими даними та результатами [4].

Якісна теорія вивчає алгоритми з точки зору складності як самих алгоритмів, так і обчислень, визначених алгоритмами, тобто процесів послідовних перетворень конструктивних об'єктів. Важливо відзначити, що як обчислювальна складність алгоритму, так і складність опису можуть бути визначені різними способами, і цілком може бути, що А виявиться складнішим, ніж В, коли прийнято одне визначення, поки В виявиться складнішим, ніж А за іншим визначенням. Для того, щоб можна було говорити про складність алгоритмів, необхідно спочатку вказати якусь точну мову, на якій алгоритми повинні бути написані, і визначити складність алгоритму як складність його анотації; складність анотації, у свою чергу, може бути визначена різними способами (наприклад, як кількість символів даного типу, які включені в анотацію, або як вибірка таких чисел, обчислених для різних типів символів).

Щоб говорити про обчислювальну складність, необхідно вказати точну форму обчислень у вигляді ланцюжка конструктивних об'єктів, що послідовно змінюють один одного, а також якийсь критерій складності такого ланцюжка — кількість «ланок» (або кроків) в ланцюжку - окремо або в поєднанні з розміром кроків і т.д. [4].

У будь-якому випадку складність обчислення залежатиме від вхідних даних, з яких розпочато обчислення; з цієї причини обчислювальна складність є функцією, яка призначає кожному об’єкту в межах області алгоритму складність відповідного ланцюжка. Розробка методів оцінки складності алгоритмів і обчислень має велике теоретичне і практичне значення.

На відміну від описової теорії алгоритмів, яка зараз викристалізувалася в цілісну математичну дисципліну і має велике теоретичне і практичне значення, розробка методів оцінки складності алгоритмів і обчислень, тобто метрична теорія алгоритмів лише в процесі створення.

Характеристики алгоритму

Алгоритми повинні мати такі характеристики:

Чіткість і недвозначність: Алгоритм має бути однозначним. Кожен його крок має бути зрозумілим у всіх аспектах і вести лише до одного значення.

Визначені вхідні дані: якщо алгоритм вимагає приймати вхідні дані, це мають бути чітко визначені вхідні дані. Він може приймати або не приймати вхідні дані.

Чітко визначені результати: алгоритм повинен чітко визначати, який результат буде отримано, і він також має бути чітко визначеним. Він повинен створити принаймні 1 результат.

Можливість: Алгоритм має бути простим, загальним і практичним, таким, щоб його можна було виконати за допомогою доступних ресурсів. Він не повинен містити якісь технології майбутнього чи щось інше.

Незалежність від мови: розроблений алгоритм має бути незалежним від мови, тобто це мають бути просто прості інструкції, які можна реалізувати будь-якою мовою, і все ж результат буде таким самим, як і очікувалося.

Вхідні дані: алгоритм має нуль або більше вхідних даних. Кожен, що містить фундаментальний оператор, повинен приймати нуль або більше вхідних даних.

Вихід : Алгоритм створює принаймні один вихід.

Визначеність: усі дії в алгоритмі мають бути однозначними, точними та легкими для тлумачення. Посилаючись на будь-яку з дій в алгоритмі, можна чітко зрозуміти, що потрібно зробити. Кожна дія повинна бути визначена без будь-якої двозначності.

Скінченність: Алгоритм повинен закінчуватися після кінцевої кількості кроків у всіх випадках. Нескінченні цикли або рекурсивні функції без базових умов не мають скінченності.

Ефективність: Алгоритм має бути розроблений за допомогою базових, простих і здійсненних операцій, щоб можна було простежити його [1].

Властивостями алгоритму є:

повинен закінчитися через кінцевий час.

повинен давати принаймні один результат.

повинен приймати нуль або більше вхідних даних [4].

Кожен крок в алгоритмі повинен бути ефективним, тобто кожен крок повинен виконувати певну роботу. Отже, ефективність алгоритму необхідно перевіряти та підтримувати. Це може проходити в два етапи:

1.Перевірка алгоритму перед його реалізацією. При цьому алгоритм перевіряється, коли він записаний у вигляді теоретичних кроків. Ця ефективність алгоритму вимірюється шляхом припущення, що всі інші фактори, наприклад, швидкість процесора, постійні і не впливають на реалізацію. Зазвичай це робить розробник алгоритму. Цей аналіз не залежить від типу обладнання та мови компілятора. Дає приблизні відповіді щодо складності програми.

2.Перевірка алгоритму після його реалізації. У цьому випадку алгоритм перевіряється шляхом його реалізації на будь-якій мові програмування та його виконання. Цей аналіз допомагає отримати фактичний і реальний звіт аналізу щодо правильності (для кожного можливого введення/ів, якщо він показує/повертає правильні результати чи ні), необхідного місця, спожитого часу тощо. Тобто це залежить від мови компілятор і тип використовуваного обладнання [5].

Алгоритми можуть бути простими і складними. Алгоритм визначається як складний на основі кількості простору та часу, які він споживає. Отже, складність алгоритму стосується вимірювання часу, який йому знадобиться для виконання та отримання очікуваного результату, а також простору, який йому знадобиться для зберігання всіх даних (вхідних даних, тимчасових даних і вихідних даних). Отже, ці два фактори визначають ефективність алгоритму.
Два фактори складності алгоритму:

Фактор часу: час вимірюється шляхом підрахунку кількості ключових операцій, таких як порівняння в алгоритмі сортування.

Коефіцієнт простору: простір вимірюється шляхом підрахунку максимального обсягу пам'яті, необхідного алгоритму для запуску/виконання.

Тому складність алгоритму можна розділити на два типи [5].

1. Просторова складність: просторова складність алгоритму стосується обсягу пам'яті, необхідного алгоритму для зберігання змінних і отримання результату. Це можуть бути входи, тимчасові операції або виходи. Просторова складність алгоритму обчислюється шляхом визначення наступних двох компонентів:

Фіксована частина: це стосується простору, необхідного для алгоритму. Наприклад, вхідні змінні, вихідні змінні, розмір програми тощо.

