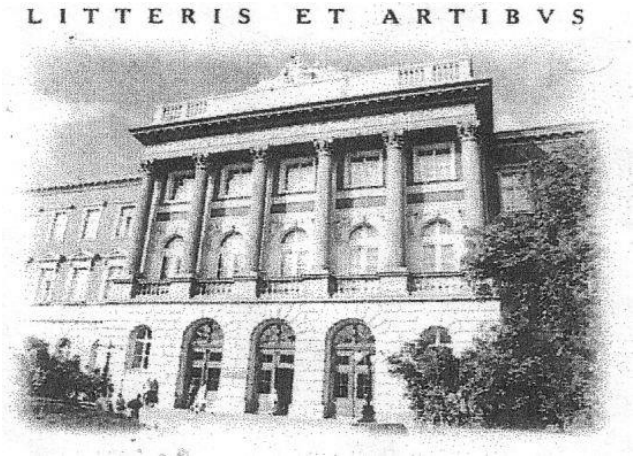
Міністерство освіти і науки України
Національний університет «Львівська політехніка»



Основи інформатики та інформаційних технологій.

НАВЧАЛЬНИЙ ПОСІБНИК

КОНСПЕКТ ЛЕКЦІЙ

з курсу
«Основи інформатики і програмування, частина 1»
для першого (бакалаврського) рівня освіти
спеціальності 105 – "Прикладна фізика та наноматеріали"

Затверджено
на засіданні кафедри
обчислювальної математики
та програмування
Протокол № 4 від 13.11.2019р.

Львів - 2019

Основи інформатики та інформаційних технологій. Конспект лекцій з курсу «Основи інформатики і програмування, частина 1» для першого (бакалаврського) рівня освіти спеціальності 105 – "Прикладна фізика та наноматеріали" / Укл.: Гнатів Л.Б., Ментинський С.М., Пелех Я.М., Федюк Є.М. - Львів: Самвидав, 2019.- 132 с.

Посібник складається з 15 лекцій з основних тем загального курсу інформатики. Посібник призначений для студентів бакалаврського рівня вищої освіти НУ “Львівська Політехніка”, спеціальності 105 Прикладна фізика та наноматеріали.

Укладачі: Гнатів Л.Б., к. ф-м. н., доцент,
Ментинський С.М., ст. викл,
Пелех Я.М., к. ф-м. н., доцент,
Федюк Є.М., к. ф-м. н., доцент.

Рецензенти: Глинський Я.М., к.ф.-м.н.,доцент кафедри ОМП
Каркульовський В.І., к.т.н.,доцент кафедри САПР

Рекомендовано Науково-методичною радою Національного університету «Львівська політехніка» як навчальний посібник для студентів бакалаврського рівня вищої освіти спеціальності 105 Прикладна фізика та наноматеріали.
(протокол № 9060 від 17 грудня 2019р.)

ЗМІСТ

Передмова

Базова підготовка студентів інженерних спеціальностей передбачає освоєння основних понять інформатики та інформаційних технологій. Важливим елементом цього циклу фундаментальних дисциплін є ознайомлення студентів із сучасним станом сфери інформаційних технологій комп'ютерної техніки та програмного забезпечення. Вивчення цього розділу завжди супроводжується суперечністю між прагненням студентів та викладачів до знайомства з новітніми технологіями та необхідністю вивчення базових понять та принципів, що історично сягають часів "докомп'ютерної ери". Вивчення фундаментальних засад використання комп'ютерної техніки та програмного забезпечення, а також принципів функціонування окремих прикладних програм загального призначення, таких як текстові редактори, процесори електронних таблиць, системи керування базами даних, є важливим компонентом базової освіти фахівців будь-якого технічного напряму.

Конспект лекцій містить відомості з теоретичних основ інформатики та комп'ютерної техніки, огляд можливостей та приципів роботи окремих офісних додатків (текстових редакторів, табличних процесорів, систем керування базами даних), а також засад організації і функціонування комп'ютерних мереж. Рекомендовано для студентів бакалаврського рівня вищої освіти спеціальності 105 «Прикладна фізика та наноматеріали» і укладено відповідно до робочої навчальної програми з дисципліни «Основи інформатики та програмування».

Лекція 1. Предмет, методи і завдання дисципліни.

1. Предмет і завдання дисципліни

2. Інформація та її властивості.

3. Вимірювання інформації. Інформаційні процеси.

4. Апаратне та програмне забезпечення інформаційних процесів.

1. Предмет і завдання дисципліни

Єдиного загальноприйнятого означення терміну «Інформатика» в літературі немає, проте, зважаючи на направленість курсу, можна сформулювати таке тлумачення:

Інформатика - це фундаментальна наука про інформацію та інформаційні процеси і способи та методи їх здійснення за допомогою комп'ютерної техніки. До ***інформаційних процесів*** відносять збір (отримання), зберігання, обробку, передачу, захист та використання інформації. Об'єктом інформатики виступають як комп'ютери, так і інформаційні системи.

Об'єктом інформатики виступають інформаційні системи, які забезпечують вирішення і організаційних задач, що виникають в технічних системах.

Інформаційна система – це сукупність програмно-апаратних засобів, способів і людей, які забезпечують збір, зберігання, обробку і передачу інформації для забезпечення підготовки і прийняття рішень.

Мета курсу полягає в набутті студентами знань і вмінь, необхідних всім без винятку користувачам для практичної роботи на персональних комп'ютерах, виховання інформаційної культури, отримання стартових можливостей в подальшому освоєнні вибраної спеціальності із застосуванням ПК, при вирішенні інженерних задач, дослідженні різних технологічних процесів.

Завдання навчальної дисципліни. Внаслідок вивчення навчальної дисципліни студент повинен бути здатним продемонструвати такі **результати навчання**:

- знати види сучасного програмного забезпечення;
- знати призначення і функціональні можливості сучасних операційних систем;
- знати можливості прикладного програмного забезпечення: текстового редактора, програми обробки електронних таблиць, системи керування базами даних;
- знати основні принципи функціонування локальних та глобальної комп’ютерних мереж;
- знати основи алгоритмізації та структуру процесу розв’язання прикладних задач за допомогою обчислювальної техніки;
- вміти користуватися персональним комп’ютером і працювати в мережі на рівні користувача;
- вміти використовувати текстовий редактор для підготовки текстів та технічної документації;
- вміти використовувати прикладне програмне забезпечення (електронні таблиці, системи керування базами даних тощо) для розв’язування типових задач;

- вміти розробляти програми на мові програмування високого рівня, дотримуючись вимог структурного програмування;

Вивчення навчальної дисципліни передбачає формування та розвиток у студентів компетентностей:

- Уміння думати абстрактно, здатність до аналізу та синтезу, що дозволяє формулювати висновки (діагноз) для різних типів складних управлінських задач, здійснювати планування, аналіз, контроль та оцінювання власної роботи та роботи інших осіб;
- Навички використання інформаційних та комунікативних технологій, впровадження комп'ютерних програм та використання існуючих.
- Уміння ідентифікувати, класифікувати та описувати роботу, пов'язану з прикладною фізикою та наноматеріалами, шляхом використання аналітичних методів і методів моделювання.

Результати навчання даної дисципліни деталізують такі **програмні результати навчання:**

- Здатність здійснювати пошук та аналізувати інформацію з різних джерел;
- Набуття гнучкого способу мислення, який дає можливість зрозуміти й розв'язати проблеми та задачі, зберігаючи при цьому критичне відношення до усталених наукових концепцій;
- Уміння думати абстрактно, здатність до аналізу та синтезу, що дозволяє формулювати висновки (діагноз) для різних типів складних управлінських задач, здійснювати планування, аналіз, контроль та оцінювання власної роботи та роботи інших осіб.

2. Інформація та її властивості

У літературі можна знайти досить багато визначень терміну «інформація» , що відображають різні підходи до тлумачення цього поняття. Наприклад:

- Відомості про навколишній світ і процеси, які в ньому відбуваються, що сприймаються людиною або спеціальним пристроєм.
- Повідомлення, що інформують про положення справ, про стан чого-небудь. (Науково-технічна і газетна інформації, засоби масової інформації - друк, радіо, телебачення, кіно).

Інформація та її властивості є об'єктом дослідження цілого ряду наукових дисциплін, таких як теорія інформації (математична теорія систем передачі інформації), кібернетика (наука про зв'язок та управлінні в машинах і тварин , а також в людському суспільстві), семіотика (наука про знаки в знакових системах) , теорія масової комунікації (дослідження засобів масової інформації та їх впливу на суспільство), інформатика (вивчення процесів збору, перетворення, зберігання, захисту, пошуку і передачі всіх видів інформації і засобів їх автоматизованої обробки), інформодинаміка (наука про відкриті інформаційні системи), інформаціологія (наука про отримання, збереження і передачу інформації для різних множин об'єктів) і т. д.

В інформатиці найбільш часто використовується наступне визначення цього терміну: ***Інформація - це усвідомлені відомості про навколишній світ , які є об'єктом***

зберігання , перетворення , передачі і використання. Відомості - це знання , виражені в сигналах , повідомленнях, вістях, оповіщеннях і т. д. Незважаючи на те, що єдиного строгого визначення інформації не існує, є можливість описати цей термін через характерні властивості:

Об'єктивність інформації. Об'єктивний - що існує поза і незалежно від людської свідомості. Інформація - це відображення зовнішнього об'єктивного світу. Інформація об'єктивна, якщо вона не залежить від методів її фіксації, чиєїсь думки, судження. Об'єктивну інформацію можна отримати за допомогою справних датчиків , вимірювальних приладів. Відбиваючись у свідомості людини, інформація може спотворюватися (більшою чи меншою мірою) в залежності від думки, судження, досвіду, знань конкретного суб'єкта, і, таким чином, перестати бути об'єктивною.

Достовірність інформації. Інформація достовірна , якщо вона відображає справжній стан справ. Об'єктивна інформація завжди достовірна, але достовірна інформація може бути як об'єктивною, так і суб'єктивною. Достовірна інформація допомагає прийняти нам правильне рішення. Недостовірною інформація може бути з наступних причин: *навмисне спотворення* (дезінформація) або *ненавмисне перекручення суб'єктивного походження, спотворення в результаті дії перешкод* («зіпсований телефон») і недостатньо точних засобів її фіксації.

Повнота інформації. Інформацію можна назвати повною, якщо її достатньо для розуміння і прийняття рішень. Неповна інформація може привести до помилкового висновку або рішенням .

Точність інформації визначається ступенем її близькості до реального стану об'єкта , процесу, явища і т. п.

Актуальність інформації - важливість для теперішнього часу, злободенність, нагальність. Тільки вчасно отримана інформація може бути корисна .

Корисність (цінність) інформації. Корисність може бути оцінена стосовно потреб конкретних її споживачів і оцінюється за тим завданням, які можна вирішити з її допомогою.

Для позначення інформації можуть використовуватися такі поняття як: дані, інформація і знання. Ці поняття часто використовуються як синоніми, проте між цими поняттями існують принципові відмінності.

Термін **дані** походить від слова **data** - факт, а **інформація** (**informatio**) означає роз'яснення, виклад, тобто відомості або повідомлення.

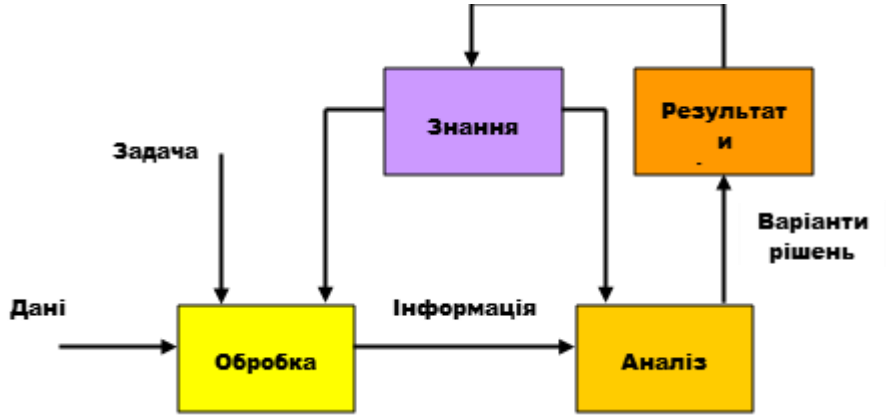
Дані - це сукупність відомостей, зафіксованих на певному носієві у формі, придатній для постійного зберігання, передачі та обробки. Перетворення і обробка даних дозволяє отримати інформацію.

Інформація - це результат перетворення і аналізу даних. Відмінність інформації від даних полягає в тому, що дані - це фіксовані відомості про події і явища, які зберігаються на певних носіях, а інформація з'являється в результаті обробки даних при вирішенні конкретних задач. Наприклад, в базах даних зберігаються різні дані, а за певним запитом система управління базою даних видає необхідну інформацію.

Існують й інші означення інформації, наприклад, інформація – це відомості про об'єкти і явища навколишнього середовища, їх параметри, властивості і стан, які зменшують міру невизначеності, неповноту знань про них.

Знання – це зафіксована і перевірена практикою оброблена інформація, яка використовувалася і може багато разів використовуватися для прийняття рішень.

Знання – це вид інформації, яка зберігається в базі знань і відображає знання фахівця в конкретній предметній області. Знання – це інтелектуальний капітал.



Формальні знання можуть бути у вигляді документів (стандартів, нормативів), що регламентують прийняття рішень або підручників, інструкцій з описом розв'язування задач. Неформальні знання – це знання і досвід фахівців в певній предметній області.

Необхідно відзначити, що універсальних означень понять **дані**, **інформація**, **знання** немає, вони трактуються по-різному. Прийняття рішень здійснюються на основі отриманої інформації і наявних знань.

Прийняття рішень – це вибір найкращого в певному розумінні варіанту рішення із множини допустимих на підставі наявної інформації.

Взаємозв'язок даних, інформації і знань в процесі прийняття рішень представлено на малюнку.

Для вирішення поставленого завдання фіксовані дані обробляються на підставі наявних знань, далі отримана інформація аналізується за допомогою наявних знань. На підставі аналізу, пропонуються всі допустимі рішення, а в результаті вибору ухвалюється одне рішення, найкраще в певному розумінні. Результати рішення доповнюють знання.

3. Вимірювання інформації. Інформаційні процеси.

У ЕОМ застосовується двійкова система числення, тобто всі числа в комп'ютері представляються за допомогою нулів і одиниць, тому комп'ютер може обробляти тільки інформацію, представлену в цифровій формі.

Для перетворення числової, текстової, графічної, звукової інформації в цифрову необхідно застосувати кодування. Кодування – це перетворення даних одного типу в дані іншого типу. У ЕОМ застосовується система двійкового кодування, заснована на представленні даних послідовністю двох знаків: 1 і 0, які називаються двійковими цифрами (binary digit – скорочено bit).

Таким чином, одиницею інформації в комп'ютері є один біт, тобто двійковий розряд, який може набувати значення 0 або 1. Вісім послідовних бітів складають байт. В одному байті можна закодувати значення одного символу з 256 можливих (256 = 28). Крупнішою одиницею інформації є кілобайт (Кбайт), рівний 1024 байтам (1024 = 210). Ще крупніші одиниці вимірювання даних: мегабайт, гігабайт, терабайт (1 Мбайт = 1024 Кбайт; 1 Гбайт = 1024 Мбайт; 1 Тбайт = 1024 Гбайт).

Цілі числа кодуються двійковим кодом досить просто (шляхом ділення числа на два). Для кодування нечислової інформації використовується наступний алгоритм: всі можливі значення кодованої інформації нумеруються і ці номери кодуються за допомогою двійкового коду.

Наприклад, для представлення текстової інформації використовується таблиця нумерації символів або таблиця кодування символів, в якій кожному символу відповідає ціле число (порядковий номер). Вісім двійкових розрядів можуть закодувати 256 різних символів.

Існуючий стандарт ASCII (8 – розрядна система кодування) містить дві таблиці кодування – базову і розширену. Перша таблиця містить 128 основних символів, в ній розміщені коди символів англійського алфавіту, а в другій таблиці кодування містяться 128 розширених символів.

Оскільки в цей стандарт не входять символи національних алфавітів інших країн, то в кожній країні 128 кодів розширених символів замінюються символами національного алфавіту. В даний час існує багато таблиць кодування символів, в яких коди розширених символів замінено символами національного алфавіту.

Так, наприклад, кодування символів російської мови Widows-1251 використовується для комп'ютерів, які працюють під ОС Windows. Інше кодування для російської мови – це КОИ-8, яка також широко використовується в комп'ютерних мережах і російському секторі Інтернет.

В даний час існує універсальна система UNICODE, заснована на 16-розрядному кодуванні символів. Ця 16-розрядна система забезпечує універсальні коди для 65536 різних символів, тобто в цій таблиці можуть розміститися символи мов більшості країн світу.

Для кодування графічних даних застосовується, наприклад, такий метод кодування як растр. Координати точок і їх властивості описуються за допомогою цілих чисел, які кодуються за допомогою двійкового коду. Так чорно-білі графічні об'єкти можуть бути описані комбінацією точок з 256 градаціями сірого кольору, тобто для кодування яскравості будь-якої точки досить 8-розрядного двійкового числа.

Режим представлення кольорової графіки в системі RGB (Red, Green, Blue) з використанням 24 розрядів (по 8 розрядів для кожного з трьох основних кольорів) називається повнокольоровим. Для повнокольорового режиму в системі CMYK (Cyan, Magenta, Yellow, blacK) необхідно мати 32 розряди (чотири кольори по 8 розрядів).

4. Апаратне та програмне забезпечення інформаційних процесів.

Комп'ютер – це пристрій або засіб, призначений для обробки інформації. Комп'ютер може обробляти лише інформацію, представлену в числовій формі. Інформацію в іншій формі представлення для введення в комп'ютер необхідно перетворити в числову форму.

Сучасним комп'ютерам передували ЕОМ декількох поколінь. У розвитку ЕОМ виділяють п'ять поколінь. В основу класифікації закладена елементна база, на якій будуються ЕОМ.

У 1943 році була створена обчислювальна машин ЕОМ першого покоління на базі електронних ламп.

Друге покоління (50-60 р.р.) комп'ютерів побудоване на базі напівпровідникових елементів (транзисторах).

Основна елементна база комп'ютерів третього покоління (60-70 р.р.) – інтегральні схеми малої і середньої інтеграції.

У комп'ютерах четвертого покоління (70 – до теперішнього часу) застосовані великі інтегральні схеми (мікропроцесори). Застосування мікропроцесорів в ЕОМ дозволило створити персональний комп'ютер (ПК), особливістю якого є невеликі розміри і низька вартість.

В даний час використовуються ЕОМ п'ятого покоління, які розробляються на надвеликих інтегральних схемах.

Існує і інші різні системи класифікації ЕОМ:

- За продуктивністю і швидкодією
- За призначенням
- За рівнем спеціалізації
- За типом використаного процесора
- За особливостями архітектури
- За розмірами

Розглянемо схему класифікації ЕОМ, виходячи з їх обчислювальної потужності і габаритів.

Суперкомп'ютери – це найпотужніші за швидкодією і продуктивністю обчислювальні машини. До СУПЕРЕОМ відносяться “Cray” і “IBM SP2” (США). Використовуються для вирішення великомасштабних обчислювальних задач і моделювання, для складних обчислень в аеродинаміці, метеорології, фізиці високих енергій, також знаходять застосування і у фінансовій сфері.

