МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

М. В. Добролюбова

ОБЧИСЛЮВАЛЬНА ТЕХНІКА, ОСНОВИ АЛГОРИТМІЗАЦІЇ ТА ПРОГРАМУВАННЯ

КОНСПЕКТ ЛЕКЦІЙ

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для здобувачів ступеня бакалавра за освітньою програмою
«Інформаційні вимірювальні технології»
спеціальності 152 «Метрологія та інформаційно-вимірювальна техніка»

Київ
КПІ ім. Ігоря Сікорського
2021

Рецензент *Попов, А. О.*, к.т.н., доцент, доцент, кафедра електронної інженерії, ФЕЛ, КПІ ім. Ігоря Сікорського

Данильченко, Т. В., к.т.н., доцент, доцент, кафедра інформаційних систем в економіці, ІІТЕ, КНЕУ імені Вадима Гетьмана

Відповідальний редактор *Шевченко, К. Л.*, д.т.н., доцент

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 1 від 16.09.2021 р.) за поданням Вченої ради Приладобудівного факультету (протокол № 7/21 від 30.08.2021 р.)

Електронне мережне навчальне видання

Добролюбова Марина Валеріївна, канд. техн. наук, доц.

ОБЧИСЛЮВАЛЬНА ТЕХНІКА ОСНОВИ АЛГОРИТМІЗАЦІЇ ТА ПРОГРАМУВАННЯ

КОНСПЕКТ ЛЕКЦІЙ

Обчислювальна техніка, основи алгоритмізації та програмування. Конспект лекцій [Електронний ресурс] : навч. посіб. для студ. спеціальності 152 «Метрологія та інформаційно-вимірювальна техніка» / М. В. Добролюбова ; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 51,2 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 417 с.

Навчальний посібник забезпечує лекційний курс з кредитного модуля «Обчислювальна техніка, основи алгоритмізації та програмування» дисципліни «Обчислювальна техніка та програмування».

Навчальний посібник знайомить студентів з фундаментальними поняттями в області обчислювальної техніки, сучасних інформаційних технологій, алгоритмізації і програмування, які необхідні при проєктуванні та експлуатації інформаційно-вимірювальних систем і приладів. Чітка структурованість лекцій допоможе студентові швидше і легше оволодіти пропонованим матеріалом та застосовувати набуті уміння при розв'язанні за допомогою ПК реальних науково-технічних задач різного ступеня складності.

ПЕРЕДМОВА

Курс лекцій з кредитного модуля «Обчислювальна техніка, основи алгоритмізації та програмування» розроблений на основі багаторічного досвіду викладання автором основ обчислювальної техніки та програмування.

Навчальний посібник відповідає програмам з обчислювальної техніки та програмування бакалаврів першого курсу за спеціальністю 152 «Метрологія та інформаційно-вимірювальна техніка».

Курс лекцій призначений для самостійного опанування студентами таких тем, як поняття інформації, персональний комп'ютер, структури та алгоритми даних.

Навчальний посібник містить чітко викладений теоретичний матеріал, комплект презентаційних матеріалів та необхідні для закріплення знань контрольні питання до кожної лекції.

ЗМІСТ

РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА. ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Тема 1.1 ПОНЯТТЯ ІНФОРМАЦІЇ

Лекція 1

ВСТУП. ПРОФЕСІЙНІ ТЕРМІНОЛОГІЧНІ ВИЗНАЧЕННЯ В ОБЛАСТІ ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ, ІНФОРМАТИКИ ТА ПРОГРАМУВАННЯ

Мета лекції: розгляд термінології в області обчислювальної техніки, інформатики та програмування; вивчення принципів дискретизації та кодування сигналів.

Зміст лекції

1. Зміст та значення курсу. Професійні термінологічні визначення в області обчислювальної техніки, інформатики та програмування. Інформація.
2. Дискретизація та кодування сигналів.
   - 2.1. Класифікація кодів і їх характеристики.
   - 2.2. Системи числення.
   - 2.3. Формати надання інформації.

Професійні термінологічні визначення в області обчислювальної техніки, інформатики та програмування. Інформація

- *Обчислювальна техніка* – пристрій або система, здатний виконувати задану, чітко визначену послідовність операцій (введення-виведення, числові розрахунки, маніпулювання даними).
- *Програмування* – процес проєктування, написання, тестування та підтримки комп'ютерних програм.
- *Інформація* – відомості про осіб, предмети, факти, події, явища і процеси незалежно від форми їх надання.
- *Документована інформація* (документ) – зафіксована на матеріальному носії інформація з реквізитами, що дозволяють її ідентифікувати;
- *Інформаційні процеси* – процеси збору, обробки, накопичення, зберігання, пошуку і розповсюдження інформації.
- *Інформаційна система (ІС)* – організаційно впорядкована сукупність документів (масивів документів) і інформаційних технологій, у тому числі з використанням засобів обчислювальної техніки і зв'язку, що реалізовує інформаційні процеси.
- *Інформаційні ресурси (ІР)* – окремі документи і окремі масиви документів, документи і масиви документів в інформаційних системах (бібліотеках, архівах, фондах, банках даних, інших інформаційних системах).

- *Групи ІР:*
  - інформація на фізичних носіях, а також засоби її обробки;
  - засоби передачі і комунікації, включаючи навички і прийоми їх використання;
  - засоби обробки інформації;
  - вчені і фахівці – «виробники» інформації в різних сферах діяльності;
  - органи і служби, що займаються збором, обробкою і зберіганням інформації.
- *Інформаційні технології (ІТ)* – це засновані на застосуванні персонального комп'ютера (ПК) способи обробки семантичної інформації – даних і знань, які реалізуються за допомогою автоматизованих інформаційних систем.
- Дані можуть розглядатися як ознаки або записані спостереження, які із якихось причин не використовуються, а тільки зберігаються. В тому випадку, якщо з'являється можливість використовувати ці дані для зменшення невизначеності про що-небудь, дані перетворюються на інформацію.
- В процесі обробки інформація може змінювати *структуру* і *форму*. Ознаками структури є елементи інформації і їх взаємозв'язок. Розрізняють *змістовну* і *формальну* структури.
- *Змістовна структура* природно орієнтована на зміст інформації, а *формальна* – на форму надання інформації. Основні форми надання інформації:
  - символьна (заснована на використанні символів – букв, цифр, знаків),
  - текстова (використовує тексти – символи, розташовані в певному порядку),
  - графічна (різні види зображень),
  - звукова.
- Залежно від області знань розрізняють наукову, технічну, виробничу, правову і іншу інформацію.
- *Інформатика* – це сукупність засобів всієї сучасної інформаційної теорії, техніки і технології, сумарне, комплексне позначення цієї області знань.
- *Кібернетика* існує незалежно від наявності або відсутності комп'ютерів. Кібернетика і інформатика, дуже схожі дисципліни, але розрізняються в наголосі на різні акценти:
  - в інформатиці – на властивостях інформації і апаратно-програмних засобах її обробки;
  - в кібернетиці – на розробці концепцій і побудові моделей об'єктів з використанням, зокрема, інформаційного підходу.

Дискретизація та кодування сигналів

Класифікація кодів і їх характеристики

- *Кодування* – це процес перетворення повідомлень в упорядковану послідовність символів (знаків, елементів) одного алфавіту.

- *Кодова комбінація* – це впорядкований набір символів (знаків, елементів) одного алфавіту.
- *Код* – сукупність кодових комбінацій, побудованих за одним правилом і на основі одного алфавіту.
- За ***потужністю*** алфавіту (кількістю q символів алфавіту) коди розділяють на *двійкові* (q=2) та *недвійкові* (q>2). Останні називають ще багатопозиційними або багатоосновними.
- За ***коректувальною здатністю*** коди розділяють на *безнадмірні* та *надмірні*. До *безнадмірних* кодів належать так звані прості або первинні коди, які використовують для первинного кодування джерел повідомлень. Ці коди не дозволяють виявляти і виправляти помилки. Це пояснюється тим, що такі коди використовують всі можливі комбінації і будь-яка помилка призводить до появи дозволеної кодової комбінації. До *надмірних* кодів належать коди, які дозволяють виявляти та/або виправляти помилки. У цих кодах використовується тільки визначена частина можливих комбінацій, що дозволяє при спотворенні елементів кодових комбінацій виявити або виправити їх. Кількість виявлених або виправлених помилок буде залежати від коректувальної здатності коду, тобто від його надмірності та особливостей побудови.
- За ***способом побудови*** коди розділяють на *блокові* та *неперервні*. До першої групи належать коди, які є сукупністю кодових комбінацій, а до другої – коди, для яких кодування і декодування становлять неперервний у часі процес.
- Блокові коди ***за кількістю елементів у кодових комбінаціях*** коду розділяють на *рівномірні* та *нерівномірні*. До першої групи належать коди, у яких всі комбінації, що складають код, мають однакову кількість елементів, а до другої – ті, у яких кодові комбінації коду можуть містити різну кількість елементів.
- Блокові коди ***за використанням перевірочних елементів*** у кодових комбінаціях розділяють на *роздільні* та *нероздільні*. До першої групи належать коди, кодові комбінації яких містять інформаційні та перевірочні елементи, до другої – коди, у кодових комбінаціях яких не відокремлюються інформаційні та перевірочні елементи.
- Блокові коди ***за способом побудови перевірочних елементів*** у кодових комбінаціях розділяють на *лінійні* (*групові*, *систематичні*) та *нелінійні* (*несистематичні*). До першої групи належать коди, у яких перевірочні елементи отримують як результат лінійних операцій над визначеними інформаційними елементами (для двійкових кодів за модулем 2), а до другої – коди, що будуються за іншими принципами.
- Різновидом лінійних (систематичних) кодів є *циклічні* коди, для котрих характерною є така особливість: циклічний зсув будь-якої комбінації коду дає комбінацію, яка завжди належить до цього коду.
- Основні характеристики кодів

- довжина коду – кількість елементів (символів, знаків), які складають кодову комбінацію;
- кількість інформаційних елементів;
- кількість перевірочних елементів (для коректувальних роздільних кодів);
- алфавіт коду Q – множина символів (знаків), що різняться між собою, яка використовується для побудови кодових комбінацій коду (для двійкових кодів);
- потужність алфавіту коду – кількість символів (знаків) алфавіту (для двійкового коду);
- потужність коду – кількість дозволених кодових комбінацій, які використовуються для передачі повідомлень (для блокових роздільних кодів у загальному вигляді, зокрема для двійкових кодів);
- повна кількість кодових комбінацій N – кількість всіх можливих комбінацій для даного коду;
- надмірність коду;
- швидкість коду;
- вага кодової комбінації – для двійкового коду визначається кількістю одиниць у кодовій комбінації;
- кодова відстань між двома кодовими комбінаціями однакової довжини – визначається як кількість одноіменних елементів (розрядів) з різними значеннями символів (відстань Хеммінга);
- мінімальна кодова відстань – визначається для коду в цілому як мінімальне значення кодових відстаней між усіма парами кодових комбінацій, що належать до даного коду. Мінімальна кодова відстань визначає його здатність виявляти та виправляти помилки.

Системи числення

- В загальному випадку число N в деякій позиційній системі числення з основою P записується як

$$N = \sum_{i=-s}^{m-1} a_i P^i ,$$

де a – цифра від 0 до *P-1*,
P – основа системи числення.

- Позиційними системами числення називаються такі, в яких вага кожної цифри a залежить від позиції в зображенні числа.
- Максимальне ціле число, яке може бути надане в m розрядах:

$$N_{\max} = P^m - 1.$$

- Мінімальне значуще (не рівне 0) число, яке можна записати в s розрядах дробової частини:

$$N_{\min} = P^{-s}.$$

- Розрізняють двійкову, вісімкову, десяткову, шіснадцяткову системи числення тощо.

Формати надання інформації

Формати надання чисел

- В обчислювальних машинах застосовуються дві форми надання двійкових чисел:
  - природна форма або форма з фіксованою комою (крапкою);
  - нормальна форма або форма з плаваючою комою (крапкою).
- З фіксованою комою всі числа зображаються у вигляді послідовності цифр з постійним для всіх чисел положенням коми, що розділяє цілу частину від дробової.
- З плаваючою комою кожне число зображається у вигляді двох груп цифр. Перша група цифр називається мантисою, друга – порядком, причому абсолютна величина мантиси повинна бути менше 1, а порядок – цілим числом. В загальному вигляді число у формі з плаваючою комою може бути надано таким чином:

$$N = \pm MP^{\pm r},$$

де M – мантіса числа ($|M| < 1$);
r – порядок числа (r – ціле число);
P – основа системи числення.

- Знак числа зазвичай кодується двійковою цифрою, при цьому код 0 означає знак «+», код 1 – знак «-».
- Для алгебраїчного надання чисел (тобто для надання додатних і від'ємних чисел) в машинах використовуються спеціальні коди: прямий, зворотний і додатковий. Зворотний і додатковий дозволяють замінити операцію віднімання на операцію додавання з від'ємним числом; додатковий код забезпечує більш швидке виконання операцій.
- Одиниці вимірювання об'ємів інформації, збереженої або оброблюваної в ПК:
  - 1 біт = 1 двійковий розряд,
  - 1 байт = 8 біт,
  - 1 слово = 2 байти,
  - 1 кілобайт (Кбайт або Кб або К) = 2^{10} байт = 1024 байт,
  - 1 мегабайт (Мбайт або Мб або М) = 2^{20} байт = 1024 Кбайт = 1 048 576 байт,
  - 1 гігабайт (Гбайт або Гб або Г) = 2^{30} байт = 1024 Мбайт = 220 Кбайт = 1 073 741 824 байт,
  - 1 терабайт (Тбайт або Тб або Т) = 2^{40}байт = 1024 Гбайт = 220 Мбайт = 1 099 511 627 776 байт.
- Послідовність декількох бітів або байтів часто називають полями даних. Біти в числі (в слові, в полі тощо) нумеруються справа наліво, починаючи з 0-го розряду.
- В ПК можуть оброблятися поля постійної і змінної довжини. Поля постійної довжини:
  - слово – 2 байти,

- подвійне слово – 4 байти,
- розширене слово – 8 байт.

- Числа з фіксованою комою частіше за все мають формат слова і байта, числа з плаваючою комою – формат подвійного слова (*Single*), слова з 6 байт (*Real*), розширеного слова (*Double*) і слова з 10 байт (*Extended*).
- Поля змінної довжини можуть мати будь-який розмір від 0 до 256 байт, але обов'язково рівний цілому числу байтів.

Надання текстової інформації

- Код ASCII (*American Standard Coding for Information Interchange)* – американський стандартний код для обміну інформацією має основний стандарт і його розширення. Основний стандарт для кодування символів використовує шістнадцятирічні коди 00 - 7F (десяткові 0 – 127), розширення стандарту – 80 - FF (десяткові коди 128 – 255).
- Unicode розроблений в результаті об’єднаних зусиль декількох провідних фірм-виробників програмного і апаратного забезпечення. Unicode включає 65 536 різних двійкових кодів, що цілком достатньо навіть для надання всіх китайських і японських алфавітів, які широко використовуються.
- Міжнародна організація із стандартизації (International Organization for Standardization, ISO, від грецького isos – однаковий) розробила код, в якому для виразу символів використовуються комбінації з 32 біт, внаслідок чого цей код дозволяє надати більше 17 мільйонів символів.

Надання зображень

- *Растрові методи* (bitmap techniques) представляють зображення як сукупність крапок, званих *пікселями* (pixel, скорочення від picture element – елемент зображення).
- *Векторні методи* (vector techniques) дозволяють уникнути проблем масштабування, характерних для растрових методів. Зображення надається у вигляді сукупності ліній і кривих.

Надання звуку

- При найпоширенішому способі кодування звукової інформації амплітуда сигналу вимірюється через рівні проміжки часу і записуються отримані значення.

Закріплення матеріалу

1. Роз’яснити поняття «обчислювальна техніка», «програмування», «інформація», «інформаційна система», «інформаційний ресурс», «інформаційні технології».
2. Навести класифікацію кодів.
3. Які основні характеристики кодів вам відомі?
4. Які системи числення вам відомі?
5. Які формати надання числової інформації вам відомі?

РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА. ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Тема 1.1 ПОНЯТТЯ ІНФОРМАЦІЇ

Лекція 2

СТИСНЕННЯ ДАНИХ

Мета лекції: вивчення основних методів стиснення інформації та їх характеристик; розгляд основних принципів оптимального кодування.

Зміст лекції

1. Основні методи стиснення інформації та їх характеристики.
    1.1. Поняття про стиснення та архівування.
    1.2. Основні алгоритми (методи) стиснення даних та їх узагальнена класифікація.
2. Основні принципи оптимального кодування.
3. Стиснення даних у комп'ютерних інформаційних технологіях.

Основні методи стиснення інформації та їх характеристики

Поняття про стиснення та архівування

- Фактори, які впливають на ступінь надлишковості даних: тип даних, прийнята система кодування.
- Надлишковість даних призводить до зростання вартості зберігання та передачі інформації, тому виникає проблема позбавлення надлишковості або стиснення/архівування даних.
- *Архів* – стиснений варіант даних
- *Архіватори* – програмні засоби, що реалізують методи стиснення.
- *Стиснення (архівування) файлів*: використовується для зменшення розмірів файлів при підготовці їх до передавання каналами зв'язку або до транспортування на зовнішніх носіях;
- *Стиснення (архівування) папок*: використовується як засіб зменшення обсягу папок перед довготерміновим зберіганням;
- *Стиснення (ущільнення) дисків*: використовується для підвищення ефективності використання дискового простору шляхом стиснення даних при запису їх на носії інформації.

Основні алгоритми (методи) стиснення даних та їх узагальнена класифікація

- Теоретичні способи зменшення надлишковості даних: зміна вмісту даних, зміна структури даних, зміна вмісту і структури.

- Якщо при стисненні даних відбувається зміна їх вмісту, то метод стиснення є незворотнім. Такі методи часто називаються методами стиснення з регульованими втратами інформації.
- Якщо при стисненні даних відбувається тільки зміна структури даних, то метод стиснення є зворотнім. Такі методи часто називаються методами стиснення без втрати інформації.
- Алгоритми, що є основою методів стиснення без втрати інформації:
- 1) алгоритм кодування довжин серій RLE (Run Length Encoding);
- 2) словникові алгоритми або алгоритми групи KWE(KeyWord Encoding);
- 3) алгоритм Хаффмена.

Алгоритм RLE (метод кодування довжин серій)

- В основі алгоритму RLE лежить ідея виявлення послідовностей даних, що повторюються, та заміни цих послідовностей більш простою структурою, в якій вказується код даних та коефіцієнт повторення.
- *Коефіцієнт стиснення* – визначається як відношення обсягу вихідних нестислих даних до обсягу стислих:

$$k = \frac{S_0}{S_c},$$

де k – коефіцієнт стиснення,

S_0 – обсяг вихідних даних,

S_c – обсяг стислих.

- Таким чином, чим вище коефіцієнт стиснення, тим алгоритм ефективніше.
- Якщо k = 1, то алгоритм не робить стиснення, тобто вихідне повідомлення виявляється за обсягом рівним вхідному. Якщо k < 1, то алгоритм породжує повідомлення більшого розміру, ніж нестиснене, тобто, здійснює «шкідливу» роботу.
- Алгоритм RLE буде давати кращий ефект стиснення при більшій довжині послідовності даних, що повторюються.
- Недолік алгоритму RLE: низька пристосованість до багатьох розповсюджених типів файлів.
- Алгоритм можна ефективно використовувати лише у комбінації з вторинним кодуванням.

Алгоритми групи KWE (словниковий метод)

- В основу алгоритмів стиснення за ключовими словами покладено принцип кодування лексичних одиниць групами байт фіксованої довжини. Існує досить багато реалізацій таких алгоритмів, серед яких найбільш поширеними є алгоритм Лемпеля-Зіва (алгоритм LZ) та його модифікація алгоритм Лемпеля-Зіва-Велча (алгоритм LZW).
- Алгоритм LZW побудований навколо таблиці фраз (словника), яка відображає рядки символів стискаємого повідомлення в коди фіксованої довжини. Таблиця володіє так званою властивістю передування, тобто для

кожної фрази словника, що складається з деякої фрази *w* і символу *K* фраза *w* також міститься в словнику. Якщо всі частинки словника повністю заповнені, кодування перестає бути адаптивним.

- Суть алгоритму:
  - словником є потенційно нескінченний список фраз;
  - алгоритм починає роботу з майже порожнього словника з одним закодованим рядком;
  - коли зчитується черговий символ вхідної послідовності даних, він додається до поточного рядка;
  - процес продовжується доти, поки поточний рядок відповідає деякій фразі з словника;
  - коли поточний рядок є останнім збігом зі словником плюс щойно прочитаним символом повідомлення, кодер видає код, що складається з індексу збігу і наступного за ним символу, що порушив збіг рядків;
  - нова фраза, що складається з індексу збігу і наступного за ним символу, додається до словника;
  - наступного разу, коли ця фраза з'явиться в повідомленні, вона може бути використана для побудови більш довгої фрази, що підвищує міру стиснення інформації.

Алгоритм Хаффмена (ентропійний метод)

- В основі алгоритму Хаффмена лежить ідея кодування бітовими групами. Спочатку проводиться частотний аналіз вхідної послідовності даних. Після цього символи сортуються по спаданню частоти входження. Основна ідея полягає в наступному: чим частіше зустрічається символ, тим меншою кількістю біт він кодується. Результат кодування зводиться в словник, що необхідний для декодування.

Основні принципи оптимального кодування

- *Оптимальність* коду – властивість такого коду, який забезпечує найменшу ймовірність виявлення помилки серед усіх кодів тієї ж довжини *n* і надмірності *r*.
- Кодування, при якому досягається мінімальна середня довжина кодового слова, називається *оптимальним*.
- Основні властивості оптимальних кодів:

  1. Мінімальна середня довжина кодового слова оптимального коду забезпечується в тому випадку, коли надмірність кожного кодового слова зведена до мінімуму (в ідеальному випадку до нуля).

  2. Кодові слова оптимального коду повинні будуватися з рівноймовірних і незалежних символів.
- Принципи побудови оптимальних кодів:

  1. Вибір кожного кодового слова необхідно робити так, щоб кількість інформації, що міститься в ньому, була максимальною.

  2. Літерам первинного алфавіту, які мають велику ймовірність, присвоюються коротші кодові слова у вторинному алфавіті. (Символи, за

допомогою яких записано передане повідомлення, складають первинний алфавіт, а символи, за допомогою яких повідомлення трансформується в код, – вторинний алфавіт).

- Система передачі інформації, в якій апаратура не вносить жодних спотворень, а канал зв'язку – загасань і завад, називається *інформаційною системою без завад*.
- Вимоги до кодів:

  1) різні символи первинного алфавіту, з якого складені повідомлення, повинні мати різні кодові комбінації;

  2) код повинен бути побудований так, щоб можна було чітко відокремити початок і кінець букв первинного алфавіту;

  3) код повинен бути максимально коротким.
- Найбільш розповсюджені коди з мінімальною надлишковістю, які застосовуються при економному кодуванні: код Шеннона-Фано, код Хаффмена, код Лемпеля-Зіва.

Принцип побудови коду Шеннона–Фано

1. Всі повідомлення (літери) записуються в таблицю в порядку зменшення ймовірності їх появи.

2. Всю послідовність повідомлень (букв) ділять на 2 групи так, щоб суми ймовірностей в кожній групі були б приблизно однаковими. Верхній групі повідомлень (букв) присвоюється цифра «0» (як перший символ коду), а нижній цифра «1».

3. Кожна група повідомлень (букв) ділиться на дві підгрупи із збереженням тієї ж умови однакової суми ймовірностей. Верхнім підгрупами в обох групах знову присвоюється цифра «0» (як другий символ коду), а нижнім – цифра «1».

4. Такий розподіл продовжується до тих пір, поки в підгрупах не залишиться тільки по одному повідомленню (букві).

Принцип побудови коду Хаффмена

1. Повідомлення виписуються в порядку збільшення ймовірностей.

2. Два останніх повідомлення об'єднуються в одне допоміжне повідомлення, якому приписується сумарна ймовірність.

3. Ймовірності повідомлень знову розташовуються в порядку збільшення ймовірностей в додатковому стовпці, а дві останні об'єднуються.

4. Процес продовжується до тих пір, поки не отримаємо єдине повідомлення з ймовірністю, рівною одиниці.

Код Лемпеля-Зіва

Компресор постійно зберігає певну кількість останніх оброблених символів у деякому буфері (ковзаючому словнику – sliding dictionary). Назва «ковзаючий» зумовлена тим, що його довжина постійна: кожного разу, коли компресор кодує наступний ланцюжок, він дописує його в кінець словника та «відрізає» відповідну кількість символів на початку буфера. Під час обробки вхідного потоку символи, що надійшли,

потрапляють у кінець буфера, зсуваючи попередні символи та витісняючи найстаріші. Алгоритм виділяє (шляхом пошуку в словнику) найдовший початковий підрядок вхідного потоку, що співпадає з одним із підрядків у словнику, і подає на вихід пару (length, distance), де length – довжина знайденого у словнику підрядка, а distance – відстань від нього до вхідного підрядка (тобто фактично індекс підрядка в буфері, віднятий від його розміру). В разі, коли такого підрядка не знайдено, до вихідного потоку просто копіюється черговий символ вхідного потоку.

Стиснення даних у комп'ютерних інформаційних технологіях

- Основні послуги, що надаються архіваторами:
  1) створення нового архіву;
  2) додавання файлів в існуючий архів;
  3) розпаковування файлів з архіву;
  4) створення архівів, що саморозпаковуються (self-extractor archive);
  5) створення розподілених архівів фіксованих розмірів для носіїв малої ємності;
  6) захист архівів паролями від несанкціонованого доступу;
  7) перегляд вмісту файлів різних форматів без попереднього розархівування;
  8) пошук файлів і даних всередині архіву;
  9) перевірка на віруси в архіві до розпаковування;
  10) вибір та налаштування коефіцієнта стиснення.
- *Програма WinRar* спеціально призначена для архівації і має зручності, через які популярна серед користувачів. Вона потрібна на комп'ютері, щоб при одержанні архівних файлів, виготовлених цією програмою, можна було б дістати з них потрібну інформацію.
- *Формат ZIP* – формат стиснення і архівації даних.

Закріплення матеріалу

1. Які основні алгоритми (методи) стиснення даних вам відомі?
2. Надати узагальнену класифікацію алгоритмів (методів) стиснення даних.
3. Назвати основні властивості та принципи побудови оптимальних кодів.
4. Яким вимогам повинні відповідати коди для однозначного декодування прийнятих повідомлень?
5. Пояснити роботу алгоритмів Шеннона–Фано, Хаффмена, Лемпеля-Зіва.
6. Які архіватори вам відомі?
7. Які послуги для роботи з архівами може отримати користувач при роботі з сучасними архіваторами?

РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА. ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Лекція 3

АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ ПК. КЛАСИФІКАЦІЯ ТА ОСНОВНІ ХАРАКТЕРИСТИКИ ЕОМ

Мета лекції: розгляд класифікації електронних обчислювальних машин; вивчення їх основних технічних характеристик.

Зміст лекції

1. Класифікація ЕОМ.
2. Основні технічні характеристики ЕОМ.

Класифікація ЕОМ

За принципом дії (за способом надання інформації)

- ЦОМ працюють з інформацією, наданою в дискретній, а точніше, в цифровій формі.
- АОМ працюють з інформацією, наданою в безперервній формі, тобто у вигляді безперервного ряду значень будь-якої величини (частіше за всю електричної напруги).
- ГОМ працюють з інформацією, наданою і в цифровій, і в аналоговій формі.
- Електронні обчислювальні машини (ЕОМ) – ЦОМ з електричним наданням дискретної інформації.

За етапами створення і елементною базою, що використовується

- **1-е покоління**: ЕОМ на електронних вакуумних лампах;
- **2-е покоління**: ЕОМ на дискретних напівпровідникових приладах;
- **3-е покоління**: ЕОМ на напівпровідникових інтегральних схемах з малим і середнім ступенем інтеграції;
- **4-е покоління**: ЕОМ на великих і надвеликих інтегральних схемах – мікропроцесорах;
- **5-е покоління**: ЕОМ з багатьма десятками паралельно працюючих мікропроцесорів, що дозволяють будувати ефективні системи обробки знань; ЕОМ на надскладних мікропроцесорах з паралельно-векторною структурою, одночасно виконуючих десятки послідовних команд програми;

- **6-е покоління і наступні покоління**: оптоелектронні ЕОМ з масовим паралелізмом і нейронною структурою.

За призначенням

- *Універсальні ЕОМ* призначені для вирішення самих різних інженерно-технічних задач, що відрізняються складністю алгоритмів і великим об'ємом оброблюваних даних.
- *Проблемно-орієнтовані ЕОМ* служать для вирішення більш вузького кола задач, зв'язаних, як правило, з управлінням технологічними об'єктами; реєстрацією, накопиченням і обробкою відносно невеликих об'ємів даних.
- *Спеціалізовані ЕОМ* використовуються для вирішення вузького кола задач або реалізації чітко визначеної групи функцій.

За кількістю обчислювальних пристроїв і ступенем розподілення

- *Обчислювальна система* – складна сукупність апаратних засобів, які з'єднані внутрішніми шинами і реалізовують загальні програми обчислень.
- *Обчислювальний комплекс* – сукупність двох і більш ЕОМ одного або різних типів, призначених для вирішення загального класу задач і з'єднаних між собою за допомогою загальної (зовнішньої) пам'яті (з непрямим зв'язком) або через канали введення-виведення (з прямим зв'язком).
- *Обчислювальна мережа* – безліч ЕОМ, з'єднаних стандартними телекомунікаційними каналами зв'язку або стандартними каналами передачі даних.

За розмірами і функціональними можливостями

- За розмірами і функціональними можливостями ЕОМ можна розділити на *суперЕОМ*, *великі*, *малі* і *мікроЕОМ*.
- *Багатокористувальницькі мікроЕОМ* – це потужні мікроЕОМ, обладнані декількома відеотерміналами і функціонуючі в режимі розподілу часу, що дозволяє ефективно працювати на них відразу декільком користувачам.
- *Персональні комп'ютери* (ПК) – розраховані на одного користувача мікроЕОМ, що задовольняють вимогам загальнодоступності і універсальності застосування.
- *Робочі станції* – розраховані на одного користувача могутні мікроЕОМ, спеціалізовані для виконання певного виду робіт (графічних, видавничих і ін.).
- *Сервери* – багатокористувальницькі потужні мікроЕОМ в обчислювальних мережах, виділені для обробки запитів від всіх станцій мереж.

За кількістю процесорів

- За кількістю процесорів ЕОМ поділяються на однопроцесорні і багатопроцесорні.

За способом управління

- За способом управління ЕОМ поділяються на ЕОМ, що керуються потоком інструкцій (команд) та ЕОМ, що керуються потоком даних (потокова архітектура).

Основні технічні характеристики ЕОМ

Операційні ресурси

- *Операційні ресурси* – це перелік дій (операцій), які може робити (виконувати) апаратура в плані обробки інформації (початкових даних).

Об'єм пам'яті

- *Об'єм пам'яті* – очевидна технічна характеристика, яка характеризує об'єм збереження програм і даних.

Швидкодія

- *Швидкодія* – це характеристика, яка відповідає на питання, як швидко діє (працює) апаратура ЕОМ. Швидкодія визначається кількістю операцій в одиницю часу і залежить від часу виконання операції. Швидкодія – це паспортна характеристика, вказується в документі на пристрій або у вигляді вектора швидкостей *V*, або у вигляді набору часу: *t*+, *t*-, *t**, *t*/, …
- *Швидкодія процесора* визначається часом виконання команд.
- Одиниці вимірювання швидкодії:
  - МІПС (MIPS – Mega Instruction Per Second) — мільйон операцій над числами з фіксованою комою (крапкою);
  - МФЛОПС (MFLOPS – Mega Floating Operations Per Second) – мільйон операцій над числами з плаваючою комою (крапкою);
  - КОПС (KOPS – Kilo Operations Per Second) для низьковиробничих ЕОМ – тисяча будь-яких усереднених операцій над числами;
  - ГФЛОПС (GFLOPS – Giga FLoating Operations Per Second) – мільярд операцій в секунду над числами з плаваючою комою (крапкою).
- *Швидкодію пам'яті* прийнято характеризувати кількістю операцій зчитування/запису в одиницю часу.
- Продуктивність ЕОМ оцінюється кількістю задач в одиницю часу.
- Тактова частота – кількість синхронізуючих тактів, що надходять ззовні на вхід схеми за одну секунду.
- ***Способи підвищення продуктивності***

  1) *Вдосконалення технології виробництва ЕОМ* («фізичний» шлях) – підвищення швидкодії логічних елементів.

  2) *Розпаралелювання обчислень* – знаходження алгоритму рішення задачі, що використовує паралелізм, і реалізація цього алгоритму в ОС. *Паралельні ОС* – це багатопроцесорні ОС, в яких паралелізм використовується для підвищення продуктивності при вирішенні завдань за рахунок одночасного виконання різних операцій на різних процесорах або обробляючих пристроях одного процесора.

3) *Конвейеризація обчислень* – розбиття обчислень на послідовні етапи з метою реалізації цих етапів на окремих сходинах конвеєра для підвищення продуктивності. *Конвеєр* – пристрій, що складається з N послідовно з'єднаних частин (сходин конвеєра), кожна з яких виконує черговий крок обчислень за час t (такт конвеєра).

4) *Спеціалізація обчислень* дозволяє підвищити продуктивність за рахунок апаратної реалізації рішення задачі в цілому, або будь-якої складної частини цього завдання, тобто методом програмування апаратної структури.

5) *Апаратна реалізація складних функцій* (арифметичних, матричних операцій тощо).

Надійність ОК

- *Надійність ОК* – це властивість ОК виконувати покладені на нього функції протягом заданого відрізку часу.
- *Відмови апаратури* – випадкові події, частоту яких прийнято характеризувати інтенсивністю відмов λ, тобто кількістю відмов в одиницю часу.
- *Напрацювання на відмову*: $T = 1/\lambda$ – це проміжок часу між двома сусідніми (за часом) відмовами.

Вартість ОК

- *Вартість ОК* – інтегральна характеристика, визначається всіма перерахованими характеристиками.

Закріплення матеріалу

1. На які класи поділяються ЕОМ за принципом дії?
2. Як поділяються ЕОМ за етапами створення і елементною базою, що використовується?
3. На які групи поділяються ЕОМ за призначенням?
4. Як поділяються ЕОМ за кількістю обчислювальних пристроїв і ступенем розподілення?
5. Як можна розділити ЕОМ за розмірами і функціональними можливостями?
6. Як поділяються ЕОМ за кількістю процесорів?
7. Як поділяються ЕОМ за способом управління?
8. Які основні технічні характеристики ЕОМ вам відомі?
9. Що таке операційні ресурси?
10. В чому різниця між швидкодією процесора та швидкодією пам’яті?
11. Від яких чинників залежить швидкодія ЕОМ в цілому?
12. Які основні шляхи підвищення продуктивності вам відомі?
13. Що таке розпаралелювання?
14. Назвати три проблеми розпаралелювання.
15. Що таке конвейеризація обчислень?
16. Що таке надійність обчислювального комплексу?

РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА. ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Лекція 4

АПАРАТНЕ ЗАБЕЗПЕЧЕННЯ ПК. ОСНОВНІ СКЛАДОВІ ПК. ОСНОВНІ ЗОВНІШНІ ПРИСТРОЇ

Мета лекції: розгляд основних складових персонального комп'ютера, статичної та динамічної пам'яті; характеристика основних зовнішніх пристроїв.

Зміст лекції

1. Основні складові персонального комп'ютера, принципи їх побудови та роботи, статична та динамічна пам'ять ПК
2. Основні зовнішні пристрої.

Основні складові персонального комп'ютера, принципи їх побудови та роботи

- *Персональний комп'ютер* – універсальна технічна система.
- Переваги ПК:
  - мала вартість; автономність експлуатації без спеціальних вимог до умов навколишнього середовища;
  - гнучкість архітектури;
  - «дружність» операційної системи та іншого програмного забезпечення;
  - висока надійність роботи.

Системний блок

- *Системний блок* – основний вузол, всередині якого встановлені найбільш важливі компоненти.
- Параметри системного блока: форма корпуса, форм-фактор, потужність блоку живлення.

Мікропроцесор

- *Мікропроцесор* – це велика інтегральна схема, що представляє собою кремнієвий кристал у пластмасовому, керамічному або металокерамічному корпусі, на якому розташовані виводи для прийому і видачі електричних сигналів. Ступінь інтеграції інтегральної схеми визначається розміром кристала і кількістю розміщених у ньому транзисторів.
- Основні функції мікропроцесора:
  - виконання обчислень;
  - пересилання даних між внутрішніми регістрами;

  - керування ходом обчислювального процесу.
- Склад мікропроцесора:
  - АЛП (виконує арифметичні і логічні операції над даними);
  - пристрій управління (виробляє керуючі сигнали для виконання команд);
  - внутрішні регістри.
- Етапи реалізації кожної машинної команди:
  - вибірка команди з пам'яті;
  - декодування;
  - виконання;
  - запис результату.
- Основні параметри процесорів:
  - робоча напруга;
  - розрядність;
  - робоча тактова частота;
  - коефіцієнт внутрішнього множення тактової частоти;
  - розмір кеш-пам'яті.

Основна пам'ять. ОЗП і ПЗП

- *Основна пам'ять* містить оперативний (RAM – Random Access Memory – пам'ять з довільним доступом) та постійний (ROM – Read-Only Memory) запам'ятовуючі пристрої.
- *Оперативний запам'ятовуючий пристрій* призначений для зберігання інформації (програм і даних), безпосередньо бере участь у обчислювальному процесі на поточному етапі функціонування ПК.
- *Оперативна пам'ять* – це масив кристалічних комірок, які здатні зберігати дані. За фізичним принципом дії розрізняють динамічну пам'ять (DRAM) і статичну пам'ять (SRAM).
- *DRAM* – найбільш поширений і економічно доступний тип пам'яті. Недоліки: запис даних відбувається порівняно повільно, швидка втрата даних.
- *SRAM* – забезпечує істотно більш високу швидкодію, хоча технологічно складніший і, відповідно, дорожчий.
- Основні характеристики модулів оперативної пам'яті: об’єм пам'яті, час доступу.
- Комплект програм, що знаходяться в ПЗП, утворює *базову систему введення-виводу* (BIOS – Basic Input Output System), основне призначення якої: перевірити склад та працездатність комп'ютерної системи; забезпечити взаємодію її компонентів.

Мікропроцесорний комплект (чіпсет)

- *Чіпсет* – набір мікросхем, що зв’язують пам’ять, процесор, відеоадаптер, пристрій вводу-виводу та інші елементи.

Системна шина

- *Системна шина* – це механізм, що дозволяє організовувати взаємодію різних підсистем.

Зовнішня пам'ять

- Зовнішні запам'ятовуючі пристрої розрізняють за: видом носія, типом конструкції, принципом запису та зчитування інформації, методом доступу тощо.
- *Носій* – матеріальний об'єкт, здатний зберігати інформацію.
- *Жорсткий диск*, або жорсткиий магнітний диск, або накопичувач на магнітних дисках (англ. hard (magnetic) disk drive, англ. HDD), у комп'ютерному сленгу – «вінчестер» (від маркування набоїв гвинтівки «Вінчестер») – магнітний диск, основа якого виконана з твердого матеріалу. У більшості ЕОМ виконує функцію енергонезалежного носія інформації (комп'ютерної пам'яті чи нагромаджувача інформації) з довільним доступом (англ. random access).

Основні зовнішні пристрої

Клавіатура

- *Клавіатура* – клавішний пристрій керування персональним комп'ютером. Служить для введення алфавітно-цифрових (знакових) даних, а також команд управління.

Миша

- *Миша* – маніпулятор, що дозволяє оптимізувати роботу з великою категорією комп'ютерних програм.
- *За способом переміщення* миші поділяються на: механічні, оптичні, лазерні, трекбол, індукційні, сенсорні.
- *За способом передачі даних в комп'ютер* миші поділяються на: дротові і бездротові.

Відеотермінальні пристрої

- *Відеотермінал* складається з *відеомонітора (дисплея)* і *відеоконтролера (адаптера)*. Видеоконтролер входять до складу системного блоку ПК (знаходяться на відеокарті, що встановлюється в роз'єм материнської плати), а відеомонітори – це зовнішні пристрої ПК.
- *Відеомонітор*, дисплей або просто монітор – пристрій відображення текстової та графічної інформації на екрані. Класифікація за типом екрана: ЕПТ, рідкокристалічні, плазмові, проектори, LED-монітори, лазерні, віртуальні ретинальні.
- *Видеоконтролери (відеоадаптери)* є внутрішньосистемними пристроями, безпосередньо керуючими моніторами і виведенням інформації на їх екран.

Принтери

- *Принтери (друкувальні пристрої)* – це пристрої виведення даних з ЕОМ, що перетворюють інформаційні ASCII-коди у відповідні їм графічні символи (літери, цифри, знаки і т.п.) і фіксують ці символи на папері.
- За принципом дії розрізняють матричні, лазерні, світлодіодні та струменеві принтери.
- Друк у принтерів може бути посимвольний, порядковий, посторінковий.

Сканери

- *Сканер* – пристрій оптичного введення інформації, її сканування, фотографування, що служить для копіювання зображень навколишньої дійсності в комп'ютер.
- Розрізняють: планшетні, рулонні, проєкційні сканери.

Закріплення матеріалу

1. Надати визначення персонального комп’ютера.
2. Яку конфігурацію персонального комп’ютера називають базовою?
3. Які переваги персонального комп’ютера вам відомі?
4. Навести узагальнену структурну схему персонального комп’ютера та охарактеризувати її складові.
5. Які параметри визначають для системного блоку персонального комп’ютера?
6. Які основні складові системного блоку персонального комп’ютера вам відомі?
7. Що таке мікропроцесор?
8. Які основні функції мікропроцесора вам відомі?
9. Які основні параметри процесорів вам відомі?
10. Назвати складові мікропроцесора та пояснити їх призначення.
11. Для чого призначається оперативна пам’ять?
12. Для чого призначається постійна пам’ять?
13. Які дані зберігаються в пам’яті автономного живлення?
14. Пояснити різницю між ОЗП та ПЗП.
15. Для чого призначається відеопам’ять?
16. Для чого призначається кеш-пам’ять?
17. Описати організацію статичної і динамічної пам’яті персонального комп’ютера.
18. Які основні зовнішні пристрої вам відомі?
19. Які різновиди клавіатури вам відомі?
20. Які типи мишей вам відомі?
21. Які відео термінальні пристрої вам відомі?
22. Які різновиди принтерів вам відомі?
23. Які різновиди сканерів вам відомі?

РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА. ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Лекція 5

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПК. УЗАГАЛЬНЕНА КЛАСИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Мета лекції: розгляд архітектури та класифікації операційних систем.

Зміст лекції

1. Поняття та еволюція операційних систем.
2. Архітектура операційних систем: класифікація програмного забезпечення.

- *Інтерфейс* комп'ютерної програми – це її зовнішній вигляд на екрані дисплея, що включає її оформлення, вигляд і розташування елементів управління роботою цієї програми. Розрізняють текстовий та графічний інтерфейси.

Еволюція операційних систем

- *Завдання* (job) – запуск кожної програми.
- *Черга* – це спосіб організації пам'яті, коли об'єкти, що в ній зберігаються (у нашому випадку – завдання), впорядковані за принципом «*Першим увійшов – першим вийшов*» (FIFO, First-in, First-out), тобто об'єкти покидають чергу в тому ж порядку, в якому вони до неї надходять.
- *Пакетна обробка* – режим виконання сукупності завдань, при якому всі завдання виконуються автоматично, без синхронізації з подіями за межами даної системи обробки інформації, зокрема без зв'язку з особами, які представили завдання для виконання. ОС з пакетною обробкою передбачають, що з програм, які підлягають виконанню формується пакет, який завантажується в керуючий комплекс і далі обробляється ОС і МПр. У цьому випадку користувачі безпосередньо з ОС в процесі виконання пакета не взаємодіють.
- *Інтерактивний режим* – режим взаємодії в процесі обробки інформації з людиною, що виражається у різного роду впливах на цей процес, що передбачені механізмом управління конкретної системи і які викликають відповідну реакцію процесу.
- *Поділ часу* – процес обробки інформації, при якому ресурси системи обробки інформації надаються кожному процесу з групи процесів обробки інформації на інтервали часу тривалість і черговість надання яких визначаються управляючою програмою системи обробки інформації з

метою забезпечення одночасної роботи процесів даної групи в інтерактивному режимі.

- Для багатокористувальницьких і багатозадачних ОС розрізняють дві основні дисципліни обслуговування процесів з точки зору виділення процесорного часу на ту чи іншу задачу. Це *вишитовхуючий* та *узгоджувальний режим багатозадачної роботи*.
- При вишитовхуючому режимі розподілом процесорного часу займається тільки ОС, внаслідок чого для виконання кожного завдання МПр займається на строго певний інтервал часу з урахуванням пріоритету завдання. Перевагою вишитовхуючого режиму є відсутність монопольного доступу, до МПр для будь-якої програми або завдання. Недоліком вишитовхуючого режиму є негнучке і малоефективне використання ресурсів процесора, наприклад в тому випадку, якщо завдання вимагає більше часу на виконання, ніж відведено розкладом.
- При узгоджувальному режимі кожна задача, отримавши управління ресурсами і доступ до МПр, сама визначаємо коли МПр слід «віддати» іншій задачі. Перевагою узгоджувального режиму є можливість ефективного використання ресурсів процесора в разі перевантажень і критичних пошкоджень, коли процесор повинен обслуговувати тільки програми самовідновлення, тестування, експлуатації або випробувально-налагоджувальних програм. Недоліком узгоджувального режиму є можливість монопольного доступу до процесора з боку однієї програми, що тягне за собою зупинку виконання інших програм і підвищує ймовірність перезавантаження/перезапуску в разі, якщо стався збій.
- *Мережева операційна система* – це операційна система, що забезпечує обробку, зберігання і передачу даних в інформаційно-обчислювальній мережі. Мережева ОС визначає взаємопов'язану групу протоколів верхніх рівнів моделі взаємозв'язку відкритих систем, що забезпечують основні функції інформаційно-обчислювальної мережі: адресацію об'єктів, функціонування служб, забезпечення безпеки даних, управління мережею.

Архітектура операційних систем: класифікація програмного забезпечення

- Програмне забезпечення поділяється на дві загальні категорії: прикладне програмне забезпечення (application software) і системне програмне забезпечення (system software).
- *Прикладне програмне забезпечення* включає програми, призначені для вирішення завдань, що випливають зі специфічних особливостей використання даної машини.
- *Системне програмне забезпечення* виконує завдання, загальні для всіх обчислювальних систем. В цілому фактично системне програмне забезпечення формує середовище, в якому функціонує прикладне програмне забезпечення.

- *Операційною системою (ОС)* називається *комплект програм*, які спільно управляють ресурсами системи й процесами, що використовують ці ресурси. Виконання будь-якої програми на комп'ютері відбувається під управлінням ОС.
- *Ядро ОС* виконує основні функції ОС (в основному завантаження її компонентів і підтримку виконання комп'ютерних програм, у тому числі й цих компонентів).
- *Програма управління файлами та директоріями* служить для класифікації й перегляду інформації, з якою має справу користувач на комп'ютері.
- *Драйвери* дозволяють ОС працювати з апаратурою: периферійними пристроями (монітор, клавіатура, миша, принтери тощо) і пристроями, що входять до складу системного блоку (відеокарта, жорсткий диск тощо).
- *Утиліти* – це обслуговуючі програми, призначені для виконання дій, необхідних для успішного функціонування комп'ютера, але ще не включені в операційну систему. Утиліти займають проміжне положення між прикладними програмами та ОС.
- *Архіватор* – програма, що використовується для скорочення обсягу збереженої або переданої інформації.
- *Архів* – файл зі стислою інформацією.
- *Антивірусна програма* або *антивірус* – програма для боротьби з комп'ютерними вірусами.
- *Комп'ютерний вірус* або *вірус* – комп'ютерна програма, що не має свого виконуваного файлу, а впроваджується, самодописується у файли інших програм.
- *Прикладна програма*, або додаток, дозволяє користувачеві робити те, заради чого він використовує комп'ютер, тобто застосовувати комп'ютер у різних областях людської діяльності. Прикладна програма виконується на комп'ютері під керуванням ОС.
- *Програми-автомати* – це прикладні програми, де користувач експлуатує алгоритми й дані, а також способи класифікації даних і їхнього перегляду, створені іншими людьми.
- *Навчальні програми* допомагають користувачеві навчитися в якій-небудь області знань (мови, набір на клавіатурі, математика тощо).
- *Ігри* використовуються для відпочинку за комп'ютером, спортивних змагань, тренування логічного мислення, тренажерного тренування певних навичок і вмінь, а також навчання.
- *Бази знань* – найрізноманітніша інформація, організована в логічні структури.
- *Програми-інструменти* – це прикладні програми, за допомогою яких користувач створює нову авторську інформацію, що зберігається у відповідних файлах.

- *Редактори* – програми для створення, редагування, перегляду та зміни нової інформації, за винятком комп'ютерних програм.
- *Системи програмування*, або мови програмування – програми для створення комп'ютерних програм.
- *Текстові редактори* (текстові процесори) служать для створення різноманітних текстів природними і комп'ютерними мовами.
- *Графічні редактори* обробляють графічну інформацію та дозволяють додавати в неї графічні ефекти.
- *Мультимедійні редактори* мають справу з повною колекцією мультимедіа, у тому числі звуком і відео. *Звукові редактори* дозволяють візуально переглядати оцифрований звук, редагувати й прослуховувати його. *Відеоредактори* працюють з оцифрованим відео: здійснюють покадровий перегляд, редагування й додавання відео-ефектів, монтаж і озвучування відеоінформації.
- *Редактори баз даних*, або системи управління базами даних (СУБД), займаються *базами даних (БД)*, тобто найрізноманітнішою інформацією, організованою в логічні структури.
- *Системи програмування* або *мови програмування* – це прикладні програми, які дозволяють програмістові створювати будь-які комп'ютерні програми.

Закріплення матеріалу

1. Що таке пакетна обробка?
2. Що таке розподіл часу?
3. Що таке інтерактивний режим?
4. Які основні режими обслуговування процесів з точки зору виділення процесорного часу на ту чи іншу задачу вам відомі?
5. Пояснити різницю між виштовхуючим та узгоджувальним режимами багатозадачної роботи.
6. Що таке прикладне програмне забезпечення?
7. Що таке системне програмне забезпечення?
8. Навести класифікацію програмного забезпечення.
9. Що таке операційна система?
10. Що таке ядро операційної системи?
11. Для чого призначена програма управління файлами та директоріями?
12. Що таке утиліти?
13. В чому різниця між програмами-автоматами та програмами-інструментами?
14. На які складові поділяються програми-автомати?
15. На які складові поділяються програми-інструменти?

РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА. ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Лекція 6

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПК. АРХІТЕКТУРА ОПЕРАЦІЙНИХ СИСТЕМ. ФАЙЛОВА СИСТЕМА

Мета лекції: розгляд компонентів операційних систем; розгляд файлової системи.

Зміст лекції

1. Архітектура операційних систем: види і компоненти операційної системи, процес запуску операційної системи.
2. Структура та створення дерева: поняття файлу та директорії.

- *Операційна система* – це комплекс управляючих та обробляючих програм, який, з одного боку, виступає як інтерфейс між користувачем та апаратними компонентами обчислювальних машин та обчислюваних систем, а з іншого боку призначений для ефективного управління обчислюваними процесами, а також найбільш раціонального розподілу та використання обчислювальних ресурсів.
- Основні вимоги до операційної системи:
    1. Здатність виконання основних функцій – ефективного управління ресурсами та забезпечення зручного інтерфейсу для користувача та прикладних програм.
    2. Розширюваність.
    3. Перенесення.
    4. Надійність та відмовостійкість.
    5. Сумісність.
    6. Безпечність.
    7. Продуктивність.
- *Монолітна* операційна система працює як єдина програма в режимі ядра; написана у вигляді набору процедур, пов'язаних разом в одну велику виконувану програму.
- *Багаторівнева* операційна система організована у вигляді ієрархії рівнів, кожен з яких є надбудовою над нижнім рівнем.
- *Мікроядра* – операційні системи, які розбиваються на невеликі, конкретні модулі, один з яких – мікроядро – запускається в режимі ядра, а всі інші

запускаються у вигляді звичайних процесів користувача, відносно слабко наділених повноваженнями.

- Суть *клієнт-серверної моделі* полягає в наявності клієнтських і серверних процесів, зв'язок між якими часто організовується за допомогою передачі повідомлень.
- *Віртуальна машина* – модель обчіслювальної машини, створеної шляхом віртуалізації обчислювальних ресурсів, яка забезпечує повну емуляцію фізичної машини чи середовища виконання.
- *Екзоядра* – ОС, в основу якої покладена стратегія надання кожному користувачеві підмножини ресурсів замість клонування справжньої машини, як це робиться в віртуальних машинах.
- Операційні системи *мейнфреймів* орієнтовані переважно на одночасну обробку безлічі завдань, більшість з яких вимагає величезних обсягів вводу-виводу даних. Зазвичай вони пропонують три види обслуговування: пакетну обробку, обробку транзакцій і роботу в режимі поділу часу.
- *Серверні операційні системи* працюють на серверах, які представлені дуже потужними персональними комп'ютерами, робочими станціями або універсальними машинами; одночасно обслуговують по мережі безліч користувачів, забезпечуючи їм загальний доступ до апаратних і програмних ресурсів.
- *Багатопроцесорні операційні системи* підтримують режим розподілених ресурсів декількох процесорів для розв’язання одної задачі.
- *Операційні системи персональних комп'ютерів* підтримують багатозадачність; їх завданням є якісна підтримка роботи окремого користувача.
- *Операційні системи кишенькових персональних комп'ютерів* – це операційні системи планшетів, смартфонів та інших КПК.
- *Вбудовані операційні системи* працюють на комп'ютерах, які керують різними пристроями, та відрізняються тим, що на них ні за яких умов не буде працювати стороннє програмне забезпечення.
- *Операційні системи сенсорних вузлів* – це невеликі за обсягом і нескладні операційні системи, які зазвичай керуються подіями і відгукуються на зовнішні події або періодично виконують вимірювання за сигналами вбудованого годинника. Всі програми таких систем є попередньо завантаженими, і користувачі не можуть запустити програму, завантажену з Інтернету.
- *Операційні системи реального часу* – це операційні системи, які характеризуються тим, що час для них є ключовим параметром. Розрізняють системи жорсткого та м'якого реального часу.
- *Операційні системи смарт-карт* – це найменші операційні системи, на які накладаються дуже жорсткі обмеження щодо необхідної обчислювальної потужності процесора і обсягу пам'яті.

- *Системний виклик* – це інтерфейс між операційною системою та програмою користувача.
- Системні виклики створюють, видаляють та використовують різні об'єкти, головні з яких процеси та файли. Програма користувача робить запит до операційної системи на сервіс, здійснюючи системний виклик. Системні виклики ще називають *програмними перериваннями*.
- *Переривання* – це подія, яка генерується зовнішніми (по відношенню до процесора) пристроями. За допомогою апаратних переривань апаратура інформує центральний процесор про те, що виникла якась подія, яка потребує миттєвої реакції.
- *Виключна ситуація* – це подія, яка виникла в результаті спроби виконання програмою неприпустимої команди, доступу до ресурсу при відсутності достатніх привілеїв або звернення до відсутньої сторінки пам'яті.
- Виключні ситуації бувають виправним (після усунення їх причини програми продовжують працювати) и та невиправними (зазвичай виникають в результаті помилок в програмах).
- *Оболонка операційної системи* – режим виконання сукупності завдань, при якому всі завдання виконуються автоматично, без синхронізації з подіями за межами даної системи обробки інформації, зокрема без зв'язку з особами, які представили завдання для виконання.
- *Система керування вікнами* або *віконний менеджер* (Windows manager) – розподіляє окремі блоки простору екрана, які називають вікнами, і відслідковує, який додаток асоціюється з кожним із цих вікон.
- *Ядро* – виконує основні функції ОС в процесі приведення ПК в робочий стан.
- *Система керування файлами* або *файлова система* (file manager) координує використання запам'ятовуючих пристроїв.
- *Файл* – іменований набір даних; блок інформації на запам'ятовуючому пристрої ПК, який має певне логічне надання, відповідні йому операції читання-запису та, як правило, фіксоване ім'я, що дозволяє отримати доступ до цього файлу та відрізнити його від інших.
- *Файлова система* – це спосіб організації даних, який використовується операційною системою для збереження інформації у вигляді файлів на носіях інформації; сукупність файлів та директорій, які розміщуються на логічному або фізичному пристрої.
- *Драйвери пристроїв* (devices drivers) – елементи програмного забезпечення, що взаємодіють з контролерами пристроїв (або ж безпосередньо із пристроями) з метою виконання різних операцій у периферійних пристроях машини.
- *Система керування пам'яттю* (memory manager) – вирішує завдання координації використання машиною її основної пам'яті.

- В системах з поділом часу *планувальник* (scheduler) визначає послідовність виконуваних дій, а *диспетчер* (dispatcher) контролює розподіл тимчасових квантів для них.
- *Процес* – це програма в момент її виконання. З кожним процесом пов'язується його адресний простір – список адрес в пам'яті від деякого мінімуму (зазвичай нуля) до деякого максимуму, які процес може прочитати і в які він може писати.

Закріплення матеріалу

1. Що таке операційна система?
2. Які основні вимоги до операційних систем вам відомі?
3. В чому різниця між монолітною та багаторівневою операційними системами?
4. В чому суть клієнт-серверної моделі?
5. Що таке віртуальна машина?
6. Для чого призначені серверні операційні системи?
7. Для чого призначені багатопроцесорні операційні системи?
8. Для чого призначені операційні системи кишенькових персональних комп'ютерів?
9. Для чого призначені вбудовані операційні системи?
10. Для чого призначені операційні системи сенсорних вузлів?
11. Для чого призначені операційні системи реального часу?
12. Для чого призначені операційні системи смарт-карт?
13. Що таке віконний менеджер?
14. Що таке системний виклик?
15. Що таке переривання?
16. Що таке виключна ситуація?
17. Що таке оболонка операційної системи?
18. Пояснити принцип роботи віконного менеджера.
19. Пояснити принцип роботи ядра операційної системи.
20. Описати процес запуску операційної системи.
21. Як формується дерево директорій?
22. Які задачі вирішує система керування пам’яттю?
23. Які функції виконує планувальник в системах з поділом часу?
24. Які функції виконує диспетчер в системах з поділом часу?
25. Що таке процес?
26. Що таке потік?
27. В чому різниця між процесом та потоком?

РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА. ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Лекція 7

ТЕКСТОВІ ПРОЦЕСОРИ ТА ЕЛЕКТРОННІ ТАБЛИЦІ. ВИКОРИСТАННЯ MS WORD ПРИ ФОРМУВАННІ ЗВІТІВ В СФЕРІ НАУКИ І ТЕХНІКИ

Мета лекції: розгляд компонентів операційних систем. Розгляд файлової системи.

Зміст лекції

1. Вимоги до порядку викладення матеріалів звіту.
2. Правила оформлення звіту.

Вимоги до порядку викладення матеріалів

Структура звіту (документу)

- Звіт умовно поділяється на такі частини, як вступна, основна та додатки.
- Вступна частина складається з наступних структурних елементів: титульний аркуш; зміст; перелік умовних позначень.
- Основна частина містить наступні структурні елементи: вступ; суть звіту; висновки; перелік посилань.
- Додатки, якщо вони передбачені, розміщують після основної частини звіту.
- Структурні елементи «Титульний аркуш», «Вступ», «Суть звіту», «Висновки» є обов'язковими.

Вимоги до структурних елементів вступної частини

- Титульний аркуш є першою сторінкою звіту. Титульний аркуш містить дані, які подають у такій послідовності: а) назва міністерства та установи; б) назва дисципліни; в) назва звіту (документу); г) ідентифікація групи та ПІБ студента; д) рік написання;
- Зміст розташовують безпосередньо після титульного аркушу, починаючи з нової сторінки. Зміст складається із переліку умовних позначень, вступу, назв всіх розділів, підрозділів, пунктів та підпунктів, суті звіту, висновків, переліку посилань, назв додатків. Всі складові звіту супроводжуються номерами сторінок, які відповідають початку матеріалу. У змісті також можуть бути перелічені номери та назви ілюстрацій і таблиць з зазначенням сторінок, на яких вони розміщені.
- В переліку умовних позначень, символів, одиниць, скорочень, термінів. пояснюються всі використані у звіті малопоширені умовні позначення,

скорочення, одиниці, символи і терміни. Перелік розміщують, починаючи з нової сторінки, безпосередньо після змісту. При цому, перша поява у тексті зазначених елементів супроводжується розшифровкою.

Вимоги до структурних елементів основної частини

- Вступ складається з короткого викладу оцінки сучасного стану проблеми. При цьому зазначають світові тенденції розв'язання поставлених задач, прогалини знань даної галузі, задачі, які практично розв'язані тощо. Вступ розташовують на окремій сторінці.
- В звіті викладають відомості про предмет (об'єкт) дослідження або розробку. Ці відомості повинні бути необхідними й достатніми для того, щоб розкрити суть представленої роботи та її результатів. З метою розкриття суті звіту, матеріал поділяють на розділи, які в свою чергу, можуть бути поділені на підрозділи, пункти та підпункти. Кожен пункт і підпункт формується як змістовно закінчена частина.
- Висновки розміщують безпосередньо після викладеної суті звіту, починаючи з нової сторінки. У висновках для деяких тем необхідно навести таблиці з порівняльними характеристиками різновидів предмету дослідження та викласти власну думку щодо переваг і недоліків їх використання. Текст висновків може бути поділений на пункти.
- Якщо звіт в основній частині має посилання на літературні джерела, то в кінці звіту наводиться перелік цих джерел, починаючи з нової сторінки. Бібліографічні описи в переліку посилань потрібно надавати в тому порядку, в якому вони вперше згадуються в тексті основної частини звіту. Номерні посилання – це посилання в тексті, які відповідають порядковим номерам описів у переліку. Бібліографічні описи посилань у переліку формуються відповідно до чинних стандартів з бібліотечної та видавничої справи. Якщо матеріали звіту посилаються на джерела, які є тільки в додатках, то дозволяється створювати окремий перелік посилань. Такий перелік посилань розміщується в кінці цього додатку.

Вимоги до додатків

- Наповненням додатків можуть бути матеріали, які:
  - через способи відтворення та/або великий обсяг не можуть послідовно викладатись в основній частині звіту;
  - вважаються необхідним для повноти звіту, але їх розміщення в основній частині звіту може змінити порядок і логіку надання роботи;
  - є важливим для фахівців даної галузі, але не є необхідним для широкого кола читачів.
- Додатки можуть містити матеріали, які з певних причин (великий обсяг, форма подання, специфіка викладення) не були внесені до основної частини звіту, додаткові таблиці та ілюстрації, додатковий перелік літературних джерел. Під додатковим переліком джерел мають на увазі літературні джерела, на які відсутні посилання у звіті, але які можуть зацікавити фахівців даної галузі.

Правила оформлення

Загальні вимоги

- Звіт, в залежності від особливостей і змісту, складається у вигляді тексту, таблиць, ілюстрацій або ж їх сполучень. Звіт оформлюється на аркушах формату А4 (210 × 297 мм). Також допускається застосування аркушів формату А3 (297 × 420 мм). Засобами MS Word 2010 формат аркушу встановлюється наступним чином: п.м.(пункт меню) Розмітка сторінки → вкл. (вкладинка) Розмір → п. (пункт) Інші розміри сторінки → д.о. (діалогове вікно) Параметри сторінки → вкл. Розмір паперу. Якщо потрібно змінити орієнтацію аркушу з книжкової на альбомну, використовують замість вкл. Розмір паперу вкл. Поля або п.м. Розмітка сторінки → вкл. Орієнтація. Звіт виконують на одному боці аркуша білого паперу. Звіт друкують через півтора інтервали шрифтом Times New Roman 14пт за умови рівномірного заповнення сторінки. Засобами MS Word міжрядковий інтервал встановлюється наступним чином: п.м. Головна → вкл. Абзац → д.о. Абзац → вкл. Відступи і інтервали. Текст звіту друкується з дотриманням наступних розмірів границь аркушу: верхній, лівий та нижній – не менше 20 мм, правий – не менше 10 мм. Засобами MS Word 2010 розміри берегів аркуша встановлюються наступним чином: п.м. Розмітка сторінки → вкл. Розмір → п. Інші розміри сторінки → д.о. Параметри сторінки → вкл. Поля.
- Всі зображення у звіті повинні бути рівномірно щільними, контрастними та чіткими. Лінії, літери, цифри та інші знаки впродовж всього звіту повинні бути чіткимиі, нерозпливчастимиі та однаково чорними.
- Якщо окремі слова, формули або знаки вписуються у надрукований текст, то їх щільності повинна максимально наближуватись до щільності надрукованого тексту, а колір повинен бути чорним.
- Якщо текст звіту має помилки, описки та графічні неточності, то їх допускається виправляти чорним кольором шляхом підчищення або зафарбовування білою фарбою і нанесення на тому самому місці або між рядками виправленого тексту в друкованому чи рукописному вигляді.
- Власні назви у звіті (наприклад, прізвища, назви установ тощо) наводяться мовою оригіналу. Також допускається транслітерація власних назв і наведення назв установ у перекладі на мову звіту з додаванням (при першій згадці) оригінальної назви.
- Скорочення слів та словосполучень у звіті виконується у відповідності до чинних стандартів з бібліотечної і видавничої справи.
- Структурні елементи у звіті не нумеруються. Назви структурних елементів приймаються за заголовки. До структурних елементів належать: «СПИСОК АВТОРІВ», «РЕФЕРАТ», «ЗМІСТ», «ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ», «ПЕРЕДМОВА», «ВСТУП», «ВИСНОВКИ», «РЕКОМЕНДАЦІЇ», «ПЕРЕЛІК ЛІТЕРАТУРНИХ ПОСИЛАНЬ».

- Розділи і підрозділи звіту обов'язково повинні мати заголовки, а пункти і підпункти заголовків можуть не мати. Заголовки структурних елементів і розділів звіту розміщуються посередині рядка та друкуються великими літерами без крапки в кінці та підкреслень. Панель інструментів MS Word 2010 (п.м. Головна) має засоби для різноманітного розташування тексту на сторінці: з лівого краю, посередині, з правого краю, по ширині.
- Перетворити маленькі літери у великі можна за допомогою відповідної кнопки з Панелі інструментів (п.м. Головна). Якщо такої кнопки (або будь-якої іншої потрібної) немає на Панелі інструментів одразу після встановлення MS Word, її можна додати до елементів Панелі інструментів наступним чином: пр.кн.м. (права кнопка миші) на пустому місці Панелі інструментів → п. Налаштування стрічки… → д.о. Параметри Word → п. Налаштування стрічки → список Вибрати команди. В списку Налаштування стрічки перелічені всі пункти лінійки меню та є можливість створити новий пункт меню, який налаштовується користувачем (кн. Створити групу). Додати відсутню команду зі списку Вибрати команди у відповідний пункт меню можна виділивши її та натиснувши кнопку Додати.
- Заголовки підрозділів, пунктів і підпунктів звіту починаються з абзацного відступу і друкуються маленькими літерами (крім першої великої) без крапки в кінці та підкреслень.
- Засобами MS Word 2010 абзацний відступ встановлюється наступним чином: п.м. Головна → вкл. Абзац → д.о. Абзац → вкл. Відступи і інтервали. Ще можна використати лінійку розмірів та повзунець.
- Абзацний відступ дорівнює 1,25 см та є однаковим впродовж всього тексту звіту.
- В заголовоках, які складаються з двох і більше речень, речення розділяють крапками.
- Перенесення слів у заголовках розділів не допускається. Прибрати автоматичне перенесення: п.м. Розмітка сторінки → вкл. Розстановка переносів → прапорець Ні.
- Відстань між заголовком і подальшим чи попереднім текстом має бути 12 пт, а між основами рядків заголовку або двома заголовками – така, як в тексті. Назви розділів, підрозділів, пунктів та підпунктів заборонено розміщувати в нижній частині сторінки, якщо після цих назв розміщений лише один рядок тексту.

Нумерація сторінок звіту

- Сторінки звіту нумеруються арабськими цифрами за дотриманням наскрізної нумерації впродовж всього тексту звіту. Номер сторінки необхідно проставляти в правому верхньому куті сторінки без крапки в кінці.
- Засобами MS Word 2010 номер сторінки встановлюється наступним чином: п.м. Вставка → вкл. Номер сторінки → обрати місце розміщення

номеру → відкривається вкл. Робота з колонтитулами, де можна налаштувати формат номера (вкл. Колонтитули → вкл. Номер сторінки → п. Номер сторінки.

- Титульний аркуш входить до загальної нумерації сторінок звіту, але номер сторінки на ньому не ставиться (вкл. Робота з колонтитулами → вкл. Парамети → встановити прапорець для п. Особливий колонтитул для першої сторінки).
- До загальної нумерації сторінок звіту також входять розміщені на окремих сторінках ілюстрації та таблиці.

Нумерація розділів, підрозділів, пунктів, підпунктів

- Розділи, підрозділи, пункти та підпункти звіту нумеруються арабськими цифрами. Розділи звіту нумеруються арабськими цифрами без крапки (наприклад, 1, 2, 3), при цьому нумерація має бути порядковою в межах викладення суті звіту. Нумерація підрозділів має бути порядковою в межах кожного розділу. Номер підрозділу складається з номера розділу і відокремленого від нього крапкою порядкового номера підрозділу без крапки (наприклад, 1.1, 1.2).
- Нумерація пунктів має бути порядковою в межах кожного розділу або підрозділу. Варіантами формування номеру пункту: номер розділу і порядковий номер пункту; номер розділу, порядковий номер підрозділу та порядковий номер пункту. При цьому порядкові номери відповідних елементів розділяються крапкою, але після номера пункту крапка не ставиться (наприклад, 1.1, 1.1.2). За умови поділу тексту лише на пункти, вони позначаються порядковими номерами (за винятком додатків). Номер підпункту формується з відокремлених крапкою порядкових номерів розділу, підрозділу, пункту і підпункту (наприклад, 1.1.1.1, 1.1.1.2). Якщо в розділі відсутні підрозділи, але він поділяється на пункти та підпункти, то номер підпункту формується з номерів розділу, пункту і підпункту, відокремлених крапкою (наприклад, 1.1.3, 1.2.1). Після номера підпункту крапка не ставиться. Якщо розділ та/або підрозділ містить один пункт, або ж пункт містить один підпункт, то він підлягає нумерації.
- Засобами MS Word 2010 нумерацію розділів, підрозділів, пунктів та підпунктів можна здійснити автоматично: п.м. Головна → вкл. Маркований, вкл. Нумерований або вкл. Багаторівневий список.

Ілюстрації

- Ілюстрації в звіті (рисунки, схеми, графіки, креслення, діаграми або фотознімки) розміщуються безпосередньо після тексту, де вони згадуються вперше, або на наступній сторінці. В тексті звіту обов'язково повинні бути посилання на всі ілюстрації. Ілюстрації можуть супроводжуватись назвою, яка розміщується під ілюстрацією. Інколи під ілюстрацією розміщується підрисунковий текст, тобто пояснювальні дані. Ілюстрація позначається словом «Рисунок __», що разом з назвою ілюстрації розміщується після підрисункового тексту (наприклад, «Рисунок 7.1 – Топології мереж».

Ілюстрації нумеруються за допомогою порядкової нумерації в межах розділу та арабськими цифрами. Винятком є ілюстрацій, які наведені в додатках. Номер ілюстрації формується з порядкових номерів розділу і ілюстрації та відокремлюються крапкою (наприклад, Рисунок 7.5 – п'ятий рисунок сьомого розділу. Ілюстрації та їх назви центруються відносно лівого та правого поля сторінки без абзацного відступу. Якщо ілюстрація не вміщується на одній сторінці, то дозволяється її перенести на інші, при цьому назва ілюстрації розміщується на першій сторінці, а пояснювальні дані – на кожній сторінці, під якими зазначається: «Рисунок _, аркуш _». В деяких випадках ілюстрації вводяться до змісту із зазначенням їх номерів, назв та номерів сторінок, де вони розміщені.

Таблиці

- Зазвичай таблиці призначаються для оформлення цифрового матеріалу. Розділяючі рядки таблиці горизонтальні та вертикальні лінії, а також обмежувальні лінії зліва, справа і знизу можна не проводити, якщо їх відсутність не ускладнить розуміння таблиці. Таблиця розміщується безпосередньо після тексту, де вона згадана вперше, або ж на наступній сторінці. В тексті звіту на всі таблиці обов'язково повинні бути посилання. Таблиці нумеруються за допомогою порядкової нумерації в межах розділу та арабськими цифрами. Винятком є таблиці, які наведені в додатках. Номер таблиці формується з розділених крапкою порядкових номерів розділу і таблиці (наприклад, Таблиця 5.4 – четверта таблиця п'ятого розділу. Таблиця може супроводжуватись назвою, надрукованою малими літерами (крім першої великої). Назва розміщується над таблицею та повинна бути стислою. Назва має відбивати зміст таблиці. Якщо рядки або графи таблиці виходять за границі формату сторінки, то таблицю дозволяється поділити на частини. При цьому, частини можна розміщувати декількома способами: одну частину під одною, поруч, переносячи частину таблиці на наступну сторінку. У випадку переносу на іншу сторінку необхідно або повторювати в кожній частині таблиці її рядок-заголовок і спільну бокову частину, або ж замінити їх відповідно номерами граф чи рядків, при цьому нумерувати їх необхідно арабськими цифрами у першій частині таблиці. Слово «Таблиця _ » пишеться один раз зліва над першою частиною таблиці, а над іншими частинами зазначається: «Продовження таблиці _» із вказуванням номеру таблиці. Заголовки граф таблиці починають з великої літери. Якщо в таблиці є підзаголовки, які складають одне речення з заголовком, то їх починають з малої літери. Якщо в таблиці є підзаголовки, які мають самостійне значення, то вони пишуться з великої літери. Крапка в кінці заголовків і підзаголовків таблиць не ставиться. Заголовки і підзаголовки граф пишуться в однині. В деяких випадках таблиці вводяться до змісту із зазначенням їх номерів, назв та номерів сторінок, де вони розміщені.

- Засобами MS Word побудувати таблицю можна наступним чином: п.м. Вставка → вкл. Таблиці → Таблиця.

Переліки

- Інколи переліки наведяться всередині пунктів або підпунктів. Переліку передує двокрапка. Кожній позиції переліку передує мала літера української абетки з дужкою або дефіс, які відповідають першому рівню деталізації. Для другого та інших рівнів деталізації використовують арабські цифри з дужкою.
- Переліки першого рівня деталізації необхідно друкувати з абзацного відступу та малими літерами. Переліки другого рівня деталізації необхідно друкувати з відступом відносно місця розміщення переліків першого рівня деталізації.
- Засобами MS Word переліки можна робити автоматично за допомогою маркованих, нумерованих та багаторівневих списків: п.м. Головна → вкл. Маркированный, вкл. Нумерований або вкл. Багаторівневий список.

Примітки

- Якщо тексті, таблиці або ілюстрації звіту необхідно внести деякі пояснення, використовують примітки, які розміщуються безпосередньо після тих елементів звіту, яких вони стосуються. Одна примітка нумерації не підлягає. Слово «Примітка» пишеться з абзацного відступу та великої літери без підкреслень. Після нього ставиться крпка в тому самому рядку розміщується текст примітки.

Виноски

- Якщо окремі дані в тексті або таблиці звіту потребують пояснень, їх можна надати за допомогою виноски. Для позначення виносок використовують надрядкові знаки у вигляді порядкових номерів, надрукованих арабськими цифрами з дужкою. При цьому нумерація виносок робиться в рамках однієї сторінки, тобто не є наскрізною. Знаки виноски друкуються безпосередньо після того елементу, до якого належить пояснення, та перед текстом пояснення. Текст виноски починається з абзацного відступу, записується з мінімальним міжрядковим інтервалом, розміщується під таблицею або в кінці сторінки та відокремлюють від тексту (таблиці) лінією в лівій частині сторінки, довжина якої складає 30 – 40 мм.
- Засобами MS Word 2010 виноски встановлюється наступним чином: п.м. Посилання → вкл. Виноски → Вставити виноску. Виноски можна встановити в кінці тексту, в кінці сторінки або в кінці документу.

Формули та рівняння

- Розміщення формул та рівнянь відбувається без абзацного відступу, посередині сторінки, безпосередньо після тексту, в якому вони згадуються. Формули або рівняння виокремлюються знизу та зверху мінімум одним порожнім рядком. Нумерація формул і рівнянь у звіті проводиться за допомогою порядкової нумерації в межах розділу (за винятком формул і рівнянь, що наведені в додатках). Номер формули або рівняння у звіті

формується з відокремлених крапкою порядкових номерів розділу і формули або рівняння (наприклад, формула (7.2) – друга формула сьомого розділу). Номер формули або рівняння пишеться в правому крайньому положенні на одному рядку з ними та укладається в дужки. Безпосередньо під формулою надаються пояснення її складових в тому порядку, в якому вони в ній зазначені. При цьому пояснення до кожної складової пишеться з нового рядка, а перший рядок починається з абзацу словом «де» без двокрапки.

- Перенесення формули або рівняння на наступний рядок виконується лише на знаках операцій з їх повторенням на початку наступного рядка. При перенесенні формули або рівняння на знакові операції множення, необхідно використовувати знак «×». Формули, які не розділяються текстом, а наводяться одна за одною необхідно відокремлювати комою.
- Для того, щоб формула була розміщена посередині, а її номер справа, потрібно на горизонтальній лінійці встановити знаки табуляції для рядка з формулою, а потім у цьому рядку за допомогою клавіші TAB (використати перед формулою та за формулою перед номером в дужках) зробити вірне розташування.
- Для набору формул використовують спеціальні редактори: Microsoft Equation (надбудова MS Office) або MathType (окрема програма).

Посилання

- В тексті звіту посилання на літературні джерела позначаються укладеним в квадратні дужки порядковим номером за переліком посилань (наприклад, «... в працях [4-9] ...»). Також, можна наводити посилання на літературні джерела, використовуючи виноски – при цьому оформлення посилання повинно відповідати його бібліографічному опису за переліком посилань із зазначенням відповідного номера.
- Якщо в звіті використовуються посилання на розділи, підрозділи, пункти, підпункти, ілюстрації, таблиці, формули, рівняння або додатки, то необхідно зазначати їх номери. При посиланнях необхідно писати: «... у розділі 4 ...», «... за 4.1 ...», «... відповідно до 5.7 ...», «... у додатку С ...», «... на рисунку 2.2 ...»» або «... на рис. 4.7 ...», «... дивись 5.1 ...», «... у таблиці 4.3 ...», «... (див. табл. 4.3) ...», «... за формулою (6.3) ...», «... у рівняннях (1.2) – (1.5) ...».

Перелік умовних позначень, символів, одиниць, скорочень і термінів

- Перелік в звіті розміщується у вигляді стовпця. Умовні позначення, символи, одиниці, скорочення і терміни наводяться ліворуч в абетковому порядку. Праворуч наводиться їх детальна розшифровка.