Змінна частина: це стосується простору, який може відрізнятися залежно від реалізації алгоритму. Наприклад, тимчасові змінні, динамічний розподіл пам'яті, простір стеку рекурсії тощо.
Тому складність простору S(P) будь-якого алгоритму P дорівнює S(P) = C + SP(I) , де C – фіксована частина, а S(I) є змінною частиною алгоритму, яка залежить від характеристики екземпляра I.

2. Часова складність: Часова складність алгоритму означає кількість часу, необхідного алгоритму для виконання та отримання результату. Це можуть бути звичайні операції, умовні оператори if-else, оператори циклу тощо. Часова складність алгоритму також обчислюється шляхом визначення наступних двох компонентів:

Частина постійного часу: будь-яка інструкція, яка виконується лише один раз, входить до цієї частини. Наприклад, введення, вихід, if-else, перемикач, арифметичні операції тощо.

Частина змінного часу: будь-яка інструкція, яка виконується більше одного разу, скажімо, n разів, входить до цієї частини. Наприклад, цикли, рекурсія тощо. Отже, часова складність будь-якого алгоритму P дорівнює T(P) = C + TP(I) , де C — постійна частина часу, а TP(I) — змінна частина алгоритму, яка залежить від характеристики екземпляра I.

Типи алгоритмів

Існує кілька типів алгоритмів[1,3,4,5]

1. Алгоритм грубої сили:

Це найпростіший підхід до проблеми. Алгоритм грубої сили - це перший підхід, який приходить до пошуку, коли ми бачимо проблему.

2. Рекурсивний алгоритм:

Рекурсивний алгоритм заснований на рекурсії . У цьому випадку завдання розбивається на кілька підчастин і знову і знову викликається та сама функція

3. Алгоритм зворотного відстеження:

Алгоритм зворотного відстеження будує рішення шляхом пошуку серед усіх можливих рішень. Використовуючи цей алгоритм, ми продовжуємо будувати рішення за критеріями. Щоразу, коли рішення не вдається, ми повертаємося до точки збою, будуємо наступне рішення та продовжуємо цей процес, доки не знайдемо рішення або не перевіримо всі можливі рішення.

4. Алгоритм пошуку:

Алгоритми пошуку — це алгоритми, які використовуються для пошуку елементів або груп елементів із певної структури даних. Вони можуть бути різних типів залежно від підходу або структури даних, у якій потрібно знайти елемент.

5. Алгоритм сортування:

Сортування — це впорядкування групи даних певним чином відповідно до вимог. Алгоритми, які допомагають виконувати цю функцію, називаються

алгоритмами сортування. Зазвичай алгоритми сортування використовуються для сортування груп даних за зростанням або спаданням.

6. Алгоритм хешування:

Алгоритми хешування працюють подібно до алгоритму пошуку. Але вони містять індекс з ідентифікатором ключа. У хешуванні певним даним призначається ключ.

7. Алгоритм «розділяй і володарюй»:

Цей алгоритм розбиває проблему на підпроблеми, вирішує одну підпроблему та об’єднує рішення для отримання остаточного рішення. Він складається з наступних трьох кроків:

розділити

розв'язати

комбінувати

8. Жадібний алгоритм:

У цьому типі алгоритму рішення будується частина за частиною. Рішення для наступної частини будується на основі безпосередньої вигоди від наступної частини. Рішення, яке дає найбільшу користь, буде вибрано як рішення для наступної частини.

9. Алгоритм динамічного програмування:

Цей алгоритм використовує концепцію використання вже знайденого рішення, щоб уникнути повторного обчислення тієї самої частини задачі. Він ділить проблему на менші підпроблеми, що перетинаються, і вирішує їх.

10. Рандомізований алгоритм:

У рандомізованому алгоритмі ми використовуємо випадкове число, щоб отримати негайну користь. Випадкове число допомагає визначити очікуваний результат.

В залежності від впливу вхідних даних на функцію ефективності алгоритму алгоритми бувають:

1. Кількісно-залежні по ефективності алгоритми

Це алгоритми, функція трудомісткості яких залежить тільки від розмірності конкретного входу, і не залежить від конкретних значень.

2. Параметрично-залежні по ефективності алгоритми

Це алгоритми, ефективності яких визначається не розмірністю входу, а конкретними значеннями оброблюваних слів пам'яті.

3. Кількісно-параметричні по ефективності алгоритми

Це алгоритми, ефективності яких залежить як від кількості даних на вході, так і від значень вхідних даних, у цьому випадку.

4. Порядково-залежні по ефективності алгоритми

Це алгоритми, в яких кількість операцій залежить від порядку розташування вихідних об'єктів.

Поняття формальних мов та їх використання

Теорія формальних мов— це система ідей, призначених для пояснення мов і граматик як обчислювальних об’єктів. Вона заснований на теорії множин і її математичних властивостях. У теорії формальних мов мова визначається як (можливо нескінченний) набір рядків над деяким кінцевим алфавітом символів. В інформатиці ця формальна мова необхідна для визначення комп’ютерних програм і вираження алгоритмічних проблем. Отже, формальна мовна грамотність вимагає практичних знань множин, відносин (як регулярних, так і раціональних), рядків і мов; такі операції множини, як об’єднання, заперечення, перетин, доповнення та декартів добуток; і властивості замикання мов. Це підтримує детальне розуміння рядків, мов і операцій над рядками та мовами (таких як конкатенація, композиція, розворот, ітерація, замикання Кліні тощо) [2].