Великі машини або мейнфрейми (Mainframe) використовуються у фінансовій сфері, оборонному комплексі, застосовуються для комплектування відомчих, територіальних і регіональних обчислювальних центрів.

Середні ЕОМ широкого призначення використовуються для управління складними технологічними виробничими процесами.

МІНІ-ЕОМ орієнтовані на використання керівниками обчислювальних комплексів, як мережеві сервери.

Мікро-ЕОМ - це комп'ютери, у яких в якості центрального процесора використовується мікропроцесор. До них відносяться вбудовані мікро-ЕОМ (вбудовані в різне обладнання, апаратуру або прилади) і персональні комп'ютери PC.

Сучасні персональні комп'ютери мають практично ті ж характеристики, що й МІНІ-ЕОМ вісімдесятих років. На базі цього класу ЕОМ будуються автоматизовані робочі місця (АРМ) для фахівців різного рівня, використовуються як засіб обробки інформації в інформаційних системах.

До персональних комп'ютерів відносяться настільні і переносні ПК. До переносних ЕОМ відносяться Notebook (блокнот або записник) і кишенькові персональні комп'ютери (Personal Computers Handheld - Handheld PC, Personal Digital Assistants – PDA і Palmtop).

Архітектура ЕОМ включає в себе як структуру, що відображає склад ПК, так і програмно-математичне забезпечення. Структура ЕОМ – сукупність елементів і зв'язків між ними. Основним принципом побудови всіх сучасних ЕОМ є програмне управління.

Основи вчення про архітектуру обчислювальних машин були закладені Джоном фон Нейманом. Сукупність цих принципів породила класичну (фон-нейманівську) архітектуру ЕОМ.

Фон Нейман не лише висунув основоположні принципи логічної побудови ЕОМ, але і запропонував її структуру, представлену на малюнку.

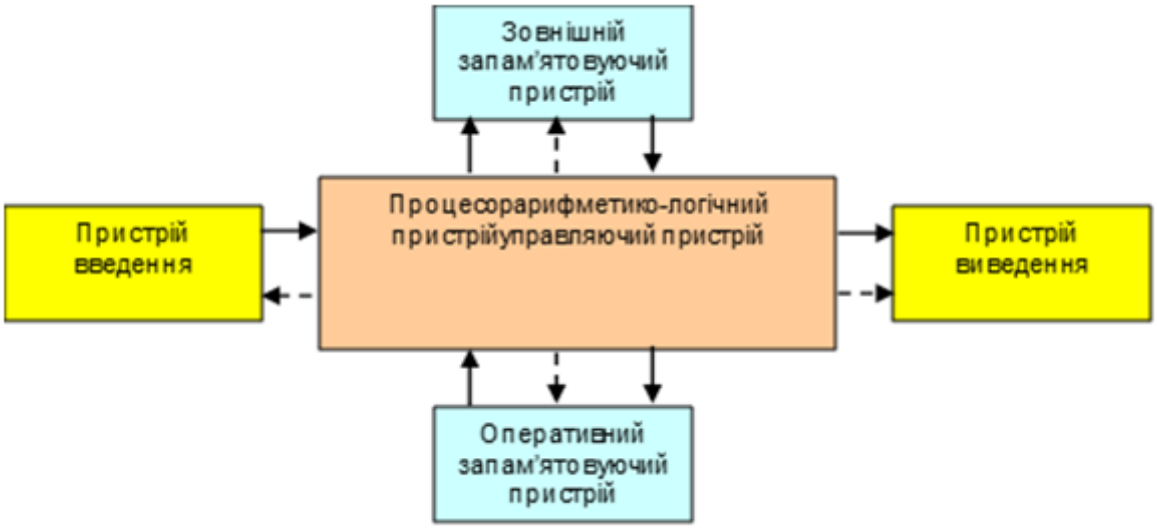
Положення фон Неймана:

- Комп'ютер складається з декількох основних пристроїв (арифметико-логічний пристрій, управляючий пристрій, оперативна пам'ять, зовнішня пам'ять, пристрої введення і виведення)
- Арифметико-логічний пристрій – виконує логічні і арифметичні дії, необхідні для переробки інформації, що зберігається в пам'яті
- Управляючий пристрій – забезпечує управління і контроль всіх пристроїв комп'ютера (управляючі сигнали вказані пунктирними стрілками)
- Дані, які зберігаються в запам’ятовуючому пристрої, представлені в двійковій формі
- Програма, яка задає роботу комп'ютера, і дані зберігаються в одному і тому ж запам’ятовуючому пристрої
- Для введення і виведення інформації використовуються пристрої введення і виведення

Один з найважливіших принципів – принцип програми, що зберігається, – вимагає, щоб програма закладалася в пам'ять машини так само, як в неї закладається вихідна інформація.

Арифметико-логічний пристрій і пристрій управління в сучасних комп'ютерах утворюють процесор ЕОМ. Процесор, який складається з однієї або декількох великих інтегральних схем, називається **мікропроцесором** або **мікропроцесорним комплектом**.

Процесор – функціональна частина ЕОМ, що виконує основні операції по обробці даних і управлінню роботою інших блоків. Процесор є перетворювачем інформації, що



надходить з пам'яті і зовнішніх пристроїв.

Запам’ятовуючі пристрої забезпечують зберігання початкових (вхідних) і проміжних даних, результатів обчислень, а також програм. Вони включають: оперативні (ОЗП), супероперативні (СОЗП), постійні (ПЗП) і зовнішні (ЗЗП) запам’ятовуючі пристрої.

Оперативні ЗП зберігають інформацію, з якою комп'ютер працює безпосередньо зараз (резидентна частина операційної системи, прикладна програма, оброблювані дані). У СОЗП зберігаються дані, що найчастіше використовуються процесором. Лише та інформація, яка зберігається в СОЗП і ОЗП, безпосередньо доступна процесору.

Зовнішні запам’ятовуючі пристрої (накопичувачі на магнітних дисках, наприклад, жорсткий диск або вінчестер) за місткістю є набагато більші, ніж ОЗП, але водночас з істотно повільнішим доступом. Вони використовуються для тривалого зберігання великих об'ємів інформації. Наприклад, операційна система (ОС) зберігається на жорсткому диску, але при запуску комп'ютера резидентна частина ОС завантажується в ОЗП і знаходиться там до завершення сеансу роботи ПК.

ПЗП (постійні запам’ятовуючі пристрої) і ППЗП (перепрограмовані запам’ятовуючі пристрої) призначені для постійного зберігання інформації, яка записується туди при її виготовленні, наприклад, ППЗП для BIOS.

Як пристрій введення інформації служить, наприклад, клавіатура. Як пристрій виведення – дисплей, принтер і т. д.

У побудованій за схемою фон Неймана ЕОМ відбувається послідовне зчитування команд з пам'яті і їх виконання. Номер (адреса) чергового елементу пам'яті, з якої витягуватиметься наступна команда програми, вказується спеціальним пристроєм – лічильником команд в пристрої управління.

Лекція 2. Огляд операційних систем. Текстові процесори.

1. *Призначення та функції операційної системи.*
2. *Особливості поширених операційних систем.*
3. *Призначення та основні функції текстового процесора.*
4. *Створення та форматування текстових документів.*
5. *Шаблони документів.*

1. Призначення та функції операційної системи.

Операційна система (ОС) – це сукупність програмних засобів, що здійснюють управління ресурсами ЕОМ, запуск прикладних програм і їх взаємодію із зовнішніми пристроями і іншими програмами, а також забезпечують діалог користувача з комп'ютером. Вона завантажується при увімкненні ПК і підтримує роботу користувача і інших програм із його ресурсами. Під ***ресурсом*** розуміємо будь-який компонент апаратної частини комп'ютера, а також можливості, які ним надаються.

Функції типової операційної системи:

- Керування виконанням програм. Щоб запустити програму потрібно виконати такі дії: завантажити в ОЗП програму та дані, ініціалізувати файли (тобто виконати дії, які дозволяють читати або писати дані з файла або у файл), ініціалізувати пристрої вводу-виводу, підготувати інші ресурси.
- Керування доступом до пристроїв ввода-вивода. Кожен пристрій має свій набір команд. ОС дає користувачу "єдинообразний" інтерфейс, який ховає деталі. Команди доступу до пристроїв прості і зрозумілі. Наприклад, диски різні, а процедура відкриття компакт-диску або вінчестера або цифрового фотоапарата однакова.
- Керування файловою системою
- Керування системним доступом. Якщо обчислювальна система доступна кільком користувачам, то ОС повинна керувати загальнодоступними ресурсами та розв'язувати конфліктні ситуації.
- Знаходження та обробка помилок. Наприклад, помилки в апаратному забезпеченні, збої пристроїв, програмні переривання, спроба звернутись до комірки пам'яті, доступ до якої заборонено тощо. В кожному випадку ОС повинна виконати дії, які мінімізують вплив помилки на роботу додатку – від простого повідомлення про помилку до аварійного завершення додатку.
- Облік використання ресурсів і відображення параметрів продуктивності. Ця інформація важлива для покращення налаштування ОС з метою підвищення продуктивності її роботи.

Операційні системи розрізняються особливостями реалізації алгоритмів управління ресурсами комп'ютера, областями використання.

- Однозадачні і багатозадачні системи;
- Однокористувацькі та багатокористувацькі системи;
- Однопроцесорні і багатопроцесорні системи;
- Локальні і мережеві системи.

За кількістю одночасно виконуваних завдань операційні системи діляться на два класи:

- Однозадачні (MS-DOS);
- Багатозадачні (OS/2, Unix, Windows).

Залежно від областей використання багатозадачні ОС підрозділяються на три типи:

- Системи пакетної обробки (ОС ЄС);
- Системи з розділенням часу (Unix, Linux, Windows);
- Системи реального часу (RT11).

Системи пакетної обробки призначені для вирішення завдань, які не вимагають швидкого отримання результатів. Головною метою ОС пакетної обробки є максимальна пропускна спроможність або вирішення максимального числа завдань в одиницю часу. Ці системи забезпечують високу продуктивність при обробці великих обсягів інформації, але знижують ефективність роботи користувача в інтерактивному режимі.

У ***системах з розділенням часу*** для виконання кожного завдання виділяється невеликий проміжок часу, і жодне завдання не займає процесор надовго. Якщо цей проміжок часу вибраний мінімальним, то створюється видимість одночасного виконання декількох завдань. Ці системи володіють меншою пропускною спроможністю, але забезпечують високу ефективність роботи користувача в інтерактивному режимі.

Системи реального часу застосовуються для управління технологічним процесом або технічним об'єктом, наприклад, літальним об'єктом, верстатом і т.д.

Однією з важливих властивостей ОС є наявність в ній засобів підтримки багатопроцесорної обробки даних. Такі засоби існують в OS/2, Net Ware, WidowsNT|. За способом організації обчислювального процесу ці ОС можуть бути розділені на асиметричні і симетричні.

Однією з найважливіших ознак класифікації ЕОМ є поділ їх на локальні і мережеві. Локальні ОС застосовуються на автономних ПК або ПК, які використовуються в комп'ютерних мережах як клієнт.

Архітектура ОС. Під архітектурою операційної системи розуміють структурну і функціональну організацію ОС на основі деякої сукупності програмних модулів. До складу ОС входять виконувані і об'єктні модулі стандартних для даної ОС форматів, програмні модулі спеціального формату (наприклад, завантажувач ОС, драйвери вводу-виводу), конфігураційні файли, файли документації, модулі довідкової системи і т.д.

Класичною вважається архітектура ОС, заснована на концепції ієрархічної багаторівневої машини, привілейованому ядрі і користувальницькому режимі роботи транзитних модулів. Модулі ядра виконують базові функції ОС: управління процесами, пам'яттю, пристроями введення-виведення і т.п. Ядро складає серцевину ОС, без якої вона є повністю непрацездатною і не може виконати ні одну зі своїх функцій. У ядрі вирішуються внутрішньосистемні задачі організації обчислювального процесу, недоступні для додатка.

Особливий клас функцій ядра служить для підтримки додатків, створюючи для них так зване прикладне програмне середовище. Додатки можуть звертатися до ядра з

запитами - системними викликами - для виконання тих чи інших дій, наприклад, відкриття та читання файлу, отримання системного часу, виведення інформації на дисплей і т.д. Функції ядра, які можуть викликатися додатками, утворять інтерфейс прикладного програмування - API (Application Programming Interface).

Решта модулі ОС виконують не настільки важливі функції, як ядро, і є транзитними. Наприклад, це можуть бути програми архівування даних, дефрагментації диска, стиснення дисків, очищення дисків і т.п.

Допоміжні модулі звичайно поділяються на групи:

- утиліти - програми, які виконують окремі завдання управління і супроводу обчислювальної системи;
- системні програми для обробки даних - текстові та графічні редактори (Paint, Imaging в Windows 2000), компілятори та ін;
- програми надання користувачу додаткових послуг (спеціальний варіант для користувача інтерфейсу, калькулятор, ігри, засоби мультимедіа Windows 2000);
- бібліотеки процедур різного призначення, спрощення розробки додатків, наприклад, бібліотека функцій вводу-виводу, бібліотека математичних функцій і т.п.

Ці модулі ОС оформляються як звичайні додатки, звертаються до функцій ядра за допомогою системних викликів і виконуються в режимі користувача (user mode). У цьому режимі забороняється виконання деяких команд, які пов'язані з функціями ядра ОС (управління ресурсами, розподіл і захист пам'яті і т.п.).

2. Особливості поширених операційних систем.

Операційна система MS-DOS. Однією з найпоширеніших операційних систем до середини 90-х років була дискова операційна система фірми Microsoft MS-DOS (Microsoft Disk Operating System). У сучасних ОС Windows для роботи з командами DOS використовується командний рядок, який можна викликати: Пуск/Выполнить, у вікні діалогу ввести cmd і натиснути ОК. Інший спосіб виклику командного рядка – Пуск/Программы/Стандартные/Командная строка.

В операційну систему MS-DOS входять наступні основні модулі:

- Базова система введення – виведення (BIOS);
- Блок початкового завантаження (Boot Record);
- Модуль розширення BIOS (IO.SYS);
- Модуль обробки переривань (MSDOS.SYS);
- Командний процесор (COMMAND.COM);
- Файли-драйвери, які після їх завантаження в пам'ять забезпечують роботу таких пристроїв, як миша, CD-ROM та ін.
- Утиліти ОС, що виконують різні сервісні функції (форматування дисків та ін.).

Основними поняттями призначеного для користувача інтерфейсу в середовищі ***Windows*** є вікно і піктограма. Все, що відбувається в рамках оболонки Windows, в певному сенсі являє собою або операцію з піктограмою, або операцію з вікном (або у вікні). Стандартизована в середовищі Windows і структура вікон і розташування елементів управління ними. Стандартизовані набори операцій і структура меню для сервісних

програм. Стандартні операції виконують за допомогою миші для всіх сервісних і прикладних програм.

Windows являє собою графічну оболонку. Від користувача не потрібне введення директив з клавіатури у вигляді текстових рядків. Необхідно тільки уважно дивитися на екран і вибирати з запропонованого набору потрібну операцію за допомогою маніпулятора миші. Курсор миші слід позиціонувати на поле потрібної директиви меню, або на що цікавить піктограму, або на полі перемикача систем розраховані на виконання в даний момент тільки однієї програми. В рамках Windows користувач може запустити декілька програм для паралельного (незалежного) виконання. Кожна з виконуваних програм має своє власне вікно. Перемикання між виконуваними програмами проводиться за допомогою миші фіксацією курсора у вікні потрібної програми.

LINUX – багатокористувацька і багатозадачна операційна система, яка належить до UNIX-подібних ОС.

Обліковий запис (account) в багатокористувацьких обчислювальних системах є звичайні користувачі, які запускають прикладні програми, і адміністратори, які можуть запускати програми, що змінюють саму ОС. Щоб розрізнити таких користувачів і, відповідно, надати ним різні права роботи в ОС LINUX обов'язково для кожного користувача створюють обліковий запис.

Обліковий запис – об'єкт системи Linux, за допомогою якого операційна система веде облік роботи користувача в системі. Облікові записи створюють під час інсталяції системи або після інсталяції.

Що входить в обліковий запис:

- ***Loqin*** – назва користувача. Linux розрізняє малі та великі букви. Тому Student student STUDENT – різні логіни.
- ***Ідентифікатор користувача UID*** (User ID). Кожному логіну в ОС відповідає ідентифікатор користувача UID. UID - унікальне число, яке однозначно ідентифікує в системі обліковий запис.
- ***GID - ідентифікатор групи***. Групи користувачив використовують для організації доступу до групових ресурсів (наприклад, до мережного диску). В Linux кожний користувач повинен належати як мінімум до однієї групи (групи за замовчанням).
- ***Прізвище та ім'я людини***, якій належить обліковий запис. Ця інформація потрібна людям, щоб ідентифікувати логін. Ім'я та прізвище можна задавати необовязково своє.
- ***Назва домашнього каталога.*** Домашній каталог містить права користувача в системі, налаштування робочого середовища, каталоги та файли користувача. Доступ інших користувачів до цього каталогу обмежений.
- ***Назва програми- інтерпретатора командної стрічки***. Інтерпретатор – це програма, яка використовується для організації діалога людини і компютера.

Операційні системи UNIX, LINUX працюють в режимі командної стрічки. Для зручності роботи пересічного користувача до ОС LINUX інсталюють графічну оболонку. Є різні оболонки: Education Open Suse, UBUNTU. Оболонка надає користувачу дружній WINDOWS-подібний інтерфейс.

3. Призначення та основні функції текстового процесора.

Текстовий редактор – комп'ютерна програма, призначена для створення й зміни текстових файлів, а також їхнього перегляду на екрані, виводу на друк, пошуку фрагментів тексту й т.п. Умовно виділяють два типи редакторів. Перший тип орієнтований на роботу з послідовністю символів у текстовому файлі. Такі редактори забезпечують розширену функціональність – підсвічування синтаксису, сортування рядків, шаблони, конвертацію кодувань, показ кодів символів і т.п. Другий тип текстових редакторів (їх інколи називають текстовими процесорами) має розширені функції форматування тексту, впровадження в нього графіки, формул, таблиць і об'єктів. До текстових редакторів можна віднести Microsoft Word, Lotus Word Pro, Corel Word Perfect, Star Office, з якого виріс OpenOffice, та інші.

OpenOffice.org Writer – програма для створення й обробки текстових документів. Представлення WYSIWIG (від англ. What You See Is What You Get) дозволяє переглядати на екрані готовий до друку документ. Відформатовані символи відображаються на екрані так, як вони будуть виглядати при друці.

Текстові редактори можуть забезпечувати виконання різноманітних функцій:

- редагування рядків тексту;
- використання різних шрифтів символів;
- копіювання й перенесення частини тексту з одного місця на інше або з одного документа в інший;
- контекстний пошук і заміну частин тексту;
- задання довільних міжрядкових проміжків;
- автоматичний перенос слів на новий рядок;
- автоматичну нумерацію сторінок;
- вирівнювання країв абзацу;
- створення таблиць і побудова діаграм;
- перевірка правопису слів і підбір синонімів;
- побудова змісту і предметного покажчика;
- роздрук підготовленого тексту на принтері і т.п.