Додатки

- Додатки оформлюються як продовження звіту і розміщуються або на його наступних сторінках, або у вигляді окремої частини. Додатки розташовуються в порядку появи на них посилань в тексті звіту. У разі оформлення додатків на наступних сторінках звіту, кожний з них має

починатися з нової сторінки. Заголовок додатоку друкується вгорі малими літерами з першої великої і центрується відносно тексту сторінки. Над заголовком додатку малими літерами з першої великої пишеться слово «Додаток __» і велика літера, що його позначає. Слово «Додаток __» центрується відносно тексту сторінки.

- У разі оформлення додатку (додатків) окремою частиною звіту, додаток супроводжується титульним аркушем, який оформлюється за наступним шаблоном: після номера частини великими літерами друкується слово «ДОДАТОК __» та його назва (якщо є) або слово «ДОДАТКИ».
- Позначаються додатки послідовно, великими літерами української абетки (наприклад, додаток Д). Додатки не позначаються наступними літерами: Ґ, Є, З, І, Ї, Й, О, Ч, Ь. Якщо додаток один, він позначається літерою А. Додатки мають наскрізну нумерацію сторінок, спільну з рештою звіту. Інколи текст додатків поділяють на пронумеровані в межах кожного додатку розділи, підрозділи, пункти і підпункти. Нумерація елементів додатку формується в межах кожного додатку за шаблоном: позначення додатку (літеру), крапка, номер елементу (наприклад. Д.3 – третій розділ додатку Д; А.3.6.5 – пункт 3.6.5 додатку А; Рисунок В.5 – п’ятий рисунок додатку В; Таблиця Б.1 – перша таблиця додатку Б; формула (Д.4) – четверта формула додатку Д.
- В разі існування літературних джерел, які цитуються тільки у додатках, необхідно розглядати їх окремо від тих, що цитують в основній частині звіту. Такі джерела фіксуються в переліку посилань, що формується наприкінці кожного додатку.
- Інколи в якості додатків звіту використовуються документи, які оформлюються у відповідності до вимог, що висуваються до документів певного виду і мають самостійне значення. В цьому разі у звіт вміщується копія документу без змін в оригіналі, перед якою вміщується аркуш з надрукованим посередині словом «ДОДАТОК __» і його назвою (за наявності). Порядковий номер сторінки друкується праворуч у верхньому куті аркуша, а сторінки копії документу нумеруються наскрізною щодо сторінок звіту нумерацією (при цьому власна нумерація сторінок документу не береться до уваги).

Закріплення матеріалу

1. Надати опис структури звіту (документу).
2. Описати правила нумерації структурних елементів звіту.
3. Описати правила оформлення рисунків та таблиць у звітах.
4. Описати правила оформлення формул у звітах.
5. Описати правила оформлення посилань у звітах.
6. Описати правила оформлення переліку літературних джерел у звітах.
7. Описати правила оформлення додатків.

РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА. ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Лекція 8

ТЕКСТОВІ ПРОЦЕСОРИ ТА ЕЛЕКТРОННІ ТАБЛИЦІ. ІНСТРУМЕНТИ ТАБЛИЧНОГО ПРОЦЕСОРА MS EXCEL

Мета лекції: розгляд компонентів операційних систем; розгляд файлової системи.

Зміст лекції

1. Інструменти табличного процесора MS EXCEL для обробки та графічного відображення результатів експерименту.
2. Інструменти табличного процесора MS EXCEL для створення баз даних та розв'язанні задач математичного програмування.
   2.1. Бази даних в MS Excel.
   2.2. Математичне програмування.
   2.3. Мова програмування Visual Basic в MS Excel.

Інструменти табличного процесора MS Excel для обробки та графічного відображення результатів експерименту

Структура таблиці і основні типи даних

- *Електрона таблиця* (ЕТ) – це програма, призначення якої полягає в обробці даних та автоматизації математичних обчислень. Дані, наведені в ЕТ, зазвичай носять економічний, бухгалтерський або статистичний характер. Електронна таблиця складається з комірок, що утворюються на перетині рядків і стовпців. Рядки ЕТ позначаються цифрами (1, 2, 3, ..., 65536), стовпці – літерами (A, B, C…Z, AA, AB, …, AZ – максимальна кількість стовпців дорівнює 256). У кожної комірки ЕТ є адреса (наприклад, B4 – адреса комірки, що утворюється перетином другого стовпця та 4 рядка).
- *Робоча таблиця* – це комірки, заповнені даними.
- *Робоча сторінка* – це сторінка, на якій розміщується робоча таблиця. Назва робочої сторінки відображається на ярлику внизу екрану (наприклад Лист 4).
- *Робоча книга* – це декілька робочих сторінок. Робоча книга зберігається у файлі з розширенням xls, xlsx.
- Основними типами даних, що вводяться користувачем у комірки є числа, текст і дати, а також формули для роботи з даними.

- Число на екрані може виглядати як заокруглене, з символом грошової одиниці, з комами або пропусками.
- Формат числа задається командою: п.к.м. (права кнопка миші) Формат комірки → вкл. (вкладинка) Число. Формат Числовий використовується для роботи з числами. З його допомогою можна задати кількість десяткових знаків після коми. Інші формати для роботи з даними – загальний, грошовий, фінансовий, дата, час, відсотковий, дробовий, експоненційний, текстовий, додатковий, всі формати. Наприклад, число 1703,49 у форматі користувача # ##0,00 ₽.; [Червоний] – # ##0,00 ₽ на екрані відобразиться у грошовому форматі 1703,49 ₽., а -1703,49 відобразиться червоним кольором. Символ 0 у форматі вказує, що у відповідній позиції буде відображатися конкретна цифра, символ # – лише значущі цифри, а пробіл означає відокремлення груп цифр.
- Ціла і дробова частини чисел, в залежності від налаштувань ОС Windows, відокремлюються крапкою або комою.
- Рядок формул розміщується над ЕТ і відображає дані / формулу, які вводяться або введені в комірку.
- Формули використовуються для виконання дій над вмістом комірок. Ознакою формули є символ «=» на її початку (наприклад, = A1+E12). Після введення формули в комірку вона відображається лише в рядку формул, а вміс комірки набуває вигляду результату обчислень.
- Для перегляду всіх формул таблиці задається режим відображення формул в комірках: вкл. Формули → кн. (кнопка) Показати формули.
- Для перегляду результатів обчислень, необхідно вимкнути режим відображення формул, натиснувши на кнопку Показати формули.
- *Відносні адреси* – це адреси комірок вигляду A7 або B5.
- Для прискорення виконання задач ЕТ надає можливість копіювати однотипні формули, а не вводити їх окремо в кожну комірку. При копіюванні формула вводиться в інші комірки та модифікується автоматично.
- *Автоматична модифікація формули* – це автоматична зміна відносних адрес, на які є посилання у формулі.
- Основними засобами автоматизації обчислень в ЕТ є копіювання формул і автоматичне переобчислення.

Числові операції. Складання математичних виразів

- Всі обчислення в MS Excel виконуються за допомогою формул. В якості знаків математичних і логічних операцій MS Excel використовує стандартні комп'ютерні символи операцій: «+» – додавання; «-» – віднімання; «*» – множення; «/» – ділення; «^» – піднесення до ступеня; при зміні порядку обчислення використовують дужки «( )»; «=» – дорівнює; «<>» – не дорівнює; «>» – більше; «<» – менше; « %» – відсоток; «>=» – більше або дорівнює; «<=» – менше або дорівнює.

- Формула складається з трьох частин: знака рівності (якщо його немає, це сприймається як звичайне введення в комірку); сукупності значень або посилань на комірки, з яких виконується розрахунок; операторів.
- Формули із складним обчисленням включають кілька операцій. При цьому треба враховувати порядок виконання цих операцій: формули обчислюються у напрямку зліва направо; множення та ділення завжди виконується до «+» або «-»; дія в дужках обчислюється в першу чергу.
- В MS Excel є однокроковий метод додавання – кнопка Автосума на панелі інструментів.
- Щоб встановити у формулі порядок групування комірок і діапазонів для виконання розрахунків, використовуються оператори посилань (ОП). ОП дозволяють об'єднувати комірки та обробляти їх як ціле, посилатися на загальну область перетину діапазонів.
- Вкладинки діалогового вікна Формат комірок: Число, Вирівнювання, Шрифт, Границя, Заливка, Захист.
- Вкладка Число містить список різних числових форматів і визначає кількість десяткових знаків, типи роздільників груп розрядів, режими вигляду від'ємних чисел.
- Існує десять спеціальних форматів:
  - Дата – для оформлення дати й часу;
  - Дробовий – визначає тип оформлення, що базується або на числі знаків до та після відокремлювача, або на вигляді дробової частини (половин, четвертих, десятих);
  - Додатковий – для оформлення чисел, що не мають математичного значення (№ телефону, поштовий індекс);
  - Текстовий – перетворює число в текстове подання (воно не може бути числом і не може бути використане в обчисленнях). Наприклад, для числа 0777 у текстовому форматі 0 не зникає.
- *Функція* – це формула, тому вона починається з «=», потім йде ім'я функції, потім аргумент (= СУММ (D6:D11)). Доступ до функцій здійснюється за допомогою палітри формул із списком функцій, який відображається в лівій верхній частині вікна при введені в комірку знаку «=».
- Якщо потрібної функції немає в списку використовують п. (пункт) Вставити функції, який активує діалогове вікно Майстер функцій, або кнопку Вставка функцій на панелі інструментів.
- При копіюванні формули з однієї комірки в іншу, посилання на комірки коригуються автоматично. Це відносна адресація. Якщо посилання на комірку не залежіть від розміщення формули на листі – це абсолютне посилання. Для заміни відносного посилання на абсолютне, вводять знак $ перед тією частиною посилання, що повинна стати абсолютною.
- $A1 – завжди посилатися на стовпець А, посилання на рядок може змінюватися.

- A$1 – завжди посилатися на рядок 1, посилання на стовпець можуть змінюватися.
- A1 – завжди посилатися на комірку A1.

Логічні функції

- *Логічний вираз* – це форма запису умови: простої або складної.
- Розгалуження в ЕТ реалізуються за допомогою функції ЯКЩО. Дана функція використовується у формулах та має наступну структуру: ЯКЩО(<логічний вираз>; <вираз 1>; <вираз 2>).
- Якщо умова істинна, то функція набуває значення першого виразу, інакше – другого.
- І <вираз 1>, і <вираз 2> може бути функцією ЯКЩО, забезпечуючи тим самим вкладеність. В більшості випадків <вираз 1> або <вираз 2> – це адреса комірки, яка містить деяке значення або конкретне число.
- В MS Excel прості умови записуються за допомогою операцій порівняння (=, >, <, <=, >=, <>), визначених над виразами (наприклад, 17 < 25, A8 >= 2).
- В MS Excel складні умови записуються за допомогою логічних функцій AND(< умова 1>; <умова 2>; ...) та OR(<умова1>; <умова 2>;...).
- Функція AND є істиною, якщо всі умови в її списку є істинними.
- Функція OR є істиною, якщо хоча б одна з умов в її списку є істиною.
- Функція NOT – змінює на протилежне значення свого аргументу.
- Функція ХИБНО – видає логічне значення false
- Функція ІСТИНО – повертає логічне значення true.
- Функції AND, OR, NOT використовуються для реалізації операцій бульової алгебри.

Матриці і розв'язок лінійних рівнянь

- Формула масиву (матриці) може виконати кілька обчислень, а потім повернути одне значення або групу значень. Формула масиву обробляє кілька наборів значень, які називають аргументами масиву. Кожний аргумент масиву повинен включати однакове число рядків і стовпців. Формула масиву створюється так само, як і інші формули, з тією різницею, що для введення такої формули використовуються клавіші Ctrl+Shift+Enter. Константи масиву можуть використовуватися замість посилань, якщо не потрібно вводити кожну постійну величину в окрему комірку на аркуші. Деякі вбудовані функції є формулами масиву, і, для отримання вірних результатів, їх необхідно вводити як масиви.
- Деякі функції повертають масиви значень або потребують масив значень в якості аргументу. Для обчислення декількох значень за допомогою формули масиву необхідно ввести масив у діапазон комірок, що складається з того ж числа рядків або стовпців, що і аргументи масиву, виділити його і натиснути Ctrl+Shift+Enter.
- Основні функцій для роботи с матрицями в Excel:

- МОПРЕД – повертає визначник матриці (матриця зберігається в масиві).
- МОБР – повертає обернену матрицю для матриці, що зберігається в масиві.
- МУМНОЖ – повертає добуток матриць (матриці зберігаються в масивах). Результатом є масив з таким самим числом рядків, як <масив1> і з таким самим числом стовпців, як <масив2>.
- ТРАНСП – повертає вертикальний діапазон комірок у вигляді горизонтального та навпаки.
- Функція ТРАНСП повинна бути введена як формула масиву в інтервал, що має стільки ж рядків і стовпців, скільки стовпців і рядків має аргумент масиву. Функція ТРАНСП використовується для того, щоб поміняти орієнтацію масиву на робочому аркуші з вертикальної на горизонтальну та навпаки.

Статичні функції. Пакет аналізу: «Статистика»

- В MS Excel надана велика кількість статистичних функцій. Деякі з них є вбудованими, інші доступні тільки після додаткового встановлення Пакету аналізу.
- Статистичні функції дозволяють виконувати статистичний аналіз діапазонів даних.
- Дістатися до переліку статистичних функцій в MS Excel можна наступним чином: натиснути кн. $f_x \rightarrow$ у вікні Категорії Майстру функцій вибрати Статистичні (при цьому у вікні Функції відкривається весь їх перелік).
- Деякі статистичні функцій в MS Excel:
  - FРАСП – повертає F-розподіл ймовірності.
  - ВЕРОЯТНОСТЬ – повертає ймовірність того, що значення діапазону знаходяться в заданих границях.
  - ДИСП – повертає дисперсію за вибіркою.
  - ДИСПР – повертає дисперсію для генеральної сукупності.
  - ДОВЕРИТ – повертає довірчий інтервал для середнього значення за генеральною сукупністю.
  - КВАДРОТКЛ – повертає суму квадратів відхилень.
  - КВПИРСОН – повертає квадрат коефіцієнту кореляції Пірсона.
  - КОВАР – повертає коваріацію, тобто середнє добутків відхилень для кожної пари точок.
  - КОРРЕЛ – повертає коефіцієнт кореляції між двома множинами даних.
  - ЛИНЕЙН – повертає параметри лінійного тренду.
  - МАКС – повертає максимальне значення із списку аргументів.
  - МЕДИАНА – повертає медіану заданих чисел.
  - МИН – повертає мінімальне значення із списку аргументів.
  - МОДА – повертає значення моди множини даних.
  - ПУАССОН – повертає розподіл Пуассона.

- СРЗНАЧ – повертає середнє арифметичне аргументів.
- СРОТКЛ – повертає середнє абсолютних значень відхилень точок даних від середнього.
- СТАНДОТКЛОН – оцінює стандартне відхилення за вибіркою.
- СТЬЮДРАСП – повертає t-розподіл Ст'юдента.
- ФИШЕР – повертає перетворення Фішера.
- ЭКСЦЕСС – повертає ексцес множини даних.

Побудова графіків

- Діаграми призначаються для графічного відображення числових даних у звітах, презентаційних, рекламних сторінках тощо та стандартні і нестандартні.
- До типів стандартних діаграм належать: гістограма, графік, кругова, поверхнева, кільцева, біржова, циліндрична, конічна, точкова діаграми та діаграма з областями. В свою чергу, кожен тип стандартної діаграми також має декілька різновидів. До типів нестандартних діаграм належать: блакитна кругова, дерев'яна та блоки з областями.
- Найчастіше використовують стовпчикові, кругові та точкові стандартні діаграми різних видів.
- *Кругова діаграма* – це діаграма, яка відображає один виділений рядок чи стовпець числових даних з таблиці у вигляді круга з секторами та показує співвідношення частин і цілого, де ціле відповідає 100 %.
- *Точкова та діаграма-графік* (X-Y діаграма) – це діаграми, призначені для побудови традиційних математичних графіків. Вони дозволяють побудувати на одній координатній площині графіки відразу декількох функцій. Якщо значення аргументу заносяться в перший стовпець, а значення функцій – в другий, третій тощо, то перший стовпець у таблиці інтерпретується програмою як вісь X, а інші – як значення однієї чи декількох функцій уздовж осі ординат. В такій таблиці кількість рядків повинна бути більшою за кількість стовпців.
- *Гістограма* (стовпцева діаграма) – це діаграми, яка надає числові дані з вибраних стовпців таблиці у вигляді стовпчиків. В більшості випадків гістограма використовується для представлення змін у часі або просторі.
- Більшість діаграм мають дві осі: горизонтальну, на якій відкладаються категорії, та вертикальну, де відкладаються значення. У об'ємних діаграм є третя вісь, де відкладаються ряди.
- Складовими діаграми є: область побудови, область об'єкта, легенда, заголовок, назва осей та їх форматування тощо.
- Діаграми будують за допомогою вкладинки Вставка/Діаграми.
- Рекомендують перед побудовою діаграми виокремлювати діапазони з даними для графічного відображення. Це, в більшості випадків, суміжні рядки або стовпці (часто з назвами). Для виділення несуміжних діапазонів, потрібно натиснути на клавішу Ctrl.

- Контекстне меню елементів дозволяє внести зміни у вже побудовану діаграму.

Інструменти табличного процесора MS Excel для створенні баз даних та розв'язанні задач математичного програмування

Бази даних в MS Excel

- *База даних* (БД) – це програма, яка містить зв'язані одна з одною частини інформації. Наприклад, можна створити БД імен та адрес співробітників. Вона буде містити ім'я і прізвище кожного співробітника, його адресу з вказаними містом, областю тощо.
- Найбільш розумний спосіб групування цих даних – розміщення всієї зв'язаної інформації в одному рядку, а однорідної інформації – в одному стовпці. Тоді повна інформація про кожного співробітника буде займати один рядок, а стовпці цієї таблиці будуть призначені для запису окремих елементів цієї інформації: імені, прізвища, адреси, міста та області. Таким чином, перший стовпець цієї таблиці буде містити ім'я співробітника, другий – його прізвище тощо.
- Повні рядки БД називаються записами, а окремі її елементи – полями. Таким чином, кожний співробітник буде представлений у БД записом. Кожен елемент такого запису: ім'я, прізвище або займаєма посада – є полем. Інформація повинна міститися в кожному полі кожного запису; іншими словами, повинна бути присутньою повна інформація по кожному співробітнику. Тобто можна сказати, що рядки БД є синонімами запису, а стовпці – синонімами поля.
- Формат таблиці повинен залишатися незмінним.
- *Пласька БД* – це двомірна БД, тобто рядки і стовпці одного робочого листа. Плаські БД можна створити за допомогою інструментарію MS Excel.
- *Реляційна БД* – стає тривимірною. Це означає, що данні зберігаються у великій кількості таблиць чи файлів, зв'язаних між собою за допомогою однієї певної ознаки (наприклад, імені).
- Розмір БД в MS Excel обмежується розміром робочої області. Якщо БД містить заголовок, то кількість можливих записів обмежується числом 65535.
- Створення структури БД складається з наступних кроків:
  - заповнення першого рядка БД заголовками стовпців, які називаються іменами полів (рекомендується використовувати тільки унікальні заголовки);
  - визначення стовпця робочого аркуша, який буде першим стовпцем таблиці даних;
  - розміщення всіх стовпців поруч один з одним;
  - використання окремого стовпця для кожного поля або елемента інформації;
  - налаштування параметрів форматування після вводу всіх заголовків.

- З даними БД можна виконувати наступні типові дії:
  - впорядковувати в деякому стовпці рядки за зростанням чи спаданням значень;
  - шукати дані за деяким критерієм;
  - підбивати підсумки.

Впорядкування

- Спочатку обирають або частину таблиці з назвами полів та даними, або цілу таблицю (без заголовка таблиці і рядків з підсумками).
- Сортування виконується наступним чином: п. Дані → вкл.. Сортування і фільтр → Сортування. Після цього отримують список назв полів, з якого необхідно вибрати потрібну назву (наприклад Країна) та задати порядок сортування: за зростанням або спаданням. В результаті отримують таблицю, де рядки з назвами країн будуть впорядковані за алфавітом або в зворотному порядку.

Фільтрація: автофільтр, розширений фільтр та фільтрація з діапазоном критеріїв

- *Автофільтр* – інструмент для знаходження і виокремлення інформації. Для проведення операції пошуку необхідної інформації за допомогою Автофільтру необхідно виконати наступні дії: п. Данні → вкл. Сортуванняі фільтр → Фільтр. Після цього зверху кожного стовпця БД з'являться кнопки розкриття списку. Необхідно натиснути на кнопці того стовпця, в якому буде здійснюватись пошук. З'явиться розгортаємий список.
- *Розширений фільтр* – дозволяє виконувати пошук в БД не тільки за одним елементом але й за елементами, які входять в певний діапазон або елементам, які задовольняють певні критерії. Для проведення операції пошуку необхідної інформації за допомогою Розширеного фільтру необхідно виконати наступні дії: п. Данні → вкл. Сортування і Фільтр → Додатково.
- Розширена фільтрація за допомогою діапазонів критеріїв або точного критерію пошуку дозволяє досягти максимальної точності фільтрації інформації БД.

Проміжні підсумки

- За допомогою інструментарію MS Excel можна отримати проміжні і загальні підсумки числових елементів бази даних:
  - Сума – повертає значення суми елементів групи.
  - Кількість значень – повертає кількість елементів групи.
  - Середнє – повертає середнє значення групи.
  - Максимум – повертає максимальне значення групи.
  - Мінімум – повертає мінімальне значення групи.
  - Добуток – повертає добуток всіх членів групи.
  - Кількість чисел – повертає кількість всіх числових елементів.
  - Зміщене відхилення – повертає значення відхилення по вибірці з бази даних.

- Незміщене відхилення – повертає значення відхилення по генеральній сукупності з бази даних.
- Зміщена дисперсія – повертає значення дисперсії по вибірці з бази даних.
- Незміщена дисперсія – повертає значення дисперсії по генеральній сукупності з бази даних.

- Для створення проміжних підсумків необхідно спочатку відсортувати дані у стовпці, для якого будуть підведені проміжні підсумки, а потім вибрати команду п. Данні → вкл. Структура → Проміжні підсумки.
- У діалоговому вікні Проміжні підсумки Можна встановити наступні параметри:
  - *При кожній зміні в*. З цього списку, що розкривається, обирається поле, по якому будуть підводити проміжні підсумки. Можна вибрати тільки одне поле.
  - *Операція*. З цього списку, що розкривається, вибирається функція, яка буде використовуватися при створенні підсумків.
  - *Додати підсумки по*. В цьому списку вибираються поля, в яких будуть підводити проміжні підсумки. Кількість обраних полів не обмежується. Проте, при використанні функції Сума результат буде відображатися тільки для полів із числовими значеннями.
  - *Замінити поточні підсумки*. Всі розраховані раніше значення замінюються новими результатами. Якщо проміжні підсумки для даного робочого аркуша проводяться в перший раз, то цю відмітку необхідно зняти.
  - *Кінець сторінки між групами*. Кожна група, для якої підводять проміжні підсумки, розміщується на окремій сторінці.
  - *Підсумки під даними*. Встановлення цієї відмітки призведе до відображення проміжних підсумків внизу групи. В зворотному випадку проміжні підсумки будуть розміщені вгорі групи.

Консолідація даних

- Процедура консолідації використовується з метою зв’язку різних вихідних областей, підбиття підсумків для них та об’єднання їх в одну таблицю.
- Для виконання процедури консолідації необхідно зробити наступні кроки: п. Дані → вкл. Робота з даними → Консолідація.
- Розрізняють декілька видів консолідації: консолідація даних з використанням тривимірних посилань, консолідація даних за розташуванням, консолідація даних за категоріями, консолідація даних шляхом створення зведеної таблиці.

Математичне програмування

- Табличні процесори використовуються для прогнозування поведінки складних систем. Найбільш простий спосіб прогнозування – рішення задач методом «Що буде, якщо ...?», при якому задаються різні набори значень деяких вихідних параметрів системи і оцінюється значення розрахункових.

Багаторазові розрахунки дозволяють оцінити, як реагує система, що вивчається, коли вона описана у вигляді математичних співвідношень, на ті чи інші зміни в умовах її функціонування і вибрати те рішення, яке найбільше задовольняє необхідним вимогам, – оптимальне рішення.

- Проте в задачах прогнозування число можливих варіантів дій часто настільки велике, що вручну перебрати все неможливо. В таких ситуаціях доцільно використання спеціальних математичних методів. З цією метою в сучасні табличні процесори включаються потужні математичні блоки, що дозволяють проводити статистичну обробку даних, вирішувати окремі рівняння і системи рівнянь, а також завдання математичного програмування (оптимізаційні задачі).
- *Математичне програмування* – математична дисципліна, яка займається вивченням методів розв'язання, аналізу та використання задач зі знаходження екстремуму функції на множині допустимих варіантів функції. Математичне програмування використовують при розв'язанні різноманітних практичних задач. Математичне програмування – це частковий випадок системного аналізу однієї чітко вираженої мети, досягнення якої здійснюється за одним критерієм.
- Послідовність використання техніко-математичних та/або економіко-математичних моделей:
  - формується технічна та/або економічна проблема;
  - створюється математична модель задачі, в якій логічні зв'язки технічної та/або економічної моделі перетворюються на математичні співвідношення: функції, рівняння, нерівності;
  - розв'язується математична задача, перевіряється рішення;
  - розв'язок перекладається на технічну та/або економічну мову і аналізується результат.
- В загальному вигляді математичне формулювання задачі виглядає таким чином:
  - необхідно знайти найбільше або найменше значення цільової функції $f(x_1, x_2, x_3\ldots, x_n)$;
  - при цьому треба виконати умови $g_i (x_1, x_2, x_3\ldots, x_n) \le b_i$, $i = 1,2,3\ldots,m$; де: f, g_i – відомі функції, b_i – деякі дійсні числа, $n > m$.
- Задачу оптимізації з точки зору техніки та/або економіки можна сформулювати таким чином: знайти такі значення змінних, що надають ефективності діяльності максимального або мінімального значення, за умов виконання обмежень, які пов'язані зі змінними, за допомогою яких, здійснюється управління діяльністю.
- Конкретна мета, поставлена в технічній та/або економічній задачі, пояснюється цільовою функцією (критерієм ефективності), екстремум якої і треба знайти.
- Обмеження відображають умови, при розв'язанні задачі.

- Змінні, з яких будується цільова функція, та на які накладаються обмеження, використовують як «інструмент», за допомогою якого досягається той чи інший варіант цілі. Якщо змінні задовольняють усім обмеженням, то отриманий варіант називають припустимим (допустимим).
- Задача математичного програмування полягає в тому, щоб з усіх допустимих варіантів значень інструментальних змінних (невідомих моделі) знайти такі, при яких функція цілі (критерій оптимальності) досягає екстремуму.
- Розв'язати задачу – це знайти її оптимальне рішення або з'ясувати його відсутність.
- Функція цілі (критерій оптимальності) повинна об'єктивно характеризувати суспільно-корисну значущість соціально-економічного явища або процесу. Критерій оптимальності можливо з'ясувати лише з сутності проблеми – задача, яку розв'язує фахівець. Принципово неможливо визначити цільову функцію на етапі розв'язання задачі математиком. Класифікація моделей задач математичного програмування залежить від властивостей функції цілі та функцій обмежень.
- Якщо функція цілі та усі функції обмежень лінійні, така задача математичного програмування має назву задачі лінійного програмування; якщо ж хоча б одна з функцій нелінійна, така задача має назву задачі нелінійного програмування.
- Якщо у математичній моделі враховується поетапно час, така задача має назву задачі динамічного програмування; в іншому випадку – задачі статичного програмування.
- В залежності від того, який характер мають вихідні дані моделі – детермінований або стохастичний – задачі мають назву відповідно детермінованого та стохастичного програмування.
- Серед задач нелінійного програмування особливо досконало досліджені задачі опуклого програмування – задачі знаходження екстремуму опуклої функції, заданої на опуклій замкненій множині. В свою чергу, серед задач опуклого програмування найпростіші і найдосконаліше досліджені задачі квадратичного програмування, в яких функція цілі – квадратична, а обмеження – лінійні.
- Якщо змінні задачі математичного програмування приймають тільки цілочисельні значення, така задача має назву задачі цілочисельного програмування; у іншому випадку – задачі неперервного програмування.
- В задачах дробово-лінійного програмування цільова функція є співвідношенням двох лінійних функцій, а обмеження – лінійні.
- Якщо в задачі математичного програмування відсутні усі обмеження, така задача має назву задачі безумовного програмування.
- Табличний процесор MS Excel також має інструменти для розв'язання задач математичного програмування: п. Дані → вкл. Робота з даними → Аналіз «що якщо?».

Мова програмування Visual Basic в MS Excel

- При роботі з MS Office часто потрібно розв'язувати нестандартні задачі або автоматизувати процес обробки об'єктів. Частково цю проблему вирішують макроси, проте велику програму можна написати за допомогою мови Visual Basic (VB), яка дозволяє створювати професійно оформлені проєкти, автоматизувати операції з програмами, здійснювати обробку інформації, працювати лише з необхідними даними. Кожна з програм MS Office підтримує VB, у тому числі і MS Excel.

Створення макросів

- Коли ви помічаєте, що необхідно повторювати одну й ту саму операцію знову і знову, вам потрібно використовувати макрос. До найбільш відомих типів макросів належать:
- Макроси друку – для автоматизації вибору різних діапазонів друку, параметрів розташування сторінок, нижніх та верхніх колонтитулів тощо.
- Макроси вводу даних – для вводу текстів, числових даних та формул безпосередньо у комірки всього робочого аркушу.
- Макроси форматування – для форматування всієї робочої книги.
- Макроси побудови діаграм – для автоматизації процесу створення діаграм або форматування та додавання легенди до діаграм.
- MS Excel має інструменти для створення макросів: п. Вид → Макроси → Запис макросу. З'явиться вікно діалогу Запис макросу.
- Для того, щоб переглянути макрос, написаний мовою Visual Basic, необхідно увійти у відповідний редактор: п. Вид → вкл. Макроси → д.в. Макроси → Обрати потрібний макрос → Увійти.

Закріплення матеріалу

1. Описати структуру таблиці та типів даних в MS Excel.
2. Описати процедуру складання математичних виразів.
3. Вказати відмінність між роботою із звичайними формулами та формулами призначеними для операцій над матрицями?
4. Описати процедуру побудови графіків.
5. Описати інструменти MS Excel для створення і роботи з пласькими базами даних.
6. Що таке математичне програмування?
7. Надати класифікацію задач математичного програмування та описати засоби MS Excel для їх розв'язання.
8. Що таке макрос?
9. Описати інструменти MS Excel для створення макросів.

РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА. ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Лекція 9

РОЗРОБКА ТА ЕКСПЛУАТАЦІЯ БАЗ ДАНИХ. КОНЦЕПЦІЇ ПІДТРИМКИ БД ПРИ ЇЇ ФУНКЦІОНУВАННІ. СХЕМИ ОПИСУ ДАНИХ

Мета лекції: розгляд основних понять теорії баз даних.

Зміст лекції

1. Бази даних.
2. Адміністрація та управління базами даних.
3. Інфологічна модель даних «сутність – зв'язок».
4. Даталогічні моделі даних.

Бази даних

Приклад бази даних

- *Дані* – це інформація, що знаходиться в пам'яті комп'ютера або на машинних носіях.
- *Обробка даних* – це всі операції з даними, що необхідні для одержання бажаного результату.
- Обробка даних передбачає:
  - введення даних до системи;
  - вибір даних за критеріями і параметрами;
  - перетворення структури даних;
  - редагування даних;
  - виведення даних у табличному або іншому зручному для користувача вигляді.
- *База даних (БД)* – це сукупність взаємозалежних даних певної предметної галузі, збережених у пам'яті комп'ютера і організованих таким чином, що ці дані можуть бути використані для розв'язання різних завдань багатьма користувачами.
- Прикладом бази даних може бути інформаційна система навчального процесу університету.

Вимоги до бази даних

- *Ненадлишковість даних* передбачає, що будь-які дані мають зберігатися в БД лише в одному екземплярі.
- *Спільне використання даних* – одні й ті самі дані БД можуть використовуватися декількома користувачами (задачами).

- *Можливість розширення* бази даних може бути здійснене за рахунок: збільшення числа екземплярів однотипних даних та введення до БД нових типів об'єктів або нових типів взаємозв'язків.
- *Простота роботи з базою даних* (вимоги: структура даних має бути логічною і зрозумілою; операції доступу до даних здійснюються за допомогою зрозумілих і чітко окреслених функцій; різноманітні сервісні операції (копіювання, перенесення БД, розширення бази тощо) виконуються без значних витрат часу та зусиль користувачів).
- *Ефективність доступу* до бази даних – швидкість доступу до даних при обмежених ресурсах комп'ютерної системи.
- *Цілісність* бази даних – її готовність до роботи. Розрізняють фізичну цілісність, логічну цілісність та актуальність даних.
- *Захист даних* – неможливість несанкціонованого доступу до БД. Розрізняють повний захист даних та захист даних від модифікації.

Адміністрація та управління базами даних

Адміністрація бази даних

- Адміністрація БД – це колектив, відповідальний за правильну роботу БД.
- Функції адміністрації:
  - проєктування структури БД;
  - вибір засобу представлення даних;
  - виконання обслуговуючих функцій;
  - планування розвитку БД і пов'язаний із цим вибір нових засобів обчислювальної техніки;
  - консультації користувачів щодо використання БД;
  - контроль користувачів, що працюють із БД, врегулювання різноманітних конфліктних ситуацій.

Система управління базою даних

- *СУБД* – це системне програмне забезпечення, інструмент для розробки прикладних програм і підтримки БД.
- Функції типової СУБД:
  - опис даних (схема бази даних) – опис незмінних властивостей даних БД за допомогою обмеження на припустимі операції з даними;
  - маніпулювання даними;
  - завантаження бази і формування звітів;
  - мова запитів;
  - діалогові засоби;
  - мультидоступ до БД.

Опис даних

- *Інфологічна модель* – узагальнений, неформальний опис створюваної БД, що є об’єднанням різноманітних уявлень про зміст БД, отриманих в результаті опитування користувачів, а також уявлення про дані адміністрації БД (людиноорієнтована, незалежна від фізичних параметрів середовища збереження даних).

- Властивості інфологічної моделі:
  - вся множина однотипних записів представляється одним абстрактним записом (типом запису);
  - кожному типу запису надається ім’я і список атрибутів;
  - множині однотипних зв’язків у концептуальній схемі відповідає один тип зв’язку;
  - обмеження цілісності, як обмеження на припустимі значення атрибутів.
- *Даталогічна модель* (комп'ютерноорієнтована) – опис, створений адміністрацією БД за інфологічною моделлю даних мовою опису даних конкретної СУБД для здійснення фізичного доступу до них.
- *Фізична модель* (комп'ютерноорієнтована) – модель, за допомогою якої СУБД дає можливість програмам і користувачам здійснювати доступ до збережених даних лише за їх іменами, не зважаючи на фізичне розташування цих даних.

Інфологічна модель даних «сутність – зв'язок»

Основні елементи моделі

- *Сутність* – будь-який помітний об'єкт (об'єкт, що можна відрізнити від іншого), інформацію про який необхідно зберігати в базі даних.
- Поняття сутності:
  - тип сутності – набір однорідних особистостей, предметів, подій або ідей, що виступають як ціле;
  - екземпляр сутності – це конкретна річ у наборі.
- *Атрибут* – характеристика сутності, яка має ім'я.
- *Ключ* – мінімальний набір атрибутів, за значеннями яких можна однозначно знайти необхідний екземпляр сутності.
- *Зв'язок* – асоціювання двох або більше сутностей.

ER-діаграми

- ER-діаграми представлені відповідними графічними позначеннями наступних елементів: стрижень, асоціація, позначення, характеристика, ключ.

Види зв'язків між сутностями

- Зв’язок ОДИН-ДО-ОДНОГО (1:1) – в кожний момент часу кожному екземпляру сутності A відповідає 1 або 0 екземплярів сутності B.
- Зв’язок ОДИН-ДО-БАГАТЬОХ (1:M) – одному представнику сутності A відповідають 0, 1 або декілька екземплярів сутності B.
- Оскільки між двома сутностями можливі зв'язки в обох напрямках, то існує ще два типи зв'язку – БАГАТО-ДО-ОДНОГО (M:1) і БАГАТО-ДО-БАГАТЬОХ (M:N).

Класи сутностей

- *Стрижнева сутність* (стрижень) – це незалежна сутність.

- *Асоціативна сутність* (асоціація) – це зв'язок вигляду «багато-до-багатьох» та «один-до-багатьох» між двома або більше сутностями або екземплярами сутності.
- *Характеристична сутність* (характеристика) – це зв'язок вигляду «багато-до-одного» або «один-до-одного» між двома сутностями.
- *Позначаюча сутність* або *позначення* – це зв'язок вигляду «багато-до-одного» або «один-до-одного» між двома сутностями, який відрізняється від характеристики тим, що не залежить від позначеної сутності.

Поняття «ключ бази даних»

- *Ключ* – це мінімальний набір атрибутів, за значеннями яких можна однозначно знайти необхідний екземпляр сутності. Мінімальність означає, що виключення з набору будь-якого атрибута не дозволяє ідентифікувати однозначно відповідну сутність.
- *Первинний ключ* – це ключ, складений з мінімального числа атрибутів. Первинний ключа має бути унікальним.
- *Правило для зовнішніх ключів*: якщо сутність С зв'язує сутності А та В, то вона повинна включати зовнішні ключі, що відповідають первинним ключам сутностей А та В. Значення зовнішнього ключа повинно або бути рівним значенню первинного ключа цілі, або бути цілком невизначеним, тобто кожне значення атрибута, що бере участь у зовнішньому ключі, повинно бути невизначеним.

Даталогічні моделі даних

Ієрархічна модель

- Види ієрархічних відношень:
  - відношення «частина – ціле»;
  - відношення роду або виду;
  - відношення підпорядкованості.
- Об'єкти, пов'язані ієрархічними відношеннями, утворюють дерево – орієнтований граф, у якого є тільки одна вершина, не підпорядкована ніякій іншій вершині (цю вершину прийнято називати коренем дерева); будь-яка інша вершина графа підпорядкована тільки одній іншій вершині.

Мережева модель

- У мережевій моделі даних будь-який об'єкт може бути і головним, і підпорядкованим.
- *Власник набору* – це головний об'єкт мережевої моделі.
- *Член набору* – це підпорядкований об'єкт мережевої моделі.
- Один і той самий об'єкт може одночасно виступати й у ролі власника, і в ролі члена набору. Це означає, що кожний об'єкт може брати участь у будь-якому числі взаємозв'язків.

Реляційна модель

- *Реляційна база даних* – це сукупність відношень, що містять всю інформацію, яка повинна зберігатися в БД.

- Вимоги до реляційної бази даних:
  - кожна таблиця складається з однотипних рядків і має унікальне ім'я;
  - рядки мають фіксоване число полів (стовпців) і значень (множинні поля і повторювані групи неприпустимі);
  - рядки таблиці обов'язково відрізняються один від одного хоча б єдиним значенням, що дозволяє однозначно ідентифікувати будь-який рядок такої таблиці;
  - стовпцям таблиці однозначно присвоюються імена і в кожному з них розміщуються однорідні значення даних;
  - повний інформаційний зміст бази даних представляється у вигляді явних значень даних і такий метод представлення є єдиним;
  - при виконанні операцій з таблицею її рядки і стовпці можна обробляти в будь-якому порядку, не зважаючи на їх інформаційний зміст.
- Недоліки реляційних баз даних:
  - модель не надає достатніх засобів для представлення змісту даних;
  - для багатьох програм важко моделювати предметну галузь на основі плоских таблиць;
  - реляційна модель даних не пропонує апарату для поділу сутностей і зв'язків.

Алгоритм одержання реляційної схеми з ER-схеми

1) Представити кожний стрижень (незалежну сутність) таблицею бази даних (базовою таблицею) і специфікувати первинний ключ цієї базової таблиці.
2) Представити кожну асоціацію (зв'язок вигляду "багато-до-багатьох" або "багато-до-багатьох-до-багатьох" тощо між сутностями) як базову таблицю. Використати в цій таблиці зовнішні ключі для ідентифікації учасників асоціації і специфікувати обмеження, пов'язані з кожним із цих зовнішніх ключів.
3) Представити кожну характеристику як базову таблицю з зовнішнім ключем, що ідентифікує сутність, яка описується цією характеристикою. Специфікувати обмеження на зовнішній ключ цієї таблиці та її первинний ключ.
4) Представити кожне позначення, що не розглядалося в попередньому пункті, як базову таблицю з зовнішнім ключем, що ідентифікує позначену сутність. Специфікувати обмеження, пов'язані з кожним таким зовнішнім ключем.
5) Представити кожну властивість як поле в базовій таблиці, що представляє сутність, яка безпосередньо описується цією властивістю.
6) Для виключення ненавмисних порушень виконується процедура нормалізації, яка ґрунтується на тому, що єдиними функціональними залежностями в будь-якій таблиці мають бути залежності вигляду K→F, де K – первинний ключ, а F – деяке інше поле. Зауважимо, що це випливає з визначення первинного ключа таблиці, відповідно до якого K→F завжди

має місце для всіх полів такої таблиці. "Один факт в одному місці" говорить про те, що не мають сили ніякі інші функціональні залежності. Мета нормалізації полягає саме в тому, щоб позбутися всіх "інших" функціональних залежностей, тобто таких, що мають інший вигляд, ніж K→F.

7) Якщо в процесі нормалізації був зроблений поділ будь-яких таблиць, то варто модифікувати інфологічну модель бази даних і повторити перераховані кроки.
8) Вказати умови цілісності бази даних і дати (якщо це необхідно) стислий опис отриманих таблиць і їх полів.

Закріплення матеріалу

1. Що таке дані?
2. Що передбачає обробка даних?
3. Що таке база даних?
4. Назвати вимоги до організації бази даних.
5. Що розуміють під адміністрацією бази даних?
6. Які функції виконує адміністрація баз даних?
7. Що таке система управління базою даних?
8. Назвати функції типової СУБД.
9. Що таке інфологічна модель?
10. Назвати властивості інфологічної моделі.
11. Що таке даталогічна модель?
12. Що таке фізична модель?
13. Назвати основні елементи інфологічної моделі даних «сутність-зв’язок».
14. Назвати елементи розширеної мови ER-діаграм.
15. Назвати види зв’язків між сутностями.
16. Які класи сутностей вам відомі?
17. Що таке ключ бази даних?
18. Що таке первинний ключ бази даних та в чому його призначення?
19. Що таке зовнішній ключ бази даних та в чому його призначення?
20. Назвати різновиди даталогічної моделі даних.
21. Назвати види ієрархічних відношень.
22. Назвати вимоги до реляційної бази даних.
23. Вказати недоліки реляційної баз даних.
24. Описати алгоритм отримання реляційної схеми з ER-схеми.
25. В чому суть процедури нормалізації?
26. Які нормальні форми вам відомі?
27. В чому суть процедури денормалізації?

РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА. ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Лекція 10

ІНСТРУМЕНТИ СТВОРЕННЯ БД. MS ACCESS

Мета лекції: розгляд інструментів створення баз даних; знайомство з основами мови SQL.

Зміст лекції

1. Створення та редагування основних об'єктів в MS Access.
2. Мова SQL.
   2.1. Оператори маніпуляції даними.
   2.2. Оператори опису даних.

Створення та редагування основних об'єктів

- *Процесор обробки баз даних MS Access* – програма для створення і підтримки баз даних довільного рівня складності. Вона має стандартний інтерфейс та підтримує інтеграцію з іншими програмами MS Office.
- Більшість процедур в Access здійснюється за допомогою меню, яке складається з таких пунктів: Файл, Головна, Створення, Зовнішні дані, Робота з базами даних, Робота з таблицями.

Основні об'єкти баз даних

- *Таблиці* – набір даних, що мають двовимірну структуру. Кожна таблиця містить декілька стовпців з назвами, які називаються полями. Сама інформація міститься в рядках.
- *Запити* – своєю структурою аналогічні до таблиць, хоча можуть мати розрахункові поля. Як правило, запити створюються шляхом обробки інформації з однієї чи декількох таблиць або запитів.
- *Форми* – створюються для полегшення вводу інформації, її редагування, а також для спілкування з користувачем та реакції на його дії.
- *Звіти* – містять кінцеву проаналізовану інформацію, іноді агреговану в певний спосіб, створену на основі таблиць та звітів. Виведення інформації контролюється розробником звіту.
- *Сторінки доступу до даних* – це розвинене поняття форми, що дозволяє отримувати доступ до даних через глобальну чи локальну мережі, створювати web-сайти з аналізом інформації в базі даних.

- *Макроси* – невеликі підпрограми, що дозволяють реагувати на дії користувача, як правило, запускають зручний користувачеві режим обробки даних.
- *Модулі* – програми, написані на VB, що дозволяють повністю автоматизувати роботу користувача, створити свій інтерфейс, проводити аналіз інформації.

База даних

- Розробка бази даних починається зі створення файлу, в якому зберігатимуться всі об'єкти поточної бази даних. Такий файл має розширення *.mdb. На початку роботи вказується ім'я бази даних та повне ім'я файлу.

Таблиці

- Для створення таблиці в MS Access використовується конструктор (вказується структура таблиці, її поля, типи даних тощо); є можливість копіювання таблиці з поточної або іншої бази даних та імпорту таблиці з іншого середовища (наприклад, з MS Excel).
- При використанні конструктора вказується назва поля, тип даних цього поля і опис поля.
- При створенні нових полів таблиці можуть використовуватися такі типи даних, як текстовий, мемо, числовий, дата/час, лічильник, логічний, об'єктний, гіперпосилання, підстановка (використовується для вибору значень зі списку, іншої таблиці чи запиту).
- Внизу вікна можна встановити інші параметри для кожного з полів.
- Ключове поле таблиці полегшує пошук та сортування елементів у базі даних. Наявність ключового поля не є обов'язковою умовою, проте більшість таблиць створюється саме з ним. Значення поля, яке позначене як ключове, має містити унікальні елементи, тому найчастіше ключовим полем вибирають поле з лічильником, адже в такому полі дані не повторюються.
- Такого самого ефекту можна досягти, натиснувши правою кнопки миші на необхідному полі і вказавши опцію Ключове поле.
- При імпортуванні таблиці слід вказати шлях до файлу, з якого здійснюється імпорт даних. Відповідаючи на питання програми, отримуємо нову таблицю. Слід зауважити, що імпорт даних пройде вдало, якщо відповідна база даних в іншому середовищі була добре згрупована.

Запити

- Запит може бути створений за допомогою таких засобів:
  - шляхом використання майстра: при цьому вказуються поля таблиць та запитів, які необхідні для побудови запиту;
  - шляхом використання конструктора: при цьому є можливість змінювати структуру запиту, додавати розрахункові поля;
  - шляхом написання програми на SQL;
  - шляхом комбінації перерахованих способів.

- Найчастіше для створення запиту використовують конструктор.
- На першому етапі вказують ті таблиці та запити, поля яких будуть використані у створюваному запиті. На другому етапі з виділених таблиць та запитів створюють нові поля вихідного запиту. Для додавання розрахункового поля необхідно вказати в режимі конструктора такі відомості:
  - Назва_поля : Формула
  - Для побудови формули можна використовувати майстер формул.
  - Для всіх полів можна вказати:
  - видимість поля (прапорець у рядку Виведення на екран);
  - сортування (за зростанням чи спаданням);
  - умова відбору (одна чи декілька умов, які відбирають лише потрібні записи);
  - групування (наприклад, знаходження максимального, мінімального, середнього значення, числа записів, дисперсії, першого елемента, останнього елемента, суми для групи елементів).
- Для вказування умови відбору можна використовувати один чи декілька рядків Умови відбору. Якщо умови записані в одному рядку, то для виводу в запит вони всі мають одночасно виконуватися. Якщо умови записуються в різних рядках, то відбираються записи, для яких виконується хоча б одна з умов.

Форми

- Форми у MS Access використовуються для введення даних до таблиць або реакції на дії користувача. Як правило, форма зв'язується з таблицею чи запитом для її правильного відображення. Форма може мати розрахункові поля, а також програми обробки подій.
- Для створення форм можна використовувати:
  - конструктор, який дозволяє вручну змінювати будь-які параметри форми;
  - майстер форм, що полегшує створення стандартних форм;
  - майстер зведених діаграм та таблиць.
- Для швидкого створення форми варто використовувати саме майстер форм, адже в такому випадку всі об'єкти будуть розташовані автоматично.

Звіти

- Звіти створюються для агрегування інформації та виведення її для друку. Вони, як правило, будуються за допомогою майстра звітів, після чого редагуються в конструкторі. При цьому можуть бути вказані:
  - рівень групування записів (починаючи з найважливішого);
  - порядок сортування (до 4 різних полів);
  - вигляд та дизайн звіту.
- При роботі з конструктором можна вказати також колонтитули для звіту, нумерацію сторінок тощо. Кожен об'єкт звіту має свої властивості, які

можна змінити. Взагалі, структуру звіту можна форматувати так само, як і форму, вносячи свої елементи управління.

Сторінки доступу до даних

- Створення сторінки доступу здійснюється аналогічно побудові звіту або форми. Єдиною принциповою відмінністю є можливість працювати зі сторінками в Інтернеті.

Макроси та модулі

- Макроси – це набір команд, виконуваних MS Access у вибраному користувачем порядку з умовою або безумовно, що запускаються на виконання, як правило, комбінацією клавіш.
- При розробці в правому стовпці вибирається одна чи декілька команд, які будуть послідовно виконуватися при запуску макросу.
- Наразі макроси залишаються потужним інструментом роботи з об'єктами MS Access, проте для написання великих програм використовуються модулі, написані за допомогою мови Visual Basic. Вони є найбільш сучасним методом написання програм обробки баз даних. Кожен з модулів містить одну чи декілька пов'язаних між собою процедур чи функцій.

Мова SQL

Оператори маніпуляції даними

- За допомогою запитів здійснюється відбір і сортування інформації, вони використовуються для побудови звітів та форм. Запити MS Access реалізуються через інструкції мови структурованих запитів Structured Query Language (SQL), які можна змінювати в процесі роботи.
- Формат запису операторів SQL вільний. Можна писати усе підряд в одному рядку, один оператор на декількох рядках, слова операторів можна розділяти довільною кількістю пробілів і коментарів. Закінчення операторів визначається за контекстом. Компілятору мови байдуже, великими або маленькими літерами пишуться оператори.
- Як і в інших мовах, весь набір ключових слів мови SQL зарезервований, їх не можна використовувати для інших цілей (для імен об'єктів і змінних SQL).
- При побудові запиту у вікні конструктора MS Access працює у фоновому режимі, записуючи еквівалентні інструкції SQL. Для перегляду програми SQL необхідно вибрати з контекстного меню відповідного запиту пункт Режим SQL:
- У випадку написання складної команди іноді додаються коментарі, які позначаються знаками фігурних дужок "{" та "}", або знаком " – –" (два знаки мінус) до кінця рядка.
- Слід враховувати обмеження щодо довжини назв змінних: ім'я бази даних має містити не більше 10 символів, імена інших об'єктів SQL (таблиць, стовпців, псевдотаблиць, синонімів) – не більше 18 символів.
- Класичний SQL містить такі групи операторів:
  - оператори маніпуляції даними: INSERT, DELETE, SELECT, UPDATE;

- оператори опису даних: CREATE, DROP, ALTER.

- Група операторів маніпуляції даними призначена для маніпулювання даними в таблицях. До неї входять оператори:
  - вибору (SELECT) рядків із таблиці (або таблиць);
  - видалення (DELETE) рядків у таблиці;
  - вставки (INSERT) рядків;
  - зміни (UPDATE) значень в існуючих у таблиці рядках.

Оператор SELECT

- Найчастіше оператором ПК використовується оператор SELECT, в якому вказується, які саме поля потрібно вивести. Таблиця, з якої виводяться дані, вказується після службового слова FROM.
- Умова вибору в операторі SELECT вказується за службовим словом WHERE.
- SQL дозволяє вибирати з усіх необхідних записів лише деяку частину за допомогою додаткових параметрів оператора SELECT.
- SQL дозволяє створювати нові поля, які комбінують інформацію з інших полів.
- Функція ALL вибирає всі записи, що задовольняють умову в інструкції SQL. Цей аргумент використовується, якщо не вказано інших параметрів
- Функція DISTINCT вибирає тільки ті записи, всі значення яких у полях, перерахованих в інструкції SELECT, є унікальними. Комбінація всіх обраних полів має бути унікальною.
- Функція DISTINC TROW пропускає дубльовані записи, в яких співпадають не одне обране поле, а всі поля.
- Функція TOP n повертає визначене число записів із початку до кінця впорядкованого списку. Замість n підставляється необхідне число записів.
- Функція TOP n PERCENT повертає визначений відсоток записів з початку до кінця впорядкованого списку. Замість n підставляється необхідний відсоток записів.

Оператор DELETE

- Оператор DELETE видаляє записи, які задовольняють певну умову.

Оператор INSERT

- Оператор INSERT вставляє до таблиці один або декілька рядків.
- Якщо використовувати складений вираз, то оператор INSERT INTO може вставити цілий набір, обраних під запитом SELECT з іншої таблиці записів.

Оператор UPDATE

- Оператор UPDATE змінює значення полів у рядках, що задовольняють певну умову.

Речення WHERE

- Речення WHERE може міститися в будь-якому з операторів DELETE, UPDATE, SELECT, коли потрібно задати умови на записи, які слід

опрацювати (відповідно, знищити, змінити або вибрати). У реченні WHERE пишеться логічна умова, що отримується за допомогою логічних операторів AND, OR і NOT.

- Можна з'ясувати, чи підходить символьний рядок під визначений шаблон. Для цього використовується операція порівняння за шаблоном – LIKE: символьний вираз LIKE "шаблон".
- В операторі LIKE використовуються тільки два спеціальних символи:
  - * – заміщує довільну кількість символів;
  - ? – заміщує тільки один символ.
- Умови з підзапитом застосовуються для:
  - порівняння виразу з результатом іншого SELECT оператора;
  - визначення належності виразу результатам іншого SELECT оператора;
  - з'ясування порожності множини іншого SELECT оператора.

Спеціальні операції

- Для сортування записів передбачений параметр ORDER BY, який визначає поля, за якими слід провести сортування.
- Функція AVG(стовпець) обчислює середнє значення в стовпці.
- Функція COUNT(*) обчислює кількість записів, що задовольняють умову.
- Функція FIRST(стовпець) відображає перший рядок заданого стовпця.
- Функція LAST(стовпець) відображає останній рядок заданого стовпця.
- Функція MAX(стовпець) обчислює максимальне значення в стовпці.
- Функція MIN(стовпець) обчислює мінімальне значення в стовпці.
- Функція STDEV(стовпець) обчислює середньоквадратичне відхилення в стовпці.
- Функція SUM(стовпець) обчислює суму всіх значень в стовпці.
- Функція VAR(стовпець) обчислює дисперсію в стовпці.

Оператори опису даних

- Оператори опису даних призначені для створення опису, зміни опису і знищення об'єктів бази даних. В SQL різняться такі види об'єктів:
  - база даних (database);
  - таблиця (table);
  - стовпець (column);
  - індекс (index);
  - представлення (view);
  - синонім (synonym).

Створення об'єктів

- Кожний об'єкт має власне ім'я – ідентифікатор, а також власника – користувача, що його створив. Якщо в якомусь запиті або операторі SQL згадуються два стовпці з однаковими назвами, то їх потрібно уточнювати повними іменами таблиць, що їх містять. Перед ім'ям будь-якого об'єкта

можна (а іноді й необхідно) зазначити ім'я його власника (owner-name) – вхідне ім'я користувача, який створив цей об'єкт.

- Для створення нової бази даних використовують оператор CREATE DATABASE.
- Оператор DATABASE відкриває нову базу, закриваючи при цьому доступ до об'єктів поточної бази. Оператор CLOSE DATABASE закриває поточну базу даних.
- Слід зауважити, що у будь-який момент часу доступ відкритий до об'єктів тільки однієї – поточної (current) – бази даних. Саме через цю причину використовувати три наведені вище оператори можна лише для об'єктів, які не належать поточній базі даних. Іншими словами, цей оператор має викликатися не з MS Access.
- Створення таблиці здійснюється за допомогою оператора CREATE TABLE. Аргументами цього оператора є назви необхідних полів з указуванням їх типів.

Типи даних

- Мова SQL підтримує багато різних варіантів типів змінних та їх підтипів : текстовий (CHAR або TEXT), числовий (BYTE, INT, LONG, FLOAT, REAL), часовий (DATE, TIME, DATETIME), грошовий (MONEY), лічильник (COUNTER), логічний (LOGICAL), поле MEMO, об'єкт (IMAGE).

Операції з таблицями

- У вже існуючій таблиці можна поміняти тип стовпця, додати новий, видалити старий. Для цього використовується оператор ALTER TABLE. Зміна структури таблиці призводить до фізичного перетворення даних у ній. Якщо змінений тип стовпця, то дані в ньому перетворяться до нового типу; якщо це неможливо здійснити, то оператор ALTER видає помилку, а таблиця залишається у незмінному стані.
- Для видалення об'єктів використовується оператор DROP.

Закріплення матеріалу

1. Описати основні інструменти створення та підтримки баз даних в MS Access.
2. Які оператори мови SQL використовуються для визначення даних?
3. Які оператори мови SQL використовуються для маніпулювання даними?
4. Описати призначення оператора CREATE.
5. Описати призначення оператора SELECT.
6. Описати призначення оператора DROP.
7. Описати призначення оператора ALTER.
8. Описати призначення спеціальних операцій мови SQL. Навести приклади.

РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА. ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Лекція 11

ПОНЯТТЯ І ФУНКЦІЇ ЛОКАЛЬНИХ КОМП'ЮТЕРНИХ МЕРЕЖ. ПРИНЦИПИ ПОБУДОВИ І ВИКОРИСТАННЯ

Мета лекції: розгляд основних понять та принципів побудови локальних комп'ютерних мереж.

Зміст лекції

1. Місце та роль локальних мереж. Історія комп'ютерного зв'язку.
2. Визначення локальної мережі.
3. Стандарт OSI (Open System Interconnection).
4. Топологія локальних мереж.

Місце та роль локальних мереж. Історія комп'ютерного зв'язку

- *Комп'ютерна мережа* – це система розподіленої обробки інформації між комп'ютерами за допомогою засобів зв'язку.
- Передача інформації між комп'ютерами відбувається за допомогою електричних сигналів, які бувають цифровими й аналоговими. Для перетворення даних із цифрового виду в аналоговий використовуються модеми, які двійковий нуль перетворюють у сигнал низької частоти, а одиницю – високої частоти. Існують локальні (Local Area Network) і глобальні мережі (Wide Area Network).

Етапи розвитку комп'ютерного зв'язку

- 1. Підключення терміналів («інтелектуальних дисплеїв») до центрального комп'ютера (mainframe).
- 2. Об'єднання в мережу перших мікрокомп'ютерів.
- 3. Об'єднання в мережу персональних комп'ютерів.
- 4. Використання локальної мережі для організації спільної роботи комп'ютерів.

Визначення локальної мережі

- «*Локальна мережа*» (ЛОМ) або «*локальна обчислювальна мережа*» (LAN, Local Area Network) – система передачі даних, що дозволяє поєднувати багато ПК за допомогою спеціально прокладених, високоякісних і добре захищених від завад ліній зв'язку та забезпечує прозорий зв'язок, високу швидкість передачі інформації, низький рівень помилок передачі, високу інтенсивність обміну.

Переваги і функції ЛОМ

- *Поділ ресурсів*. Дозволяє заощадливо використати ресурси.
- *Поділ даних*. Надає можливість доступу і керування базами даних з периферійних робочих місць, що потребують інформації.
- *Поділ програмних засобів*. Надає можливість одночасного використання централізованих, раніше встановлених програмних засобів.
- *Поділ ресурсів процесора*. Надає можливість використання обчислювальних потужностей спеціальних процесорів для обробки даних іншими засобами, що входять у мережу.
- *Багатокористувальницький режим*. Сприяє одночасному використанню централізованих прикладних програмних засобів, раніше встановлених і керуємих.

Недоліки ЛОМ

- Додаткові матеріальні витрати на мережеве устаткування, програмне забезпечення, прокладку з'єднувальних кабелів і навчання персоналу.
- Введення посади адміністратора мережі.
- Обмеження можливості переміщення комп'ютерів, підключених до мережі.
- Поширення комп'ютерних вірусів.
- Підвищення небезпеки несанкціонованого доступу до інформації з метою її крадіжки або знищення.

Терміни

- *Абонент* (вузол, хост, станція) – це пристрій, підключений до мережі, що активно бере участь в інформаційному обміні.
- *Сервер* – це абонент (вузол) мережі, що надає свої ресурси іншим абонентам, але сам не використовує їх ресурси.
- *Виділений* (dedicated) *сервер* – це сервер, що займається тільки мережевими завданнями.
- *Невиділений сервер* може крім обслуговування мережі виконувати й інші завдання.
- *Клієнт* – абонент мережі, що тільки використовує мережеві ресурси, але сам свої ресурси в мережу не віддає.

Стандарт OSI (Open System Interconnection)

- *Стандарт OSI* (Open System Interconnect) – «Еталонна модель взаємодії відкритих систем» – основна архітектурна модель для систем передачі повідомлень.

Опис рівнів еталонної моделі OSI

- ***Прикладний рівень (рівень 7).*** Забезпечує послугами прикладні процеси, що лежать за межами масштабу моделі OSI, а саме: ідентифікує і встановлює наявність передбачуваних партнерів для зв'язку, синхронізує спільно працюючі прикладні процеси, встановлює й погоджує процедури усунення помилок і керування цілісністю інформації, визначає, чи є в наявності достатньо ресурсів для передбачуваного зв'язку.

- *Протокол передачі даних* – набір правил і процедур, що регулюють порядок здійснення зв'язку.
- *Синхронізація* – механізм розпізнавання початку блоку даних і його кінця.
- *Ініціалізація* – встановлення з'єднання між взаємодіючими партнерами.
- *Блокування* – розбивка переданої інформації на блоки даних строго певної максимальної довжини.
- *Адресація* – забезпечення ідентифікації різного використовуваного устаткування даних, що обмінюється один з одним інформацією під час взаємодії.
- *Виявлення помилок* – установка бітів парності й обчислення контрольних бітів.
- *Нумерація блоків* – установлення помилково переданої інформації та інформації, що втрачено.
- *Управління потоками даних* – для розподілу та синхронізації інформаційних потоків.
- *Методи відновлення* – повернення до певного положення для повторної передачі інформації після переривання процесу.
- *Дозвіл доступу* – розподіл, контроль і керування обмеженнями доступу до даних.
- ***Рівень подання даних (рівень 6).*** Відповідає за те, щоб інформація, що посилається із прикладного рівня однієї системи, була читабельною для іншої системи прикладного рівня.
- ***Сеансовий рівень (рівень 5).*** Встановлює, управляє і завершує сеанси взаємодії між прикладними завданнями, надає засоби для відправлення інформації, класи послуг і повідомлення у виняткових ситуаціях про проблеми сеансового рівня, рівня подання даних і прикладного рівня.
- ***Транспортний рівень (рівень 4)***. Забезпечує механізми для установки, підтримки та впорядкованого завершення дії каналів, систем виявлення і усунення несправностей транспортування та керування інформаційним потоком.
- ***Мережевий рівень (рівень 3)***. Забезпечує можливість з'єднання і вибір маршруту між двома кінцевими системами.
- ***Канальний рівень (рівень 2).*** Забезпечує надійний транзит даних через фізичний канал, вирішує питання фізичної адресації, топології мережі, лінійної дисципліни, повідомлення про помилки, упорядкованої доставки блоків даних і керування потоком інформації.
- ***Фізичний рівень (рівень 1)***. Визначає електротехнічні, механічні, процедурні і функціональні характеристики встановлення, підтримки та роз'єднання фізичного каналу між кінцевими системами.

Топологія локальних мереж

- *Топологія* – це фізичне розташування комп'ютерів мережі по відношенню один до одного та спосіб з'єднання їх лініями зв'язку.
- *Шина* (bus) – всі комп'ютери паралельно підключаються до однієї лінії зв'язку. Інформація від кожного комп'ютера одночасно передається всім іншим комп'ютерам.
- *Зірка* (star) – до одного центрального комп'ютера приєднуються інші периферійні комп'ютери, причому кожний з них використовує окрему лінію зв'язку. Інформація від периферійного комп'ютера передається тільки центральному комп'ютеру, від центрального – одному або декільком периферійним.
- *Кільце* (ring) – комп'ютери послідовно об'єднані в кільце. Передача інформації в кільці завжди здійснюється тільки в одному напрямку. Кожний з комп'ютерів передає інформацію тільки одному комп'ютеру, що стоїть в ланцюжку за ним, а одержує інформацію тільки від попереднього в ланцюжку комп'ютера.
- Окрім базових топологій розрізняють мережеву топологію дерево (tree), зірково-шинну, зірково-кільцеву мережеву (mesh).
- Фактори, що впливають на фізичну працездатність мережі і безпосередньо пов'язані з поняттям топологія:
  - Справність комп'ютерів (абонентів), підключених до мережі.
  - Справність мережевого устаткування.
  - Цілісність кабелю мережі.
  - Обмеження довжини кабелю.
- Види топологій:
  - Фізична топологія (географічна схема розташування комп'ютерів і прокладки кабелів).
  - Логічна топологія (структура зв'язків, характер поширення сигналів мережею).
  - Топологія керування обміном (принцип і послідовність передачі права на захват мережі між окремими комп'ютерами).
  - Інформаційна топологія (напрямок потоків інформації, переданої по мережі).

Закріплення матеріалу

1. Надати визначення локальної мережі. Вказати переваги, функції та недоліки ЛОМ.
2. Пояснити поняття «Еталонна модель взаємодії відкритих систем».
3. Які базові топології вам відомі? Навести їх характеристики.
4. Які види топологій вам відомі?

РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА. ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Лекція 12
Апаратні мережеві засоби

Мета лекції: розгляд апаратних мережевих засобів та мережевих технологій.

Зміст лекції
1. Кабелі на основі витих пар.
2. Коаксіальні кабелі.
3. Оптоволоконні кабелі.
4. Безкабельні канали зв'язку.
5. Мережеві пристрої.
6. Мережеві технології.

Кабелі на основі витих пар

- *Кабель на основі витих пар* – це декілька пар скручених парами ізольованих мідних дротів у єдиній діелектричній (пластиковій) оболонці. Зазвичай в кабель входить дві або чотири виті пари. Розрізняють неекрановані та екрановані виті пари.
- Неекрановані виті пари характеризуються слабкою захищеністю від зовнішніх електромагнітних перешкод та від підслуховування.
- У випадку екранованої витої пари, кожна з витих пар поміщається в металеве обплетення-екран.
- Основні преваги неекранованих витих пар: простота монтажу роз'ємів на кінцях кабелю; простота ремонту будь-яких ушкоджень. Недоліки: згасання сигналу; низька завадозахищеність; короткі лінії зв'язку.
- Розрізняють 7 категорій кабелів на основі неекранованої витої пари.
- Найчастіше кабелі на основі витих пар використовуються для передачі даних в одному напрямку (точка-точка), тобто в топологіях типу зірка або кільце.

Коаксіальні кабелі

- *Коаксіальний кабель* – це електричний кабель, що складається із центрального мідного дроту і металевого обплетення (екрана), розділених між собою шаром діелектрика (внутрішньої ізоляції) і розміщених у загальній зовнішній оболонці.
- Характеристики коаксіального кабелю: висока завадозахищеність; широка смуга пропускання; велика припустима відстань передачі; складне механічне підключення при несанкціонованому прослуховуванні мережі; мало електромагнітних випромінювань назовні; складні монтаж і ремонт; висока вартість.

- Основне застосування коаксіальний кабель знаходить у мережах з топологією типу шина.

Оптоволоконні кабелі

- *Оптоволоконний* (волоконно-оптичний) *кабель* – це кабель, інформація з якого передається не електричним сигналом, а світловим. Головний його елемент – це прозоре скловолокно, по якому світло проходить на величезні відстані (до десятків кілометрів) з незначним послабленням.
- Характеристики оптоволоконного кабелю: висока завадозахищеність; висока секретність інформації, що передається; смуга пропускання 1000 ГГц; доступна ціна; величина згасання сигналу в оптоволоконних кабелях від 5 до 20 дБ/км; при зростанні частоти переданого сигналу згасання збільшується дуже мало.
- Недоліки: висока складність монтажу; використання спеціальних оптичних приймачів і передавачів; допускає розгалуження сигналів; менш міцний і гнучкий, ніж електричний; чутливий до іонізуючих випромінювань.
- Застосовують оптоволоконний кабель тільки в мережах з топологією зірка і кільце.

Безкабельні канали зв’язку

- Радіоканал використовує передачу інформації по радіохвилях, тому теоретично він може забезпечити зв'язок на багато десятків, сотні і навіть тисячі кілометрів. Швидкість передачі досягає десятків мегабіт за секунду. Головними недоліками радіоканалу є його поганий захист від прослуховування та слабка завадозахищеність.
- Популярна технологія Wi-Fi (Wireless Fidelity) дозволяє організовувати зв’язок між невеликою кількістю комп’ютерів за допомогою концентратора (він має назву точка доступу, Access Point, AP), або декількох концентраторів, якщо кількість комп’ютерів значна. Крім того, ця технологія дає можливість зв’язувати локальні мережі за допомогою потужних безпровідних мостів.
- Радіоканал широко застосовується в глобальних мережах як для наземного, так і для супутникового зв'язку.
- Інфрачервоний канал також не вимагає з’єднувальних проводів, тому що використовує для зв'язку інфрачервоне випромінювання. Головна його перевага в порівнянні з радіоканалом – нечутливість до електромагнітних завад.
- Інфрачервоні канали поділяться на дві групи: прямої видимості; на розсіяному випромінюванні.
- Всі бездротові канали зв'язку підходять для топології типу шина.

Мережеві пристрої

Мережева карта

- *Мережева карта*, мережевий контролер, мережевий адаптер, Ethernet-адаптер, NIC (англ. Network interface controller) – периферійний пристрій, що дозволяє комп'ютеру взаємодіяти з іншими пристроями мережі.

- За конструктивною реалізацією розрізняють: внутрішні; зовнішні, вбудовані в материнську плату.
- Мережевий адаптер разом зі своїм драйвером реалізує другий, канальний рівень моделі відкритих систем в кінцевому вузлі мережі – комп'ютері.

Хаб, концентратор

- *Мережевий концентратор* або хаб (hub – центр діяльності) – мережевий пристрій, призначений для об'єднання декількох пристроїв Ethernet в загальний сегмент мережі.
- Концентратор працює на фізичному рівні мережевої моделі OSI, повторює сигнал, що приходить на один порт, на всі активні порти. В разі надходження сигналу на два і більше порти одночасно виникає колізія, і кадри даних, які передаються, втрачаються. Концентратори завжди працюють в режимі напівдуплексу, всі підключені пристрої Ethernet розділяють між собою смугу доступу, що надається.
- Характеристики мережевих концентраторів: кількість портів; швидкість передачі даних; тип мережевого носія.

Мережевий комутатор, світч

- *Мережевий комутатор* або світч (switch – перемикач) – пристрій, призначений для з'єднання декількох вузлів комп'ютерної мережі в межах одного сегменту. Комутатор передає дані лише безпосередньо одержувачеві, що підвищує продуктивність і безпеку мережі.
- Комутатор працює на канальному рівні моделі OSI і об'єднує вузли однієї мережі за їх MAC-адресам.
- Комутатор зберігає в пам'яті таблицю, в якій вказується відповідність MAC-адреси вузла порту комутатора. При включенні комутатора ця таблиця порожня і він працює в режимі навчання. У цьому режимі дані, що надходять на будь-який порт передаються на всі інші порти комутатора. При цьому комутатор аналізує кадри і, визначивши MAC-адресу хоста-відправника, заносить її в таблицю. Згодом, якщо на один з портів комутатора поступить кадр, призначений для хосту, MAC-адреса якого вже є в таблиці, то цей кадр буде переданий лише через порт, вказаний в таблиці. Якщо MAC-адреса хосту-отримувача ще не відома, то кадр буде продубльований на всі інтерфейси. З часом комутатор будує повну таблицю для всіх своїх портів, і в результаті трафік локалізується.
- Режими комутації: з проміжним зберіганням; наскрізний (cut-through); безфрагментний (fragment-free) або гібридний.
- Різновиди комутаторів: керовані; некеровані.

Маршрутизатор

- *Маршрутизатор* – мережевий пристрій, що приймає рішення про пересилку пакетів мережевого рівня (рівень 3 моделі OSI) між різними сегментами мережі на підставі інформації про топологію мережі і певних правил.

- Маршрутизатор використовує адресу одержувача, вказану в пакетах даних, і визначає за таблицею маршрутизації шлях, за яким слід передати дані. Якщо в таблиці маршрутизації для адреси немає описаного маршруту, пакет відкидається.
- Таблиця маршрутизації містить інформацію, на основі якої маршрутизатор приймає рішення про подальшу пересилку пакетів. Таблиця складається з деякого числа записів – маршрутів, в кожному з яких міститься адреса мережі одержувача, адреса наступного вузла, якому слід передавати пакети, і вага запису, – метрика. Метрики записів в таблиці грають роль в обчисленні найкоротших маршрутів до різних одержувачів.
- Способи складання таблиці маршрутизації: статична маршрутизація; динамічна маршрутизація.
- Маршрутизатори застосовують для: зменшення завантаження мережі; об'єднання мереж різних типів; забезпечення доступу з локальної мережі в глобальну мережу Інтернет.

Мережеві технології

Мережі Ethernet і Fast Ethernet

- З'явилася в 1972 році (розробником виступила відома фірма Xerox), в 1980 році її підтримали такі найбільші компанії, як DEC і Intel. Стала міжнародним стандартом IEEE 802.3. Визначає множинний доступ до моноканалу типу шина з виявленням конфліктів і контролем передачі.

Мережа Token-Ring

- Мережа Token-Ring запропонована компанією IBM в 1985 році, призначалася для об'єднання в мережу всіх типів комп'ютерів, що випускаються IBM. Token-Ring є на даний час міжнародним стандартом IEEE 802.5. Мережа Token-Ring має топологію кільце, хоча зовні вона більше нагадує зірку, оскільки окремі абоненти приєднуються до мережі не безпосередньо, а через спеціальні концентратори або багатостанційні пристрої доступу. Фізично мережа утворює зоряно-кільцеву топологію.

Мережа Arcnet

- Мережа Arcnet (або ARCnet англійською Attached Resource Computer Net, комп'ютерна мережа об’єднаних ресурсів) – це одна із старих мереж, розроблена компанією Datapoint Corporation в 1977 році. Міжнародні стандарти на цю мережу відсутні. Виробництво апаратури Arcnet практично припинене.
- Серед основних переваг мережі Arcnet в порівнянні з Ethernet можна назвати обмежену величину часу доступу, високу надійність зв'язку, простоту діагностики, а також порівняно низьку вартість адаптерів. До найбільш істотних недоліків мережі належать низька швидкість передачі інформації, система адресації і формат пакету.
- Як топологію мережа Arcnet використовує класичну шину, а також пасивну зірку. У зірці застосовуються концентратори (хаби), у шині – адаптери.

Мережа FDDI

- Мережа FDDI (англійською Fiber Distributed Data Interface, оптоволоконний розподілений інтерфейс даних) – це одна з новітніх розробок стандартів локальних мереж.
- Переваги мережі: висока завадозахищеність, максимальна секретність передачі інформації і прекрасна гальванічна розв'язка абонентів.
- Топологія мережі FDDI – це кільце, найбільш відповідна топологія для оптоволоконного кабелю.
- Дана мережа не набула великого поширення, що пов'язане з високою вартістю її апаратури. Основна сфера застосування FDDI – це базові, опорні мережі, об'єднуючі декілька мереж.

Мережа 100VG-AnyLAN

- Мережа 100VG-AnyLAN – це одна з розробок високошвидкісних локальних мереж компаніями Hewlett-Packard і IBM. Відповідає міжнародному стандарту IEEE 802.12. Головними перевагами є велика швидкість обміну, порівняно невисока вартість апаратури, централізований метод управління обміном без конфліктів, а також сумісність на рівні форматів пакетів з мережами Ethernet і Token-Ring. У мережі 100VG-AnyLAN передбачено два режими обміну: напівдуплексний і повнодуплексний.

Надвисокошвидкісні мережі

- Мережа Gigabit Ethernet – це шлях розвитку концепції, закладеної в стандартній мережі Ethernet. Має велику пропускну спроможність. Мережа Gigabit Ethernet знаходить застосування в мережах, об'єднуючих комп'ютери великих підприємств, які розташовуються в декількох будівлях. Вона дозволяє за допомогою відповідних комутаторів, що перетворюють швидкості передачі, забезпечити канали зв'язку з високою пропускною спроможністю між окремими частинами складної мережі або лінії зв'язку комутаторів з надшвидкодіючими серверами.
- Мережа з технологією ATM (Asynchronous Transfer Mode) використовувалась як в локальних, так і в глобальних мережах. Основна ідея – передача цифрових, голосових і мультимедійних даних по одних і тих самих каналах. Принципова відмінність ATM від інших мереж полягає у відмові від звичних пакетів з полями адресації, управління і даних. Вся інформація, що передається, упакована в мікропакети (комірки, cells). Головний недолік мереж з технологією ATM полягає в їх повній несумісності ні з однією з наявних мереж. Специфікації ATM на теперішній час повністю зупинені.

Закріплення матеріалу

1. Охарактеризувати відомі вам різновиди кабелів.
2. Охарактеризувати відомі вам мережеві пристрої.
3. Охарактеризувати відомі вам мережеві технології.

РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА. ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Лекція 13

ПРОГРАМНІ МЕРЕЖЕВІ ЗАСОБИ

Мета лекції: ознайомлення із стандартними мережевими програмними засобами.

Зміст лекції

1. Однорангові мережі.
2. Мережі на основі сервера.
3. Операційні системи.

- Функції верхніх рівнів еталонної моделі OSI виконують мережеві програмні засоби.
- Вибір програмного забезпечення впливає на: допустимий розмір мережі, зручність використання і контролю мережі, режими доступу до ресурсів, продуктивність мережі в різних режимах тощо.
- *Однорангові мережі* – мережі, що складаються з рівноправних (з точки зору доступу до мережі) комп'ютерів.
- *Мережі на основі серверів* – мережі, в яких існують лише виділені (dedicated) сервери, що займаються виключно мережевими функціями.

Однорангові мережі

- Однорангові мережі (Peer-to-Peer Network) і відповідні програмні засоби, як правило, використовуються для об'єднання невеликої кількості комп'ютерів.
- Кожен комп'ютер однорангової мережі може одночасно бути і сервером і клієнтом.
- В одноранговій мережі будь-який сервер може бути невиділеним (non-dedicated), може не лише обслуговувати мережу, але і працювати як автономний комп'ютер. В одноранговій мережі можуть бути і виділені сервери, лише обслуговуючі мережу.
- Переваги однорангових мереж:
  - висока гнучкість;
  - проста установка.
- Недоліки однорангових мереж:
  - слабка система контролю і протоколювання роботи мережі;
  - труднощі з резервним копіюванням розподіленої інформації;

- пошкодження будь-якого комп'ютера-сервера приводить до втрати частини загальної інформації;
- ефективна швидкість передачі інформації часто виявляється недостатньою.