Бажання формалізувати природні мови призвело до започаткування цього предмета в 1956 році Ноамом Хомським. Прийнято формальні мови класифікувати на різні рівні на основі ієрархії Хомського. Кожен рівень в ієрархії Хомського пов’язаний із певним типом формальної граматики та конкретним типом абстрактної машини. Усі ці мови мають різні набори правил побудови та забезпечують різні рівні виразності [7].

Наприклад, звичайна мова вважається найбільш простою і часто використовується в пошукових системах і текстових редакторах.

Звичайна мова =anbn:n≥0

Наведена вище формула є прикладом регулярного виразу, що демонструє ряд «a», за яким слідує рівна кількість «b».

Використання «формальних мов» є невід'ємною частиною програмування і знаходять численні застосування в різних секторах:

Основним застосуванням формальних мов є визначення синтаксису мови програмування. Формальні граматики, такі як BNF (формат Бекуса-Наура), використовуються для точного опису синтаксичної структури мови програмування.

Формальні мови також є невід’ємною частиною регулярних виразів, що має вирішальне значення для таких завдань, як розпізнавання образів, заміна тексту та розбір.

Вони також складають значну частину реалізації компіляторів та інтерпретаторів, де правила граматики використовуються для аналізу коду та перевірки його відповідності правилам синтаксису мови

Формальні мови відіграють значну роль у підвищенні ефективності програмування. Наприклад, використовуючи формальні мови як наріжний камінь синтаксису мови програмування, можна ефективніше виявляти помилки кодування. Коли програміст пише код формально визначеною мовою, аналізатор може перевірити, чи відповідає цей код усім правилам мови, і позначити будь-які розбіжності. Це скорочує час, витрачений на налаштування і пошук помилок, які інакше важко знайти [2].

Ключові компоненти у визначенні формальної мови

Теорія формальної мови в інформатиці базується навколо точного визначення та розуміння мов, які використовуються для передачі команд і інструкцій комп’ютерній системі.

Термін «формальна мова» в інформатиці насамперед стосується створення, вираження та аналізу явних інструкцій, спрямованих до комп’ютерної системи.

Це мова, розроблена зі спеціальним синтаксисом і семантикою, визначеною з суворою математичною точністю. Будівельними блоками формальної мови є символи та рядки, створені з цих символів за допомогою граматичних правил мови [2].

Формальна мова в інформатиці може бути визначена як кінцевий або нескінченний набір рядків над кінцевим набором символів. Скінченний набір символів називається «алфавітом». Структуровані рядки, створені за допомогою цього алфавіту на основі визначених граматичних правил, становлять формальну мову.

Розкриття формальних мов передбачає розуміння ключових понять, притаманних їхній структурі: *алфавіт, рядок і граматика* [7].

Алфавіт: в рамках офіційних мов — алфавіт, який часто позначають грецькою літерою Σ, це просто кінцевий набір різних символів.

Рядок: рядок — це кінцева послідовність символів, вибраних з алфавіту. Важливо, що порядок символів має значення в рядку. Порожній рядок, часто позначається як ^@, це рядок із нульовими символами.

Граматика: Граматика – це набір формальних правил, які керують комбінацією символів для створення рядків у формальній мові. Структурна природа цих правил створення рядків невід'ємно пов'язана з класифікацією формальних мов: необмежені, контекстно-вільні, контекстно-залежні та звичайні.

Ієрархія Хомського та її значення в теорії обчислень

Ієрархія Хомського — це система для класифікації формальних граматик і мов у інформатиці та лінгвістиці. Він складається з чотирьох рівнів, які описують дедалі складніші типи мов, які можуть бути створені формальними граматиками.

Ці рівні: тип 0 (необмежений),

тип 1 (контекстно-залежний),

тип 2 (безконтекстний),

тип 3 (звичайний).

Ієрархія названа на честь Ноама Хомського, який запропонував її як спосіб характеристики виразної сили різних типів формальних мов і граматик. Це фундаментальна концепція у вивченні формальних мов і використовується при розробці алгоритмів розбору та інших інструментів для роботи з формальними мовами [7].

Відповідно до наведеної ієрархії граматика поділяється на чотири типи

Тип 0	Необмежена граматика
Тип 1	Контекстно-залежна граматика
Тип 2	Контекстно-вільна граматика
Тип 3	Звичайна граматика

Тип 0: необмежена граматика

Мова, розпізнана машиною Тьюринга, відома як граматика типу 0. Вони також відомі як рекурсивно перелічувані мови.

Виробництво граматики для типу 0 задано

α —> β

Наприклад:

Sba —> a

S —> B

Де S і B – змінні.

A і b — термінали.

Тип 1: Контекстно-залежна граматика

Мови, які розпізнаються лінійними автоматами , відомі як граматика типу 1 . Контекстно-залежна граматика представляє контекстно-залежні мови.

Щоб граматика була контекстно-залежною, вона має бути необмеженою. Виробництво граматики для типу 1 подано

α —> β (переконайтеся, що символ підрахунку в LHS має бути меншим або дорівнювати RHS)

Наприклад,

S —> BA

BA —> bca

B —> b

Тип 2: Контекстно-вільна граматика

Мови, які розпізнаються Pushdown Automata, відомі як граматика типу 2 . Контекстно-вільна граматика представляє контекстно-вільні мови.

Щоб граматика була контекстно-вільною, вона має бути контекстно-залежною. Виробництво граматики для типу 2 подано

A —> α

Де A — одиночний нетермінал.

Наприклад,

A —> aB

B —> b

Тип 3: звичайна граматика

Мови, які розпізнаються Finite Automata, відомі як граматика типу 3. Звичайна граматика представляє звичайні мови.