4. Створення та форматування текстових документів.

- Редагування тексту

Для виправлення тексту спочатку необхідно встановити курсор у те місце документа, куди потрібно внести зміни. Наступним кроком буде редагування. Це може бути видалення, додавання або заміна символів, переміщення або видалення частини тексту або зміна регістра тексту.

Зміни вносяться саме в те місце, де в цей момент розташований курсор (чорна вертикальна миготлива лінія). Вказівник мишки (тонка чорна немиготлива вертикальна лінія) може перебувати в іншому місці або зовсім бути відсутнім.

Видалення символів відбувається за допомогою клавіш Delete і Backspace (стрілка вліво), причому перша видаляє один символ правіше від курсора, а друга – лівіше. Якщо при цьому утримувати натиснутою клавішу Ctrl, то відбудеться видалення групи символів

до кінця слова чи до початку слова відповідно. Щоб повернути текст, вилучений помилково, потрібно виконати команду Edit → Undo (Виправлення → Скасувати), скористатись відповідною кнопкою на панелі інструментів або комбінацією клавіш Ctrl+Z.

- Виділення тексту

Більшість методів редагування Writer вимагають попереднього виділення фрагмента тексту. Обрана після цього команда застосовується тільки до виділеного фрагмента.

Виділення дозволяє чітко визначити частину документа, на яку подіє команда, – від одного символу або малюнка до цілого документа. Наприклад, спочатку виділяється фрагмент тексту, а потім вже натискається клавіша Delete і видаляється увесь фрагмент тексту. У виділеному фрагменті тексту колір фону і колір тексту міняються на протилежні.

Виділяти текст можна як мишкою так і за допомогою клавіатури. Основний спосіб виділення за допомогою клавіатури – тримати натиснутою клавішу Shift і використовувати на клавіші управління курсором (клавіші зі стрілками). При цьому не тільки переміщається курсор, але й виділяється текст. Натискання на клавіші управління курсором вліво і вправо у сполученні з клавішею Shift виділяє по одному символу, а вгору і вниз – по одному рядку у відповідному напрямку.

Інший спосіб виділення за допомогою клавіатури – використання клавіші F8. Курсор встановлюють на початок фрагмента, натискають клавішу F8 і встановлюють курсор у кінець фрагмента (чи за допомогою миші, чи за допомогою клавіатури). Щоб скасувати виділення, необхідно натиснути клавішу Esc на клавіатурі.

Нарешті, можна виділити весь документ, вибравши команду Edit → Select All (Виправлення → Виділити все) або натиснувши комбінацію клавіш Ctrl+A. Щоб скасувати виділення й знову побачити на екрані курсор, потрібно клацнути мишкою на будь-якому місці документа або натиснути клавішу зі стрілкою.

Щоб виділити текст за допомогою мишки, потрібно встановити вказівник мишки на початок фрагмента, натиснути ліву кнопку мишки і, не відпускаючи її, протягнути мишку до кінця фрагмента тексту, який треба виділити. Якщо при протягуванні було досягнуто межі робочого поля, Writer прокрутить документ, щоб можна було продовжити виділення.

- Виправлення виділеного фрагмента

Після виділення фрагмента тексту використовуються команди для виправлення або форматування. Можна просто вводити новий текст, який замінить виділений. Як тільки буде набрана перша буква, весь виділений текст автоматично буде вилучений, а новий текст буде вводитись на його місце. Щоб видалити виділений текст, слід натиснути клавіші Backspace або Delete. Щоб змінити регістр букв виділеного фрагмента (зробити всі букви малими або прописними), потрібно скористатись командою Chsnge Case (Регістр) із меню Format (Формат).

Якщо необхідно замінити існуючий текст на новий, то потрібно перейти у режим заміни. Поточні режими (вставки чи заміни) відображаються в рядку стану. Клацанням мишкою на написі у рядку стану можна перемикатися між режимами вставки – INSRT і

заміни – OVER. Ще один спосіб перемикання режимів – клавіша Insert на клавіатурі. У режимі OVER вихідний текст буде замінятися новим.

- Переміщення й копіювання тексту за допомогою миші

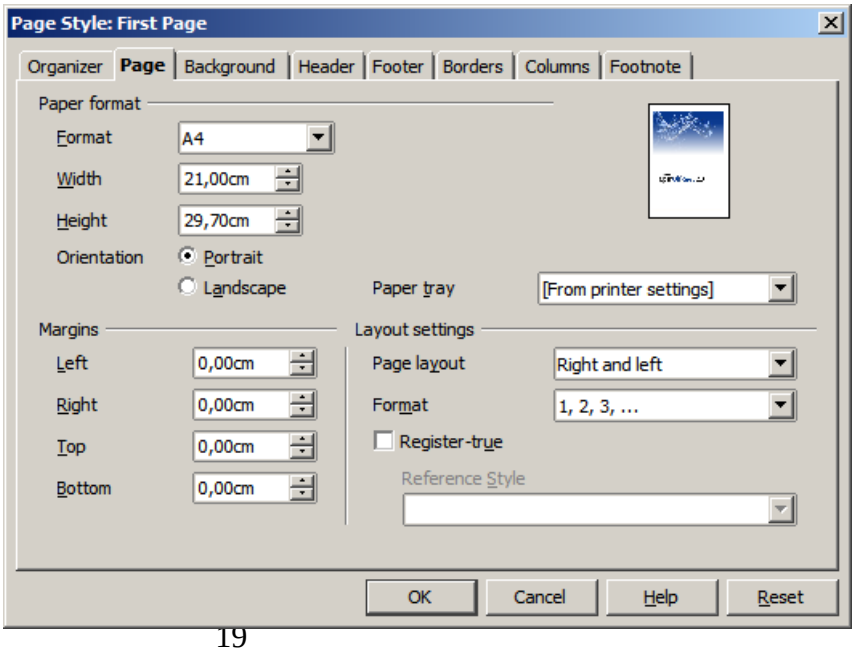
Якщо при переміщенні чи копіюванні тексту Ви волієте користуватись мишкою, то буде зручний наступний метод. Виділити текст, помістити вказівник мишки на виділеному фрагменті (вказівник набуде форми стрілки) і натиснути ліву кнопку мишки. Не відпускаючи лівої кнопки, перетягнути курсор у нове положення. Щоб скопіювати текст, потрібно при перетягуванні тримати натиснутою клавішу Ctrl. Якщо місце, куди потрібно помістити текст, не видно на екрані, то, тримаючи натиснутою ліву кнопку мишки, перемістіть вказівник мишки до краю вікна: документ автоматично покрутиться в потрібному напрямку. Місце нового розташування тексту може бути у тому ж документі Writer, в іншому документі Writer і навіть у документі, створеному в іншому додатку пакета OpenOffice.org.

Копіювання і переміщення фрагмента тексту можна здійснювати за допомогою команд Cut (Вирізати), Copy (Копіювати) і Paste (Вставити) з меню Edit (Виправлення). Команда Вирізати переміщає, а команда Копіювати відповідно копіює виділений фрагмент тексту в буфер обміну – спеціальну область пам'яті. Команда Вставити поміщає фрагмент із буфера обміну в позицію курсора. Командами Вирізати, Копіювати і Вставити також можна скористатись через контекстне меню фрагмента (відкривається при натисканні правої кнопки мишки) і за допомогою відповідних кнопок на панелі інструментів Стандартна. Вони стають активними, якщо виділено фрагмент тексту або інший об'єкт у тексті.

- Форматування документа

Основними об'єктами документа Writer є сторінка, абзац і символ. Для кожного із цих об'єктів необхідно задати значення параметрів форматування, які визначають зовнішній вигляд документа.

Вибір параметрів сторінки. Будь-який документ складається зі сторінок, тому на початку роботи над документом необхідно задати значення параметрів сторінки: формат (розмір сторінки), орієнтацію (книжкова чи альбомна), розмір полів та ін.



Ці параметри можна задати в діалоговому вікні Page Style (Стиль сторінки), яке викликається командою меню Format → Page… (Формат → Сторінка…). На вкладці Page задаються розміри полів, які визначають відстань від країв сторінки до межі області тексту. Там же задається розмір паперу та

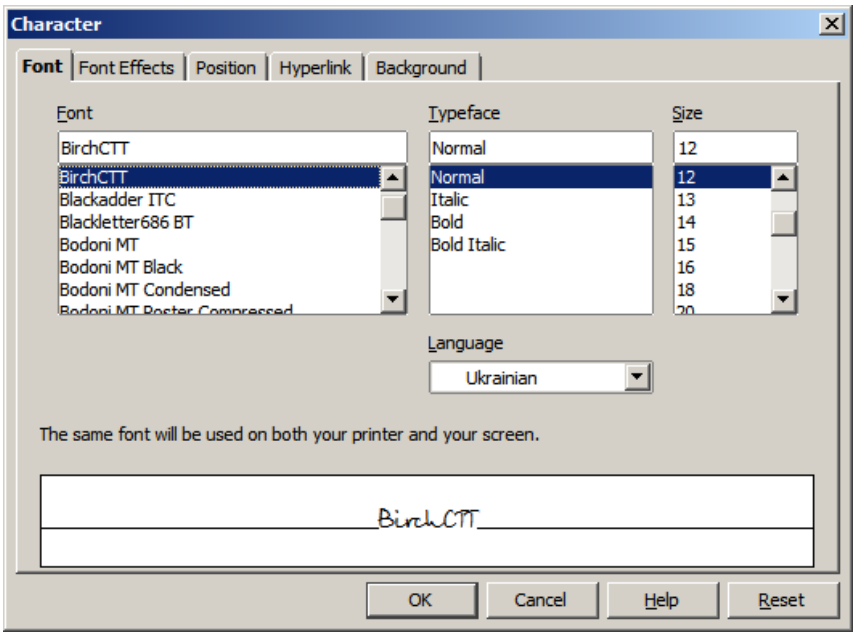
орієнтація сторінки. Крім того, на відповідних вкладках можна регулювати розташування колонтитулів. Колонтитули – це спеціальні області сторінки (верхній – Header та нижній – Footer), призначені для виводу на кожній сторінці документа однакового тексту, наприклад імені автора, назви документа й ін. У колонтитулі також можна вивести номер сторінки.

Щоб вивести номери сторінок, потрібно вибрати команду Insert → Fields → Page Number (Вставка → Поля → Номер сторінки). У цьому випадку номер сторінки буде вставлений туди, де перебував курсор.

Форматування символів. Символи – основні об'єкти, з яких складається текст. Символи – це букви, цифри, пробіли, знаки пунктуації, спеціальні символи (наприклад, @, *, &). Символи можна форматувати (змінювати їхній зовнішній вигляд).

Серед основних властивостей символів можна виділити наступні: шрифт, розмір, накреслення і кольори.

Шрифт – повний набір символів певного зовнішнього вигляду, включаючи великі і малі літери, розділові знаки, спеціальні символи, цифри й знаки арифметичних дій. Для кожного історичного періоду і різних країн характерний шрифт певного малюнка. Кожен шрифт має свою назву, наприклад DejaVu Serif, DejaVu Sans, Courier New та ін. Діалогове вікно Character (Символи), що викликається командою Format → Character… (Формат → Символи…), дозволяє задавати параметри символів.



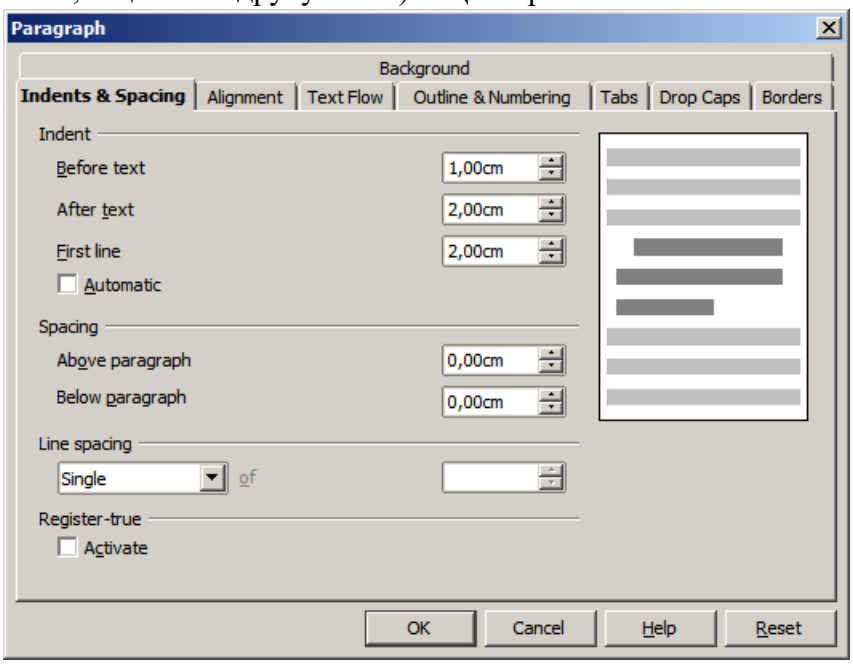
Одиницею вимірювання розміру шрифта є пункт (1 пт = 1/72 дюйма). Розміри шрифтів можна змінювати у широких межах, причому у більшості редакторів за замовчуванням використовується шрифт розміром 10 пт.

Крім нормального (звичайного) накреслення символів, також застосовують жирне, курсивне, жирне курсивне. Можна встановити додаткові параметри форматування символів: підкреслення символів лініями різних типів, зміна вигляду символів (піднятий, утоплений), зміна відстані між символами (розріджений, ущільнений) та ін.

Якщо планується багатокольоровий роздрук документа, то для різних груп символів доцільно задати різні кольори, обрані із пропонованої текстовим редактором палітри.

Форматування абзаців. Абзац із літературної точки зору – це частина тексту, що представляє собою закінчений за змістом фрагмент твору, закінчення якого служить природною паузою для переходу до нової думки.

У комп'ютерних документах абзацом вважається будь-який текст, що закінчується спеціальним керуючим символом (маркером) кінця абзацу. Введення кінця абзацу забезпечується натисканням клавіші Enter і відображається символом ¶ (якщо включено режим відображення символів, що не друкуються). Цей режим включається й відключається натисканням кнопки на панелі інструментів Стандартна.



Абзац може складатися з будь-якого набору символів, малюнків, об'єктів. Форматування абзаців дозволяє підготувати правильно й красиво оформлений документ. Діалогове вікно форматування абзацу відкривається за допомогою команди Format → Paragraph… (Формат → Абзац…). Команда Абзац також присутня у контекстному меню виділеного фрагменту тексту.

Важливим параметром при наборі тексту є його вирівнювання. Вирівнювання відображає розташування тексту щодо границь полів сторінки. Існує чотири способи вирівнювання абзаців:

- по лівому краю – лівий край рівний, а правий рваний;
- по центру – обидва краї мають нерівні обриси, однак кожен рядок абзацу симетричний щодо середини;
- по правому краю – правий край рівний, а лівий рваний;
- по ширині – обидва краї рівні, тобто розташовуються точно по границях сторінки.

Кнопки для команд вирівнювання абзацу розміщуються на панелі інструментів Форматування. Найчастіше абзац починається відступом першого рядка (абзацним відступом). Відступ може бути різних типів:

- Додатний відступ, коли перший рядок починається правіше від усіх інших рядків абзацу, застосовується у звичайному тексті.
- Від'ємний відступ, коли перший рядок виходить вліво щодо інших рядків абзацу, застосовується в словниках і означеннях.
- Нульовий відступ, застосовується для абзаців, вирівняних по центру.

Увесь абзац повністю може мати відступи зліва і справа, які відміряються від меж полів сторінки. Так, епіграф до художнього твору або реквізити адресата в заяві мають відступ зліва, а при виготовленні кутового штампа можна використати відступ справа. Також може бути необхідно відокремити текст абзацу від попереднього й наступного тексту.

Відстань між рядками документа можна змінювати, задаючи різні значення міжрядкових інтервалів (одинарний, подвійний і т.д.). Для візуального відділення абзаців один від одного потрібно встановлювати збільшені інтервали між абзацами. Міжрядковий інтервал вибирають за допомогою списку Line spacing (Міжрядковий інтервал), а інтервали перед чи після абзацу задають за допомогою лічильників Above Paragraph (Перед абзацом) та Below Paragraph (Після абзацу).

Вкладка Text Flow (Положення на сторінці) дозволяє встановити необхідний розподіл абзаців по сторінках, тобто заборонити або дозволити розривати абзац між сторінками, залишати на сторінці перший чи останній (висячий) рядок та ін.

Використання стилів. У процесі створення документа часто доводиться мати справу із заголовками. Як правило, заголовки відрізняються від основного тексту документа. Значно полегшує процес форматування заголовків використання стилів.

У загальному випадку, стилі – це набори характеристик, що визначають зовнішній вигляд і форматування тексту, до якого вони застосовуються. У Writer є п’ять категорій стилів:

- стиль Абзаца зачіпає всі абзаци, що використовують цей стиль;
- стиль Символа впливає на блок тексту всередині абзаца;
- стиль Сторінки впливає на форматування сторінки (размір сторінки, поля і т.п.);
- стиль Врізок (Фреймів) впливає на врізки та зображення;
- стиль Списку впливає на структуру, нумеровані та маркіровані списки.

Щоб застосувати стиль, досить вибрати текст, який потрібно змінити, а потім клацнути на кнопці розгортання списку стилів у панелі інструментів Форматування. Доступні в даний момент стилі відобразяться з заданими для них шрифтами і розмірами.

5. Шаблони документів

Шаблон – заготовка, що використовується для створення документів. Наприклад, Ви можете створити шаблон для комерційного звіту, який містить на першій сторінці логотип вашої компанії. Нові документи, які створюються на основі цього шаблона, будуть мвстити логотип вашей компанії на першій сторінці.

Шаблоны можуть містити необхідні елементи документа: текст, графіку, набір стилів і особливу інформацію користувачів (одиниці виміру, мову, заданий за замовчуванням принтер, настройки меню та панелей інструментів).

Всі документи в OpenOffice.org створюються на основі шаблонів. Якщо Ви не вказуєте шаблон, то при створенні нового документа Writer цей документ створюється на основі шаблона для текстових документів, заданого за замовчуванням. Якщо ж Ви не визначали шаблон за замовчуванням, Writer використовує порожній шаблон для текстових документів, який міститься в OpenOffice.org.

Щоб відкрити діалогове вікно Templates and Documents (Шаблони і документи), потрібно виконати одну з наступних дій:

- вибрати команду File → New → Templates and Documents (Файл → Створити → Шаблони і документи);
- натиснути кнопку  на панелі інструментів Стандартна.

З’явиться вікно Шаблони і документи. Ліворуч у вікні відображається список категорій. Виберіть категорію Шаблони, розкриється список готових шаблонів, у якому перераховані доступні шаблони або документи для обраної категорії. Виберіть шаблон або документ і клацніть Відкрити.