Мережі на основі сервера

- Мережі на основі сервера (Server-based Network) застосовуються в тих випадках, коли в мережу має бути об'єднані багато користувачів. В мережу включається спеціалізований комп'ютер – сервер (виділений), який обслуговує лише мережу і не вирішує жодних інших завдань.
- В мережі може бути декілька серверів, кожен з яких виконує свою задачу.
- Мережі на основі сервера називаються масштабованими.
- Переваги мережі на основі сервера:
  - надійність;
  - висока швидкість обміну.
- Недоліки мережі на основі сервера:
  - громіздкість в разі невеликої кількості комп'ютерів;
  - залежність всіх комп'ютерів-клієнтів від сервера;
  - вища вартість мережі.

Операційні системи

Microsoft Windows Server

- *Windows Server* – найбільш поширена операційна система для серверів. Комп'ютер, на якому встановлена така операційна система, може виконувати ролі файлового серверу, серверу служби веб-додатків, сервера терміналів, поштового сервера, серверу віддаленого доступу, служби DNS (доменних імен), служби каталогів, сервера потоків мультимедіа тощо.
- Windows Server є досить швидкою і надійною ОС. Надійність забезпечує платформа додатків, в яку вбудовані функції сервера додатків, а також інтегроване середовище, що забезпечує доступність і безпеку інформації.
- Для організацій, які працюють з важливими мережевими бізнес-додатками дуже важлива служба кластерів, що дозволяє об'єднати кілька серверів.
- Сучасні ОС Windows Server ретельно перевіряються на наявність слабких місць щодо безпеки і точок помилок. Наприклад, безпеку обчислювального середовища забезпечує загальна мовне середовище виконання.

Операційна система UNIX

- *UNIX* – це багатозадачна, розрахована на багато користувачів система. Вона сама розподіляє процесорний час між завданнями, що дозволяє використовувати сервер для запуску доповнень, а результат отримувати на комп’ютерах-клієнтах.
- ОС UNIX дозволяє створювати комп'ютерні мережі з великою кількістю комп'ютерів (сотень або тисяч) різних типів. Недолік системи полягає в тому, що для кожного сервера потрібне окреме адміністрування.

- В операційній системі UNIX застосовується файлова організація програм і даних. Файлова система має ієрархічну структуру каталогів і файлів. Існує 6 типів файлів, які розрізняють за функціональним призначенням і діям ОС під час виконання різних операцій над файлами: звичайний файл, каталог, спеціальний файл пристрою, іменований канал, зв'язок, сокети.

Операційна система Linux

- *Linux* – вільно розповсюджуване ядро Unix-подібної системи, написане Linus Torvalds за допомогою великого числа добровольців по всій Мережі. Linux має всі властивості сучасної Unix-системи, включаючи справжню багатозадачність, розвинену підсистему управління пам'яттю і мережеву підсистему. Ядро Linux, що поставляється разом з вільно поширюваними прикладними і системними програмами, утворює повнофункціональну універсальну операційну систему.
- Велика частина ядра Linux написана мовою С, завдяки чому система достатньо легко переноситься на різні апаратні архітектури. Розробники Linux намагаються дотримувати стандарти POSIX і Open Group, забезпечуючи тим самим перенесення ПЗ на інші Unix-платформи.
- TCP / IP стек в Linux відповідає усім стандартам і в багатьох можливостях перевершує реалізацію TCP / IP в інших ОС.
- Основною файловою системою Linux є його власна ext2fs. Офіційне ядро містить підтримку більш ніж 20 різних файлових систем.

Родина BSD (FreeBSD, NetBSD, OpenBSD, BSDI)

- Існує чотири клонових BSD-системи: FreeBSD, NetBSD, OpenBSD і BSDI.
- На даний момент FreeBSD є найбільш підтримуваним і популярним клоном серед BSD-систем. Крім базового набору утиліт, який включає в себе практично всі необхідні для роботи в мережі утиліти, а також і деякі основні серверні пакети, для ОС FreeBSD існує «Port Collection» (колекція портів).
- Порт (port) в термінології FreeBSD – набір правил, які зводять іноді дуже складне завдання по встановленню / видаленню ПЗ до мінімального і абсолютно тривіального переліку дій.
- В операційній системі FreeBSD поліпшені підтримка мережі, швидкодія, захист і сумісність.
- FreeBSD є прекрасною основою для створення Internet або Intranet сервера.
- Якість FreeBSD чудово комбінується з дешевими високошвидкісними апаратними засобами.

Закріплення матеріалу

1. Що таке однорангова мережа та мережа на основі сервера?
2. Що таке локальні і розподілені мережеві ресурси?
3. Які програмні засоби забезпечують роботу мережі?
4. Опишіть загальні функції мережевих операційних систем.

РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА. ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР

Лекція 14

ВИКОРИСТАННЯ INTERNET

Мета лекції: розгляд ключових принципів, покладених в основу глобальної телекомунікаційної мережі.

Зміст лекції

1. Ключові питання INTERNET.
2. Служби Інтернет.

Ключові питання Internet

- *Інтернет* (скор. від Interconnected Networks – об'єднані мережі) – глобальна телекомунікаційна мережа інформаційних і обчислювальних ресурсів. Служить фізичною основою для Всесвітнього павутиння.
- Всесвітня комп'ютерна мережа Інтернет разом з персональними комп'ютерами утворює технологічну основу для розвитку міжнародної концепції «Всесвітнього інформаційного суспільства».

Історія

- В 1957 році агентство передових оборонних дослідницьких проєктів США (DARPA) запропонувало розробити комп'ютерну мережу. Розробка такої мережі була доручена Каліфорнійському університету в Лос-Анджелесі, Стенфордському дослідницькому центру, Університету штату Юта і Університету штату Каліфорнія в Санта-Барбарі. Комп'ютерна мережа була названа ARPANET (англ. Advanced Research Projects Agency Network). В 1969 році в рамках проєкту мережа об'єднала чотири вказані наукові установи.
- Перший сервер ARPANET був встановлений 1 вересня 1969 року в Каліфорнійському університеті в Лос-Анджелесі. Комп'ютер Honeywell DP-516 мав 24 Кб оперативної пам'яті.
- До 1971 року була розроблена перша програма для відправки електронної пошти по мережі. Ця програма відразу стала дуже популярна.
- 1 січня 1983 року мережа ARPANET перейшла з протоколу NCP на TCP/IP. Саме у 1983 році термін «Інтернет» закріпився за мережею ARPANET.
- У 1984 році була розроблена система доменних імен (англ. Domain Name System, DNS).

- У 1988 році був розроблений протокол Internet Relay Chat (IRC), завдяки чому в Інтернеті стало можливе спілкування в реальному часі.
- У 1990 році мережа ARPANET припинила своє існування, повністю програвши конкуренцію NSFNet. У тому ж році було зафіксовано перше підключення до Інтернету по телефонній лінії.
- У 1991 році Всесвітнє павутиння стало загальнодоступним в Інтернеті, а в 1993 році з'явився знаменитий веб-браузер.
- У 1995 році маршрутизацією всього трафіку Інтернету почали займатися мережеві провайдери, а не суперкомп'ютери Національного наукового фонду.
- У тому ж 1995 року Всесвітня павутина стала основним постачальником інформації в Інтернеті, обігнавши по трафіку протокол пересилки файлів FTP. З 1996 року Всесвітнє павутиння майже повністю підміняє собою поняття «Інтернет».
- В даний час підключитися до Інтернету можна через супутники зв'язку, радіоканали, кабельне телебачення, телефон, сотовий зв'язок, спеціальні оптико-волоконні лінії або електропроводи.
- Основи технології пакетної комунікації були незалежно один від одного винайдені Полом Бараном і Дональдом Ват Девісом.
- У 1980 році англійський фізик Тім Бернес-Лі всього на півроку влаштувався на роботу в женевську Європейську лабораторію CERN на посаду консультанта по розробці програмного забезпечення. Проявив він себе непогано, але повноправним співробітником лабораторії він став лише в 1984 році, коли і приступив до вирішення проблеми обробки і надання результатів наукових досліджень в режимі реального часу.
- Восени 1990 року співробітники CERN отримали в користування перший «веб-сервер» і «веб-браузер», написані Тімом Бернесом-Лі.

Ключові принципи

- Всі комп'ютери в мережі користуються мережевими протоколами (протоколами управління передачею) з назвою TCP/IP, які підтримуються такими операційними системами: Windows всіх версій, UNIX, Macintosh тощо. Протокол TCP відповідає за організацію зв'язку між двома комп'ютерами, а протокол IP – за маршрутизацію. Кожен комп'ютер, підключений до Internet, має унікальну адресу (IP).
- Адреси Internet поставлені відповідно до назви. За правильним перекладом чисел в назви і навпаки, стежать спеціальні комп'ютери – сервери доменних назв.
- Домен – ланка усередині процесу в .NET, що містить потоки.
- Хост – будь-який пристрій, що надає сервіси формату «клієнт-сервер» в режимі сервера по інтерфейсах і унікально визначений на цих інтерфейсах.

Протоколи

- *Протокол* – це «мова», що використовується комп'ютерами для обміну даними при роботі в мережі, тобто правила передачі даних між вузлами

комп'ютерної мережі. Систему протоколів Інтернет називають «стеком протоколів TCP/IP».

IP-телефонія

- *IP-телефонія* – це технологія, що дозволяє в режимі реального часу вести телефонні розмови з використанням мережі Internet. Телефонні сервери IP-телефонії, з одного боку, пов'язані з телефонними лініями і можуть зв'язатися з будь-яким телефоном світу, з іншого боку, – через Internet і можуть зв'язатися з будь-яким комп'ютером світу. Сервер приймає стандартний телефонний сигнал, перетворює його на цифровий формат (якщо він не цифровий), стискає, розбиває на пакети і відправляє через Internet за адресою з використанням протоколу TCP/IP.

Захист інформації в INTERNET

- В процесі роботи в Internet користувач стикається з такими проблемами безпеки передачі даних: перехоплення інформації, модифікація інформації, підміна авторства інформації.
- *Аутентифікація* – це процес розпізнавання користувача системи і надання йому певних прав і повноважень.
- *Цілісність* – стан даних, при якому вони зберігають свій інформаційний вміст і однозначність інтерпретації в умовах різних дій.
- *Секретність* – попередження несанкціонованого доступу до інформації.
- *Брандмауер* – це система, що дозволяє розділити мережу на декілька частин і реалізувати набір правил, які визначають умови проходження пакетів з однієї частини в іншу.

Служби Інтернет

Глобальна інформаційна служба WORLD WIDE WEB (WWW)

- *WWW* – це служба для пошуку документів в різних базах даних, яка заснована на гіпертекстовій логіці перегляду документів.
- *Гіпертекст* – це багатовимірний текст, який може містити посилання різного напряму або покажчики (адреси) на інші документи і посилання.
- *URL* (uniform resours locator) – уніфікований покажчик ресурсів, який дозволяє браузеру перейти безпосередньо до файлу, який знаходиться на будь-якому сервері мережі.
- *Web-браузери* – спеціальні програми, які створюють команду, пересилають її на сервер і отримують відповідь.
- Обробка даних в http складається з чотирьох етапів: відкриття зв'язку, пересилка повідомлень запиту, пересилки даних відповіді і закриття зв'язку.

Пошукові системи Web-сторінки

- Пошукові системи розділяються на тематичні (класифікатори) і індексні системи пошуку.
- Тематичні системи пропонують користувачам список категорій, в якому web-сторінки упорядковуються за ієрархічною схемою. До тематичних пошукових систем належать: www.mckinley.com, www.yahoo.com та інші.

- Індексні системи виконують пошук сторінок, які містять задані ключові слова. До індексних пошукових систем належать: www.google.com, www.rambler.ru, www.yandex.ru, www.go.com тощо.

Електронна пошта

- *Електронна пошта* – це система, що дозволяє пересилати повідомлення з одного комп'ютера на іншій за допомогою модему або мережевого зв'язку.

Телеконференції (USENET)

- *USENET* – загальнодоступна мережа користувачів, в межах якої люди із загальними інтересами можуть об'єднуватися в дискусійні групи і обмінюватися знаннями, проблемами і тому подібне. Такий спосіб обміну інформацією називається телеконференцією, в кожної з них є своя тематика, група новин.
- Телеконференції розділяються на керовані, в яких розміщення статей робиться спеціальною людиною, і некеровані, в яких розміщення статей виконується автоматично за запитом кого-небудь з користувачів.

TELNET

- *TELNET* – протокол, який дозволяє перетворити комп'ютер на віддалений термінал іншого комп'ютера (у сучасній формі – за допомогою транспорту TCP, стандартно через порт 23). Всі команди і дані, які вводяться з клавіатури, передаються для використання і обробки віддаленому комп'ютеру, а отримані результати виводяться на екран свого комп'ютера. В даний час більшість інформаційних систем, раніше доступних лише за наявності TELNET, стали доступними за допомогою WWW.

Закріплення матеріалу

1. Що таке протокол?
2. Які проблеми передачі даних вам відомі?
3. Якими засобами забезпечується безпека при передачі даних?
4. Назвіть основні типи даних, які застосовуються в WWW.
5. Що таке web-браузери і для чого вони використовуються?
6. Які пошукові системи Web-сторінок вам відомі?
7. Що таке протокол передачі файлів?
8. Який протокол дозволяє перетворити комп'ютер на віддалений термінал іншого комп'ютера?
9. Що таке телеконференція?
10. Що таке IP-телефонія?
11. Що таке аутентифікація?
12. Що таке брандмауер?
13. Що таке URL?
14. Що таке електронна пошта?

РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ

Лекція 15

ДАНІ ТА ЇХ ХАРАКТЕРИСТИКИ

Мета лекції: розгляд різновидів типів даних.

Зміст лекції

1 Дані.
2. Типи даних.

Дані

Примітиви

- *Дані* – це інформація, подана у формалізованому вигляді, прийнятному для обробки автоматичними засобами за можливої участі людини. Розрізняють числові, текстові, графічні, звукові та відеодані.
- *Вхідна інформація* – це дані, які потрапляють в комп’ютер для подальшої обробки.
- *Оперативна (потокова) обробка даних* – це обробка комп'ютером даних при будь-якій швидкості їх надходження.
- *Вихідна інформація* – це дані, що видаються комп'ютером після обробки.
- *Дані в програмуванні* – це величини, які опрацьовуються програмою. Вони поділяються на: константи та змінні; скалярні та структуровані; стандартні та дані користувача.
- *Константи* – це величини, що не змінюють своїх значень в ході виконання програми.
- *Змінні* – об’єкти, що можуть приймати різні значення.
- *Скалярні величини* – це прості значення. Скалярний об’єкт може приймати в будь-який момент виконання програми лише одне значення.
- *Структуровані величини* – це величини, які складаються з декількох значень, тобто, одній величині відповідає деякий набір значень одразу.
- *Операнд* – це дані, що піддаються обробці.

Типи даних

- *Тип даних* – характеристика, яку явно чи неявно надано об'єкту (змінній, функції, полю запису, константі, масиву тощо). Тип даних визначає множину припустимих значень, формат їх збереження, розмір виділеної пам'яті та набір операцій, які можна виконувати над даними. Типи даних поділяються на машинні типи даних (біт, байт, слово, подвійне слово),

прості типи даних (числові – цілі та дійсні типи, логічний (булевий), символьний та байтовий), складні типи даних (масиви, множини, рядки, записи, файли, динамічні змінні, вказівники, лінійні списки (стеки, черги), нелінійні списки (двійкові дерева, несиметричні дерева, тексти, графи), процедурний тип, об'єкти).

- *Складні типи даних* – це типи, які складаються з елементів, що належать до простих типів.
- *Абстрактний тип даних* – це тип даних, який надає для роботи з елементами певного набору цього типу функції, а також можливість створювати елементи цього типу за допомогою спеціальних функцій. Абстрактний тип даних визначає набір функцій, незалежних від конкретної реалізації типу, для оперування його значеннями.
- В програмуванні абстрактні типи даних зазвичай подаються у вигляді інтерфейсів, які приховують відповідні реалізації типів.
- *Векторний тип даних* – це тип даних, який будується на основі простих типів даних. Усі елементи векторного типу даних розміщенні один за одним, у межах створеного об'єкту.
- *Масив* (англ. array) – одна з найпростіших структур даних, сукупність елементів переважно одного типу даних, впорядкованих за індексами, які зазвичай репрезентовані натуральними числами, що визначають положення елемента в масиві. Розрізняють одновимірні (вектор) та багатовимірні (наприклад, двовимірна таблиця) масиви.
- Переваги та недоліки масивів:
  - ефективність операцій;
  - збереження в пам'яті;
  - індекси в масивах;
  - збереження багатовимірних масивів.
- Найпоширеніші способи організації масивів в пам'яті:
  - розташування «рядок за рядком»;
  - розташування «стовпчик за стовпчиком»;
  - масив з масивів.
- *Рядок* (англ. string) або *рядковий тип даних* – це тип нечисловий даних, значеннями якого є довільна послідовність (рядок) символів алфавіту. Кожна змінна такого типу (рядкова змінна) може бути представлена фіксованою кількістю байтів або мати довільну довжину.
- Внутрішнє представлення рядка в пам'яті: деякі мови програмування накладають обмеження на максимальну довжину рядка, але в більшості мов подібні обмеження відсутні. При використанні Unicode кожен символ рядкового типу може вимагати двох або навіть чотирьох байтів для свого представлення.
- Основні проблеми в машинному поданні рядкового типу:
  - істотний розмір;

- розмір змінюється з часом.

- Підходи надання рядків в пам'яті комп'ютера:

  1) Представлення масивом символів

  Переваги:
  - програма в кожен момент часу містить відомості про розмір рядка;
  - рядок може містити будь-які дані;
  - можливість на програмному рівні стежити за виходом за границі рядка при його обробці;
  - можливість швидкого виконання операції виду «взяття N-ого символу з кінця рядка».

  Недоліки:
  - проблеми зі збереженням і обробкою символів довільної довжини;
  - збільшення витрат на збереження рядків;
  - обмеження максимального розміру рядка.

  2) Метод «завершального байту»

  Переваги:
  - відсутність додаткової службової інформації про рядок;
  - можливість надання рядка без створення окремого типу даних;
  - відсутність обмеження на максимальний розмір рядка;
  - економне використання пам'яті;
  - простота отримання суфіксу рядка;
  - простота передачі рядків у функції.

  Недоліки:
  - значний час виконання операцій отримання довжини і конкатенації рядків;
  - відсутність засобів контролю за виходом за границі рядка;
  - неможливість використовувати символ завершального байту в якості елемента рядка.
  - неможливість використовувати деякі кодування з розміром символу в кілька байт.

  3)Представлення у вигляді списку

  Переваги:
  - мова більш «теоретично елегантна» за рахунок дотримання ортогональності в системі типів.

  Недоліки:
  - суттєві втрати швидкодії.

- *Запис* (record) – це структурований тип даних, призначений для обробки даних, що складаються з полів даних різних типів, та збереження їх в оперативній пам'яті. Розрізняють фіксовані (звичайні) та варіантні записи.

- Доступ до певного поля запису здійснюється за допомогою кваліфікуємих (уточнених) ідентифікаторів (складені імена), в яких вказується весь ланцюг імен від ідентифікатора змінного типу «запис» до ідентифікатора потрібного поля.
- Записи з варіантами (варіантними полями) – це записи, в яких можна задавати тип утримуючого визначення декількох варіантів структури.
- *Множина* (англ. set) – це набір однотипних елементів порядкового типу, номер якого у відповідному типі не виходить за границі діапазону 0...255.
- Всяка множинна змінна зберігає не самі значення її елементів, а лише інформацію про те, присутній або відсутній конкретний елемент в заданій множині.
- *Множинні константи* – це константи, які визначаються шляхом переліку в квадратних дужках через кому або з використанням інтервалу всіх елементів множини.
- *Файл* – це структурований тип даних. Використовуючи ідентифікатор змінної файлового типу можна отримати доступ до файлу – сукупності даних необмеженого об'єму.
- *Динамічна змінна* – це змінна в програмі, місце в оперативній пам'яті під яку виділяється під час виконання програми.
- *Вказівник* (англ. pointer) – це змінна, діапазон значень якої складається з адрес комірок пам'яті або спеціального значення – нульової адреси.
- *Лінійний список* – це екземпляр абстрактного типу даних, що формалізує концепцію впорядкованої множини елементів. Лінійний список з однорівневою структурою даних.
- *Нелінійний список* – це багаторівнева структура даних, яка не впорядковує дані послідовно, а є впорядкованою у відсортованому порядку.
- *Процедурний тип даних* – це тип, призначений для збереження посилань на процедури або функції.
- *Об'єкт* – це деяка сутність в цифровому просторі, яка має певні стан і поведінку, властивості (атрибути) і операції над ними (методи).

Закріплення матеріалу

1. Роз’яснити поняття «дані», «константи», «змінні», «скалярні величини», «структуровані величини».
2. Які типи даних вам відомі?
3. Що таке масив, рядок,запис, запис з варіантами, множина, файл.
4. Що таке динамічна змінна та вказівник?
5. За допомогою яких конструкцій програмування можна визначити абстрактний тип даних?
6. Що таке лінійний та нелінійний списки?
7. Що таке об’єкт?

РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ

Лекція 16

МОДЕЛЬ, АЛГОРИТМ, ПРОГРАМА

Мета лекції: розгляд поняття моделі, алгоритму та процедур, які з ними пов’язані; розгляд основних видів структур даних

Зміст лекції

1. Життєвий цикл програмного забезпечення: модель, алгоритм, програма.
2. Поняття та властивості алгоритму.
3. Способи запису алгоритмів.
    3.1. Примітиви.
    3.2. Псевдокоди.
4. Базові структури.

Життєвий цикл програмного забезпечення: модель, алгоритм, програма

- Щоб залучити комп'ютер до дослідження об'єкта, процесу, явища або до «рутинної» обробці інформації, перш за все потрібно:
  - чітко поставити завдання (розробити модель);
  - визначити вихідні дані, форму представлення результатів;
  - створити алгоритм розв’язку задачі;
  - написати програму, яка буде зрозуміла комп'ютеру.
- Етапи життєвого циклу програмного забезпечення:

  1. Аналіз вимог (будується математична модель об'єкта чи явища, яка відображає лише суттєві сторони об'єкта, але не тотожна йому).

  2. Визначення специфікацій (формулювання висновків, що випливають з результатів попереднього етапу; специфікації відображають вимоги до програми і оформлюються у вигляді технічного завдання).

  3. Проєктування (обирається метод розв’язку, складається загальний проєкт програми, виділяються основні частини, їх взаємодія (інтерфейс), уточнюються вхідні та вихідні дані, розробляється загальний алгоритм розв’язку, який поступово деталізується).

  4. Кодування (переклад на мову програмування конструкцій алгоритму; виявлення помилок та неточностей в алгоритмі).

  5. Тестування (налагодження, виявлення, локалізація і виправлення помилок).

6. Супровід (налаштування програми на конкретні цілі, навчання користувачів, усунення дрібних неточностей та аналіз результатів експлуатації програми).

Поняття та властивості алгоритму

- *Алгоритмізація* – найважливіший етап у процесі вирішення завдань на ЕОМ.
- *Алгоритм* – це зрозумілий і точний припис (вказівка) виконавцю зробити певну послідовність дій над вхідною інформацією для вирішення поставленого завдання.
- Властивості алгоритму:

  1) Дискретність. Алгоритм повинен бути розбитий на окремі досить прості дії, причому виконання кожного кроку починається після завершення попереднього.

  2) Визначеність. Однозначність виконання кожного кроку. Алгоритм не повинен допускати довільного трактування виконавцем.

  3) Результативність (виконуваність). Алгоритм повинен надавати можливість отримувати результат за кінцеве число кроків.

  4) Масовість. Це придатність для розв’язання багатьох або навіть всіх задач даного типу при різних вхідних даних.

  5) Інваріантність. Алгоритм повинен бути складений таким чином, щоб він, в ідеальному випадку, міг виконуватися різними виконавцями: різними ЕОМ, в різних середовищах, різними людьми.

Способи запису алгоритмів

Примітиви

- Причини неправильного розуміння подання алгоритмів:

  1) Неоднозначність термінології, що використовується.

  2) Неправильне розуміння алгоритму, викликане недостатньою деталізацією його опису.
- Проблеми сприйняття виникають у тих випадках, коли обрана для представлення алгоритму мова неточно визначена або представлена в описі алгоритму інформація недостатньо деталізована.
- В комп'ютерних науках ці проблеми вирішують шляхом створення чітко визначеного набору складових блоків, з яких можуть конструюватися подання алгоритмів. Такі блоки називаються примітивами (primiteve).
- Набір примітивів разом з набором правил, які визначають яким чином ці примітиви можуть комбінуватися для представлення більш складних ідей, утворюють мову програмування.
- Кожен примітив складається з двох частин: синтаксичної та семантичної. Синтаксис належить до символьного подання примітиву, а семантика – до представленої концепції, тобто до значення примітиву.
- Примітиви можуть надаватись у вигляді словесного запису або у вигляді схеми.

- Словесний запис, орієнтовано на виконавця-людину. Основна перевага такого способу подання – зрозумілість. Недоліки: неоднозначність, надмірність, відсутність наочності зв'язків.

Схеми програм

- *Схеми програм* – це графічне відображення алгоритму згідно затвердженим стандартам, при якому кожна дія записується всередині блоків. Блоки з'єднуються лініями, які вказують послідовність дій. Лінії можуть закінчуватися стрілками, але зазвичай стрілки не ставлять, якщо лінії відображають природню послідовність дій. Природня послідовність: зверху вниз і зліва направо. В іншому випадку стрілки обов'язкові. Кожен блок повинен мати розміри, що дозволяють вписати його в, так зване, основне поле розміром a на b, крім блоків початку/кінця і розриву ліній, які мають менший розмір. Висота основного поля a вибирається як будь-яке значення кратне 10, 15, 20 мм, а довжина b = 2a чи b = 1.5a.
- Перевага подання алгоритмів у вигляді схем програм – наочність. Недолік – велика трудомісткість виконання.

Псевдокоди

- *Псевдокоди* займають проміжне місце між природньою і формальною мовою. Єдиного стандарту на псевдокод не існує. Псевдокод використовує деякі службові слова (вони підкреслюються), але в той же час допускає твердження і природною мовою.

Мови програмування

- Основний виконавець алгоритмів – ЕОМ. Для ЕОМ запис алгоритму повинен бути абсолютно точним, тобто мова для запису алгоритмів повинна бути формалізованою. Така мова називається мовою програмування, а запис на такій мові називається програмою.

Базові структури

- *Структури даних* – це програмна одиниця, яка дозволяє зберігати та обробляти безліч однотипних і / або логічно пов'язаних даних в обчислювальній техніці. Для додавання, пошуку, зміни і видалення даних структура даних надає певний набір функцій, з яких складається інтерфейс.
- Розрізняють:

  1) Лінійні структури даних:
     - список:
       - ✓ масив;
       - ✓ зв’язний список.
     - асоціативний масив;
     - хеш-таблиця;
     - стек;
     - черга.
  2) Граф;
  3) Дерева.

- *Список* – це абстрактний тип даних, що представляє собою впорядкований набір значень, в якому деяке значення може зустрічатися більше одного разу. Екземпляр списку є комп'ютерною реалізацією математичного поняття кінцевої послідовності – кортежу.
- *Елемент списку* – це екземпляр значень, що знаходяться в списку. Якщо значення зустрічається кілька разів, то кожне вважається окремим елементом.
- *Однозв'язний список* – це список, кожен елемент якого містить інформацію, необхідну для знаходження наступного елемента списку (вузла).
- *Двозв'язний список* – це список, який застосовується при необхідності переміщення по списку як в прямому, так і в зворотному напрямку, коли заданому елементу потрібно знайти попередній і наступний відносно нього елемент.

Закріплення матеріалу

1. З яких етапів складається життєвий цикл програмного забезпечення?
2. Що таке алгоритм?
3. Перелічити властивості алгоритму.
4. Що таке дискретність алгоритму?
5. Що таке визначеність алгоритму?
6. Що таке результативність алгоритму?
7. Що таке масовість алгоритму?
8. Що таке інваріантність алгоритму?
9. Які базові види алгоритмів вам відомі?
10. Що таке примітиви? З якою метою вони використовуються?
11. Що таке схеми програм? З якою метою вони використовуються?
12. Які основні елементи схем програм вам відомі? Пояснити їх призначення.
13. Назвати правила побудови схем програм.
14. Що таке псевдокоди? З якою метою вони використовуються?
15. Що таке компілятор? Поясніть схему його роботи.
16. Що таке структури даних?
17. Назвати основні види структур даних.
18. Що таке списки?
19. Які види списків вам відомі та в чому їх відмінності?
20. Що таке елемент списку?

РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ

Лекція 17

ОСНОВИ АНАЛІЗУ АЛГОРИТМІВ

Мета лекції: ознайомлення з параметрами алгоритмів, принципами аналізу алгоритмів та визначенням критеріїв якості програм.

Зміст лекції

1. Параметри алгоритмів.
2. Принципи аналізу алгоритмів.
3. Критерії якості програм.

Параметри алгоритмів

- Розрізняють сім параметрів, які характеризують алгоритм:
  - сукупність можливих вихідних даних;
  - сукупність можливих результатів;
  - сукупність можливих проміжних результатів;
  - правила початку;
  - правило безпосередньої переробки;
  - правило закінчення;
  - правило отримання результату.

Принципи аналізу алгоритмів

- *Аналіз* – це ключ до розуміння алгоритмів в ступені, достатньому для їх ефективного застосування до практичних завдань.
- Якщо алгоритм повинен бути реалізований як частина великої системи, використовуються абстрактні типи даних або аналогічний механізм, який дозволяє змінити алгоритми або деталі реалізації після того, як стане ясно, яка частина системи заслуговує на найпильнішу увагу.
- На самому початку потрібно розуміння характеристик продуктивності кожного алгоритму. В багатьох випадках продуктивність всієї системи залежить від продуктивності декількох базових алгоритмів.
- Програми повинні бути розроблені таким чином, щоб їх зміна була нескладною, щоб їх могли швидко читати і розуміти інші програмісти, щоб вони добре поєднувалися з різними частинами системи, і зберігалася можливість їх перенесення в інші середовища.

- При аналізі кожного алгоритму основну увагу слід приділяти істотним характеристикам продуктивності алгоритму. Головний інтерес викликають найбільш ефективні алгоритми, особливо якщо вони ще й більш прості.
- Для ефективного використання алгоритмів необхідно розуміння характеристик продуктивності алгоритмів. Формування такого розуміння є метою аналізу алгоритмів.
- Кроки в розумінні продуктивності алгоритмів: емпіричний аналіз, математичний аналіз.
- Проблеми при емпіричному аналізі:
  - розробка коректної та повної реалізації;
  - визначення природи вхідних даних і інших чинників, які безпосередньо впливають на проведені експерименти;
  - визначення типу вхідних даних для порівняння алгоритмів.
- Реальні дані дозволяють точно виміряти параметри використовуваної програми.
- Випадкові дані гарантують, що експерименти тестують алгоритм, а не дані.
- Помилкові дані гарантують, що програми можуть сприймати будь-які подані на їх вхід дані.
- Помилки при виборі алгоритму:
  - ігнорування характеристик продуктивності;
  - занадто велика увага до характеристик продуктивності.
- Математичний аналіз алгоритмів виконується з метою:
  - порівняння різних алгоритмів, призначених для вирішення однієї задачі;
  - для приблизної оцінки продуктивності програми в новому середовищі;
  - для встановлення значень параметрів алгоритму.
- Підходи до аналізу алгоритмів, які ґрунтуються на моделюванні наборів даних, які можуть бути подані на вхід алгоритму:
  - припущення, що вхідні дані випадкові, і вивчення середньої продуктивності програми;
  - розгляд самих незручних даних і вивчення найгіршої продуктивності програми.
- Більшість алгоритмів мають головний параметр N, який найбільш сильно впливає на час їх виконання. Параметр N може бути ступенем полінома, розміром файлу при сортуванні або пошуку, кількістю символів в рядку або деякою іншою абстрактною мірою розміру розглянутої задачі: найчастіше він прямо пропорційний обсягу оброблюваного набору даних.
- Коли таких параметрів існує більше одного (наприклад, M і N), часто аналіз зводиться до одного параметру як функції інших або до розгляду одночасно тільки одного параметру (вважаючи інші постійними).

- Метою є вираження вимог програм до ресурсів в залежності від N з використанням математичних формул, які максимально прості і справедливі для великих значень параметрів.
- Алгоритми зазвичай мають час виконання, пропорційний одній з наступних функцій: 1, $logN$, N, $NlogN$, N^2, N^3, 2^N.
- Час виконання певної програми зазвичай дорівнює деякій константі, помноженій на один з цих елементів (головний член) плюс менші складові.
- В алгоритмах і їх аналізі часто бувають потрібні дискретні одиниці, тому часто потрібні спеціальні функції, що перетворюють дійсні числа в цілі:
  - $\lfloor x \rfloor$ – найбільше ціле, менше або рівне x;
  - $\lceil x \rceil$ – найменше ціле, більше або рівне x.
- При аналізі алгоритмів часто виникає дискретизована версія функції натурального логарифма, звана гармонійними числами.
- Математичний запис, що дозволяє відкидати деталі при аналізі алгоритмів, називається *О-нотацією*.
- **Визначення**. Кажуть, що функція $g(N)$ має порядок $O(f(N))$, якщо існують такі постійні c_0 і N_0, що $g(N) < c_0 f(N)$ для всіх $N > N_0$.
- Причини використання О-нотації:
  - обмеження помилки, що виникає при відкиданні малих доданків в математичних формулах;
  - обмеження помилки, що виникає при ігноруванні частин програми, які вносять невеликий вклад в аналізуєму суму;
  - класифікація алгоритмів по верхніх границях їх загального часу виконання.
- Як правило, алгоритм, ефективніший в асимптотичному розумінні, буде більш продуктивним для всіх вхідних даних, за винятком дуже малих. Позначення, що використовуються для опису асимптотичної поведінки часу роботи алгоритму, використовують функції, область визначення яких – множина цілих невід'ємних чисел. Подібні позначення зручні для опису часу роботи $T(N)$ в найгіршому випадку як функції, що визначена лише для цілих чисел, які є розміром вхідних даних.
- У наближених формулах часто використовується символ O, за допомогою якого позначають асимптотичну верхню границю.
- Поліноміальним алгоритмом називається алгоритм, у якого часова складність дорівнює $O(p(N))$, де $p(N)$ – поліном, N – вхідна довжина. Алгоритми, часова складність яких не піддається вказаній оцінці, називають експоненціальними.
- *Декомпозиція* – поділ цілого на частини – це науковий метод, який використовує структуру завдання і дозволяє замінити рішення однієї великої задачі рішенням серії менших завдань, нехай і взаємопов'язаних, але більш простих.
- Правила декомпозиції:

  - кожне розчленовування утворює свій рівень;
  - вихідна система розташовується на нульовому рівні; після її розчленування виходять підсистеми першого рівня; розчленування цих підсистем або деяких з них призводить до появи підсистем другого рівня тощо.
- *Ієрархічна структура декомпозованої системи* – це її спрощене графічне представлення.
- Для аналізу ієрархічної структури можуть застосовувати теорію графів. Це дозволяє перейти від графічної моделі до математичної.
- Система розчленовується тільки за однією, постійною для всіх рівнів, ознакою.
- Ознаки декомпозиції:
  - функціональне призначення частин;
  - конструктивна будова;
  - структурні ознаки;
  - види етапів і процесів;
  - предметні характеристики.
- *Глибина декомпозиції* – це число рівнів ієрархії.
- *Похибка дискретизації* (sampling error) – це похибка, яка виникає при дискретизації неперервної задачі.
- *Похибка округлення* (rounding error) – це похибка, яка виникає завдяки скінченній довжині розрядної сітки комп'ютера, що призводить до наближеного представлення дійсних чисел.
- *Похибка зрізу* (truncation error) – це похибка, що виникає при заміні нескінченних рядів скінченними.
- *Абсолютна похибка* (absolute error) – це похибка, що визначається як модуль різниці між точним A та наближеним a значеннями числа: $\Delta = |a - A|$.
- *Відносна похибка* (relative error) – це похибка, що визначається, як: $\delta = \Delta / |a|$.
- *Приведена похибка* – це відносна похибка, де в знаменнику замість $|a|$ використовується діапазон a від максимального $a_{\max}$ до мінімального $a_{\min}$ значення: $S = \Delta / |a_{\max} - a_{\min}|$.
- *Локальна похибка* – це похибка, що виникає на кожному конкретному кроці обчислювального алгоритму.
- *Глобальна похибка* – це похибка, що включає всі похибки, які виникли як на цьому кроці обчислень, так і на попередніх.
- При оцінці глобальних похибок часто використовуються поняття максимальної похибки, середньої похибки, середньоквадратичної похибки.

- *Швидкодія* – один з головних критеріїв якості обчислювальних алгоритмів – кількість елементарних обчислювальних операцій, яка потрібна для реалізації алгоритму.
- *Ітераційний алгоритм* (iterative algorithm) – це алгоритм, що визначається як багатокроковий, в якому кожний наступний крок виконується на основі попереднього.
- *Стійкість алгоритму* (stability of the algorithm) – здатність виконувати обчислення і отримувати кінцевий результат із заданою точністю при зміні параметрів алгоритму і вхідних даних в деякій області, яка називається областю стійкості.
- *Збіжність* (convergence) – це властивість алгоритму шляхом зміни його параметрів виконувати обчислення з скільки завгодно малою похибкою для заданого класу вхідних даних.
- *Коректність обчислювального методу* (the correctness of computational method) – це властивість безперечного існування розв'язку задачі та забезпечення стійкості обчислювального алгоритму, що реалізує цей метод.

Критерії якості програм

- *Мобільність програмних продуктів* – їх незалежність від технічного комплексу системи обробки даних, операційного середовища, мережевої технології обробки даних, специфіки предметної області.
- *Надійність роботи програмного продукту* визначається безперебійністю і стійкістю в роботі програм, точністю виконання запропонованих функцій обробки, складністю діагностики, що виникає в процесі роботи програм помилок.
- *Ефективність програмного продукту* оцінюється як з позицій прямого призначення – вимог користувача, так і з точки зору витрат обчислювальних ресурсів, необхідних для його експлуатації.
- *Врахування людського фактора* – забезпечення дружнім інтерфейсом, наявність контекстно-залежної підказки або навчальної системи в складі програмного засобу та якісної супровідної документації.
- *Модифікованість програмних продуктів* означає здатність до внесення змін.
- *Комунікативність програмних продуктів* заснована на максимально можливій їх інтеграції з іншими програмами, забезпеченні обміну даними в загальних форматах представлення.

Закріплення матеріалу

1. Які параметри характеризують алгоритм?
2. Які похибки виникають при дискретизації неперервної задачі?
3. Які критерії якості програм вам відомі?

РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ

Лекція 18

ДИНАМІЧНІ МАСИВИ

Мета лекції: розгляд реалізації динамічних масивів; розгляд стеку та черги як структур даних та методів для роботи з ними; розгляд множини як структури даних та операцій роботи з нею.

Зміст лекції

1. Динамічний масив та його реалізація на основі класу ArrayList.
2. Стек, черга, множина.

Динамічний масив та його реалізація на основі класу ArrayList

- *Динамічний масив* – це масив, розмір якого може змінюватися під час виконання програми. Динамічний масив реалізується на основі фіксованого масиву, розмір якого змінюється за допомогою спеціального методу. Динамічні масиви дають можливість більш гнучкої роботи з даними, регулюють розмір масиву у відповідності до необхідних об’ємів.
- Клас ArrayList – це клас, який призначений для підтримки динамічних масивів, що за необхідності можуть збільшуватися або скорочуватися.
- В C# стандартні масиви мають фіксовану довжину, яка не може змінюватися під час виконання програми. Об'єкт класу ArrayList – це масив змінної довжини, елементами якого є об’єктні посилання. Будь-який об'єкт класу ArrayList створюється з деяким початковим розміром. При перевищенні цього розміру колекція автоматично його збільшує. У разі видалення об'єктів масив можна скоротити.
- Клас ArrayList реалізує інтерфейс ICollection, IList, IEnumerable, IClonable.
- Конструктори, визначені в класі ArrayList:
  - public ArrayList();
  - public ArrayList(ICollection c);
  - public ArrayList(int capacity);
- Конструктор ArrayList() призначений для створення порожнього ArrayList-масиву з початковою ємністю, що дорівнює 16-ти елементам.
- Конструктор ArrayList(ICollection c) призначений для побудови масиву, який ініціалізується елементами і ємністю колекції, заданої параметром c.
- Конструктор ArrayList(int capacity) створює список із заданою початковою ємністю.

- *Ємність* або *місткість* – це розмір масиву для збереження елементів.
- При додаванні елементів в ArrayList-масив його ємність автоматично збільшується, причому кожного разу, коли список повинен розширюватися, його ємність подвоюється.
- *Стек* – це динамічна структура даних, що представляє собою колекцію елементів, доступ до яких організований за принципом LIFO (Last In – First Out, «останнім прийшов – першим вийшов»).
- Всі операції в стеку можна проводити тільки з одним елементом, який знаходиться на верхівці стеку та був введений в стек останнім.
- Найчастіше принцип роботи стека розглядається як певна аналогія до стопки тарілок, з якої можна взяти верхню, і на яку можна покласти верхню тарілку.
- Найчастіше стек реалізується на основі двозв'язного списку або на основі звичайного масиву.
- Основні операції зі стеком: додавання елемента (push), видалення елемента (pop) і читання головного елементу (peek).
- При виконанні операції push («заштовхнути елемент») елемент додається в стек та розміщується в його верхівці. Розмір стеку збільшується на одиницю. При перевищенні розміру стека граничної величини, відбувається переповнення стека (stack overflow).
- При виконанні операції pop («виштовхнути елемент») отримується елемент з верхівки стеку. При цьому він видаляється зі стеку і його місце в верхівці стеку займає наступний за ним відповідно до правила LIFO, а розмір стеку зменшується на одиницю. При намаганні «виштовхнути» елемент з вже порожнього стеку, відбувається ситуація "незаповнення" стеку (stack underflow).
- *Черга* – динамічна структура даних, що є колекцією, доступ до елементів якої здійснюється за принципом «перший прийшов – першим вийшов» (FIFO, First In – First Out).
- Основні операції з чергою: додавання елемента (enqueue – поставити в чергу), видалення елемента (dequeue – прибрати з черги).
- *Enqueue* – це операція додавання елемента в «хвіст» черги. При цьому довжина черги збільшується на одиницю. Якщо відбувається намагання додати елемент у вже заповнену чергу, відбувається її переповнення (queue overflow).
- *Dequeue* – це операція, яка повертає елемент з голови та видаляє його з черги, таким чином встановлюючи голову на наступний за видаленим елемент та зменшуючи довжину на одиницю. При намаганні видалити елемент з порожньої черги, виникає ситуація «незаповнення» (queue underflow).
- Черги реалізовані практично у всіх розвинених мовах програмування. В CLI для реалізації черги передбачений клас System.Collections.Queue з методами Enqueue і Dequeue.

- Черга в програмуванні використовується, як і в реальному житті, коли потрібно зробити якісь дії в порядку їх надходження, виконавши їх послідовно. Прикладом може бути організація подій в Windows. Коли користувач здійснює якусь дію в додатку, то в ньому одразу не викликається відповідна процедура (оскільки в цей момент додаток може здійснювати інші дії), виконання дії ставиться в чергу, і тільки коли будуть оброблені повідомлення, що прийшли раніше, додаток виконає необхідну дію.
- *Двозв'язна черга* (дек від англ. deque – double ended queue; двостороння черга, двозв'язний список, черга з двома кінцями) – динамічна структура даних, в якій елементи можна додавати і видаляти як на початок, так і в кінець, одночасно реалізує принципи FIFO і LIFO.
- *Множина* – динамічна структура даних в інформатиці, є реалізацією математичного об'єкта множина.
- Множина дозволяє зберігати необмежену кількість значень певного типу без певного порядку. Повторне додавання значення в множину, як правило, неприпустимо. Для цієї структури даних в мовах програмування зазвичай передбачені стандартні операції над множинами.
- *Об'єднання множин* (сума або з'єднання) в теорії множин – множина, що містить в собі всі елементи вихідних множин. Об'єднання двох множин A і B зазвичай позначається $A \cup B$.
- *Різниця двох множин* – це теоретико-множинна операція, результатом якої є множина, в яку входять всі елементи першої множини, які не входять до другої множини. Зазвичай різниця множин A і B позначається як A\B (іноді A-B або A~B).
- *Перетин множин* – це множина, якій належать ті і тільки ті елементи, які одночасно належать всім даними множинам.
- *Симетрична різниця множин* – операція над двома множинами, результатом якої є нова множина, що включає всі елементи вихідних множин, які не належать одночасно обом вихідним множинам. Іншими словами, якщо є дві множини A і B, то їх симетрична різниця є об'єднання елементів множини A, які не входять до множини B, з елементами множини B, які не входять до множини A. Для позначення симетричної різниці множин A і B використовується позначення $A \Delta B$.

Закріплення матеріалу

1. Що таке динамічний масив?
2. Що таке стек та за яким принципом він організований?
3. Що таке черга та за яким принципом вона організована?
4. Що таке дек та за яким принципом він організований?
5. Що таке множина? Які основні операції для роботи з множинами вам відомі?

РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ

Лекція 19

БАЗОВІ АЛГОРИТМИ СОРТУВАННЯ

Мета лекції: розгляд базових алгоритмів сортування.

Зміст лекції

1. Метод лінійного сортування.
2. Сортування методом «бульбашки».
3. Алгоритм сортування методом вставки.

- *Сортування* – один з процесів перетворення масивів, які використовуються найчастіше.
- Масив є впорядкованим за зростанням його елементів, якщо для деякого масиву *R* встановлені наступні закономірності: *R[1]< R[2]*, *R[2]< R[3]*, *R[3]< R[4]*, *. . .*, *R[n-1]< R[n]*. Звідси випливає правило: якщо кожний елемент масиву менший за безпосередньо наступний за ним елемент, то це є індикатором того, що всі елементи масиву повністю відсортовані.

Метод лінійного сортування

- *Метод лінійного сортування* (сортування відбором) – це метод, на кожному кроці алгоритму якого відбирається єдиний елемент при перегляді елементів масиву один за одним.
- При застосуванні даного алгоритму для сортування масиву з *n* елементів знадобиться *(n-1)* цикл.
- Суть алгоритму метода:
  1. Знайти найменший елемент масиву й помістити його в *R[1]*. Для здійснення цієї операції необхідно у вихідному масиві порівняти *R[1]* з кожним наступним елементом (тобто з *R[2]*, *R[3]*, *R[4]*, *R[5]*). Якщо виявиться, що якийсь із цих елементів менше за *R[1]*, то варто поміняти їх місцями й продовжувати порівняння доти, доки з *R[1]* не буде порівняно останній елемент масиву. Якщо при такому порівнянні виявиться, що якийсь елемент більше або дорівнює *R[1]*, то необхідно перейти до порівняння *R[1]* з наступним елементом, а цей елемент залишити на тому ж місці.
  2. Виключивши з розгляду елемент у позиції 1, повторити зазначений процес, починаючи з позиції 2, тобто серед елементів, що залишилися, знайти тим самим способом найменший і помістити його в *R[2]*.

3. Ці дії виконати для всіх елементів масиву, за винятком останнього.

Сортування методом «бульбашки»

- *Метод «бульбашки»* – багатопрохідний процес попарного порівняння сусідніх елементів, який триває доти, доки не буде зареєстрований цикл, на якому не відбулося жодної перестановки. Метод працює тим ефективніше, чим більше впорядковані елементи масиву у вихідному стані. В методі «бульбашки» заздалегідь не відомо, скільки циклів знадобиться для того, щоб *n* елементів масиву виявилися повністю відсортованими.
- Суть метода: Виходячи з закономірності: *R[1]< R[2], R[2]< R[3], R[3]< R[4], . . ., R[n-1]< R[n]* при сортуванні методом бульбашки *R[1]* достатньо порівняти тільки з *R[2]*. Якщо виявиться, що *R[1]* менше або дорівнює йому, то варто відразу ж перейти до порівняння *R[2]* з *R[3]*, *R[3]* з *R [4]*, . . . , *R [n-1] з R[n]*. Якщо в результаті таких порівнянь буде виявлено, що якийсь елемент більший за наступний за ним елемента, то, помінявши їх місцями, потрібно продовжити процес порівнянні до кінця масиву. Завершивши черговий цикл по елементах масиву, необхідно повернутися до його початку і знову повторити весь процес попарного порівняння елементів. Якщо в результаті чергового циклу не буде зроблено ні однієї перестановки, то це означає, що масив став впорядкованим.
- Алгоритм сортування методом «бульбашки»: повторювати наступний процес доти, доки не буде зареєстрований цикл, на якому не було ні однієї перестановки: порівнювати суміжні елементи масиву (тобто *R[1]* з *R[2]*, *R[2]* з *R[3]*, *R[3]* з *R[4]*,...,*R[n-1]* з *R[n]*). Якщо вони стоять за зменшенням значень, то поміняти їх місцями.

Алгоритм сортування методом вставки

- Суть сортування методом вставки: Процес повинен виконуватися багаторазово. За опорний елемент обирається другий елемент списку. Потім, вибирається новий опорний елемент, що знаходиться на одну позицію нижче попереднього, і так до кінця списку.
- Суть алгоритму сортування методом вставки: Алгоритм реалізується за допомогою вкладених циклів. Кожне виконання тіла зовнішнього циклу призводить до того, що внутрішній цикл ініціалізується і виконується до тих пір, поки не буде виконана умова його закінчення. Таким чином, одноразове виконання тіла зовнішнього циклу супроводжуватиметься багаторазовим виконанням тіла внутрішнього циклу.

Закріплення матеріалу

1. Пояснити роботу лінійного алгоритму сортування.
2. Пояснити роботу алгоритму сортування методом «бульбашки».
3. Пояснити роботу алгоритму сортування методом вставки.
4. Які алгоритми сортування, окрім розглянутих на лекції, вам відомі?

РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ

Лекція 20

ХЕШ-ТАБЛИЦІ. ДЕРЕВА

Мета лекції: розгляд хеш-таблиць та дерев як структур даних.

Зміст лекції

1. Хеш-таблиця як структура даних.
2. Дерево як структура даних.

Хеш-таблиця як структура даних

- *Хеш-таблиця* – це динамічна структура даних, що реалізує інтерфейс асоціативного масиву, тобто дозволяє зберігати пари (ключ, значення) і виконувати три операції: операцію додавання нової пари, операцію пошуку і операцію видалення пари по ключу.
- Основні варіанти хеш-таблиць:
  - з ланцюжками, тобто хеш-таблиця містить деякий масив, елементами якого є списки пар;
  - з відкритою адресацією, тобто хеш-таблиця містить деякий масив, елементами якого є пари.
- Виконання операції в хеш-таблиці починається з обчислення хеш-функції від ключа. Отримане хеш-значення відіграє роль індексу в масиві H. Потім виконується операція додавання, видалення або пошук об'єкта за знайденим індексом.
- *Колізія* – це ситуація, коли для різних ключів отримується одне й те саме хеш-значення.
- Механізм вирішення колізій – це важлива складова будь-якої хеш-таблиці.
- В деяких спеціальних випадках вдається уникнути колізій взагалі. Наприклад, якщо всі ключі елементів відомі заздалегідь (або дуже рідко змінюються), то для них можна знайти деяку досконалу хеш-функцію, яка розподілить їх по коміркам хеш-таблиці без колізій.
- *Хеш-таблиці з прямою адресацією* – це хеш-таблиці, що використовують досконалі хеш-функції і не потребують механізму вирішення колізій.
- *Коефіцієнт заповнення хеш-таблиці* (load factor) – це число збережених елементів, поділене на розмір масиву H. Load factor є важливим параметром, від якого залежить середній час виконання операцій.

Дерево як структура даних

- *Дерево* – в інформатиці та програмуванні одна з найпоширеніших структур даних. Формально дерево визначається як скінченна множина T з однією або більше вершинами (вузлами, nodes), яке задовольняє наступним вимогам:
  - існує один виокремлений вузол – корінь (root) дерева;
  - інші вузли (за виключенням кореня) розподілені серед $m \geq 0$ непересічних множин T1…Tm і кожна з цих множин в свою чергу є деревом. Дерева T1…Tm мають назву піддерев (subtrees) даного кореня.
- *Двійкове дерево* – деревоподібна структура даних, в якій кожен вузол має не більше двох нащадків (дітей). В такій структурі перший вузол називається батьківським вузлом, а діти називаються лівим і правим нащадками відповідно.
- *Двійкове дерево пошуку* (binary search tree, BST) – це двійкове дерево, для якого виконуються наступні умови:
  - обидва піддерева – ліве і праве, є двійковими деревами пошуку;
  - у всіх вузлів лівого піддерева довільного вузла «A» значення ключів менші, ніж значення ключа самого вузла «A», в той час, як у всіх вузлів правого піддерева того ж вузла «A» значення ключів не менше, ніж значення ключа вузла «A»;
  - дані в кожному вузлі повинні володіти ключами, для яких визначена операція порівняння.
- При реалізації двійкове дерево пошуку можна визначити наступним чином: Двійкове дерево складається з вузлів – записів виду (data, left, right), де data – деякі дані, прив'язані до вузла, left і right – посилання на вузли, які є дітьми даного вузла – лівий і правий нащадки відповідно. Для оптимізації алгоритмів конкретні реалізації припускають також визначення поля parent в кожному вузлі (крім кореневого) – посилання на батьківський елемент. Дані (data) володіють ключем (key), на якому визначена операція порівняння "менше". В конкретних реалізаціях це може бути пара (key, value) – (ключ і значення), або посилання на таку пару, або просте визначення операції порівняння на необхідній структурі даних. Для будь-якого вузла «A» виконуються властивості дерева пошуку: $key[left[A]] < key[A] \leq key[right[A]]$, тобто ключі батьківського вузла більше ключів лівого сина і нестрого менше ключів правого.
- Основною перевагою двійкового дерева пошуку перед іншими структурами даних є висока ефективність реалізації заснованих на ньому алгоритмів пошуку і сортування.

Закріплення матеріалу

1. Що таке хеш-таблиця, хеш-код та колізія?
2. Що таке дерево? Які правила бінарного дерева пошуку вам відомі?

РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ

Лекція 21

БАЗОВІ АЛГОРИТМИ ПОШУКУ

Мета лекції: розгляд базових алгоритмів пошуку.

Зміст лекції

1. Алгоритм послідовного пошуку.
2. Алгоритм двійкового пошуку.

Алгоритм послідовного пошуку

- *Алгоритм послідовного пошуку* (лінійний пошук) – знаходження заданого значення довільної функції на деякому її відрізку.
- Алгоритм послідовного пошуку є найпростішим алгоритмом пошуку. Пошук значення функції здійснюється порівнянням чергового розглянутого значення (як правило пошук відбувається зліва направо) і, якщо значення збігаються, то пошук вважається завершеним.
- Якщо відрізок має довжину N, то знайти рішення з точністю до ε можна за час N/ε. Таким чином асимптоматична складність алгоритму – $O(n)$.
- Через малу ефективність в порівнянні з іншими алгоритмами алгоритм послідовного пошуку використовують лише тоді, коли відрізок пошукової системи містить дуже мало елементів.
- Алгоритм послідовного пошуку може працювати в потоковому режимі при безпосередньому отриманні даних з будь-якого джерела.
- Лінійний пошук часто використовується у вигляді лінійних алгоритмів пошуку максимуму/мінімуму.

Алгоритм двійкового пошуку

- *Алгоритм двійкового пошуку* – це алгоритм знаходження заданого значення у впорядкованому масиві, який полягає у порівнянні серединного елемента масиву з шуканим значенням і повторенні алгоритму для тієї або іншої половини залежно від результату порівняння.
- Трудомісткість алгоритму $1+\log_2 n$, де n — кількість елементів в масиві.

Закріплення матеріалу

1. Пояснити роботу алгоритму послідовного пошуку.
2. Пояснити роботу алгоритму двійкового пошуку.
3. Які алгоритми пошуку, окрім розглянутих на лекції, вам відомі?

РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ

Лекція 22
БІНАРНЕ ДЕРЕВО ПОШУКУ. АВЛ-ДЕРЕВО

Мета лекції: розгляд принципів роботи з бінарними деревами пошуку.

Зміст лекції
1. Видалення вузла з бінарного дерева пошуку.
2. АВЛ-дерева.

Видалення вузла з бінарного дерева пошуку

- При видаленні вузла з дерева потрібно враховувати три можливих варіанти:
- Перший варіант: вузол, що видаляється, не має правого нащадка.
- Другий варіант: вузол, що видаляється, має правого нащадка, у якого немає лівого нащадка.
- Третій варіант: вузол, що видаляється, має правого нащадка, у якого є лівий нащадок.

АВЛ-дерева

- *АВЛ-дерево* – збалансоване по висоті двійкове дерево пошуку, для кожної вершини якого висота її двох піддерев відрізняється не більше, ніж на 1.
- *АВЛ* – абревіатура, утворена першими літерами творців Адельсон-Вельського Георгія Максимовича і Ландіса Євгена Михайловича.
- Висота дерева логарифмічно залежить від числа його вузлів: висота h АВЛ-дерева з n ключами лежить в діапазоні від $\log_2(n + 1)$ до $1.44\log_2(n + 2) - 0.328$. А оскільки основні операції над двійковим деревом пошуку (пошук, вставка і видалення вузлів) лінійно залежать від його висоти, то отримуємо гарантовану логарифмічну залежність часу роботи цих алгоритмів від числа ключів, що зберігаються в дереві.
- Дерева пошуку забезпечують збалансованість тільки в ймовірнісному сенсі: ймовірність отримання сильно незбалансованого дерева при великих n хоча і є дуже малою, але залишається не рівною нулеві.
- *Балансування вершини* (щодо АВЛ-дерева) – це операція, яка в разі різниці висот лівого та правого піддерев змінює зв'язку предок-нащадок в піддереві даної вершини так, що різниця стає <= 1, інакше – не змінює нічого. Зазначений результат отримується обертаннями піддерева даної вершини. Існують чотири типи обертання: обертання вліво; обертання вправо; обертання вліво, а потім вправо; обертання вправо, а потім вліво.

Закріплення матеріалу
1. Яка головна відмінність АВЛ-дерева від інших дерев пошуку?

РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ

Лекція 23

ВСТУП В ТЕОРІЮ ГРАФІВ

Мета лекції: розгляд загальних понять теорії графів.

Зміст лекції

1. Загальні поняття теорії графів.
2. Способи представлення графа.
3..

Загальні поняття теорії графів

- *Граф* – це найважливіше математичне поняття дискретної математики. На основі теорії графів будуються моделі різноманітних прикладних задач, таких як маршрутизація, розподіл ресурсів, дискретна оптимізація, сіткове планування і керування, аналіз і проєктування організаційних структур, аналіз процесу їх функціонування тощо.
- *Граф* (неформально) – це множина точок і ліній зі стрілками або без них, які з'єднують ці точки.
- *Граф* G (класичне визначення) – це сукупність двох множин V точок і E ліній, між якими визначене відношення інцидентності, причому, кожен елемент $e \in \mathrm{E}$ інцидентний рівно двом елементам $v', v'' \in \mathrm{V}$.
- Першою роботою теорії графів як математичної дисципліни вважають статтю Ейлера (1736 г.), в якій розглядалася задача про Кенігсбергські мости. Ейлер показав, що не можна обійти сім міських мостів і повернутися у вихідну точку, пройшовши по кожному мосту рівно один раз.
- Графи служать зручним засобом опису зв'язків між об'єктами. Наприклад, розглядаючи граф, який зображає мережу доріг між населеними пунктами, можна визначити маршрут проїзду від пункту А до пункту Б. Якщо таких маршрутів виявиться кілька, хотілося б вибрати в певному сенсі оптимальний, наприклад Пошук по графу, найкоротший або найбезпечніший. Для розв’язку задачі вибору потрібно проводити певні обчислення над графами.
- Графи є способом "візуалізації" зв'язків між певними об'єктами. Зв'язки ці можуть бути "направленими", як, наприклад, в генеалогічному древі, або "ненаправленими" (мережа доріг з двостороннім рухом). Відповідно до

цього в теорії графів виділяють два основні типи графів: орієнтовані (або направлені) і неорієнтовані.

- Неорієнтований граф G задається двома множинами G = (V, E), де V – кінцева множина, елементи якої називають вершинами або вузлами; E – множина неврегульованих пар на V, тобто підмножина множини двоелементних підмножин V, елементи якої називають ребрами.
- *Суміжні вершини* – це вершини *u* і *v*, з'єднані ребром (*u* | -> *v*). Суміжні вершини ще називають кінцями ребра {*u*, *v*}.
- Ребро e називають *інцидентним* вершині v, якщо вона є одним з його кінців.
- *Ступінь вершини v* – кількість всіх ребер G, інцидентних вершині *v*.
- *Ациклічний граф* – це неорієнтований граф, що не містить циклів.
- Орієнтований граф G задається двома множинами G = (V, E), де V – кінцева множина, елементи якої називають вершинами або вузлами; E – множина впорядкованих пар на V, тобто підмножина множини V*V, елементи якої називають дугами.
- Якщо дуга *e* = (*u*, *v*), то говорять, що дуга *e* веде з вершини *u* в вершину *v*, і позначають це $u \to v$.
- *Шлях в графі* – це набір вершин $v_1 ... v_n$, при якому з вершини v_1 можна потрапити в вершину v_n.
- *Ланцюг* – це маршрут, в якому всі ребра попарно різні.
- *Цикл* – це замкнений маршрут, який є ланцюгом.
- *Кратні ребра* – це ребра, інцидентні до однієї і тієї ж вершини.
- *Мультограф* – це граф, що містить кратні ребра.
- *Псевдограф* – це граф, що містить кратні ребра і петлі.
- *Кінцевий граф* – це граф, множина вершин і ребер якого звичайна.
- Порожній граф – це граф, множина ребер якого порожня.
- *Доповнення графа* G – це граф $\overline{G}$, що має ті ж вершини, що і граф G і тільки ті ребра, які необхідно додати до графа G, щоб він став повним.
- *Задати граф* – означає описати множини його вершин і ребер, а також відношення інцидентності. Коли граф G – кінцевий, для його опису досить занумерувати вершини і ребра.
- *Повністю заданий граф у точному значенні* – це граф, нумерація вершин і ребер якого зафіксована.
- *Ізоморфні графи* – це графи, що відрізняються тільки нумерацією.
- Графи G_1 і G_2 ізоморфні якщо їх вершини можна пронумерувати таким чином, що ребро e_j тоді і тільки тоді з'єднує вершини v_i і v_k у графі G_1, коли ребро e'_j з'єднує вершини v'_i і v'_k у графі G_2.
- *Правильно укладений на площині граф* – це граф, графічне подання якого таке, що ребра графа перетинаються тільки в його вершинах.

- *Планарний граф* – це граф G, ізоморфний деякому графу G^*, правильно укладеному на площині. Тобто плоский граф – це граф, який можна правильно укласти на площині.

Способи представлення графа

- Для представлення графа можна використовувати графічний вигляд, але він непридатний для машини.
- Тому існують інші способи представлення графів. В теорії графів застосовуються:
- 1. *Матриця інцидентності* – це матриця A з n рядками, що відповідають вершинам, і m стовпцями, що відповідають ребрам. Для орієнтованого графа стовпець, що відповідає дузі (x, y) містить - 1 в рядку, що відповідає вершині x і 1, в рядку, що відповідає вершині y. У всіх інших 0. Петлю, тобто дугу (x, x) можна представляти іншим значенням в рядку x, наприклад, 2. Якщо граф неорієнтований, то стовпець, що відповідає ребру (x, y), містить 1, відповідні x і y, і нулі у всіх інших рядках.
- 2. *Матриця суміжності* – це матриця $n \times n$, де n – число вершин, де $b_{ij} = 1$, якщо існує ребро, що йде з вершини x в вершину y і $b_{ij} = 0$ в іншому випадку).

Пошук по графу

- *Пошук в глибину* (англ. Depth-First Search, DFS) – один з методів обходу графа. Стратегія пошуку в глибину, як і випливає з назви, полягає в тому, щоб йти «вглиб» графа, наскільки це можливо. Алгоритм пошуку описується рекурсивно: перебираємо всі вихідні з розглянутої вершини ребра. Якщо ребро веде в вершину, яка не була розглянута раніше, то запускаємо алгоритм від цієї нерозглянутої вершини, а після повертаємося і продовжуємо перебирати ребра. Повернення відбувається в тому випадку, якщо в даній вершині не залишилося ребер, які ведуть в нерозглянуту вершину. Якщо після завершення алгоритму не всі вершини були розглянуті, то необхідно запустити алгоритм від однієї з нерозглянутих вершин.
- Пошук в ширину (англ. Breadth-First Search, BFS) працює шляхом послідовного перегляду окремих рівнів графа, починаючи з вузла-джерела u. Розглянемо всі ребра (u, v), що виходять з вузла u. Якщо черговий вузол v є цільовим вузлом, то пошук завершується; в іншому випадку вузол v додається в чергу. Після того, як будуть перевірені всі ребра, що виходять з вузла u, з черги отримується наступний вузол u і процес повторюється.

Закріплення матеріалу

1. Де використовується теорія графів?
2. Пояснити різницю між орієнтованим графом і неорієнтованим.
3. Як можна представити граф?
4. Пояснити суть пошуку в глибину та в ширину.

РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ

Лекція 24

МЕТОДИ ТА АЛГОРИТМИ ТЕОРІЇ ГРАФІВ

Мета лекції: розгляд методів та алгоритмів теорії графів.

Зміст лекції

1. Компоненти зв'язності.
2. Ейлерів цикл.
3. Алгоритм Дейкстри.
4. Хвильовий алгоритм.

Компоненти зв'язності

- *Компонента зв'язності графа* – це деяка множина вершин графа, така, що для будь-яких двох вершин з цієї множини існує шлях з однієї в іншу, і не існує шляху з вершини цієї множини в вершину не з цієї множини.
- Найбільш поширені методи пошуку компоненти зв'язності – за допомогою обходу в глибину або в ширину.

Ейлерів цикл

- *Ейлерів шлях* (Ейлерів ланцюг) в графі – це шлях (ланцюг), що проходить по всім дугам (ребрам) графа до того ж лише по одному разу.
- *Ейлерів цикл* – це цикл графа, що проходить через кожне ребро (дугу) графа рівно по одному разу.
- *Ейлерів граф* – граф, що містить Ейлерів цикл.
- Напівейлерів граф – граф, що містить Ейлерів шлях (ланцюг).
- *Зв'язний граф* – це граф, що містить рівно одну компоненту зв'язності. Це означає, що між будь-якою парою вершин цього графа існує як мінімум один шлях.
- *Лема про рукостискання* – це положення теорії графів, згідно з яким будь-який кінцевий неорієнтовані граф має парне число вершин непарних ступенів. Лема бере назву від популярної аналогії: в групі людей, деякі з яких потискують один одному руки, парне число людей привітало таким чином непарне число колег.
- В неорієнтованому графі: Згідно з теоремою, доведеною Ейлером, в графі без одиночних вершин Ейлерів цикл існує тоді і тільки тоді, коли граф зв'язний і в ньому відсутні вершини непарного ступеня. Ейлерів ланцюг в графі існує тоді і тільки тоді, коли граф зв'язний і містить не більше двох

вершин непарного ступеня. З огляду на лему про рукостискання, число вершин з непарним ступенем повинно бути парним. А значить, Ейлерів шлях існує тільки тоді, коли це число дорівнює нулю або двом. Причому коли воно дорівнює нулю, Ейлерів шлях вироджується в Ейлерів цикл.

- Орієнтований граф G = (V, A) містить Ейлерів цикл тоді і тільки тоді, коли він сильно зв'язаний і для кожної вершини графа її вихідний напівступінь дорівнює її напівступеню результату, тобто в вершину входить стільки ж ребер, скільки з неї і виходить.

Алгоритм Дейкстри

- *Задача про найкоротший шлях* (англ. shortest path problem) – задача пошуку найкоротшого шляху (ланцюга) між двома точками (вершинами) на графі, в якій мінімізується сума ваг ребер, що складають шлях.
- *Алгоритм Дейкстри* (англ. Dijkstra's algorithm) – алгоритм на графах, винайдений нідерландським ученим Е. Дейкстрою в 1959 році. Знаходить найкоротшу відстань від однієї з вершин графу до всіх інших. Алгоритм працює тільки для графів без ребер від'ємної ваги. Алгоритм широко застосовується в програмуванні і технологіях, наприклад, його використовують протоколи маршрутизації OSPF і IS-IS.

Хвильовий алгоритм

- *Алгоритм Лі* (хвильовий алгоритм) – алгоритм пошуку шляху, алгоритм пошуку найкоротшого шляху на планарному графі. Належить до алгоритмів, заснованих на методах пошуку в ширину.
- В основному використовується при комп'ютерному трасуванні (розводці) друкованих плат, з'єднувальних провідників на поверхні мікросхем. Інше застосування хвильового алгоритму – пошук найкоротшої відстані на карті в комп'ютерних стратегічних іграх.
- Робота алгоритму включає в себе три етапи: ініціалізацію, поширення хвилі і відновлення шляху.

Закріплення матеріалу

1. Що таке компонента зв'язності в графі?
2. Що таке Ейлерів шлях?
3. Що таке Ейлерів цикл?
4. Що таке Ейлерів граф?
5. В чому суть леми про рукостискання?
6. За яких обставин Ейлерів шлях можливий в неорієнтованому графі?
7. Які умови повинні виконатися, щоб орієнтований граф містив Ейлерів цикл?
8. В чому суть задачі про найкоротший шлях?
9. В чому суть алгоритму Дейкстри?
10. В чому суть хвильового алгоритму?

РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ

Лекція 25

МЕТОДИ ТА АЛГОРИТМИ ТЕОРІЇ ГРАФІВ (ПРОДОВЖЕННЯ)

Мета лекції: розгляд методів та алгоритмів теорії графів.

Зміст лекції

1. Алгоритм Флойда-Уоршелла.
2. Топологічне сортування та його реалізація.

Алгоритм Флойда-Уоршелла

- *Алгоритм Флойда-Уоршелла* – алгоритм для знаходження найкоротших відстаней між усіма вершинами зваженого графа без циклів з від'ємною вагою з використанням методу динамічного програмування.
- Цей алгоритм був одночасно опублікований в статтях Роберта Флойда (Robert Floyd) і Стівена Уоршела (Stephen Warshall) в 1962 р., хоча в 1959 р. Бернард Рой (Bernard Roy) опублікував практично такий самий алгоритм, але це пройшло повз увагу.
- В основі алгоритму лежать дві властивості найкоротшого шляху графа.
  - Перша властивість: Є найкоротший шлях $p_{1k} = (v_1, v_2, ..., v_k)$ від вершини v_1 до вершини v_k, а також його підшлях $p'(v_i, v_{i+1}, ..., v_j)$, при цьому діє умова 1 <= i <= j <= k. Якщо p – найкоротший шлях від v_1 до v_k, то p' також є найкоротшим шляхом від вершини v_i до v_j.
  - Друга властивість є основою алгоритму. Розглядається граф G з пронумерованими від 1 до n вершинами $\{v_1, v_2, ..., v_n\}$ і шлях p_{ij} від v_i до v_j, що проходить через певну множину дозволених вершин, обмежену індексом k. Тобто якщо k = 0, то розглядаються прямі з'єднання вершин одна з одною, оскільки множина дозволених проміжних вершин дорівнює нулю. Якщо k = 1 – розглядаються шляхи, що проходять через вершину v_1, при k = 2 – через вершини $\{v_1, v_2\}$, при k = 3 – $\{v_1, v_2, v_3\}$ і так далі.

Топологічне сортування та його реалізація

- *Топологічне сортування* – впорядкування вершин бесконтурного орієнтованого графа згідно часткового порядку, заданого ребрами орграфа на множині його вершин.
- *Маршрут* в орграфі – це послідовність вершин і дуг, що чергуються, виду $v_0\{v_0, v_1\}, v_1\{v_1, v_2\}, v_2 ... v_n$ (вершини можуть повторюватися).

- *Довжина маршруту* – це кількість дуг в ньому.
- *Шлях* – це маршрут в орграфі без повторюваних дуг.
- *Простий шлях* – це маршрут в орграфі без повторюваних вершин.
- Якщо існує шлях з однієї вершини в іншу, то друга вершина досяжна з першої.
- *Контур* – це замкнений шлях.
- *Частково впорядкована множина* – це математичне поняття, яке формалізує інтуїтивні ідеї впорядкування, розташування елементів в певній послідовності. Неформально множина частково впорядкована, якщо вказано, які елементи слідують за якими (які елементи більше за які). В загальному випадку може виявитися так, що деякі пари елементів не пов'язані відношенням «слідує за».
- *Задача топологічного сортування графа*: вказати такий лінійний порядок на його вершинах, щоб будь-яке ребро вело від вершини з меншим номером до вершини з більшим номером. Очевидно, що якщо в графі є цикли, то такого порядку не існує.
- *Орієнтована мережа* (або просто мережа) – це бесконтурний орієнтований граф. В задачах подібного плану розглядаються тільки кінцеві мережі.
- Топологічне сортування застосовується в самих різних ситуаціях, наприклад при розпаралелюванні алгоритмів, коли за деяким описом алгоритму потрібно скласти граф залежностей його операцій і, відсортувавши його топологічно, визначити, які з операцій є незалежними і можуть виконуватися паралельно (одночасно). Прикладом використання топологічного сортування може служити створення карти сайту, де має місце деревоподібна система розділів.

Закріплення матеріалу

1. Для чого потрібен алгоритм Флойда-Уоршела?
2. Якщо в графі є цикли, чи можна застосувати алгоритм Флойда-Уоршелла для такого графа?
3. Які властивості алгоритму Флойда-Уоршела вам відомі?
4. Що таке маршрут?
5. Що таке довжина маршруту?
6. Що таке шлях?
7. Що таке простий шлях?
8. Що таке контур в графі?
9. В чому полягає задача топологічного сортування?
10. Що таке частково впорядкована множина?
11. Що таке орієнтована множина?
12. Навіщо сортувати граф?
13. Які способи сортування графа ви знаєте?

РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ

Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ

Лекція 26

ДИНАМІЧНЕ ПРОГРАМУВАННЯ

Мета лекції: розгляд поняття динамічного програмування.

Зміст лекції

1. Поняття динамічного програмування.
2. Принципи динамічного програмування.

Поняття динамічного програмування

- *Динамічне програмування* (динаміка) – це спосіб вирішення складних задач шляхом розбиття їх на більш прості підзадачі.
- Цей спосіб можна застосовувати до задач з оптимальною структурою, що виглядають як набір підзадач, які перекриваються, складність якої менше за вихідну.
- Ідея полягає в розбитті складної задачі на менші підзадачі, розв’язати їх і сконструювати відповідь з цих підзадач для складної задачі. Часто ці підзадачі дублюються.
- Підхід динаміки полягає в тому, щоб розв’язати ці задачі, які дублюються, тільки 1 раз.
- *Динамічне програмування* – це коли є задача, яку незрозуміло як вирішувати, і її розбивають на менші задачі, які теж незрозуміло як вирішувати. © А. Кумок.

Принципи динамічного програмування

- Методи динаміки:
  - верху – просто запам'ятовування результатів тих підзадач, які можуть зустрітися надалі (рахуємо відразу від 1 до N);
  - знизу – включає в себе переформулювання складної задачі у вигляді рекурсивної послідовності простіших задач (рахуємо, «опустившись вниз», і, піднімаючись, збираємо результати в інші завдання).
- *Оптимальна підструктура* в динамічному програмуванні означає, що оптимальний розв’язок підзадач меншого розміру може бути використаний для розв'язання вихідної задачі.
- Щоб успішно розв’язати задачі динамікою, потрібні:
  1) Стан динаміки: параметр(и), які однозначно задають підзадачу.
  2) Значення початкових станів.

3) Переходи між станами: формула перерахунку.
4) Порядок перерахунку.
5) Положення відповіді на задачу: іноді це сума або, наприклад, максимум зі значень декількох станів.

- Порядок перерахунку: прямий, зворотний, лінива динаміка.
- *Підзадачі, що перекриваються*, в динамічному програмуванні означають підзадачі, які використовуються для розв’язку певної кількості задач (не однієї) більшого розміру (тобто декілька разів необхідно зробити одне й те саме).
- *Мемоізація* (запам'ятовування, від англ. memoization) в програмуванні – це збереження результатів виконання функцій для запобігання повторних обчислень. Це один із способів оптимізації, що застосовується для збільшення швидкості виконання комп'ютерних програм.
- Однією з основних властивостей задач, що вирішуються за допомогою динамічного програмування, є адитивність. Неаддитивні задачі розв’язуються іншими методами.
- *Адитивність* (лат. additivus – що додається) – це властивість величин, яка полягає в тому, що значення величини, яке відповідає цілому об'єкту, дорівнює сумі значень величин, що відповідають його частинам, в деякому класі можливого розбиття об'єкта на частини.

Закріплення матеріалу

1. Що таке динамічне програмування?
2. Які методи динаміки ви знаєте?
3. Що таке оптимальна підструктура?
4. Які види перерахунку ви знаєте?
5. Що таке завдання, які перекриваються?
6. В чому полягає суть мемоізації?
7. Що таке адитивність?

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. *Жураковский Ю.Л.* Теорія інформації та кодування: підручник / Ю.Л. Жураковский, В.П. Полторак. – К. : Вища школа, 2001. – 255 с.
2. *Алексеев А.П.* Информатика 2015: уч.пос. / А.П. Алексеев. – М.: Солон-пресс, 2015. – 400 с.
3. *Вернер М.* Основы кодирования / М. Вернер. – М: Техносфера, 2004. – 288 с.
4. *Мациевский С. В.* Информатика: учебное пособие / С. В. Мациевский, С.А. Ишанов, С.В. Клевцур / Калининград: Изд-во КГУ, 2003. – 140 с.
5. *Ремонтов А.П.* Вычислительные машины и системы: учебное пособие / А.П. Ремонтов, А.П. Писарев. – Пенза: Изд-во Пенз. гос. ун-та, 2006. – 96 с.
6. *Алексеев А.* Новейший самоучитель на компьютере / А. Алексеев, Г. Евсеев ; под ред. С. Шмоновича – М.: ДЕСС-КОМ, 2011. – 654с.
7. Информатика. Базовый курс / под ред. С.В. Симоновича. – СПб.: Питер, 2001. – 640с.
8. ДСТУ 3008:2015 Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання.
9. *Стинсон К.* Эффективная работа MicroSoft Word Professional / К. Стинсон, К. Зихерт. – СПб.: Питер, 2011. – 864 с.
10. *Холберг Брюс* Использование MS Excel : пер. с англ. / Брюс Холберг и др. – СПб. : Изд. дом «Вильямс», 2006. – 736с.
11. *Добролюбова М.В.* Програмування в середовищі Excel : метод. вк. / М.В.Добролюбова. – К.: НТУУ «КПІ» ООО «КАЖАН», 2011. – 47с.
12. *Андерсон Тим* Visual Basic / Тим Андерсон. – СПб.: БХВ – Санкт-Петербург, 2010. – 234с.
13. *Добролюбова М.В.* Програмування в середовищі Access : метод. вк. / М.В.Добролюбова. – К.: НТУУ «КПІ» ООО «КАЖАН», 2011. – 60с.
14. *Габассов Ю.* Internet. Эффективная технология работы в сети / Ю. Габассов. – СПб.: БХВ – Санкт-Петербург, 2008. – 446 с.
15. *Бондаренко М.Ф.* Комп'ютерна дискретна математика: підручник / Д.И. Бондаренко, Н.В. Білоус, А.Г. Руткас. – Харків: Компанія СМІТ, 2004. – 480с.
16. *Ахо Альфред* Структуры данных и алгоритмы / Альфред Ахо, Джон Хопкрофт, Джеффри Ульман. – М.: Издательский дом «Вильямс», 2003. – 384 с.
17. *Вирт Н.* Алгоритмы и структуры данных / Н. Вирт. – М.: Мир, 1989. – 360 с.
18. *Кормен Т.* Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Риверст. – М.: МЦМНО, 2002. – 960 с.
19. *Кнут Д.* Искусство программирования. Основные алгоритмы / Д. Кнут. – М.: Издательский дом «Вильямс», 2000. – 720 с.