Щоб граматика була правильною, вона повинна бути вільною від контексту. Виробництво граматики для типу 3 подано

V → T*V / T*

Наприклад,

A —> ab

Кожному класу мови відповідають певні граматичні форми та обчислювальні моделі [2]. Стратифікація складності мовного класу представлена в таблиці:

Мовний клас	Граматична форма	Обчислювальна модель
Регулярний	Прямолінійний	Скінченний автомат
Безконтекстний	Без обмежень	Пушдаун автомат
Контекстно-залежний	Контекстно-залежний	Лінійно-обмежений автомат

Мовний клас	Граматична форма	Обчислювальна модель
Контекстно-вільний	Без обмежень	Машина Тьюрінга

Після розуміння структури формальних мов вирішальним стає вивчення того, як ними користуватися. Операції на формальних мовах зазвичай аналогічні операціям на множинах. Ось деякі стандартні операції на офіційних мовах, які імітують передбачувані значення у пов’язаних програмах:

Союз: враховуючи дві офіційні мови L1і L2, об'єднання в L1 і L2, позначається як L1∪ L2, містить усі рядки, які знаходяться в L1, або в L2, або в обох.

Конкатенація: конкатенація двох формальних мов L1 і L2, позначається як L1.L2, містить усі рядки, отримані шляхом додавання рядка з L2до рядка з L1.

Зірка: зірка формальної мови L, позначається як L∗, містить усі рядки, отримані конкатенацією будь-якої кінцевої (можливо іншої) кількості рядків із L, включаючи порожній рядок [2].

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Бородкіна І. Теорія алгоритмів. Посібник для студентів вищих навчальних закладів .К.: Центр навчальної літератури, 2019. 184 с.
2. Гаврилків В.М. Формальні мови та алгоритмічні моделі: навч. посібник. івано-Франківськ: Голіней, 2023. 180 с.
3. Матвієнко М. П. Математична логіка та теорія алгоритмів. Навч. посібник / Матвієнко М. П. Шаповалов С. П. – К.: Ліра, 2015.. – 212 с.
4. Нікітченко М.С. Математична логіка та теорія алгоритмів. / Нікітченко М. С.,Шкільняк С. С. — К. : ВПЦ Київський університет, 2008. — 284 с.
5. Прийма С.М. Теорія алгоритмів: Навчальний посібник. – Мелітополь: ФОП Однорог Т.В., 2018. – 116 с.
6. Стусь О.В. Математична логіка та теорія алгоритмів: Лекції [Електронний ресурс] : навч. посіб. для студ. К. : КПІ ім. Ігоря Сікорського, 2017. 150 с.

7. Теорія цифрових автоматів та формальних мов. Вступний курс : навч. посібник / Гавриленко С. Ю., Клименко А. М., Любченко Н.Ю. та ін. Харків : НТУ "ХПІ", 2011. 176 с.

8. Шкільняк, С. С. Математична логіка; Основи теорії алгоритмів : навч. посіб. / С. С. Шкільняк. — К.: ДП «Вид. дім «Персонал», 2009. — 280 с.

Питання для самоконтролю

- Що таке алгоритм?
- Які основні характеристики алгоритму?
- Які основні класи алгоритмів існують?
- Що таке формальна мова?
- Які основні типи формальних мов існують?
- Що таке граматика?
- Які основні типи граматик існують?
- Яка різниця між детермінованим і стохастичним алгоритмом?
- Яка різниця між рекурсивним і нерекурсивним алгоритмом?
- Яка різниця між регулярною і контекстно-вільною мовою?
- Яка різниця між контекстно-вільною і безконтекстною мовою?
- Що таке термінальна множина?
- Що таке нетермінальна множина?
- Що таке правило переходу?
- Яка різниця між теорією алгоритмів та теорією автоматів?
- Яка роль теорії алгоритмів та формальних мов в інформатиці?
- Які приклади застосування теорії алгоритмів та формальних мов в інформатиці?
- Визначити, чи є даний алгоритм детермінованим або стохастичним.
- Визначити, чи є даний алгоритм рекурсивним або нерекурсивним.
- Визначити, чи належить дане слово до формальної мови.
- Побудувати граматику для даної формальної мови.

Тема 6. Основи інформаційного моделювання предметної галузі.

Мета: надати студентам загальне уявлення про інформаційне моделювання предметної галузі, його основні поняття та визначення, розглянути основні етапи та методи інформаційного моделювання предметної галузі, ознайомити студентів із основними видами інформаційних моделей.

План:

- Категорії предметної галузі.
- Багаторівнева система моделювання предметної галузі.
- Інформаційний опис об'єктів предметної галузі.
- Подання інфологічної моделі предметної області ER-діаграмами.

Дані змінюють спосіб функціонування світу. Це може бути дослідження про лікування хвороб, стратегія доходів компанії, ефективне будівництво і або цільова реклама на в соціальних мережах.

Ці дані стосуються інформації, яку читає машина, а не людиною. Наприклад, дані про клієнтів не мають сенсу для команди продукту, якщо вони не вказують на покупки конкретного продукту. А команда маркетингу не використовуватиме ті самі дані, якщо ідентифікатори не стосуватимуться конкретних цін під час покупки.

Саме тут на допомогу приходить моделювання даних. Це процес, який призначає реляційні правила для даних. Модель даних перетворює дані на корисну інформацію, яку організації в предметних галузях потім можуть використовувати для прийняття рішень і стратегії [4].

Дані дозволяють організаціям встановлювати базові показники, контрольні показники та цілі. Для того, щоб дані дозволяли це вимірювання, вони повинні бути організовані через опис даних, семантику даних і обмеження узгодженості даних. Моделювання даних — це процес концептуалізації та візуалізації того, як дані збиратимуться, зберігатимуться та використовуватимуться в предметній галузі. Кінцевою метою моделювання даних є встановлення чітких стандартів даних для всієї галузевої організації [1].

Під предметною галуззю в загальному значенні ми розуміємо множину всіх об’єктів, властивості яких і відношення між якими розглядаються у відповідній галузі (медицині, освіті, інженерії, хімії тощо). В інформатиці під предметною галуззю розуміють високорівневу організацію даних, що представляють групу пов’язаних концепцій у певній функціональній області організації. Приклади предметних областей включають облікові записи, виставлення рахунків, ресурси, процеси та фінанси. Наприклад, коли тематична область «Облікові записи» імпортується в програму, усі таблиці під нею також імпортуються [6].