Шаблон містить інформацію про задане за замовчуванням форматування для нових текстових документів. При необхідності можна створити новий шаблон і використати його як шаблон за замовчуванням. Для цього потрібно виконати наступні дії:

- створити документ, який містить потрібний текст і стилі форматування;
- вибрати команду File → Templates → Save (Файл → Шаблони → Зберегти);
- у вікні, яке відкриється, в поле Новий шаблон ввести ім’я для шаблона;
- у списку Категорії вибрати потрібну категорію (категорія – папка, у якій буде зберігатись шаблон) і натиснути кнопку ОК для збереження створеного шаблона;
- вибрати команду File → Templates → Organize… (Файл → Шаблони → Управління);
- у списку категорій вибрати категорію, у якій був збережений шаблон;
- клацнути правою кнопкою мишки на потрібному шаблоні і вибрати команду Set As Default Template (Зробити шаблоном за замовчуванням);
- натиснути кнопку Закрити.

Л.Б. Гнатів, С.М. Ментинський, Я.М. Пелех, Є.М.Федюк

Лекція 3. Табличні процесори. Введення, редагування та форматування даних ЕТ.

1. *Робоча книга ЕТ, дії з листами.*
2. *Введення, редагування та форматування даних в таблиці.*
3. *Введення формул з використанням вбудованих функцій.*

1. Робоча книга ЕТ, дії з листами.

Табличний процесор - це прикладна програма, призначена для створення електронних таблиць і автоматизованої обробки табличних даних. Табличними процесорами є OpenOffice.org CALC, Microsoft Excel, Google SpreadSheet і т.п.

Електронна таблиця – електронна матриця, розділена на рядки і стовпці, на перетині яких утворюються комірки з унікальними іменами. Комірки є основним елементом таблиці, в які можуть вводитися дані і на які можна посилатись за іменами комірок. До даних відносяться: числа, дати, час доби, текст або символьні дані і формули.

До обробки табличних даних відноситься:

- проведення різних обчислень за допомогою формул і функцій, вбудованих в редактор;
- побудова діаграм;
- обробка даних в списках (Сортування, Автофільтр, Розширений фільтр, Форма, Підсумки, Зведена таблиця);
- вирішення задач оптимізації (Підбір параметра, Пошук рішення, Сценарії "що - якщо" й інші задачі);
- статистична обробка даних, аналіз і прогнозування.

Таким чином, **Calc** є не лише засобом автоматизації розрахунків, але і засобом моделювання різних ситуацій. Сфера застосування Calc: **планово – фінансові і бухгалтерські розрахунк**и, **облік матеріальних цінностей, системи підтримки прийняття рішень (СППР)** та ін.

Створення нової робочої книги При запуску Calc відкривається вікно, в якому відображається новий документ – Untitled 1.

Вікно має п'ять основних областей:

- рядок меню;
- панель інструментів;
- рядок стану;
- рядок формул (введення);
- зона вікна робочої книги.

Основна обробка даних здійснюється за допомогою команд з рядка меню. Панелі інструментів Стандартна і Форматування містять певні набори кнопок. Основна частина

кнопок призначена для виконання команд з рядка меню, які найбільш часто використовуються.

Рядок формул використовується для введення і редагування значень або формул в комірках чи діаграмах. ***Поле імені*** – це вікно зліва від рядка формул, в якому виводиться ім'я активної комірки.

Рядок стану розташований в нижній частині екрану і у ньому виводиться деяка додаткова інформація.

Документ Calc складається з робочих листів, кожен з яких є електронною таблицею. За замовчуванням відкритими є три робочих листи, перехід до яких можна здійснити, клацаючи на ярличках, розташованих внизу книги. При необхідності в документ можна додати робочі листи або видалити їх з документа.

Кнопки прокрутки ярликів здійснюють прокрутку ярликів документа. Крайні кнопки здійснюють прокрутку до першого і останнього ярлика. Внутрішні кнопки здійснюють прокрутку до попереднього і наступного ярлика.

Основні поняття електронної таблиці: заголовок стовпця, заголовок рядка, комірка, ім'я комірки, маркер виділення, маркер заповнення, активна комірка, рядок формул, поле імені, активна область листа.

Робоча область електронної таблиці складається з рядків і стовпців, що мають свої імена. Імена рядків – це їх номери. Нумерація рядків починається з 1 і закінчується максимальним числом, встановленим для даної програми. Імена стовпців – це літери латинського алфавіту спершу від A до Z, потім від AA до AZ, BA до BZ і т.д.

Максимальна кількість рядків і стовпців визначається особливостями програми, наприклад, в табличному процесорі Calc 245 стовпців (від A до IV) і 65 536 рядків.

Перетин рядка і стовпця утворює елемент таблиці, що має свою унікальну адресу. Для вказування адрес комірок у формулах використовують ***посилання*** (наприклад, A6 або D8).

Комірка – область, що визначена перетином стовпця і рядка електронної таблиці, яка має свою унікальну адресу. ***Адреса комірки*** визначається ім'ям (номером) стовпця і ім'ям (номером) рядка, на перетині яких перебуває комірка, наприклад A10. Посилання – вказування адреси комірки.

Активна комірка - це виділена комірка, ім'я якої відображається в полі імені. Маркером виділення називається напівжирна рамка довкола виділеної комірки. Маркер заповнення - це чорний квадрат в правому нижньому кутку виділеної комірки.

Активна область листа - це область, яка містить введені дані.

У електронних таблицях можна працювати як з окремими комірками, так і з групами комірок, які утворюють блок. ***Блок комірок*** – група суміжних комірок. Адреса блоку комірок задається вказуванням посилань на першу та останню його комірки, між якими ставиться розділовий символ – двокрапка. Якщо блок має вигляд прямокутника, то його адреса задається адресами лівої верхньої і правої нижньої комірок, що входять в блок.

Блок комірок можна вказати двома шляхами: або задаванням з клавіатури початкової і кінцевої адрес комірок блоку, або виділенням відповідної частини таблиці за допомогою лівої клавіші миші.

Приклад задання адрес комірок і блоків:

- адреса комірки, що знаходиться на перетині стовпця F і рядка 9, виражається посиланням F9;
- адреса блоку, утвореного у вигляді частини рядка 1 - B1:E1;
- адреса блоку, утвореного у вигляді стовпця C - C1:C21;
- адреса блоку, утвореного у вигляді прямокутника - A3:G10.

2. Введення, редагування та форматування даних в таблиці.

Будь-яка обробка інформації починається з її введення в комп'ютер. У електронній таблиці Calc можна вводити текст, числа, дати, час, послідовні ряди даних і формули.

- Введення даних здійснюється в три етапи:
- виділення комірки;
- введення даних;
- підтвердження введення (натискування клавіші **Enter**).

Після того, як дані введені, їх потрібно представити на екрані в певному форматі. Для представлення даних в Calc існують різні категорії форматів.

Для редагування даних в комірці необхідно двічі клацнути на комірці і провести редагування або виправлення даних.

До ***операцій редагування*** відносяться:

- видалення і вставка рядків, стовпців, комірок і листів;
- копіювання і переміщення комірок і блоків комірок;
- редагування тексту і чисел в комірках.

До **о*перацій форматування*** відносяться:

- зміна числових форматів або форми представлення чисел;
- зміна ширини стовпців;
- вирівнювання тексту і чисел в комірках;
- зміна шрифту і кольору;
- вибір типу і кольору межі;
- заливка комірок.

Введення чисел і тексту. Будь-яку інформацію, яка обробляється на комп'ютері, можна представити у вигляді чисел або тексту. Числа і текст за замовчуванням Calc вводить у форматі Загальний.

Введення тексту. ***Текст*** - це будь-яка послідовність введених в комірку символів, яка не може бути інтерпретована Calc як число, формула, дата, час доби. Введений текст вирівнюється в комірці по лівому краю.

Щоб ввести текст, виділяють комірку і набирають текст з клавіатури. Комірка може вміщати до 255 символів. Якщо потрібно ввести деякі числа як текст, то для цього виділяють комірку, а потім вибирають команду **Format → Cells… (Формат → Комірки…)**. Далі вибирають вкладку **Числа** і в списку форматів, що з'явився, вибирають **Текстовий**. Ще один спосіб введення числа як тексту – це ввести перед числом символ апострофа.

Якщо текст не поміщається в комірку, то необхідно збільшити ширину стовпця або встановити перенос по словах **Format → Cells… → Alignment (Формат → Комірки… → Вирівнювання)**.

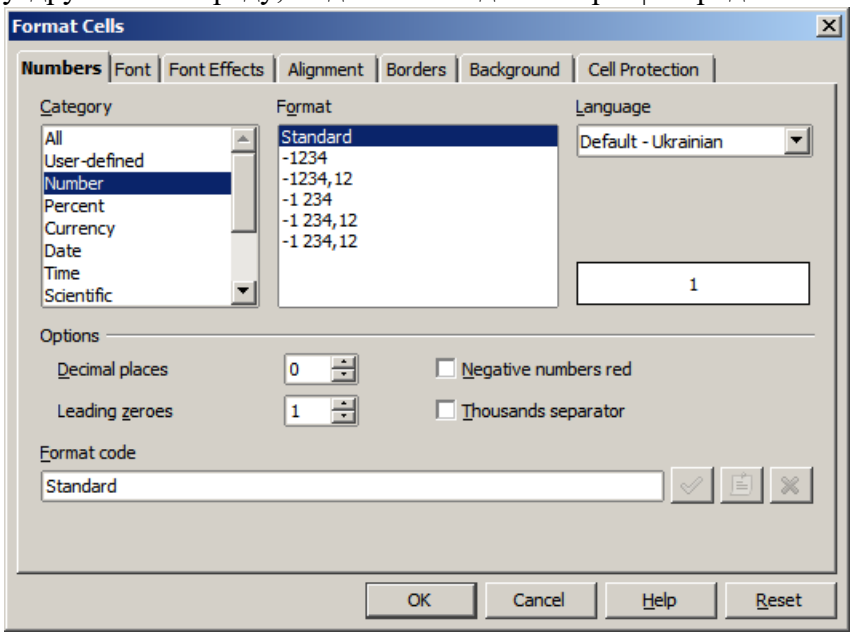
Введення чисел. ***Числові дані*** – це числові константи: **0-9**, **+**, **-**, **/**, *****, **E**, **%**, крапка і кома. При роботі з числами необхідно вміти змінювати вид чисел, що вводяться: число знаків після коми, вид цілої частини, порядок і знак числа. Calc самостійно визначає, чи відноситься введена інформація до числа. Якщо введені в комірку символи відносяться до тексту, то після підтвердження введення в комірку вони вирівнюються по лівому краю комірки, а якщо символи утворюють число – то по правому краю комірки.

Числа в Calc відображаються в форматах **Числовий**, **Експоненціальний**, **Фінансовий**, **Грошовий**, **Процентний**, **Дріб**.

Введення послідовних рядів даних. Під рядами даних розуміємо дані, що відрізняються один від одного на фіксований крок. При цьому дані не обов'язково мають бути числовими.

Для створення рядів даних необхідно виконати наступне. Ввести в комірку перший член ряду. Виділити область, де буде розташований ряд. Для цього потрібно підвести покажчик миші до маркера заповнення, і в момент, коли вказівник мишки набуде вигляду хрестика, натиснути ліву кнопку мишки. Далі, утримуючи натиснутою кнопку, треба виділити потрібну частину рядка або стовпця. Після того, як відпустити кнопку мишки, виділена область заповниться даними.

Можна побудувати ряд даних і іншим способом, якщо вказати крок побудови. Для цього потрібно ввести вручну другий член ряду, виділити обидві комірки і продовжити виділення до потрібної області. Дві перші комірки, введені вручну, задають крок ряду даних.



Формат даних. Дані в Calc виводяться на екран в певному форматі. За замовчуванням інформація виводиться у форматі **Загальний**. Можна змінити формат представлення інформації у виділених комірках. Для цього виконають команду **Format → Cells… (Формат → Комірки…)**.

З'явиться вікно діалогу, в якому потрібно вибрати вкладку **Числа**. У лівій частині вікна діалогу в списку категорій наведені назви всіх форматів, що використовуються в Calc.

Для формату кожної категорії наводиться повний його список. У нижній частині вікна можна переглянути всі коди форматів, які використовуються для зображення на екрані інформації. Для представлення даних можна використовувати вбудовані коди форматів Calc або ввести свій (визначений користувачем) код формату. Для введення коду формату вибирають рядок коду і вводять символи коду формату.

Розглянемо технологію створення електронної таблиці на прикладі проектування таблиці ***Облік товарів на складі***. Спочатку необхідно здійснити розмітку таблиці. Наприклад, таблиця **Облік товарів** має сім колонок, які закріпимо за стовпцями від A до G. Тепер треба сформувати заголовки таблиці. Спочатку потрібно ввести загальний заголовок таблиці, а потім назви полів. Вони повинні знаходитися в одному рядку і слідувати один за одним. Заголовок можна розташувати в один або два рядки, вирівняти по центру, правому, лівому, нижньому або верхньому краю комірки.

Для введення заголовка таблиці необхідно встановити курсор в комірку A2 і ввести назву таблиці «Залишки товарів на складі».

Виділити комірки A2:G2 і виконати команду **Format → Merge Cells (Формат → Об'єднання комірок)**.

Створення «шапки» таблиці. Ввести назви полів, наприклад № складу, Постачальник і так далі.

Для розташування тексту в комірках "шапки" в два рядки необхідно виділити цю комірку, виконати команду **Format → Cells… (Формат → Комірки…)** і на вкладці **Alignment (Вирівнювання)** встановити прапорець **Wrap text automatically (Переносити по словах)**.

Вставка різних шрифтів. Виділити текст і вибрати команду **Format → Cells… (Формат → Комірки…).** На вкладці **Font (Шрифт)** встановити гарнітуру шрифту, наприклад, Times New Roman, його розмір (кегль) і зображення.

Здійснити вирівнювання тексту в «шапці» таблиці (виділити текст і клацнути на кнопці центрування на панелі інструментів **Форматування**).

При необхідності змінити ширину стовпців за допомогою команди **Format → Column → Width (Формат → Стовпець → Ширина)**.

Змінити висоту рядка можна командою **Format → Row → Height (Формат → Рядок → Висота)**.

Додавання рамки і заливки комірок можна здійснити командою **Format → Cells… (Формат → Комірки…)** на вкладці **Borders (Межі)** і **Background (Фон)** відповідно.

3. Введення формул з використанням вбудованих функцій.

Формули – це вирази, що починаються із знаку рівності і складаються з числових величин, адрес комірок, функцій, імен, які з'єднані знаками арифметичних операцій. До

знаків арифметичних операцій, які використовуються в Calc, відносяться: *додавання; віднімання; множення; ділення; піднесення до степеня.*

Деякі операції у формулі мають вищий пріоритет і виконуються в такій послідовності:

- піднесення до степеня і вирази в дужках;
- множення і ділення;
- додавання і віднімання.

Результатом виконання формули є значення, яке виводиться в комірці, а сама формула відображається в рядку формул. Якщо значення в комірках, на які є посилання у формулах, змінюються, то результат зміниться автоматично.

Внесення змін до формул. Для внесенння змін до формули потрібно клацнути мишею на рядку формул або на клавішу F2. Потім внести зміни і натиснути кнопку Введення ☑ в рядку формул або на клавішу **Enter**. Якщо необхідно внести зміни до формули безпосередньо в комірці, де вона записана, то двічі клацнути мишею на комірці з цією формулою. Для відміни змін натиснути кнопку Відміна ☒ в рядку формул або клавішу **Esc**.

Використання посилань. Посилання однозначно визначає комірку або групу комірок робочого листа. За допомогою посилань можна використовувати у формулі дані, що знаходяться в різних місцях робочого листа, а також значення однієї й тої ж комірки в декількох формулах. Можна також посилатися на комірки, що знаходяться на інших листах документа, в іншому документі, або навіть на дані іншого додатку. Посилання на комірки інших документів називаються ***зовнішніми***. Посилання на дані в інших додатках називаються ***віддаленими***.

Переміщення і копіювання формул. ісля того, як формула введена в комірку, можна її перенести, скопіювати або розповсюдити на блок комірок. При переміщенні формули в нове місце таблиці посилання у формулі не змінюються, а комірка, де раніше була формула, стає вільною. При копіюванні формула переміщається в інше місце таблиці, при цьому абсолютні посилання не змінюються, а відносні посилання змінюються.

При копіюванні формул можна управляти зміною адрес комірок або посилань. Якщо перед всіма атрибутами адреси комірки поставити символ “$” (наприклад A1), то це буде **абсолютне** посилання, яке при копіюванні формули не зміниться. Зміняться тільки ті атрибути адреси комірок, перед якими не міститься символ “$”, тобто **відносні** посилання. Для швидкої установки символів “$” в посиланні його необхідно виділити у формулі і натиснути комбінацію клавіш **Shift+F4**.

Для ***переміщення*** формули підводять покажчик миші до межі комірки. Потім натискують ліву кнопку миші і, утримуючи її, переміщають комірку в потрібне місце таблиці. Завершивши переміщення, відпускають кнопку миші. Якщо в записі формули є адреси комірок, вони при переміщенні формули не змінюються.

Для ***копіювання*** формули підводять покажчик миші до межі комірки або блоку. Потім натискують клавішу **Ctrl** та ліву кнопку миші і переміщають комірку в потрібне місце таблиці. Для завершення копіювання відпускають кнопку миші і клавішу **Ctrl**. Якщо в записі формули є відносні адреси комірок, при копіюванні формули вони зміняться.

Розповсюдження (розмноження) формул. Крім копіювання і переміщення формулу можна розповсюдити на частину рядка або стовпця. При цьому відбувається зміна відносних посилань. Для розповсюдження формули необхідно виконати наступні дії:

- Встановити курсор в комірку з формулою.
- Підвести покажчик миші до маркера заповнення. Зображення покажчика змінюється на хрестик.
- Натиснути ліву кнопку миші і, утримуючи її, перемістити курсор до потрібного місця. Для завершення розповсюдження формули відпустити кнопку.

Необхідно відзначити, що Calc виводить в комірку значення помилки, якщо формула для цієї комірки не може бути правильно обчислена. Якщо формула містить посилання на комірку, що містить значення помилки, то ця формула також виводитиме значення помилки.

Функції Calc – спеціальні, заздалегідь створені формули для складних обчислень, в які користувач повинен ввести лише аргументи.