ДОДАТОК А
ПРЕЗЕНТАЦІЇ ДО ЛЕКЦІЙ


Обчислювальна техніка та програмування
РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА.
ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Тема 1.1 ПОНЯТТЯ ІНФОРМАЦІЇ
Лекція 1
«Вступ. Професійні термінологічні визначення в області обчислювальної техніки, інформатики та програмування»
© М.В.Добролюбова

Терміни та визначення. Поняття інформації

Обчислювальна техніка – пристрій або система, здатний виконувати задану, чітко визначену послідовність операцій (введення-виведення, числові розрахунки, маніпулювання даними).

Програмування – процес проектування, написання, тестування та підтримки комп'ютерних програм.

Інформація – відомості про осіб, предмети, факти, події, явища і процеси незалежно від форми їх надання.

Документована інформація (документ) – зафіксована на матеріальному носії інформація з реквізитами, що дозволяють її ідентифікувати.

Інформаційні процеси – процеси збору, обробки, накопичення, зберігання, пошуку і розповсюдження інформації.

Інформаційна система (ІС) – організаційно впорядкована сукупність документів (масивів документів) і інформаційних технологій, у тому числі з використанням засобів обчислювальної техніки і зв'язку, що реалізовує інформаційні процеси.

Інформаційні ресурси (ІР) – окремі документи і окремі масиви документів, документи і масиви документів в інформаційних системах (бібліотеках, архівах, фондах, банках даних, інших інформаційних системах).

Групи ІР:

- інформація на фізичних носіях, а також засоби її обробки;
- засоби передачі і комунікації, включаючи навички і прийоми їх використання;
- засоби обробки інформації;
- вчені і фахівці – «виробники» інформації в різних сферах діяльності;
- органи і служби, що займаються збором, обробкою і зберіганням інформації.

© М.В.Добролюбова 2

Терміни та визначення. Поняття інформації

Інформаційні технології (ІТ) – це засновані на застосуванні персонального комп'ютера (ПК) способи обробки семантичної інформації – даних і знань, які реалізуються за допомогою автоматизованих інформаційних систем.

Дані – це інформація, яка із якихось причин не використовується, а тільки зберігається.

Загальна схема передачі інформації

Змістовна та формальна структури інформації

Змістовна структура природно орієнтована на зміст інформації, а **формальна** – на форму надання інформації.

Основні форми надання інформації:

- символьна (заснована на використанні символів – букв, цифр, знаків),
- текстова (використовує тексти – символи, розташовані в певному порядку),
- графічна (різні види зображень),
- звукова.

© М.В.Добролюбова 3

Терміни та визначення. Поняття інформації

Інформатика – це сукупність засобів всієї сучасної інформаційної теорії, техніки і технології, сумарне, комплексне позначення цієї області знань.

Кібернетика – наука про загальні принципи керування в комплексі складними (множинними) системами різної природи походження. Кібернетика існує незалежно від наявності або відсутності комп'ютерів.

Кібернетика і інформатика, дуже схожі дисципліни, але розрізняються в наголосі на різні акценти:

- в інформатиці – на властивостях інформації і апаратно-програмних засобах її обробки;
- в кібернетиці – на розробці концепцій і побудові моделей об'єктів з використанням, зокрема, інформаційного підходу.

@ М.В.Добролюбова 4

Історична довідка

Перша інформаційна революція

Винахід і освоєння людством мови (1-2 млн. років тому)

Друга інформаційна революція

Винахід винаході писемності (Александрійська бібліотека (3 ст. до н.е. - 4 ст. н.е., 500 тис. сувоїв))

Третя інформаційна революція

Друкарство (Першою друкованою книгою вважають текст, створений за допомогою ксилографії в Кореї в період з 704 по 751 роки)

@ М.В.Добролюбова

5

Історична довідка

Четверта та п'ята інформаційні революція

Створення сучасних інформаційних технологій

Телеграф

Телефон
(перший пристрій, що нагадує телефон, винайшов Шарль Бурсель, 1854)

Радіо і телебачення

Комп'ютер

@ М.В.Добролюбова

6

Класифікація кодів та їх характеристики

Дискретизація – це перетворення функцій неперервних змінних у функції дискретних змінних, за якими початкові неперервні функції можуть бути відновлені із заданою точністю.

Кодування – це процес перетворення повідомлень в упорядковану послідовність символів (знаків, елементів) одного алфавіту.

Кодова комбінація – це впорядкований набір символів (знаків, елементів) одного алфавіту.

Код – сукупність кодових комбінацій, побудованих за одним правилом і на основі одного алфавіту.

Класифікація кодів

За потужністю алфавіту (кількістю q символів алфавіту) коди розділяють на *двійкові* (q=2) та *недвійкові* (q>2). *Недвійкові* називають ще *багатопозиційними* або *багатоосновними*.

За коректувальною здатністю коди розділяють на *безнадмірні* та *надмірні*.

До *безнадмірних* кодів належать так звані прості або первинні коди, які використовують для первинного кодування джерел повідомлень. Ці коди не дозволяють виявляти і виправляти помилки. Це пояснюється тим, що такі коди використовують всі можливі комбінації і будь-яка помилка призводить до появи дозволеної кодової комбінації.

До *надмірних* кодів належать коди, які дозволяють виявляти та/або виправляти помилки. У цих кодах використовується тільки визначена частина можливих комбінацій, що дозволяє при спотворенні елементів кодових комбінацій виявити або виправити їх. Кількість виявлених або виправлених помилок буде залежати від коректувальної здатності коду, тобто від його надмірності та особливостей побудови.

За способом побудови коди розділяють на *блокові* та *неперервні*. До першої групи належать коди, які є сукупністю кодових комбінацій, а до другої – коди, для яких кодування і декодування становлять неперервний у часі процес.

7

Класифікація кодів та їх характеристики

Класифікація кодів

Блокові коди *за кількістю елементів у кодових комбінаціях* коду розділяють на *рівномірні* та *нерівномірні*. До першої групи належать коди, у яких всі комбінації, що складають код, мають однакову кількість елементів, а до другої – ті, у яких кодові комбінації коду можуть містити різну кількість елементів.

Блокові коди *за використанням перевірочних елементів* у кодових комбінаціях розділяють на *роздільні* та *нероздільні*. До першої групи належать коди, кодові комбінації яких містять інформаційні та перевірочні елементи, до другої – коди, у кодових комбінаціях яких не відокремлюються інформаційні та перевірочні елементи.

Блокові коди *за способом побудови перевірочних елементів* у кодових комбінаціях розділяють на *лінійні* (*групові*, *систематичні*) та *нелінійні* (*несистематичні*). До першої групи належать коди, у яких перевірочні елементи отримують як результат лінійних операцій над визначеними інформаційними елементами (для двійкових кодів за модулем 2), а до другої – коди, що будуються за іншими принципами.

@ М.В.Добролюбова 8

Класифікація кодів та їх характеристики

Основні характеристики кодів

- **довжина коду** – кількість елементів (символів, знаків), які складають кодову комбінацію;
- **кількість інформаційних елементів**;
- **кількість перевірочних елементів** (для коректувальних роздільних кодів);
- **алфавіт коду** Q – множина символів (знаків), що різняться між собою, яка використовується для побудови кодових комбінацій коду (для двійкових кодів);
- **потужність алфавіту коду** – кількість символів (знаків) алфавіту (для двійкового коду);
- **потужність коду** – кількість дозволених кодових комбінацій, які використовуються для передачі повідомлень (для блокових роздільних кодів у загальному вигляді, зокрема для двійкових кодів);
- **повна кількість кодових комбінацій** N – кількість всіх можливих комбінацій для даного коду;
- **надмірність коду**;
- **швидкість коду**;
- **вага кодової комбінації** – для двійкового коду визначається кількістю одиниць у кодовій комбінації;
- **кодова відстань між двома кодовими комбінаціями однакової довжини** – визначається як кількість одноіменних елементів (розрядів) з різними значеннями символів (відстань Хеммінга);
- **мінімальна кодова відстань** – визначається для коду в цілому як мінімальне значення кодових відстаней між усіма парами кодових комбінацій, що належать до даного коду. Мінімальна кодова відстань визначає його здатність виявляти та виправляти помилки.

Системи числення

Загальний випадок запису числа N в деякій позиційній системі числення з основою P

$$N = \sum_{i=-s}^{m-1} a_i P^i ,$$

де a – цифра від 0 до *P-1*,
P – основа системи числення.

Позиційними системами числення називаються такі, в яких вага кожної цифри *a* залежить від позиції в зображенні числа.

Максимальне ціле число, яке може бути надане в *m* розрядах:

$$N_{max} = P^m - 1.$$

Мінімальне значуще (не рівне 0) число, яке можна записати в *s* розрядах дробової частини:

$$N_{min} = P^{-s}.$$

Розрізняють двійкову, вісімкову, десяткову, шіснадцяткову системи числення тощо.

@ М.В.Добролюбова 10

Формати надання інформації

1 Формати надання чисел

Форми надання двійкових чисел:

- **природна форма** або форма з фіксованою комою (крапкою)

  +00721,35500; +00000,00328; -10301,20260

- **нормальна форма** або форма з плаваючою комою (крапкою)

  $+0{,}721355\times10^{3}$; $+0{,}328\times10^{-2}$; $-0{,}103012026\times10^{5}$

З фіксованою комою всі числа зображаються у вигляді послідовності цифр з постійним для всіх чисел положенням коми, що розділяє цілу частину від дробової.

З плаваючою комою кожне число зображається у вигляді двох груп цифр. Перша група цифр називається мантисою, друга – порядком, причому абсолютна величина мантиси повинна бути менше 1, а порядок – цілим числом. В загальному вигляді число у формі з плаваючою комою може бути надано таким чином:

$$N = \pm MP^{\pm r},$$

де M – мантиса числа ($|M| < 1$);
r – порядок числа (r – ціле число);
P – основа системи числення.

Знак числа зазвичай кодується двійковою цифрою, при цьому код 0 означає знак «+», код 1 – знак «-».

@ М.В.Добролюбова 11

Формати надання інформації

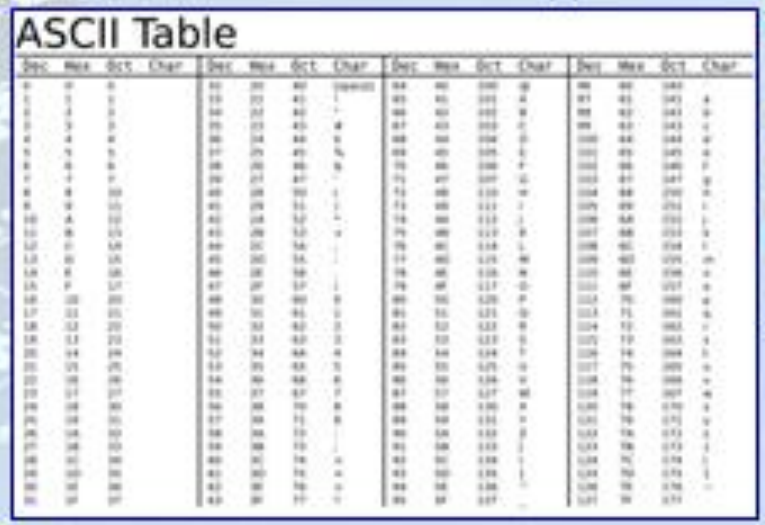
Одиниці вимірювання об'ємів інформації, збереженої або оброблюваної в ПК

- **1 біт** = 1 двійковий розряд
- **1 байт** = 8 біт
- **1 слово** = 2 байти
- **1 кілобайт** (Кбайт або Кб або К) = 2^{10} байт = 1024 байт
- **1 мегабайт** (Мбайт або Мб або М) = 2^{20} байт = 1024 Кбайт = 1 048 576 байт
- **1 гігабайт** (Гбайт або Гб або Г) = 2^{30} байт = 1024 Мбайт = 220 Кбайт = 1 073 741 824 байт
- **1 терабайт** (Тбайт або Тб або Т) = 2^{40} байт = 1024 Гбайт = 220 Мбайт = 1 099 511 627 776 байт

@ М.В.Добролюбова 12

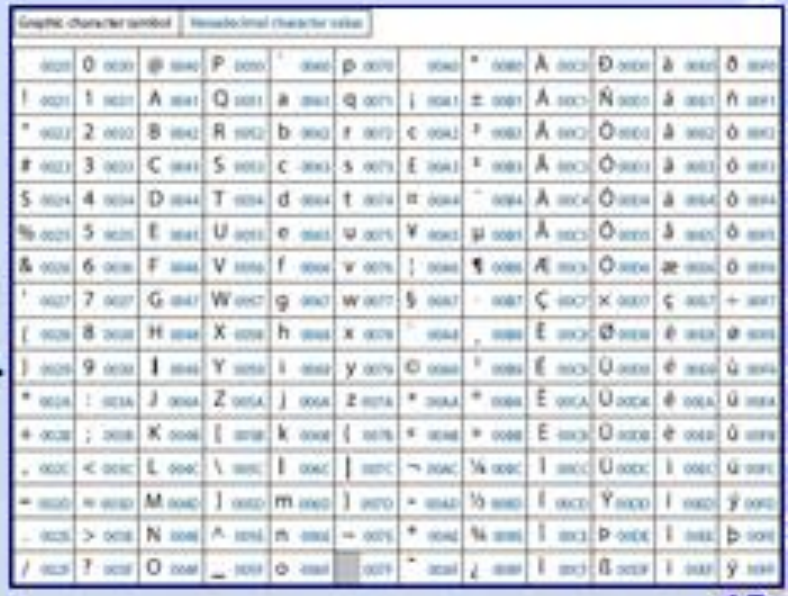
Формати надання інформації

2 Надання текстової інформації



Код ASCII (*American Standard Coding for Information Interchange*) – американський стандартний код для обміну інформацією має основний стандарт і його розширення. Основний стандарт для кодування символів використовує шістнадцятирічні коди 00 - 7F (десяткові 0 – 127), розширення стандарту – 80 - FF (десяткові коди 128 – 255).

Unicode розроблений в результаті об'єднаних зусиль декількох провідних фірм-виробників програмного і апаратного забезпечення. Unicode включає 65 536 різних двійкових кодів, що цілком достатньо навіть для надання всіх китайських і японських алфавітів, які широко використовуються.



@ М.В.Добролюбова

13

Формати надання інформації

3 Надання зображень

Растрові методи (bitmap techniques) представляють зображення як сукупність крапок, званих *пікселями* (pixel, скорочення від picture element – елемент зображення).

Переваги: чітко і максимально правдоподібно відображає відтінки кольорів, їх перетікання з одного в інший, а також тіні.
Недоліки: при збільшенні помітно втрачає в чіткості і виглядає неякісно.
Застосування: при роботі з фотографіями і іншими зображеннями з насиченою кольоровою гамою і плавними переходами кольору; при дизайні сайтів, іконок додатків.

Векторні методи (vector techniques) дозволяють уникнути проблем масштабування, характерних для растрових методів. Зображення надається у вигляді сукупності ліній і кривих.

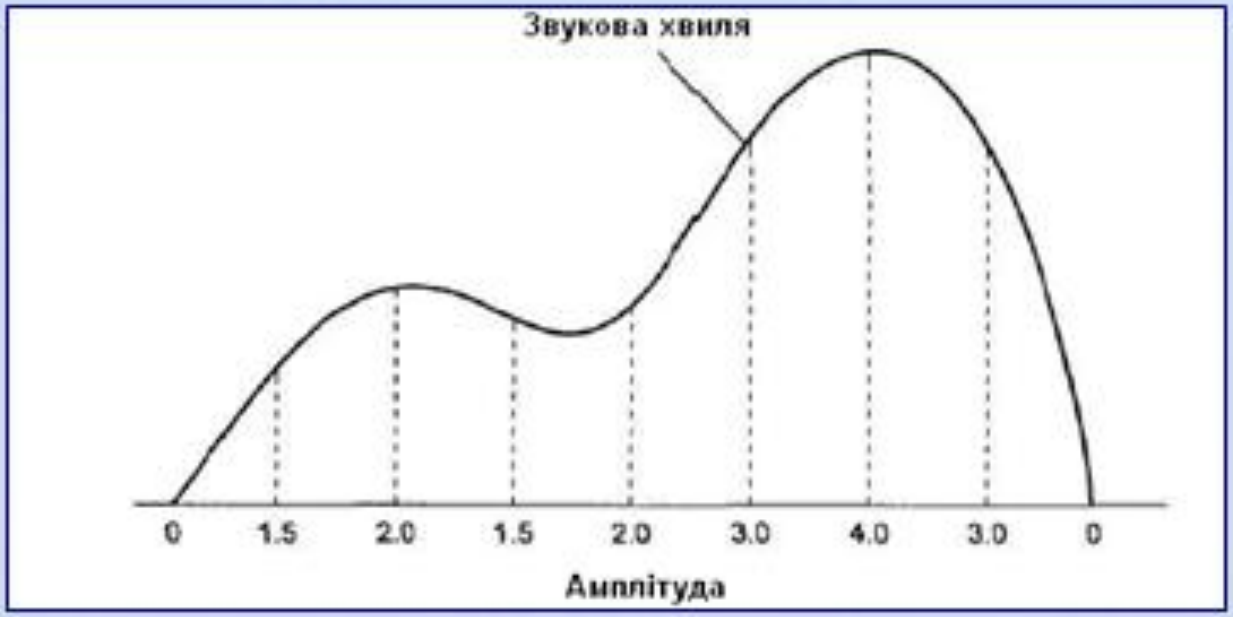
Переваги: масштабування без втрати чіткості зображення, малий розмір зображень.
Недоліки: дуже складно передати плавні переходи кольору і домогтися фотографічної якості.
Застосування: при створенні логотипів компаній, візиток, буклетів та іншої друкованої продукції; при створенні нових, оригінальних шрифтів та ілюстрацій.

© М.В.Добролюбова 14

Формати надання інформації

4 Надання звуку

Звуковий сигнал, представлений послідовністю 0, 1.5, 2.0, 1.5, 2.0, 3.0, 4.0, 3.0, 0




Обчислювальна техніка та програмування
РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА.
ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Тема 1.1 ПОНЯТТЯ ІНФОРМАЦІЇ
Лекція 2
«Стиснення даних»
@ М.В.Добролюбова

Основні методи стиснення інформації та їх характеристики

Фактори, які впливають на ступінь надлишковості даних:

- тип даних
- прийнята система кодування

Архів – стиснений варіант даних

Архіватори – програмні засоби, що реалізують методи стиснення.

Стиснення (архівування) файлів використовується для зменшення розмірів файлів при підготовці їх до передавання каналами зв'язку або до транспортування на зовнішніх носіях;

Стиснення (архівування) папок використовується як засіб зменшення обсягу папок перед довготерміновим зберіганням;

Стиснення (ущільнення) дисків використовується для підвищення ефективності використання дискового простору шляхом стиснення даних при запису їх на носії інформації.

Основні алгоритми (методи) стиснення даних та їх узагальнена класифікація

Теоретичні способи зменшення надлишковості даних:

- зміна вмісту даних
- зміна структури даних
- зміна вмісту і структури

Якщо при стисненні даних відбувається зміна їх вмісту, то метод стиснення є незворотнім. Такі методи часто називаються **методами стиснення з регульованими втратами інформації**. Приклади: JPEG (Joint Photographic Experts Group) для графічних даних; MPG – для відеоданих; MP3 – для аудіоданих

Якщо при стисненні даних відбувається тільки зміна структури даних, то метод стиснення є зворотнім. Такі методи часто називаються **методами стиснення без втрати інформації**. Приклади: GIF (Graphics Interchange Format), TIFF (Tagged Image File Format) – для графічних даних; AVI – для відеоданих; ZIP, ARJ, RAR, CAB, LH – для довільних типів даних.

© М.Е.Добролюбова 2

Основні методи стиснення інформації та їх характеристики

Алгоритми, що є основою методів стиснення без втрати інформації:

1) алгоритм кодування довжин серій RLE (Run Length Encoding)
2) словникові алгоритми або алгоритми групи KWE(KeyWord Encoding)
3) алгоритм Хаффмена

Алгоритм RLE (метод кодування довжин серій)

В основі алгоритму RLE лежить ідея виявлення послідовностей даних, що повторюються, та заміни цих послідовностей більш простою структурою, в якій вказується код даних та коефіцієнт повторення.

Коефіцієнт стиснення – визначається як відношення обсягу вихідних нестислих даних до обсягу стислих:

$$k = \frac{S_0}{S_c},$$

де k – коефіцієнт стиснення,
S_0 – обсяг вихідних даних,
S_c – обсяг стислих.

Якщо = 1, то алгоритм не робить стиснення, тобто вихідне повідомлення виявляється за обсягом рівним вхідному. Якщо < 1, то алгоритм породжує повідомлення більшого розміру, ніж нестиснене, тобто, здійснює «шкідливу» роботу.

Недолік алгоритму RLE: низька пристосованість до багатьох розповсюджених типів файлів. Алгоритм можна ефективно використовувати лише у комбінації з вторинним кодуванням.

© М.В.Добролюбова 3

Основні методи стиснення інформації та їх характеристики

Алгоритм RLE (метод кодування довжин серій)

Приклад

Нехай задана така послідовність даних, що підлягає стисненню:

1 1 1 1 2 2 3 4 4 4

В алгоритмі RLE пропонується замінити її наступною структурою:

1 4 2 2 3 1 4 3

де перше число кожної пари чисел – це код даних, а друге – коефіцієнт повторення. Якщо для зберігання кожного елементу даних вхідної послідовності відводиться 1 байт, то вся послідовність займатиме 10 байт пам'яті, тоді як вихідна послідовність (стиснений варіант) займатиме 8 байт пам'яті.

@ М.В.Добролюбова 4

Основні методи стиснення інформації та їх характеристики

Алгоритми групи KWE (словниковий метод)

В основу алгоритмів стиснення за ключовими словами покладено принцип кодування лексичних одиниць групами байт фіксованої довжини. Існує досить багато реалізацій таких алгоритмів, серед яких найбільш поширеними є алгоритм Лемпеля-Зіва (алгоритм LZ) та його модифікація алгоритм Лемпеля-Зіва-Велча (алгоритм LZW).

Алгоритм LZW побудований навколо таблиці фраз (словника), яка відображає рядки символів стискаємого повідомлення в коди фіксованої довжини. Таблиця володіє так званою властивістю передування, тобто для кожної фрази словника, що складається з деякої фрази w і символу K фраза w також міститься в словнику. Якщо всі частинки словника повністю заповнені, кодування перестає бути адаптивним.

Суть алгоритму:

- словником є потенційно нескінченний список фраз;
- алгоритм починає роботу з майже порожнього словника з одним закодованим рядком;
- коли зчитується черговий символ вхідної послідовності даних, він додається до поточного рядка;
- процес продовжується доти, поки поточний рядок відповідає деякій фразі з словника;
- коли поточний рядок є останнім збігом зі словником плюс щойно прочитаним символом повідомлення, кодер видає код, що складається з індексу збігу і наступного за ним символу, що порушив збіг рядків;
- нова фраза, що складається з індексу збігу і наступного за ним символшу, додається в словник;
- наступного разу, коли ця фраза з'явиться в повідомленні, вона може бути використана для побудови більш довгої фрази, що підвищує міру стиснення інформації.

© М.В.Добролюбова 5

Основні методи стиснення інформації та їх характеристики

Алгоритм Хаффмена (ентропійний метод)

В основі алгоритму Хаффмена лежить ідея кодування бітовими групами. Спочатку проводиться частотний аналіз вхідної послідовності даних. Після цього символи сортуються за спаданням частоти входження. Основна ідея полягає в наступному: чим частіше зустрічається символ, тим меншою кількістю біт він кодується. Результат кодування зводиться в словник, що необхідний для декодування.

Приклад

Нехай символ буде мати довжину 8 біт.

beep_boop_beer!

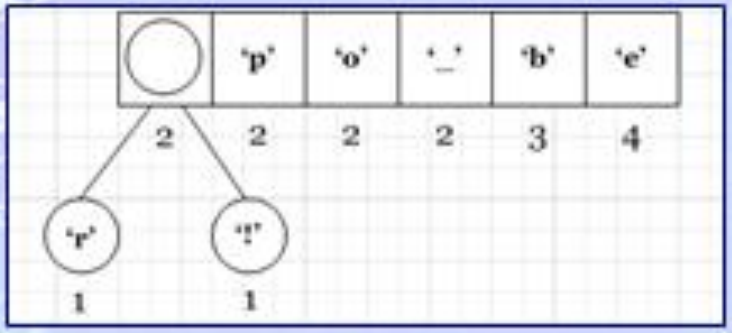
Весь рядок займає 120 біт пам'яті. Після кодування він займатиме 40 біт пам'яті.

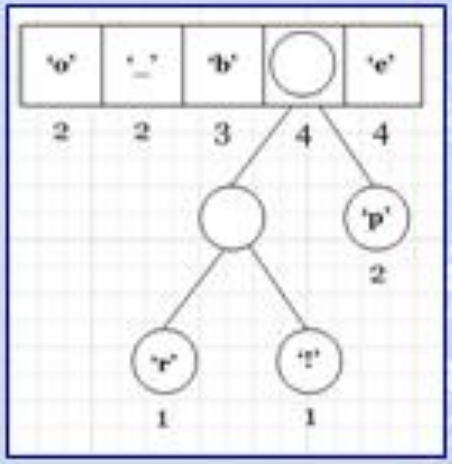
Частоти всіх символів

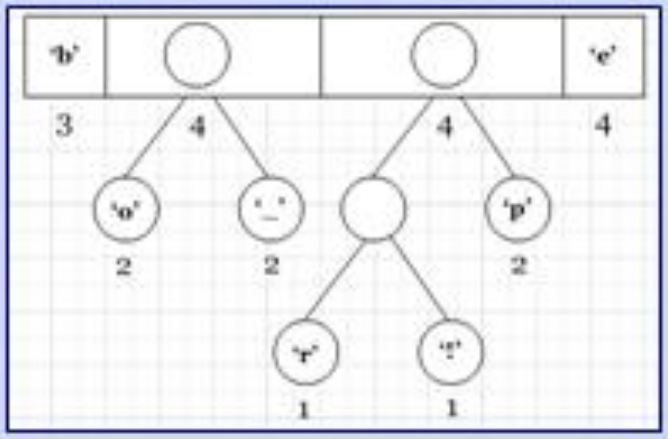
Символ	Частота
'b'	3
'e'	4
'p'	2
'_'	2
'o'	2
'r'	1
'!'	1

Основні методи стиснення інформації та їх характеристики

Створимо вузли бінарного дерева, використовуючи чергу з пріоритетами.





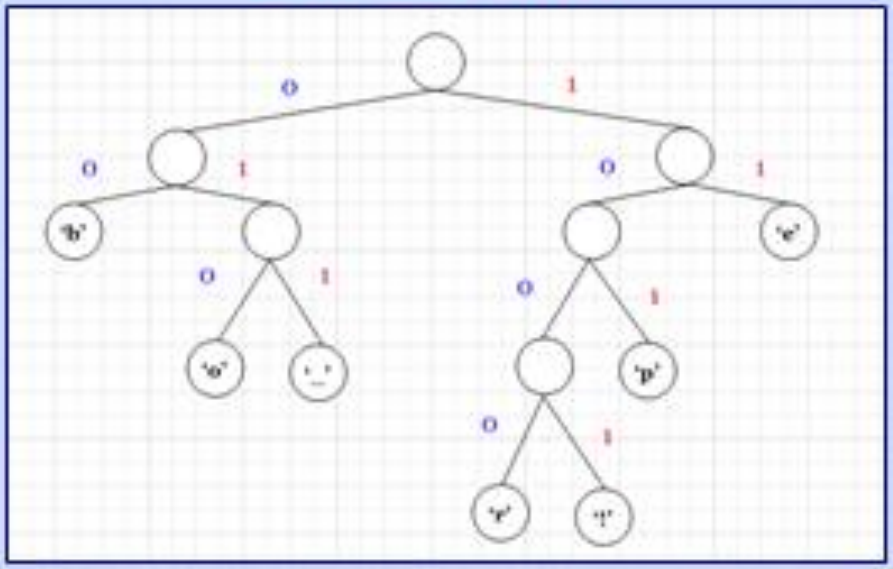


@ М.В.Добролюбова

7

Основні методи стиснення інформації та їх характеристики

Результуюче дерево



Коди для символів

Символ	Частота
'ь'	00
'е'	11
'р'	101
' '	011
'о'	011
'г'	1000
'!'	1001

Вхідний рядок у бінарному вигляді

0110 0010 0110 0101 0110 0101 0111 0000 0010 0000
0110 0010 0110 1111 0110 1111 0111 0000 0010 0000
0110 0010 0110 0101 0110 0101 0111 0010 0000

Закодований рядок

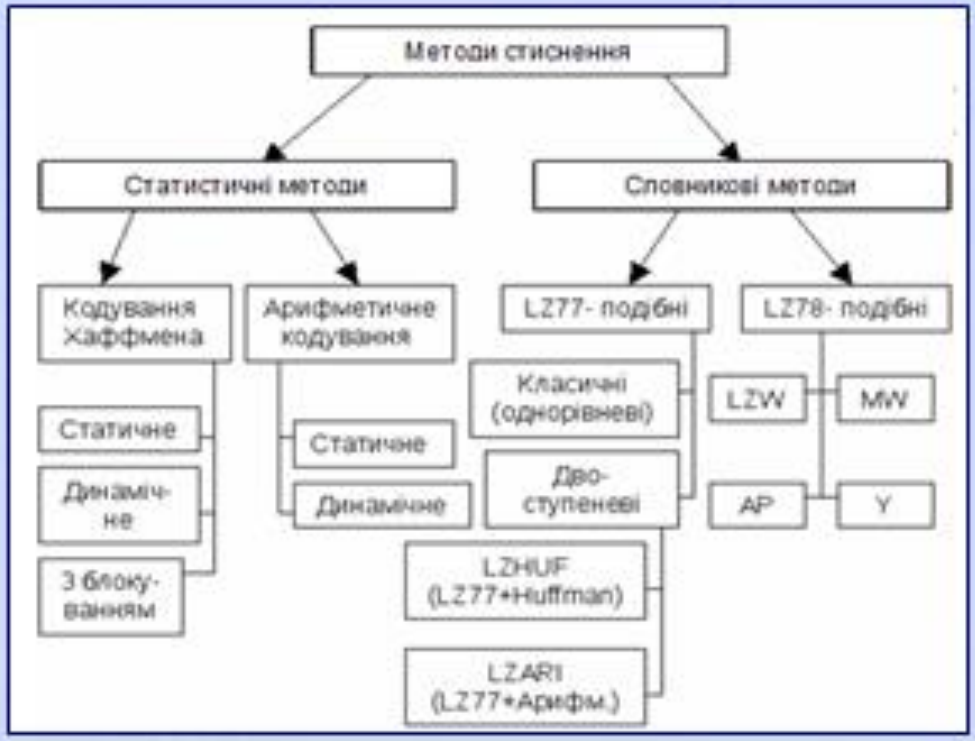
0011 1110 1011 0001 0010 1010 1100 1111 1000 1001

@ М.В.Добролюбова

8

Основні методи стиснення інформації та їх характеристики

Класифікація методів стиснення інформації



@ М.В.Добролюбова

9

Основні принципи оптимального кодування

Принцип побудови кода Шеннона–Фано

1. Всі повідомлення (літери) записуються в таблицю в порядку зменшення ймовірності їх появи.

2. Всю послідовність повідомлень (букв) ділять на 2 групи так, щоб суми ймовірностей в кожній групі були б приблизно однаковими. Верхній групі повідомлень (букв) присвоюється цифра «0» (як перший символ коду), а нижній цифра «1».

3. Кожна група повідомлень (букв) ділиться на дві підгрупи із збереженням тієї ж умови однакової суми ймовірностей. Верхнім підгрупами в обох групах знову присвоюється цифра «0» (як другий символ коду), а нижнім – цифра «1».

4. Такий розподіл продовжується до тих пір, поки в підгрупах не залишиться тільки по одному повідомленню (букві).

@ М.В.Добролюбова 11

Основні принципи оптимального кодування

Принцип побудови кода Хаффмена

1. Повідомлення виписуються в порядку збільшення ймовірностей.

2. Два останніх повідомлення об'єднуються в одне допоміжне повідомлення, якому приписується сумарна ймовірність.

3. Ймовірності повідомлень знову розташовуються в порядку збільшення ймовірностей в додатковому стовпці, а дві останні об'єднуються.

4. Процес продовжується до тих пір, поки не отримаємо єдине повідомлення з ймовірністю, рівною одиниці.

Основні принципи оптимального кодування

Код Лемпеля-Зіва

1. Компресор постійно зберігає певну кількість останніх оброблених символів у деякому буфері (ковзаючому словнику – sliding dictionary). Назва «ковзаючий» зумовлена тим, що його довжина постійна: кожного разу, коли компресор кодує наступний ланцюжок, він дописує його в кінець словника та «відрізає» відповідну кількість символів на початку буфера.

2. Під час обробки вхідного потоку символи, що надійшли, потрапляють у кінець буфера, зсуваючи попередні символи та витісняючи найстаріші.

3. Алгоритм виділяє (шляхом пошуку в словнику) найдовший початковий підрядок вхідного потоку, що співпадає з одним із підрядків у словнику, і подає на вихід пару (length, distance), де length – довжина знайденого у словнику підрядка, а distance – відстань від нього до вхідного підрядка (тобто фактично індекс підрядка в буфері, віднятий від його розміру). В разі, коли такого підрядка не знайдено, до вихідного потоку просто копіюється черговий символ вхідного потоку.

Стиснення даних у комп'ютерних інформаційних технологіях

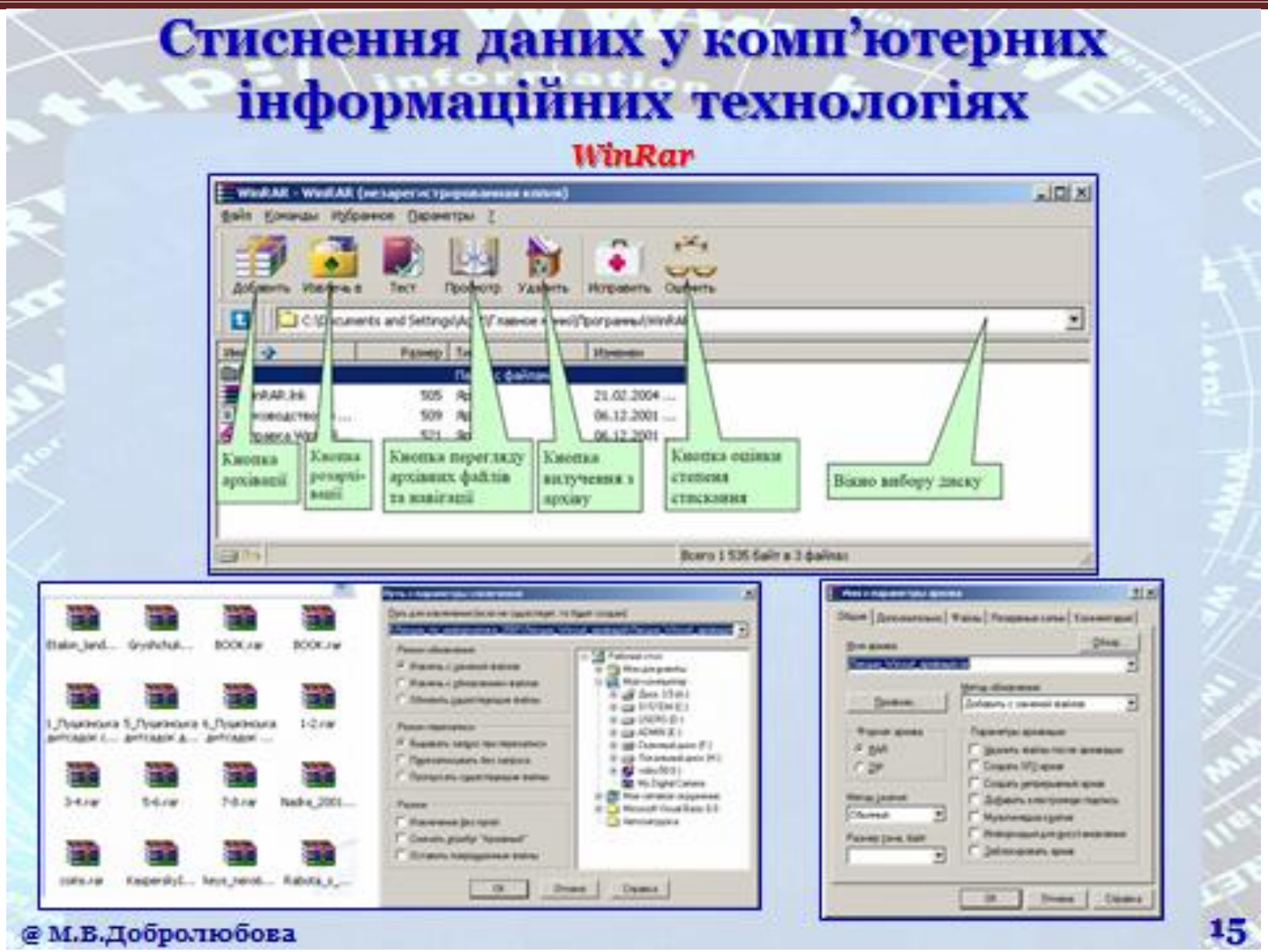
Основні послуги, що надаються архіваторами

1) створення нового архіву
2) додавання файлів в існуючий архів
3) розпаковування файлів з архіву
4) створення архівів, що саморозпаковуються (self-extractor archive)
5) створення розподілених архівів фіксованих розмірів для носіїв малої ємності
6) захист архівів паролями від несанкціонованого доступу
7) перегляд вмісту файлів різних форматів без попереднього розархівування
8) пошук файлів і даних всередині архіву
9) перевірка на віруси в архіві до розпаковування
10) вибір та налаштування коефіцієнта стиснення

@ М.В.Добролюбова 14

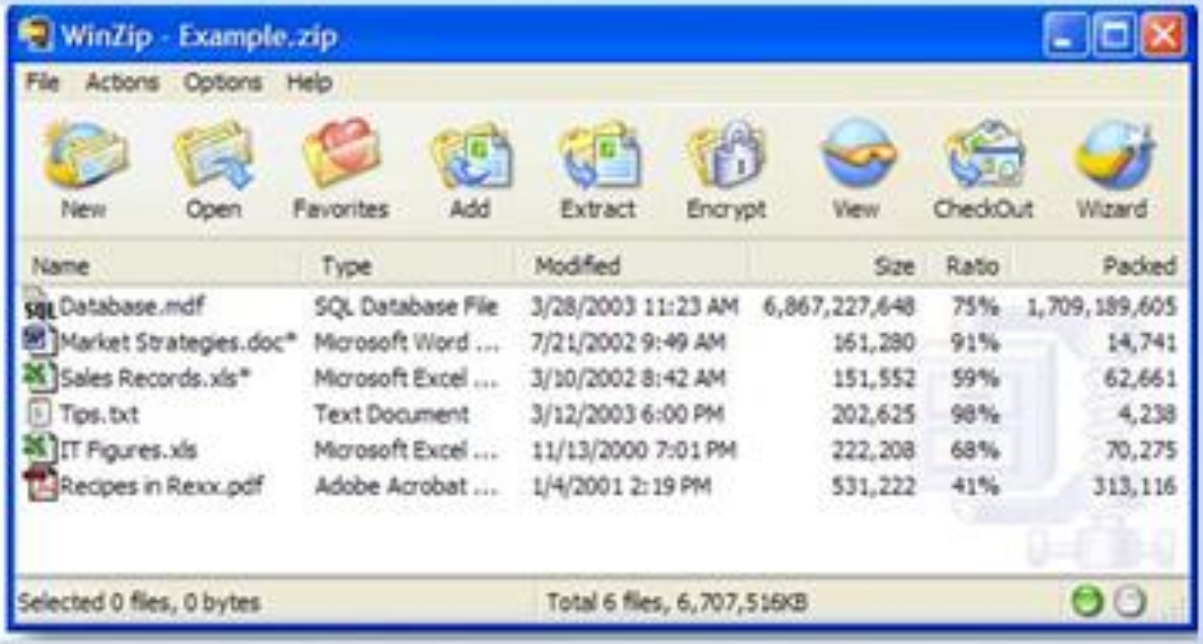
Стиснення даних у комп'ютерних інформаційних технологіях

WinRar



Стиснення даних у комп'ютерних інформаційних технологіях

WinZip



@ М.В.Добролюбова 16


Обчислювальна техніка та програмування
РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА.
ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Лекція 3
«Апаратне забезпечення ПК. Класифікація та основні характеристики ЕОМ»
© М.В.Добролюбова

Історія розвитку комп'ютерної техніки

Слово «комп'ютер» походить від англ. to compute – обчислювати, рахувати.

Лічильні машини докомп'ютерної ери

1632 р. – лічильна машина Вільгельма Шіккарда

1642 р. – лічильна машина Блеза Паскаля

Арифмометр Готфріда Вільгельма Лейбніца

Аналітична машина Чарльза Бебіджа

2

Історія розвитку комп'ютерної техніки

Електронні обчислювальні машини

Машина Тьюринга

МЭСМ – мала електронно-лічильна машина

БЭСМ – велика електронно-лічильна машина

© М.В.Добролюбова

3

Систематизація ЕОМ за основними показниками

Класифікація ЕОМ

1. За принципом дії

Цифрові, аналогові, гібридні обчислювальні машини.

2. За етапами створення і елементною базою, що використовується

1-е покоління: ЕОМ на електронних вакуумних лампах;

2-е покоління: ЕОМ на дискретних напівпровідникових приладах;

3-е покоління: ЕОМ на напівпровідникових інтегральних схемах з малим і середнім ступенем інтеграції;

4-е покоління: ЕОМ на великих і надвеликих інтегральних схемах – мікропроцесорах;

5-е покоління: ЕОМ з багатьма десятками паралельно працюючих мікропроцесорів, що дозволяють будувати ефективні системи обробки знань; ЕОМ на надскладних мікропроцесорах з паралельно-векторною структурою, одночасно виконуючих десятки послідовних команд програми;

6-е покоління і наступні покоління: оптоелектронні ЕОМ з масовим паралелізмом і нейронною структурою.

3. За призначенням

Універсальні, проблемно-орієнтовані, спеціалізовані ЕОМ.

4

Систематизація ЕОМ за основними показниками

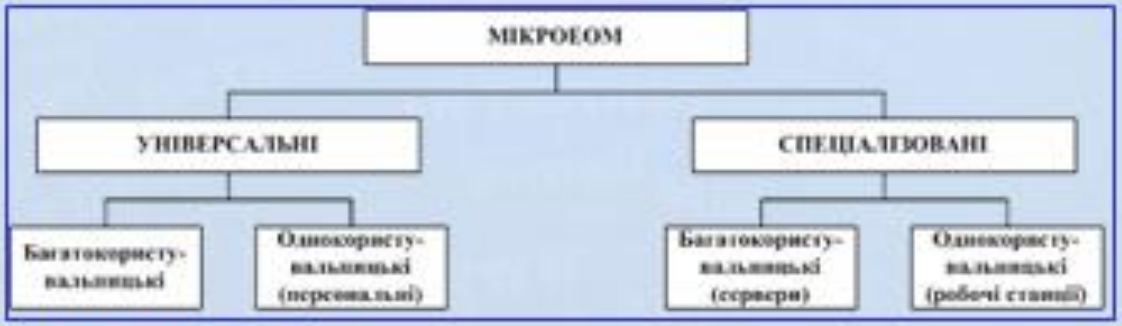
Класифікація ЕОМ

4. За кількістю обчислювальних пристроїв і ступенем розподілення

Автономні ЕОМ, обчислювальні системи, обчислювальні комплекси, обчислювальні мережі.

5. За розмірами і функціональними можливостями

СуперЕОМ, великі, малі і мікроЕОМ.



6. За кількістю процесорів

Однопроцесорні і багатопроцесорні ЕОМ.

7. За способом управління

ЕОМ, що керуються потоком інструкцій та ЕОМ, що керуються потоком даних.

@ М.В.Добролюбова 5

Основні технічні характеристики ЕОМ

1. Операційні ресурси

Операційні ресурси – це перелік дій (операцій), які може робити (виконувати) апаратура в плані обробки інформації (початкових даних).

Перелік дій:

- система машинних операцій – список $F=\{+,-,*,/,\ldots\}$;
- система машинних команд – $K=\{K_1,\ldots,K_N\}$;
- способи надання інформації в ЕОМ.

2. Об'єм пам'яті

Об'єм пам'яті – очевидна технічна характеристика, яка характеризує об'єм збереження програм і даних.

$E = 2m$, m – довжина адреси.

@ М.В.Добролюбова 6

Основні технічні характеристики ЕОМ

3. Швидкодія

Швидкодія – це характеристика, яка відповідає на питання, як швидко діє (працює) апаратура ЕОМ.

Швидкодія АЛП характеризує швидкість, з якою цей пристрій може виконувати операції: $V_{Алп} = \{V+, V-, V^*, V_{діл}, ...\}$.

Швидкодія визначається кількістю операцій в одиницю часу і залежить від часу виконання операції: $V=1/t$.

Швидкодія – це паспортна характеристика, вказується в документі на пристрій або у вигляді вектора швидкостей V, або у вигляді набору часу: t+, t-, t*, t/, ...

Різновиди швидкодії

1) Швидкодія процесора визначається часом виконання команд t_K:

$$t_K = t_{ВК} + t_{ВО} + t_{АЛП} + t_{ЗР},$$

де перший доданок визначає час вибірки команди з пам'яті, другий – час вибірки операнда(ів), третій – час виконання операції в АЛП, четвертий – час засилання результату операції.

Одиниці вимірювання швидкодії:

- МІПС (MIPS – Mega Instruction Per Second);
- МФЛОПС (MFLOPS – Mega Floating Operations Per Second);
- КОПС (KOPS – Kilo Operations Per Second);
- ГФЛОПС (GFLOPS – Giga FLoating Operations Per Second).

© М.Б.Добролюбова 7

Основні технічні характеристики ЕОМ

3. Швидкодія

Різновиди швидкодії

2) *Швидкодію пам'яті* прийнято характеризувати кількістю операцій зчитування/запису в одиницю часу.

3) *Продуктивність ЕОМ*

Швидкодія ЕОМ в цілому залежить від багатьох чинників: від швидкодії пристроїв, внутрішньої організації самого обчислювального комплексу, від операційної системи, під управлінням якої працює апаратура, тобто від організації обчислювальних процесів тощо.

Поняття «швидкодія» на ОК (ЕОМ) не розповсюджується.

Продуктивність ЕОМ **оцінюється кількістю задач в одиницю часу.**

Теорія масового обслуговування дозволяє отримати значення різних характеристик ОС:

- **часу розв'язання задач *Троз*;**
- **продуктивності ОС *Λ=1/Троз* ;**
- **завантаження процесора *ρ* ;**
- **часу очікування в черзі тощо.**

Час розв'язання конкретної задачі можна приблизно оцінити за формулою:

$$Троз = N(p1t1+p2t2+...),$$

де *N* – довжина програми (кількість команд), *p1, p2, ...* – ймовірності (частоти), а *t1, t2 ...* – часи виконання операцій.

@ М.В.Добролюбова **8**

Основні технічні характеристики ЕОМ

3. Швидкодія

Різновиди швидкодії

3) *Продуктивність ЕОМ*

Шляхи підвищення продуктивності

1) Вдосконалення технології виробництва ЕОМ («фізичний» шлях) – підвищення швидкодії логічних елементів.

2) Розпаралелювання обчислень – знаходження алгоритму рішення задачі, що використовує паралелізм, і реалізація цього алгоритму в ОС. ***Паралельні ОС*** – це багатопроцесорні ОС, в яких паралелізм використовується для підвищення продуктивності при вирішенні завдань за рахунок одночасного виконання різних операцій на різних процесорах або обробляючих пристроях одного процесора.

Проблеми розпаралелювання:

1. Розпаралелювання алгоритму (математична проблема).
2. Розпаралелювання обчислювальної структури (архітектурна і схемотехнічна, системна проблема).
3. Перенесення алгоритму на структуру.

3) Конвеєризація обчислень – розбиття обчислень на послідовні етапи з метою реалізації цих етапів на окремих сходинах конвеєра для підвищення продуктивності. ***Конвеєр*** – пристрій, що складається з N послідовно з'єднаних частин (сходин конвеєра), кожна з яких виконує черговий крок обчислень за час t (такт конвеєра).

Основні технічні характеристики ЕОМ

3. Швидкодія

Різновиди швидкодії

3) *Продуктивність ЕОМ*

Шляхи підвищення продуктивності

3) Конвейеризація обчислень

Час розв'язання однієї задачі на конвеєрі:

$$T_{роз} = N*t.$$

Продуктивність конвеєра:

$$P = N'/(T0 + Nt),$$

де N' – кількість завдань, що поступають на вхід конвеєра; $T0$ – час підготовки даних.

При конвейеризації виникають проблеми створення алгоритму, створення структури і перенесення алгоритму на структуру.

При цьому до алгоритму висувають вимоги: відсутність (мінімальна кількість) циклів і розвилок; можливість розбиття на кроки однакової тривалості (і складності); послідовне і рівномірне переміщення даних за алгоритмом тощо.

4) Спеціалізація обчислень дозволяє підвищити продуктивність за рахунок апаратної реалізації рішення задачі в цілому, або будь-якої складної частини цього завдання, тобто методом програмування апаратної структури.

5) Апаратна реалізація складних функцій (арифметичних, матричних операцій тощо).

4) *Тактова частота* – кількість синхронізуючих тактів, що надходять ззовні на вхід схеми за одну секунду.

@ М.В.Добролюбова 10

Основні технічні характеристики ЕОМ

4. Надійність ОК

Надійність ОК – це властивість ОК виконувати покладені на нього функції протягом заданого відрізку часу.

Характеристики надійності

Відмови апаратури – випадкові події, частоту яких прийнято характеризувати інтенсивністю відмов λ, тобто кількістю відмов в одиницю часу.

Напрацювання на відмову: $T = 1/\lambda$ – це проміжок часу між двома сусідніми (за часом) відмовами.

5. Вартість ОК

Вартість ОК – інтегральна характеристика, визначається всіма перерахованими характеристиками.

11


Обчислювальна техніка та програмування
РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА.
ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Лекція 4
«Апаратне забезпечення ПК. Основні складові ПК. Основні зовнішні пристрої»
© М.В.Добролюбова

Основні складові персонального комп'ютера, принципи їх побудови та роботи

Персональний комп'ютер – універсальна технічна система - електронна обчислювальна машина, що призначена для зберігання і переробки інформації, ціна, розміри та можливості якої задовольняють потреби багатьох людей.

Базова конфігурація ПК: системний блок, монітор, клавіатура, миша.

Переваги ПК:

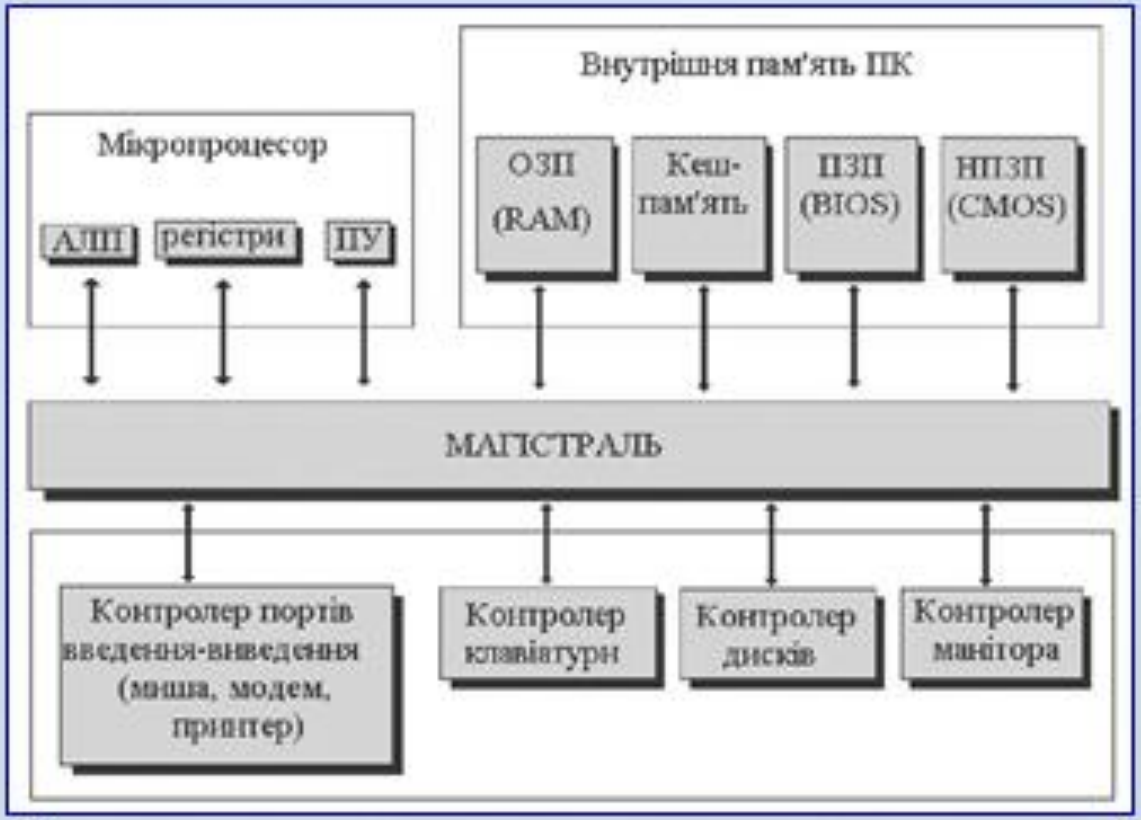
- мала вартість; автономність експлуатації без спеціальних вимог до умов навколишнього середовища;
- гнучкість архітектури;
- «дружність» операційної системи та іншого програмного забезпечення;
- висока надійність роботи.

@ М.В.Добролюбова

2

Основні складові персонального комп'ютера, принципи їх побудови та роботи

Структурна схема внутрішньої архітектури ПК



© М.В.Добролюбова 3

Основні складові ПК, принципи їх побудови та роботи

Системний блок

Параметри системного блока

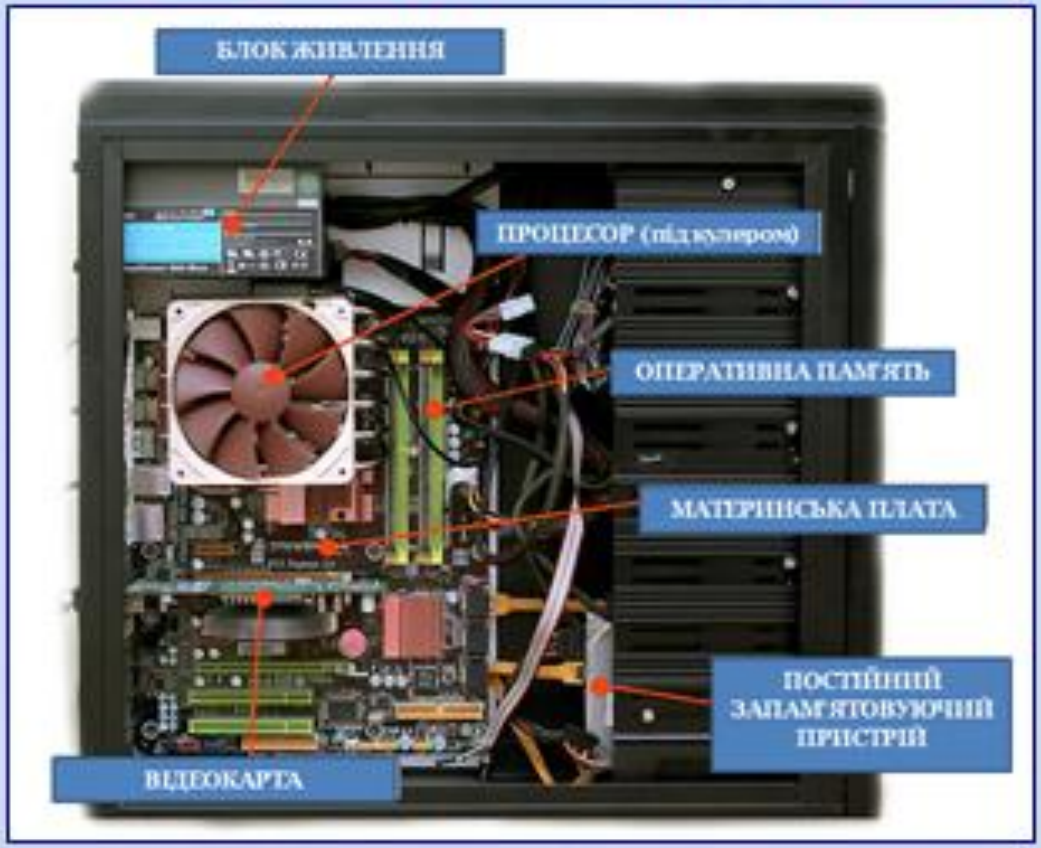
- форма корпусу;
- форм-фактор;
- потужність блоку живлення.

@ М.В.Добролюбова

4

Основні складові ПК, принципи їх побудови та роботи

Внутрішня будова системного блоку



© М.В.Добролюбова

5

Основні складові ПК, принципи їх побудови та роботи

Складові системного блоку

@ М.В.Добролюбова

6

Основні складові ПК, принципи їх побудови та роботи

Мікропроцесор

Мікропроцесор – це велика інтегральна схема, що представляє собою кремнієвий кристал у пластмасовому, керамічному або металокерамічному корпусі, на якому розташовані виводи для прийому і видачі електричних сигналів. Ступінь інтеграції інтегральної схеми визначається розміром кристала і кількістю розміщених у ньому транзисторів.

Основні функції мікропроцесора

- виконання обчислень;
- пересилання даних між внутрішніми регістрами;
- керування ходом обчислювального процесу.

Склад мікропроцесора

- АЛП (виконує арифметичні і логічні операції над даними);
- пристрій управління (виробляє керуючі сигнали для виконання команд);
- внутрішні регістри.

Етапи реалізації кожної машинної команди

- вибірка команди з пам'яті;
- декодування;
- виконання;
- запис результату.

Основні параметри процесорів

- робоча напруга;
- розрядність;
- робоча тактова частота;
- коефіцієнт внутрішнього множення тактової частоти;
- розмір кеш-пам'яті.

@ М.В.Добролюбова

7

Основні складові ПК, принципи їх побудови та роботи

Внутрішня (основна) пам'ять

Пристрої внутрішньої пам'яті виготовляють у вигляді мікросхем (модулів), які вставляються в спеціальні роз'єми на материнській платі.

Оперативна пам'ять призначається для зберігання програм і даних, з якими працює процесор у поточний момент.

Оперативний запам'ятовуючий пристрій — ОЗП (RAM — від англ. *Random Access Memory — пам'ять із довільним доступом*) — швидка та енергозалежна пам'ять. Оперативна пам'ять призначена для тимчасового зберігання вхідних даних, проміжних і кінцевих результатів обчислень, програм опрацювання даних. Це своєрідний робочий простір для комп'ютера. ОЗП може використовуватися як для читання даних, так і для записування. Дані в ОЗП зберігаються доти, поки на їх місце не буде записано нові дані. При вимкненні електроживлення дані в ОЗП втрачаються.

Оперативна пам'ять сучасних комп'ютерів має обсяги 128, 256, 512 МБ, 1024 МБ=1 ГБ і навіть сягає 4 ГБ , 8 ГБ, 16 ГБ та 32 ГБ. Від оперативної пам'яті не менеше, ніж від процесора, залежить продуктивність комп'ютера, його швидкодія (особливо під час роботи з графікою, анімацією, відео).

@ М.В.Добролюбова 8

Основні складові ПК, принципи їх побудови та роботи

Внутрішня (основна) пам'ять

Постійний запам'ятовувочай пристрій — ПЗП (ROM — від англ. *Read Only Memory — пам'ять тільки для читання*) — швидка та енергонезалежна пам'ять. Дані заносяться до неї один раз і назавжди (як правило, в заводських умовах) і зберігаються постійно (при ввімкненому й вимкненому живленні).

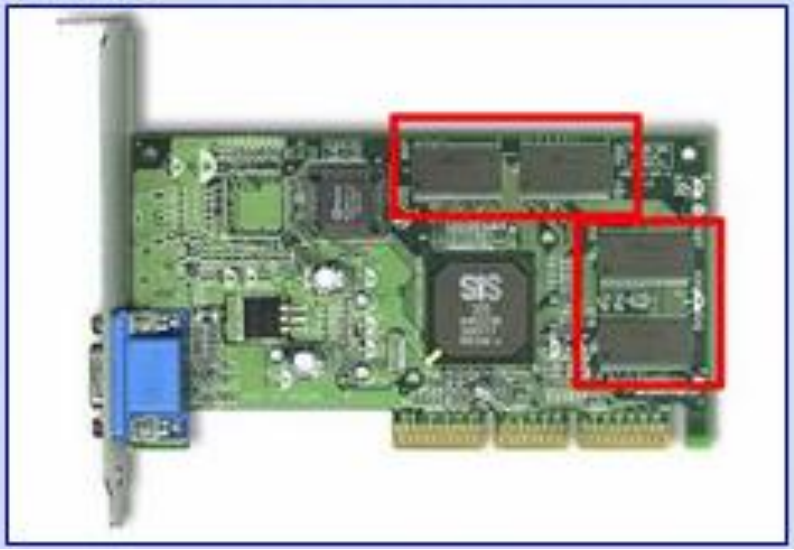
Постійна пам'ять — мікросхема, в якій містяться програми для керування роботою комп'ютера та програми тестування основних складових комп'ютера, а також набір програм для керування всіма його пристроями (BIOS — від англ. *Basic Input/Output System* — ***базова система введення/виведення***). Постійна пам'ять також розміщена на материнській платі.

@ М.В.Добролюбова 9

Основні складові ПК, принципи їх побудови та роботи

Внутрішня (основна) пам'ять

Дані, що зберігаються в ***напівпостійному програмованому запам'ятовуючому пристрої*** — НПЗП (пам'ять, виконана за технологією *CMOS* — від англ. *Complementary Metal-Oxide Semicondoctor* — технологія виготовлення мікросхем), можуть бути замінені у спеціальному режимі роботи комп'ютера — режимі програмування, коли користувач має спеціальні знання та може написати спеціальні програми для управління комп'ютером. До таких даних належать дані щодо зберігання та зміни конфігурації комп'ютера, календаря та годинника. НПЗП також називають ***пам'яттю автономного живлення*** або пам'яттю «на батарейках», оскільки дані зберігаються за допомогою акумуляторної батарейки, за своїми функціями подібної до батарейок кварцових годинників.



@ М.В.Добролюбова

10

Основні складові ПК, принципи їх побудови та роботи

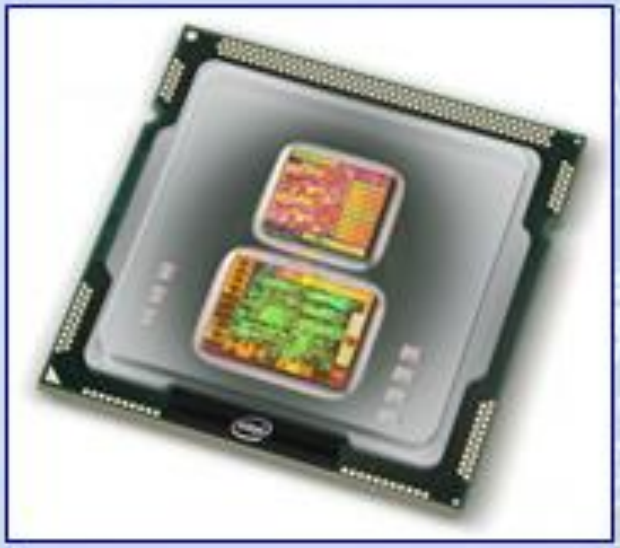
Внутрішня (основна) пам'ять

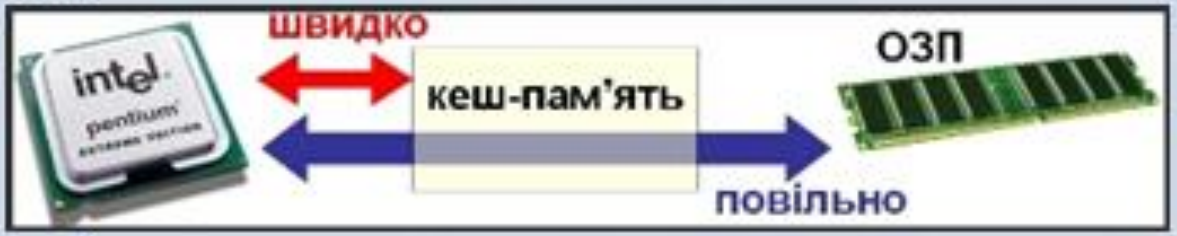
Відеопам'ять — швидка оперативна пам'ять для зберігання коду зображення, що відображається на екрані монітора. Відеопам'ять (VRAM — від англ. *Video Random Access Memory*) може бути різної ємності; розміщена на відеокарті. Найпродуктивніші відеокарти застосовують для комп'ютерних ігор або для роботи з просторовими зображеннями. Чим більша ємність відеопам'яті комп'ютера, тим більші можливості відображення на моніторі графіки з високою роздільною здатністю й великою кількістю кольорів. Ємність відеопам'яті сучасних комп'ютерів становить 64, 128, 256 МБ і більше.

Основні складові ПК, принципи їх побудови та роботи

Внутрішня (основна) пам'ять

Кеш-пам'ять (англ. *cache - запас, тайник*) — це спеціальний вид пам'яті або частини ОЗП, де зберігаються копії часто використовуваних даних. Кеш-пам'ять сучасних комп'ютерів має кілька рівнів: *кеш-пам'ять першого рівня* ємністю 32 КБ вбудовується в процесор, а *кеш-пам'ять другого рівня* ємністю до 2 МБ зазвичай розміщується на материнській платі. Час доступу до вмісту кеш-пам'яті в декілька разів менший, ніж до вмісту оперативної пам'яті, тому вона є «надоперативною». Кеш-пам'ять врівноважує швидкості роботи процесора й більш повільної оперативної пам'яті (*оскільки тактова частота роботи процесора значно вища за тактову частоту ОЗП, без кеш-пам'яті процесор "простоює" в очікуванні даних*).





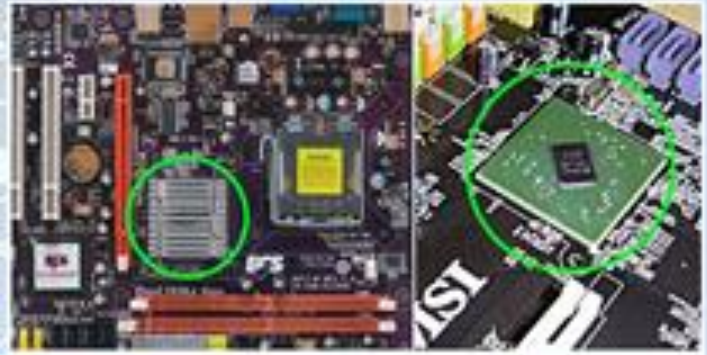
@ М.В.Добролюбова

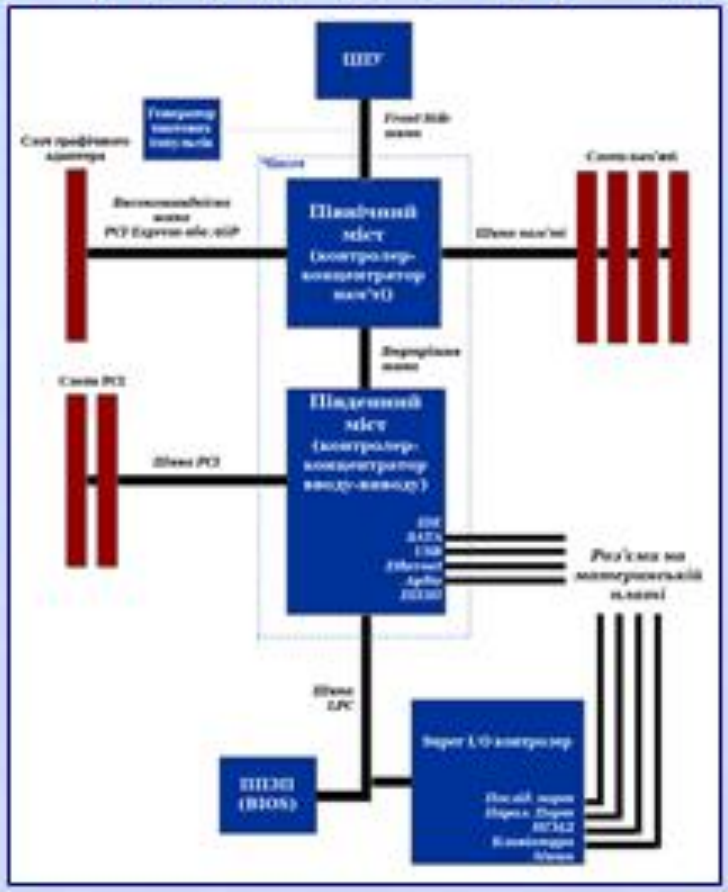
12

Основні складові ПК, принципи їх побудови та роботи

Мікропроцесорний комплект (чіпсет)

Чіпсет – набір мікросхем, що зв'язують пам'ять, процесор, відеоадаптер, пристрій вводу-виводу та інші елементи.





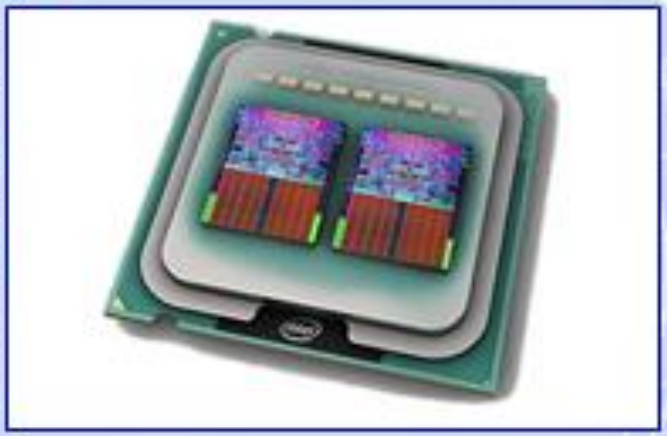
@ М.В.Добролюбова

13

Основні складові ПК, принципи їх побудови та роботи

Системна шина

Системна шина – це механізм, що дозволяє організовувати взаємодію різних підсистем. Це комп'ютерна шина, що з'єднує компоненти комп'ютерної системи, і поєднує функції ***шини даних*** для передавання інформації, ***шини адреси*** для визначення розміщення інформації, та ***шини керування*** для передачі команд. Технологія системної шини розвинулась для зменшення ціни та покращення модульності, і була популярною в 1970-ті та 1980-ті. Більшість сучасних комп'ютерів використовують набір окремих більш спеціалізованих шин.



Різновиди

Шина ISA, Шина MCA, Шина EISA, Шина VESA, Шина PCI, Шина AGP, PCI-Express, PC Card, Шина SCSI, Шина USB

@ М.В.Добролюбова

14

Основні складові ПК, принципи їх побудови та роботи

Зовнішня пам'ять

Зовнішні запам'ятовуючі пристрої розрізняють за: видом носія, типом конструкції, принципом запису та зчитування інформації, методом доступу тощо.

Носій – матеріальний об'єкт, здатний зберігати інформацію.

Диски – машинні носії інформації з прямим доступом.

Накопичувачі на дисках

- накопичувачі на гнучких магнітних дисках (НГМД), інакше, на флоппі-дисках або на дискетах (застарілі);
- накопичувачі на жорстких магнітних дисках (НЖМД) типу «вінчестер»;
- накопичувачі на змінних жорстких магнітних дисках, які використовують ефект Бернуллі;
- накопичувачі на оптичних компакт-дисках CD-ROM (Compact Disk ROM);
- USB-накопичувачі.

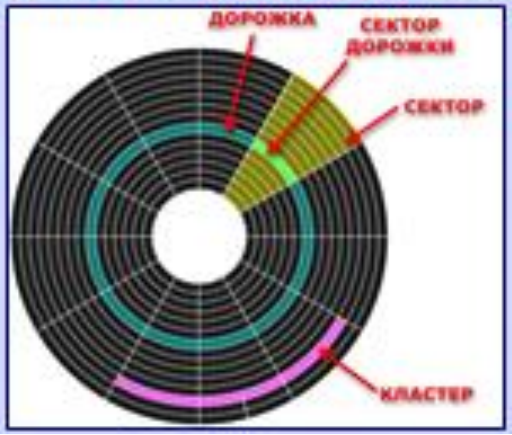
@ М.В.Добролюбова 15

Основні складові ПК, принципи їх побудови та роботи

Зовнішня пам'ять

Жорсткий диск, або жорсткий магнітний диск, або накопичувач на магнітних дисках (англ. *hard (magnetic) disk drive*, англ. *HDD*), у комп'ютерному сленгу — «*вінчестер*» (від маркування набоїв гвинтівки «Вінчестер») — магнітний диск, основа якого виконана з твердого матеріалу. У більшості ЕОМ виконує функцію енергонезалежного носія інформації (комп'ютерної пам'яті чи нагромаджувача інформації) з довільним доступом (англ. *random access*).

Логічна структура диску



Основні складові ПК, принципи їх побудови та роботи

Зовнішня пам'ять

Накопичувачі на гнучких магнітних дисках

Накопичувачі на змінних оптичних компакт-дисках

Флеш-накопичувачі

@ М.В.Добролюбова

17

Основні зовнішні пристрої

Клавіатура

Клавіатура – клавішний пристрій керування персональним комп'ютером. Служить для введення алфавітно-цифрових (знакових) даних, а також команд управління.

Типи клавіатур

Дротова клавіатура

Бездротова клавіатура

Сенсорна клавіатура

Кольорова клавіатура для дітей

© М.В.Добролюбова

18

Основні зовнішні пристрої

Миша

Миша – маніпулятор, що дозволяє оптимізувати роботу з великою категорією комп'ютерних програм. *За способом переміщення* миші діляться на: механічні, оптичні, лазерні, трекбол, індукційні, сенсорні. *За способом передачі даних в комп'ютер* миші діляться на дротові і бездротові.

© М.В.Добролюбова

19

Основні зовнішні пристрої

Відеотермінальні пристрої

Відеотермінал складається з *відеомонітора (дисплея)* і *відеоконтролера (адаптера)*. Видеоконтролер входять до складу системного блоку ПК (знаходяться на відеокарті, що встановлюється в роз'єм материнської плати), а відеомонітори – це зовнішні пристрої ПК.

Відеомонітор, дисплей або просто монітор – пристрій відображення текстової та графічної інформації на екрані. Класифікація за типом екрана: ЕПТ, рідкокристалічні, плазмові, проектори, LED-монітори, лазерні, віртуальні ретинальні.

Видеоконтролери *(відеоадаптери)* є внутрішньосистемними пристроями, безпосередньо керуючими моніторами і виведенням інформації на їх екран.

© М.В.Добролюбова

20

Основні зовнішні пристрої

Принтери

Принтери *(друкувальні пристрої)* – це пристрої виведення даних з ЕОМ, що перетворюють інформаційні ASCII-коди у відповідні їм графічні символи (літери, цифри, знаки і т.п.) і фіксують ці символи на папері. За принципом дії розрізняють матричні, лазерні, світлодіодні та струменеві принтери. Друк у принтерів може бути посимвольний, порядковий, посторінковий.

Матричні принтери

Струменеві принтери

Лазерні принтери

Світлодіодні принтери

@ М.В.Добролюбова

21

Основні зовнішні пристрої

Сканери

Сканер – пристрій оптичного введення інформації, її сканування, фотографування, що служить для копіювання зображень навколишньої дійсності в комп'ютер. Розрізняють: планшетні, рулонні (барабанні), книжкові, планетарні, ультразвукові, слайд-сканери, сканери штрих-коду, 3D-сканери, проєкційні сканери.

Планшетні сканери

Проєкційні сканери

Рулонні сканери

@ М.В.Добролюбова 22

Основні зовнішні пристрої

Пристрої введення даних

Клавіатура

Миша

Трекбол

Джойстик

Тачпед

@ М.В.Добролюбова

23

Основні зовнішні пристрої

Пристрої введення даних

Сканер

Мікрофон

Відеокамера

Електронна дошка

Графічний планшет

@ М.В.Добролюбова 24

Основні зовнішні пристрої

Пристрої виведення даних

Звукові колонки

Монітори

Принтери

Плотери

© М.В.Добролюбова

25

Основні зовнішні пристрої

Пристрої виведення даних

Навушники

Мультимедійні проектори

@ М.В.Добролюбова

26

Основні зовнішні пристрої

Пристрої збереження даних

Вінчестер

Оптичний диск

Флопі-диск

Флеш-пам'ять

@ М.В.Добролюбова

27

Основні зовнішні пристрої

Комунікаційні пристрої

— пристрої, за допомогою яких можна під'єднати комп'ютер до мереж, у яких відбувається передавання даних.

Модем

Мережева карта

@ М.В.Добролюбова

28


Обчислювальна техніка та програмування
РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА.
ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Лекція 5
«Програмне забезпечення ПК. Узагальнена класифікація програмного забезпечення»
© М.В.Добролюбова

Еволюція операційних систем

Пакетна обробка – режим виконання сукупності завдань, при якому всі завдання виконуються автоматично, без синхронізації з подіями за межами даної системи обробки інформації, зокрема без зв'язку з особами, які представили завдання для виконання. ОС з пакетною обробкою передбачають, що з програм, які підлягають виконанню формується пакет, який завантажується в керуючий комплекс і далі обробляється ОС і МПр. У цьому випадку користувачі безпосередньо з ОС в процесі виконання пакета не взаємодіють.

© М.В.Добролюбова 2

Еволюція операційних систем

Інтерактивний режим – режим взаємодії в процесі обробки інформації з людиною, що виражається у різного роду впливах на цей процес, що передбачені механізмом управления конкретної системи і які викликають відповідну реакцію процеса.