Моделі даних — це візуальне представлення, яке перетворює абстрактні ідеї у технічний план впровадження. Наприклад, в освітній галузі абстрактна ідея «Ми хочемо відстежувати наші навчальні досягнення в режимі реального часу» у технічний план впровадження «Ми зберігатимемо атрибут під назвою «Навчальні досягнення з предмету географія в 7 – ому класі» у таблиці під назвою «Успішність 7-ого класу» [2].

Моделі даних також допомагають керувати даними та дотримуватись законодавства, а також забезпечують цілісність даних. Вони дозволяють встановлювати стандарти з самого початку проєкту, щоб команди не отримували конфліктуючих наборів даних, які потрібно очистити, перш ніж вони зможуть їх використовувати або, що ще гірше, не зможуть використовувати взагалі.

Створення та підтримка моделі даних в різних предметних галузях має численні переваги:

Модель даних створює загальний комунікаційний рівень, який полегшує спілкування розробника даних і споживачів. Це забезпечує можливість детального обговорення всіх тонкощів, оскільки це візуальна модель із спільною термінологією.

Модель гарантує високу якість реалізованого коду. Код базується на рішеннях попередньої фази документації, зменшуючи кількість помилок.

Розробники можуть витрачати більше часу на розробку функцій, оскільки менше коду потрібно змінювати. У результаті кількість часу, витраченого на кодування, скорочується, а деякі витрати усуваються.

Створення моделі даних дозволяє визначити обсяг проєкту. Завдяки візуальному відображенню моделі даних складність зменшується, а складні теми сприймаються легше.

Технічний рівень додається до моделі даних, яка містить всю технічну інформацію (зазначену розробником даних), що дозволяє розробникам зосередитися на реалізації, а не на інтерпретації.

Завдяки чіткості й точності моделі даних робиться менше помилок. Розробники можуть зосередитися на розробці функцій, а не на розробці бази даних.

Оптимізація структури бази даних зменшує кількість даних, які потрібно перемістити.

Зменшення ризику для даних через наявність стратегій і заходів безпеки [1,3,6,8].

Процес моделювання даних

Організація може мати величезне сховище даних; однак, якщо немає стандарту для забезпечення основної точності та інтерпретації цих даних, тоді це не має користі. Належна модель даних засвідчує ефективні подальші результати, знання найкращих методів роботи з даними та найкращі інструменти для доступу до них.

Моделювання даних у розробці програмного забезпечення — це процес спрощення діаграми або моделі даних програмної системи шляхом застосування певних формальних методів. Це передбачає вираження даних та інформації за допомогою тексту та символів. Модель даних забезпечує схему створення нової бази даних або реінжинірингу застарілих програм [5].

Моделювання даних — це процес створення моделей даних, за допомогою яких асоціації та обмеження даних описуються та зрештою кодуються для

повторного використання. Воно концептуально представляє дані за допомогою діаграм, символів або тексту для візуалізації взаємозв'язку.

Моделювання даних передбачає створення концептуального представлення об'єктів даних та їхніх зв'язків один з одним. Процес моделювання даних зазвичай включає кілька етапів, включаючи збір вимог, концептуальне проектування, логічне проектування, фізичне проектування та впровадження. На кожному етапі процесу розробники моделей даних працюють із зацікавленими сторонами, щоб зрозуміти вимоги до даних, визначити сутності та атрибути, встановити зв'язки між об'єктами даних і створити модель, яка точно представляє дані у спосіб, який може використовуватися програмою. розробники, адміністратори баз даних та інші зацікавлені сторони [4].

Таким чином, моделювання даних допомагає підвищити узгодженість іменування, правил, семантики та безпеки. Це, у свою чергу, покращує аналітику даних. Акцент робиться на необхідності наявності та організації даних незалежно від способу їх застосування.

Проектування баз даних та інформаційних систем, як і будь-який інший процес проектування, починається з високого рівня абстракції та поступово стає більш конкретним і конкретним. Залежно від рівня абстракції, який вони надають, існують три типи моделей даних. Підхід розпочнеться з концептуальної моделі та перейде до логічної моделі, а потім прийде до фізичної моделі.

Типи моделей даних

Концептуальні моделі даних, логічні моделі даних і фізичні моделі даних складають три типи моделей даних. Хоча вони вимагають різних підходів до створення, кожен тип моделі даних передає однакову інформацію з різних точок зору. Застосування всіх трьох типів моделей даних допомагає задовольнити потреби та рівень знань різних зацікавлених сторін [4].

Це пов'язано з тим, що три моделі даних стосуються використання активів даних із різними ступенями абстракції. Моделі стають більш складними –

починаючи з концептуальних і закінчуючи фізичними моделями даних. Моделі використовуються на різних етапах процесу розробки, щоб сприяти узгодженню бізнес-цілей і вимог із тим, як використовуються ресурси даних.

Концептуальні моделі даних використовуються для передачі бізнес-структур, проєктів і концепцій на високому рівні абстракції. Ці моделі будуються без урахування системних обмежень і зазвичай розробляються бізнес-стейкхолдерами та розробниками даних для визначення та організації інформації, необхідної для розробки системи.

Концептуальна модель даних є першою серед типів моделей даних. Вони також відомі як моделі домену , і вони забезпечують загальне уявлення про зміст системи, про те, як вона буде організована та які правила будуть задіяні. Як правило, концептуальні моделі створюються як частина раннього процесу збору вимог до проєкту. Включаються типи елементів, важливі для представлення в моделі даних, їхні характеристики та обмеження, їхні зв’язки, а також відповідні вимоги безпеки та цілісності даних [5].