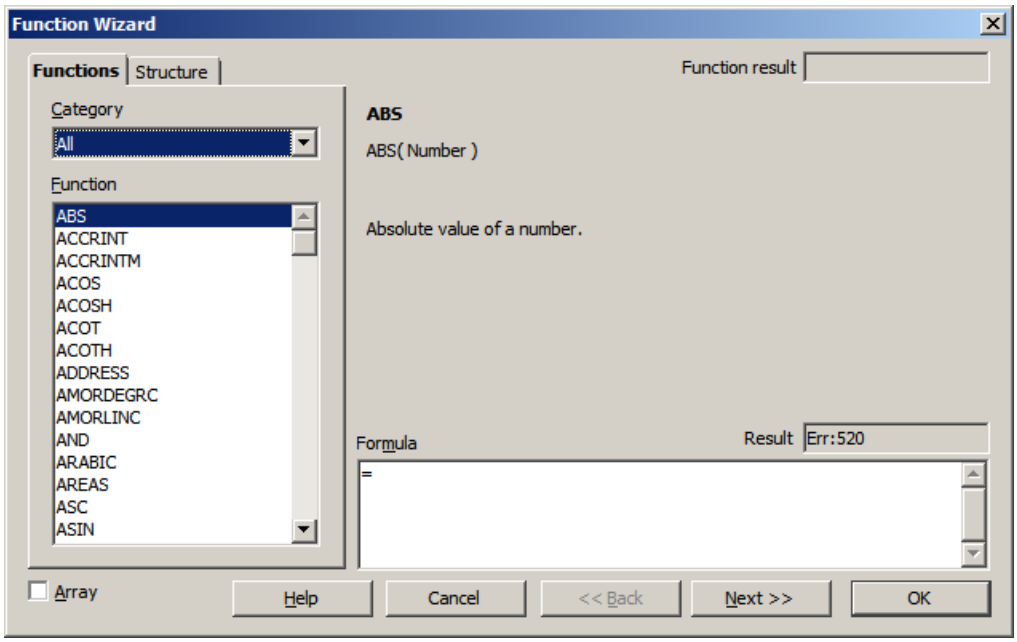
Функції складаються з двох частин: ***імені функції*** і ***одного або кількох аргументів***. Ім'я функції описує операцію, яку ця функція виконує, наприклад, **SUM**.

Аргументи функції Calc задають значення або комірки, що використовуються функцією, вони завжди поміщені в круглі дужки. Відкриваюча дужка ставиться без пропуску відразу після імені функції. Наприклад, у формулі **«=SUM(A2;A9)»**, SUM – це ім'я функції, а **A2** і **A9** – її аргументи. Ця формула підсумовує числа вкомірках **A2** і **A9**.

Навіть якщо функція не має аргументів, вона все одно повинна містити круглі дужки, наприклад функція **PI()**. При використанні у функції декількох аргументів вони відділяються один від одного крапкою з комою.

Введення функцій в робочому листі. Можна вводити функції в робочому листі просто з клавіатури або за допомогою команди **Insert → Function… (Вставка → Функція…)**. Якщо виділити комірку і вибрати цю команду, з'явиться вікно Майстра функцій.

Відкрити це вікно можна також за допомогою кнопки Вставка функції ***f(x)*** на рядку введення формул.



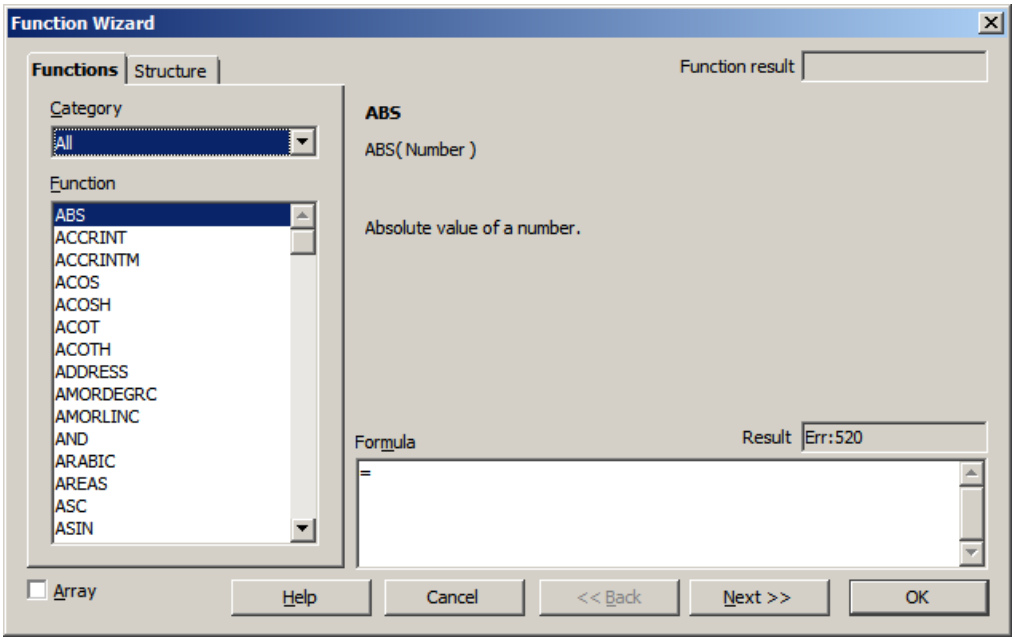
Лекція 4. Табличні процесори. Робота з формулами. Графічні об'єкти

1. *Категорії вбудованих функцій.*
2. *Математичні функції. Логічні функції.*
3. *Побудова діаграм. Використання графічних об'єктів.*

1. Категорії вбудованих функцій

Функції в електронних таблицях – спеціальні, заздалегідь створені формули для складних обчислень, в які користувач повинен ввести лише аргументи. Функції складаються з двох частин: ***імені функції*** і ***одного або кількох аргументів***. Ім'я функції описує операцію, яку ця функція виконує, наприклад, **SUM**.

Аргументи функції в ЕТ задають значення або комірки, що використовуються функцією, вони завжди поміщені в круглі дужки. Відкриваюча дужка ставиться без пропуску відразу після імені функції. Наприклад, у формулі **«=SUM(A2;A9)»**, SUM – це ім'я функції, а **A2** і **A9** – її аргументи. Ця формула підсумовує числа вкомірках **A2** і **A9**.



Навіть якщо функція не має аргументів, вона все одно повинна містити круглі дужки, наприклад функція **PI()**. При використанні у функції декількох аргументів вони відділяються один від одного крапкою з комою.

Введення функцій в робочому листі. Можна вводити функції в робочому листі просто з клавіатури або за допомогою команди **Insert → Function… (Вставка → Функція…)**. Якщо виділити комірку і вибрати цю команду, з'явиться вікно Майстра функцій. Відкрити це вікно можна також за допомогою кнопки Вставка функції ***f(x)*** на рядку введення формул.

За звичайних умов при введенні у комірку формули електронна таблиця має бути налаштована так, щоб сама формула вводилась і відображувалась тільки у рядку формул, а у комірці ж виникало значення, обчислене за цією формулою. Для цього виконують команду Сервіс-Параметри..., а далі у вкладниці Вид вікна Параметри знімають відмітку на пункті формули поля Параметри вікна, якщо вона там є.

При використанні рядка формул відомі наступні способи введення формул:

- безпосередній,
- за схемою автоматичного отримання адрес,
- за допомогою майстра функцій.

В процесі введення формули є доступними всі клавіши редагування для виправлення допущених помилок. Якщо надруковане не влаштовує повністю, то можна натиснути кнопку Відміна (червоний хрестик у рядку формул) або клавішу <Esc>. Якщо процес друкування закінчено, то слід натиснути кнопку Ввід (зелена позначка у рядку формул) або клавішу <Enter>.

При безпосередньому введенні формули всі її знаки послідовно набираються вручну. При введенні формули за схемою автоматичного отримання адрес клацаємо лівою кнопкою мишки на відповідній комірці, після чого її адреса автоматично потрапляє до складу формули. Якщо введена адреса має бути абсолютною, то починаємо послідовно натискувати клавішу, спостерігаємо за варіантами абсолютних адрес і зупиняємось на потрібному.

При введення формули за допомогою Майстра функцій роботу починаємо з натискування кнопки Вставка функції рядка формул. Внаслідок цього отримуємо в рядку формул знак “=” і текстовий курсор, а також діалогове вікно Майстер функций – крок 1 з 2. Указане вікно містить повний перелік функцій, із яких слід вибрати потрібну. Тепер виникає діалогове вікно Аргументи функції, в якому поле Число призначене для вказівки аргументів вибраної функції.

Відзначимо, що діалогове вікно Майстер функцій – крок 1 з 2 містить функції, перелік яких розбито на категорії. Перелік категорій міститься у підвікні Категорія:. Перелік функцій, що входять до складу відміченої категорії, представлений у другому підвікні Виберіть функцію:. Відмітивши потрібну функцію у цьому підвікні, нижче у вікні можна побачити стислий опис цієї функціїї. Для отримання детальної довідки (повний опис, математичне обгрунтування, приклад застосування) треба натиснути пункт Довідка для цієї функції в нижній частині вікна. Внаслідок цього буде отримано область задач довідка Excelз детальною інформацією.

Виконуючи введення аргументів функцій за допомогою діалогового вікна Аргументи функції, слід звернути увагу на кнопку з червоною стрілкою в кінці поля Число для введення аргументів. Ця кнопка дозволяє тимчасово згорнути діалогове вікно до вузенької смужки, щоб воно не заважало вибрати і відмітити аргумент в області робочого листа. Після вибору аргументу смужку можна знову розгорнути до нормального розміру за допомогою подібної ж кнопки на ній.

Як і будь-які інші дані, формули в комірках ЕТ можна копіювати. При цьому вони автоматично настроюються на нове місце свого розташування. В багатьох випадках копіювання формули можна зробити шляхом автозаповнення. Для цього треба виділити комірку з формулою і сумістити вказівник мишки з правим нижнім кутком селектора. При цьому вказівник мишки має набути форми маленького чорного хрестика. Утримуючи ліву кнопку мишки у натиснутому стані, протягуємо вказівник мишки у потрібному напрямі на потрібну кількість комірок. Після цього кнопку мишки відпускаємо.

2. Математичні функції. Логічні функції.

До складу MS Excel входить надзвичайно велика кількість різноманітних **стандартних функцій**. Для ознайомлення з ними можна скористуватись спеціальним засобом, який називається **Мастер функций**. Основні стандартні елементарні математичні функції представлено в таблиці.

Математичний запис	Запис в Excel
Sin x	SIN(x)
Cos x	COS(x)
Tg x	TAN(x)
Arcsin x	ASIN(x)
Arccos x	ACOS(x)
Arctg x	ATAN(x)
Ln x	LN(x)
Lg x	LOG10(x)
$Log_a x$	LOG(x, a)
e^x	EXP(x)
$\|x\|$	ABS(x)
	КОРЕНЬ(x)
	ПИ()
	x^3
	x^(1/3)
Заокруглення числа x до найближчого меншого цілого	ЦЕЛОЕ(x)
Обчислення остачі від ділення націло числа N на число D	ОСТАТ(N, D)
Заокруглення числа x до заданого числа розрядів k	ОКРУГЛ(x, k)

В наведених тригонометричних функціях використовується радіан як одиниця вимірювання величин кутів.

Функції ЦЕЛОЕ(x) і ОСТАТ(N,D) утворюють загальне правило обчислення частки і остачі від ділення націло числа N на число D: N=D***ЦЕЛОЕ**(N/D)+**ОСТАТ**(N,D).

Функція ОКРУГЛ(x, k) виконує заокруглення за звичайним арифметичним правилом заокруглення (якщо цифра, яка відкидається, менша 5, то попередня цифра залишається без змін, у протилежному випадку попередня цифра збільшується на одиницю). Якщо k – додатне, то число x заокруглюється до указаної кількості десяткових розрядів справа від десяткової крапки. Якщо k=0, то число x заокруглюється до найближчого цілого. Якщо k – від'ємне, то число x заокруглюється до указаної кількості десяткових розрядів зліва від десяткової крапки.

Досить часто у формулах використовуються **спеціальні функції**, призначені для розв'язування типових економічних, статистичних, планових та інших задач. Щоб всі ці функції стали доступними, необхідно установити надбудову, яка називається **Пакет аналізу**. Для цього використовується команда **Сервіс-Надбудови...**.

Приклади використання деяких функцій:

Формула «=**ЦЕЛОЕ**(5.7)» дає результат 5, а формула «=**ЦЕЛОЕ**(-5.7)» дає результат 6.

Формула «=**ЦЕЛОЕ**(23/3)» дає частку від ділення націло числа 23 на число 3, тобто 7.

Формула «=**ОСТАТ**(23,3)» дає результат 2.

Якщо x=143.3184, то формула «=**ОКРУГЛ**(x,2)» дає результат 143.32.

Якщо x=143.3184, то формула «=**ОКРУГЛ**(x,0)» дає результат 143.

Якщо x=143.3184, то формула «=**ОКРУГЛ**(x,-1)» дає результат 140.

Крім арифметичних виразів, важливим компонентом формул є логічні вирази, зокрема, **логічні функції**. Логічний вираз – це є спільна назва для висловлювання та предиката. Висловлюванням називається твердження, відносно якого відразу можна зробити висновок, вірне воно чи ні. Наприклад, значенням висловлювання «7>5» буде ІСТИНА. Значенням висловлювання «3>5» буде ХИБНІСТЬ. Висловлювання, яке містить змінні величина, називається предикатом. В залежності від значень змінних предикат може набувати значення ІСТИНА або ЛОЖЬ. Наприклад, результатом порівняння «x>3» буде ХИБНІСТЬ при x=2 і ІСТИНА – при x=6. Предикат утворюється внаслідок порівняння двох арифметичних виразів, з яких хоча б один містить змінні.

У логічних виразах можуть використовуватись такі операції порівняння: «>» – більше, «>=» – більше або дорівнює, «<» – менше, «<=» – менше або дорівнює, «=» – дорівнює, «<>» – не дорівнює. Треба пам'ятати, що операції порівняння мають нижчий пріоритет, ніж арифметичні операції.

У логічних виразах можуть використовуватись також логічні операції, реалізовані у вигляді логічних функцій: НЕ(x) – заперечення, И(x,y) – логічне множення, ИЛИ(x,y) – логічне додавання.

В арифметичних виразах логічне значення ИСТИНА поводить себе як число 1, а ЛОЖЬ – як число 0. І навпаки – в логічних виразах число 1 поводить себе як ИСТИНА, а

число 0 – як ХИБНІСТЬ. Більше того, замість ІСТИНА можна указувати будь-яке число, відмінне від нуля.

Логічні вирази найчастіше застосовуються як перший аргумент логічної функції ЕСЛИ:

ЕСЛИ(лог_выражение, значение_если_истина, значение_если_ложь).

Ця функція має три аргументи, зміст яких такий: якщо **лог_выражение** дорівнює ІСТИНА, то значення функції обчислюється як значення другого аргументу **значение_если_истина**, а якщо **лог_выражение** дорівнює ЛОЖЬ, то значення функції обчислюється як значення третього аргументу **значение_если_ложь**. Особливість функції **ЕСЛИ** полягає в тому, що її тип наперед не визначений і співпадає з типом або другого, або третього свого аргумента.

Приклад 1. Задано число z. Побудувати формулу, яка дає результат z+1, якщо z>1, і дає результат z-1 у протилежному випадку.

Розв'язок: «=**ЕСЛИ**(z>1,z+1,z-1)».

Приклад 2. Задано число z. Побудувати формулу, результатом якої є повідомлення «Перевищено порогове значення», якщо z>100, і яка дає результат z у протилежному випадку.

Розв'язок: «=**ЕСЛИ**(z>100, "Перевищено порогове значення",z)».

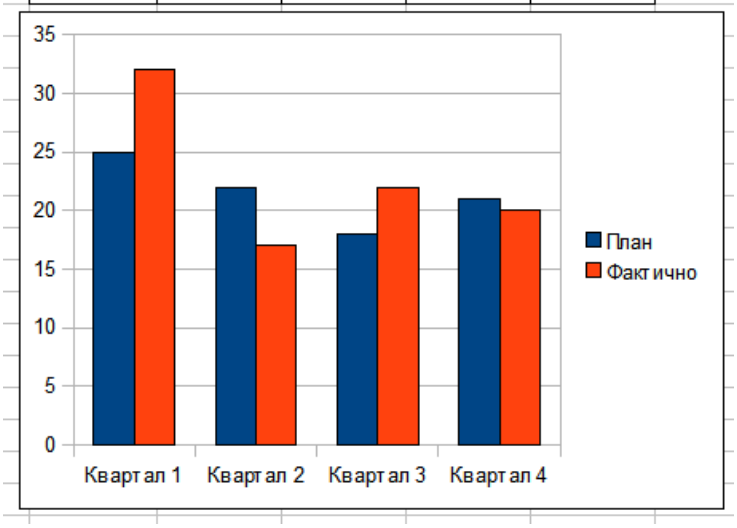
Приклад 3. Задано число z. Побудувати формулу, яка дає результат z/2, якщо , дає результат 10, якщо z<10, і дає результат 25, якщо z>25.

Розв'язок: «=**ЕСЛИ**(z<10,10,**ЕСЛИ**(z>25,25,z/2))».

3. Побудова діаграм. Використання графічних об'єктів.

Для даних робочого листа можна створювати складні діаграми. **Діаграма** – графічне зображення залежності між величинами. Діаграми є наочним засобом представлення даних робочого листа. Діаграму можна створити на окремому листі або помістити як впроваджений об'єкт на лист з даними.

	Квартал 1	Квартал 2	Квартал 3	Квартал 4
План	25	22	18	21
Фактично	32	17	22	20



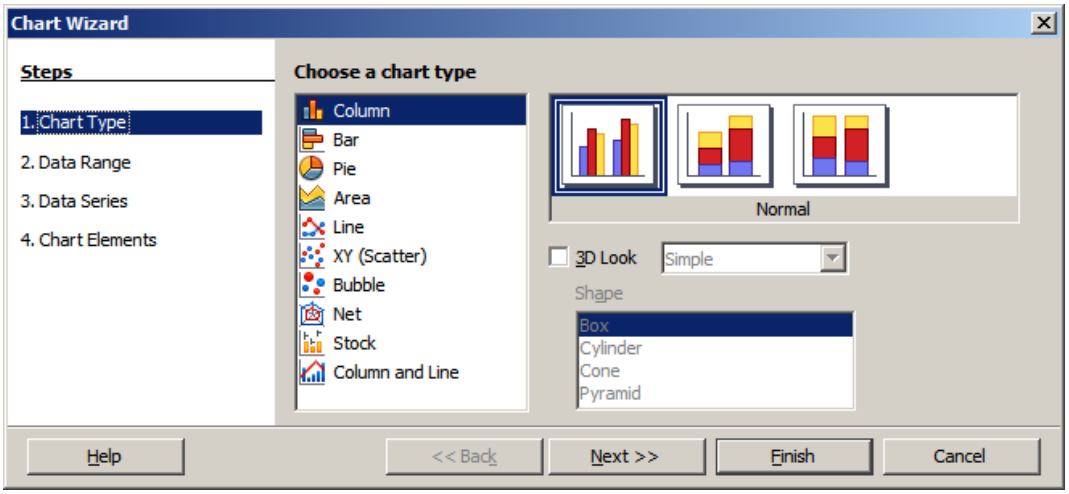
Діаграма пов'язана з даними, на основі яких вона створена, і оновлюється автоматично при зміні даних.

Щоб створити діаграму, необхідно, передусім ввести дані для діаграми на лист. Потім виділити будь-які комірки, які

містять вихідні дані діаграми. Вибрати команду **Insert → Chart… (Вставка → Діаграма)** або натиснути кнопку Діаграма  на стандартній панелі інструментів. На екран виведеться перше вікно майстра діаграм.