Поділ часу – процес обробки інформації, при якому ресурси системи обробки інформації надаються кожному процесу з групи процесів обробки інформації на інтервали часу тривалість і черговість надання яких визначаються управляючою програмою системи обробки інформації з метою забезпечення одночасної роботи процесів даної групи в інтерактивному режимі.

@ М.В.Добролюбова 3

Еволюція операційних систем

Класифікація операційних систем

@ М.В.Добролюбова

4


Еволюція операційних систем
Класифікація програмного забезпечення
Програмне забезпечення
Прикладне програмне забезпечення
Системне програмне забезпечення
Програми-автомати
Програми-інструменти
Службові програми (утиліти)
Операційна система
Редактори
Системи або мови програмування
Оболонка
Ядро
Навчальні програми
Бази знань
Ігри
Програми управління файлами і директоріями
Драйвери
текстові
графічні
СУБД
Прикладні програми (мобік)
Утиліти
Віруси
Операційні системи
@ М.В.Добролюбова
5


Обчислювальна техніка та програмування
РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА.
ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Лекція 6
«Програмне забезпечення ПК. Архітектура операційних систем. Файлова система»
@ М.В.Добролюбова

Поняття операційних систем

Операційна система – це комплекс управляючих та обробляючих програм, який, з одного боку, виступає як інтерфейс між користувачем та апаратними компонентами обчислювальних машин та обчислюваних систем, а з іншого боку призначений для ефективного управління обчислюваними процесами, а також найбільш раціонального розподілу та використання обчислювальних ресурсів.

Вимоги, що висуваються до операційної системи

1. Здатність виконання основних функцій – ефективного управління ресурсами та забезпечення зручного інтерфейсу для користувача та прикладних програм.
2. Розширюваність.
3. Перенесення.
4. Надійність та відмовостійкість.
5. Сумісність.
6. Безпечність.
7. Продуктивність.

Різновиди структур ОС

1. Монолітні системи.
2. Багаторівневі системи.
3. Мікроядра.
4. Клієнт-серверна модель.
5. Віртуальна машина.
6. Екзоядра.

@ М.В.Добролюбова 2


Поняття операційних систем
Види та компоненти операційної системи
Операційна система Windows
UNIX-подібні операційні системи
UNIX
KALI
ubuntu
Linux
Операційна система фірми Apple
Mac OS X
iOS
@ М.В.Добролюбова
3

Поняття операційних систем

Класифікація ОС

1. *Операційні системи мейнфреймів.*
2. *Серверні операційні системи.*
3. *Багатопроцесорні операційні системи.*
4. *Операційні системи персональних комп'ютерів.*
5. *Операційні системи кишенькових персональних комп'ютерів.*
6. *Вбудовані операційні системи.*
7. *Операційні системи сенсорних вузлів.*
8. *Операційні системи реального часу.*
9. *Операційні системи смарт-карт.*

@ М.В.Добролюбова 4

Поняття операційних систем

Види та компоненти операційної системи

Основні поняття операційної системи

Системний виклик – це інтерфейс між операційною системою та програмою користувача.

Системні виклики створюють, видаляють та використовують різні об'єкти, головні з яких процеси та файли. Програма користувача робить запит до операційної системи на сервіс, здійснюючи системний виклик.

Системні виклики ще називають *програмними перериваннями*.

Переривання – це подія, яка генерується зовнішніми (по відношенню до процесора) пристроями.

За допомогою апаратних переривань апаратура інформує центральний процесор про те, що виникла якась подія, яка потребує миттєвої реакції.

Виключна ситуація – це подія, яка виникла в результаті спроби виконання програмою неприпустимої команди, доступу до ресурсу при відстутності достатніх привілеїв або звернення до відсутньої сторінки пам'яті.

Виключні ситуації бувають виправним (після усунення їх причини програми продовжують працювати) и та невиправними (зазвичай виникають в результаті помилок в програмах).

Оболонка операційної системи – режим виконання сукупності завдань, при якому всі завдання виконуються автоматично, без синхронізації з подіями за межами даної системи обробки інформації, зокрема без зв'язку з особами, які представили завдання для виконання.

Система керування вікнами або ***віконний менеджер*** (windows manager) – розподіляє окремі блоки простору екрана, які називають вікнами, і відслідковує, який додаток асоціюється з кожним із цих вікон.

5

Поняття операційних систем

Види та компоненти операційної системи

Основні поняття операційної системи

Ядро – виконує основні функції ОС в процесі приведення ПК в робочий стан.

Система керування файлами або ***файлова система*** (file manager) координує використання запам'ятовуючих пристроїв.

Файл – іменований набір даних; блок інформації на запам'ятовуючому пристрої ПК, який має певне логічне надання, відповідні йому операції читання-запису та, як правило, фіксоване ім'я, що дозволяє отримати доступ до цього файлу та відрізнити його від інших.

Драйвери пристроїв (devices drivers) – елементи програмного забезпечення, що взаємодіють з контролерами пристроїв (або ж безпосередньо із пристроями) з метою виконання різних операцій у периферійних пристроях машини.

Система керування пам'яттю (memory manager) – вирішує завдання координації використання машиною її основної пам'яті.

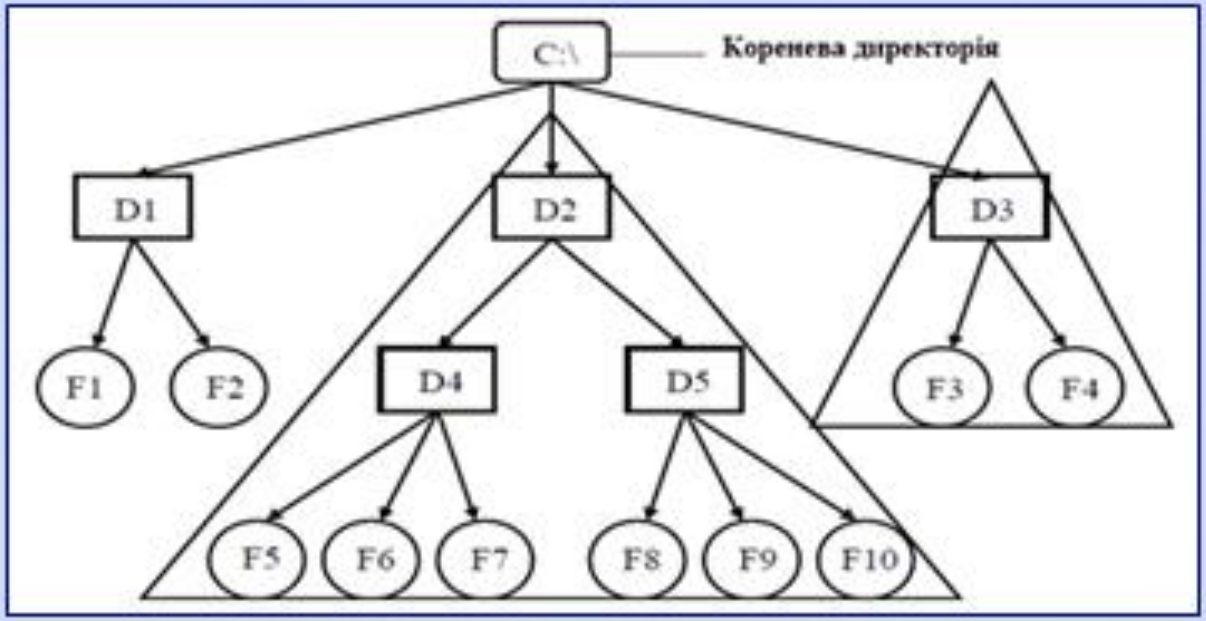
В системах з поділом часу ***планувальник*** (scheduler) визначає послідовність виконуваних дій, а ***диспетчер*** (dispatcher) контролює розподіл тимчасових квантів для них.

Процес – це програма в момент її виконання. З кожним процесом пов'язується його адресний простір – список адрес в пам'яті від деякого мінімуму (зазвичай нуля) до деякого максимуму, які процес може прочитати і в які він може писати.

@ М.В.Добролюбова 6

Поняття операційних систем

Файлова система

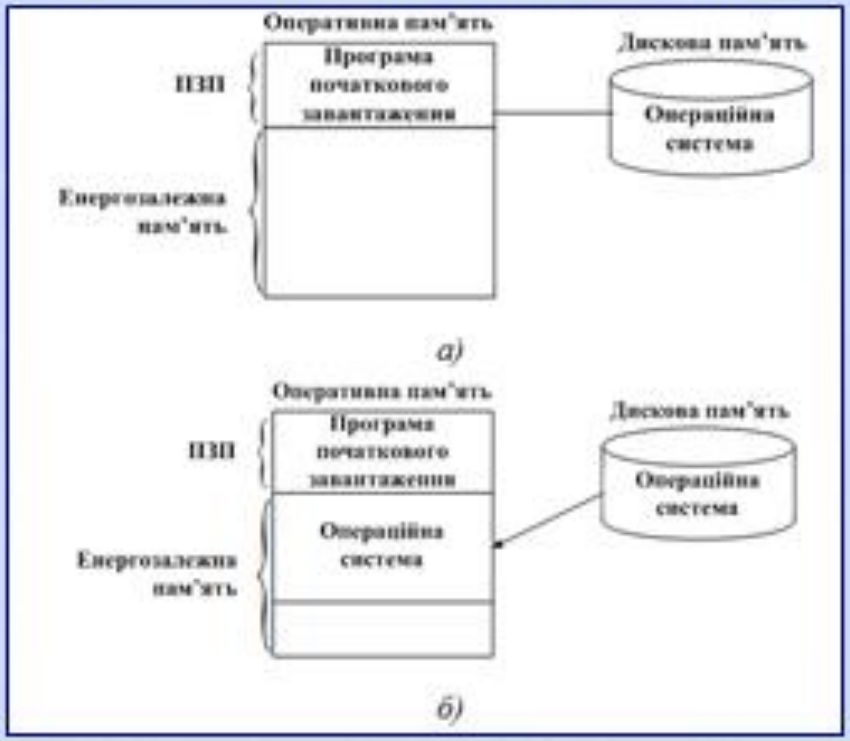


Дерево директорій

Тут кружки – файли, прямокутники – директорії, трикутники – приклади піддерев

Поняття операційних систем

Процес первісного завантаження



а – Етап 1: Машина починає виконувати програму початкового завантаження, що перебуває в пам'яті. Операційна система перебуває в масовій (зовнішній) пам'яті;

б – Етап 2: Програма початкового завантаження видає вказівку помістити операційну систему в оперативну пам'ять, а потім передає їй керування

@ М.В.Добролюбова 8


Обчислювальна техніка та програмування
РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ
Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Лекція 7
«Текстові процесори та електронні таблиці. Використання MS WORD при формуванні звітів в сфері науки і техніки»
© М.В.Добролюбова

Вимоги до порядку викладення матеріалів

Структура звіту (документу):

- вступна частина;
- основна частина;
- додатки.

Структурні елементи вступної частини:

- титульний аркуш;
- зміст;
- перелік умовних позначень.

Структурні елементи основної частини:

- вступ;
- суть звіту;
- висновки;
- перелік посилань.

Додатки розміщують після основної частини звіту.


Вимоги до порядку викладення матеріалів
ВСТУПНА ЧАСТИНА
Титульний аркуш
Зміст
Перелік умовних позначень
ОСНОВНА ЧАСТИНА
Вступ
Суть реферату
Висновки
Перелік посилань
Розділ
Підрозділ
Пункт
Підпункт
Ілюстрації і таблиці:
Ілюстрації
Рисунок 3.1
Рисунок 3.2
Рисунок 3.3
Таблиці
Таблиця 3.1
Таблиця 3.2
Таблиця 3.3
ДОДАТОК
Додаток А
Додаток Б
Ілюстрації і таблиці:
Ілюстрації
Рисунок Б.1
Рисунок Б.2
Рисунок Б.3
Таблиці
Таблиця Б.1
Таблиця Б.2
Таблиця Б.3
Структурна схема звіту
© М.В.Добролюбова
3

Правила оформлення

Загальні вимоги

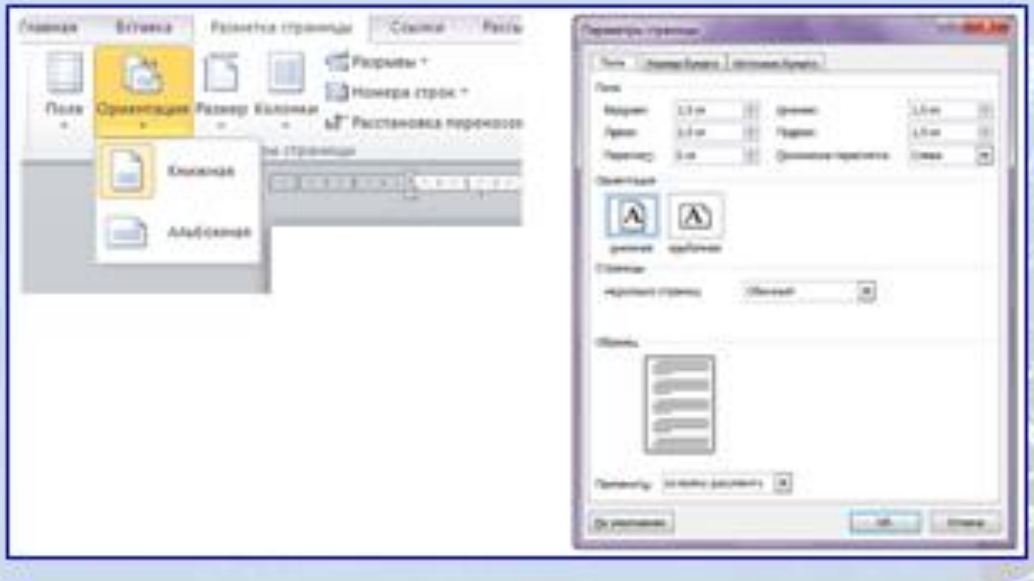
Встановлення формату аркушу:

п.м. Розмітка сторінки → вкл. Розмір → п. Інші розміри сторінки → д.о. Параметри сторінки → вкл. Розмір паперу.

Встановлення орієнтації аркушу:

- п.м. Розмітка сторінки → п. Орієнтація.
- п.м. Розмітка сторінки → д.о. Параметри сторінки → вкл. Поля → п. Орієнтація.

Формат аркушу



Орієнтація аркушу

@ М.В.Добролюбова

4

Правила оформлення

Загальні вимоги

Встановлення міжрядкового інтервалу:

п.м. Головна → вкл. Абзац → д.о. Абзац → вкл. Відступи і інтервали.

Встановлення розмірів берегів аркушу:

п.м. Розмітка сторінки → вкл. Розмір → п. Інші розміри сторінки → д.о. Параметри сторінки → вкл. Поля.

Встановлення абзацного відступу:

п.м. Головна → вкл. Абзац → д.о. Абзац → вкл. Відступи і інтервали.

Міжрядковий інтервал

Розмір берегів аркушу

Абзацний відступ

@ М.В.Добролюбова

5

Правила оформлення

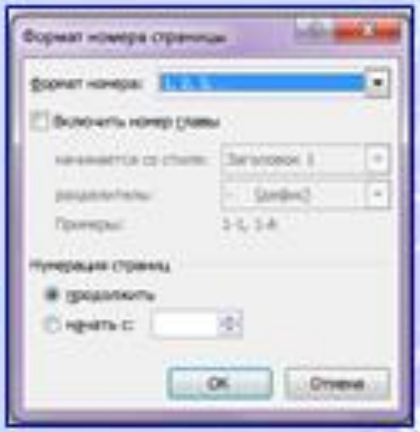
Загальні вимоги

Встановлення номеру сторінки:

п.м. Вставка → вкл. Номер сторінки → обрати місце розміщення номеру → вкл. Робота з колонтитулами, де можна налаштувати формат номера (вкл. Колонтитули → вкл. Номер сторінки → п. Номер сторінки.

Відміна видимості номеру сторінки на титульному аркуші:

вкл. Робота з колонтитулами → вкл. Парамети → встановити прапорець для п. Особливий колонтитул для першої сторінки.



Способи встановлення номеру сторінки та відміна видимості номеру на титульному аркуші

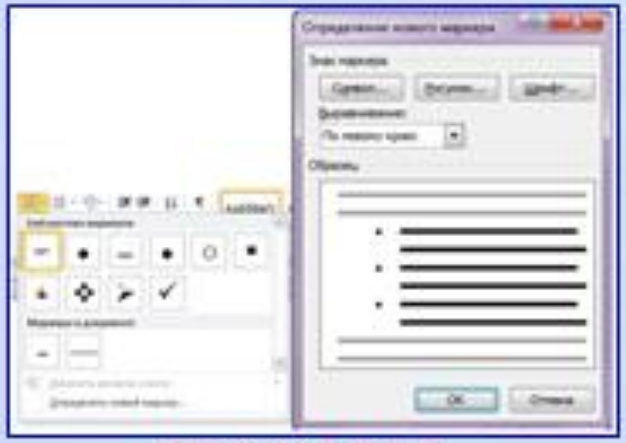
@ М.В.Добролюбова 6

Правила оформлення

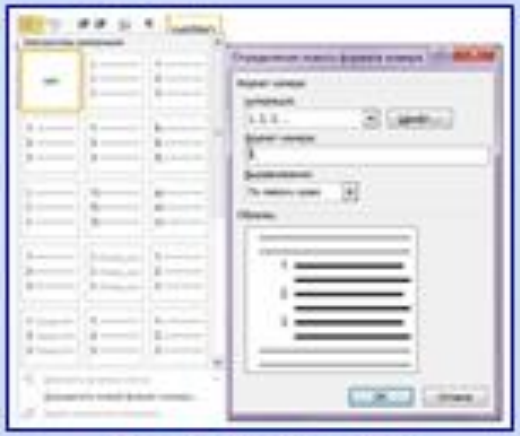
Загальні вимоги

Нумерація розділів, підрозділів, пунктів та підпунктів:

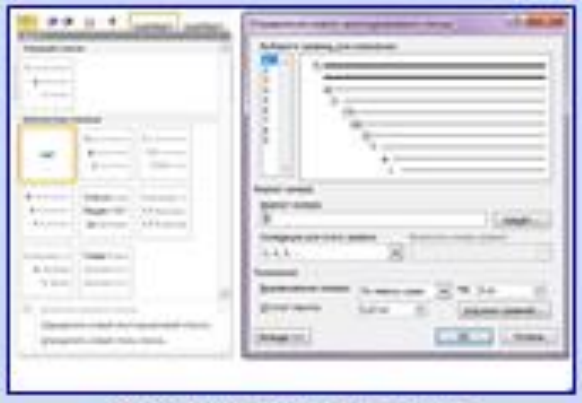
п.м. Головна → вкл. Маркований, вкл. Нумерований або вкл. Багаторівневий список.



Маркований список



Нумерований список



Багаторівневий список

@ М.В.Добролюбова

7


Правила оформлення
Загальні вимоги
Приклад оформлення ілюстрації
Оформлення таблиці
п.м. Вставка → вкл. Таблиці → Таблиця
Таблиця
Головка
Заголовки граф
Підзаголовки граф
Рядки (горизонтальні рядки)
Боковик (графа для заголовків рядків)
Графи (колонки)
Вставка
Разметка страницы
Таблица
Рисунок
Картинка
Фигуры
Вставка таблицы
Вставить таблицу...
Нарисовать таблицу
Преобразовать в таблицу...
Таблица Excel
Экспресс-таблицы
@ М.В.Добролюбова
8

Правила оформлення

Загальні вимоги

Оформлення переліку:

п.м. Головна → вкл. Маркированный, вкл. Нумерований або вкл. Багаторівневий список

а) форма і розмір клітин;
б) живий склад клітин:
1) частини клітин;
2) неживі включення протопластів;
в) утворення тканини.

Оформлення примітки:

Примітка._____

Примітки:
1._____
2._____

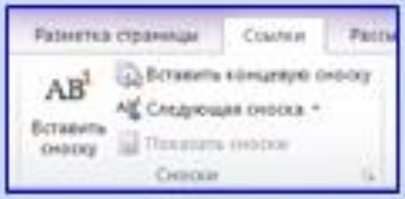
Оформлення виноски:

п.м. Посилання → вкл. Виноски → Вставити виноску.

Цитата в тексті: «Він базується на використанні інтуїтивного прогнозування за методом Дельфи[1)].

Відповідне подання виноски:

[1)]У стародавньому місті Дельфи жриці гадались у пророкуванні майбутнього. Метод, який назвали ім'ям цього міста, спочатку використовувався для «прорікання» часу настання подій, що прогнозувались. Він не допускає прямих контактів між експертами.



Оформлення посилання:

Цитата в тексті: «... у загальному обсязі робочого часу частка інформаційної роботи перевищує 70% [6][1]».

Відповідний опис у переліку посилань:

6. Автоматизація робіт в установах //ТІЕР. – № 4. – М: Мир, 1983. – С.66 - 76.

Відповідне подання виноски:

[1][6] Автоматизація робіт в установах // ТІЕР. – № 4. – М: Мир, 1983. – С.66 - 76.

@ М.В.Добролюбова

9

Правила оформлення

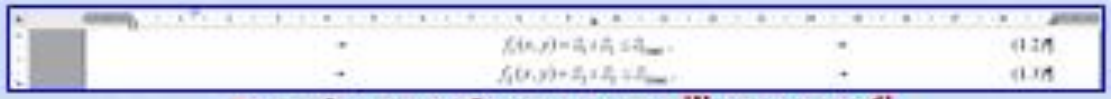
Загальні вимоги

Формули

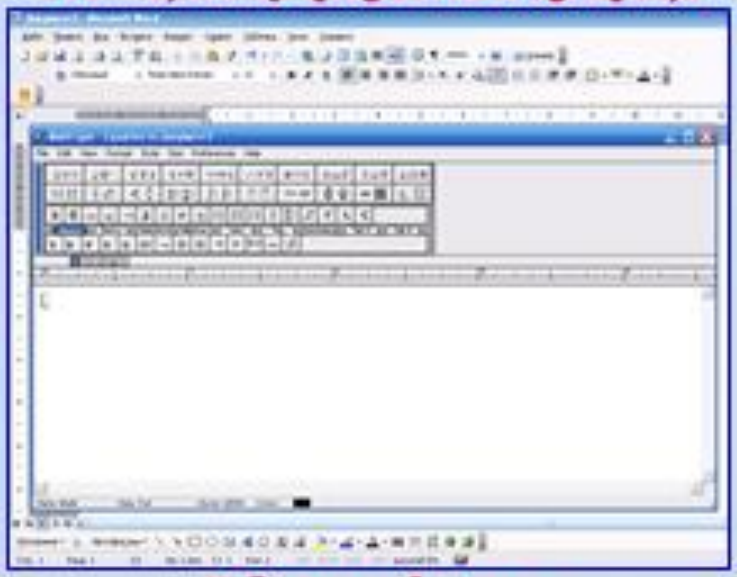
«Відомо, що

$$I = \frac{U}{Z}, \qquad (1.1)$$

де Z – комплексний опір послідовно включених нагрівача термоперетворювача і додаткового опору».



Розміщення формули та її нумерації



Редактор формул

@ М.В.Добролюбова

10

Правила оформлення

Приклад оформлення листа затвердження

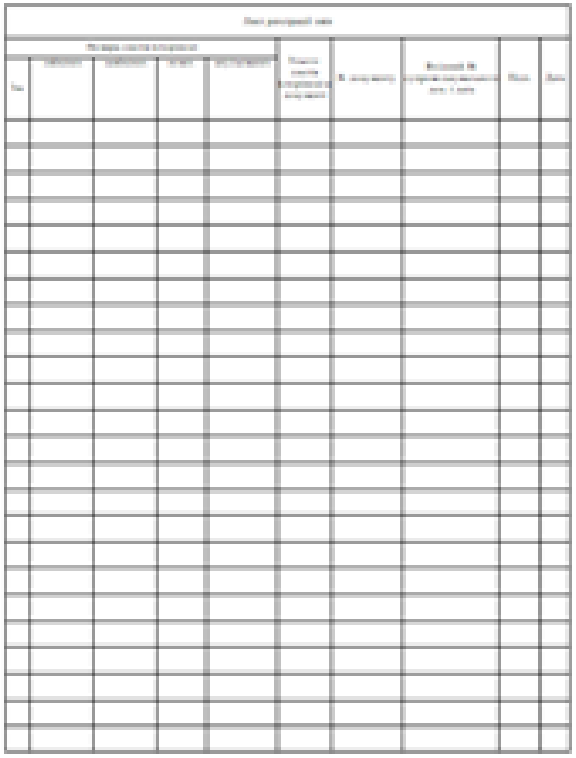
Приклад оформлення титульного листа

@ М.В.Добролюбова

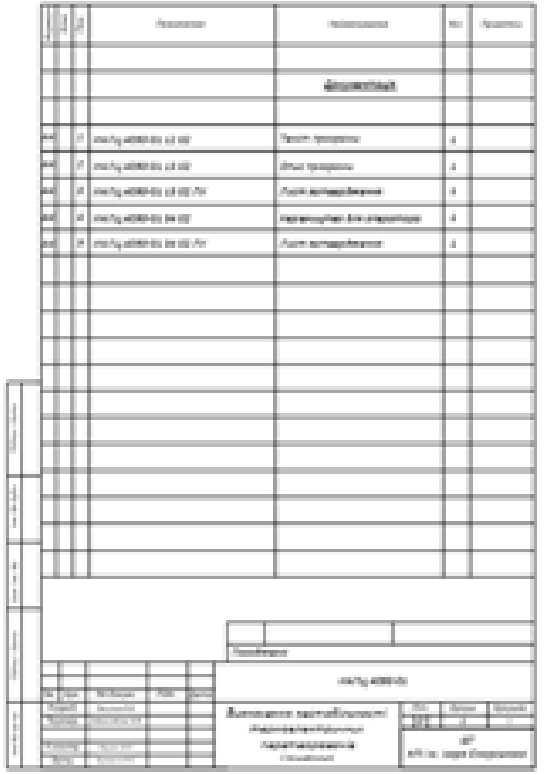
11

Правила оформлення

Приклад листа реєстрації змін



Приклад специфікації



12


Правила оформлення
Приклад додатку
ДОДАТОК А
ПРЕЗЕНТАЦІЇ ДО ЛЕКЦІЙ
@ М.В.Добролюбова
13


Обчислювальна техніка та програмування
РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ
Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Лекція 8
«Текстові процесори та електронні таблиці. Інструменти табличного процесора MS Excel»
© М.В.Добролюбова

Складання математичних та логічних виразів

Приклад

Скласти арифметичні вирази, які відповідають заданим математичним функціям.

$$y = (a+b)^2 \sqrt{\frac{a+x}{a+x^2}} \ln(a+x).$$

Арифметичний вираз для заданої математичної функції має наступний вигляд:

$y = (a+b)^2 * sqrt((a+x)/(a+x^2)) * ln(a+x)$.

Нехай a, b та a – константи, які розташовані в клітинках A1, B1, C1 відповідно, а x змінюється у діапазоні від 0 до 10 з інтервалом 1 і розташовано в клітинці D1, y – змінна, яка розташована в клітинці E1, тоді арифметичний вираз для значення y буде виглядати наступним чином:

=(A1+B1)^2*КОРЕНЬ((A1+D1)/(C1+D1^2))*LN(A1+D1).

@ М.В.Добролюбова

2

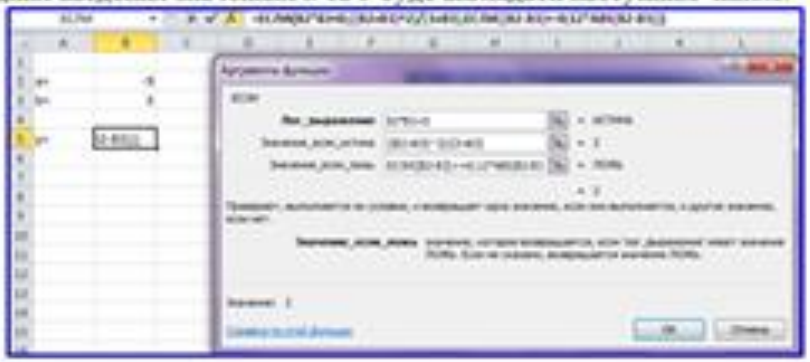
Складання математичних та логічних виразів

Приклад

Розробити програму обчислення величини

$$y = \begin{cases} (a+b)^2 / 3 + b, & \text{якщо } ab < 0; \\ 12 \cdot |a-b|, & \text{якщо } a-b \geq 0. \end{cases}$$

Нехай a та b – константи, які розташовані в клітинках B2, B3 відповідно, а y – змінна, яка розташована в клітинці B5, тоді логічний вираз при відповідних введених значеннях a та b буде виглядати наступним чином:



@ М.В.Добролюбова

3

Матриці. Розв'язок лінійних рівнянь

Приклад

На підприємство надійшла заявка на виготовлення 10 виробів трьох різних модифікацій. Загальні витрати на виготовлення цих виробів складають 4400 грн., а планується прибуток від їх реалізації повинен складати 420 грн. Визначити, скільки виробів кожної модифікації буде виготовлено на підприємстві, якщо відомо, що витрати на виготовлення одного виробу кожної модифікації складають відповідно 600, 400, 300 грн., а планується прибуток від реалізації одного виробу кожної модифікації дорівнює відповідно 30, 50, 40 грн.

Розв'язок

Позначимо через x_1, x_2, x_3 – кількість виробів кожної модифікації. Тоді $x_1 + x_2 + x_3$ – загальна кількість виробів, котра за умовою задачі дорівнює 10, т. т. отримуємо рівняння $x_1 + x_2 + x_3 = 10$.

Витрати підприємства на виробництво всіх виробів складають $600x_1 + 400x_2 + 300x_3$, а за умовою вони дорівнюють 4400.

Звідси маємо друге рівняння: $600x_1 + 400x_2 + 300x_3 = 4400$.

Плануємий прибуток підприємства від продажу всіх виробів дорівнює $30x_1 + 50x_2 + 40x_3$, а за умовою задачі він складає 420. Отримуємо третє рівняння: $30x_1 + 50x_2 + 40x_3 = 420$.

Таким чином, розв'язок задачі зводиться до розв'язку системи лінійних алгебраїчних рівнянь

$$\begin{cases} x_1 + x_2 + x_3 = 10 \\ 600x_1 + 400x_2 + 300x_3 = 4400 \\ 30x_1 + 50x_2 + 40x_3 = 420 \end{cases} \quad \text{або} \quad \begin{cases} x_1 + x_2 + x_3 = 10 \\ 6x_1 + 4x_2 + 3x_3 = 44, \\ 3x_1 + 5x_2 + 4x_3 = 42 \end{cases}$$

@ М.В.Добролюбова 4

Матриці. Розв'язок лінійних рівнянь

Для даної системи матимемо

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 6 & 4 & 3 \\ 3 & 5 & 4 \end{pmatrix}; \qquad B = \begin{pmatrix} 10 \\ 44 \\ 42 \end{pmatrix}; \qquad X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

Знайдемо розв'язок цієї системи, використав можливості Excel:

1. Вводимо дані: розмістимо матрицю A в діапазоні клітинок B2:D4, а стовпець вільних членів B – в діапазоні клітинок B6:B8.
2. Обчислимо визначник матриці A за допомогою вбудованої функції ***МОПРЕД***. Якщо він не дорівнює нулеві, то процес розв'язку системи продовжується.

При обчисленні визначника матриці необхідно:

- виокремити клітинку F7, де буде знаходитись значення визначника матриці;
- активізувати *Майстер функцій* (f_x). Серед категорій функцій вибрати *Математичні*, а серед функцій – ***МОПРЕД***. Далі вводимо діапазон клітинок, де розміщена матриця A, т. т. діапазон клітинок B2:D4,

3. Обчислюємо обернену матрицю A^{-1} за допомогою вбудованої функції ***МОБР***. При цьому необхідно:

- виокремити діапазон клітинок B10:D12, де буде знаходитись обернена матриця;
- активізувати *Майстер функцій* (f_x). Серед категорій функцій *Математичні* вибрати ***МОБР***,
- для того, щоб отримати на екрані обернену матрицю натиснути та відпустити клавішу **F2**, а потім натиснути **Ctrl+Shift+Enter**.

4. Знайти розв'язок системи рівнянь за допомогою функції множення матриць – ***МУМНОЖ***. Процедура обчислення добутку матриць на вектор аналогічна процедурі визначення оберненої матриці:

- виокремлюється діапазон клітинок F14:F16, де буде знаходитись розв'язок системи рівнянь;
- активізується *Майстер функцій* (f_x). Серед категорій функцій *Математичні* обирається ***МУМНОЖ***. Далі вводяться діапазони клітинок масивів добуток котрих обчислюється, т. т. діапазони клітинок B10:D12 та B6:B8;
- для того, щоб отримати на екрані обернену матрицю натискається та відпускається клавіша **F2**, а потім натискається комбінація **Ctrl+Shift+Enter**.

@ М.В.Добролюбова 5

Матриці. Розв'язок лінійних рівнянь

Таким чином, на виробництві планується виготовити замовлення в кількості 3, 5 і 2 виробів трьох модифікацій відповідно. На рисунку 1 відображено результат розв'язку, а на рисунку 2 – та сама таблиця в режимі показу формул, який можна визвати через меню *Формули → Показати формули*.

Рисунок 1 – Виведення результату розрахунків

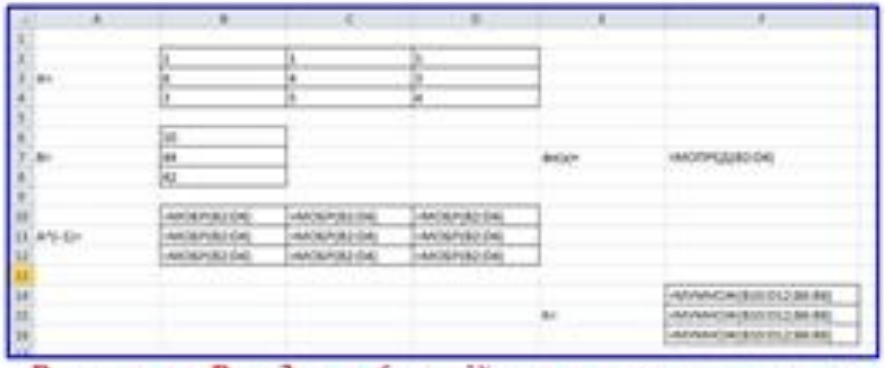


Рисунок 2 – Виведення функцій, що використовуються

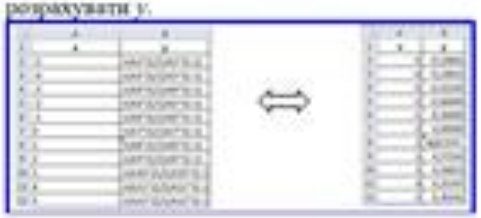
@ М.В.Добролюбова 6

Побудова графіків

Приклад

Побудувати графік $y = x^2/x^3 - 1$.

1. В наданій формулі x та y змінні, тому необхідно задати діапазон значень для x та розрахувати y.



2. Для побудови діаграми перейти до меню *Вставка* → *Графік*. Вибрати *Тип діаграми* (Графік).

3. Перейти до пункту меню *Вибрати дані*. У діалоговому вікні *Вибір джерела даних* зробити відповідні налаштування. Ряд 1 відповідає розрахованим значенням y.

Зверніть увагу, що за замовчуванням підписи горизонтальної осі починаються з 1! За завданням значення x у нас змінюються від -5 до 5 з кроком 1, тому необхідно натиснути кнопку *Змінити* та вибрати діапазон A2:A12.

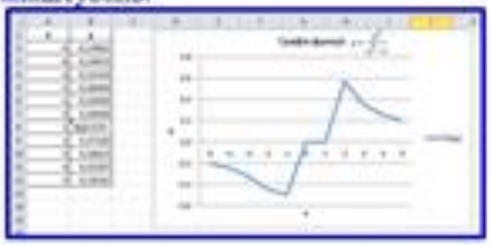
@ М.В.Добролюбова

7

Побудова графіків

4. Також можна вибрати *Макет діаграми* на вкладинці *Вставка* та оформити графік, користуючись контекстним меню та обираючі відповідні пункти. Так, наприклад, значення осі ординат мають 5 знаків після коми. Для зменшення їх кількості можна використати пункт контекстного меню *Формат осі*, виділивши перед цим вісь *y*.

5. Отриманий графік функції після всіх налаштувань.



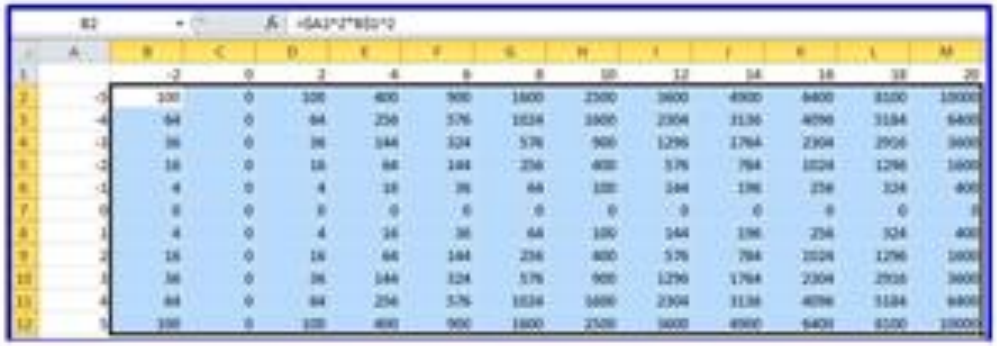
@ М.В.Добролюбова

8

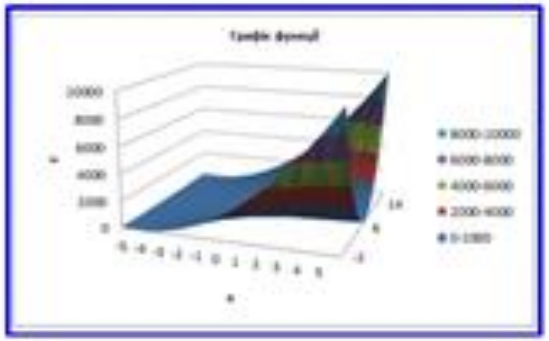
Побудова графіків

Приклад

Побудуємо графік функції $z = x^2 * y^2$, де x змінюється в діапазоні від –5 до 5 з інтервалом 1, а y від – 2 до 20 з інтервалом 2. В клітинках A2 – A12 необхідно помістити діапазон для x. В клітинках B1 – M1 необхідно помістити діапазон для y. Значення для $z1$, яке знаходиться в клітинці B2, розраховується за формулою =$A2^2*B$1^2. Після обчислення всіх значень z отримуємо наступний графік:



	A	B	C	D	E	F	G	H	I	J	K	L	M
1		-2	0	2	4	6	8	10	12	14	16	18	20
2	-5	100	0	100	400	900	1600	2500	3600	4900	6400	8100	10000
3	-4	64	0	64	256	576	1024	1600	2304	3136	4096	5184	6400
4	-3	36	0	36	144	324	576	900	1296	1764	2304	2916	3600
5	-2	16	0	16	64	144	256	400	576	784	1024	1296	1600
6	-1	4	0	4	16	36	64	100	144	196	256	324	400
7	0	0	0	0	0	0	0	0	0	0	0	0	0
8	1	4	0	4	16	36	64	100	144	196	256	324	400
9	2	16	0	16	64	144	256	400	576	784	1024	1296	1600
10	3	36	0	36	144	324	576	900	1296	1764	2304	2916	3600
11	4	64	0	64	256	576	1024	1600	2304	3136	4096	5184	6400
12	5	100	0	100	400	900	1600	2500	3600	4900	6400	8100	10000



@ М.В.Добролюбова

9

Створення та робота з таблицею

Приклад

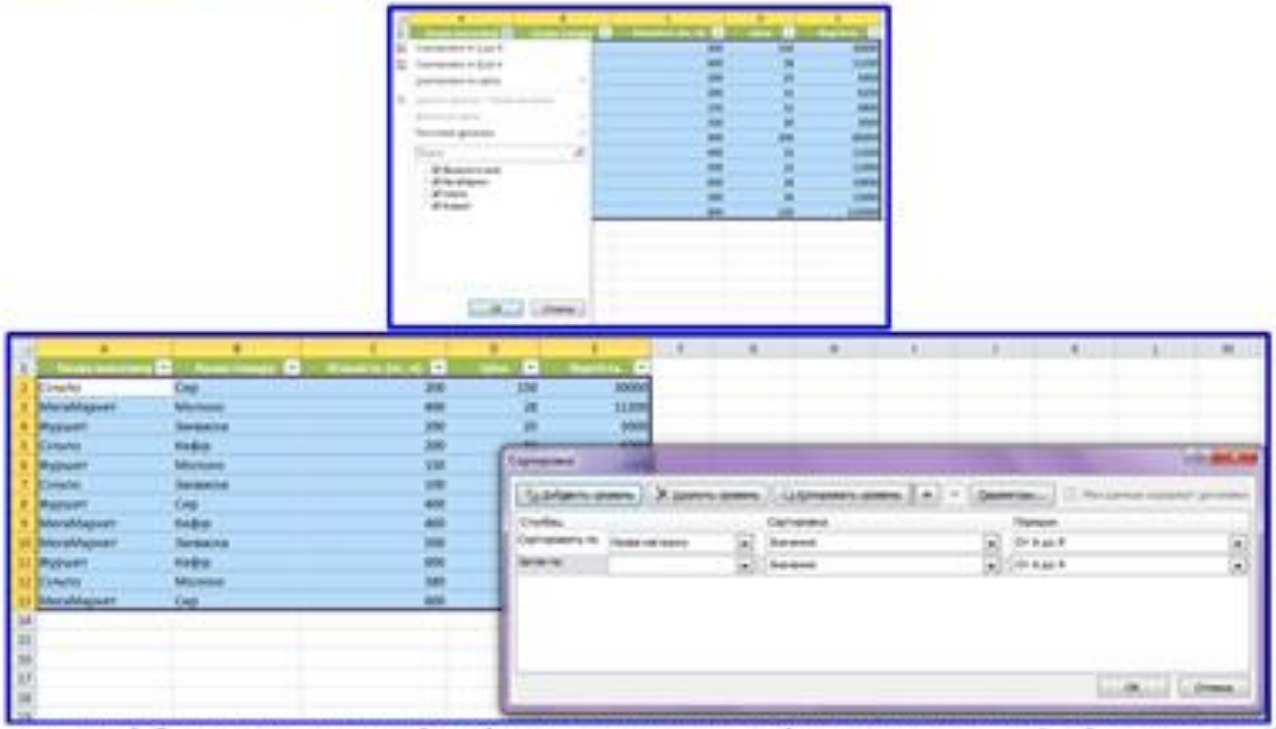
Створення таблиці

Створити таблицю продажів товарів з 20 рядків і п'яти стовпців: «Назва магазину», «Назва товару», «Кількість», «Ціна», «Вартість». Перші чотири стовпці заповнити довільними даними, а значення стовпця «Вартість» розрахувати за формулою. За допомогою розширеного фільтру визначити рядки попередньої таблиці, в яких кількість продажів перевищує 200 кг(л) або вартість не перевищує 100 грн. Впорядкувати таблицю за назвами товарів і за допомогою функції підбиття підсумків, визначити загальну вартість товарів кожного виду. На основі двох таблиць, які містять дані щодо продажу товарів за березень і травень поточного року, створити консолідовану таблицю, яка б містила інформацію про кількість проданих за два місяці товарів. На основі першої таблиці створити зведену таблицю, де визначити загальну кількість товарів кожного виду, які були продані в кожному магазині.

@ М.В.Добролюбова 10

Створення та робота з таблицею

Сортування поля таблиці



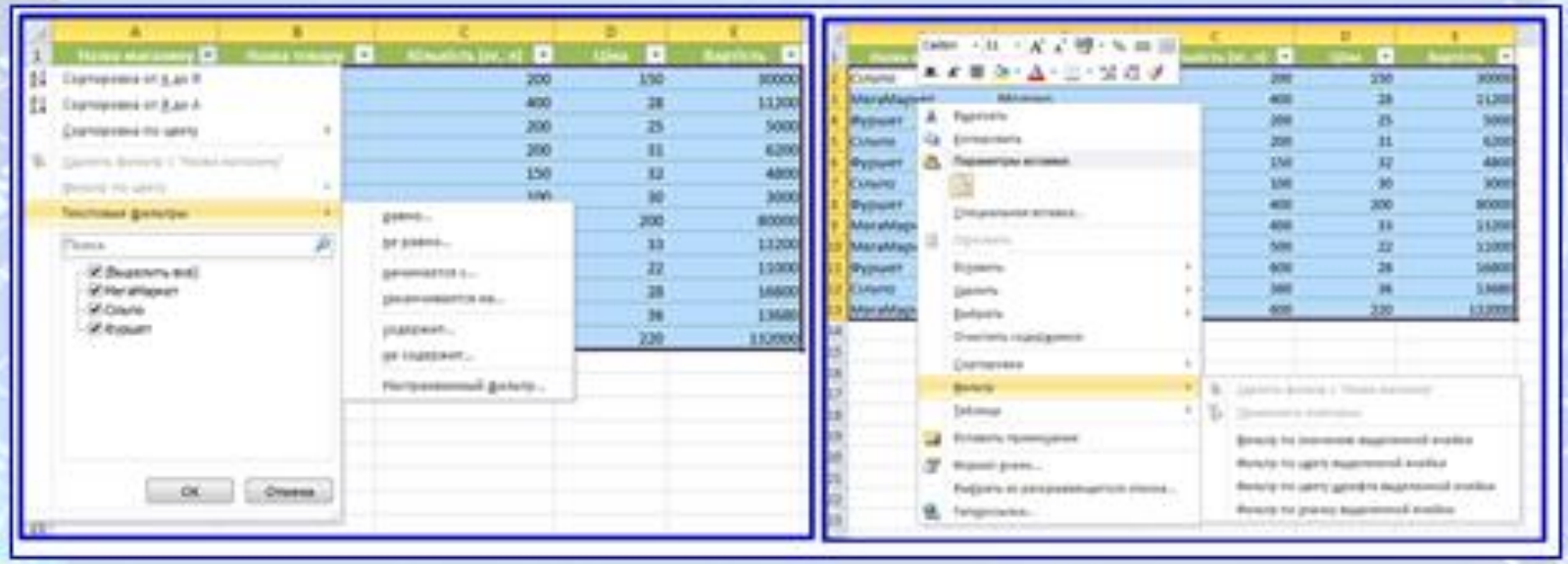
Коли сортування відбулося, в рядку назв відповідне поле позначається символом сортування (стрілка вверх вниз).

	A	B
1	Назва магазину	Назва товару
2	МегаМаркет	Молоко
3	МегаМаркет	Кефір
4	МегаМаркет	Заквасна
5	МегаМаркет	Сир

@ М.В.Добролюбова 11

Створення та робота з таблицею

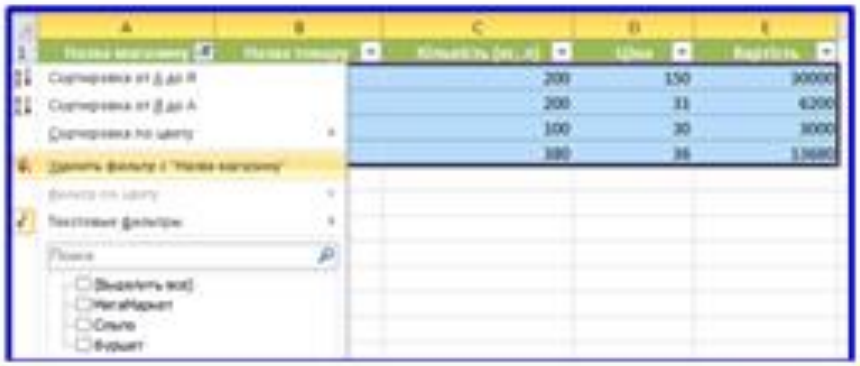
Фільтрація



Створення та робота з таблицею

Автофільтр або простий фільтр

	A	B	C	D	E
1	Назва магазину	Назва товару	Кількість (кг, л)	Ціна	Вартість
2	Сільпо	Сир	200	150	30000
5	Сільпо	Кефір	200	31	6200
7	Сільпо	Закваска	100	30	3000
12	Сільпо	Молоко	380	36	13680



@ М.В.Добролюбова

13

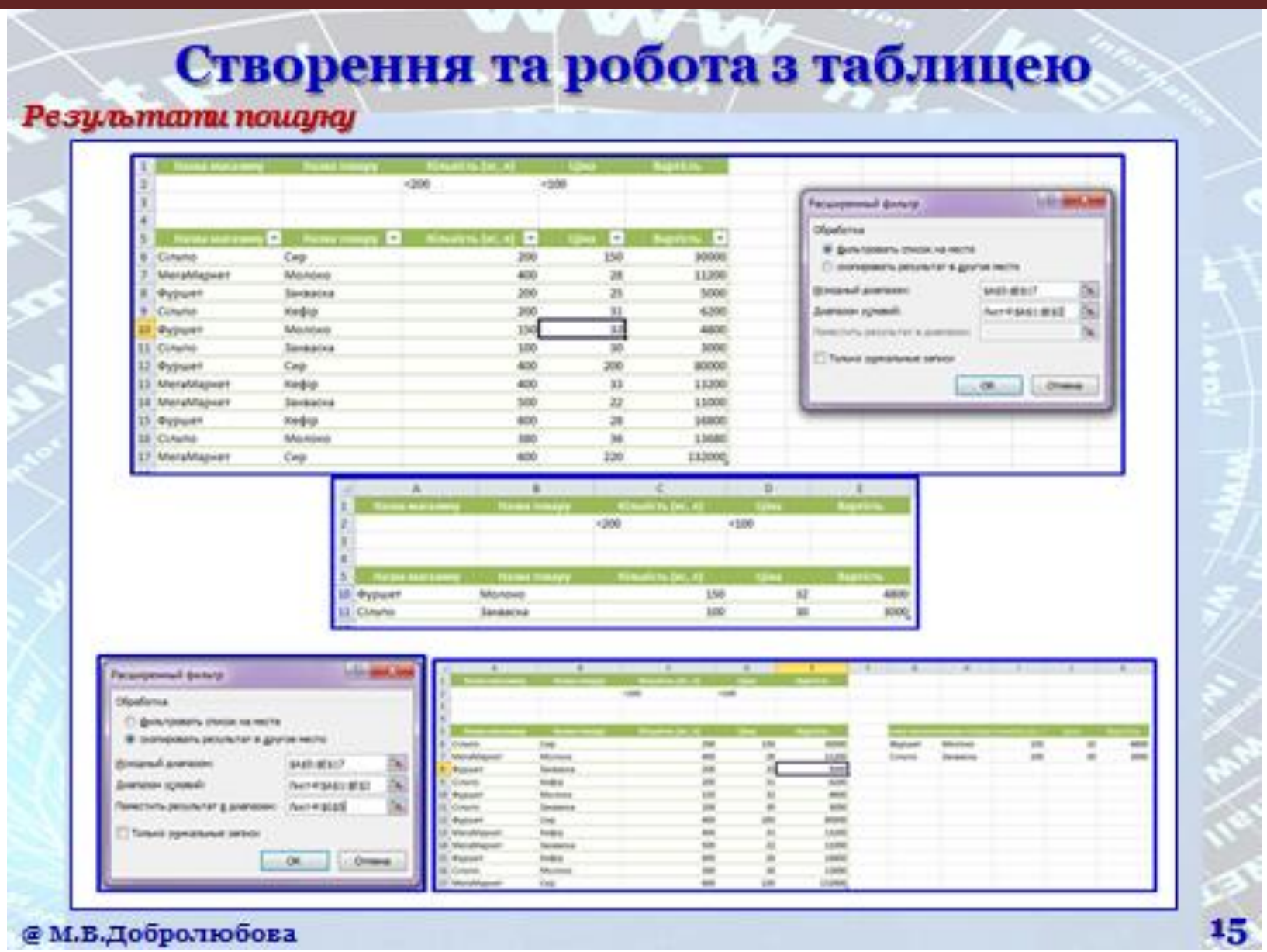
Створення та робота з таблицею

Розширений фільтр

Розширена фільтрація за допомогою діапазонів критеріїв

	Назва магазину	Назва товару	Кількість (кг, л)	Ціна	Вартість
2			<200	<100	
3					
4					
5	Назва магазину	Назва товару	Кількість (кг, л)	Ціна	Вартість
6	Сільпо	Сир	200	150	30000
7	МегаМаркет	Молоко	400	28	11200
8	Фуршет	Закваска	200	25	5000
9	Сільпо	Кефір	200	31	6200
10	Фуршет	Молоко	150	32	4800
11	Сільпо	Закваска	100	30	3000
12	Фуршет	Сир	400	200	80000
13	МегаМаркет	Кефір	400	33	13200
14	МегаМаркет	Закваска	500	22	11000
15	Фуршет	Кефір	600	28	16800
16	Сільпо	Молоко	380	36	13680
17	МегаМаркет	Сир	600	220	132000

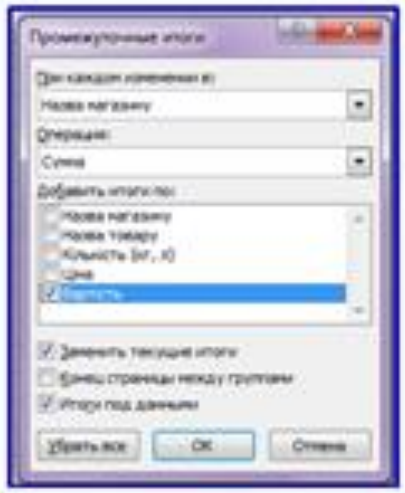
14


Створення та робота з таблицею
Результати пошуку
@ М.В.Добролюбова
15

Створення та робота з таблицею

Проміжні підсумки

5	Назва магазину	Назва товару	Кількість (кг, л)	Ціна	Вартість
6	Фуршет	Закваска	200	25	5000
7	Сільпо	Закваска	100	30	3000
8	МегаМаркет	Закваска	500	22	11000
9	Сільпо	Кефір	200	31	6200
10	МегаМаркет	Кефір	400	33	13200
11	Фуршет	Кефір	600	28	16800
12	МегаМаркет	Молоко	400	28	11200
13	Фуршет	Молоко	150	32	4800
14	Сільпо	Молоко	380	36	13680
15	Сільпо	Сир	200	150	30000
16	Фуршет	Сир	400	200	80000
17	МегаМаркет	Сир	600	220	132000



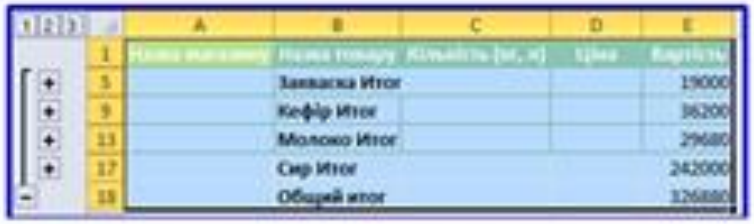
© М.В.Добролюбова

16

Створення та робота з таблицею

Результат

	A	B	C	D	E
1	Назва магазину	Назва товару	Кількість (кг, л)	Ціна	Вартість
2	Фуршет	Закваска	200	25	5000
3	Сільпо	Закваска	100	30	3000
4	МегаМаркет	Закваска	500	22	11000
5		**Закваска Итог**			19000
6	Сільпо	Кефір	200	31	6200
7	МегаМаркет	Кефір	400	33	13200
8	Фуршет	Кефір	600	28	16800
9		**Кефір Итог**			36200
10	МегаМаркет	Молоко	400	28	11200
11	Фуршет	Молоко	150	32	4800
12	Сільпо	Молоко	380	36	13680
13		**Молоко Итог**			29680
14	Сільпо	Сир	200	150	30000
15	Фуршет	Сир	400	200	80000
16	МегаМаркет	Сир	600	220	132000
17		**Сир Итог**			242000
18		**Общий итог**			326880

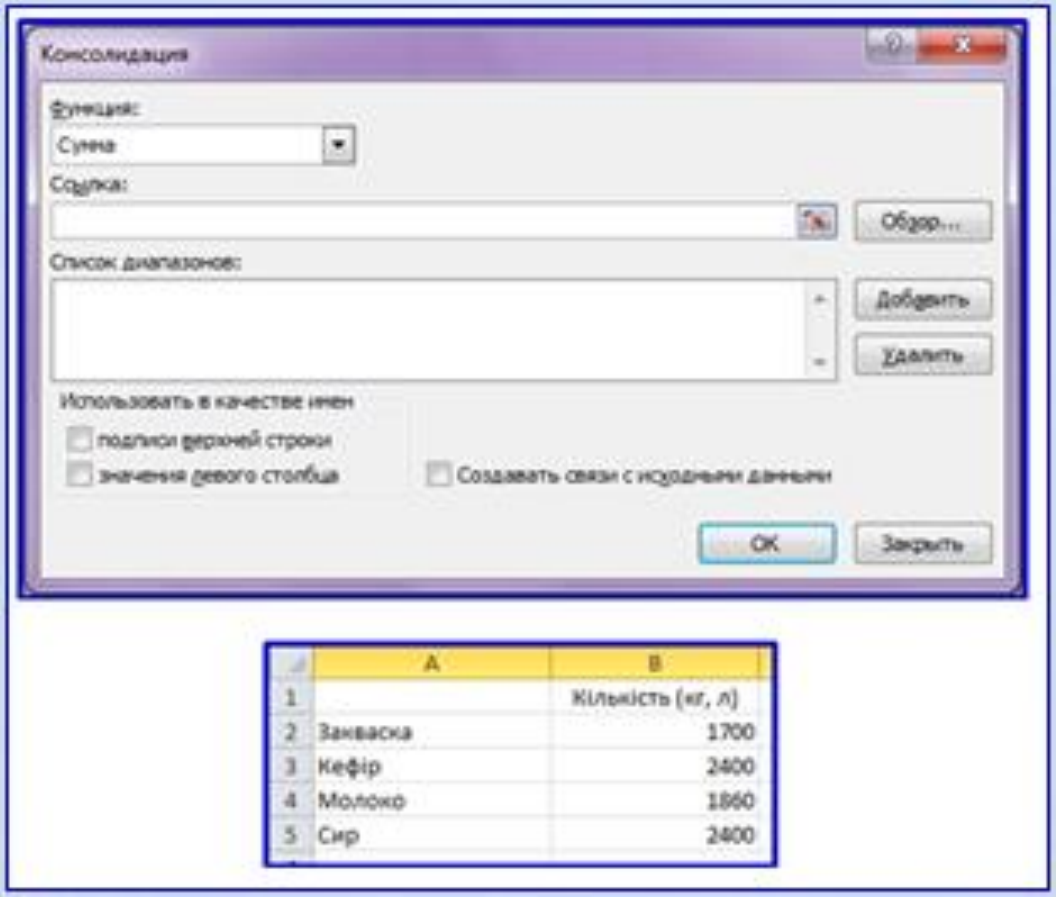


@ М.В.Добролюбова

17

Створення та робота з таблицею

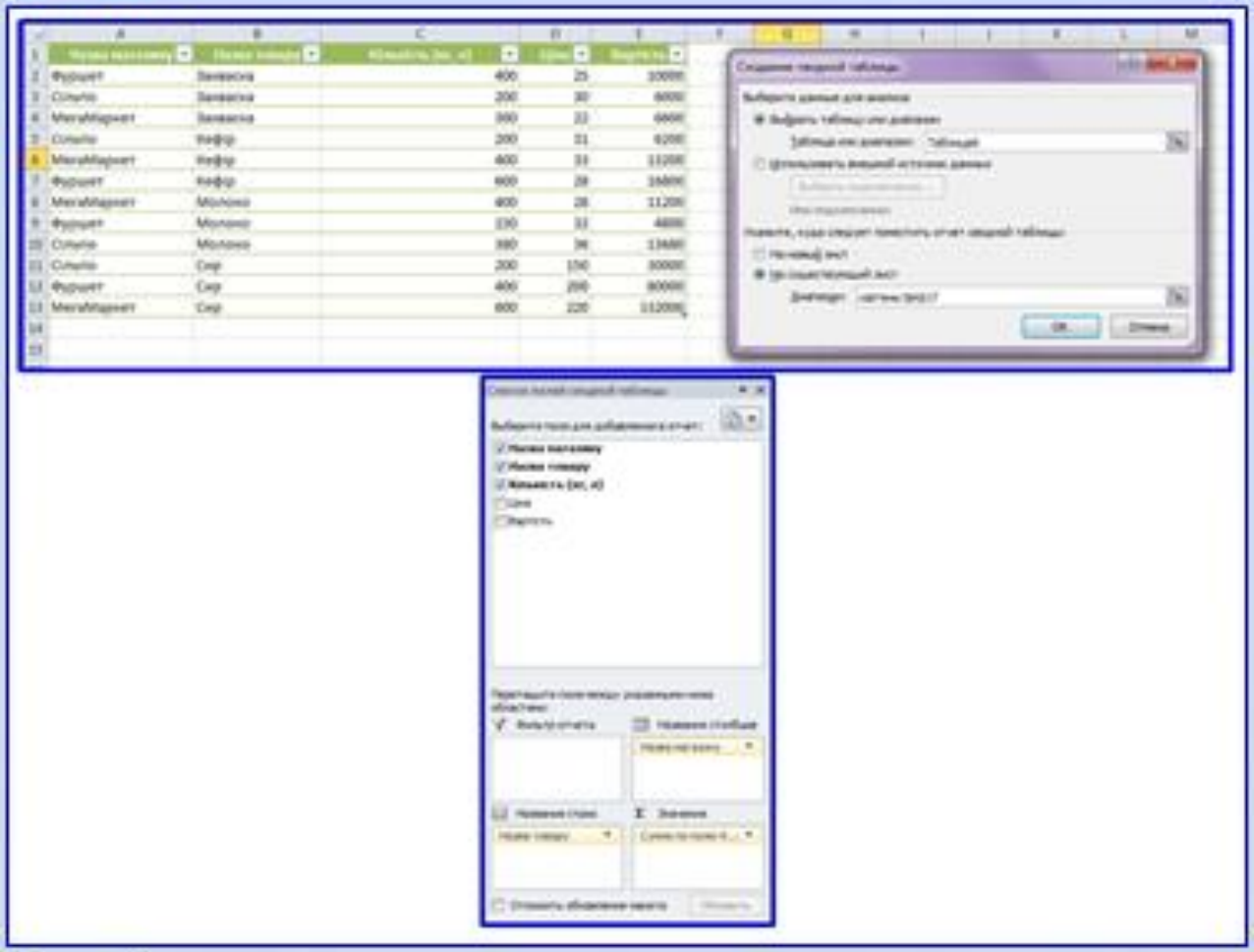
Консолідація даних



	A	B
1		Кількість (кг, л)
2	Закваска	1700
3	Кефір	2400
4	Молоко	1860
5	Сир	2400

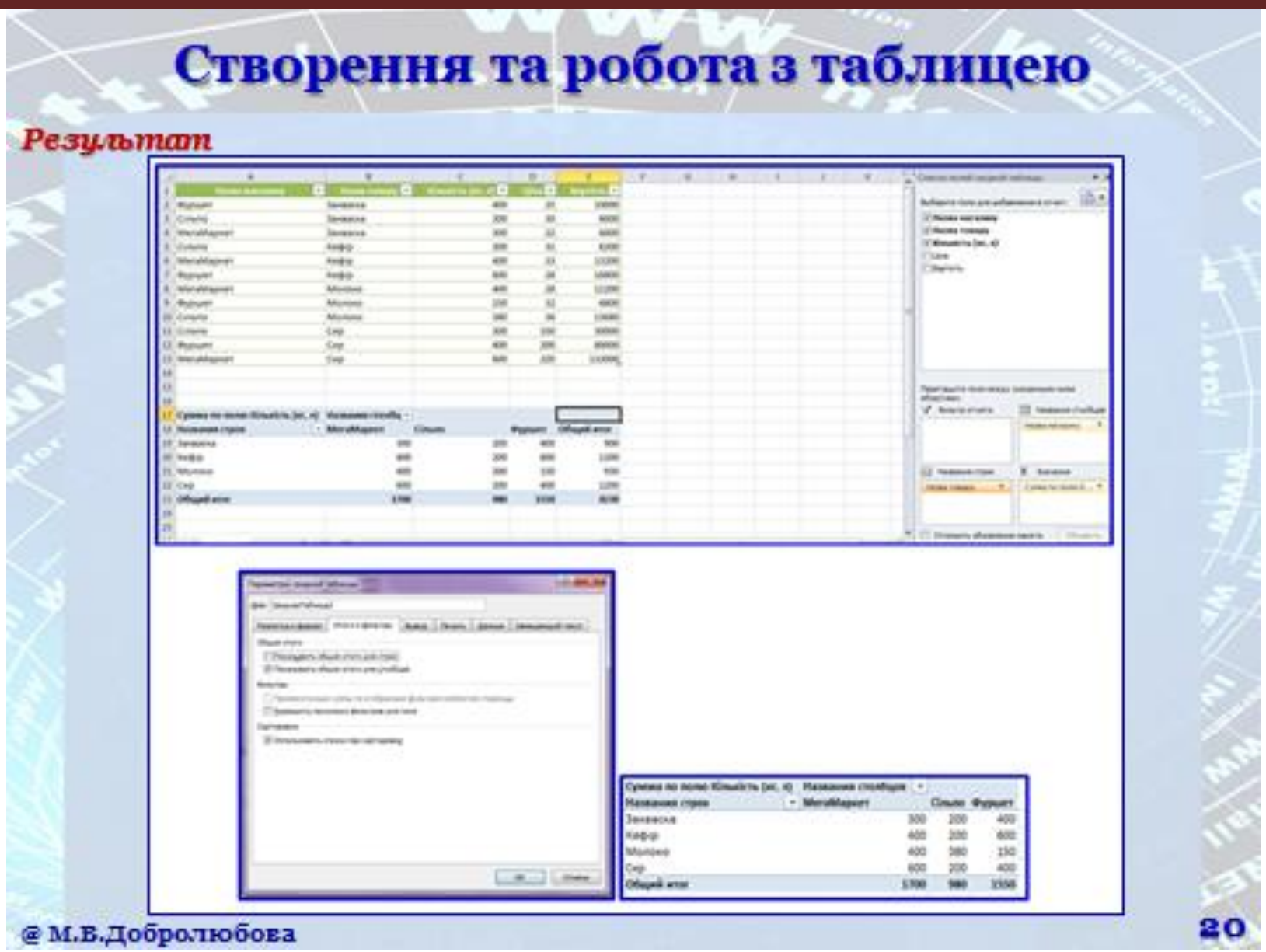
Створення та робота з таблицею

Зведена таблиця



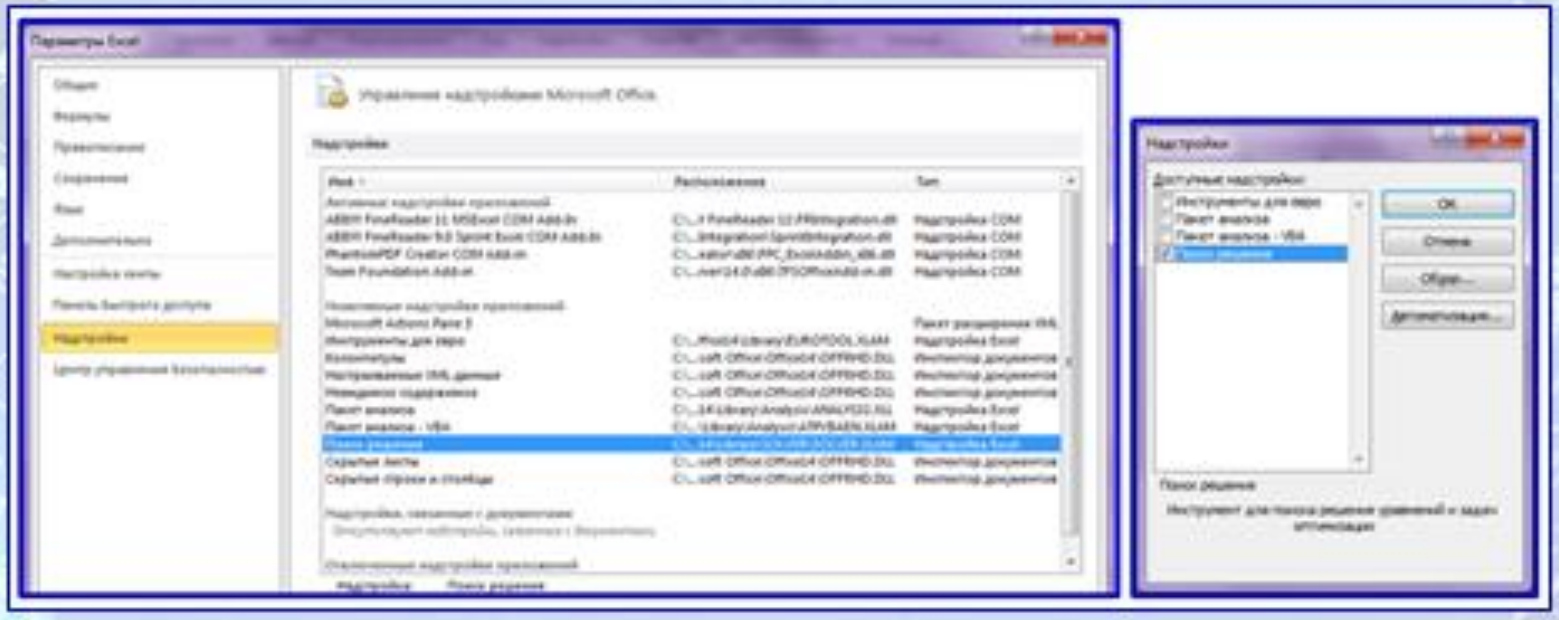
@ М.В.Добролюбова

19


Створення та робота з таблицею
Результат
@ М.В.Добролюбова
20

Розв'язок задач математичного програмування

Надбудова «Пошук розв'язку»



@ М.В.Добролюбова 21

Розв'язок задач математичного програмування

Приклад

Для виготовлення виробів x, y, z використовують три види сировини: I, II, III. У таблиці задані норми витрат сировини на один виріб кожного виду, ціна одного виробу, а також кількості сировини кожного виду, які можна використовувати. Скільки виробів кожного виду необхідно виготовити, щоб прибуток був максимальним?

	x	y	z	Загальна кількість сировини
I	18	15	12	360-n
II	6	4	8	192
III	5	3	3	180+n
Ціна	9	10	16	

Математична модель завдання. Позначимо через x, y, z шукані кількості виробів трьох видів. Необхідно визначити цілі значення x, y, z, для яких досягається максимум функції прибутку $f = 9x + 10y + 16z$ при наступних обмеженнях:

$18x + 15y + 12z \le 360 - n$

$6x + 4y + 8z \le 192$

$5x + 3y + 3z \le 180 + n$

$x, y, z \ge 0; x, y, z$ – цілые

@ М.В.Добролюбова 22

Розв'язок задач математичного програмування

Розв'язок:

1. Клітинкам A1, B1, C1 привласнити імена *x, y, z* командою з контекстного меню.
2. У клітинку D1 ввести формулу =9*x+10*y+16*z.
3. Запустити програму *Пошук рішення*.
4. Задати адресу цільової клітинки D1 і вказати дію досягнення максимуму функції.
5. Задати клітинки, у яких повинне знаходитися рішення: *x, y, z*, використавши для цього параметр *Змінюючи комірки*.

@ М.В.Добролюбова

23

Розв'язок задач математичного програмування

Розв'язок:

6. За допомогою кнопки **Додати** додати обмеження у вигляді дев'яти умов:

$x \le (360 - 15 * y - 12 * z) / 18$

$y \le (192 - 6 * x - 8 * z) / 4$

$z \le (180 - 5 * x - 3 * y) / 3$

$x \ge 0, y \ge 0, z \ge 0$ //*Як окрема умова*

x – *ціле*, y – *ціле*, z – *ціле*

Якщо необхідно досягти max F, то в обмеженнях для x, y, z ставиться знак <=, якщо min, то >=.

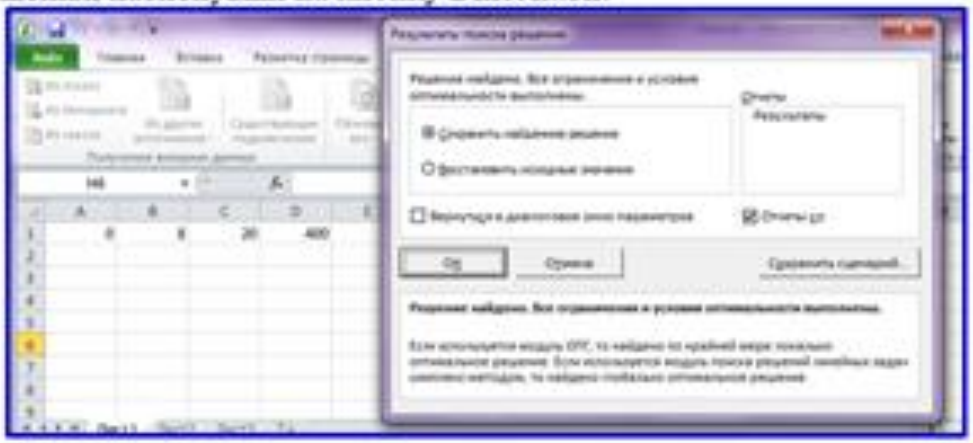
@ М.В.Добролюбова

24

Розв'язок задач математичного програмування

Розв'язок:

7. Отримати рішення, натиснувши на кнопку **Виконати**.



Створення макросу

Приклад

Записати макрос на простому прикладі «Виведення на екран назв місяців», тобто необхідно створити макрос, який буде виводити на екран стовпець із назвами місяців в комірках A1:A12.

Розв'язок:

1. Встановити курсор в будь-яку комірку книги, за винятком A1.
2. Виконати команду *Вид → Макроси*. З'явиться вікно діалогу *Запис макросу*.
3. Ввести в поле введення *Ім'я макросу* назву створюваного макросу *Місяць_абс*.

В поле введення *Опис* ввести текст: Введіть назви місяців.

4. Ввести клавішу швидкого виклику макросу. Для цього перейти в поле вводу *Ctrl*, перемкнутися на латинський регістр, видалити букву, що перебуває там, і ввести букву *m*. Після цього натиснути **ОК**. Починаючи із цього моменту, здійснюється запис макросу.

5. Виконати послідовність дій, що буде виконувати макрос: встановити курсор у комірку A1; ввести слово січень; помістити курсор миші в правий нижній кут комірки A1, при цьому курсор миші змінить вигляд на чорний хрестик; натиснути кнопку миші та, утримуючи її натиснутою, продовжити виділення до комірки A12.
6. Натиснути кнопку **Зупинити макрос**.

Якщо після цього видалити дані з комірок A1:A12 і натиснути *Ctrl+m* або обрати команду *Виконати макрос*, то на екрані автоматично в діапазоні A1:A12 з'явиться стовпець з назвою місяців.

Цей макрос створено в звичайному режимі, тобто назви місяців будуть з'являтися тільки в клітинках A1:A12. Щоб назви місяців з'являлися в будь-якому місці, треба встановити відносні посилання.

@ М.В.Добролюбова 26


Обчислювальна техніка та програмування
РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА.
ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Лекція 9
«Розробка та експлуатація баз даних. Концепції підтримки БД при її функціонуванні. Схеми опису даних»
@ М.В.Добролюбова

Бази даних. Адміністрація та система управління

Дані – це інформація, що знаходиться в пам'яті комп'ютера або на машинних носіях.

Обробка даних – це всі операції з даними, що необхідні для отримання бажаного результату.

База даних – це сукупність взаємозалежних даних певної предметної галузі, збережених у пам'яті комп'ютера і організованих таким чином, що ці дані можуть бути використані для розв'язання різних завдань багатьма користувачами.

Вимоги до організації бази даних

1. Ненадлишковість даних.
2. Спільне використання даних.
3. Можливість розширення бази.
4. Простота роботи з базою даних.
5. Ефективність доступу до бази даних.
6. Цілісність бази даних.
7. Захист даних.

@ М.В.Добролюбова 2

Бази даних. Адміністрація та система управління

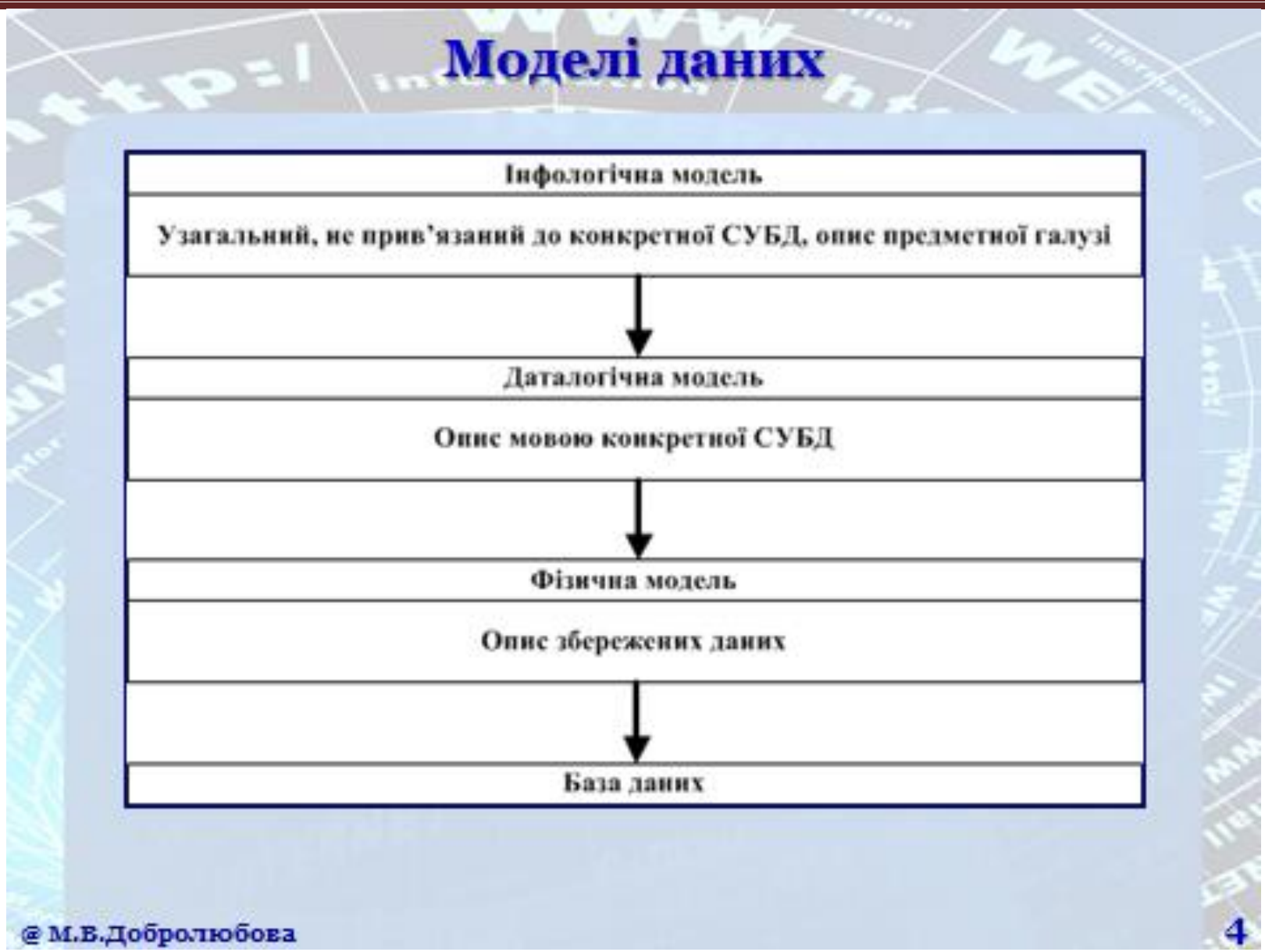
Функції адміністрації бази даних

1. Проектування структури БД.
2. Вибір засобу представлення даних.
3. Виконання обслуговуючих функцій.
4. Планування розвитку БД і пов'язаний із цим вибір нових засобів обчислювальної техніки.
5. Консультації користувачів щодо використання БД.
6. Контроль користувачів, що працюють із БД, врегулювання різноманітних конфліктних ситуацій.