Логічні моделі даних пов’язані з типами, атрибутами та зв’язками сутностей, які створюватимуть систему. Логічна модель створюється розробником даних і використовується аналітиками. Мета полягає в тому, щоб розробити незалежне від платформи представлення елементів та їхніх зв’язків.

Цей етап моделювання даних дає галузевим організаціям уявлення про обмеження їхніх поточних технологій. Логічна модель даних є другою серед типів моделей даних. Вони менш абстрактні та надають більше інформації про концепції та зв’язки в домені. Логічні моделі даних показують зв’язки між елементами, а також властивості даних, наприклад типи даних та їх відповідну довжину. Технічні потреби системи не вказані в логічних моделях даних. Логічні моделі даних можуть виявитися корисними для проєктів, які за своєю природою орієнтовані на дані, таких як проектування сховищ даних або розробка системи звітності [5].

Фізичні моделі даних використовуються для визначення реалізації логічних моделей даних із застосуванням конкретної системи керування базами

даних (СУБД). Вони побудовані з урахуванням поточних – або очікуваних – технологічних можливостей. Розробники та аналітики баз даних працюють із фізичними моделями даних, щоб реалізувати ідеї та процеси, вдосконалені концептуальними та логічними моделями.

Фізична модель даних є останньою серед типів моделей даних. Вона визначає формат, у якому дані фізично зберігатимуться в базі даних. Як наслідок, вони є найменш абстрактними з групи. Фізичні моделі даних забезпечують готовий дизайн, який можна реалізувати як реляційну базу даних із асоціативними таблицями, які показують асоціації між елементами. Первинні ключі та зовнішні ключі будуть використовуватися для підтримки цих зв'язків в актуальному стані [5].

Концепції та вимоги до системи уточнюються на кожному кроці, коли вони переходять від концептуальних моделей до логічних і затверджуються у фізичних моделях.

Методи моделювання даних

По-перше, існує діаграма сутності-зв'язку або техніка ERD для моделювання та проектування реляційних або традиційних баз даних. По-друге, UML або діаграми класів уніфікованої мови моделювання є стандартизованою сім'єю нотацій для моделювання та проектування інформаційних систем. І, нарешті, третій — це техніка моделювання словника даних, де виконується табличне визначення або представлення ресурсів даних

Оскільки потреби предметних галузей у сховищах даних розширилися, моделювання даних розвивалося разом із системами керування базами даних, а типи моделей ускладнювалися [7].

Ієрархічні моделі даних. У деревоподібному стилі ієрархічні моделі даних представляють зв'язки «один до багатьох». Кожен запис у цій структурі має одну кореневу або материнську таблицю, яка перетворюється на одну або декілька дочірніх таблиць. Ця парадигма була прийнята в IBM Information Management System (IMS), яка була випущена в 1966 році і швидко стала популярною, особливо в банківській галузі. Незважаючи на те, що цей метод є

менш ефективним, ніж нещодавно створені моделі баз даних, він, тим не менш, використовується в системах XML і географічних інформаційних системах (ГІС).

Реляційні моделі даних: Е. Ф. Кодд, дослідник IBM, уперше представив реляційні моделі даних у 1970 році. Вони все ще використовуються в сучасних реляційних базах даних, які широко використовуються в корпоративних обчисленнях. Реляційне моделювання даних не потребує досконалого знання фізичних аспектів системи зберігання даних. Це зменшує складність бази даних, явно об'єднуючи сегменти даних за допомогою таблиць.

Моделі даних сутності та зв'язку (ER): моделі даних ER представляють зв'язки між сутностями в базі даних за допомогою формальних діаграм. Розробники даних використовують різноманітні інструменти моделювання ER для створення візуальних карт, які передають цілі структури бази даних.

Об'єктно-орієнтовані моделі даних: ці різні типи моделей даних стали популярними одночасно з об'єктно-орієнтованим програмуванням у середині 1990-х років. Предмети, про які йде мова, є фіктивними представленнями об'єктів реального світу. Об'єкти організовані в ієрархії класів, кожен із яких має власний набір атрибутів. Таблиці можна включати в об'єктно-орієнтовані бази даних, але вони також можуть обробляти більш складні взаємодії з даними. Цей метод використовується в мультимедійних і гіпертекстових базах даних, серед інших програм.

Моделі розмірних даних: Ральф Кімбол створив моделі розмірних даних , які були створені для прискорення пошуку даних для аналітичних цілей у сховищі даних. У той час як реляційні та ER моделі зосереджені на ефективному зберіганні, розмірні моделі включають резервування, щоб полегшити пошук інформації для звітування та пошуку. Цей тип моделювання поширений у системах OLAP.

Зіркова схема, у якій дані структуровані на факти (вимірювані елементи) і виміри (довідкова інформація), і кожен факт обведений відповідними вимірами у вигляді зірки, є значимою моделлю вимірних даних. Інша схема — це схема

сніжинки, яка схожа на схему зірки, але включає додаткові шари пов'язаних розмірів, що збільшує складність рисунка розгалуження.

Інструменти моделювання даних

Моделювання даних — це процес застосування певних технік і методологій до даних з метою перетворення їх у корисну форму. Це робиться за допомогою інструментів моделювання даних, які допомагають створити структуру бази даних із схем. Це полегшує підключення даних і формує ідеальну структуру даних відповідно до вимог. Сьогодні широко використовуються різноманітні комерційні та безкоштовні рішення автоматизованої розробки програмного забезпечення (CASE), включаючи інструменти моделювання даних, діаграм та візуалізації [4]. Наприклад:

Erwin Data Modeler — це інструмент моделювання даних, який підтримує альтернативні підходи до позначень, у тому числі розмірний підхід, і базується на мові моделювання даних Integration DEFinition for information modeling (IDEF1X).

Enterprise Architect — це інструмент візуального моделювання та проєктування корпоративних інформаційних систем, програмних додатків і баз даних. Він побудований на основі об'єктно-орієнтованих мов і стандартів.