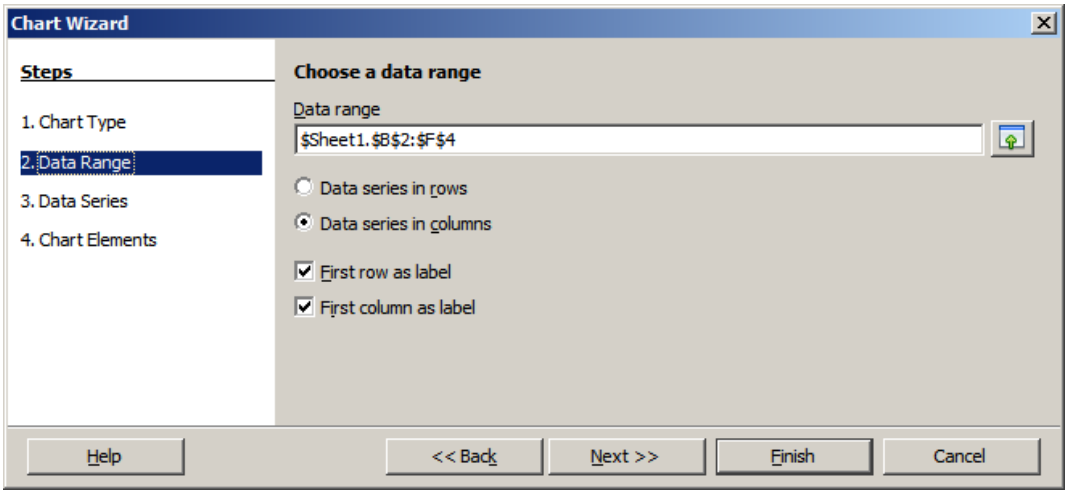
Крок 1. Вибір типу діаграми.

Перше вікно діалогу пропонує вибрати тип діаграми.



Крок 2. Задання вихідних даних діаграми.

У другому вікні майстра діаграм можна задати дані, які будуть використовуватись при побудові діаграми.

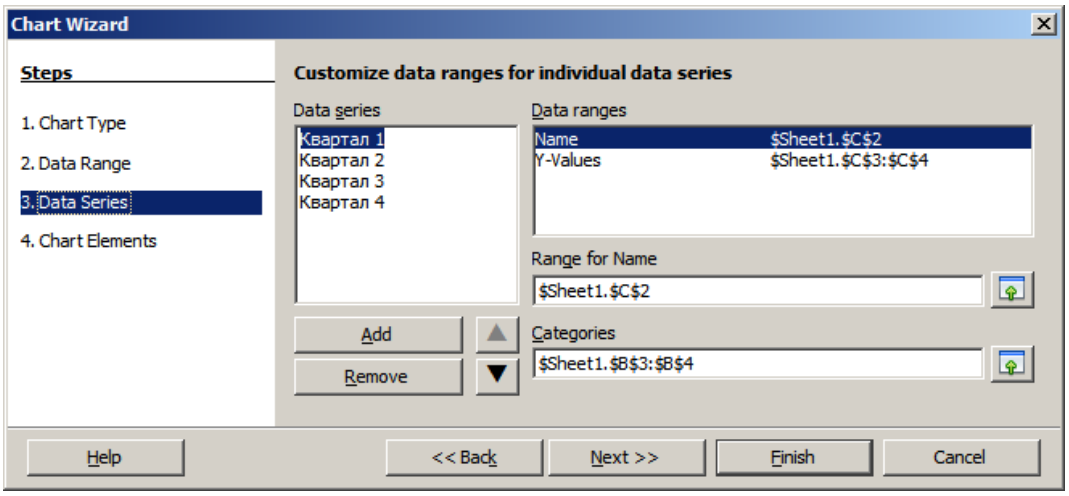


Друге вікно діалогу Майстра діаграм дозволяє задати вихідний діапазон даних. Якщо перед запуском Майстра був виділений діапазон з вихідними даними, то це поле міститиме посилання на виділений діапазон.

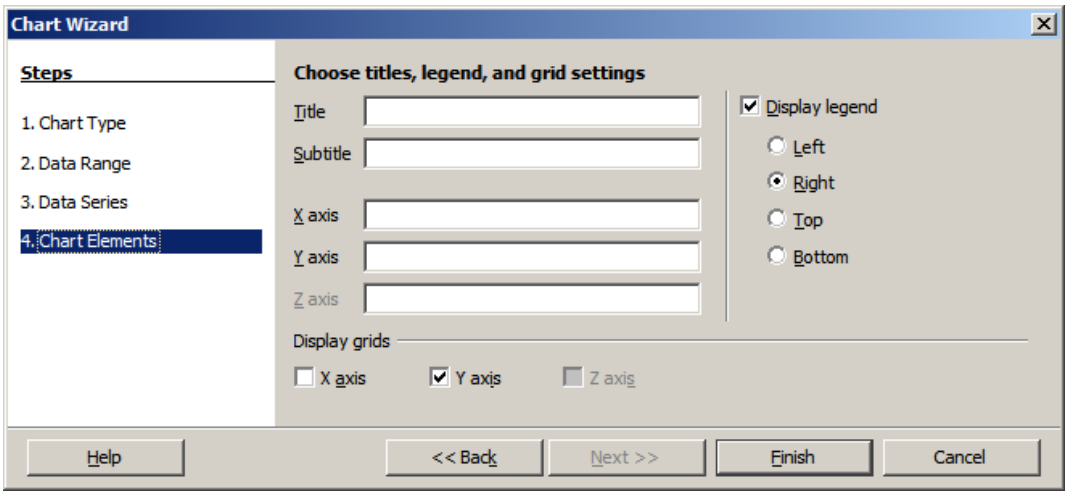
Якщо з певних причин вихідний діапазон вказаний неправильно, необхідно виділити потрібний діапазон і ввести його у вікні діалогу Майстра діаграм.

Calc зазвичай вибирає орієнтацію рядів, припускаючи, що діаграма повинна містити менше рядів, ніж точок. Переглядаючи зразок при різній орієнтації рядів, можна вибрати найбільш ефективний спосіб відображення даних в діаграмі.

Крок 3. Задання рядів даних.



Крок 4. Елементи діаграми. У цьому вікні можна задати назви елементів діаграми, місце розміщення легенди та ін.



Після побудови діаграми її можна відредагувати в режимі редагування діаграми. Для цього потрібно двічі клацнути кнопку миші на діаграмі або скористатися контекстним меню.

Лекція 5. Табличні процесори. Опрацювання списків.

1. *Обробка та аналіз табличних даних засобами ЕТ.*
2. *Впорядкування і фільтрування даних.*
3. *Підсумки, зведені таблиці.*
4. *Консолідація даних.*

1. Обробка та аналіз табличних даних засобами ЕТ.

Список є електронною таблицею з великим об'ємом взаємозв'язаної інформації (наприклад, список товарів на складах, список номерів телефонів і адрес абонентів). ***Список*** – це набір рядків електронної таблиці зі взаємозв'язаними однотипними даними постійного формату. Іншими словами список – це плоска база даних, а рядки і стовпці списку відповідають записам і полям в базі даних.

До списків в Calc пред'являються суворіші вимоги, ніж до звичайних електронних таблиць. Кількість стовпців в списку має бути постійною, а кількість рядків змінною. Це дозволяє додавати, видаляти або переставляти рядки таблиці або запису списку (бази даних).

Наявність порожніх рядків і стовпців в списку є неприпустимою. Дані в списку повинні мати постійний формат. Перший рядок в списку містить назви стовпців або імена полів як в базах даних.

До засобів, які призначені для обробки і аналізу даних в списку відносяться команди з меню **Data (Дані)**: **Sort… (Сортування…)**, **Filter (Фільтр)**, **Subtotals… (Підсумки…)**, **Validity… (Перевірка…)**. При виконанні цих команд редактор автоматично розпізнає список як базу даних і здійснює обробку і аналіз даних в списку як в базі даних.

При застосуванні команди сортування можна відсортувати записи за одним чи кількома полями. За допомогою фільтрів (Автофільтр, Стандартний фільтр і Розширений фільтр) можна швидко знайти (відфільтрувати) необхідні дані в списках за одним або кількома параметрами пошуку. Командою Підсумки можна впорядкувати дані в списках за допомогою підсумкових значень.

Для перевірки даних при введенні використовується засіб, який називається перевіркою введення (команда Перевірка).

При створенні списку необхідно виконати певні вимоги:

- Щоб Calc автоматично розпізнавав список як базу даних і обробляв дані при виконанні команд обробки, необхідно на робочому листі розташовувати один список.
- Формат шрифту заголовків (підписів) стовпців або імен полів в списках повинен відрізнятися від формату шрифту записів. Зазвичай шрифту заголовкам стовпців призначається напівжирний шрифт, а коміркам для заголовків задається текстовий формат.

- Комірки під заголовками стовпців необхідно відформатувати відповідно до даних, які вводитимуться в ці комірки (встановити формат, вибрати вирівнювання і т. д.).
- У списку не повинно бути порожніх записів (рядків) і полів (стовпців), навіть для відділення імен полів від записів слід використовувати межі комірок, а не порожні рядки.

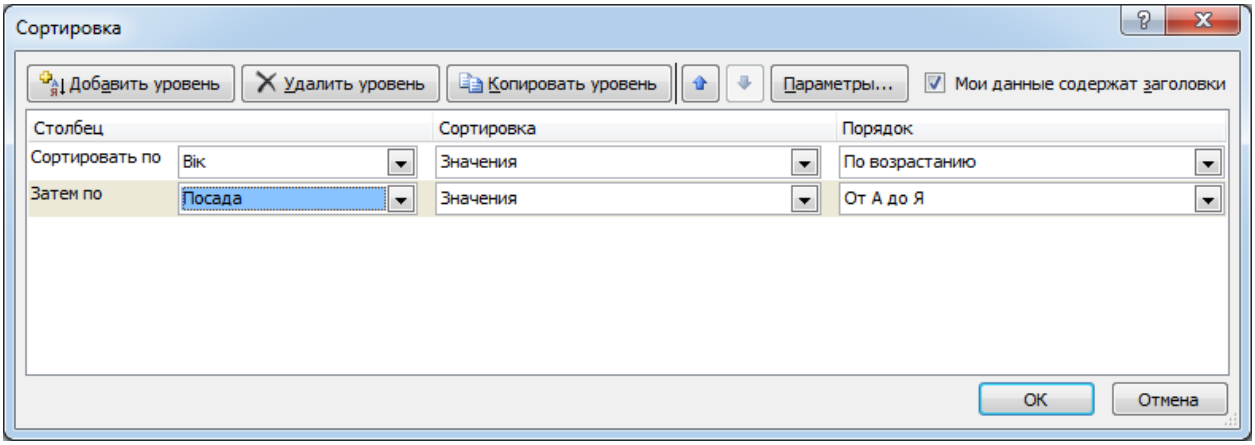
2. Впорядкування і фільтрування даних.

Для відбору даних за певними критеріями щодо величин в окремих полях в ЕТ використовуються інструменти, що називають фільтрами. Їх дія подібна до виконання запитів на вибірку даних з БД. Деколи для швидкої оцінки найбільших - найменших значень, чи діапазону в межах якого знаходяться дані достатньо впорядкувати (посортувати) записи списку. Сортування в ЕТ виконується у двох «напрямках»: від меншого значення до більшого (зростання) і від більшого до меншого (спадання). Впорядковуються не тільки числові дані, а також і дати у хронологічному порядку та текст у алфавітному.

Сортування може також використовуватися як перший етап інших операцій з обробки списку, наприклад, як при виконання проміжних підсумків в табличному процесорі MS Excel.

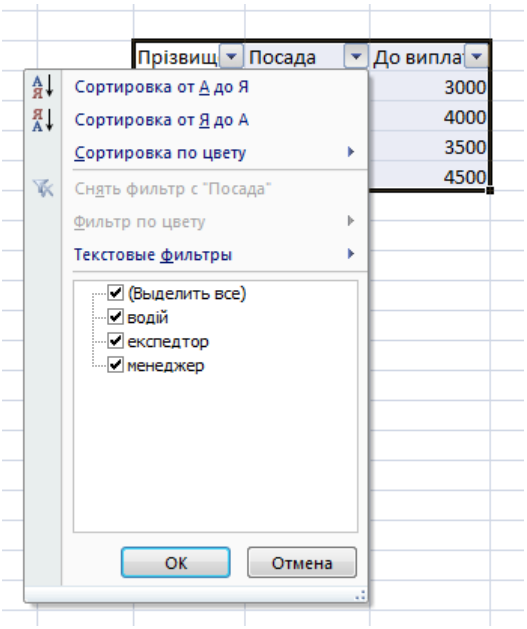
Сортування може здійснюватися за декількома полями списку одночасно (див. малюнок), при цьому має значення послідовність задання критеріїв: спочатку дані впорядковуються за значеннями у першому вказаному полі, і лише якщо дані у двох записів у пешому полі рівні, такі записи порівнюються за даними у другому вказаному полі, якщо і ці значення рівні, то за значеннями третього поля, і т.д.

На малюнку нижче наведено приклад діалогового вікна сортування табличного процессора Microsoft Excel 2007.



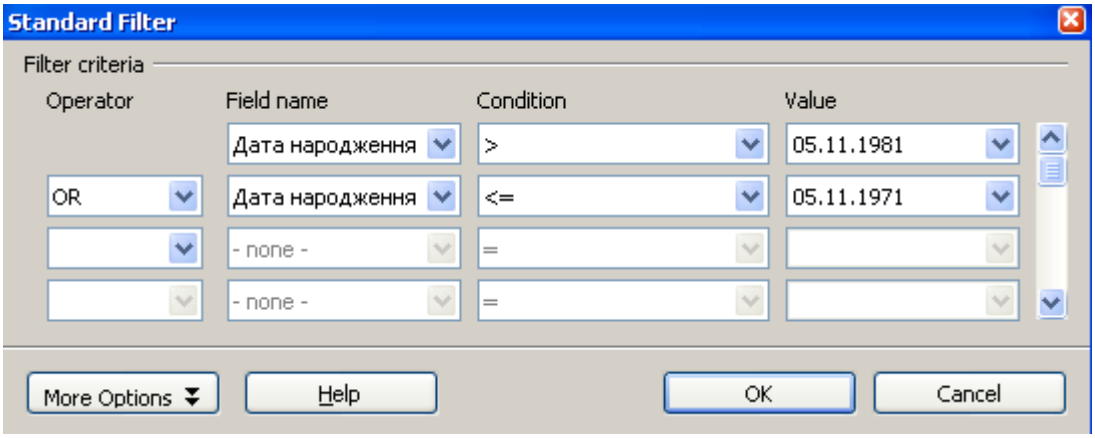
Для відбору записів, що відповідають деякому критерієві табличні процесори надають користувачеві набір засобів, що носять назву фільтри. В більшості електронних таблиць реалізовано три принципові різні інструменти фільтрування. Залежно від версії та локалізації офісних пакетів, вони можуть мати різні назви:

1. ***Автофільтр*** **(Фільтр, Звичайний фільтр, тощо) –** інструмент який вмикає відкривні списки біля заголовків таблиці. Ці списки містять набори різних значень стовпця і дозволяють відібрати записи за точним збігом значень



2. ***Стандартний фільтр*** **(Фільтр за умовою, Користувацький фільтр).** – часто реалізовується як наступний крок після автофільтра. Дозволяє додатково до можливостей відбору записів за конкретним значенням у полі вводити критерії у вигляді діапазонів значень, поєднання декількох умов логічними операторами «і» чи «або». На малюнку нижче показано для прикладу діалогове вікно стандартного фільтра табличного процесора Calc з офісного пакету Libre Office.

Обидва згадані вище фільтри щодо результату вибірки виконують «фільтрування на місці» шляхом приховування зі списку рядків із записами, що не відповідають критеріям. Для збереження результатів фільтрів цього типу слід їх копіювати в інший діапазон таблиці.

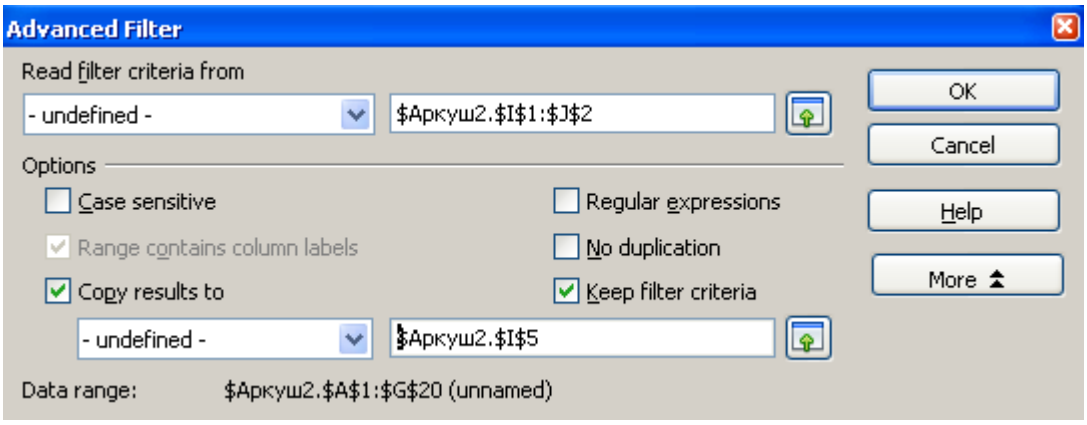


3. Розширений фільтр. На відміну від перших двох інструментів має опцію для копіювання результатів вибірки у вказаний діапазон таблиці. Ще одна принципова відмінність – задання критеріїв для фільтрації на робочому листі ЕТ. Розширений фільтр передбачає опис умов на відбір записів у вигляді спеціального діапазону комірок із двох рядків, де у верхньому рядку вказується назва поля списку, а в нижньому критерій (див. малюнок).

I	J	K	L	M	N	O
Ціна	***Ціна***					
>300	<500					
№	***Фірма***	***Код товар***	***Ціна***	***Дата про***	***Кількість***	***Вартість***
13	Кварц	Принтер SA	320	06.04.11	8	2560
14	Альфа	Принтер SA	335	06.04.11	9	3015
15	Партнер	Принтер HP	310	07.04.11	11	3410

При цьому критерії можуть стосуватися різних полів списку, поєднуються умови критерії логічною зв'язкою «і».