Функції типової системи управління бази даних

1. Опис даних (схема бази).
2. Опис незмінних властивостей даних БД за допомогою обмеження на припустимі операції з даними.
3. Маніпулювання даними.
4. Завантаження бази і формування звітів.
5. Мова запитів.
6. Діалогові засоби.
7. Мультидоступ до БД.

@ М.В.Добролюбова 3


Моделі даних
Інфологічна модель
Узагальнений, не прив'язаний до конкретної СУБД, опис предметної галузі
Даталогічна модель
Опис мовою конкретної СУБД
Фізична модель
Опис збережених даних
База даних
@ М.В.Добролюбова
4

Основні елементи інфологічної моделі даних «сутність-зв'язок»

Сутність – будь-який помітний об'єкт (об'єкт, що можна відрізнити від іншого), інформацію про який необхідно зберігати в базі даних.

Поняття сутності:

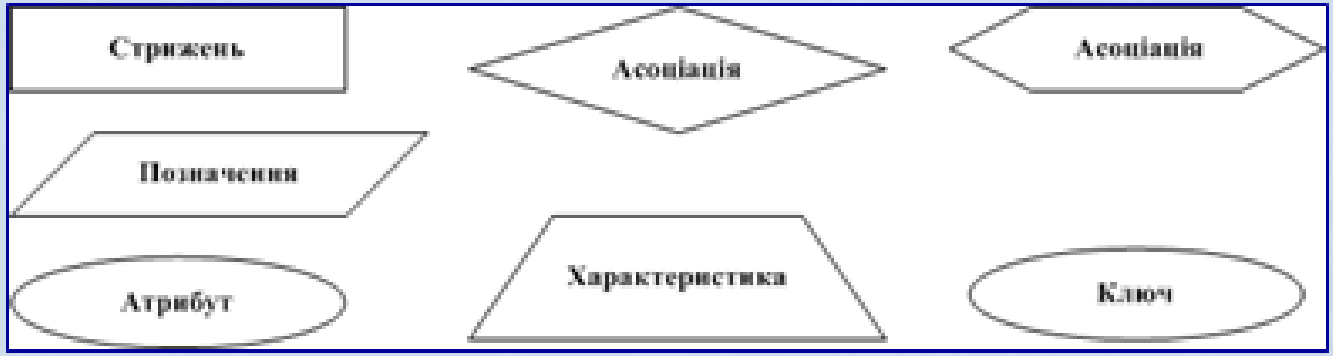
- тип сутності – набір однорідних особистостей, предметів, подій або ідей, що виступають як ціле;
- екземпляр сутності – це конкретна річ у наборі.

Атрибут – характеристика сутності, яка має ім'я.

Ключ – мінімальний набір атрибутів, за значеннями яких можна однозначно знайти необхідний екземпляр сутності.

Зв'язок – асоціювання двох або більше сутностей.

ER-діаграми



© М.Б.Добролюбова

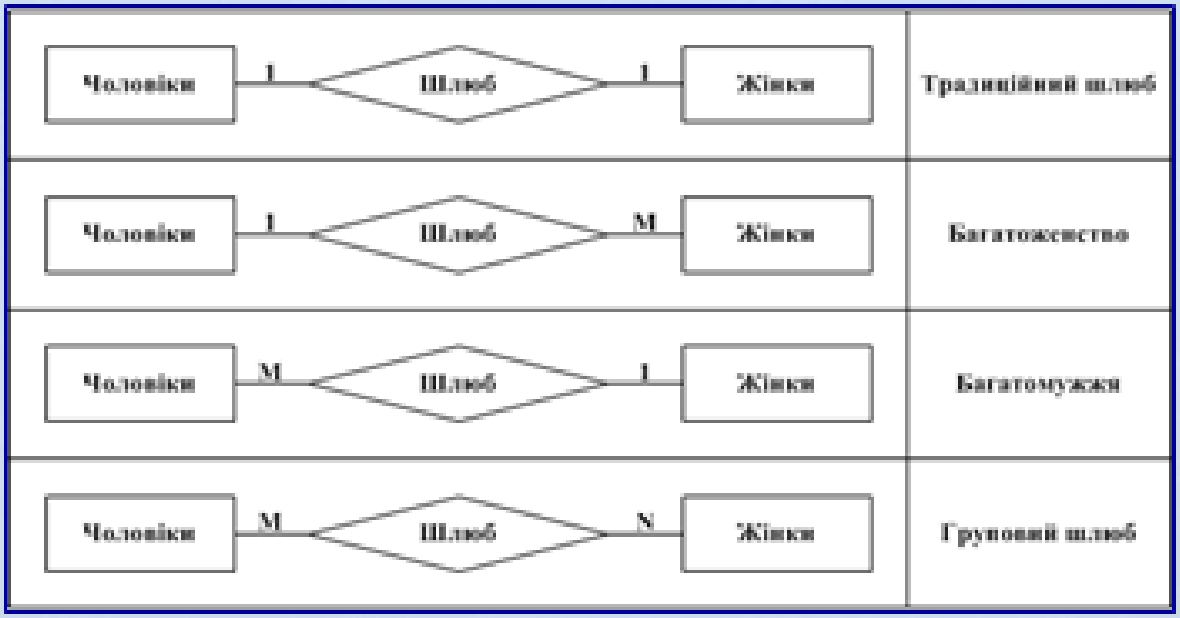
5

Види зв'язків між сутностями

Один-до-Одного (1:1)

Один-до-Багатьох (1:M 1:∞)

Приклад



© М.В.Добролюбова

6

Класи сутностей. Ключ бази даних

Класи сутностей

Стрижнева сутність (стрижень) – це незалежна сутність.

Асоціативна сутність (асоціація) – це зв'язок вигляду «багато-до-багатьох» та «один-до-багатьох» між двома або більше сутностями або екземплярами сутності.

Характеристична сутність (характеристика) – це зв'язок вигляду «багато-до-одного» або «один-до-одного» між двома сутностями.

Позначаюча сутність або позначення – це зв'язок вигляду «багато-до-одного» або «один-до-одного» між двома сутностями, який відрізняється від характеристики тим, що не залежить від позначеної сутності.

Поняття «ключ бази даних»

Ключ – це мінімальний набір атрибутів, за значеннями яких можна однозначно знайти необхідний екземпляр сутності. Мінімальність означає, що виключення з набору будь-якого атрибута не дозволяє ідентифікувати однозначно відповідну сутність.

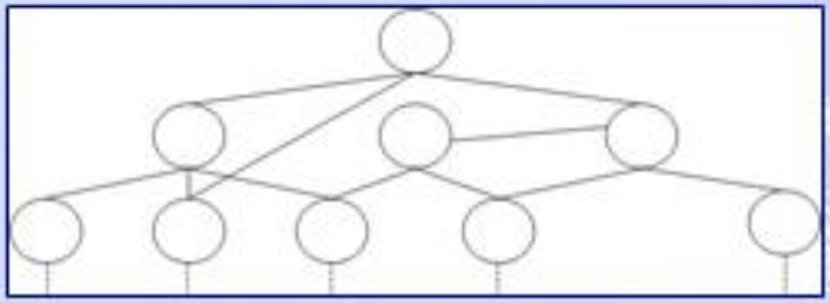
Первинний ключ – це ключ, складений з мінімального числа атрибутів. Первинний ключа має бути унікальним.

Правило для зовнішніх ключів: якщо сутність С зв'язує сутності А та В, то вона повинна включати зовнішні ключі, що відповідають первинним ключам сутностей А та В. Значення зовнішнього ключа повинно або бути рівним значенню первинного ключа цілі, або бути цілком невизначеним, тобто кожне значення атрибута, що бере участь у зовнішньому ключі, повинно бути невизначеним.

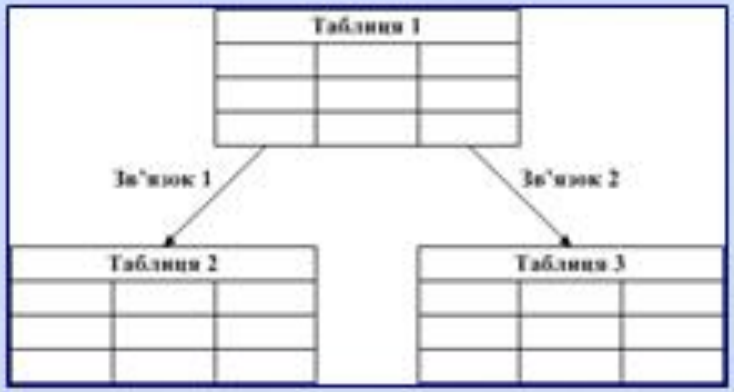
Даталогічні моделі даних

Ієрархічна модель

Мережева модель



Реляційна модель



@ М.В.Добролюбова

8

Алгоритм отримання реляційної схеми з ER-схеми

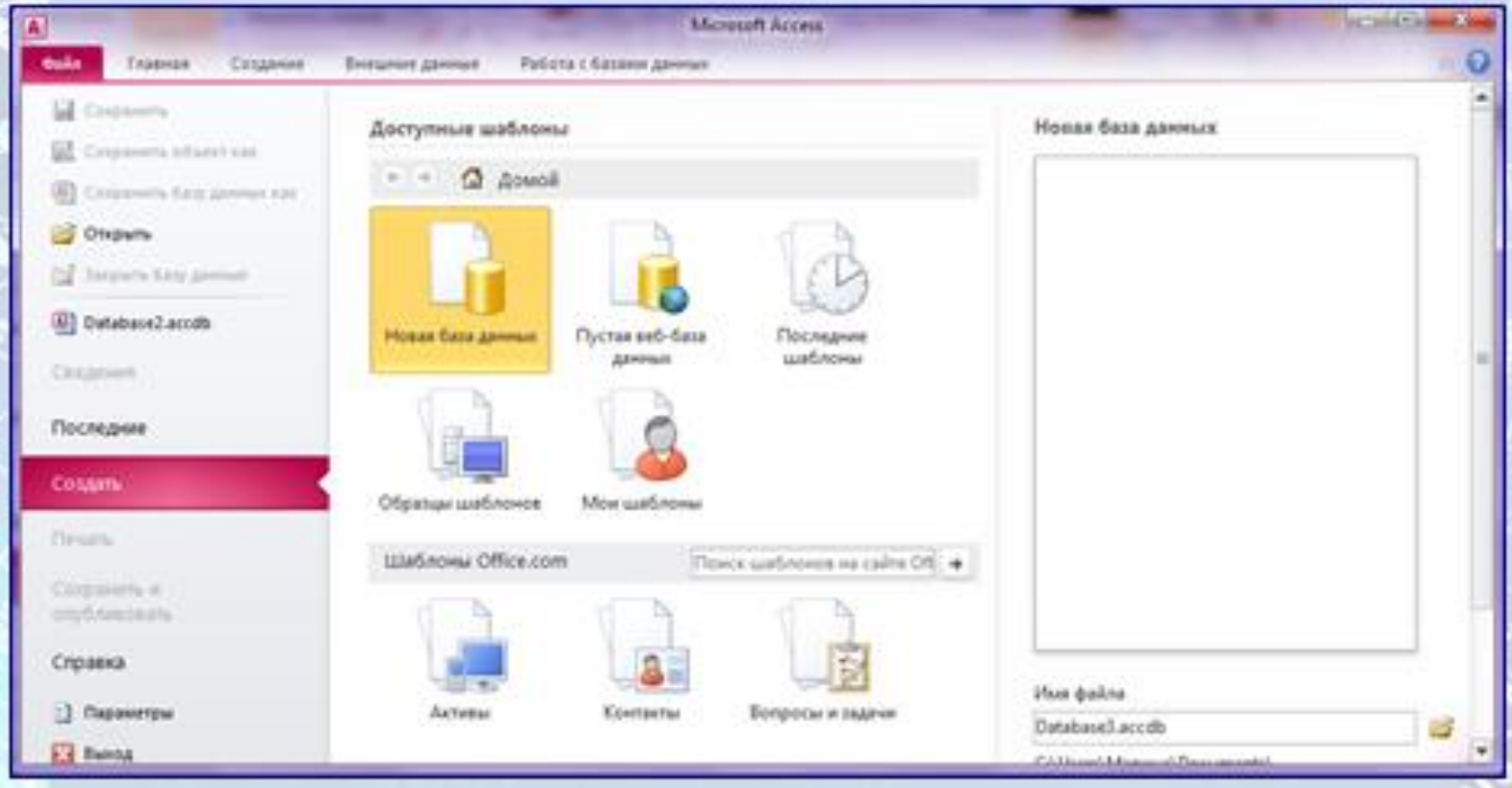
1. Представити кожний стрижень таблицею бази даних і специфікувати первинний ключ цієї базової таблиці.
2. Представити кожну асоціацію як базову таблицю. Використати в цій таблиці зовнішні ключі для ідентифікації учасників асоціації і специфікувати обмеження, пов'язані з кожним із цих зовнішніх ключів.
3. Представити кожну характеристику як базову таблицю з зовнішнім ключем, що ідентифікує сутність, яка описується цією характеристикою. Специфікувати обмеження на зовнішній ключ цієї таблиці та її первинний ключ.
4. Представити кожне позначення, що не розглядалося в попередньому пункті, як базову таблицю з зовнішнім ключем, що ідентифікує позначену сутність. Специфікувати обмеження, пов'язані з кожним таким зовнішнім ключем.
5. Представити кожну властивість як поле в базовій таблиці, що представляє сутність, яка безпосередньо описується цією властивістю.
6. Для виключення ненавмисних порушень виконується процедура нормалізації, яка ґрунтується на тому, що єдиними функціональними залежностями в будь-якій таблиці мають бути залежності вигляду K→F, де K – первинний ключ, а F – деяке інше поле. Зауважимо, що це випливає з визначення первинного ключа таблиці, відповідно до якого K→F завжди має місце для всіх полів такої таблиці. "Один факт в одному місці" говорить про те, що не мають сили ніякі інші функціональні залежності. Мета нормалізації полягає саме в тому, щоб позбутися всіх "інших" функціональних залежностей, тобто таких, що мають інший вигляд, ніж K→F.
7. Якщо в процесі нормалізації був зроблений поділ будь-яких таблиць, то варто модифікувати інфологічну модель бази даних і повторити перераховані кроки.
8. Вказати умови цілісності бази даних і дати (якщо це необхідно) стислий опис отриманих таблиць і їх полів.

© М.В.Добролюбова 9


Обчислювальна техніка та програмування
РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА.
ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Лекція 10
«Інструменти створення БД. MS Access»
© М.В.Добролюбова

MS ACCESS

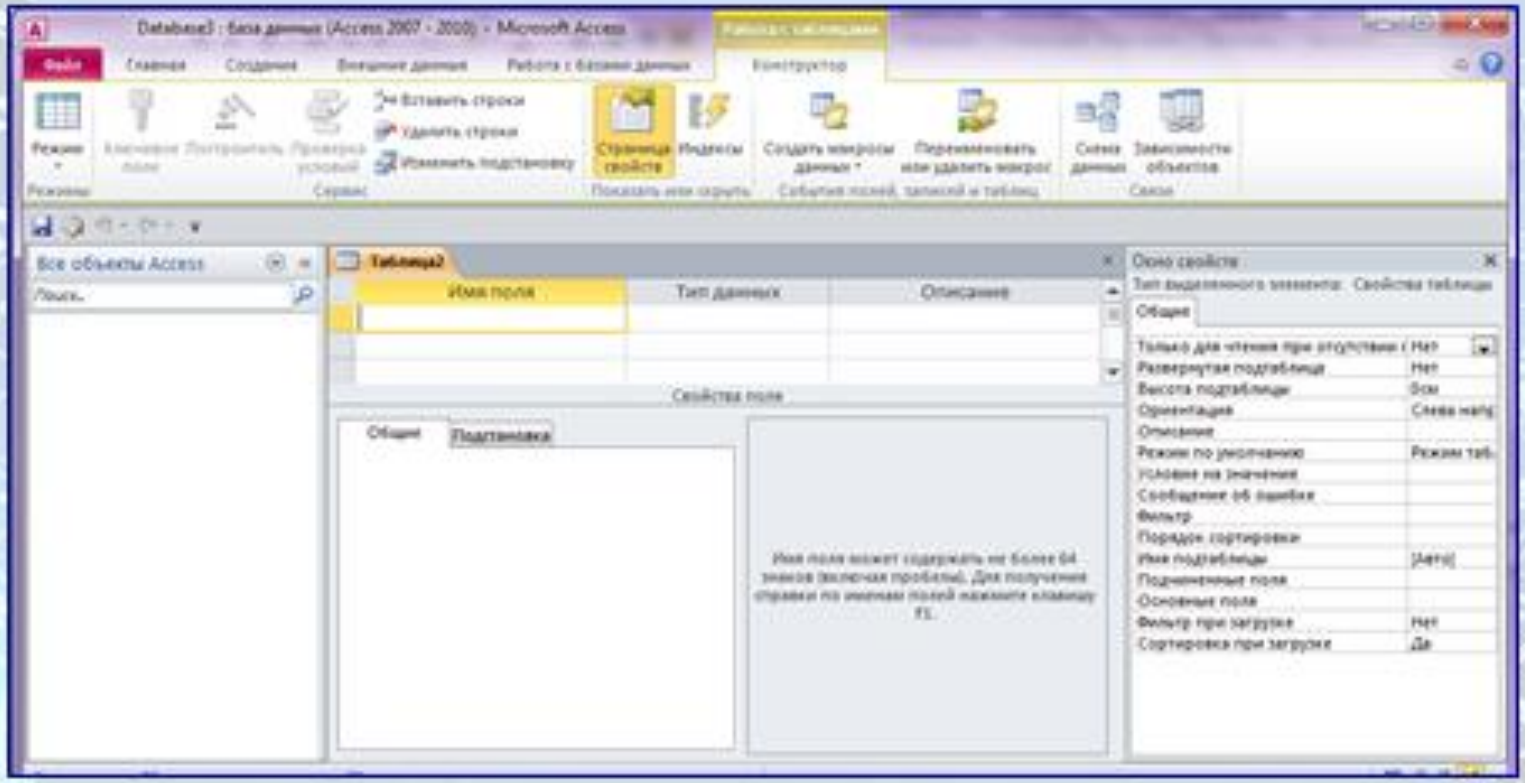
Створення файла нової БД



@ М.В.Добролюбова

MS ACCESS

Конструктор таблиць

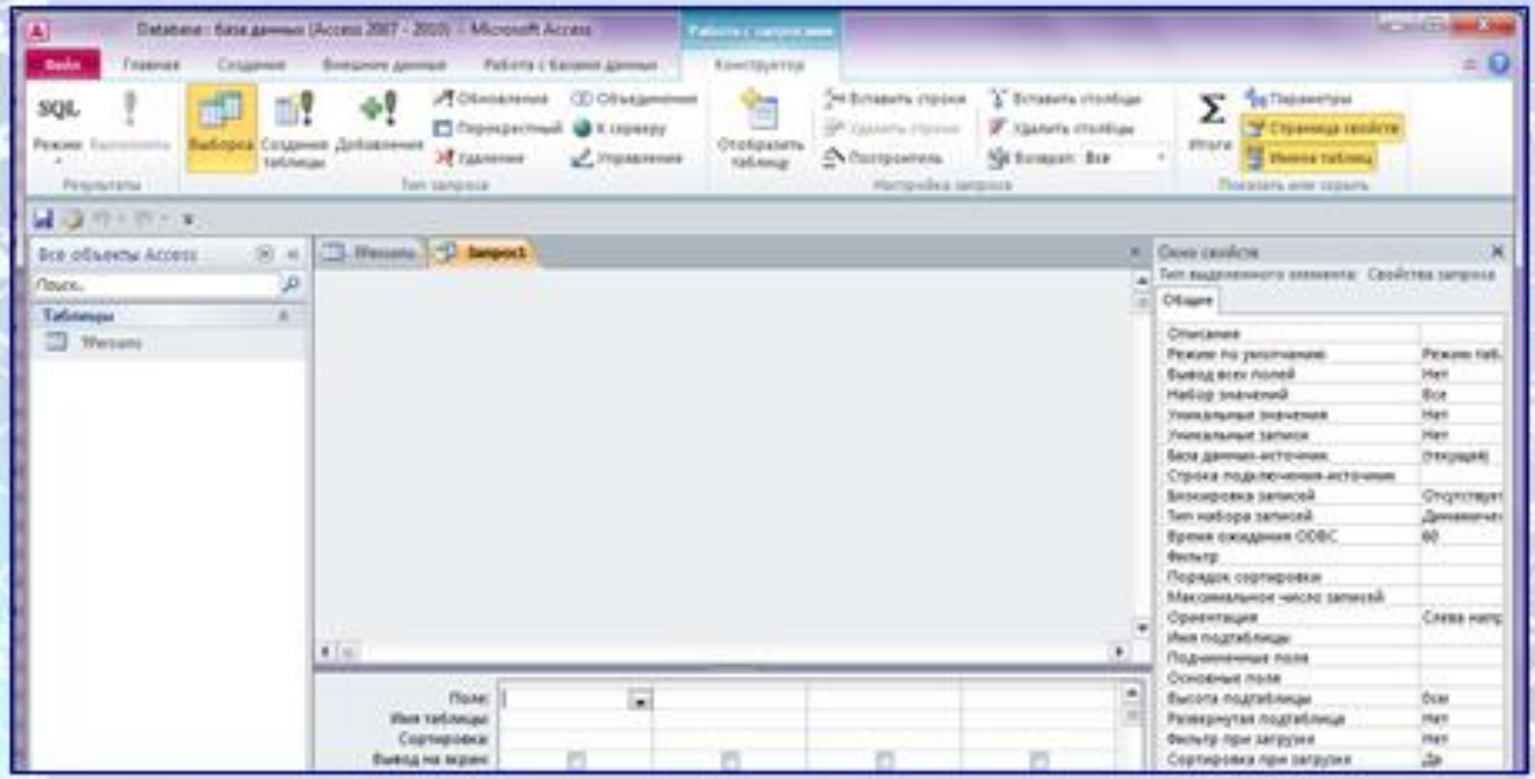


@ М.В.Добролюбова

3

MS ACCESS

Конструктор запитів

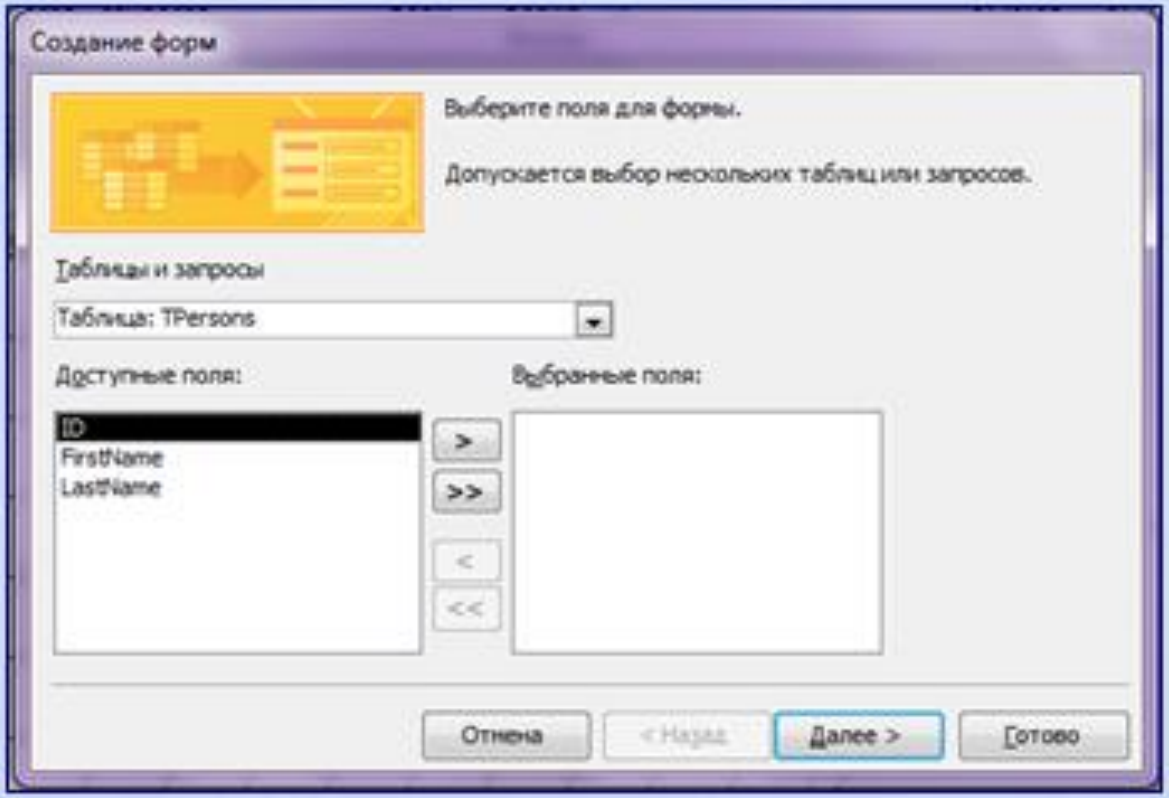


@ М.В.Добролюбова

4

MS ACCESS

Використання майстра форм

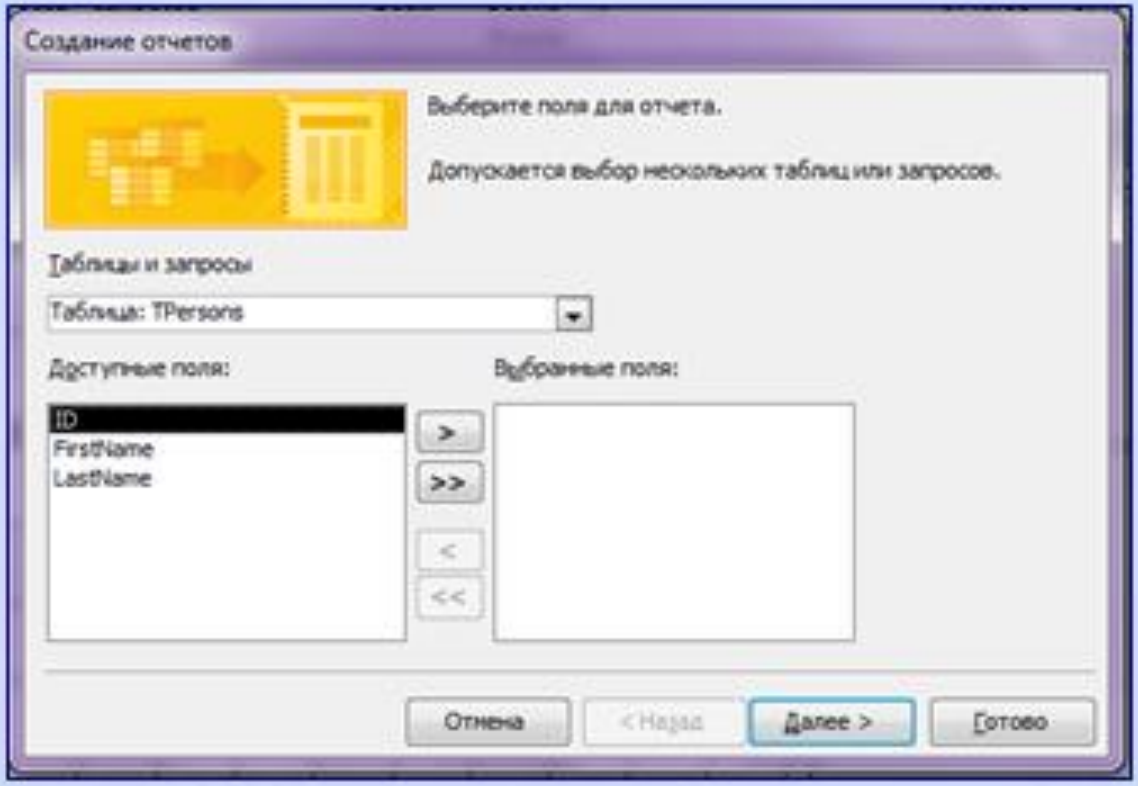


@ М.В.Добролюбова

5

MS ACCESS

Використання майстра звітів

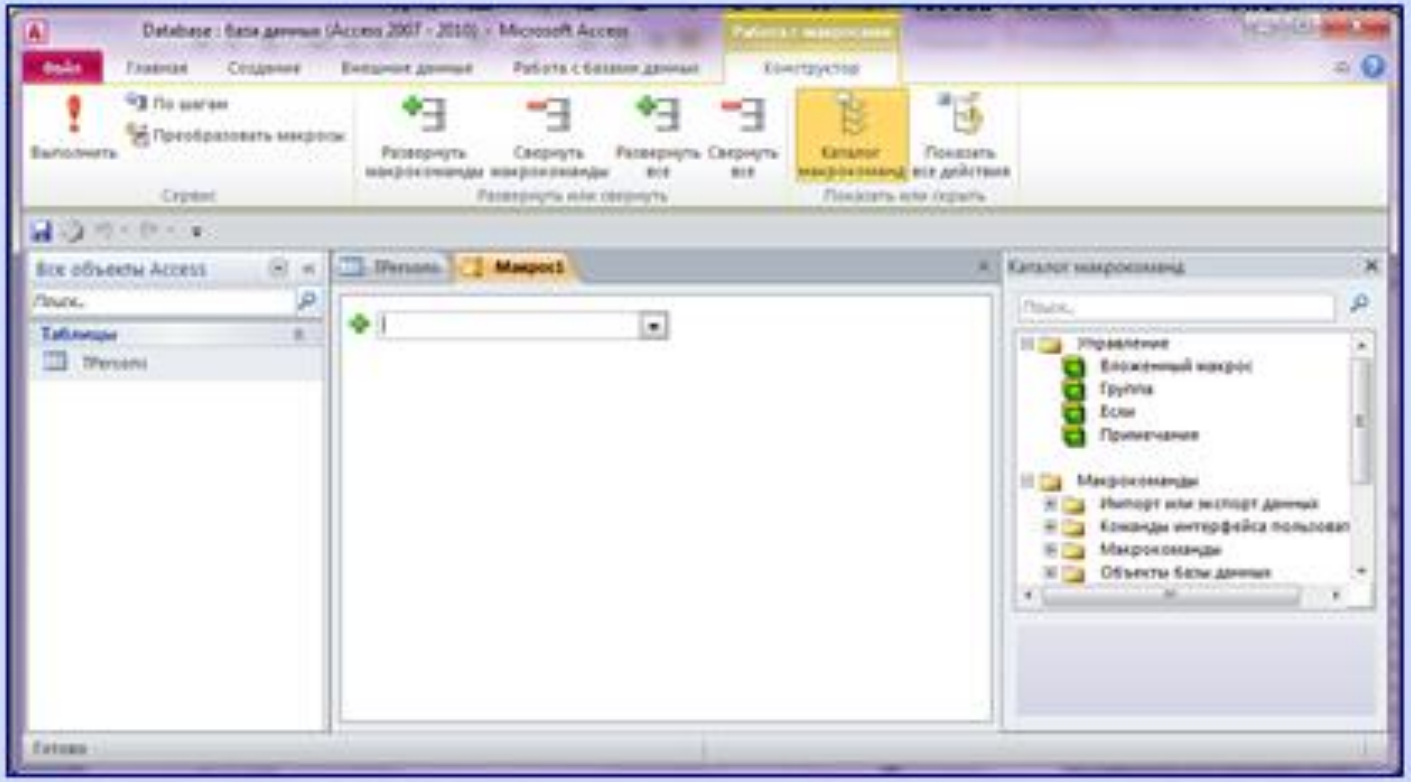


@ М.В.Добролюбова

6

MS ACCESS

Інструмент для створення і роботи з макросами



SQL

Інтерактивна форма SQL застосовується для виконання дій безпосередньо в БД з метою отримання бажаного результату на основі формування відповідної команди.

Вбудована форма SQL складається з включених в програми команд SQL, які написані іншою мовою програмування, що робить програму більш ефективною.

Мова визначення даних (Data Definition Language – DDL), в ANSI вона також зветься мовою визначення схеми (Scheme Definition Language) складається з команд, які створюють та знищують у БД такі об'єкти як таблиці, індекси, представлення.

Мова маніпулювання даними (Data Manipulation Language – DML) – це множина команд, що визначають, які дані надані в таблицях в будь-який момент часу.

Мова управління даними (Data Control Language – DCL) складається з речень, які визначають, чи може користувач виконати певну окрему дію.

@ М.В.Добролюбова 8

Мова DDL

Основні DDL-оператори мови SQL

CREATE SCHEMA		DROP SCHEMA
CREATE DOMAIN	ALTER DOMAIN	DROP DOMAIN
CREATE TABLE	ALTER TABLE	DROP TABLE
CREATE VIEW		DROP VIEW
CREATE INDEX		DROP INDEX

Оператор визначення (створення)

```
CREATE SCHEMA [name | AUTHORIZATION creator-identifier]

CREATE DATABASE database_name

CREATE TABLE table_name
(column_name data_type [NULL | NOT NULL] [, . . .])

CREATE [UNIQUE] INDEX index_name
ON table_name (column [ASC | DESC] [, . . .])
```

Оператор видалення

```
DROP SCHEMA name [RESTRICT | CASCADE]

DROP DATABASE database_name

DROP TABLE table_name [RESTRICT | CASCADE]

DROP INDEX index_name
```

@ М.В.Добролюбова 9

Мова DML

Основні DML-оператори мови SQL

SELECT – вибірка даних з бази
INSERT – вставка даних в таблицю
UPDATE – оновлення (зміна) даних таблиці
DELETE – видалення даних з таблиці

Контекст БД DreamHome для побудови прикладів DML-операторів

Branch	(Bno, Street, Area, City, Pcode, Tel_No, Fax_No)
Staff	(Sno, FName, LName, Address, Tel_No, Position, Sex, DOB, Salary, NIN, Bno)
Property_for_Rent	(Pno, Street, Area, City, Pcode, Type, Rooms, Rent, Ono, Sno, Bno)
Renter	(Rno, FName, LName, Address, Tel_No, Pref_Type, Max_Rent, Bno)
Owner	(Ono, FName, LName, Address, Tel_No)
Viewing	(Rno, Pno, Date, Comment)

@ М.В.Добролюбова 10

Мова DML. Оператор SELECT

Загальний формат оператора SELECT

```
SELECT [DISTINCT | ALL] {* | [column_expression [AS new_name]]
[...]}
FROM table_name [alias] [...]
[WHERE condition]
[GROUP BY column_list] [HAVING condition]
[ORDER BY column_list]
```

FROM	Визначаються імена використовуваної таблиці або декількох таблиць
WHERE	Виконується фільтрація рядків об'єкта відповідно до заданих умов
GROUP BY	Утворюються групи рядків, що мають одне й те саме значення в зазначеному стовпці
HAVING	Фільтруються групи рядків об'єкта відповідно до зазначеної умови
SELECT	Встановлюється, які стовпці повинні бути присутніми у вихідних даних
ORDER BY	Визначається впорядкованість результатів виконання оператора

@ М.В.Добролюбова 11

Мова DML. Оператор SELECT

Приклади роботи оператора SELECT

Вибірка рядків

Приклад 1

```
SELECT sno, fname, lname, address, tel_no, position, sex, dob, salary, nin, bno
FROM staff;

SELECT *
FROM staff;
```

Приклад 2

```
SELECT sno, fname, lname, salary
FROM staff;
```

Приклад 3

```
SELECT sno, fname, lname, salary/12
FROM staff;

SELECT sno, fname, lname, salary AS monthly_salary
FROM staff;
```

Вибір рядків (пропозиція WHERE)

Приклад 4

```
SELECT sno, fname, lname, position, salary
FROM staff;
WHERE salary > 10000;
```

Мова DML. Оператор SELECT

Приклади роботи оператора SELECT

Сортування результатів (фраза ORDER BY)

Приклад 5

```
SELECT sno, fname, lname, salary
FROM staff
ORDER BY salary DESC;
```

Групування результатів (фраза Group BY)

Приклад 6

```
SELECT bno, COUNT(sno) AS count, SUM(salary) AS sum
FROM staff
GROUP BY bno
ORDER BY bno;
```

Обмеження на виконання групування (фраза HAVING)

Приклад 7

```
SELECT bno, COUNT(sno) AS count, SUM(salary) AS sum
FROM staff
GROUP BY bno
HAVING COUNT (sno) > 1
ORDER BY bno;
```

@ М.В.Добролюбова 13

Мова DML. Оператор SELECT

Приклади роботи оператора SELECT

Підзапити

Приклад 8

```
SELECT sno, fnarne, lnarne, position
FROM staff
WHERE bno =
      (SELECT bno
      FROM branch
      WHERE street = '163 Main st');
```

Вигляд зовнішнього оператора SELECT:

```
SELECT sno, fnarne, lnarne, position
FROM staff
WHERE bno = 'B3';
```

Багатотабличні запити

Виконання з'єднань

```
SELECT [DISTINCT | ALL] (* | column_list)
FROM table_name1 CROSS JOIN table_name2
```

Приклад 9

```
SELECT r.rno, fname, lname, pno, comment
FROM renter r, viewing v
WHERE r.rno = v.rno;
```

Мова DML. Оператори модифікації інформації

Додавання нових даних в таблицю (оператор INSERT)

Перша форма оператора INSERT:

```
INSERT INTO table_name [(column_list)]
VALUES (data_value_list)
```

Використання конструкцій INSERT ... VALUES

Приклад 10

```
INSERT INTO staff
VALUES ('SG16', 'Alan', 'Brown', '67 Endrick Rd, Glasgow G32 8QX','0141-211-3001', 'Assistant', 'M',
DATE '1957-05-25', 8300, 'WN848391H', 'B3');
```

Друга форма оператора INSERT:

```
INSERT INTO table_name [(column_list)]
SELECT. . .
```

Використання конструкцій INSERT ... SELECT

Приклад 11

Таблиця Staff_Prop_Count

```
Staff_Prop_Count (sno, fname, lname, prop_count)

INSERT INTO staff_prop_count
(SELECT s.sno, fname, lname, COUNT (*)
FROM staff s, property_for_rent p
WHERE s.sno = p.sno
GROUP BY s.sno, fname, lname)
UNION
(SELECT sno, fname, lname, 0
FROM staff
WHERE sno NOT IN
     (SELECT DISTINCT sno
     FROM property_for_rent));
```

© М.Е.Добролюбова 15

Мова DML. Оператори модифікації інформації

Модифікація даних в базі (оператор UPDATE)

Формат оператора UPDATE:

```
UPDATE table_name
SET column_name1 = data_value1 [, column_name2 = data_value2 ...]
[WHERE search_condition]
```

Оновлення всіх рядків таблиці

Приклад 12

```
UPDATE staff
SET salary = salary * 1.03;
```

Оновлення деяких рядків таблиці

Приклад 13

```
UPDATE staff
SET salary = salary * 1.05
WHERE position = 'Manager';
```

Видалення даних з бази (оператор DELETE)

Формат оператора DELETE:

```
DELETE FROM table_name
[WHERE search_condition]
```

Видалення деяких рядків з таблиці

Приклад 14

```
DELETE FROM viewing
WHERE pno = 'PG4';
```

Видалення всіх рядків з таблиці

Приклад 15

```
DELETE FROM viewing;
```

© М.Е.Добролюбова 16

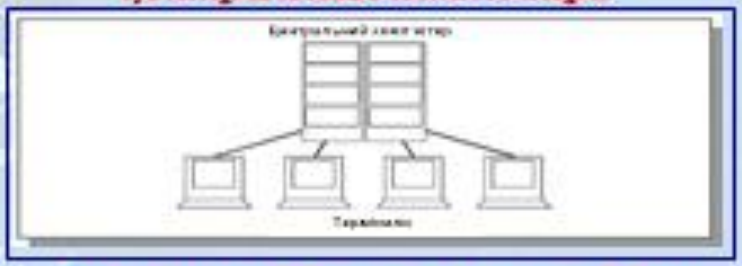

Обчислювальна техніка та програмування
РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА.
ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Лекція 11
«Поняття і функції локальних комп'ютерних мереж. Принципи побудови і використання»
© М.В.Добролюбова

Місце та роль локальних мереж. Історія комп'ютерного зв'язку

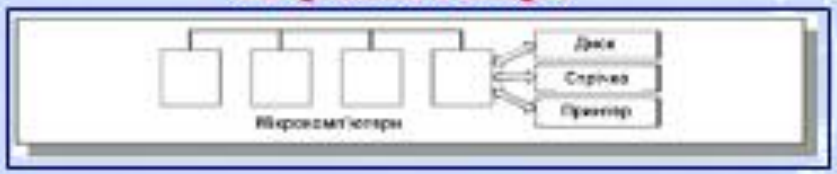
Комп'ютерна мережа – це система розподіленої обробки інформації між комп'ютерами за допомогою засобів зв'язку.

Етапи розвитку

1 Підключення терміналів до центрального комп'ютера

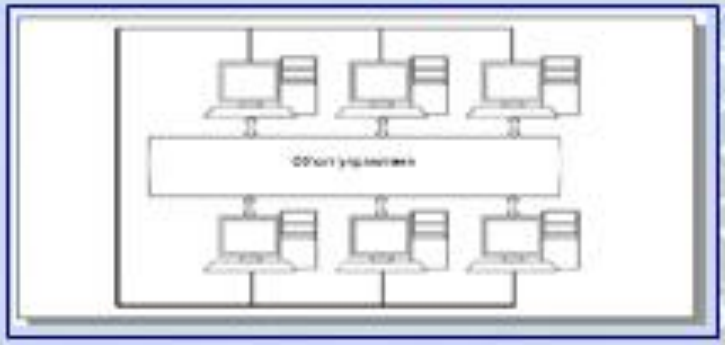


2 Об'єднання в мережу периферійних мікрокомп'ютерів



3 Об'єднання в мережу персональних комп'ютерів

4 Використання локальної мережі для організації спільної роботи комп'ютерів



@ М.В.Добролюбова

2

Визначення локальної мережі

«Локальна мережа» (ЛОМ) або *«локальна обчислювальна мережа»* (LAN, Local Area Network) – система передачі даних, що дозволяє поєднувати багато ПК за допомогою спеціально прокладених, високоякісних і добре захищених від завад ліній зв'язку та забезпечує прозорий зв'язок, високу швидкість передачі інформації, низький рівень помилок передачі, високу інтенсивність обміну.

Переваги і функції ЛОМ

- Поділ ресурсів
- Поділ даних
- Поділ програмних засобів
- Поділ ресурсів процесора
- Багатокористувальницький режим

Недоліки ЛОМ

- Додаткові матеріальні витрати на мережеве устаткування, програмне забезпечення, прокладку з'єднувальних кабелів і навчання персоналу.
- Введення посади адміністратора мережі.
- Обмеження можливості переміщення комп'ютерів, підключених до мережі.
- Поширення комп'ютерних вірусів.
- Підвищення небезпеки несанкціонованого доступу до інформації з метою її крадіжки або знищення.

@ М.В.Добролюбова 3

Визначення локальної мережі

Терміни

Абонент (вузол, хост, станція) – це пристрій, підключений до мережі, що активно бере участь в інформаційному обміні.

Сервер – це абонент (вузол) мережі, що надає свої ресурси іншим абонентам, але сам не використовує їх ресурси.

Виділений (dedicated) ***сервер*** – це сервер, що займається тільки мережевими завданнями.

Невиділений сервер може крім обслуговування мережі виконувати й інші завдання.

Клієнт – абонент мережі, що тільки використовує мережеві ресурси, але сам свої ресурси в мережу не віддає.

@ М.В.Добролюбова 4

Стандарт OSI (Open System Interconnection)

Стандарт OSI (Open System Interconnect) – «Еталонна модель взаємодії відкритих систем» – основна архітектурна модель для систем передачі повідомлень.

Протокол передачі даних – набір правил і процедур, що регулюють порядок здійснення зв'язку.

Інформація для протоколу передачі даних

- Синхронізація
- Ініціалізація
- Блокування
- Адресація
- Виявлення помилок
- Нумерація блоків
- Управління потоками даних
- Методи відновлення
- Дозвіл доступу

Рівні еталонної моделі OSI

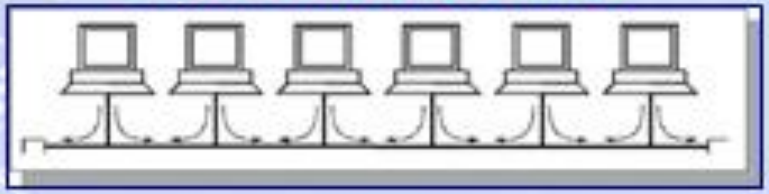
- Прикладний рівень (рівень 7)
- Рівень подання даних (рівень 6)
- Сеансовий рівень (рівень 5)
- Транспортний рівень (рівень 4)
- Мережевий рівень (рівень 3)
- Канальний рівень (рівень 2)
- Фізичний рівень (рівень 1)

@ М.В.Добролюбова 5

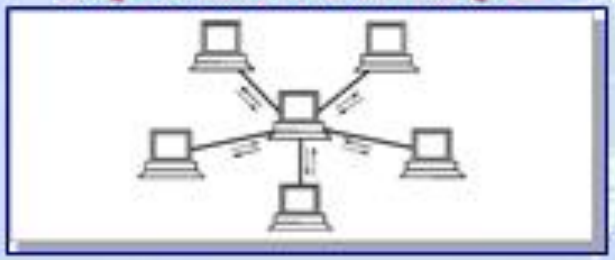
Топологія локальних мереж

Топологія – це фізичне розташування комп'ютерів мережі по відношенню один до одного та спосіб з'єднання їх лініями зв'язку.

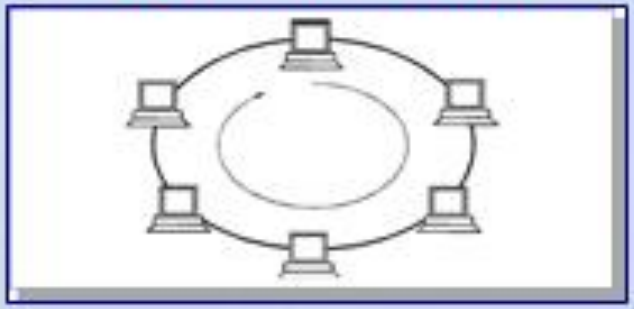
Мережева топологія шина



Мережева топологія зірка



Мережева топологія кільце



@ М.В.Добролюбова

6

Топологія локальних мереж

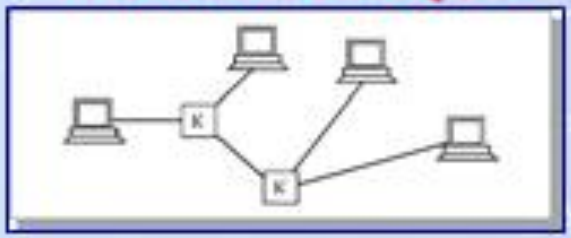
Порівняльна характеристика трьох базових топологій

Топологія	Ідентичність мережевого обладнання ПК	Рівноправність абонентів	Передача інформації	Додавання нового абонента	Вирішення (розв'язання) конфлікту	Мережеве обладнання	Розрив / ушкодження кабелю	Відмова мережевого обладнання абонента	Довжина лінії зв'язку	Відмова ПК
Шина	Відсутній явно виражений центральний абонент	Відсутній явно виражений центральний абонент	По черзі (одна лінія зв'язку), режим напівдуплексного обміну	Просте, з мінімальною кількістю кабелю, велика максимальна кількість ПК	Легас на мережеве обладнання	Складне	Обмін припиняється повністю	Виводить з роботи всю мережу	Не може бути більше $L_{доп}$, забезпечує найменшу довжину	Справні можуть працювати
Зірка	Явно виділений центральний ПК	Центральний ПК сильний потужний	Точка-точка; на кожній лінії зв'язку тільки два абонента – ЦПК та периферійний, 2 лінії зв'язку, кожна з яких передає інформацію тільки в одному напрямку	Просте, але жорстко обмежене (L-16)	Не можливі, оскільки управління централізоване	Спрощене	Порушується обмін лише з 1 ПК	Не відображається на роботі залишившоїся частини мережі	Кожний ПК посилює сигнал, що надходить, $2L_{доп}$, великі витрати кабеля	Ніяк не відображається на роботі залишившихся, відмова ЦПК – мережа непрацездатна повністю
Кільце	Немає чітко виділеного центру, рівноправні ПК	ПК рівноправні, але неоднозначні (один інформацію отримує раніше, інший – пізніше)	Точка-точка, кожний ПК з'єднаний з 2: від одного приймає, іншому передає	Просте, але потребує зупинки роботи усієї мережі. Кількість 1000 та більше	Немає конфліктів	Спрощене	Робота всієї мережі неможлива	Виводить з роботи всю мережу	Сумарна довжина кілець $NL_{доп}$, де N – кількість ПК в мережі	Порушується робота всієї мережі

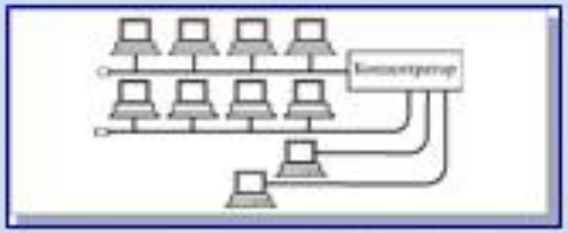
Топологія локальних мереж

Топологія активне дерево

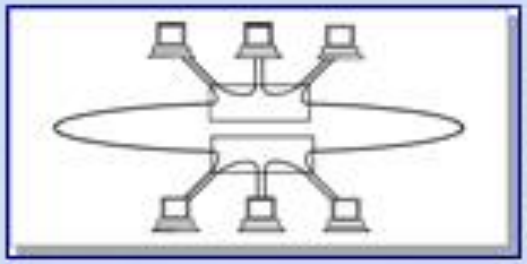
Топологія пасивне дерево



Приклад зірково-шинної топології



Приклад зірково-кільцевої топології



Мережева топологія: повна (а) і часткова (б)

© М.В.Добролюбова

8

Топологія локальних мереж

Фактори, що впливають на фізичну працездатність мережі і безпосередньо пов'язані з поняттям топологія

- Справність комп'ютерів (абонентів), підключених до мережі.
- Справність мережевого устаткування.
- Цілісність кабелю мережі.
- Обмеження довжини кабелю.

Види топологій

- Фізична топологія
- Логічна топологія
- Топологія керування обміном
- Інформаційна топологія

@ М.В.Добролюбова 9


Обчислювальна техніка та програмування
РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА.
ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Лекція 12
«Апаратні мережеві засоби»
© М.В.Добролюбова

Кабелі на основі витих пар

Кабель на основі витих пар – це декілька пар скручених парами ізольованих мідних дротів у єдиній діелектричній (пластиковій) оболонці. Зазвичай в кабель входить дві або чотири виті пари. Розрізняють неекрановані та екрановані виті пари.

Неекранована вита пара

Екранована вита пара

© М.В.Добролюбова

2

Коаксіальні кабелі

Коаксіальний кабель – це електричний кабель, що складається із центрального мідного дроту і металевого обплетення (екрана), розділених між собою шаром діелектрика (внутрішньої ізоляції) і розміщених у загальній зовнішній оболонці.

© М.В.Добролюбова

3

Оптоволоконні кабелі

Оптоволоконний (волоконно-оптичний) кабель – це кабель, інформація з якого передається не електричним сигналом, а світловим. Головний його елемент – це прозоре скловолокно, по якому світло проходить на величезні відстані (до десятків кілометрів) з незначним послабленням.

Поширення світла в одномодовому та багатомодовому кабелях

1 Центральний (осьовий) елемент
2 Оптичне волокно
3 Пластикові модулі для оптичних волокон
4 Плівка з гідрофобним гелем
5 Поліетиленова оболонка
6 Броня
7 Зовнішня поліетиленова оболонка

@ М.В.Добролюбова

4

Безкабельні канали зв'язку

Радіоканал використовує передачу інформації по радіохвилях. У радіоканалі використовується передача у вузькому діапазоні частот і модуляція інформаційним сигналом сигналу несучої частоти. Головний недолік – поганий захист від прослуховування та слабка завадозахищеність.

Інфрачервоний канал не вимагає з'єднувальних проводів, використовує для зв'язку інфрачервоне випромінювання. Головна перевага в порівнянні з радіоканалом – нечутливість до електромагнітних завад. Інфрачервоні канали поділяться на дві групи: прямої видимості; на розсіяному випромінюванні.

© М.В.Добролюбова 5

Мережеві пристрої

Мережева карта, мережевий контролер, мережевий адаптер, Ethernet-адаптер, NIC (англ. Network interface controller) – периферійний пристрій, що дозволяє комп'ютеру взаємодіяти з іншими пристроями мережі.

Мережевий концентратор або хаб (hub – центр діяльності) – мережевий пристрій, призначений для об'єднання декількох пристроїв Ethernet в загальний сегмент мережі.

Мережевий комутатор або свитч (switch – перемикач) – пристрій, призначений для з'єднання декількох вузлів комп'ютерної мережі в межах одного сегменту.

Маршрутизатор – мережевий пристрій, що приймає рішення про пересилку пакетів мережевого рівня (рівень 3 моделі OSI) між різними сегментами мережі на підставі інформації про топологію мережі і певних правил.

@ М.В.Добролюбова

6

Мережеві технології

Параметри базових варіантів стандартних мереж

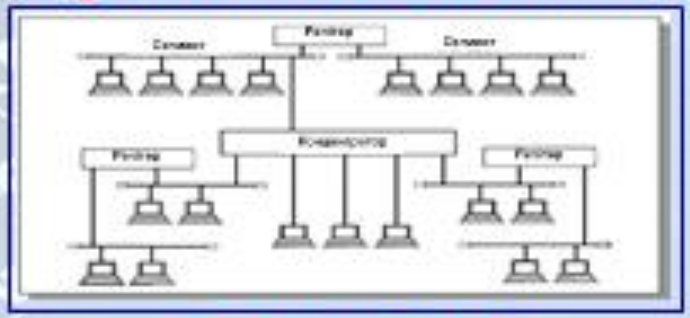
Параметр мережі	*Ethernet*	*Token-Ring*	*Arcnet*	FDDI	100VG-AnyLAN
Стандарт	IEEE 802.3	IEEE 802.5	Datapoint	ISO 9314	IEEE 802.12
Топологія	Шина	Кільце	Шина	Кільце	Зірка
Швидкість передачі	10 (1000) Мбіт/с	(16) Мбіт/с	2,5 Мбіт/с	100 Мбіт/с	100 Мбіт/с
Довжина	5 км.	120 м	6 км.	20 км.	1 км.
Середовище	КК	ВП	КК	ОВ	ВП
Метод управління	CSMA/CD	Маркер	Маркер	Маркер	Центр
Код	Манчестер	Біфазний	Arcnet	4B/5B	5B/6B
Кількість	До 1024	До 260	До 255	До 1000	До 1024
КК – коаксіальний кабель, ВП – кабель на витих парах, ОВ – оптоволоконний кабель					

@ М.В.Добролюбова

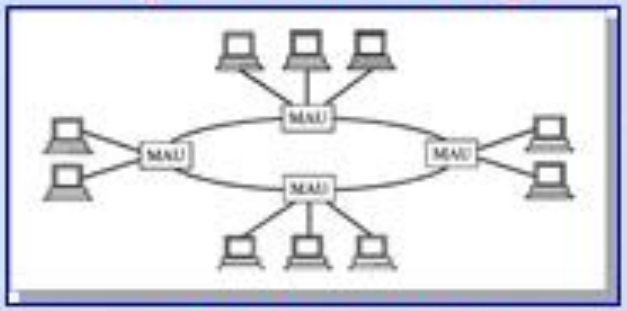
7

Мережеві технології

Мережі Ethernet і Fast Ethernet

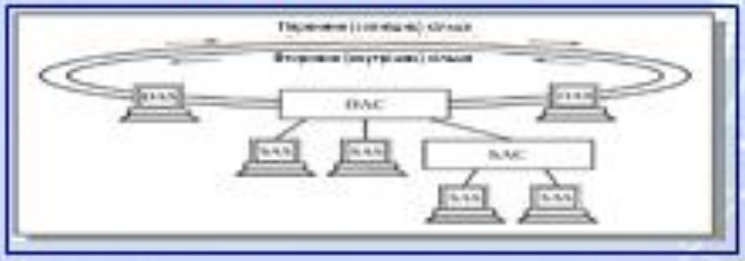


Мережа Token-Ring

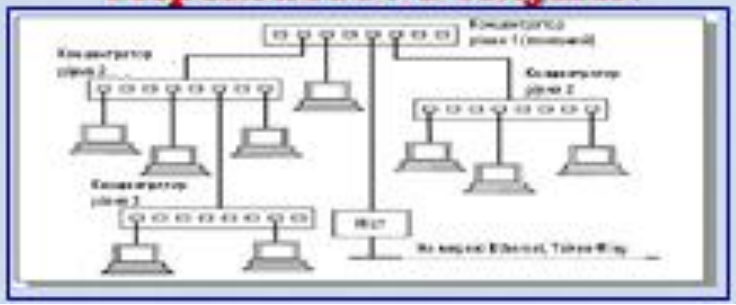


Мережа Arcnet

Мережа FDDI



Мережа 100VG-AnyLAN



@ М.В.Добролюбова

8


Обчислювальна техніка та програмування
РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА.
ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Лекція 13
«Програмні мережеві засоби»
© М.В.Добролюбова

Стандартні мережеві програмні засоби

Одноранговi мережі – мережі, що складаються з рівноправних (з точки зору доступу до мережі) комп'ютерів.

Мережі на основі серверів – мережі, в яких існують лише виділені (dedicated) сервери, що займаються виключно мережевими функціями.

@ М.В.Добролюбова

2

Peer-to-Peer Network

Переваги одноранговихмереж:

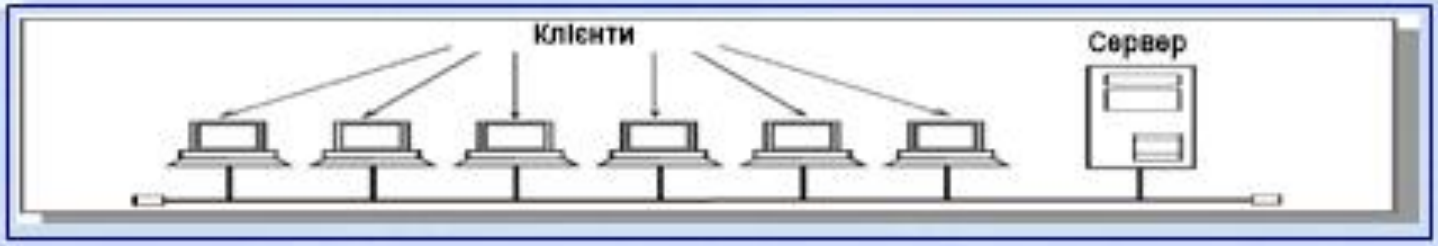
- висока гнучкість;
- проста установка.

Недоліки одноранговихмереж:

- слабка система контролю і протоколювання роботи мережі;
- труднощі з резервним копіюванням розподіленої інформації;
- пошкодження будь-якого комп'ютера-сервера приводить до втрати частини загальної інформації;
- ефективна швидкість передачі інформації часто виявляється недостатньою.

@ М.В.Добролюбова 3

Server-based Network



Переваги мережі на основі сервера:

- надійність;
- висока швидкість обміну.

Недоліки мережі на основі сервера:

- громіздкість в разі невеликої кількості комп'ютерів;
- залежність всіх комп'ютерів-клієнтів від сервера;
- вища вартість мережі.

4


Операційні системи
Windows Server
UNIX
Linux
FreeBSD
OpenBSD
@ М.В.Добролюбова
5


Обчислювальна техніка та програмування
РОЗДІЛ 1
ОБЧИСЛЮВАЛЬНА ТЕХНІКА.
ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Тема 1.2 ПЕРСОНАЛЬНИЙ КОМП'ЮТЕР
Лекція 14
«Використання Internet»
© М.В.Добролюбова

Ключові питання INTERNET

Інтернет (скор. від Interconnected Networks – об'єднані мережі) – глобальна телекомунікаційна мережа інформаційних і обчислювальних ресурсів. Служить фізичною основою для Всесвітнього павутиння.

© М.В.Добролюбова 2

Ключові питання INTERNET

Пол Баран і Дональд Девіс

Тім Бернес-Лі

Підводний кабель

@ М.В.Добролюбова

3

Ключові питання INTERNET

<ім'я комп'ютера (конкретний хост)>.<ім'я локальній мережі>.<ім'я мережі>.<ім'я домена верхнього рівня>

Типи організацій, які застосовуються в США

Ім'я домена	Тип організації
COM	Комерційна
EDU	Система освіти
GOV	Урядова
MIL	Військова
NET	Мережеві служби
ORG	Інші організації

Рівень OSI	Протоколи, приблизно відповідні рівню OSI
Прикладний	DNS, FTP, HTTP, HTTPS, IMAP, LDAP, POP3, L2TP, SNMP, SMTP, SSH, Telnet, XMPP (Jabber)
Сеансовий/Представлення	SSL, TLS
Транспортний	TCP, UDP
Мережевий	BGP, EIGRP, ICMP, IGMP, IP, IS-IS, OSPF, RIP
Канальний	Arcnet, ATM, Ethernet, Frame relay, HDLC, PPP, SLIP, Token ring

OSCAR	BitTorrent
CDDB	Gnutella
MFTP (мережа eDonkey2000)	Skype

@ М.В.Добролюбова

4

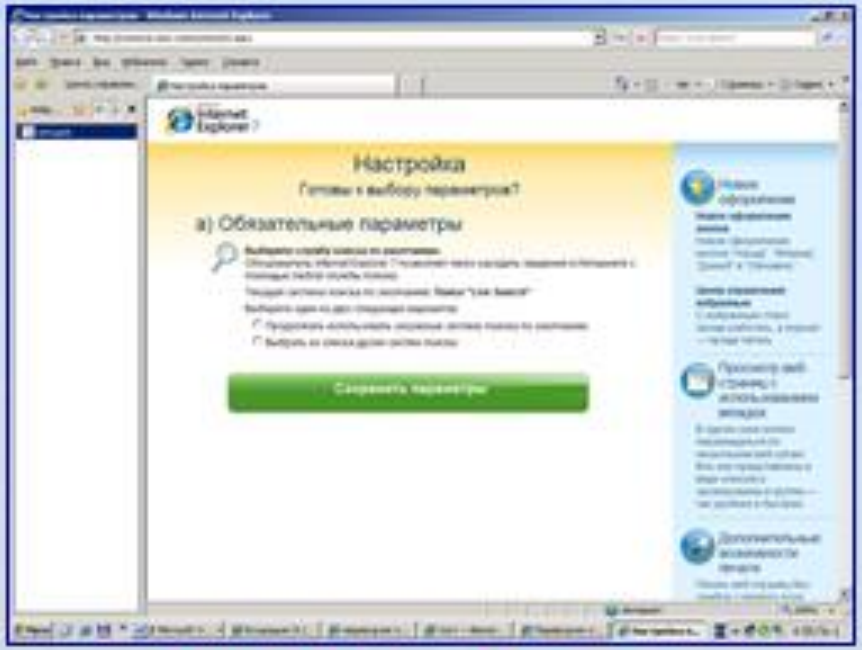
Служби Інтернет

<схема доступу>://<комп'ютер>.<адреса файлу у файловій системі комп'ютера>

Наприклад:

http://home.netscape.com/comprod/index.html

схема доступу (формат передачі) | ім'я хоста | ім'я каталога | ім'я файлу



Вікно MS Internet Explorer

@ М.В.Добролюбова

5

Служби Інтернет

Пошукові системи: yahoo.com, google.com, go.com



Система для пошуку FTP-серверів TILE.NET

Вікна поштового сервера UKR.NET та реєстрації користувача


@ М.В.Добролюбова

6


Обчислювальна техніка та програмування
РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ
Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ
Лекція 15
«Дані та їх характеристики»
© М.В.Добролюбова

Дані

Дані – це інформація, подана у формалізованому вигляді, прийнятному для обробки автоматичними засобами за можливої участі людини. Розрізняють числові, текстові, графічні, звукові та відеодані.

Вхідна інформація – це дані, які потрапляють в комп'ютер для подальшої обробки.

Оперативна (потокова) обробка даних – це обробка комп'ютером даних при будь-якій швидкості їх надходження.

Вихідна інформація – це дані, що видаються комп'ютером після обробки.

Дані в програмуванні – це величини, які опрацьовуються програмою. Вони поділяються на: константи та змінні; скалярні та структуровані; стандартні та дані користувача.

Константи – це величини, що не змінюють своїх значень в ході виконання програми.

Змінні – об'єкти, що можуть приймати різні значення.

Скалярні величини – це прості значення. Скалярний об'єкт може приймати в будь-який момент виконання програми лише одне значення.

Структуровані величини – це величини, які складаються з декількох значень, тобто, одній величині відповідає деякий набір значень одразу.

Операнд – це дані, що піддаються обробці.

Типи даних

Тип даних – характеристика, яка визначає множину припустимих значень, формат їх збереження, розмір виділеної пам'яті та набір операцій, що можна виконувати над даними.

Різновиди типів даних:

- машинні типи даних (біт, байт, слово, подвійне слово);
- прості типи даних (числові – цілі та дійсні типи, логічний (булевий), символьний та байтовий);
- складні типи даних (масиви, множини, рядки, записи, файли, динамічні змінні; вказівники, лінійні списки (стеки, черги), нелінійні списки (двійкові дерева, несиметричні дерева, тексти, графи), процедурний тип, об'єкти).

Складні типи даних – це типи, які складаються з елементів, що належать до простих типів.

Абстрактний тип даних – це тип даних, який визначає набір функцій, незалежних від конкретної реалізації типу, для оперування його значеннями.
В програмуванні абстрактні типи даних зазвичай подаються у вигляді інтерфейсів, які приховують відповідні реалізації типів.

Векторний тип даних – це тип даних, який будується на основі простих типів даних. Усі елементи векторного типу даних розміщенні один за одним, у межах створеного об'єкту.

Різновиди складних типів даних

Масив (англ. array) – це сукупність елементів переважно одного типу даних, впорядкованих за індексами, які зазвичай репрезентовані натуральними числами, що визначають положення елемента в масиві.

Види масивів:

- одновимірні (вектор);
- багатовимірні (таблиця).

	Column 1	Column 2	Column 3	Column 4
Row 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 3	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Переваги та недоліки масивів:

- ефективність операцій;
- збереження в пам'яті;
- індекси в масивах;
- збереження багатовимірних масивів.

Найпоширеніші способи організації масивів в пам'яті:

- розташування «рядок за рядком»;
- розташування «стовпчик за стовпчиком»;
- масив з масивів.

@ М.В.Добролюбова 4

Різновиди складних типів даних

Приклад:

Зберігати двовимірний масив наступного виду:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

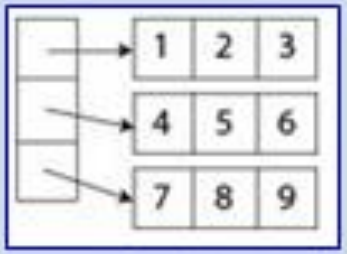
1) Розміщення «рядок за рядком»

1 2 3 4 5 6 7 8 9

2) Розміщення «стовпчик за стовпчиком»

1 4 7 2 5 8 3 6 9

3) Масив з масивів – багатовимірні масиви репрезентуються одновимірними масивами вказівників на одновимірні масиви



@ М.В.Добролюбова

5

Різновиди складних типів даних

Рядок (англ. string) або рядковий тип даних – це тип нечисловий даних, значеннями якого є довільна послідовність (рядок) символів алфавіту.

Основні проблеми в машинному поданні рядкового типу:

- істотний розмір;
- розмір змінюється з часом.

Підходи надання рядків в пам'яті комп'ютера:

1) Представлення масивом символів.
2) Метод «завершального байту».
3) Представлення у вигляді списку.

Запис (record) – це структурований тип даних, призначений для обробки даних, що складаються з полів даних різних типів, та збереження їх в оперативній пам'яті. Розрізняють фіксовані (звичайні) та варіантні записи.

```
type <ім'я запису>=record
<список імен полів 1> <тип полів 1>;
<список імен полів n> <тип полів 1>;
end;
```

Записи з варіантами (варіантними полями) – це записи, в яких можна задавати тип утримуючого визначення декількох варіантів структури.

```
type <ім'я запису> record
    case <поле вибору> <ім'я типу> of
    <список констант вибору 1> (<поле...:тип>:... );
end;
```

© М.В.Добролюбова 6

Різновиди складних типів даних

Множина (англ. set) – це набір однотипних елементів порядкового типу, номер якого у відповідному типі не виходить за границі діапазону 0...255.

Множинна змінна – це змінна, яка зберігає не самі значення її елементів, а лише інформацію про те, присутній або відсутній конкретний елемент в заданій множині.

Множинні константи – це константи, які визначаються шляхом переліку в квадратних дужках через кому або з використанням інтервалу всіх елементів множини.

Файл – це структурований тип даних. Використовуючи ідентифікатор змінної файлового типу можна отримати доступ до файлу – сукупності даних необмеженого об'єму.

Динамічна змінна – це змінна в програмі, місце в оперативній пам'яті під яку виділяється під час виконання програми.

Вказівник (англ. pointer) – це змінна, діапазон значень якої складається з адрес комірок пам'яті або спеціального значення – нульової адреси.

Лінійний список – це екземпляр абстрактного типу даних, що формалізує концепцію впорядкованої множини елементів. Лінійний список з однорівневою структурою даних.

Нелінійний список – це багаторівнева структура даних, яка не впорядковує дані послідовно, а є впорядкованою у відсортованому порядку.

Процедурний тип даних – це тип, призначений для збереження посилань на процедури або функції.

Об'єкт – це деяка сутність в цифровому просторі, яка має певні стан і поведінку, властивості (атрибути) і операції над ними (методи).

© М.Е.Добролюбова 7


Обчислювальна техніка та програмування
РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ
Тема 2.1. СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ
Лекція 16
«Модель, алгоритм, програма»
© М.В.Добролюбова

Життєвий цикл програмного забезпечення

1. Аналіз вимог
2. Визначення специфікацій
3. Проєктування
4. Кодування
5. Тестування
6. Супровід

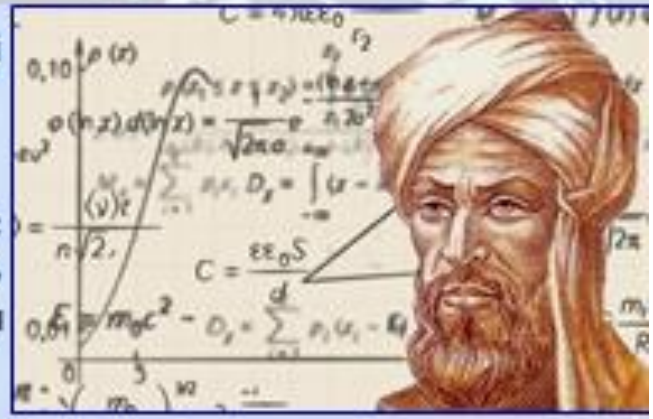
```
public void rozrahunok() {
    D = Math.Pow(b, 2) - 4 * a * c;
    if (D > 0 || D == 0) {
        x1 = (-b + Math.Sqrt(D)) / (2 * a);
        x2 = (-b - Math.Sqrt(D)) / (2 * a);
        Console.WriteLine("x1= {0}\n x2= {1}", x1, x2);
        Console.ReadKey();
    }
    else {
        Console.WriteLine("Дійсних коренів немає");
        Console.ReadKey();
    }
}
```

@ М.В.Добролюбова

2

Алгоритм

Алгоритмізація – найважливіший етап у процесі вирішення завдань на ЕОМ.



Алгоритм – це зрозумілий і точний припис (вказівка) виконавцю зробити певну послідовність дій над вхідною інформацією для вирішення поставленого завдання.

Властивості алгоритму:

1) Дискретність
2) Визначеність
3) Результативність
4) Масовість
5) Інваріантність

@ М.В.Добролюбова

3

Способи запису алгоритмів

Причини неправильного розуміння алгоритму

1) Неоднозначність термінології, що використовується

2) Неправильне розуміння алгоритму, викликане недостатньою деталізацією його опису.

@ М.В.Добролюбова 4

Способи запису алгоритмів

Примітиви

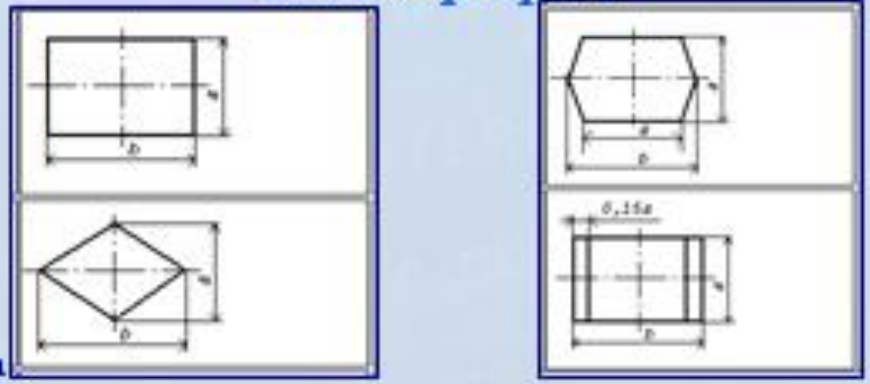
Примітиви – це чітко визначений набір складових блоків, з яких можуть конструюватися подання алгоритмів.

Словесний запис

Алгоритм Евкліда для пошуку найбільшого загального дільника

1. Задати два числа.
2. Якщо числа рівні, то відповідь дорівнює одному з чисел, інакше перейти до п. 3.
3. Визначити більше з двох чисел.
4. Замінити більше число на різницю більшого і меншого чисел.
5. Перейти до п. 2.

Схеми програм



@ М.В.Добролюбова

5

Способи запису алгоритмів

Псевдокоди

алг – назва алгоритму,

нач … **кон** – початок та кінець алгоритму

для *i* **от** *1* **до** *10* **вып** – зазначення (вказівка) на серію дій

если *a>0* – перевірка умови
то *c:=0*
иначе *c:=1*
конесли

пока *i <10* – команда повтору
нц
i:=i+1
вывод *i*
кц

ім'я ← вираз
Підсумок ← Ціна + Податок

if (умова) then (дія) else (дія)
if (рік високосний) then (розділити підсумок на 366) else (розділити підсумок на 365)

while (умова) do (дія)
while (є квитки, які можна продати) do (продати квитки)

@ М.В.Добролюбова


Способи запису алгоритмів
Мови програмування
КЛАСИФІКАЦІЯ МОВ ПРОГРАМУВАННЯ
За поколіннями
1GL
2GL
3GL
4GL
5GL
За рівнем
Низького рівня (машинні коди, асемблери)
Високого рівня (Лісп, Бейсик, Пролог)
За видом програмування
Об'єктно-орієнтовані (JavaScript, Perl)
Процедурні (Matlab, Оберон)
Функціональні (Lisp, Clean)
Логічні (Prolog)
Структурні (BASIC, Fortran)
Мультипарадигмальні (C#, Python)
За видом обробляємої інформації
Символьні (C++)
Обчислювальні (Haskell)
За видом програмування
Операторні данимеи (1GL, 2GL)
Операторні данимеи (3GL, 4GL)
Операторні данимеи (5GL)
@ М.В.Добролюбова
7


Способи запису алгоритмів
Мови програмування
Процедурні
Непроцедурні
SmallTalk
QBE
Fort
Машинно-залежні (низького рівня)
Машинно-незалежні (високого рівня)
Автокоди
Асемблер
Ada
Бейсик
Паскаль
Фортран
GPSS
Java
Delphi
Visual Basic
C++
Python
@ М.В.Добролюбова
8

Способи запису алгоритмів

Тестовий приклад

Завдання. **Розв'язати квадратне рівняння виду** $ax^2 + bx + c = 0$.

Процес створення програми

I ВИЗНАЧЕННЯ ВИМОГ ДО ПРОГРАМИ

Опис вихідної інформації, формулювання вимог до результату:

– вхідними даними для подання є коефіцієнти при ступенях невідомого (a, b, c);

– вхідні дані (коефіцієнти рівняння) повинні вводитися з клавіатури в режимі діалогу під час роботи програми;

– вихідні дані програми – значення коренів рівняння;

– якщо рівняння не має коренів, має виводитися відповідне повідомлення.

II РОЗРОБКА АЛГОРИТМА

Визначення послідовності дій, які треба виконати для отримання результату.

** Якщо можливі різні варіанти алгоритму рішення - використовується деякий критерій (швидкість рішення) і обирається відповідний.*

– Вхідні дані: значення коефіцієнтів рівняння.

– За формулою знайти дискримінант.

– Якщо отримане значення 0, обчислити за формулами значення коренів.

Результат етапу розробки алгоритму – його словесний опис і/або блок-схема.

© М.В.Добролюбова 9

Способи запису алгоритмів

Тестовий приклад

Словесний опис

Вхідні дані – це коефіцієнти рівняння: a – якщо друга степінь невідомого; b – якщо перша; c – якщо нульова.

Результат – значення коренів рівняння, які можна позначити $x1$ та $x2$.

Припис:

1. Обчислити значення дискримінанта (d) рівняння за формулою:
$$d = b^2 - 4ac.$$
2. Якщо значення дискримінанта ≥ 0, то обчислити значення коренів рівняння за формулами:
$$x1 = \frac{-b - \sqrt{d}}{2a}; \quad x2 = \frac{-b + \sqrt{d}}{2a};$$
3. Якщо значення дискримінанта < 0, то рівняння не має дійсних коренів.