ER/Studio — це програмне забезпечення для проєктування баз даних, яке працює з кількома поширеними сьогодні системами керування базами даних. Підтримуються як реляційна, так і розмірна моделі даних.

Рішення з відкритим кодом, такі як Open ModelSphere , є прикладами безкоштовних інструментів моделювання даних.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ананьєв О.М. Інформаційні системи і технології в комерційній діяльності: підручник / О.М. Ананьєв, В.М. Білик, Я.А. Гончарук. Львів: Новий Світ, 2006. 583 с.
2. Биков В.Ю. Моделі організаційних систем відкритої освіти. К.: Атіка, 2009. 684 с.
3. Биков В.Ю., Руденко В.Д. Системи управління інформаційними базами

даних в освіті. К.: ІЗМН, 1996. 288 с.

4. Вітюк О. В., Гуралюк А. Г. , Москалькова Н. М., Шикова О. М. Основи інформатики. К.: МАУП, 2005.
5. Дибкова Л.М. Інформатика і комп'ютерна техніка: навч. пос. К.: Академія, 2005.416с.
6. Глівенко С.В. Інформаційні системи в менеджменті: навч. посіб. Суми: Університетська книга, 2005. 407 с.
7. Інформатика: 11 кл.: підруч. для загальноосвіт. навч. закл.: рівень стандарту / Й.Я. Ривкінд, Т.І. Лисенко, Л.А. Чернікова, В.В. Шакотько; за заг. ред. М.З. Згуровського. К.: Генеза, 2011. 304 с.
8. Філатова Т.В. Інформаційне моделювання предметних областей як елемент оптимального організаційного управління. URL: http://dspace.opu.ua/jspui/bitstream/123456789/7263/1/7_Informative%20model%20of%20subject%20domains%20as%20element%20of%20optimum%20organizational%20control.pdf (дата звернення: 27.09.2023).

Питання для самоконтролю

- Що таке інформаційне моделювання предметної галузі?
- Які етапи інформаційного моделювання предметної галузі існують?
- Які основні методи інформаційного моделювання предметної галузі існують?
- Які основні види інформаційних моделей існують?
- Яка різниця між інформаційним моделюванням і моделюванням предметної галузі?
- Які основні цілі інформаційного моделювання предметної галузі?
- Які основні переваги інформаційного моделювання предметної галузі?
- Які основні труднощі інформаційного моделювання предметної галузі?
- Які приклади застосування інформаційного моделювання предметної галузі?
- Які перспективи розвитку інформаційного моделювання предметної галузі?

Завдання до самостійної роботи

1. Передумови виникнення інформатики.
2. Основні етапи розвитку комп’ютерної техніки.
3. Системи числення.
4. Властивості та види інформації.
5. Класифікація та покоління ЕОМ.
6. Огляд сучасних комп’ютерів.
7. Способи представлення та одиниці вимірювання даних у ПЕОМ.
8. Робота з тренажерами клавіатур.
9. Перелічіть складові комп’ютерних систем. Опишіть загальну картину функціонування комп'ютерної системи.
10. З якою метою організована таблиця стану пристроїв?
11. Опишіть структуру пам'яті комп’ютера.
12. Перелічіть елементи ієрархії пам’яті.
13. Як організовано апаратний захист пам'яті й процесора?
14. Як класифікуються сучасні комп'ютерні системи?
15. Що представляють собою кластери комп’ютерів?
16. Перелічіть особливості CISC-архітектури.
17. Перелічіть особливості RISC-архітектури.
18. Перелічіть особливості VLIW-архітектури.
19. Перелічіть особливості EPIC-архітектури.
20. Які основні функції операційної системи? Чи немає між ними протиріч?
21. Перелічіть основні компоненти операційної системи.
22. Що спільного й у чому відмінності між мережною і розподіленою операційними системами? Яка з них складніша в реалізації і чому?
23. Перелічіть види серверів у клієнт-серверних операційних системах
24. Коли використовують дерева рішень?
25. Перелічіть способи пошуку для малих та великих дерев.
26. Як за допомогою дерев гри моделюється гра в хрестики - нулики?

27. У чому зміст мінімаксної стратегії?

28. Що таке евристика? Що спільне у евристичних алгоритмів?

29. Поясніть сенс метода гілок та меж.

30. Що працює швидше евристичні методи чи метод гілок та меж?

31. Порівняйте стратегії сходження на пагорб та найменшої вартості.

32. Поясніть алгоритм методу відпалу.

33. У чому ідея методу збалансованого прибутку? Чим характеризуються динамічні структури даних?

34. Назвіть складні завдання, в рішенні яких застосовують евристики.

35. Як формулюється задача про розбивку?

36. Як моделюється задача про пожежні депо?

37. Скільки листів може мати дерево рішень для завдання про здійснимість?

38. Евристики або метод гілок та меж використовують для пошуку приблизного рішення в завданні про здійснимість? Що таке дерево рішень в задачі комівояжера, яка вирішується методом гілок і меж?

39. Дати визначення процесу сортування.

40. Наведіть найбільш відомі елементарні методи сортування.

41. Поясніть, чому сортування є завданням із точними теоретичними обмеженнями продуктивності.

42. Яку оцінку ефективності мають прості методи сортування?

43. Для чого використовують таблиці індексів?

44. У чому перевага методу сортування вибором?

45. Яку складність має алгоритм сортування вставкою?

46. Порівняйте бульбашкове сортування та швидке сортування.

47. Дати визначення поняттю піраміда у пірамідальному сортуванні.

48. Назвіть головні етапи алгоритму пірамідального сортування?

49. Як виконується побудова піраміди в алгоритмі пірамідального сортування?

50. Яку оцінку ефективності має алгоритм пірамідального сортування?

51. Що таке сортування підрахунком?