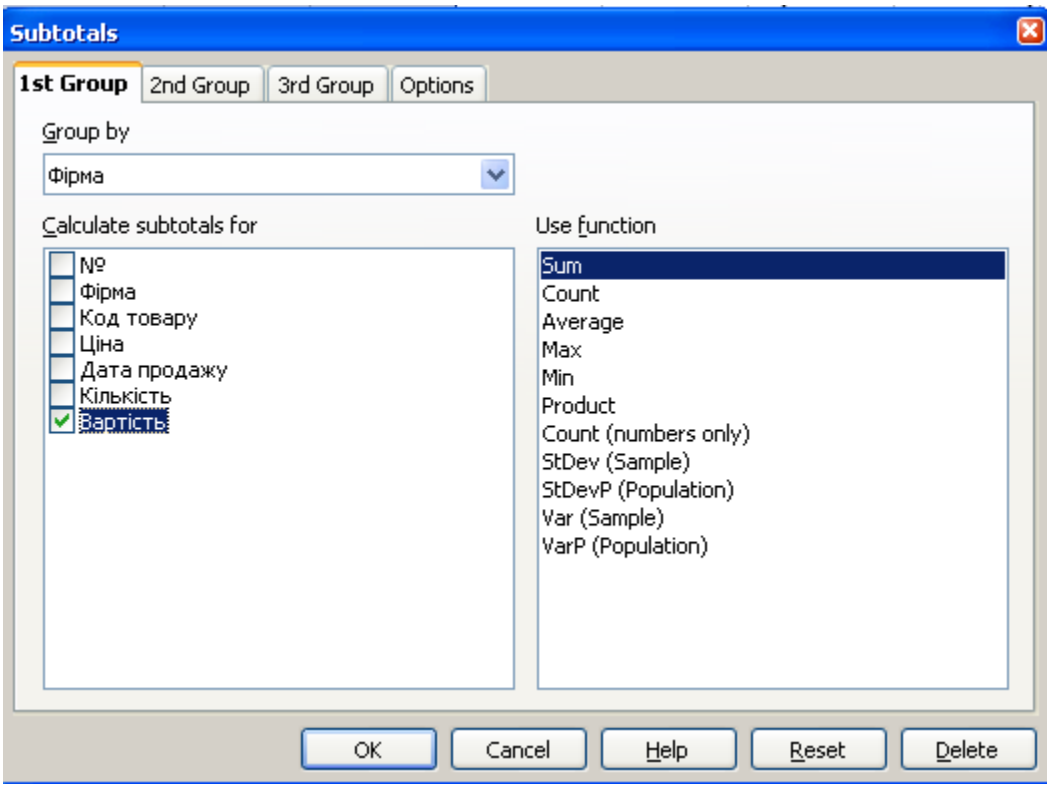
Діапазон критеріїв разом із діапазоном списку є обов'язковими параметрами розширеного фільтра, що вказуються у відповідному діалоговому вікні:



3. Підсумки, зведені таблиці.

Проміжні підсумки належать до інструментів «групових обчислень», тобто дозволяють виконувати підсумкові (агрегуючі) операції над даними поділеним на групи за деякою ознакою. До агрегуючих операцій належать ***сума***, ***кількість***, ***середнє значення***, ***мінімум***, ***максимум та добуток***. Для того, щоб записи груп, для яких обчислюються підсумки були розташовані поряд, в деяких табличних процесорах (наприклад, Microsoft Excel) перед тим, як підводити **підсумки**, таблицю потрібно відсортувати таблицю за відповідним полем.

Для підведення підсумків в меню **Дані** вибрати команду



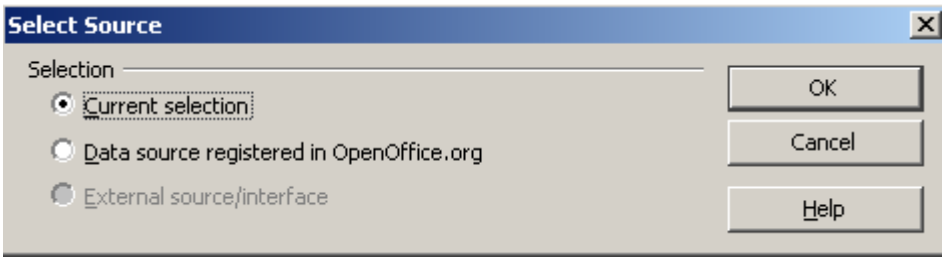
Підсумки…(Subtotals…). З'явиться діалогове вікно, в якому потрібно заповнити поля і встановити потрібні опції (див малюнок). Натиснути ОК.

Зведена таблиця - це інтерактивна таблиця, яку можна використовувати для аналізу баз даних. Зведені таблиці дозволяють виконувати «групові обчислення» подібно до проміжних підсумків, виконуючи над даними агрегатні операції. Вони мають специфічний макет, розділений на області рядків і стовпців, та даних, що дозволяє виводити не тільки дані за групами записів окремого поля, але і перехресні дані за двома полями у вигляді прямокутних таблиць.

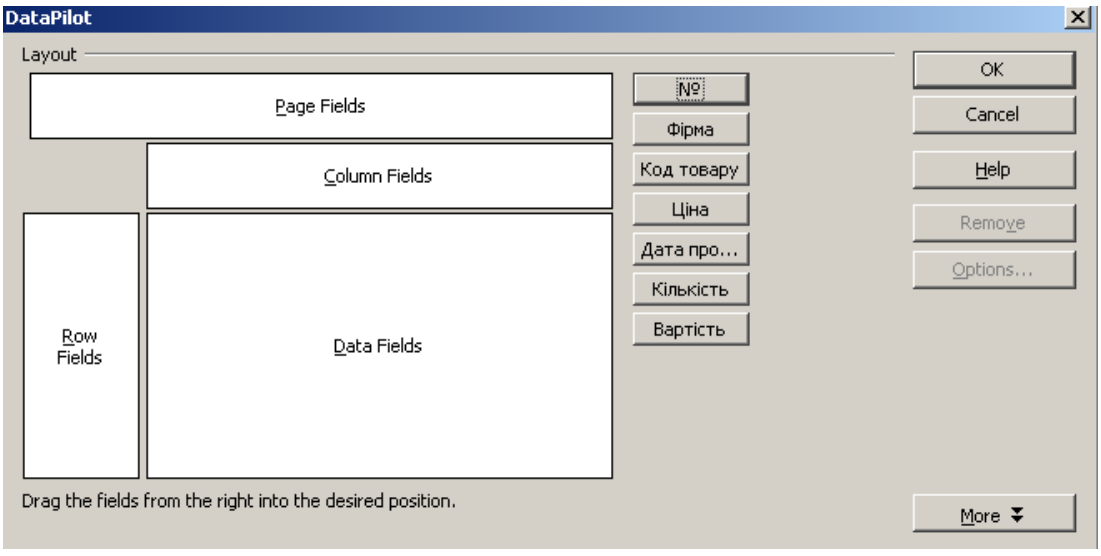
Після зміни даних в базі, дані у побудованій на її основі зведеній таблиці можна оновити.

Подані нижче малюнки ілюструють особливості макету та процесу створення зведених таблиць.

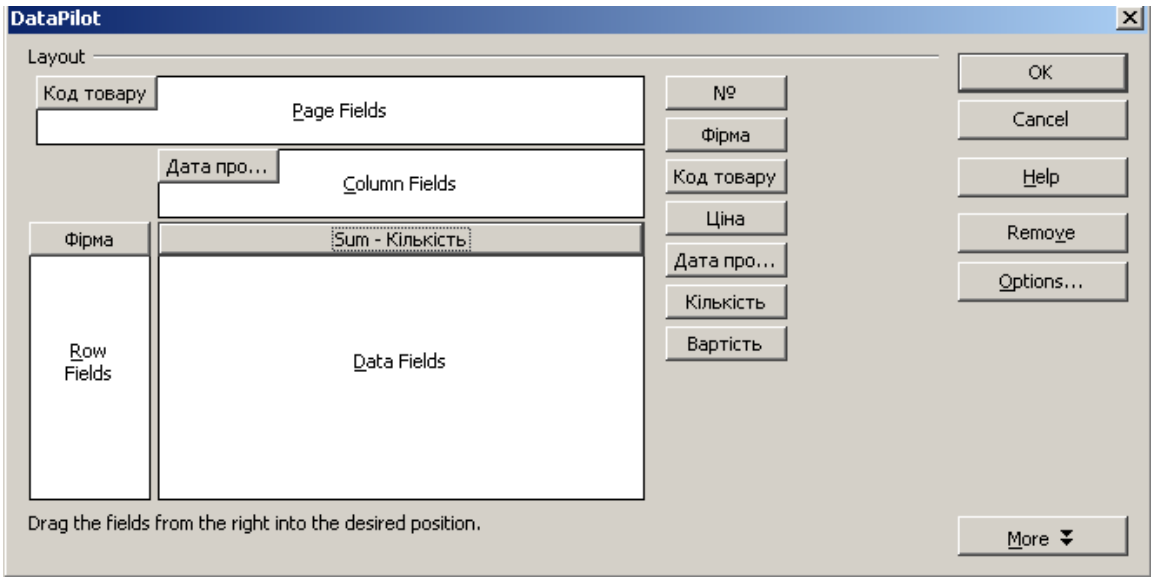
Щоб створити зведену таблицю, в меню **Дані (Data)** слід вибрати **Зведена таблиця (DataPilot)**, **Старт… (Start…)** З'явиться вікно діалогу, у якому уточняється діапазон списку, на основі якого створюється зведена таблиця:



Після натискання ОК відкриється вікно з макетом зведеної таблиці:



Достатньо перетягнути мишкою кнопки з назвами полів таблиці в поля шаблону зведеної таблиці, наприклад, наступним чином:



Після натискання ОК отримаємо таку зведену таблицю.

21							
22	Filter						
23	Код товару	- all -					
24							
25	Sum - Кількість	Дата продажу					
26	Фірма	03.04.11	04.04.11	05.04.11	06.04.11	07.04.11	**Total Result**
27	Альфа	17		45	43		**105**
28	Всесвіт	8			12		**20**
29	Кварц		23	46	8	12	**89**
30	ОРТЕКС		12			28	**40**
31	Партнер	9				34	**43**
32	**Total Result**	**34**	**35**	**91**	**63**	**74**	**297**
33							

Верхня частина зведеної таблиці - ***область сторінок*** дозволяє переглянути варіанти зведеної таблиці для кожного окремого значення поля *Код товару*, у нашому випадку.

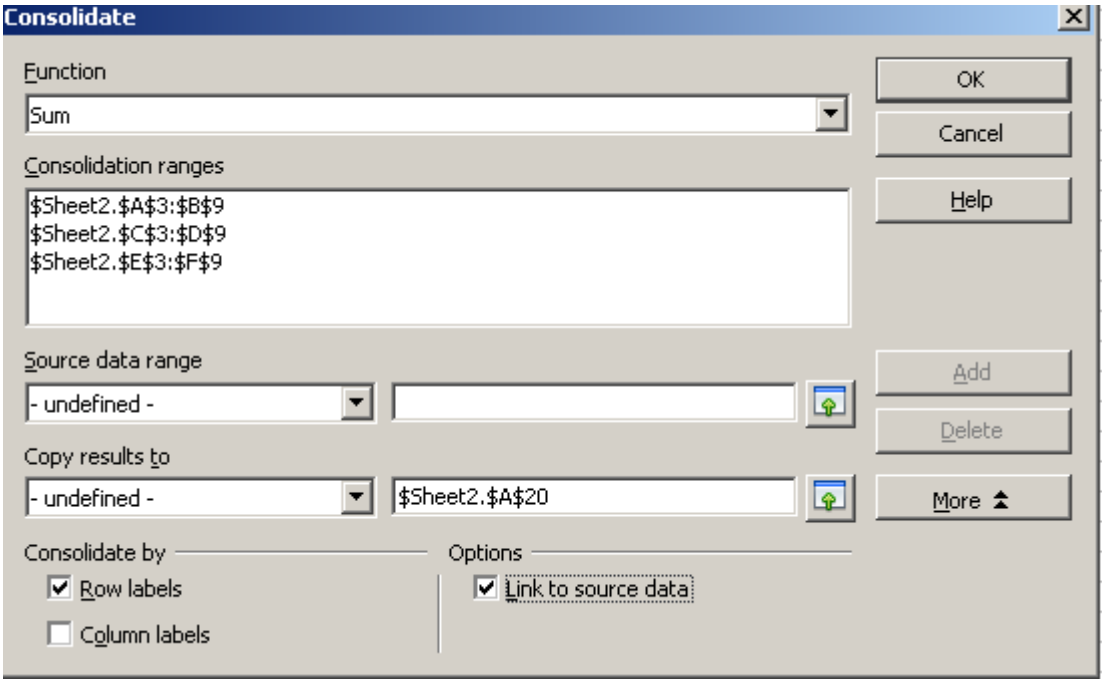
4. Консолідація даних.

Консолідація, або об’єнання даних дозволяє «акумулювати» дані із різних версій таблиці однієї і тієї ж структури. Наприклад:

	A	B	C	D	E	F
1	***Витрати***					
2	***Січень***		***Лютий***		***Березень***	
3	**Оренда**	3500	**Оренда**	3670	**Оренда**	4200
4	**Страховка**	1230	**Страховка**	1500	**Страховка**	1700
5	**Послуги**	450	**Послуги**	520	**Послуги**	590
6	**Соціальні**	4000	**Соціальні**	3700	**Соціальні**	4200
7	**Виплата**	13000	**Виплата**	13000	**Виплата**	10000
8	**Різне**	400	**Різне**	480	**Різне**	700
9	**Всього**	22580	**Всього**	22870	**Всього**	21390

Аналогічно до «групових обчислень» консолідація передбачає об’єднання даних за допомогою однієї з агрегатних операцій, наприклад суми, чи добутку. «Особливо творчі» користувачі електронних таблиць запропонували навіть використовувати інструмент консолідації для множення діапазонів комірок.

Щоб скористатися інструментом, потрібно встановити курсор в лівий верхній кут результуючої об'єднаної таблиці (тобто міця, де вона має бути розташована після завершення операції). В меню **Дані (Data**) вибрати команду **Консолідація… (Consolidate…)**. З'явиться вікно діалогу:



В полі *Function* вибрати потрібну групову операцію, наприклад *Sum.*

Встановити курсор в полі *Source data range*, виділити мишкою потрібний діапазон (у прикладі вище A3:B9) та натиснути Add. Аналогічно додати усі інші діапазони, що мають бути об'єднані (в нашому прикладі ще C3:D9 та E3:F9). Натиснути OK.

Опція «зберігати зв'язок із вихідними даними» (Link to source data) дозволяє створити таку об'єднану таблицю, що при зміні даних автоматично змінюється результуюча таблиця.

Лекція 6. Системи керування базами даних. Бази даних та їх архітектура.

1. *Бази даних.*
2. *Типи архітектур БД.*
3. *Об'єкти реляційних БД та їх призначення.*
4. *Побудова БД.*

1. Поняття про бази та сховища даних.

Банк даних (БнД) є сучасною формою організації зберігання і доступу до інформації. Існує множина визначень банку даних. За найбільш поширеною думкою, банк даних - це система спеціальним чином організованих даних (баз даних), програмних, технічних, мовних, організаційно-методичних засобів, призначених для забезпечення централізованого накопичення і колективного багатоцільового використання даних.

Послугами банку даних користується велика кількість користувачів. Тому в БнД передбачається спеціальний засіб приведення всіх запитів до єдиної термінології - словник даних. Крім того, використовуються спеціальні методи еквівалентних граматичних перетворень запитів для побудови оптимальних процедур їх обробки. Зазвичай з боку зовнішніх користувачів до банку даних формулюються вимоги. Банк даних повинен:

задовольняти актуальним інформаційним потребам зовнішніх користувачів, забезпечувати можливість зберігання і модифікації великих обсягів багатоаспектної інформації;

- забезпечувати заданий рівень достовірності збереженої інформації та її несуперечність;
- забезпечувати доступ до даних тільки користувачам з відповідними повноваженнями;
- забезпечувати можливість пошуку інформації за довільною групою ознак;
- задовольняти заданим вимогам за продуктивності при обробці запитів;
- мати можливість реорганізації та розширення при зміні ПЗ;
- забезпечувати простоту і зручність звернення зовнішніх користувачів за інформацією;

Структура банка данных

Банк даних включає до свого складу дві основні компоненти: базу даних, яка є не що інше, як даталогічне подання інформаційної моделі предметної області, і систему управління базою даних (СКБД), за допомогою якої і реалізуються централізоване керування даними, збереженими в базі, доступ до них і підтримання їх у стані, відповідному стану предметної області.

Ядром БнД є база даних. Існує множина визначень бази даних, зокрема, база даних представляє сукупність взаємопов'язаних, що зберігаються разом даних за наявності такої мінімальної надмірності, яка допускає їх використання оптимальним чином для одного або декількох додатків; дані запам'ятовуються так, щоб вони були незалежні від програм, що їх опрацьовують; для додавання нових або модифікації існуючих даних , а також для пошуку даних в базі даних застосовується загальний керований спосіб.

Перелічимо основні вимоги до організації баз даних.

- База даних - це основа для майбутнього нарощування прикладних програм. Бази даних повинні забезпечувати можливість розробки додатків легше, швидше, дешевше.
- Багаторазове використання даних. Користувачі, які по-різному розуміють одні і ті ж дані, можуть використовувати їх різним чином.
- Збереження витрат розумової праці. Існуючі програми і логічні структури даних не переробляються при внесенні змін до бази даних.
- Гнучкість використання. Звернення до даних або їх пошук здійснюється за допомогою різних методів доступу.
- Простота внесення змін. База даних може збільшуватися і змінюватися без порушення наявних способів використання даних.
- Невеликі витрати. Низька вартість зберігання і використання даних і мінімізація витрат на внесення змін.
- Зменшення надмірності даних. Вимоги нових додатків задовольняються за рахунок існуючих даних, а не шляхом створення нових файлів.
- Продуктивність. Запити на дані задовольняються з такою швидкістю, яка потрібна для використання даних.
- Достовірність даних і відповідність одного рівня оновлення. Необхідно використовувати контроль за достовірністю даних. Система запобігає наявність різних версій одних і тих же елементів даних, доступних користувачам, на різних стадіях оновлення.
- Секретність. Несанкціонований доступ до даних неможливий. Обмеження доступу до одних і тих же даних для різного їх використання може здійснюватися різними способами.
- Захист від спотворення і знищення. Дані повинні бути захищені від збоїв, катастрофічних і кримінальних ситуацій, некомпетентного або зловмисного звернення до них осіб, які можуть помилково оновити їх.
- Готовність. Користувач швидко одержує дані всякий раз, коли це йому необхідно

Программные средства банка данных

Програмні засоби БнД являють собою складний комплекс, що забезпечує взаємодію всіх частин інформаційної системи при її функціонуванні.

Основу програмних засобів БнД представляє СУБД. В ній можна виділити ядро СУБД, що забезпечує організацію введення, обробки і зберігання даних, і інші компоненти, що забезпечують налаштування системи, засоби тестування, утиліти, що забезпечують виконання допоміжних функцій, таких, як відновлення баз даних, збір

статистики про функціонування БнД та ін Важливою компонентою СУБД є транслятори для використовуваних нею мовних засобів.

Більшість СУБД працює в середовищі універсальних операційних систем і взаємодіє з ОС при обробці звернень до БнД, тому можна вважати, що ОС також входить до складу БнД. Для обробки запитів до бази даних (БД) пишуться відповідні програми, які представляють прикладне програмне забезпечення БнД.

З банком даних в процесі створення і функціонування взаємодіють користувачі різних категорій. Основною категорією користувачів є кінцеві користувачі, тобто ті користувачі, для потреб яких і створюється банк даних. Залежно від особливостей створюваного банку даних коло його кінцевих користувачів може істотно розрізнятися. Це можуть бути випадкові користувачі, які звертаються до бази даних час від часу, а можуть бути і регулярні користувачі. Від кінцевих користувачів не повинно вимагатися якихось спеціальних знань в галузі обчислювальної техніки і мовних засобів. Функціонування БнД неможливо без участі фахівців, що забезпечують створення, функціонування і розвиток БнД. Така група фахівців називається адміністратором банку даних (АБД).

Адміністратори банку даних теж є специфічними користувачами БнД. Зазвичай вони звертаються до БнД не за інформацією про предметну область, а до метаінформації, а також використовують ресурси БнД для виконання своїх функцій.