Наведений припис має всі властивості алгоритму:

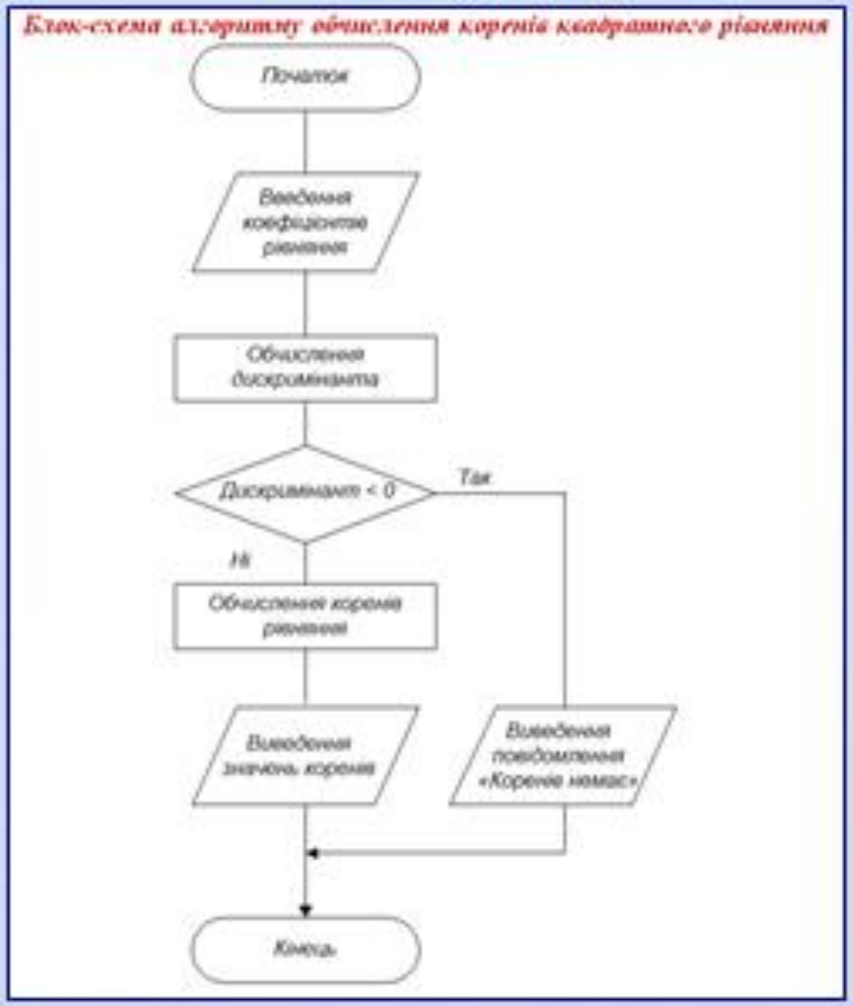
- Однозначністю (у приписі зазначено, як позначаються всі коефіцієнти рівняння, і наведені формули для обчислення значень дискримінанта і коренів рівняння).
- Масовістю (вказані не конкретні значення коефіцієнтів, а наведені формули, в яких використані позначення коефіцієнтів).
- Результативністю (при виконанні припису виходить результат – значення коренів рівняння або виводиться повідомлення про те, що рівняння не має рішення).

При описі алгоритму використовуються загальні поняття: «коефіцієнт» і «корінь рівняння». При розв'язку задачі ці поняття конкретизуються. Не можна знайти корені квадратного рівняння, можна розв'язати тільки конкретне рівняння, тобто треба задати коефіцієнти рівняння.

Способи запису алгоритмів

Тестовий приклад

Блок-схема алгоритму обчислення коренів квадратного рівняння



@ М.В.Добролюбова

11

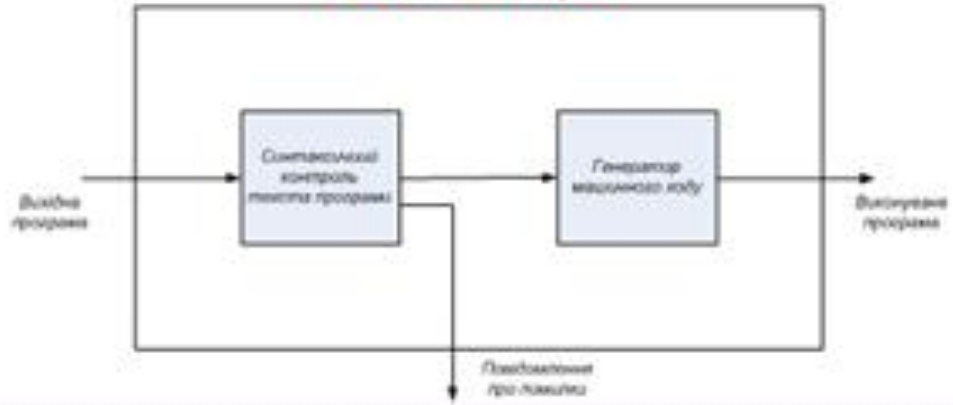
Способи запису алгоритмів

Тестовий приклад

III НАПИСАННЯ ТЕКСТУ ПРОГРАМИ (ЛІСТИНГ) ОБРАНОЮ МОВОЮ ПРОГРАМУВАННЯ

```
public void rozrahunok() {
    D = Math.Pow(b, 2) - 4 * a * c;
    if (D > 0 || D == 0) {
        x1 = (-b + Math.Sqrt(D)) / (2 * a);
        x2 = (-b - Math.Sqrt(D)) / (2 * a);
        Console.WriteLine("x1= {0}\n x2= {1}", x1, x2);
        Console.ReadKey();
    }
    else {
        Console.WriteLine("Дійсних коренів немає");
        Console.ReadKey();
    }
}
```

IV КОМПІЛЯЦІЯ



@ М.В.Добролюбова

12


Обчислювальна техніка та програмування
РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ
Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ
Лекція 17
«Основи аналізу алгоритмів»
© М.В.Добролюбова

Параметри алгоритмів

1) Сукупність можливих вихідних даних
2) Сукупність можливих результатів
3) Сукупність можливих проміжних результатів
4) Правила початку
5) Правило безпосередньої переробки
6) Правило закінчення
7) Правило отримання результату

@ М.В.Добролюбова

2

Принципи аналізу алгоритмів

Аналіз – це ключ до розуміння алгоритмів в ступені, достатньому для їх ефективного застосування до практичних завдань

Кроки в розумінні продуктивності алгоритмів:

- емпіричний аналіз
- математичний аналіз

Проблеми при емпіричному аналізі:

- розробка коректної та повної реалізації
- визначення природи вхідних даних і інших чинників, які безпосередньо впливають на проведені експерименти
- визначення типу вхідних даних для порівняння алгоритмів

Природа вхідних даних:

- реальні дані
- випадкові дані
- помилкові дані

@ М.В.Добролюбова 3

Принципи аналізу алгоритмів

Помилки при виборі алгоритму:

- ігнорування характеристик продуктивності
- занадто велика увага до характеристик продуктивності

Мета математичного аналізу алгоритмів:

- порівняння різних алгоритмів, призначених для вирішення однієї задачі
- для приблизної оцінки продуктивності програми в новому середовищі
- для встановлення значень параметрів алгоритму

Підходи до аналізу алгоритмів на основі наборів вхідних даних:

- припущення, що вхідні дані випадкові, і вивчення середньої продуктивності програми
- розгляд самих незручних даних і вивчення найгіршої продуктивності програми

@ М.В.Добролюбова 4

Принципи аналізу алгоритмів

Математичні функції для опису характеристик продуктивності алгоритмів

Функції, яким пропорційний час виконання алгоритму:

1, logN, N, NlogN, N2, N3, 2N

Спеціальні функції, що перетворюють дійсні числа в цілі:

$\lfloor x \rfloor$ – найбільше ціле, менше або рівне x

$\lceil x \rceil$ – найменше ціле, більше або рівне x

N-е гармонійне число

$$H_N = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{N}$$

@ М.В.Добролюбова

5

Принципи аналізу алгоритмів

О-нотація

О-нотація – математичний запис, що дозволяє відкидати деталі при аналізі алгоритмів

Визначення. Кажуть, що функція $g(N)$ має порядок $O(f(N))$, якщо існують такі постійні c_0 і N_0, що $g(N) < c_0 f(N)$ для всіх $N > N_0$

Причини використання О-нотації:

- обмеження помилки, що виникає при відкиданні малих доданків в математичних формулах
- обмеження помилки, що виникає при ігноруванні частин програми, які вносять невеликий вклад в аналізуєму суму
- класифікація алгоритмів по верхніх границях їх загального часу виконання

Поліноміальний алгоритм – це алгоритм, у якого часова складність дорівнює $O(p(N))$, де $p(N)$ – поліном, N – вхідна довжина

Експоненціальний алгоритм – це алгоритм, часова складність якого не піддається вказаній оцінці

@ М.В.Добролюбова 6

Принципи аналізу алгоритмів

Декомпозиція алгоритму

Декомпозиція – поділ цілого на частини – це науковий метод, який використовує структуру завдання і дозволяє замінити рішення однієї великої задачі рішенням серії менших завдань, нехай і взаємопов'язаних, але більш простих

Правила декомпозиції:

- кожне розчленовування утворює свій рівень
- вихідна система розташовується на нульовому рівні; після її розчленування виходять підсистеми першого рівня; розчленування цих підсистем або деяких з них призводить до появи підсистем другого рівня тощо

Ознаки декомпозиції:

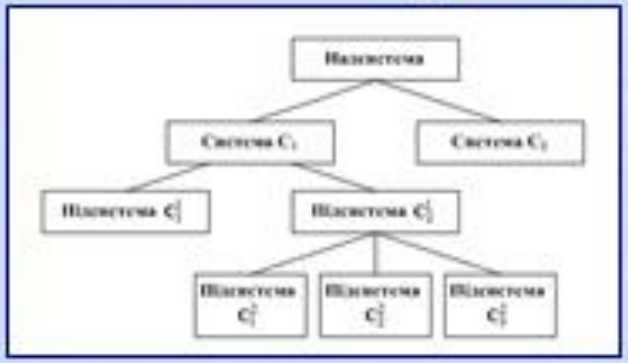
- функціональне призначення частин
- конструктивна будова
- структурні ознаки
- види етапів і процесів
- предметні характеристики
- інші

Глибина декомпозиції – це число рівнів ієрархії

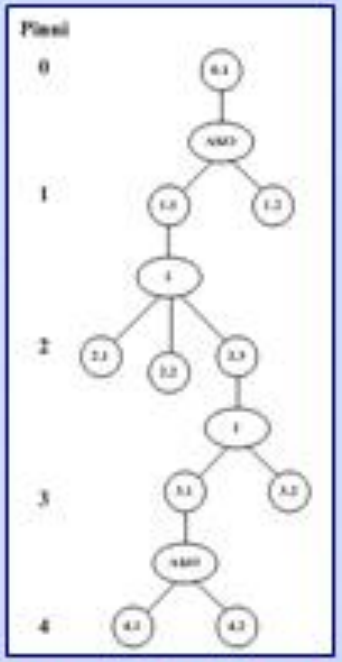
@ М.В.Добролюбова 7

Принципи аналізу алгоритмів

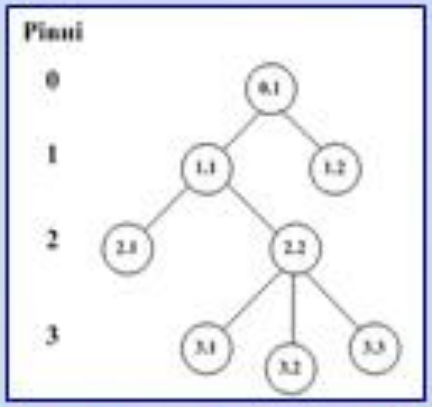
Декомпозиція алгоритму



Приклад ієрархічної структури (блок-схема)



Приклад І-АБО-дерева



Граф структури системи (І-дерево)

@ М.В.Добролюбова

8

Принципи аналізу алгоритмів

Алгоритмічні похибки

Похибка дискретизації (sampling error) – це похибка, яка виникає при дискретизації неперервної задачі

Похибка округлення (rounding error) – це похибка, яка виникає завдяки скінченній довжині розрядної сітки комп'ютера, що призводить до наближеного представлення дійсних чисел

Похибка зрізу (truncation error) – це похибка, що виникає при заміні нескінченних рядів скінченними

Абсолютна похибка (absolute error) – це похибка, що визначається як модуль різниці між точним *A* та наближеним *a* значеннями числа:

$$\Delta = |a - A|$$

Відносна похибка (relative error) – це похибка, що визначається, як:

$$\delta = \frac{\Delta}{|a|}$$

Приведена похибка – це відносна похибка, де в знаменнику замість $|a|$ використовується діапазон a від максимального a_{max} до мінімального a_{min} значення:

$$S = \frac{\Delta}{|a_{max} - a_{min}|}$$

© М.В.Добролюбова

9

Принципи аналізу алгоритмів

Алгоритмічні похибки

Головні властивості арифметичних операцій з похибками:

– *абсолютна похибка суми k* (*k* – будь-яке ціле число) наближених чисел не перебільшує суми абсолютних похибок цих чисел

$$\Delta\left[\sum_{i=1}^{k} a_i\right] \le \sum_{i=1}^{k} \Delta_i$$

– *відносна похибка суми k* наближених чисел не перебільшує максимальної відносної похибки одного з цих чисел

$$\Delta\left[\sum_{i=1}^{k} a_i\right] \le \max_{1 \le i \le k} \delta_i$$

– *відносна похибка добутку k* наближених чисел не перебільшує суми відносних похибок цих чисел

$$\delta\left[\prod_{i=1}^{k} a_i\right] \le \sum_{i=1}^{k} \delta_i,$$

де $\prod_{i=1}^{k} a_i = a_1 \cdot a_2 \cdot \ldots \cdot a_k$

@ М.В.Добролюбова 10

Принципи аналізу алгоритмів

Алгоритмічні похибки

Локальна похибка – це похибка, що виникає на кожному конкретному кроці обчислювального алгоритму

Глобальна похибка – це похибка, що включає всі похибки, які виникли як на цьому кроці обчислень, так і на попередніх

Поняття, що використовуються при оцінці глобальних похибок:

– максимальна похибка

$$\Delta_{max} = \max_{i=1}^{k} \Delta_i$$

– середня похибка

$$\Delta = \frac{1}{K} \sum_{i=1}^{K} |\Delta_i|$$

– середньо-квадратична похибка

$$\Delta = \sqrt{\frac{1}{K-1} \sum_{i=1}^{K} \Delta_i^2}$$

© М.В.Добролюбова 11

Принципи аналізу алгоритмів

Алгоритмічні похибки

Швидкодія – один з головних критеріїв якості обчислювальних алгоритмів – кількість елементарних обчислювальних операцій, яка потрібна для реалізації алгоритму

Ітераційний алгоритм (iterative algorithm) – це алгоритм, що визначається як багатокроковий, в якому кожний наступний крок виконується на основі попереднього

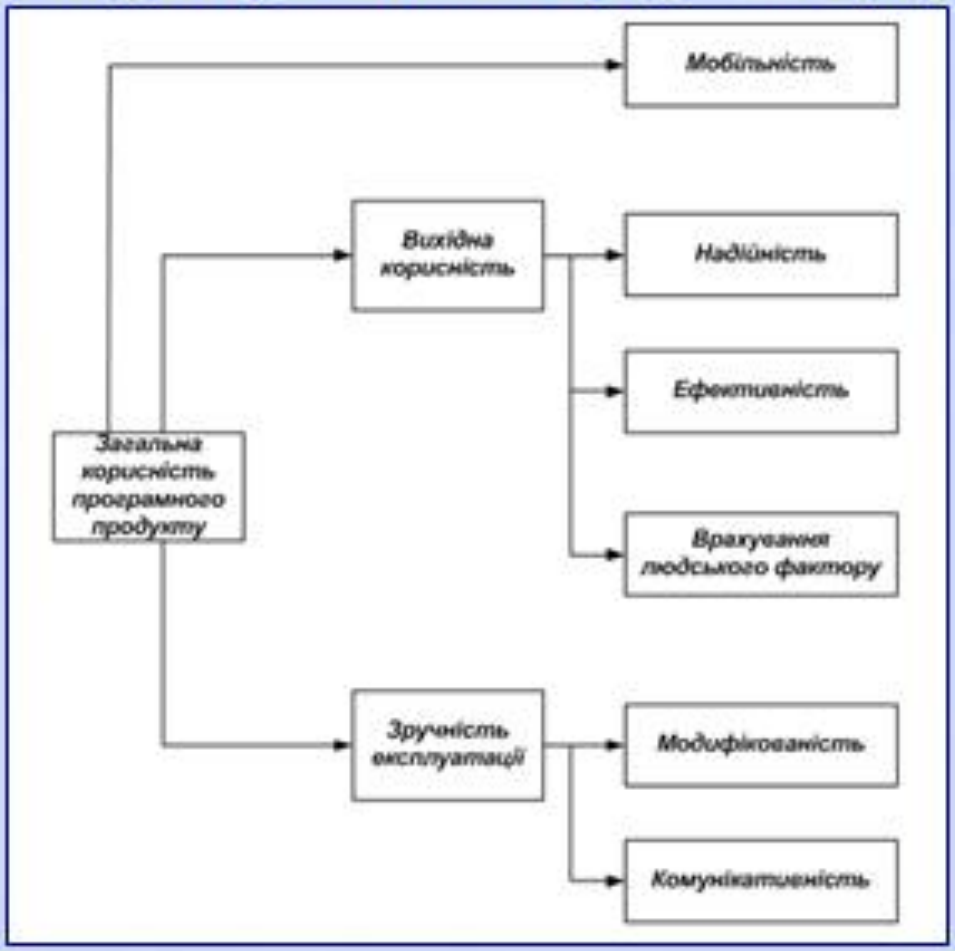
Стійкість алгоритму (stability of the algorithm) – здатність виконувати обчислення і отримувати кінцевий результат із заданою точністю при зміні параметрів алгоритму і вхідних даних в деякій області, яка називається областю стійкості

Збіжність (convergence) – це властивість алгоритму шляхом зміни його параметрів виконувати обчислення з скільки завгодно малою похибкою для заданого класу вхідних даних

Коректність обчислювального методу (the correctness of computational method) – це властивість безперечного існування розв'язку задачі та забезпечення стійкості обчислювального алгоритму, що реалізує цей метод

Критерії якості програм

Дерево характеристик якості програмних продуктів



© М.В.Добролюбова 13

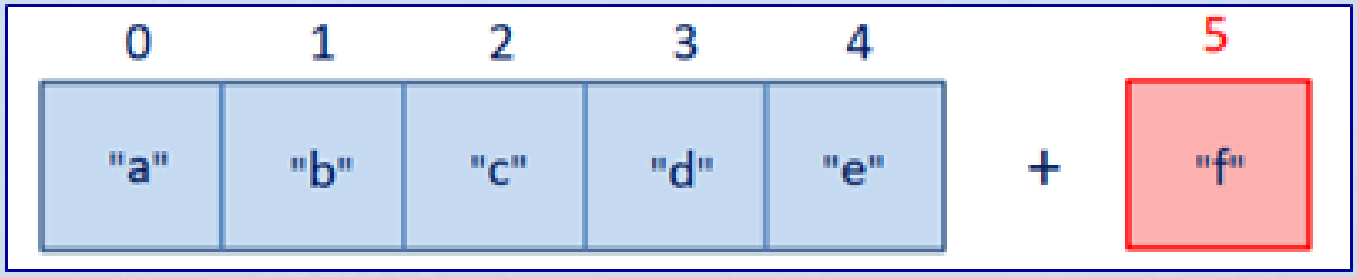

Обчислювальна техніка та програмування
РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ
Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ
Лекція 18
«Динамічні масиви»
© М.В.Добролюбова

Динамічний масив

Масиви в мові C# мають фіксовану довжину, яка не змінюється під час виконання програми.

В деяких випадках кількість елементів в масиві залишається невідомою до моменту виконання програми. В такому разі використовують динамічні масиви на основі класу ArrayList. В класі ArrayList визначається масив змінної довжини, який складається з посилань на об'єкти і може динамічно змінювати свій розмір.

Динамічний масив створюється з початковим розміром і, якщо цей розмір перевищується, то масив автоматично розширюється. При видаленні об'єкта з такого масиву, він автоматично скорочується.



© М.В.Добролюбова

Додавання нового елементу

Алгоритм додавання нового елемента в динамічний масив відрізняється від тих, які використовуються для статичних масивів і списків.

Основні відмінності:

- Можливість додавання нового елемента в середину колекції. В той час як зв'язний список підтримує додавання елементів тільки на початок і в кінець.
- Ступінь зростання складності алгоритму при додаванні елементу в зв'язаний список складає O(1), а при додаванні в динамічний масив може становити O(1) або O(n).

В разі переповнення внутрішнього масиву колекції:

- розмір масиву збільшується;
- виконується копіювання елементи з меншого масиву в більший;
- оновлюється розмір внутрішнього масиву.

@ М.В.Добролюбова 3

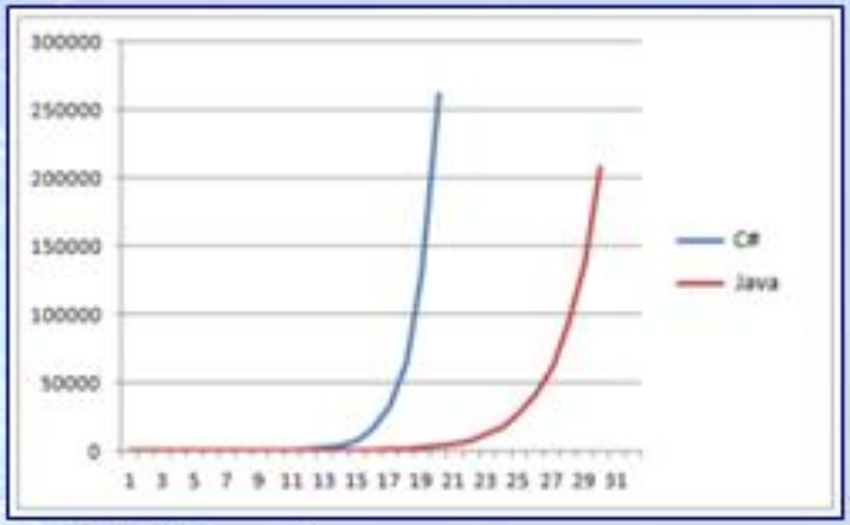
Політики росту

В разі переповнення внутрішнього масиву, його розмірність подвоюється автоматично. Якщо розмірність масиву спочатку дорівнює нулю, то при спробі додати в нього перший елемент розмір збільшується до 4.

```
int size = size == 0 ? 4 : size * 2;
```

В Java зростання розмірності масиву здійснюється повільніше:

```
int size = (size * 3) / 2 + 1;
```



Подвоюючому алгоритму на C# знадобилося 19 операцій перерозподілу пам'яті, щоб перейти границю в 200 000 елементів масиву. Більш повільному алгоритму на Java знадобилося 30 операцій.

@ М.В.Добролюбова

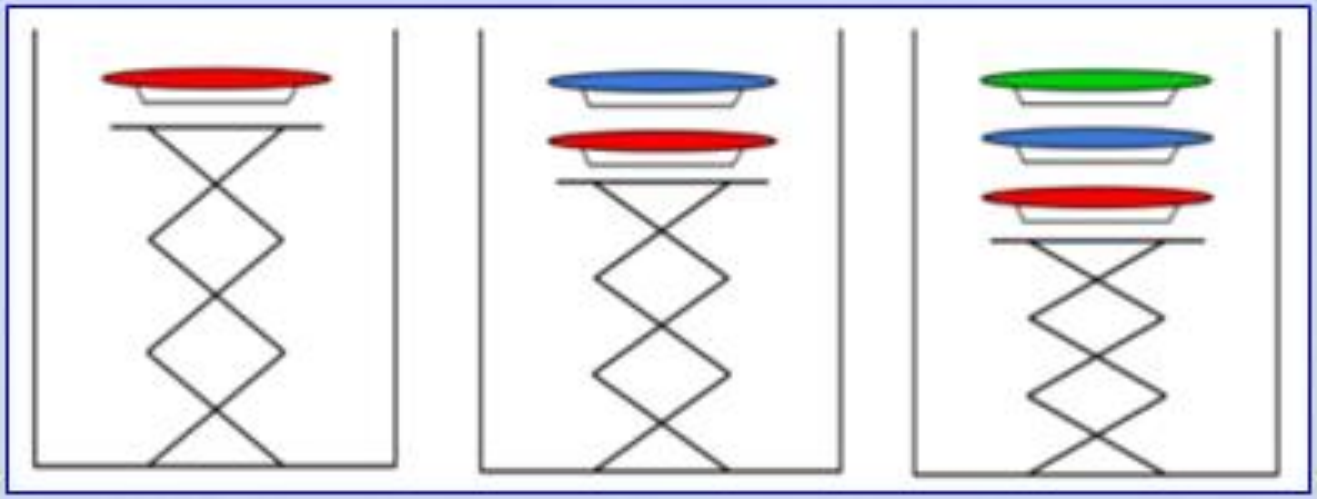
4

Стек. Визначення

Стек – це динамічна структура даних, що представляє собою колекцію елементів, доступ до яких організований за принципом LIFO (Last In – First Out, «останнім прийшов – першим вийшов»).

Всі операції в стеку можна проводити тільки з одним елементом, який знаходиться на верхівці стеку та був введений в стек останнім.

Найчастіше принцип роботи стека розглядається як певна аналогія до стопки тарілок, з якої можна взяти верхню, а додати нову лише зверху попередньої.



© М.В.Добролюбова 5

Використання стеків

Найчастіше стеки використовуються для відстеження місця, куди повинен повернути управління метод після свого завершення. При кожному виклику процедури відповідний запис для неї поміщається в стек. Таким чином, отримуючи запис з стека для останнього виклику (рекурсивного виклику) процедури, можна повернутися до її точки виклику.

@ М.В.Добролюбова 6

Особливості стеку

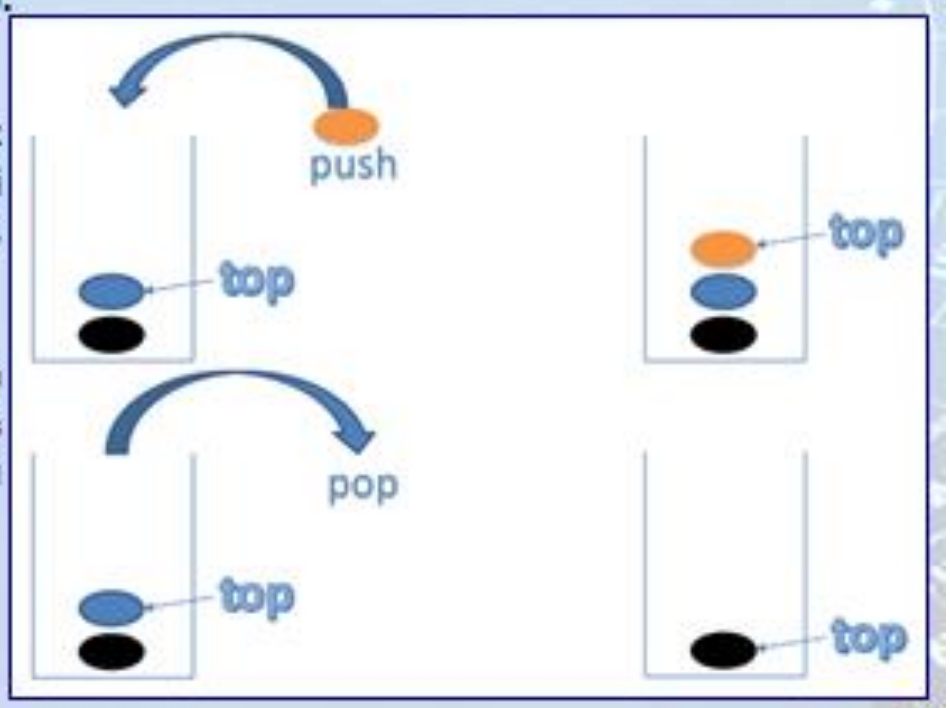
Стек не може бути індексований.

Спеціальні методи для додавання та видалення елементів при роботі із стеком:

- додавання елементу (проштовхування, push);
- видалення елементу (pop);
- читання головного елементу (peek).

При додаванні елементу в стек новий елемент стає головним і формує вказівник на наступний, який був до цього головним.

При видаленні елемента забирається перший елемент в стеку, а головним стає той, на який він до цього вказував.



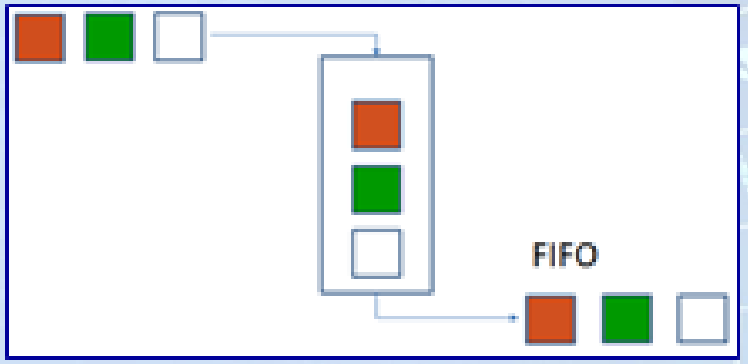
@ М.В.Добролюбова 7

Черга. Визначення

Черга – динамічна структура даних, що є колекцією, доступ до елементів якої здійснюється за принципом «перший прийшов – першим вийшов» (FIFO, First In – First Out).

Додавання елементу можливо лише в кінець черги (позначається словом enqueue).

Вибірка елементу з черги (dequeue) можлива лише спочатку.



Черга. Застосування

Черга в програмуванні використовується, коли потрібно здійснювати якісь дії, виконуючи їх послідовно.

- Буфери даних для різних додатків часто реалізуються на основі черги, куди записуються дані для майбутніх обчислень.
- База даних може підтримувати обмежену кількість одночасних з'єднань, в такому випадку черга може бути використана, щоб дозволити іншим потокам чекати свого з'єднання.

8

Метод Enqueue()

Реалізація черub на основі двозв'язного списку дозволяє використовувати методи класу LinkedList <T>.

Додавання нового елементу в кінець черги еквівалентне додаванню його в початок списку.

```
public void Enqueue(T value)
{
        items.AddFirst(value);
}
```

Метод Dequeue()

Видалення елементу з черги еквівалентне його видаленню з кінця списку.

```
public void Dequeue(T value)
{
        T last = items.Last.Value;
        items.RemoveLast();
        return last;
}
```

@ М.В.Добролюбова 9

Двозв'язна черга

Двозв'язна черга – структура даних, подібна звичайній черзі, елементи в яку можна додавати як в початок, так і в кінець.

Методи і властивості двозв'язної черги:

- EnqueueFirst() - метод додає елемент в початок черги;
- EnqueueLast() - метод додає елемент в кінець черги (кінець списку);
- DequeueFirst() - метод видаляє елемент з початку черги (початку списку);
- DequeueLast() - метод видаляє елемент з кінця черги (кінця списку);
- PeekFirst() - метод повертає перший елемент черги (початок списку);
- PeekLast() - метод повертає останній елемент черги (кінець списку).

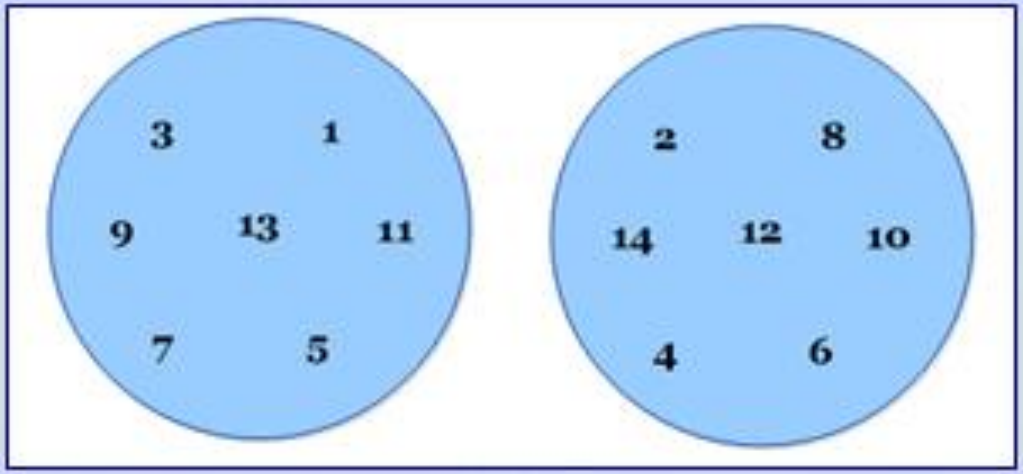
@ М.В.Добролюбова 10

Множина. Визначення

Множина – структура даних, що є колекцією елементів і реалізацією математичної множини, для якої характерні наступні операції: об'єднання, перетин, різниця, симетрична різниця.

Приклади множин:

- Множина парних чисел: [2, 4, 6, 8, 10, 12, 14 ...].
- Множина непарних чисел: [1, 3, 5, 7, 9, 11, 13 ...].

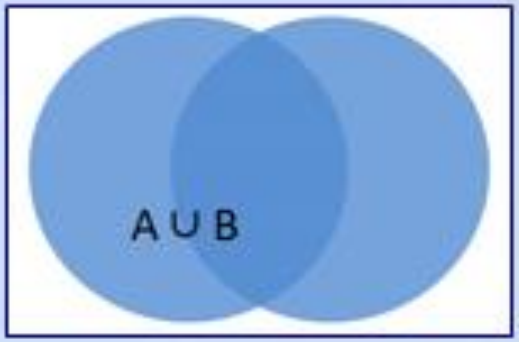


@ М.В.Добролюбова 11

Метод Union()

Об'єднання множин – множина, що містить в собі всі елементи вихідних множин. Після об'єднання двох множин результатом є третя множина, яка містить в собі всі їх елементи.

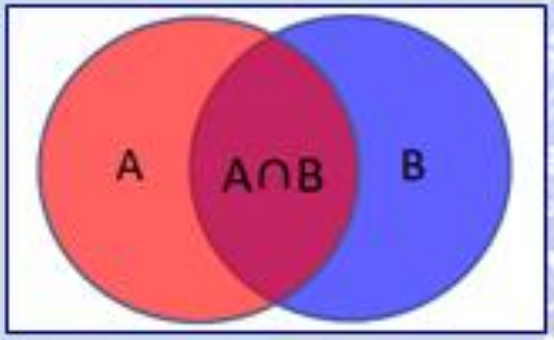
[1, 2, 3, 4] union [3, 4, 5, 6] = [1, 2, 3, 4, 5, 6]



Метод Intersection()

Перетин двох множин – множина, якій належать ті і тільки ті елементи, що одночасно належать всім даним множинам.

[1, 2, 3, 4] intersection [3, 4, 5, 6] = [3, 4]

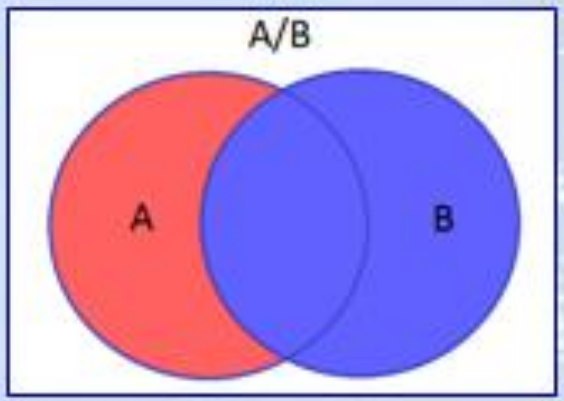


@ М.В.Добролюбова 12

Метод Difference()

Різниця двох множин – множина, в яку входять всі елементи першої множини, що не входять в другу множину.

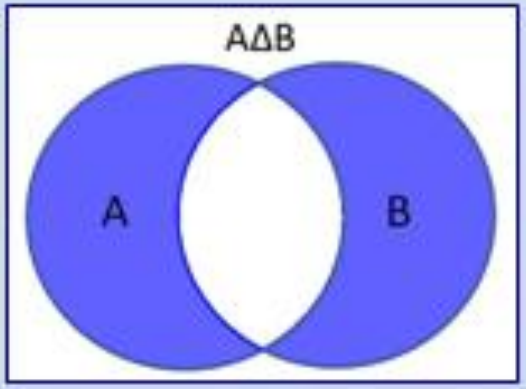
[1, 2, 3, 4] difference [3, 4, 5, 6] = [1, 2]



Метод SymmetricDifference()

Симетрична різниця двох множин – операція, яка виконується над двома множинами, результатом якої є третя множина, що включає в себе елементи, які не належать двом першим.

[1, 2, 3, 4] SymmetricDifference [3, 4, 5, 6] = [1, 2, 5, 6]

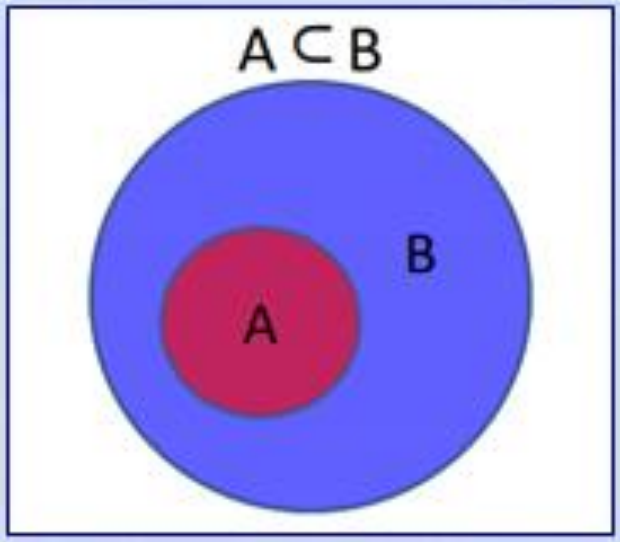


@ М.В.Добролюбова 13

Підмножина

Множина ***A*** називається ***підмножиною*** множини ***B***, якщо будь-який елемент, що належить ***A***, також належить і множині ***B***.

[1, 2, 3] is subset [0, 1, 2, 3, 4, 5] = true



@ М.В.Добролюбова

14


Обчислювальна техніка та програмування
РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ
Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ
Лекція 19
«Базові алгоритми сортування»
© М.В.Добролюбова

Лінійне сортування

Метод лінійного сортування (сортування відбором) – це метод, на кожному кроці алгоритму якого відбирається єдиний елемент при перегляді елементів масиву один за одним

@ М.В.Добролюбова 2

Сортування методом «бульбашки»

Масив є впорядкованим за зростанням його елементів, якщо для деякого масиву *R* встановлені наступні закономірності:

R[1]< R[2], R[2]< R[3], R[3]< R[4], . . ., R[n-1]< R[n]

Правило: якщо кожний елемент масиву менший за безпосередньо наступний за ним елемент, то це є індикатором того, що всі елементи масиву повністю відсортовані

Метод «бульбашки» являє собою багатопрохідний процес попарного порівняння сусідніх елементів, який триває доти, доки не буде зареєстрований цикл, на якому не відбулося ні однієї перестановки

Алгоритм сортування методом «бульбашки»: повторювати наступний процес доти, доки не буде зареєстрований цикл, на якому не було ні однієї перестановки: порівнювати суміжні елементи масиву (тобто *R[1]* з *R[2]*, *R[2]* з *R[3]*, *R[3]* з *R[4]*,…,*R[n-1]* з *R[n]*). Якщо вони стоять за зменшенням значень, то поміняти їх місцями.

@ М.В.Добролюбова 3

Сортування методом вставки

Суть сортування методом вставки: Процес повинен виконуватися багаторазово. За опорний елемент обирається другий елемент списку. Потім, вибирається новий опорний елемент, що знаходиться на одну позицію нижче попереднього, і так до тих пір, поки не буде досягнутий кінець списку

```
procedure Сортування (Список)
N ← 2;
while (N ≤ ДовжинаСписка) do
      {обрати N-й елемент як ОпорнийЕлемент;
       перемістити ОпорнийЕлемент у тимчасове сховище,
       залишивши в списку порожнє місце}
       while (над порожнім місцем є ім'я, і воно > ОпорнийЕлемент) do
            {перемістити ім'я, що знаходиться над порожнім місцем, вниз,
залишивши в
            його попередній позиції порожнє місце;
            помістити ОпорнийЕлемент на порожнє місце в списку;
            N ← N+1}
```

@ М.В.Добролюбова 4


Сортування методом вставки
Початковий список
Fred
Alice
David
Bill
Carol
Відсортований список
Alice
Bill
Carol
David
Fred
@ М.В.Добролюбова
5


Обчислювальна техніка та програмування
РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ
Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ
Лекція 20
«Хеш-таблиці. Дерева»
© М.В.Добролюбова

Хеш-таблиці. Визначення

Хеш-таблиця – це динамічна структура даних, що реалізує інтерфейс асоціативного масиву, тобто дозволяє зберігати пари (ключ, значення). Формат збереження пари ключ-значення дозволяє забезпечити швидкі вставку, пошук та видалення елементів.

Хеш-таблиці можуть використовуватись для збереження браузером даних історії користувача.

© М.В.Добролюбова

2

Ключі та значення

Задля розуміння роботи хеш-таблиць, необхідно розглянути алгоритми додавання нового елементу і пошуку елементу в таблиці.

Наприклад, в таблицю можна записати дані про співробітника компанії, для цього знадобиться створити відповідний ключ і значення для нього.
Дані співробітника – це значення, що додається, а ключ – це ім'я співробітника.

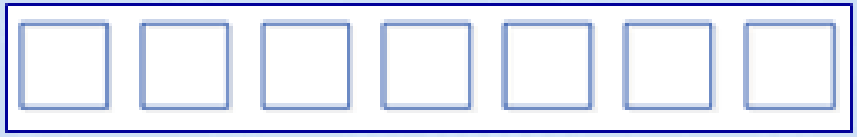
```
Dictionary <string, string> Node = new Dictionary<string, string>()
{
        {"Name": "Aline"},
        {"Date": "17/05/1971"},
        {"Department": "Engineering"}
};
```

@ М.В.Добролюбова 3

Масиви

Для швидкого доступу до елементів хеш-таблиці її значення записують в масив ($O(1)$) або список ($O(n)$).

Властивості масиву для цього випадку: розмірність і ступінь заповнення масиву.

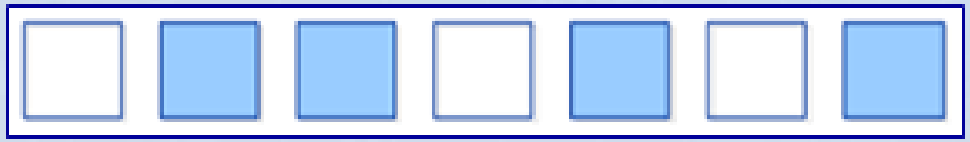


Capacity = 7

Заповнення масиву значеннями відбувається випадковим чином в порядку, що визначається хеш-функцією, яка в якості аргументу приймає значення ключа.

Функція приймає ключ "Aline", а повертає цілочисельний хеш-код. Індекс, за яким цей хеш-код буде розміщений в масиві, є результатом ділення його з залишком на розмірність масиву.

```
int index = hash ("Aline") % Capacity;
```



Колізії

Колізія – два різних вхідних блоки даних X і Y, таких, що: Hash(X) == Hash(Y).

Колізії існують для більшості хеш-функцій, але для «добрих» хеш-функцій частота їх виникнення близька до теоретичного мінімуму.

В деяких окремих випадках, коли множина різних вхідних даних кінцева, можна задати ін'єктивну хеш-функцію, яка за визначенням не має колізій.

Для хеш-функцій, що приймають вхід змінної довжини і повертають хеш постійної довжини, колізії зобов'язані існувати, оскільки хоча б для одного значення хеш-функції відповідна йому множина вхідних даних буде нескінченою – і будь-які два набори даних з цієї множини утворюють колізію.

Колізії

Види вирішення колізій:

1. Метод ланцюжків
2. Лінійне «пробивання»
3. Квадратичне «пробивання»
4. Подвійне хешування

Метод ланцюжків полягає в тому, що в кожній комірці масиву буде міститися самостійна структура даних – така, як зв'язний список.

Лінійне «пробивання» полягає в пошуку вільного місця далі, тобто пошук наступного від опорного вільного місця. Обхід відбувається на основі константи – кроку.

Квадратичне «пробивання» - має таке саме значення, що і лінійне. АЛЕ в даному методі керуються тим, що дані розміщуються поруч в деякі кластери. Тому крок буде збільшуватися пропорційно квадрату кроку.

Подвійне хешування передбачає, що інтервал між комірками фіксований, як при лінійному пробиванні, але, на відміну від нього, розмір інтервалу обчислюється другою, допоміжною хеш-функцією, а значить, може бути різним для різних ключів.

Хеш-функція має різні реалізації. У випадку з C# найкраще використовувати хеш-код об'єкту для подальших маніпуляцій з ним. В загальному випадку можна написати що завгодно, але задача полягає в тому, щоб звести до мінімуму ймовірність збігу хеш-значення для різних ключів.

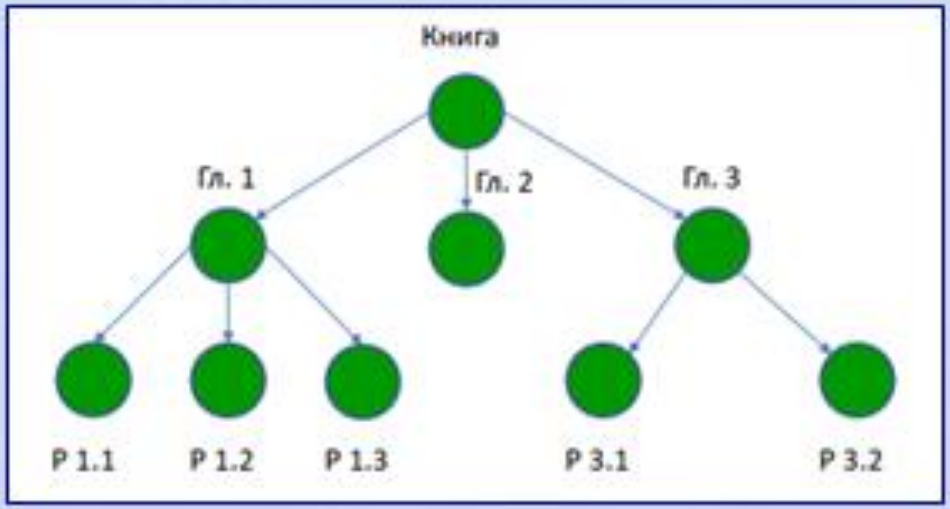
© М.В.Добролюбова 6

Дерево

Метод ланцюжків полягає в тому, що в кожній комірці масиву буде міститися самостійна структура даних – така, як зв'язний список.

Дерево – це структура даних, що є сукупністю елементів, які називаються вузлами (один з них є коренем), і батьківських відношень, що утворюють ієрархічну структуру вузлів. Сини будь-якого вузла впорядковані зліва направо, а шляхи орієнтовані від вузла до нащадків.

Книга
- **Глава 1**
  - Розділ 1.1
  - Розділ 1.2
  - Розділ 1.3
- **Глава 2**
- **Глава 3**
  - Розділ 3.1
  - Розділ 3.2



@ М.В.Добролюбова

7

Дерево

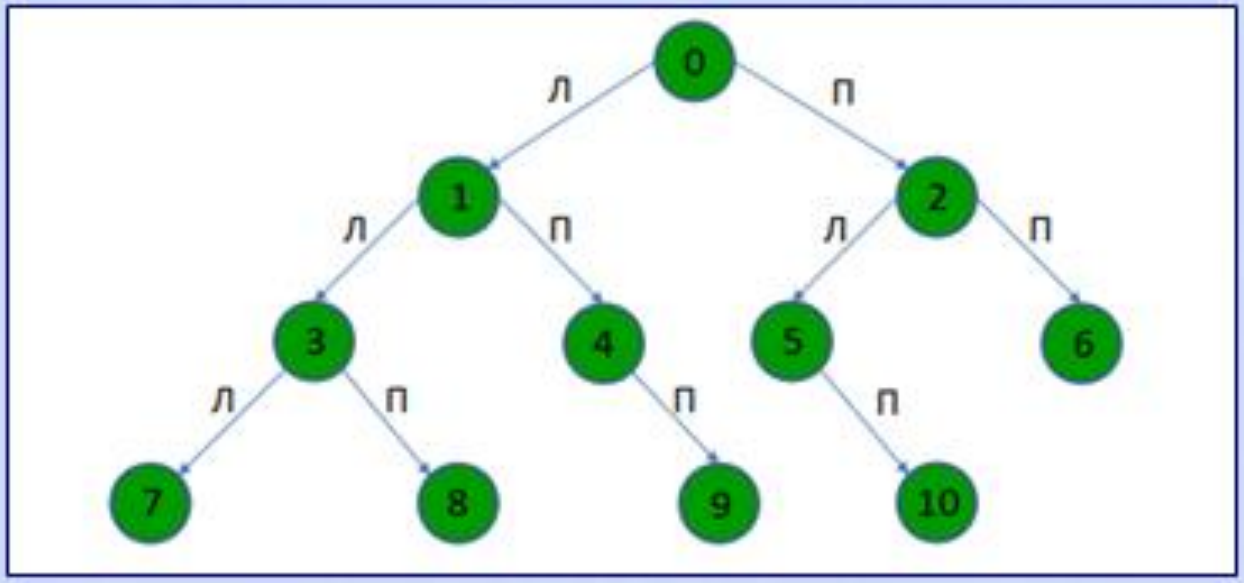
Ключові поняття дерев:

- Шлях з вузла n_1 до вузла n_k – це **послідовність** вузлів $n_1, n_2 \dots n_k$.
- ***Довжина шляху*** – це число, на одиницю менше за число вузлів, які складють цей шлях.
- Якщо існує шлях з вузла *a* у вузол *b*, то в вузол *a* називається **предком** вузла *b*, а вузол *b* – **нащадком** вузла a.
- ***Глибина вузла*** – це довжина шляху від кореня до цього вузла.
- ***Висота двійкового дерева*** – це найбільший шлях від кореня до будь-якого вузла цього дерева.

@ М.В.Добролюбова 8

Двійкове (бінарне) дерево

Двійкове дерево – це деревоподібна структура даних, де кожен вузол має не більше двох нащадків. Коли вузол має два нащадка, кожен з них називається правим і лівим сином відповідно.



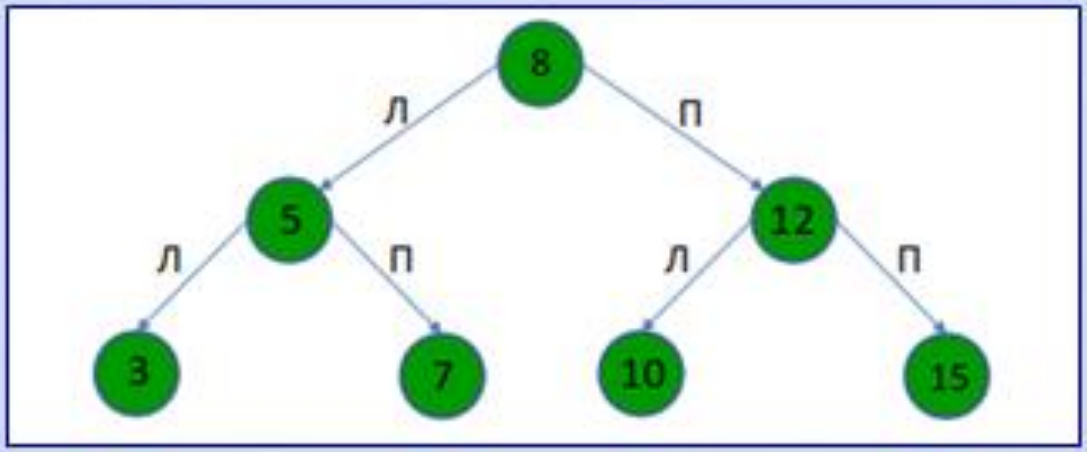
@ М.В.Добролюбова

9

Двійкове (бінарне) дерево пошуку

Двійкове дерево пошуку – це двійкове дерево, для якого виконуються наступні додаткові умови:

- Кожне нове піддерево (ліве і праве) є двійковим деревом пошуку.
- У всіх вузлів лівого піддерева значення ключів менше, ніж значення ключа батьківського вузла.
- У всіх вузлів правого піддерева значення ключів не менше, ніж значення ключа батьківського вузла.



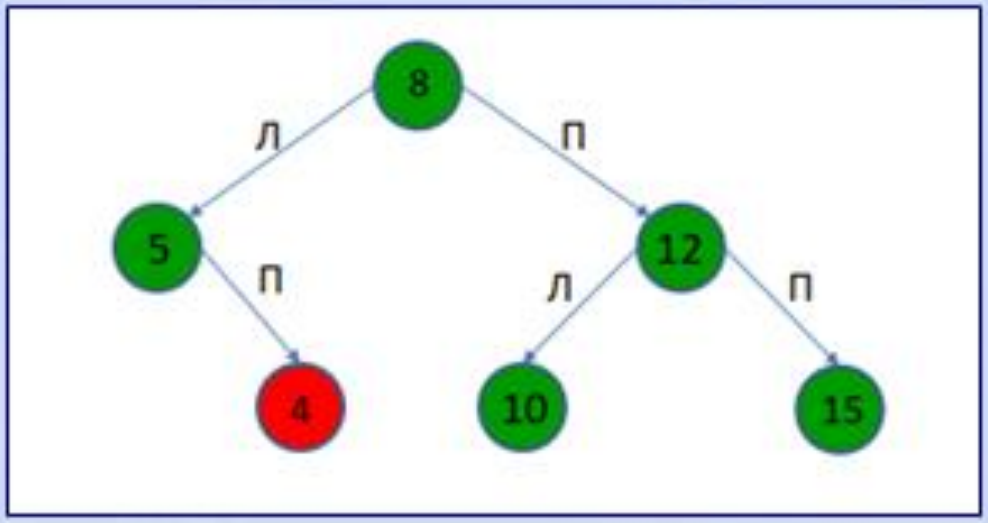
Метод додавання вузла

Двійкове дерево пошуку – це двійкове дерево, для якого виконуються наступні додаткові умови:

Метод додавання вузла найкраще реалізовувати рекурсивно.

Приклад

Додти 4 до існуючого дерева:



@ М.В.Добролюбова

11

Обхід дерева

Алгоритм обходу двійкового дерева передбачає обхід всіх вершин дерева тільки один раз.

Види обходів двійкового дерева:

I. Прямий порядок (англ. preorder), відвідування вузлів батьків до відвідування вузлів нащадків:

1. Зайти в корінь.
2. Зайти в ліве піддерево.
3. Зайти в праве піддерево.

II. Обернений порядок (англ. postorder), відвідування вузлів нащадків до відвідування вузлів їх батьків:

1. Зайти в ліве піддерево.
2. Зайти в праве піддерево.
3. Зайти в корінь.

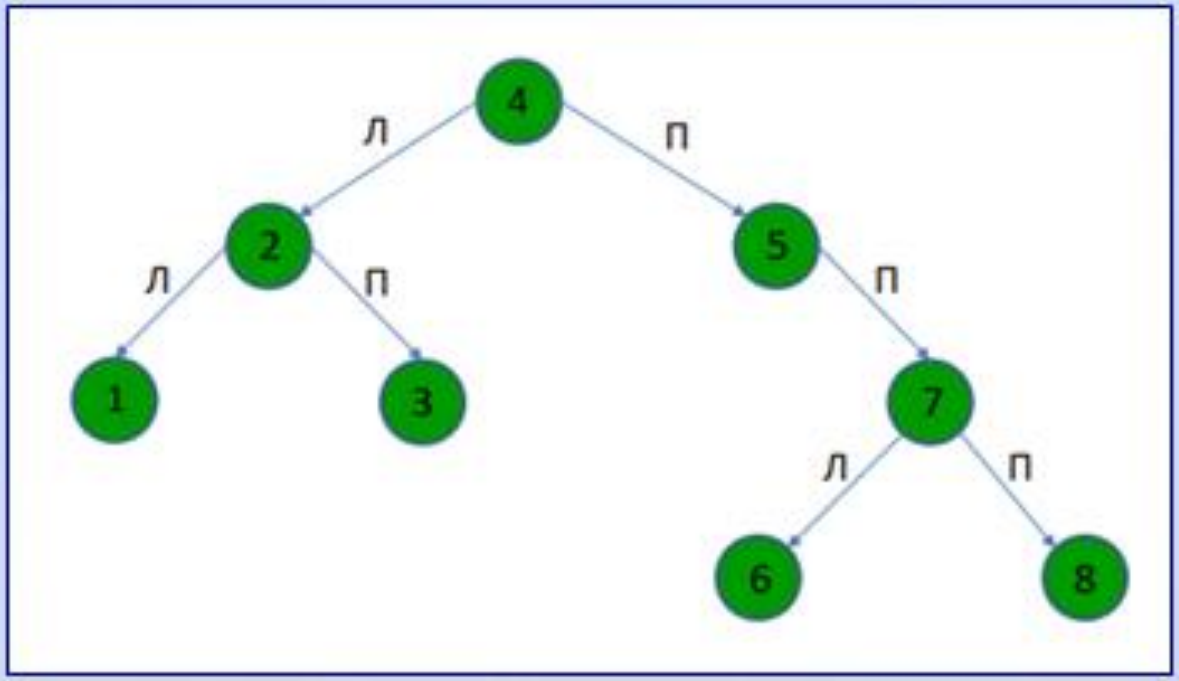
III. Симетричний порядок (англ. inorder):

1. Зайти в ліве піддерево.
2. Зайти в корінь.
3. Зайти в праве піддерево.

@ М.В.Добролюбова 12

Обхід дерева. Прямий порядок

Порядок проходження при прямому обході дерева: 4, 2, 1, 3, 5, 7, 6, 8

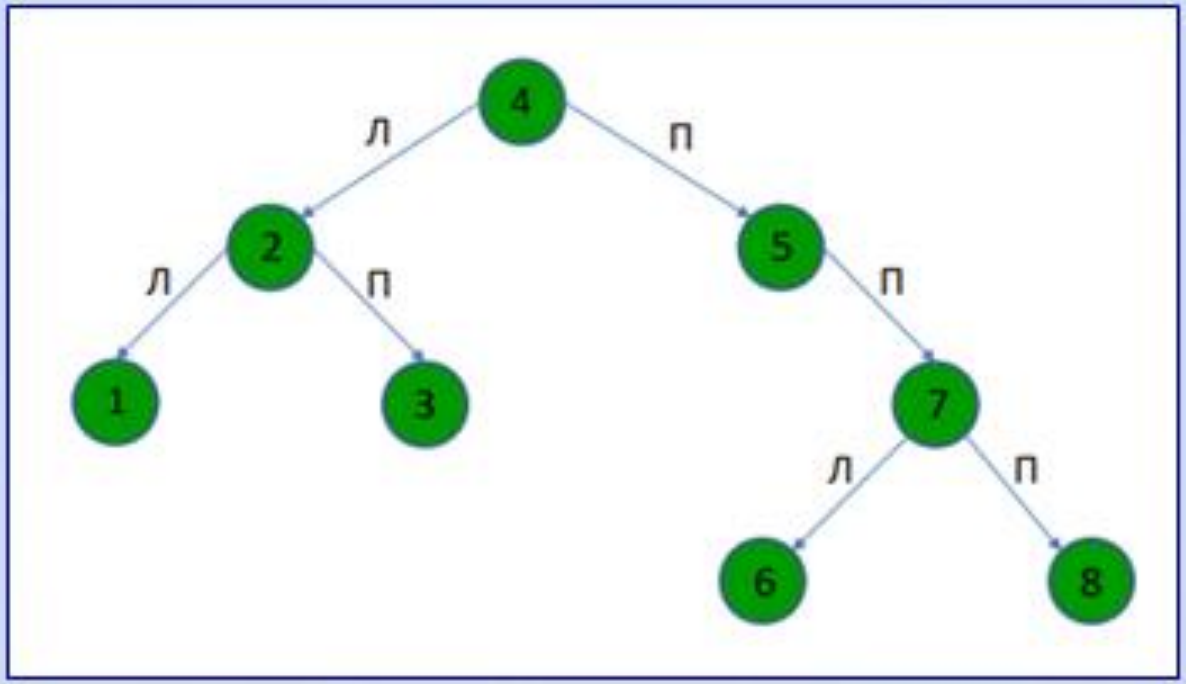


@ М.В.Добролюбова

13

Обхід дерева. Обернений порядок

Порядок проходження при оберненому обході дерева: 1, 3, 2, 6, 8, 7, 5, 4

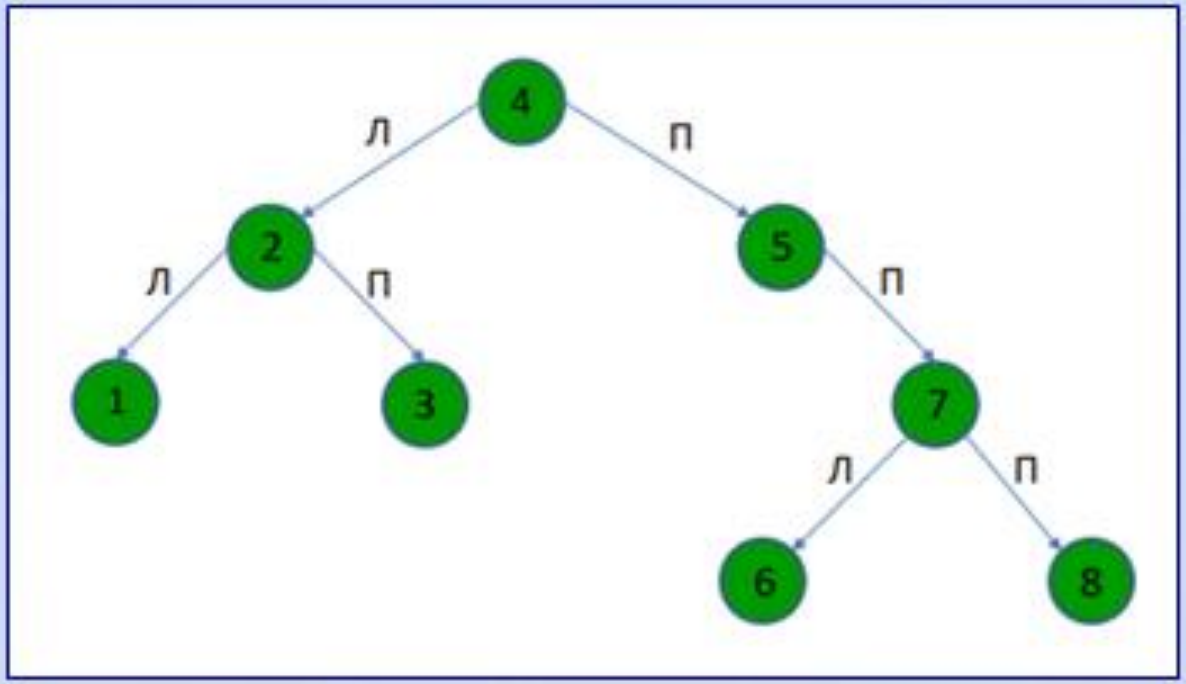


@ М.В.Добролюбова

14

Обхід дерева. Симетричний порядок

Порядок проходження при симетричному обході дерева: 1, 2, 3, 4, 5, 6, 7, 8




Обчислювальна техніка та програмування
РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ
Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ
Лекція 21
«Базові алгоритми пошуку»
© М.В.Добролюбова

Лінійний пошук

Алгоритм послідовного пошуку (лінійний пошук) – знаходження заданого значення довільної функції на деякому її відрізку

```
procedure Пошук (Список, ШуканеЗначення)
if (Список порожній)
      then {повідомити про невдачу}
      else {обрати у якості ПеревіряємеЗначення
            перший елемент Списка}
            while (ШуканеЗначення > ПеревіряємеЗначення
                  та є неперевірені елементи) do
                        {обрати у якості ПеревіряємеЗначення
                        наступний елемент Списка}
             if (ШуканеЗначення = ПеревіряємеЗначення)
                  then {повідомити про успіх}
                  else {повідомити про невдачу}
```

© М.В.Добролюбова 2

Двійковий пошук

Алгоритм двійкового пошуку – це алгоритм знаходження заданого значення у впорядкованому масиві, який полягає у порівнянні серединного елемента масиву з шуканим значенням і повторенні алгоритму для тієї або іншої половини залежно від результату порівняння

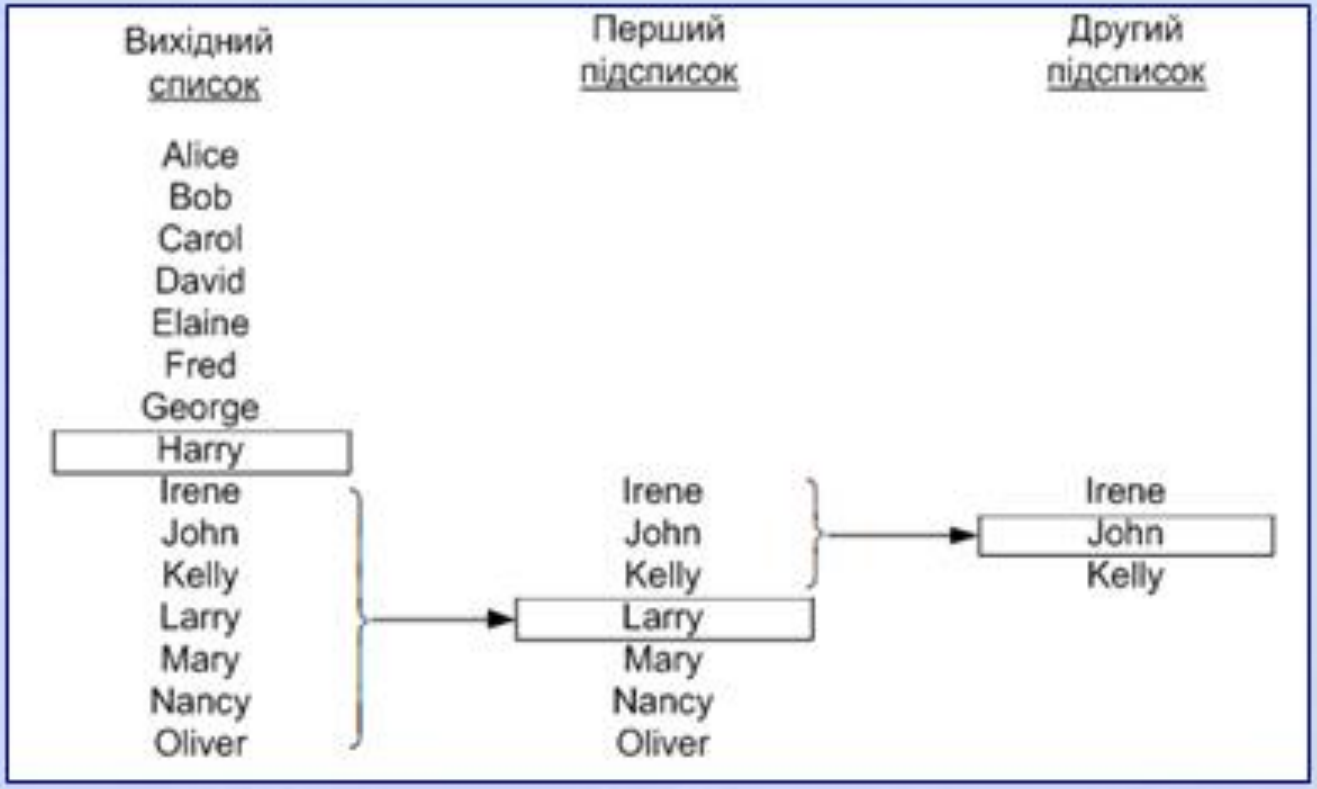
Принцип двійкового пошуку

обрати «середній» елемент як ПеревіряємеЗначення;
виконати набір команд, що відповідають одному з випадків:
Випадок 1: ШуканеЗначення = ПеревіряємеЗначення
{повідомити про успіх}
Випадок 2: ШуканеЗначення < ПеревіряємеЗначення
{застосувати процедуру Пошук для виявлення ШуканеЗначення в верхній частині Списка, та повідомити про результат цього пошуку}
Випадок 3: ШуканеЗначення > ПеревіряємеЗначення
{застосувати процедуру Пошук для виявлення ШуканеЗначення в нижній частині Списка, та повідомити про результат цього пошуку}

© М.В.Добролюбова 3

Двійковий пошук

Застосування стратегії двійкового пошуку для виявлення імені John у впорядкованому списку



Двійковий пошук

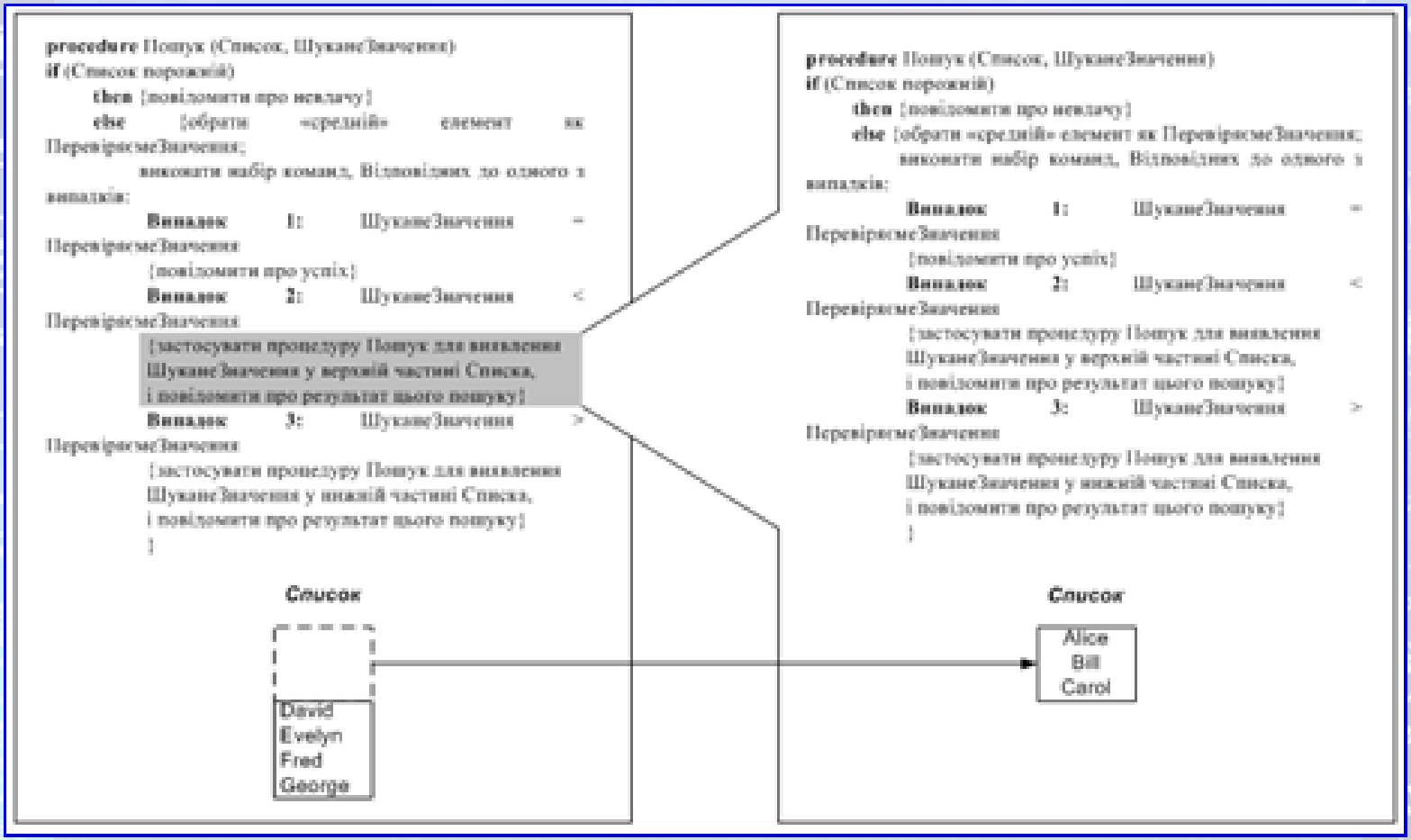
Алгоритм двійкового пошуку

```
procedure Пошук (Список, ШуканеЗначення)
if (Список порожній)
    then {повідомити про невдачу}
    else {обрати «середній» елемент як ПеревіряємеЗначення;
          виконати набір команд, що відповідають одному з випадків:
    Випадок 1: ШуканеЗначення = ПеревіряємеЗначення
               {повідомити про успіх}
    Випадок 2: ШуканеЗначення < ПеревіряємеЗначення
               {застосувати процедуру Пошук для виявлення
               ШуканеЗначення в верхній частині Списка,
               та повідомити про результат цього пошуку}
    Випадок 3: ШуканеЗначення > ПеревіряємеЗначення
               {застосувати процедуру Пошук для виявлення
               ШуканеЗначення в нижній частині Списка,
               та повідомити про результат цього пошуку}
          }
```

Двійковий пошук

Виклик другої копії процедури з її вихідної копії при пошуку запису Bill

```
procedure Пошук (Список, ШуканеЗначення)
if (Список порожній)
    then {повідомити про невдачу}
    else {обрати «середній» елемент як ПерівірюмеЗначення;
          виконати набір команд, Відповідних до одного з випадків:
          Випадок 1: ШуканеЗначення = ПерівірюмеЗначення
                {повідомити про успіх}
          Випадок 2: ШуканеЗначення < ПерівірюмеЗначення
                {застосувати процедуру Пошук для виявлення
                ШуканеЗначення у верхній частині Списка,
                і повідомити про результат цього пошуку}
          Випадок 3: ШуканеЗначення > ПерівірюмеЗначення
                {застосувати процедуру Пошук для виявлення
                ШуканеЗначення у нижній частині Списка,
                і повідомити про результат цього пошуку}
          }
```



© М.В.Добролюбова 6


Обчислювальна техніка та програмування
РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ
Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ
Лекція 22
«Бінарне дерево пошуку. АВЛ-дерево»
© М.В.Добролюбова

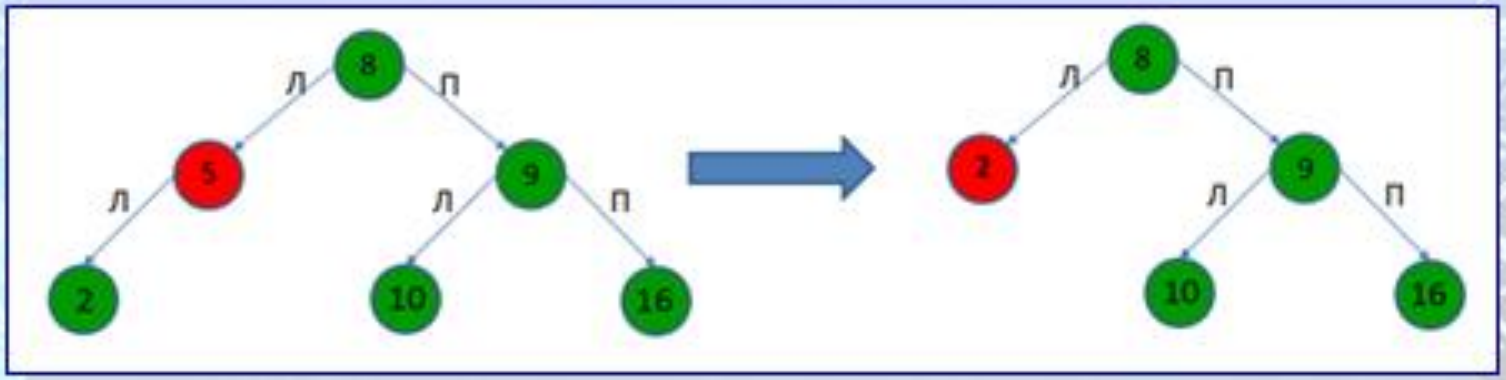
Вдалення вузла з дерева

Варіанти видалення вузла з дерева:

- Перший варіант: вузол, що видаляється, не має правого нащадка.
- Другий варіант: вузол, що видаляється, має правого нащадка, у якого немає лівого нащадка.
- Третій варіант: вузол, що видаляється, має правого нащадка, у якого є лівий нащадок.

Вдалення вузла з дерева. Перший варіант

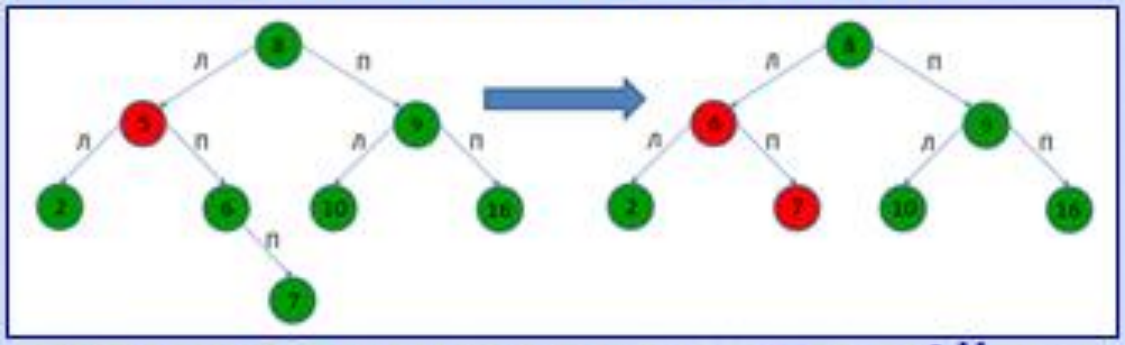
Якщо вузол (5), що видаляється, не має правого нащадка, то достатньо перемістити на його місце лівого нащадка (2).



© М.В.Добролюбова

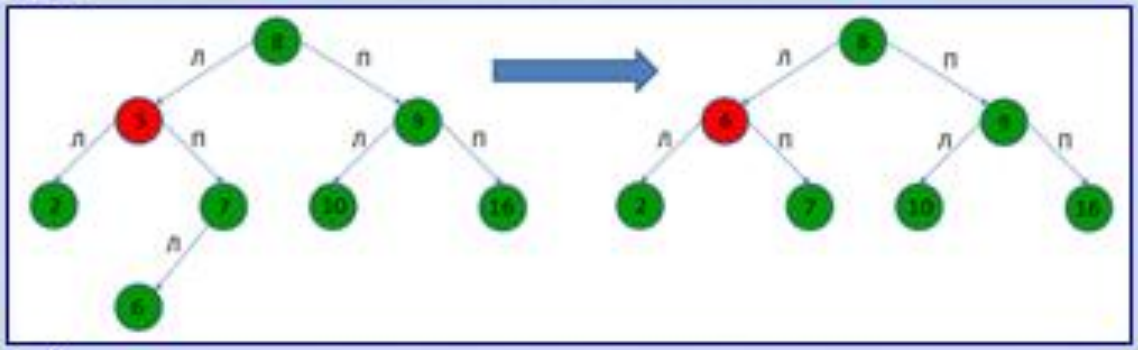
Вдалення вузла з дерева. Другий варіант

Якщо вузол (5), що видаляється, має правого нащадка (6), у якого в свою чергу немає лівого нащадка, то достатньо перемістити їх з дотриманням ієрархії.



Вдалення вузла з дерева. Третій варіант

Якщо вузол (5), що видаляється має правого нащадка (7), у якого в свою чергу є лівий нащадок (6), то крайній лівий нащадок вузла (7) повинен бути переміщений на місце видаленого вузла.



@ М.В.Добролюбова 3

Висота дерева

АВЛ-дерево – збалансоване по висоті двійкове дерево пошуку.

АВЛ-дерево було названо на честь вчених Адельсона-Бєльського Георгія Максимовича і Ландіса Євгена Михайловича, які вперше описали алгоритм і його структуру.

Правила побудови двійкових дерев пошуку:

- кожен вузол може мати не більше двох нащадків (лівий і правий);
- значення, які менше за поточне, розміщуються в лівому піддереві;
- значення, які більше або рівні поточному, розміщуються в правому піддереві.