52. Чим відрізняються блокове сортування та блокове сортування із застосування зв’язного списку?

53. В яких випадках прості алгоритми сортування можуть бути більш прийнятними, ніж вдосконалені?

54. Відсортувати файл цілих чисел методом вставки, не використовуючи масив.

55. Реалізувати програмно-пірамідальне сортування.

56. До підготовленої статті автор додав список використаної літератури, але розташував видання в порядку появи посилань на них в тексті. Редактор зажадав розташувати джерела за абеткою. Упорядкувати список літератури на вимогу редактора (кожне видання - з нового абзацу: номер п/п, автор, назва роботи і т.д.).

57. Поясніть алгоритм методу повного перебору. Як модифікувати алгоритм повного перебору, щоб підвищити його продуктивність?

58. Чим відрізняються пошуки в упорядкованих та зв’язних списках ?

59. В чому сенс двійкового пошуку? Назвіть переваги двійкового пошуку.

60. Що таке бінарне дерево пошуку?

61. Поясніть сенс інтерполяційного пошуку, коли доцільне його використання?

62. Коли переважніше використання дерева пошуку ніж сортованого масиву?

63. Що таке вироджене дерево пошуку і як воно може виникнути?

Список рекомендованої літератури (основна та додаткова, Інтернет ресурси)

Основна:

1. Брикайло Л. Ф. Інформатика та комп'ютерна техніка : навч. посіб. Л.Ф. Брикайло. К. : Вид. ПАЛИВОДА А. В., 2019. 266 с.
2. Дибкова Л. М. Інформатика і комп'ютерна техніка : навчальний посібник [для студентів вищих навч. закладів] Л. М. Дибкова. [вид. 2-е,]. К. : Академвидав, 2017. 416 с.
3. Злобін Г. Г. Основи інформатики, комп'ютерної техніки і комп'ютерних технологій : підручник. К. : Каравела, 2017. 240 с.
4. Інформатика та комп'ютерна техніка: навч. посіб. Л. Ф. Брикайло. К. : Вид. ПАЛИВОДА А. В., 2019. 266 с.
5. Литвин І. І. Інформатика: теоретичні основи і практикум : підручник. [2-ге вид., стереотип.] І. І. Литвин, О. М. Конопчук, Ю. Д. Дещинський. Львів : «Новий Світ – 2000», 2017. 304 с.

Додаткова:

1. Інформатика: комп'ютерна техніка [М. Є. Рогоза, Л. Ф. Крещенко, В. І. Клименко, О. І. Корх; за ред. М. Є. Рогоза]. К. : Академія, 2016. 365 с.
2. Інформатика: Комп'ютерна техніка. Комп'ютерні технології : підручник для студ. Вузів. В. А. Баженов, П. С. Венгерський, В. М. Горлач та ін. [2-е вид.]. К. : Каравела, 2017. 640 с.
3. Кравчук С. О. Основи комп'ютерної техніки : Компоненти, системи, мережі. К. : Каравела, 2016. 344 с.
4. Макарова М. В. Інформатика та комп'ютерна техніка. Навч. посібник для студентів ВНЗ напряму „Економіка і підприємництво" з грифом МОН України. М. В. Макарова, Г. В. Карнаухова, С. В. Запара; за ред. М. В. Макарової. Суми : ВТД «Університетська книга», 2017. 665 с.
5. Морзе Н. В. Основи інформаційно-комунікаційних технологій. Н. В. Морзе. К. : Видавнича група BHV, 2016. 352 с.
6. Основи інформатики: навч. посіб. [О. В. Вітюк, А. Г. Гуралюк, Н.

М. Москалькова, О. М. Шикова]. К. : МАУП, 2015. 104 с.

Інтернет ресурси:

1. Jamboard URL:https://jamboard.google.com
2. Wiki сервіс URL:https://uk.wikipedia.org/wiki
3. Блогер (для створення блогів)URL:https://www.blogger.com
4. Видавництво «Ранок» URL:https://www.ranok.com.ua -
5. Вимоги до уроку інформатики // Все на урок інформатики. – Режим доступу: http://urokinformatiku.ru/vimogi-do-uroku-informatiki
6. Всеосвіта URL:https://vseosvita.ua –
7. Електронна бібліотека Житомирського державного університету. URL: http://eprints.zu.edu.ua/.
8. Електронна бібліотека НАПН України. URL: http://lib.iitta.gov.ua/.
9. Електронна енциклопедія Wikipedia. URL: https://uk.wikipedia.org/.
10. Журнал "Інформаційні технології. Аналітичні матеріали"URL:http://it.ridne.net –
11. Нова українська школа URL:https://nus.org.ua -
12. Онляндія URL:http://disted.edu.vn.ua/media/bp/html/etusivu.htm
13. Педагогічна преса URL:https://pedpresa.ua -
14. Портал ZDU PROJECT. URL: https://project.zu.edu.ua.
15. Середовище для створення кросвордів URL:https://www.armoredpenguin.com/crossword/
16. Словники URL:http://slovopedia.org.ua/
17. Створення вправ URL:https://learningapps.org
18. Створення хмар URL:https://worditout.com

Навчально-методичне видання

ЖУКОВСЬКИЙ Сергій Станіславович

КРИВОНОС Олександр Миколайович

МІНГАЛЬОВА Юлія Ігорівна

ШЕВЧУК Петро Георгійович

Конспект лекцій освітньої компоненти «Теоретичні основи інформатики» для здобувачів другого (магістерського) рівня вищої освіти спеціальності 014 Середня освіта за спеціалізацією 014.09 Середня освіта (Інформатика)

Видавництво Житомирського державного університету імені Івана Франка

10008, м. Житомир, вул. Велика Бердичівська, 40

Свідоцтво суб'єкта видавничої справи:

ЖТ № 10 від 07.12.2004 р.

електронна пошта (E-mail): zu@zu.edu.ua