Додаткова умова для АВЛ-дерева:

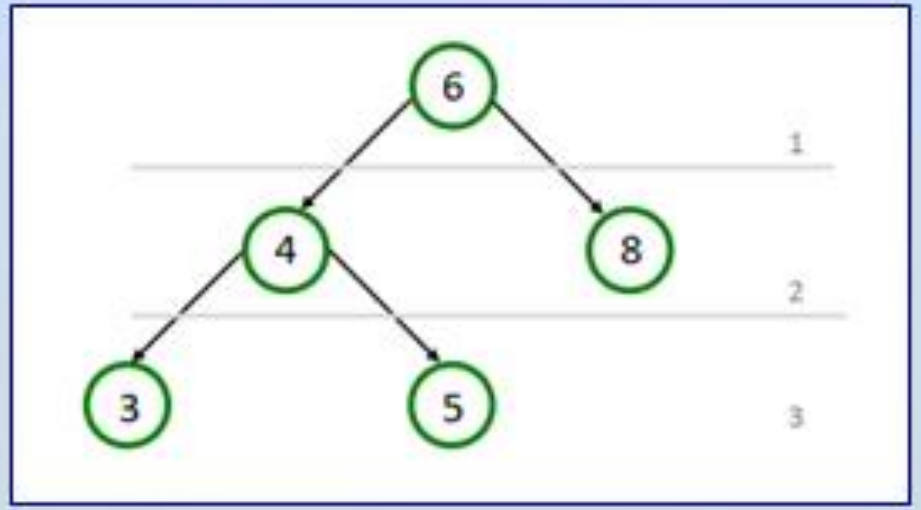
- для будь-якого вузла дерева висота його правого піддерева ніколи не перевищує висоти його лівого піддерева більше, ніж на одиницю.

- Цієї властивості достатньо, щоб висота дерева мала логарифмічну залежність від числа його вузлів.

@ М.В.Добролюбова 4

АВЛ-дерево. Визначення

Вузол «6» не має батьківського вузла, тому він є коренем дерева і знаходиться на першому рівні. Вузли «4» і «8» – це братські вузли, які знаходяться на другому рівні. Вузли «3» і «5» знаходяться на третьому рівні. Використовуючи ці рівні, можна знайти відстань між вузлами «6» і «3», воно дорівнює двом.

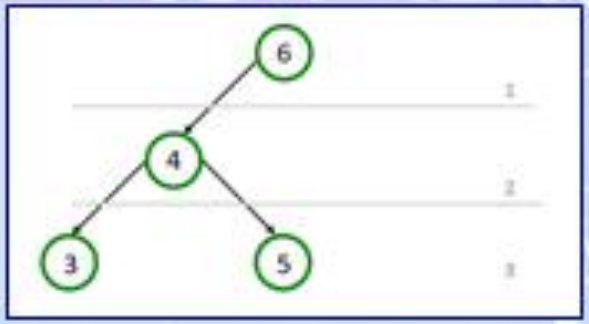


@ М.В.Добролюбова

5

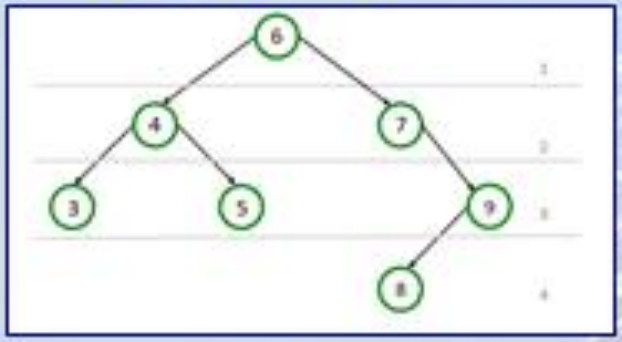
Незбалансоване дерево

Мінімальна висота лівого і правого піддерева не може відрізнятися більше, ніж на одиницю, і це правило характерно для всіх вузлів дерева. Якщо розглянути корінь вузла, то його ліве піддерево має висоту два, а праве – нуль, тому дерево незбалансоване.



Частково збалансоване дерево

Дерево називається повністю збалансованим, якщо кожен вузол дерева збалансований. Вузол «7» незбалансований, оскільки висота правого поддерева два, а лівого – нуль.



@ М.В.Добролюбова

6

Алгоритм балансування

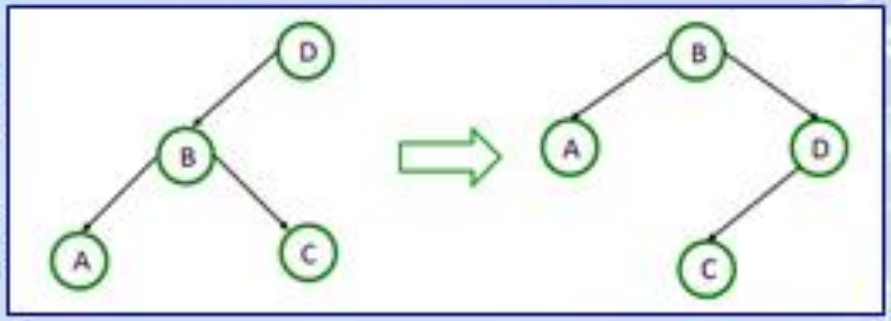
Для визначення чи збалансоване дерево по висоті чи ні, потрібно перевірити висоту правого і лівого нащадка кожного вузла. При необхідності балансування вузли підлягають видаленню, додаванню або обертанню.

Базові алгоритми обертання вузла для АВЛ-дерев:

- праве обертання;
- ліве обертання;
- два змішаних алгоритми, які базуються на перших двох.

Обертання вузла вправо

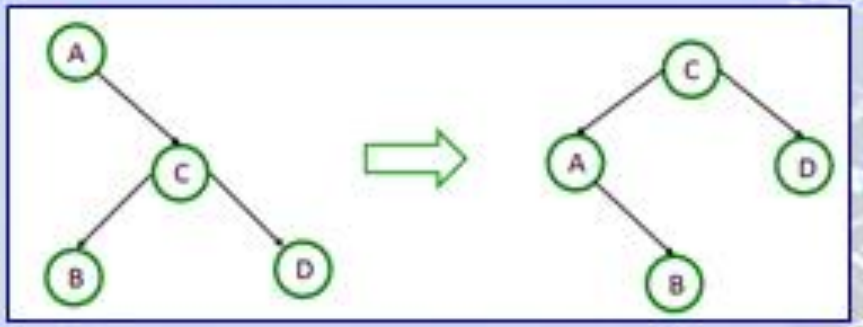
Етапи алгоритму обертання вправо:



- замінити поточний корінь на його лівого нащадка; вузол B стає коренем, а вузол A займає його місце.
- перемістити правого нащадка нового кореня на місце лівого нащадка старого кореня; B -> C to D -> C .
- присвоїти новому кореню (B) в якості правого нащадка старий корінь (D).

Обертання вузла вліво

Етапи алгоритму обертання вліво:



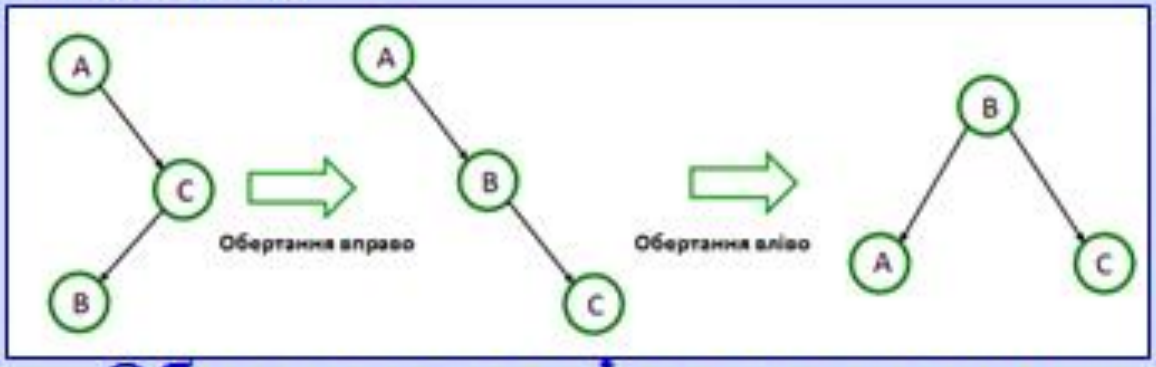
- змінити поточний корінь на його правого нащадка; вузол «C» стає коренем, а вузол «A» – його лівим нащадком.
- перемістити лівого нащадка нового кореня (B) на місце правого нащадка старого кореня; «C» -> «B» to «A» -> «B».
- присвоїти новому кореню (C) в якості правого вузла значення старого кореня (D).

@ М.В.Добролюбова

8

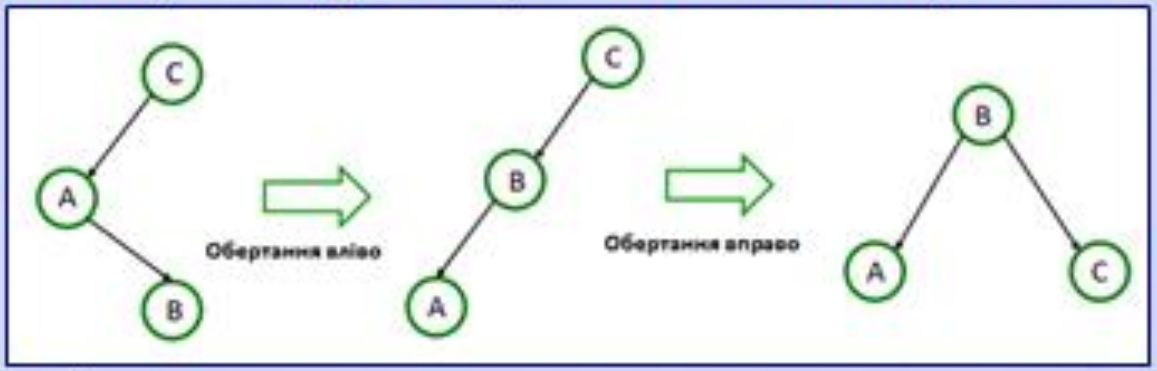
Обертання вправо та вліво

Дана ситуація відрізняється від попередніх, оскільки правий нащадок кореня має лівого нащадка, але не має правого.



Обертання вліво та вправо

Коли корінь дерева має лівого нащадка, який в свою чергу має правого нащадка, але не має лівого, застосовують послідовно обертання вліво, а потім вправо.

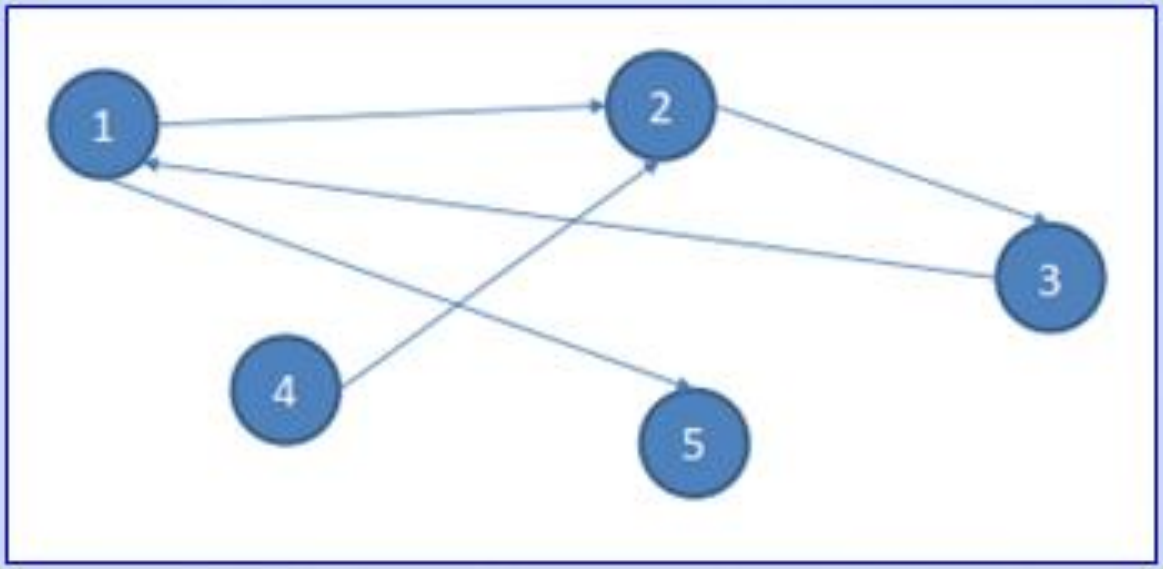


@ М.В.Добролюбова 9


Обчислювальна техніка та програмування
РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ
Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ
Лекція 23
«Вступ в теорію графів»
© М.В.Добролюбова

Теорія графів

Граф (неформальне визначення) – це множина точок і ліній, що з'єднують ці точки, зі стрілками або без них.

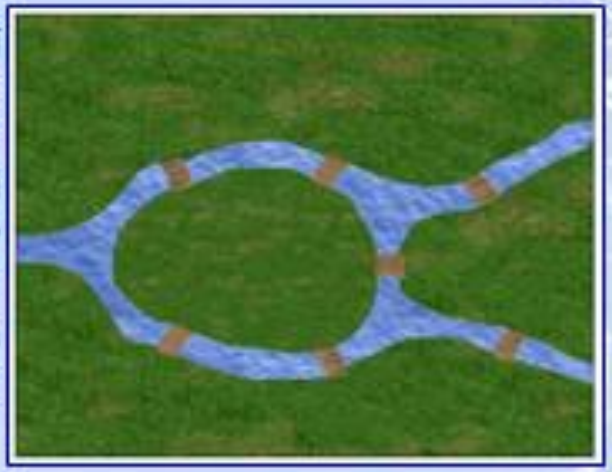


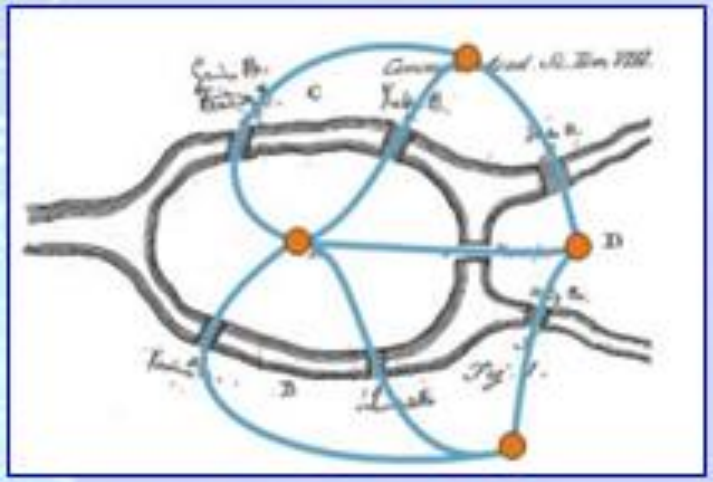
© М.В.Добролюбова

2

Теорія графів

Першою роботою теорії графів як математичної дисципліни вважають статтю Ейлера (1736 г.), в якій розглядалася задача про Кенінгсбергскі мости. Ейлер показав, що не можна обійти сім міських мостів і повернутися у вихідну точку, пройшовши по кожному мосту рівно один раз. Наступний імпульс теорія графів отримала через майже 100 років з розвитком досліджень з електричних мереж, кристалографії, органічної хімії та інших наук.





Розглянувши цю задачу, в 1736 році Ейлер довів, що це неможливо, причому він розглянув більш загальну задачу: які місцевості, розділені рукавами річок і з'єднані мостами, можливо обійти, побувавши на кожному мосту рівно один раз, а які неможливо.

@ М.В.Добролюбова 3

Теорія графів

Інші приклади графів

схема ліній метрополітену

генеалогічне древо

4

Теорія графів

Графи є зручним засобом опису зв'язків між об'єктами.

Але граф використовують аж ніяк не тільки як ілюстрацію. Наприклад, розглядаючи граф, який зображає мережу доріг між населеними пунктами, можна визначити маршрут проїзду від пункту *А* до пункту *Б*.

Якщо таких маршрутів виявиться декілька, хотілося б вибрати в певному сенсі оптимальний, наприклад, найкоротший або найбезпечніший.

Для розв'язку задачі вибору потрібно проводити певні обчислення над графами.

При розв'язку подібних задач зручно використовувати алгебраїчну техніку.

Станом на теперішній час теорія графів охоплює великий матеріал і активно розвивається. Існує велика кількість ще невирішених задач в теорії графів.

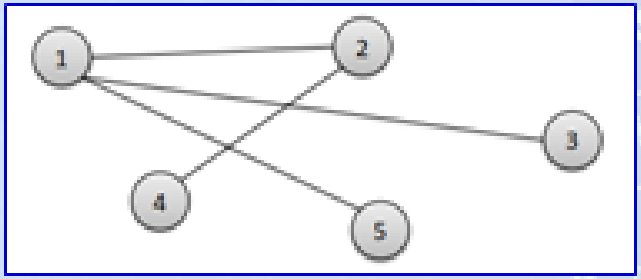
Графи є способом "візуалізації" зв'язків між певними об'єктами.

Зв'язки ці можуть бути "спрямованими", як, наприклад, в генеалогічному древі, або "неспрямованими" (мережа доріг з двостороннім рухом). Відповідно до цього в теорії графів виділяють два основних типи графів: ***орієнтовані*** (або спрямовані) і ***неорієнтовані***.

© М.В.Добролюбова 5

Неорієнтовані графи

Неорієнтований граф G задається двома множинами G = (V, E), де V - кінцева множина, елементи якої називають ***вершинами*** або вузлами; E – множина невпорядкованих пар на V, тобто підмножина множини двоелементних підмножин V, елементи якої називають ***ребрами***. Для кожного ребра {*u*, *v*} in E вважається, що *u* і *v* – різні вершини.

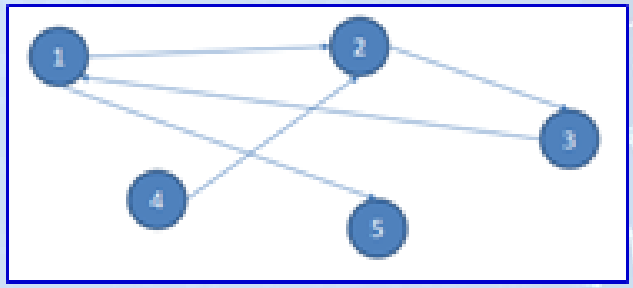


Якщо ребро *e* = {*u*, *v*}, то кажуть, що ребро *e* з'єднує вершини *u* і *v*, і позначають *u* → *v*.
Вершини *u* і *v*, з'єднані ребром (*u* → *v*), називають суміжними, а також кінцями ребра {*u*, *v*}. Якщо *u* → *v*, кажуть, що вершини *u* і *v* пов'язані відношенням безпосередньої досяжності.
Ребро *e* називають ***інцидентним*** вершині *v*, якщо вона є одним з його кінців.
Ступенем вершини *v* називають кількість всіх ребер G, інцидентних вершині *v*.
Шлях в графі – це набір вершин *v*[1]...*v*[*n*], при якому з вершини *v*[1] можна потрапити у вершину *v*[*n*].
Ланцюг – маршрут, в якому всі ребра попарно різні.
Цикл – замкнений маршрут, який є ланцюгом.
Неорієнтований граф, що не містить циклів, називають ***ациклічним*** графом.

© М.В.Добролюбова 6

Неорієнтовані графи

Орієнтований граф G задається двома множинами G = (V, E), де V - кінцева множина, елементи якої називають ***вершинами*** або вузлами; E – множина впорядкованих пар на V, тобто підмножина множини V * V, елементи якої називають ***дугами***.



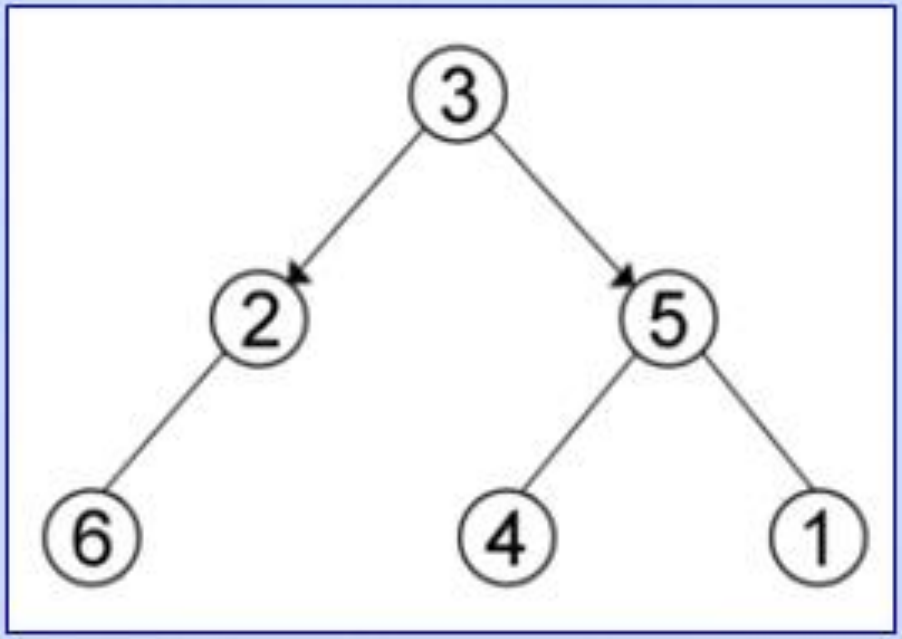
Якщо дуга *e* = (*u*, *v*), то говорять, що дуга *e* веде з вершини *u* у вершину *v*, і позначають це *u* → *v*. Вершини *u* і *v*, такі, що з вершини *u* у вершину *v* веде дуга (*u* → *v*), називають суміжними, причому *u* називають початком, а *v* – кінцем дуги (*u*, *v*). Дугу, початком і кінцем якої є одна і та сама вершина, називають ***петлею***. Якщо u → v, то кажуть, що вершини *u* і *v* пов'язані відношенням безпосередньої досяжності.

Шлях в орієнтованому графі G – це послідовність вершин (кінцева або нескінченна) *v*[0], *v*[1], ..., *v*[*n*], ..., така, що *v*[*i*] → *v*[*i* + 1] для будь-якого *i*, якщо *v*[*i* + 1] існує. Для кінцевого шляху *v*[0], *v*[1], ..., *v*[*n*] число *n* називають довжиною шляху.

Простий шлях – це шлях, всі вершини якого, крім, можливо, першої і останньої, попарно різні. Простий шлях ненульової довжини, початок і кінець якого збігаються, називають ***контуром***. Довільний шлях ненульової довжини, початок і кінець якого збігаються, називають ***замкненим шляхом***. Орієнтований граф, що не містить контурів, називають ***бесконтурним графом***.

Теорія графів

Розрізняють змішані графи, в яких є і спрямовані ребра, і неспрямовані.



© М.В.Добролюбова

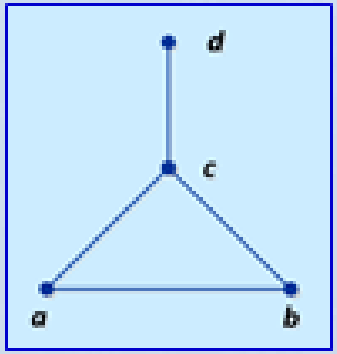
8

Теорія графів

Графічний спосіб представлення графів непридатний для ПК.

Способи представлення графів:

- **Матриця інцидентності.** Це матриця A з *n* рядками, що відповідають вершинам, і *m* стовпцями, що відповідають ребрам. Для орієнтованого графа стовпець, що відповідає дузі (*x*, *y*) містить 1 в рядку, який відповідає вершині *x* і 1, в рядку, який відповідає вершині *y*. У всіх інших 0. Петлю, тобто дугу (*x*, *x*) можна представляти іншим значенням в рядку *x*, наприклад, 2. Якщо граф неорієнтований, то стовпець, що відповідає ребру (*x*, *y*) містить 1, відповідні *x* і *y*, і нулі у всіх інших рядках.
- **Матриця суміжності.** Це матриця $n \times n$, де n – число вершин, $b_{ij} = 1$, якщо існує ребро, що йде з вершини *x* в вершину *y* і $b_{ij} = 0$ в іншому випадку.



	u	v	w	x
a	1	1	0	0
b	1	0	1	0
c	0	1	1	1
d	0	0	0	1

Матриця інцидентності

	a	b	c	d
a	0	1	1	0
b	1	0	1	0
c	1	1	0	1
d	0	0	1	0

Матриця суміжності

© М.В.Добролюбова

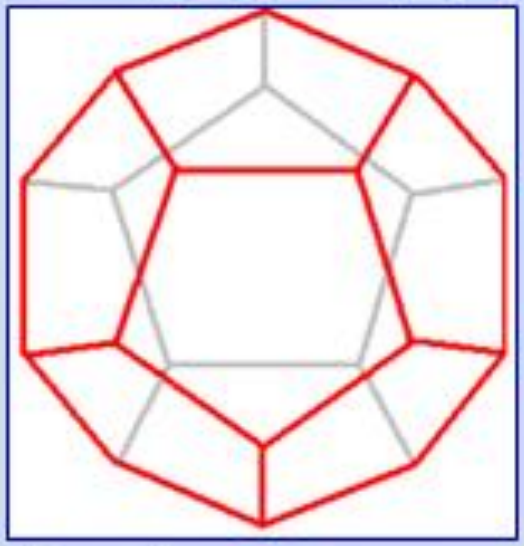
9

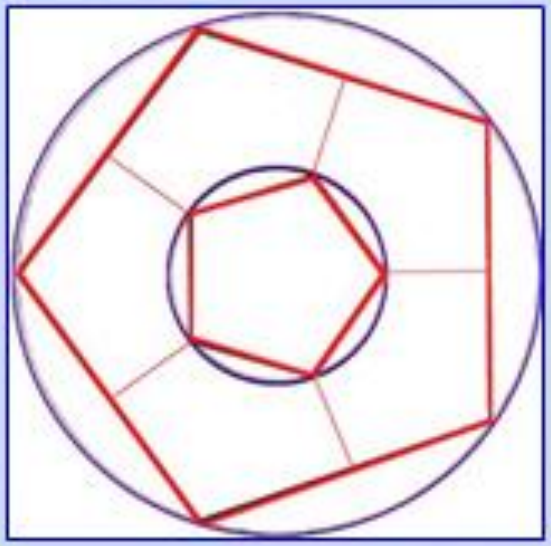
Обхід графа

Принципи обходу графу:

- ***Пошук в глибину*** (англ. Depth-First Search, DFS) – один з методів обходу графа. Стратегія пошуку в глибину, як і випливає з назви, полягає в тому, щоб йти «вглиб» графа, наскільки це можливо.

- ***Пошук в ширину*** (обхід по рівнях, BFS) – один з алгоритмів обходу графа. Метод лежить в основі деяких інших алгоритмів близької тематики. Пошук в ширину має на увазі порівневе дослідження графа.





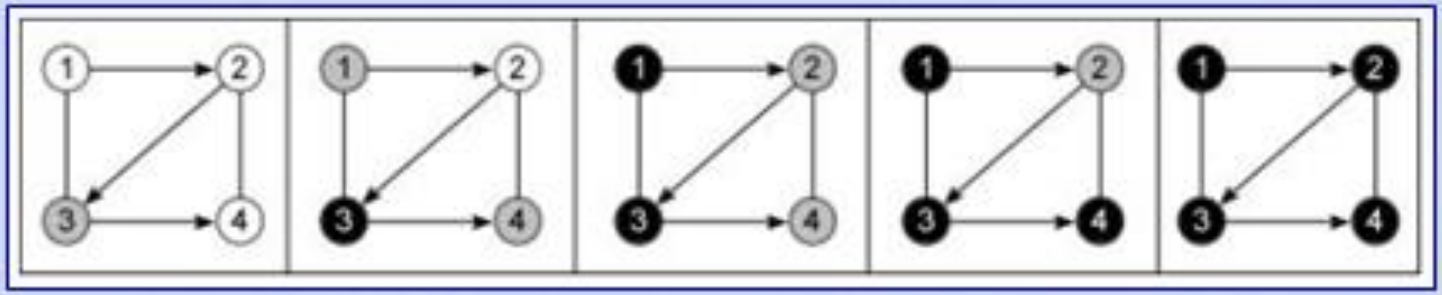
@ М.В.Добролюбова 10

Пошук в ширину

Пошук в ширину – це порівневе дослідження графа: спочатку відвідується корінь – довільно обраний вузол, потім – всі нащадки даного вузла, після цього відвідуються нащадки нащадків тощо. Вершини проглядаються в порядку зростання їх відстані від кореня.

Приклад:

Будемо вважати, що в процесі роботи алгоритму кожна з вершин графа може бути пофарбована в один з трьох кольорів: чорний, білий або сірий. Спочатку всі вершини білі. В процесі обходу кожна з вершин, в міру виявлення, забарвлюється спочатку в сірий, а потім в чорний колір. Певний момент обходу описує наступні умова: якщо вершина чорна, то все її нащадки пофарбовані в сірий або чорний колір.



@ М.В.Добролюбова 11

Пошук в ширину

Більш формальний опису алгоритму пошуку в ширину

Основні об'єктами – три структури даних, задіяних в програмі:

- матриця суміжності графа *GM*;
- черга *queue*;
- масив відвіданих вершин *visited*.

Дві перші структури мають цілочисельний тип даних, остання – логічний. Відвідані вершини заносяться в масив *visited*, що запобігти зацикленню, а черга *queue* буде зберігати задіяні вузли.

Процес обходу графа, розбитий на етапи:

1. Масив visited онуляється, тобто жодна вершина графа ще не переглянута.
2. В якості стартової вибирається деяка вершина *s* і поміщається в чергу (в масив *queue*).
3. Вершина *s* досліджується (позначається як відвідана) і всі суміжні з нею вершини поміщаються в кінець черги, а сама вона видаляється.
4. Якщо на даному етапі черга виявляється порожньою, то алгоритм припиняє свою роботу; інакше – відвідується вершина, що стоїть на початку черги, позначається як відвідана, і всї її нащадки заносяться в кінець черги.
5. Пункт 4 виконується, поки це можливо.

Пошук в ширину, починаючи зі стартової вершини, поступово відходить від неї все далі і далі, проходячи рівень за рівнем. Виходить, що після закінчення роботи алгоритму будуть знайдені всі найкоротші шляхи з початкової вершини до кожного доступного з неї вузла.

Граф представлений матрицею суміжності і, щодо ефективності, це не найкращий варіант, тому що час, витрачений на її обхід, оцінюється в $O(|V|^2)$, а скоротити його можна до $O(|V| + |E|)$, використовуючи список суміжності.

© М.В.Добролюбова 12

Пошук в глибину

Більш формальний опису алгоритму пошуку в ширину

Основні об'єктами – три структури даних, задіяних в програмі:

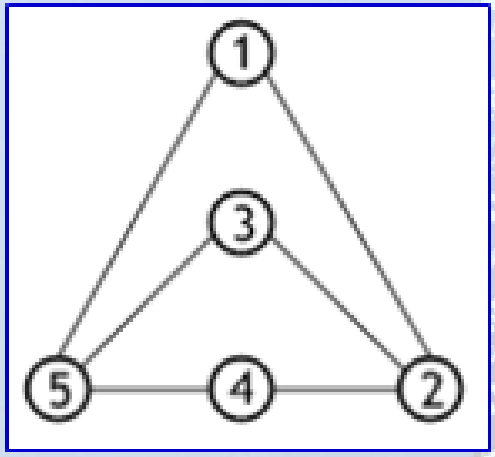
Якщо метод пошуку в ширину проводився симетрично (вершини графа проглядалися за рівнями), то даний метод передбачає просування вглиб до тих пір, поки це можливо.

Неможливість подальшого просування означає, що наступним кроком буде перехід на останній, який має кілька варіантів руху (один з яких досліджений повністю), раніше відвіданий вузол (вершина).

Відсутність останнього свідчить про одну з двох можливих ситуацій: або всі вершини графа вже переглянуті, або переглянуті всі ті, що доступні з вершини, взятої в якості початкової, але не всі (незв'язні і орієнтовані графи допускають останній варіант).

Приклад:

Наведений неорієнтовний зв'язний граф має в сумі 5 вершин. Спочатку необхідно вибрати початкову вершину. Яка б вершина в якості такої не була обрана, граф в будь-якому випадку досліджується повністю, оскільки це зв'язний граф без єдиного спрямованого ребра. Нехай обхід почнеться з вузла 1, тоді порядок послідовності переглянутих вузлів буде наступним: 1 2 3 5 4. Якщо виконання почати, наприклад, з вузла 3, то порядок обходу виявиться іншим: 3 2 1 5 4. Алгоритм пошуку в глибину базується на рекурсії, тобто функція обходу, в міру виконання, викликає сама себе, що робить код в цілому досить компактним.



© М.В.Добролюбова

13

Пошук в глибину

Псевдокод алгоритму

Заголовок функції *DFS(st)*
    Вивести (*st*)
    visited[*st*] ← переглянута;
    Для *r* = 1 до *n* виконувати
        Якщо (*graph*[*st*, *r*] ≠ 0) і (*visited*[*r*] не відвідана) то *DFS*(*r*)

Тут *DFS* (depth-firstsearch) – назва функції. Єдиний її параметр *st* – стартовий вузол, передається з головної частини програми як аргумент.

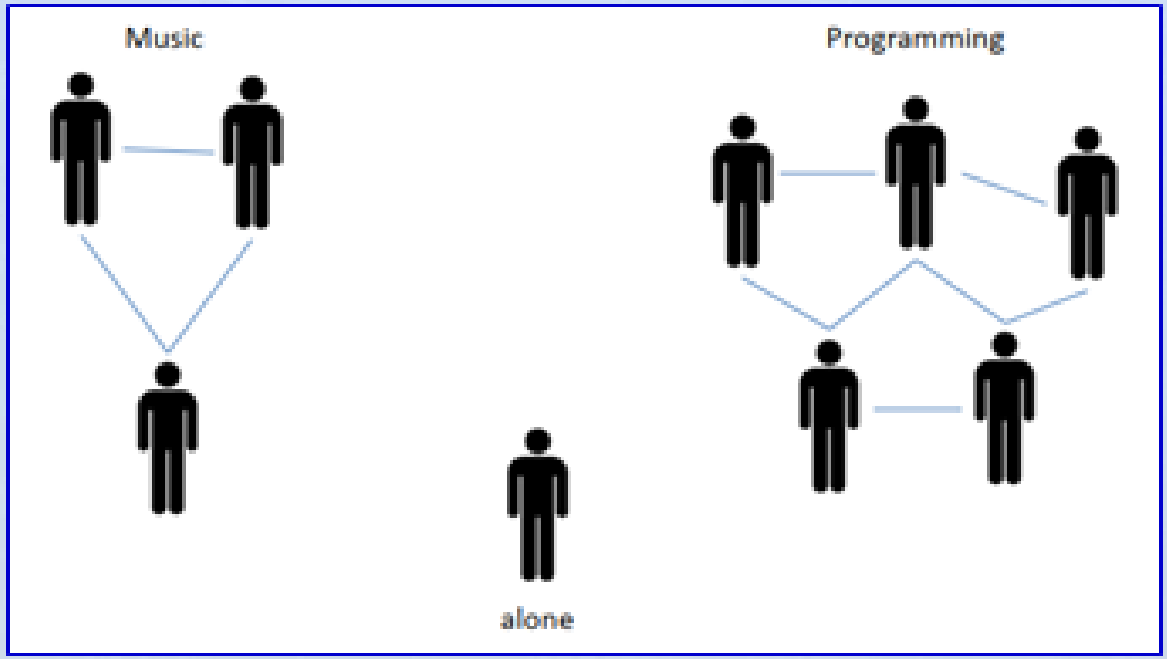
Кожному з елементів логічного масиву *visited* заздалегідь присвоюється значення *false*, тобто кожна з вершин спочатку буде значитися як невідвідана. Двовимірний масив *graph* – це матриця суміжності графа. Більшу частину уваги слід сконцентрувати на останньому рядку. Якщо елемент матриці суміжності на якомусь кроці дорівнює 1 (а не 0) і вершина з тим самим номером, що і стовпець матриці, який перевіряється, при цьому не відвідувалась, то функція рекурсивно повторюється. Інакше, функція повертається на попередню стадію рекурсії.


Обчислювальна техніка та програмування
РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ
Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ
Лекція 24
«Методи та алгоритми теорії графів»
© М.В.Добролюбова

Теорія графів

Компонента зв'язності графа – це деяка множина вершин графа така, що для будь-яких двох вершин з цієї множини існує шлях з однієї в іншу, і не існує шляху з вершини цієї множини в вершину не з цієї множини.

Найбільш поширені методи пошуку компонент зв'язності графа – за допомогою обходу в глибину або в ширину.



Теорія графів

Зв'язний граф – це граф, що містить рівно одну компоненту зв'язності. Це означає, що між будь-якою парою вершин цього графа існує як мінімум один шлях.

Приклад:
Прямим застосуванням теорії графів є теорія мереж і її додаток – теорія електронних мереж. Наприклад, всі комп'ютери, включені в мережу Інтернет, утворюють зв'язний граф, і хоча окрема пара комп'ютерів може бути не з'єднана безпосередньо (в формулюванні для графів – не з'єднана ребром), від кожного комп'ютера можна передати інформацію до будь-якого іншого (є шлях з будь-якої вершини графа в будь-яку іншу).

В орієнтованих графах розрізняють кілька понять зв'язності.

Сильно-зв'язний орієнтований граф – це граф, в якому існує (орієнтований) шлях з будь-якої вершини в будь-яку іншу, або, що еквівалентно, граф містить рівно одну сильно зв'язну компоненту.

Слабо-зв'язний орієнтований граф – це граф, з якого отриманий неорієнтований граф шляхом заміни орієнтованих ребер на неорієнтовані, і він є зв'язним.

© М.В.Добролюбова 3

Теорія графів

Ейлерів шлях (Ейлерів ланцюг) в графі – це шлях (ланцюг), що проходить по всім дугам (ребрам) графа і притому тільки по одному разу.

Ейлерів цикл – це цикл графа, що проходить через кожне ребро (дугу) графа рівно по одному разу.

Ейлерів граф – це граф, що містить Ейлерів цикл.

Напівейлерів граф – це граф, що містить Ейлерів шлях (ланцюг).

В неорієнтованому графі:

Згідно з теоремою, доведеною Ейлером, в графі без одиночних вершин Ейлерів цикл існує тоді і тільки тоді, коли граф зв'язний і в ньому відсутні вершини непарного ступеня.

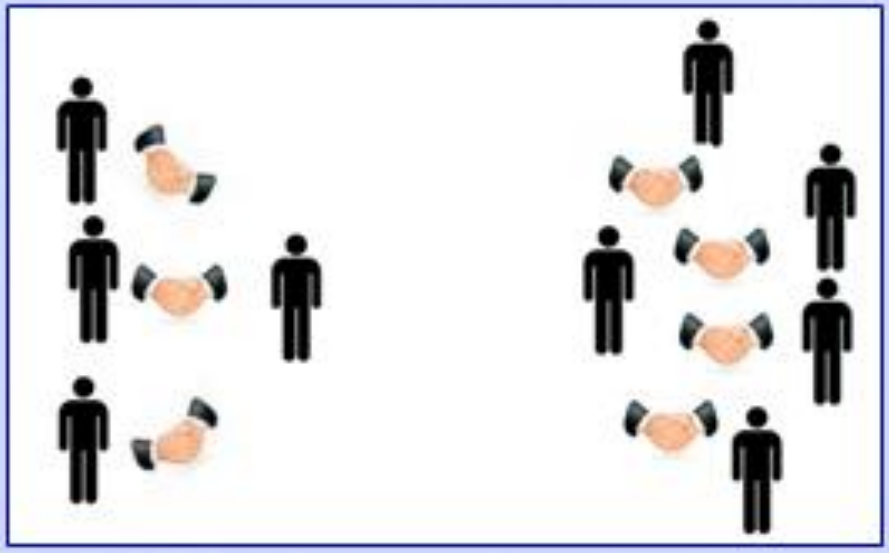
Ейлерів ланцюг в графі існує тоді і тільки тоді, коли граф зв'язний і містить не більше двох вершин непарного ступеня.

З огляду на ***лему про рукостискання***, число вершин з непарним ступенем повинно бути парним. А значить, Ейлерів шлях існує тільки тоді, коли це число дорівнює нулю або двом. Причому, коли воно дорівнює нулю, Ейлерів шлях вироджується в Ейлерів цикл.

Теорія графів

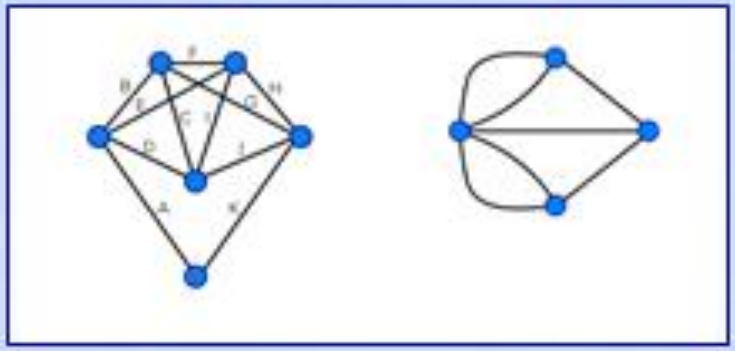
Лема про рукостискання – це положення теорії графів, згідно з яким будь-який кінцевий неорієнтований граф має парне число вершин непарних ступенів.

Лемма бере назву від популярної аналогії: в групі людей, деякі з яких потискають один одному руки, парне число людей привітало таким чином непарне число колег.

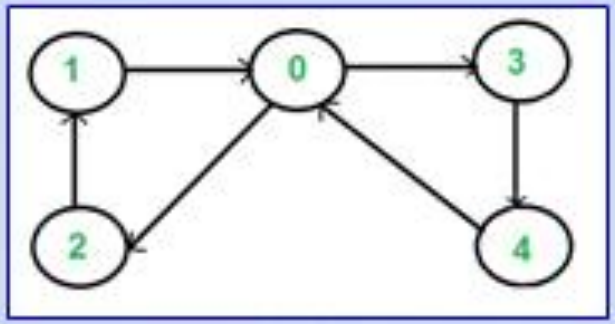


@ М.В.Добролюбова 5

Теорія графів



Орієнтований граф G = (V, A) містить Ейлерів цикл тоді і тільки тоді, коли він сильно пов'язаний і для кожної вершини графа її напівступень входження дорівнює її напівступеню результату, тобто в вершину входить стільки ж ребер, скільки з неї і виходить.



@ М.В.Добролюбова

6

Теорія графів

Шлях в графі – послідовність вершин, що має для кожної вершини ребро, що з'єднує її з наступною вершиною в послідовності.

Є кілька варіантів знаходження шляху між двома вершинами. Можна використовувати пошуки в глибину, або в ширину.

Алгоритм Лі (хвильовий алгоритм) – це алгоритм пошуку шляху, алгоритм пошуку найкоротшого шляху на планарному графі. Належить до алгоритмів, заснованих на методах пошуку в ширину.

В основному використовується при комп'ютерному трасуванні (розводці) друкованих плат, з'єднуючих провідників на поверхні мікросхем. Інше застосування хвильового алгоритму – пошук найкоротшої відстані на карті в комп'ютерних стратегічних іграх.

Етапи роботи алгоритму:

- ініціалізація;
- поширення хвилі;
- відновлення шляху.

© М.В.Добролюбова 7

Теорія графів

Приклад (прикладна ***задача про пожежу в лісі***):
Є деякий ліс і в один момент одне дерево підпалюють. За 1 одиницю часу полум'я поширюється на сусідні дерева (зверху, знизу, з боків і по діагоналях). Потрібно сказати, за скільки одиниць часу вигорить весь ліс (всі можливі дерева). Або ж потрібно показати стан лісу в будь-який момент часу.

Спочатку необхідно привести задачу до загального вигляду.

Вхідні дані:
Нехай є карта лісу, де 1 означає, що в цій точці є дерево, а 0 – дерева немає. Отримуємо деяку точку (*x*, *y*), в якій починається пожежа.

Вихідні дані:
Вивести стан лісу в заданий момент часу, де -1 означає, що дерево вже згоріло або ще горить.

© М.В.Добролюбова 8


Теорія графів
0001100
0101001
0010110
1010001
@ М.В.Добролюбова
9

Теорія графів

Задача про найкоротший шлях (англ. shortest path problem) – це задача пошуку найкоротшого шляху (ланцюга) між двома точками (вершинами) на графі, в якій мінімізується сума ваг ребер, що складають шлях.

Задача про найкоротший шлях є однією з найважливіших класичних задач теорії графів. Відомо безліч алгоритмів для її рішення.

Постановки задачі про найкоротший шлях:

- ***Задача про найкоротший шлях*** в заданий пункт призначення. Потрібно знайти найкоротший шлях в задану вершину призначення *t*, який починається в кожної з вершин графа (крім *t*). Помінявши напрямок кожного ребра, що належить графу, цю задачу можна звести до задачі про єдину вихідну вершину (в якій здійснюється пошук найкоротшого шляху з заданої вершини в усі інші).
- ***Задача про найкоротший шлях*** між заданою парою вершин. Потрібно знайти найкоротший шлях із заданої вершини *u* в задану вершину *v*.
- ***Задача про найкоротший шлях*** між усіма парами вершин. Потрібно знайти найкоротший шлях з кожної вершини *u* в кожну вершину *v*. Цю задачу теж можна розв'язати за допомогою алгоритму, призначеного для розв'язку задачі про одну вихідну вершину, однак зазвичай вона розв'язується швидше.

В різних постановках задачі роль довжини ребра можуть грати не тільки самі довжини, але і час, вартість, витрати, обсяг витрачених ресурсів (матеріальних, фінансових, паливно-енергетичних тощо) або інші характеристики, пов'язані з проходженням кожного ребра. Таким чином, завдання знаходить практичне застосування у великій кількості областей (інформатика, економіка, географія тощо).

Теорія графів

Алгоритм Дейкстри (англ. Dijkstra's algorithm) – алгоритм на графах, винайдений нідерландським ученим Е. Дейкстрою в 1959 році. Знаходить найкоротшу відстань від однієї з вершин графа до всіх інших. Алгоритм працює тільки для графів без ребер від'ємної ваги. Алгоритм широко застосовується в програмуванні і технологіях, наприклад, його використовують протоколи маршрутизації OSPF і IS-IS.

Задача:

Є деяка кількість авіарейсів між містами світу, для кожного відома вартість.
Вартість перельоту з *A* в *B* може не дорівнювати вартості перельоту з *B* в *A*. Знайти маршрут мінімальної вартості (можливо, з пересадками) від Києва до Нью-Йорка.

Формальне визначення

Наданий зважений орієнтований граф G(V, E) без дуг від'ємної ваги. Знайти найкоротші шляхи від деякої вершини графа G до всіх інших вершин цього графа.

@ М.В.Добролюбова 11

Теорія графів

Алгоритм роботи

Кожній вершині з V ставиться у відповідність мітка – мінімальна відома відстань від цієї вершини до *a*. Алгоритм працює покроково – на кожному кроці він «відвідує» одну вершину і намагається зменшувати мітки. Робота алгоритму завершується, коли всі вершини відвідані.

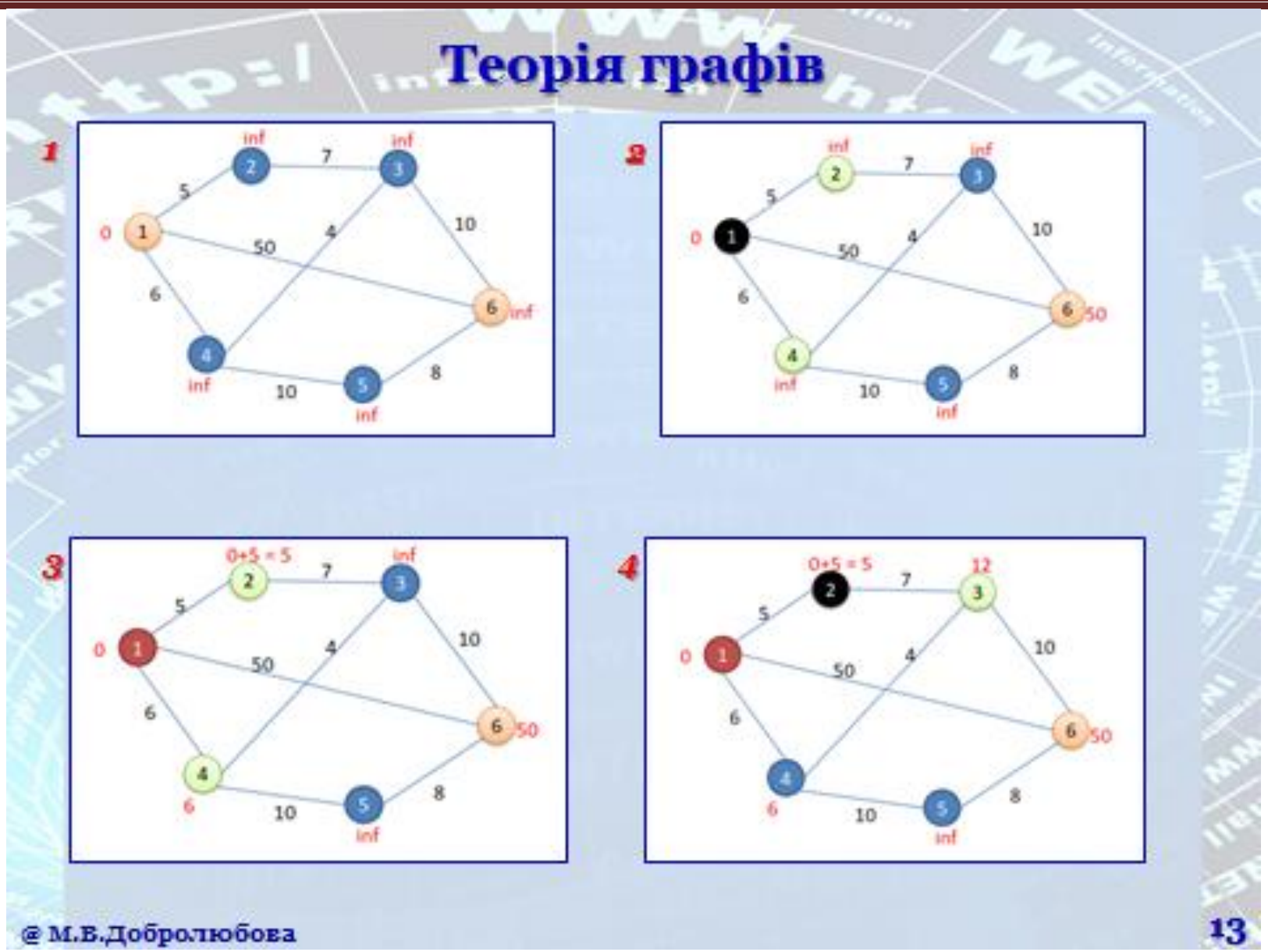
Ініціалізація

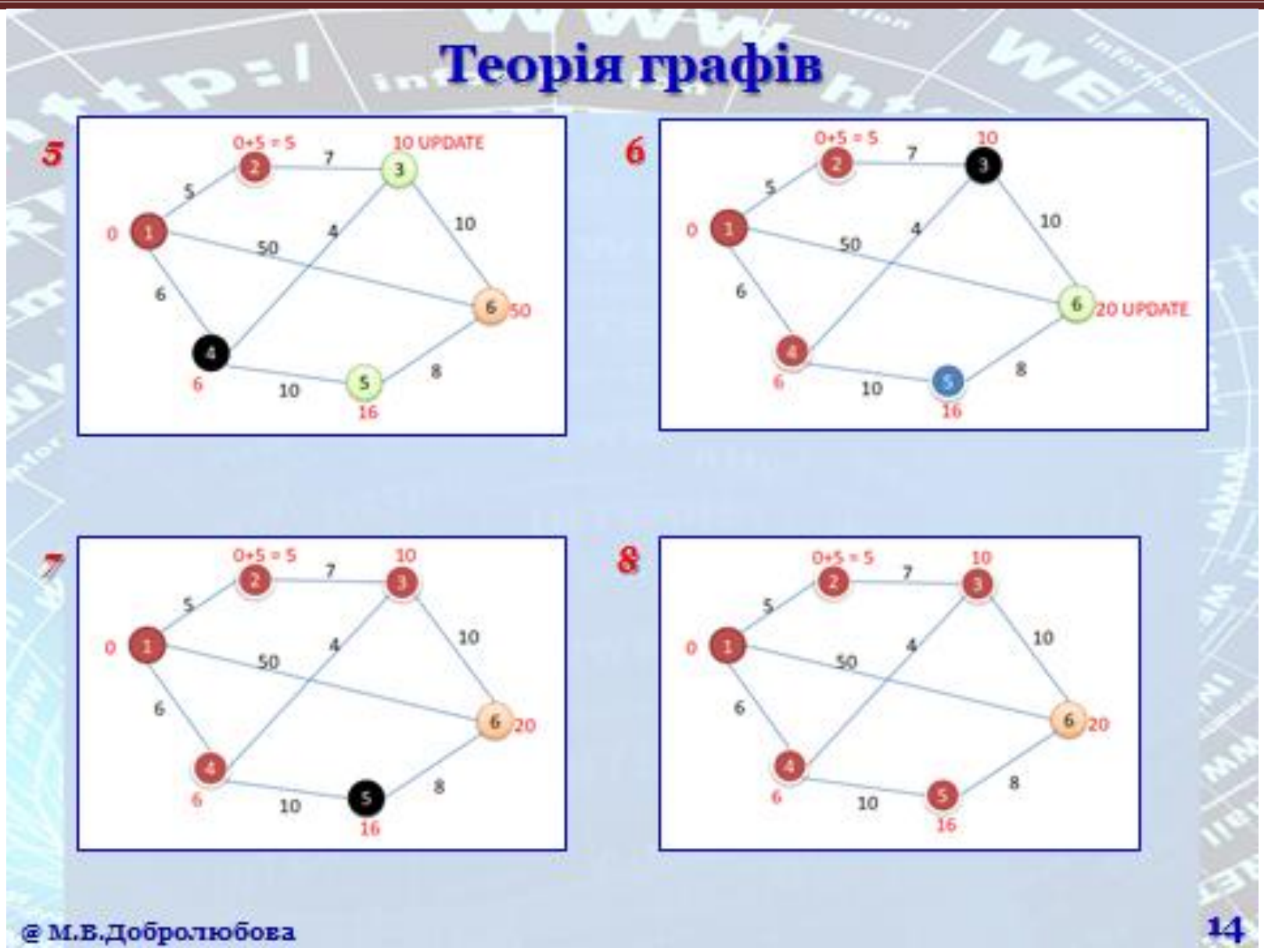
Мітка самої вершини *a* визначається рівною 0, мітки інших вершин – нескінченності. Це відображає те, що відстані від *a* до інших вершин поки невідомі. Всі вершини графа позначаються як невідвідані.

Крок алгоритму

Якщо всі вершини відвідані, алгоритм завершується. В іншому випадку, з ще невідвіданих вершин вибирається вершина *u*, що має мінімальну позначку. Розглядаються різні маршрути, в яких *u* є передостаннім пунктом. Вершини, в які ведуть ребра з *u*, називають сусідами цієї вершини. Для кожного сусіда вершини *u*, крім зазначених як відвідані, розглядається нова довжина шляху, що дорівнює сумі значень поточної мітки *u* і довжини ребра, що з'єднує *u* з цим сусідом. Якщо отримане значення довжини менше за значення мітки сусіда, значення мітки замінюється отриманим значенням довжини. Після розгляду всіх сусідів, вершина *u* позначається як відвідана і крок алгоритму повторюється.

© М.В.Добролюбова 12


Теорія графів
1
2
3
4
@ М.В.Добролюбова
13


Теорія графів
5
0+5 = 5
10 UPDATE
7
5
4
10
0
50
6
6
50
6
10
8
16
6
0+5 = 5
10
7
5
4
10
0
50
6
6
20 UPDATE
6
10
8
16
7
0+5 = 5
10
7
5
4
10
0
50
6
6
20
6
10
8
16
8
0+5 = 5
10
7
5
4
10
0
50
6
6
20
6
10
8
16
@ М.В.Добролюбова
14


Обчислювальна техніка та програмування
РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ
Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ
Лекція 25
«Методи та алгоритми теорії графів (продовження)»
© М.В.Добролюбова

Лінійне сортування

Найкоротший шлях у графі від однієї вершини до всіх інших ми шукали за допомогою алгоритму Дейкстри.

Складність алгоритму Дейкстри залежить від способу знаходження вершини *v*, а також способу збереження множини невідвіданих вершин і способу оновлення міток. Позначимо через *n* кількість вершин, а через *m* – кількість ребер в графі G.

В найпростішому випадку, коли для пошуку вершини з мінімальним *d*[*v*] проглядається вся множина вершин, а для збереження величин *d* використовується масив, час роботи алгоритму – $O(n^2)$.

Основний цикл виконується порядку *n* разів, в кожному з них на знаходження мінімуму витрачається близько *n* операцій.

На цикли по сусідах кожної відвідуваної вершини витрачається кількість операцій, пропорційна кількості ребер *m* (оскільки кожне ребро зустрічається в цих циклах рівно двічі і вимагає константне число операцій).

Таким чином, загальний час роботи алгоритму $O(n^2 + m)$, але, оскільки $m < n(n-1)$, він становить $O(n^2)$.

Теорія графів

Алгоритм Флойда-Уоршелла – алгоритм для знаходження найкоротших відстаней між усіма вершинами зваженого графа без циклів з від'ємними вагами з використанням методу динамічного програмування.

Різниця в тому, що даний алгоритм буде знаходити найкоротший шлях між *усіма* вершинами графа (зваженого і без циклів з від'ємними вагами).

Цей алгоритм був одночасно опублікований в статтях Роберта Флойда (Robert Floyd) і Стівена Уоршелла (Stephen Warshall) в 1962 р., хоча в 1959 р. Бернард Рой (Bernard Roy) опублікував практично такий самий алгоритм, але це пройшло повз увагу.

В основі алгоритму лежать дві властивості найкоротшого шляху графа.

Перша. Є найкоротший шлях $p_{1k} = (v_1, v_2, ..., v_k)$ від вершини v_1 до вершини v_k, а також його підшлях $p'(v_i, v_{i+1}, ..., v_j)$, при цьому діє $1 <= i <= j <= k$. Якщо p – найкоротший шлях від v_1 до v_k, то p' також є найкоротшим шляхом від вершини v_i до v_j.

Друга властивість є основою алгоритму. Розглядається граф G з пронумерованими від 1 до n вершинами $\{v_1, v_2, ..., v_n\}$ і шлях p_{ij} від v_i до v_j, що проходить через певну множину дозволених вершин, обмежене індексом k. Тобто якщо k = 0, то розглядають прямі з'єднання вершин одна з одною, оскільки множина дозволених проміжних вершин рівна нулю. Якщо k = 1 – розглядають шляхи, що проходять через вершину v_1, при k = 2 – через вершини $\{v_1, v_2\}$, при k = 3 – $\{v_1, v_2, v_3\}$ і так далі.

Теорія графів

Для реалізації алгоритму Флойда-Уоршелла формується матриця суміжності D[][] графа G = (V, E), в якому кожна вершина пронумерована від 1 до |V|. Ця матриця має розмір |V| * |V| і кожному її елементу D[*i*][*j*] присвоєно вагу ребра, що йде з вершини *i* у вершину *j*.

По мірі виконання алгоритму, дана матриця буде записуватись наступним чином: в кожну з її комірок внесе значення, що визначає оптимальну довжину шляху з вершини *i* в вершину *j* (відмова від виділення спеціального масиву для цієї мети збереже пам'ять і час).

Перед складанням основної частини алгоритму, необхідно розібратися з вмістом матриці найкоротших шляхів. Оскільки кожен її елемент D[*i*][*j*] повинен містити найменший з наявних маршрутів, то для одиничної вершини він дорівнює нулю, навіть якщо вона має петлю (від'ємні цикли не розглядаються), отже всі елементи головної діагоналі (D[*i*][*j*]) потрібно онулити. А щоб нульові недіагональні елементи (матриця суміжності могла мати нулі в тих місцях, де немає безпосереднього ребра між вершинами *i* та *j*) змінили по можливості своє значення, вони визначаються рівними нескінченності, яка в програмі може бути, наприклад, максимально можливою довжиною шляху в графі, або просто великим числом.

Найкоротший шлях з вершини *i* в вершину *j* може проходити, як тільки через них самих, так і через множину інших вершин *k* ∈ (1, ..., |V|). Оптимальним з *i* в *j* буде шлях, який або не проходить через *k*, або проходить. Зробити висновок про наявність другого випадку – означає встановити, що такий шлях йде з *i* до *k*, а потім з *k* до *j*, тому можна замінити значення найкоротшого шляху D[*i*][*j*] сумою D[*i*][*k*] + D[*k*][*j*].

Псевдокод:

Для *k* від 1 до |V| виконувати
Для *i* від 1 до |V| виконувати
Для *j* від 1 до |V| виконувати
Якщо D[*i*][*k*] + D[*k*][*j*] < D[*i*][*j*] то D[*i*][*j*] ← D[*i*][*k*] + D[*k*][*j*]

© М.В.Добролюбова 4

Теорія графів

Компонента зв'язності графа – деяка множина вершин графа така, що для будь-яких двох вершин з цієї множини існує шлях з однієї в іншу, і не існує шляху з вершини цієї множини в вершину не з цієї множини.

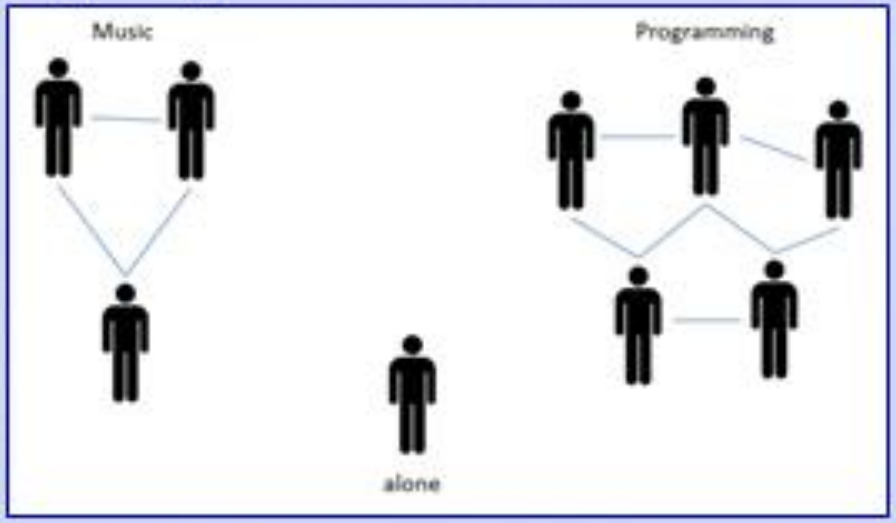
Найбільш поширені методи пошуку компонент зв'язності – за допомогою обходу в глибину або в ширину.

Задача:

Знайти всі компоненти зв'язності.

Формулювання прикладної задачі:

Граф – набір деяких людей, ребра між людьми означають, що ці люди спілкуються. Знайти кількість різних груп людей.



@ М.В.Добролюбова

5

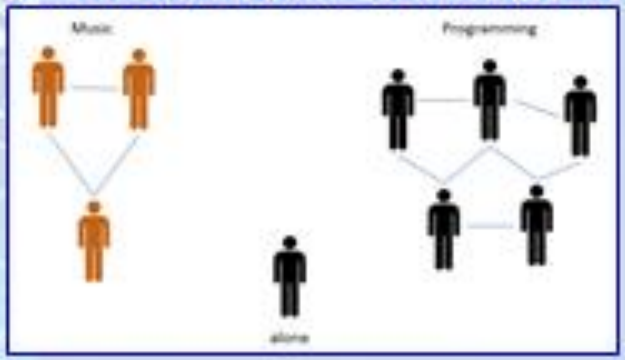
Теорія графів

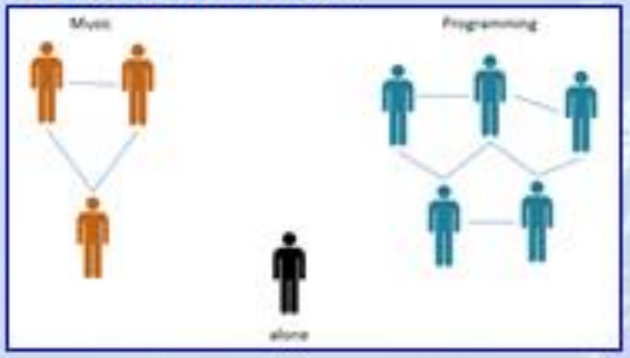
Згадаймо алгоритм Лі.

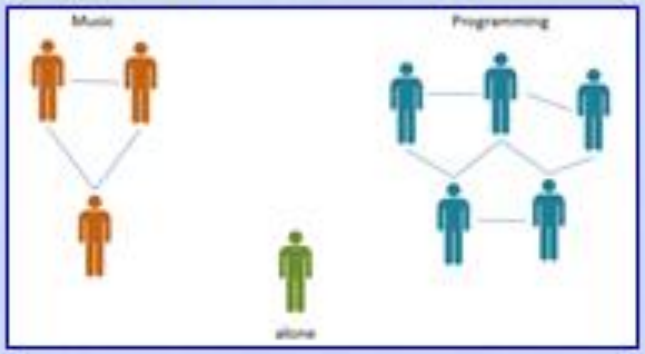
Тут використовується схожий підхід.

Необхідно «розфарбовувати» наші вершини. Нехай є набір квітів 1..*n*, де *n* – кількість вершин.

Запустивши пошук в глибину з деякої вершини, отримаємо набір вершин, які досяжні з цієї вершини. Тобто вони формують одну множину. Розфарбуємо ці вершини в колір 1, після чого повторимо даний крок, поки всі вершини не будуть відвідані.







© М.В.Добролюбова

6

Теорія графів

Топологічне сортування графа – це упорядкування вершин безконтурного орієнтованого графа згідно часткового порядку, заданого ребрами орграфа на множині його вершин.

Маршрут в орграфі – це послідовність вершин і дуг, які чергуються, виду $v_0\{v_0, v_1\}$, $v_1\{v_1, v_2\}$, v_2 ... v_n (вершини можуть повторюватися).

Довжина маршруту – кількість дуг в ньому.

Шлях – це маршрут в орграфі без повторюваних дуг.

Простий шлях – це маршрут в орграфі без повторюваних вершин. Якщо існує шлях з однієї вершини в іншу, то друга вершина досяжна з першої.

Контур – це замкнений шлях.

Приклади застосування топологічного сортування:

- при розпаралелювання алгоритмів, коли по деякому опису алгоритму потрібно скласти граф залежностей його операцій і, відсортувавши його топологічно, визначити, які з операцій є незалежними і можуть виконуватися паралельно (одночасно);
- при створенні карти сайту, де має місце деревоподібна система розділів.

За допомогою топологічної сортування будується коректна послідовність виконання дій, всяка з яких може залежати від іншої: послідовність вивчення дисциплін студентами, установки програм за допомогою пакетного менеджера.

Теорія графів

Частково впорядкована множина – математичне поняття, яке формалізує інтуїтивні ідеї впорядкування, розташування елементів в певній послідовності. Неформально множина частково впорядкована, якщо вказано, які елементи слідують за якими (які елементи більше яких). В загальному випадку може виявитися так, що деякі пари елементів не пов'язані відношенням «слідує за».

Задача топологічного сортування графа: вказати такий лінійний порядок на вершинах графа, щоб будь-яке ребро вело від вершини з меншим номером до вершини з більшим номером. Очевидно, що якщо в графі є цикли, то такого порядку не існує.

Задача:

Впорядкувати вершини за деяким правилом, згідно з пріоритетом.

Формулювання прикладної задачі:

У студента не виконані домашні завдання. Лабораторні, курсові, РГР, реферати тощо... Проблема в тому, що для виконання деяких завдань потрібно перед цим виконати інші. Наприклад, для виконання завдання 3 потрібно перед цим зробити завдання 5.

Отримаємо на вхід числа *n*, *m*, де n – кількість завдань, m – кількість зв'язків між завданнями. Потім йде *m* рядків по два числа *i*, *j*, які означають, що завдання під номером *i* має бути виконано перед завданням під номером *j*.

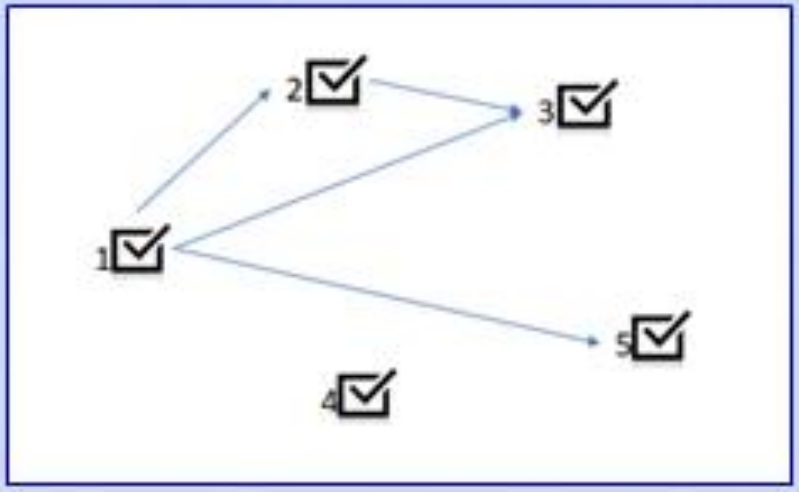
Потрібно вивести всі завдання в порядку їх виконання.

8

Теорія графів

In
5 4
1 2
2 3
1 3
1 5

Out
1 4 2 5 3



@ М.В.Добролюбова

9

Теорія графів

Алгоритм Тар'яна

1. Створення порожнього стеку.

2. Позначення всіх вершин білими кольором.

3. Проведення пошуку в глибину з кожної вершини:

   - При вході вершина розфарбовується в сірий колір, а при виході – в чорний і одночасно заноситься до стеку.

   - Якщо раптом здійснено вхід в сіру вершину – знайдений цикл, топологічне сортування неможливо.

Даний алгоритм виконується за O(*n*) часу і пам'яті.

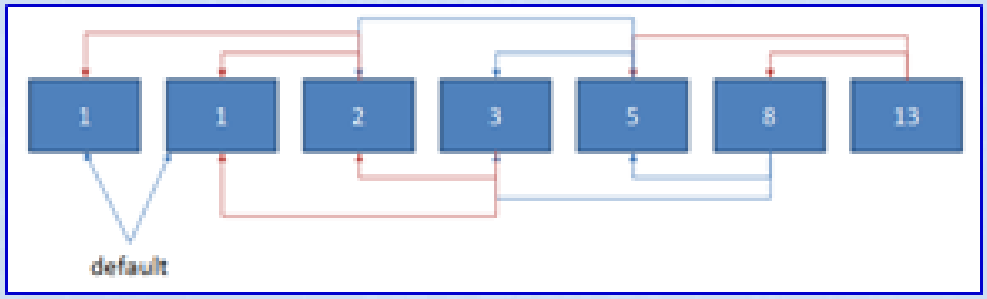
@ М.В.Добролюбова 10


Обчислювальна техніка та програмування
РОЗДІЛ 2
ОСНОВИ АЛГОРИТМІЗАЦІЇ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ
Тема 2.1 СТРУКТУРИ ТА АЛГОРИТМИ ДАНИХ
Лекція 26
«Динамічне програмування»
© М.В.Добролюбова

Динамічне програмування

Динамічне програмування (далі динаміка) – це спосіб вирішення складних задач шляхом розбиття їх на більш прості підзадачі.

Приклад: числа Фібоначчі.



Цей спосіб можна застосовувати до задач з оптимальною структурою, що виглядає як набір підзадач, що перекриваються, складність якої менше вихідної.

Оптимальна підструктура в динамічному програмуванні – це оптимальне рішення підзадач меншого розміру, яке може бути використане для розв'язку вихідної задачі.

Підзадачі, що перекриваються в динамічному програмуванні – це підзадачі, які використовуються для розв'язку певної кількості задач (не однієї) більшого розміру (тобто декілька разів виконується одне й те саме).

© М.В.Добролюбова 2

Динамічне програмування

Ідея:
розбити складну задачу на менші підзадачі, розв'язати їх і сконструювати відповідь з цих підзадач для складної задачі. Часто ці підзадачі дублюються.

Підхід динаміки:
розв'язати задачі, які дублюються, тільки 1 раз.

Динамічне програмування – це коли є задача, яку незрозуміло як вирішувати, і вона розбивається на менші задачі, які теж незрозуміло як вирішувати. © А. Кумок.

Методи динаміки:

- ***Зверху*** – просто запам'ятовування результатів тих підзадач, які можуть зустрітися надалі (рахуємо відразу від 1 до N).
- ***Знизу*** – включає в себе переформулювання складної задачі у вигляді рекурсивної послідовності простіших задач (рахуємо, «спустившись вниз», і, піднімаючись, збираємо результати в інші задачі).

@ М.В.Добролюбова 3

Динамічне програмування

Фактори успішного рішення задачі динаміки:

1) Стан динаміки: параметр (и), однозначно задають підзадачу.

2) Значення початкових станів.

3) Переходи між станами: формула перерахунку.

4) Порядок перерахунку.

5) Положення відповіді на завдання: іноді це сума або, наприклад, максимум зі значень декількох станів.

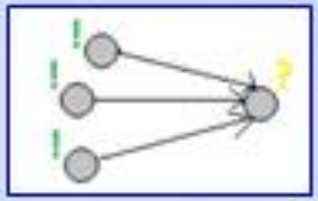
Порядок перерахунку:

- прямий;
- обернений;
- лінива динаміка.

Динамічне програмування

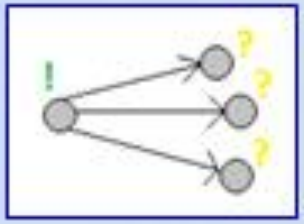
Прямий порядок

Стани послідовно перераховуються, виходячи з уже порахованих.



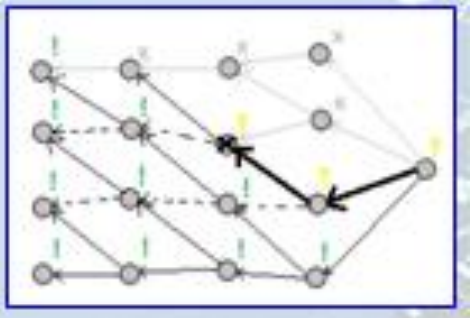
Обернений порядок

Оновлюються всі стани, що залежать від поточного стану.



Лінива динаміка

Рекурсивна мемоізована функція перерахунку динаміки. Це щось на зразок пошуку в глибину по ациклічності графу станів, де ребра – це залежності між ними.



@ М.В.Добролюбова 5

Динамічне програмування

Мемоізація (запам'ятовування, від англ. memoization) – це збереження результатів виконання функцій для запобігання повторних обчислень. Це один з способів оптимізації, який застосовується для збільшення швидкості виконання комп'ютерних програм. Перед викликом функції перевіряється, чи викликалася вона раніше: якщо не викликалася, функція викликається і результат її виконання зберігається; якщо викликалася, використовується збережений результат.

Адитивність – це одна з основних властивостей задач, що розв'язуються за допомогою динамічного програмування. Неаддитивні задачі розв'язуються іншими методами.

Адитивність (лат. additivus – те, що додається) – це властивість величин, яка полягає в тому, що значення величини, яке відповідає цілому об'єкту, дорівнює сумі значень величин, які відповідають його частинам, в деякому класі можливого розбиття об'єкта на частини.

Приклад: адитивність об'єму означає, що об'єм цілого тіла дорівнює сумі об'ємів складових його частин.

Алгоритм Флойда-Уоршелла теорії графів – для пошуку найкоротшого шляху між усіма парами вершин – використовує метод динамічного програмування. Алгоритм обчислює мінімум для кожної вершини шляхом вибору мінімуму серед прямого ребра і ребра, що проходить через якусь додаткову вершину. На кожному етапі переглядається мінімум і записується у відповідну комірку.

Динамічне програмування

Задача знаходження найбільшої спільної підпослідовності (англ. Longest Common Subsequence, LCS) – задача пошуку послідовності, яка є підпослідовністю декількох послідовностей (зазвичай двох).

Часто задача визначається як пошук всіх найбільших підпослідовностей.
Це класична задача інформатики, яка має додатки, зокрема, в задачі порівняння текстових файлів (утиліта diff), а також в біоінформатиці.
Підпослідовність можна отримати з деякої кінцевої послідовності, якщо видалити з останньої деяку множину її елементів (можливо порожню).

Приклад: BCDB є підпослідовністю послідовності ABCDBAB.
Будемо говорити, що послідовність Z є спільною підпослідовністю послідовностей X і Y, якщо Z є підпослідовністю як X, так і Y. Потрібно для двох послідовностей X і Y знайти загальну підпослідовність найбільшої довжини. При цьому, НЗП може бути кілька.

Методи розв'язку задачі:

- повний перебір;
- динамічне програмування.

Існують різні підходи щодо розв'язку даної задачі при повному переборі – можна перебирати варіанти підпослідовності, варіанти викреслювання з даних послідовностей тощо. Однак, в будь-якому випадку, час роботи програми буде експонентою від довжини рядка.

Динамічне програмування

Метод розв'язку за допомогою динаміки

Спочатку знаходять довжину найбільшої підпослідовності.
Припустимо, ведеться пошук рішення для випадку (n_1, n_2), де n_1, n_2 – довжини першого та другого рядків. Нехай вже існують рішення для всіх підзадач (m_1, m_2), менших за задану. Тоді задача (n_1, n_2) зводиться до менших підзадач наступним чином:

$f(n_1, n_2) =$

1. Якщо $n_1 = 0$ & $n_2 = 0 \rightarrow 0$
2. Якщо $s[n_1] = s[n_2] \rightarrow f(n_1-1, n_2-1) + 1$
3. Якщо $s[n_1] != s[n_2] \rightarrow \max(f(n_1-1, n_2), f(n_1, n_2-1))$

Тепер повернемося до задачі побудови підпослідовності. Для цього в існуючий алгоритм додається запам'ятовування для кожної задачі тієї підзадачі, через яку вона розв'язується. Наступною дією, починаючи з останнього елемента, піднімаємося до початку за напрямками, заданим першим алгоритмом, і записуємо символи в кожній позиції. Це і буде відповіддю в даній задачі.

Час роботи алгоритму буде $O(n_1 * n_2)$.

8

Динамічне програмування

Задача «Зайчик»

Уявіть зоопарк, в якому з'явився зайчик. Щоб йому не було сумно, в клітку поставили драбину з *N* сходинок.

Зайчик може стрибати вгору, перестрибуючи через певну кількість сходинок за 1 раз. Таким чином, зайчик одним стрибком може подолати не більше *K* сходинок.

Задача – підрахувати кількість різних способів дістатися до вершини при заданих *N* і *K*.

Наприклад, якщо *K* = 3 і *N* = 4, то існують наступні маршрути:
1 + 1 + 1 + 1, 1 + 1 + 2, 1 + 2 + 1, 2 + 1 + 1, 2 + 2, 1 + 3, 3 + 1.
Тобто при даних значеннях у зайця всього 7 різних маршрутів дістатися до вершини сходів.

Приклади:

N = 3, *K* = 1 → Answer = 1
N = 7, *K* = 2 → Answer = 21

@ М.В.Добролюбова 9


ДЯКУЮ ЗА УВАГУ!
Лектор
к.т.н., доц., доцент кафедри ІВТ
КПІ ім. Ігоря Сікорського
М. В. Добролюбова