2. Інформаційні об’єкти БД

Наявність інформації і ефективний доступ до неї - це ще не всі аспекти даної проблеми. Уявімо якусь абстрактну ситуацію:

• деяку систему, інформація про яку представляє інтерес;

• спостерігача, здатного сприймати стан системи і в певній формі фіксувати їх у своїй пам'яті (ніяких інших дій спостерігач не виконує).

В цьому випадку говорять, що в пам'яті спостерігача знаходяться дані, що описують стан системи. Таким спостерігачем в загальному випадку і виступають інформаційні системи. Відповідно двом поняттям - «інформація» і «дані» розрізняють два аспекти розгляду питань, пов'язаних з інформаційним забезпеченням: **інфологічний і даталогічий**.

Інфологічний аспект вживається при розгляді питань, пов'язаних зі смисловим змістом даних незалежно від способів їх подання в пам'яті системи.

На етапі інфологічного проектування інформаційної системи повинні бути вирішені наступні питання.

- Про які об'єкти або явища реального світу потрібно накопичувати і обробляти інформацію в системі?
- Які їх основні характеристики та взаємозв'язки між собою будуть враховуватися?
- Уточнення вводяться в інформаційну систему понять про об'єкти та явища, їх характеристики і взаємозв'язки.

Таким чином, на етапі інфологічного проектування виділяється частина реального світу, що визначає інформаційні потреби системи, тобто її предметна область.

При **даталогічному** проектуванні системи, виходячи з можливостей наявних засобів сприймання, зберігання і обробки інформації, розробляються відповідні форми подання інформації в системі за допомогою даних, а також наводяться моделі та методи представлення і перетворення даних, формулюються правила смислової інтерпретації даних.

Опис даних виконується на трьох рівнях, які породили відповідно три схеми: **концептуальну**, **зовнішню** і **внутрішню**.

Концептуальна схема являє собою опис логічної структури всієї БД. Термін «логічна» означає, що опис структури виконується на смисловому рівні, без вказівки способу подання даних в ЕОМ. Розглянемо цей підхід більш докладно. Уже зазначалося, що БД є інформаційна модель реального світу, в якій виділяються об'єкти, властивості (характеристики або ознаки) об'єктів і взаємозв'язок між об'єктами. Наприклад, для об'єкта ТОВАРИ властивостями є Код товару, Найменування товару, Опис товару, Ціна, Код постачальника. Для об'єкта ПОСТАЧАЛЬНИКИ властивостями є Код постачальника, Назва постачальника, Телефон, Факс, Адреса, Прізвище директора. Об'єкти можуть бути пов'язані властивістю Код постачальника.

В БД об'єкти представляються за допомогою записів, властивості за допомогою атрибутів, а взаємозв'язку - за допомогою зв'язків. Записи, атрибути та зв'язки є трьома основними формами подання даних в БД.

Атрибут є елементарне дане - число, символьний рядок, спеціалізоване числове дане і т.д.

Так, для об'єкта ТОВАРИ значеннями атрибутів є конкретне число, наприклад 10, відповідне коду товару; символьний рядок, наприклад «телевізор», яка визначає найменування товару, і т.д. Запис складається із значень декількох атрибутів. Зв'язки, як і атрибути, є елементарними даними. Проте їх функція - безпосередньо пов'язувати два записи. В БД зв'язку реалізуються таким чином, що СУБД, використовуючи зв'язок, може швидко перейти від однієї зв'язку до іншого. Взаємозв'язки реального світу в БД можуть представлятися не тільки у вигляді зв'язків, але і у вигляді атрибутів або записів.

Часто БД проектується таким чином, щоб один або декілька атрибутів однозначно ідентифікували запис. Сукупність значень цих атрибутів називається ключем запису, а самі атрибути - **ключовими атрибутами**. Ключ запису можна розглядати як унікальне ім'я запису, за яким СУБД завжди може знайти цей запис. Крім того, в концептуальній схемі зазвичай дається інформація про типи значень атрибутів (символьні, числові тощо) і про обмеження цілісності, які розглядаються як обмеження на допустимі значення атрибутів, наприклад, вік співробітників не може бути менше 16 років.

Зовнішня схема - це фрагмент концептуальної схеми. Зовнішню схему можна розглядати як погляд користувача на те, які його цікавлять дані БД. Кожен користувач разом з адміністратором БД складає свою зовнішню схему, і при вирішенні даної задачі він може мати доступ тільки до описаних у ній даних і не може звернутися до іншої частини БД. Одній БД, таким чином, може відповідати ряд зовнішніх схем, що визначають інтерфейси прикладних програм при їх взаємодії з БД.

Внутрішня схема являє собою опис способів розміщення даних у зовнішній пам'яті ЕОМ. Від вдалого вибору внутрішньої схеми істотно залежить ефективність доступу до БД. Способи опису внутрішньої схеми в різних СУБД істотно відрізняються один від одного.

Правомірно припустити, що інформаційні об'єкти концептуального рівня володіють більшою тривалістю життя, ніж технологія, що визначає рівень розвитку технічних засобів та програмного забезпечення. Концептуальна схема залишається нечутливою до змін цих технологій.

При проектуванні баз даних вирішуються дві основні проблеми.

1. Відображення об'єктів предметної області в абстрактні об'єкти моделі даних таким чином, щоб це відображення не суперечило семантиці предметної області і було якомога кращим (ефективним, зручним і т.п.). Часто цю проблему називають проблемою логічного проектування баз даних.

2. Забезпечення ефективного виконання запитів до бази даних, тобто раціональне розташування даних у зовнішній пам'яті, створення корисних додаткових структур (наприклад, індексів) з урахуванням особливостей конкретних СУБД. Цю проблему називають проблемою фізичного проектування баз даних.

3. Моделі даних.

Відомі три основних типи моделей даних: ***ієрархічна, мережна і реляційна***. Перші дві з них засновані на поданні інформації про об'єкти у вигляді графів, остання - на табличному. Ієрархічна модель даних організовує дані в виді деревовидної структури і є реалізацією логічних зв'язків: родо-видових відносин або відносин «ціле-частина».

Прикладом простого ієрархічного подання може служити адміністративна структура вищого навчального закладу: академія - відділення - інститут - група (студентська). Графічним способом представлення ієрархічної структури є дерево.

Иерархическая модель данных

Дерево представляє собою ієрархію елементів, які називаються вузлами. Під елементами розуміється список (сукупність, набір) атрибутів, що описують об'єкти. В ієрархічній моделі є кореневий вузол, або корінь дерева. Корінь знаходиться на самому верхньому рівні і не має вузлів, що стоять вище за нього. В одного дерева може бути тільки один корінь. Решта вузли, називаються породженими, пов'язані між собою наступним чином: кожен вузол має вихідний, що знаходиться на більш високому рівні. Так, для нашого прикладу коренем є вузол «Академія», а для вузла «Денне відділення» вузол «Академія» є вихідним. Якщо кожен вузол може бути пов'язаний тільки з одним вихідним вузлом, то на наступному рівні він може мати один, два і більшу кількість вузлів або не мати жодного. В останньому випадку вузли, які не

мають породжених, називаються листям. В ієрархії розглядають рівні, на яких розташований той чи інший вузол.

Між вихідним вузлом і породженими вузлами існує відношення «один до багатьох» («багато до одного»). Таким чином, ієрархічну структуру можна перетворити до вигляду, відображає ієрархічну базу даних. Таким поданням більш зручно користуватися при розгляді складних структур даних.

У загальному випадку ієрархія повинна відповідати таким умовам.

- Одне дерево може мати тільки один корінь.
- Вузол містить один або декілька атрибутів, що описують об'єкт в даному вузлі.
- Породжені вузли можуть додаватися в горизонтальному і у вертикальному напрямках. Практично деякі СУБД накладають обмеження на кількість рівнів ієрархії, тому при відображенні концептуальної моделі в логічну модель даних (ієрархічну) слід враховувати технічні можливості використовуваної СУБД.
- Доступ до породжених вузлів можливий тільки через вихідний вузол, тому існує тільки один шлях доступу до кожного вузла.
- Теоретично можливе існування необмеженого числа примірників вузла кожного рівня. При цьому кожен примірник вихідного вузла починає логічний запис.

Перевагами даної моделі є наявність промислових СУБД, що підтримують дану модель, простота розуміння використовуваного принципу ієрархії, забезпечення певного рівня незалежності даних.

До основних недоліків такого виду моделі можна віднести наступні: складність відображення зв'язків «Багато до багатьох»;

ієрархія в значній мірі ускладнює операції включення інформації про нові об'єкти в базу даних і видалення застарілої; доступ до будь-вузлу можливий тільки через кореневий.

Мережева модель даних. В основу мережевої моделі даних (рис. 5.25) покладені мережеві структури. Припустимо, нам необхідно графічно представити відносини між об'єктами «Студентський колектив» і «Студентська група», «Кімната в гуртожитку» і «Студент». Дана схема не є ієрархічної, так як породжений елемент «Студент» має два вихідних («Студентська група» і «Кімната в гуртожитку»). ***Відносини між об'єктами, в яких породжений елемент має більше одного вихідного, описуються у вигляді мережевої структури.*** Відмінність мережевої структури від ієрархічної полягає в тому, що будь-який елемент в мережевій структурі може бути пов'язаний з будь-яким іншим елементом.

У мережній структурі між об'єктами присутні два види взаємозв'язків: «Один до багатьох» і «Багато хто до одного». Цей вид зв'язків, як ми бачили раніше, закладений в ієрархічних структурах за умови, що дані зв'язки існують відповідно між вихідними і породженими, а зв'язок «Багато хто до одного» - між породженими і вихідними вузлами. У разі виконання цієї умови для відповідних вузлів мережевий схеми маємо просту мережеву структуру, яку необхідно відрізняти від складної. Складною мережевою структурою називають схему, в якій присутній хоча б один зв'язок «Багато до багатьох». Поділ мережевих структур на два типи (складні і прості) необхідно тому, що структури, побудовані з використанням зв'язку «Багато до багатьох», вимагають для їх реалізації використання більш складних методів.

База даних, описувана мережевий моделлю, складається з декількох областей. Кожна область складається з записів, які, в свою чергу складаються з полів. Об'єднання записів у логічну структуру можливо не тільки по областях, але і за допомогою так званих наборів. Термін набір є основною конструкцією мови систем баз даних Кодас. По суті, набір - це пойменоване дворівневе дерево, яке дозволяє будувати багаторівневі дерева і прості мережеві структури. Використовуючи множина таких дворівневих зв'язків, фахівець з аналізу систем може конструювати досить складні структури даних. Набір - це екземпляр пойменованої сукупності записів. Кожен тип набору являє собою відношення між двома або кількома типами записів. Для кожного типу набору один тип записів може бути оголошений його власником і один або декілька інших типів записів - членами набору. Кожен набір повинен містити один примірник записів, що має тип запису-власника, і може містити будь-яку кількість примірників кожного типу записів - членів набору.

Наприклад, набір можна використовувати для об'єднання записів про студентів однієї групи. Тоді тип набору можна визначити як «Склад групи» з типом запису-власника «Група» і типом записів-членів «Студент».

Перерахуємо властивості, притаманні набору:

- набір - це пойменована сукупність пов'язаних записів;
- в кожному примірнику набору є тільки один екземпляр власника;
- примірник набору може містити нуль, один або кілька записів-членів;
- набір вважається порожнім, якщо ні один примірник запису-члена не пов'язаний з відповідним примірником запису-власника;
- примірник набору існує після запам'ятовування запису-власника;
- тип набору являє логічний взаємозв'язок «Один до багатьох» між власником і членом набору. При цьому не передбачається, що екземпляри членів набору повинні розташовуватися поблизу примірника набору у фізичній пам'яті;
- кожному типу набору присвоюється ім'я, що дозволяє одній і тій же парі типів об'єктів брати участь в декількох взаємозв'язках.

Необхідно розрізняти тип та примірник набору. Але попередньо пояснимо різницю між поняттями типу і примірника запису. «Студент» - це тип запису, а рядок символів «Іванов Іван Іванович, комн.23» - примірник типу запису «Студент». Таким чином, в базі даних можуть зберігатися один примірник або кілька примірників запису деякого типу. Аналогічне відношення існує і між типом набору і примірника.

Сетевая модель данных

У моделі даних, що представляє взаємозв'язок «Один до багатьох», тип запису-власника «містить» від 0 до N примірників типу запису-члена. У свою чергу, тип запису-члена в іншому типі набору може грати роль типу запису-власника. Запис-власник даного набору може грати ту ж роль в декількох наборах. Така структура являє собою ієрархію. Отже, ієрархічна модель даних є окремим випадком мережевої моделі.

Істотне розходження між мережевою та ієрархічною моделями даних полягає в тому, що в мережевій моделі кожен запис може брати участь в будь-якому числі наборів. Наприклад, в мережевий моделі, представленої двома типами наборів «Викладач веде дисципліну» і «Студент навчається дисципліни», запис-член «Дисципліна» входить в обидва типи наборів і по суті є зв'язкою цих типів наборів. Крім того, будь-який запис мережевої моделі може грати роль як власника, так і члена набору.

Основний недолік мережевої моделі полягає в її складності. Прикладний програміст повинен детально знати логічну структуру бази даних, оскільки йому необхідно здійснювати навігацію серед різних екземплярів наборів і записів, тобто програміст повинен представляти «своє» поточний стан в примірниках наборів при «просуванні» по базі даних. Іншим недоліком є можлива втрата незалежності даних при реорганізації бази даних. Крім того, в мережевій моделі даних уявлення, що використовується прикладною програмою, складніше, ніж в ієрархічній моделі, тому і процедура складання прикладних програм може виявитися складнішою.

4. Реляційна модель даних.

В даний час найбільшого поширення при розробці БД отримала реляційна модель даних, яка дозволяє визначати:

- структури даних;
- операції із запам'ятовування та пошуку даних;
- обмеження, пов'язані із забезпеченням цілісності даних.

Основне її позитивне відміну від ієрархічної і мережної моделей - відсутність зв'язків. Взаємозв'язки в реляційної моделі розглядаються як об'єкти і представляються наступним чином: імена (ключі) записів використовуються як значення атрибутів

Рис. 5.27. Реляционная модель склада

Рис. 5.26. Реляционная концептуальная схема информационной модели фирмы-поставщика

інших записів (. Для того щоб можна було встановити зв'язок між двома БД Фірма-постачальник і Склад, користувачі цих БД повинні мати узгоджене уявлення про предметну область, яка відображається в БД, і про відповідність даного опису предметної області зовнішнього світу в кожен момент часу. У явному вигляді зв'язки можуть бути виражені, наприклад, в діаграмі зв'язків між об'єктами.

Загальна структура даних в реляційній моделі може бути представлена у вигляді таблиці, в якій кожен рядок відповідає логічному запису, а заголовки стовпців є назвами

полів (атрибутів) в записах. Кожен запис в реляційної моделі має унікальне ім'я (первинний ключ), яке в загальному випадку складається із значень декількох атрибутів. Ключ дозволяє однозначно ідентифікувати запис серед множини інших записів. Якщо ключ запису складається із значень декількох атрибутів, то він називається складовим, а якщо з одного атрибута - простим. Наприклад, будь-який запис таблиці ПОСТАВКИ ідентифікується складовим ключем: Код поставки, Код постачальника і Код товару. Імена всіх записів зберігаються в самих записах (що не мало місця для ієрархічної і мережевої моделей). Щоб зв'язати дві таблиці, необхідно ключ першої таблиці увести до складу ключа другої таблиці (можливо збіг ключів), в іншому випадку потрібно в структуру першої таблиці ввести зовнішній ключ - ключ другої таблиці. Наприклад, для зв'язку таблиць ТОВАРИ і ПОСТАЧАЛЬНИКИ в таблицю ТОВАРИ введений зовнішній ключ Код постачальника.

Тип даних і розмір первинного та зовнішнього ключів повинні збігатися. Таким чином, реляційна база даних з логічної точки зору може бути представлена множиначю двовимірних таблиць самого різного предметного наповнення.

Основними перевагами реляційної моделі даних є:

- простота і доступність;
- незалежність даних;
- гнучкість;
- можливість непроцедурних запитів.

При описі реляційних БД часто використовується своя термінологія. Наприклад, множина допустимих значень (область визначення) атрибуту називають **доменом**, запис - **кортежем**, а множина однотипних записів - **відношенням**. Список імен атрибутів одного відношення називається схемою відношення; кожне відношення, як правило, має свою назву (ім'я). Від терміна «відношення» (від англ. Relation) походить назва реляційна модель даних.

Одним з вимог, що пред'являються до відношень, є вимога **нормалізації**. Згідно з умовами нормалізації в кожному кортежі містяться дані, що відображають або властивості «реального світу», або зв'язку між двома або кількома об'єктами. Про відношення кажуть, що воно має **нормальну форму** або нормалізовано, якщо воно задовольняє певним обмежують умов. Обмеження, притаманне всім нормальним формам, полягає в тому, що відношення не повинні містити вкладень, тобто ніяке відношення не може бути визначено як член іншого відношення. Метою введення будь нормальної форми є запобігання різного роду порушень нормального функціонування (аномалій оновлення) в результаті коригувань. Порядок записів у відношенні довільний. Неприпустимо наявність відносно двох записів з однаковими ключами. Часто замість терміну відношення використовується термін таблиця або реляційна таблиця, в якій кортеж є рядок, кожен стовпець відповідає домену.

Фундаментальні властивості відношень.

1. ***Відсутність кортежів-дублікатів.*** У класичній теорії множин, за визначенням, кожна множина складається з різних елементів. Так як відношення визначається як множина кортежів, отже, воно не може містити кортежів-дублікатів. З цієї властивості випливає наявність у кожного відношення так званого первинного ключа - набору атрибутів, значення яких однозначно визначають кортеж відношенні. Для кожного

відношення принаймні повний набір його атрибутів володіє цією властивістю. Однак при формальному визначенні первинного ключа потрібне забезпечення його «мінімальності», тобто в набір атрибутів первинного ключа не повинні входити такі атрибути, які можна відкинути без втрати основної властивості - однозначного визначення кортежу.

2. ***Відсутність упорядкованості кортежів.*** Відсутність вимоги до підтримання порядку на множини кортежів відношенні дає додаткову гнучкість СУБД при зберіганні баз даних у зовнішній пам'яті і при виконанні запитів до бази даних. Це не суперечить тому, що при формулюванні запиту до БД можна зажадати сортування результуючої таблиці у відповідності зі значеннями деяких стовпців. Такий результат - це не відношення, а деякий впорядкований список кортежів.

3. ***Відсутність упорядкованості атрибутів.*** Атрибути відношень не впорядковані, оскільки, за визначенням, схема відношення є множина пар ім'я атрибута, ім'я домену. Для посилання на значення атрибута в кортежі відношенні завжди використовується ім'я атрибута. Ця властивість теоретично дозволяє, наприклад, модифікувати схеми існуючих відношень не тільки шляхом додавання нових атрибутів, але і шляхом видалення існуючих атрибутів. Однак у більшості існуючих систем така можливість не допускається, і хоча впорядкованість набору атрибутів відношенні явно не потрібно, часто як неявного порядку атрибутів використовується їх порядок у лінійній формі визначення схеми відношенні.

4. ***Атомарність*** значень атрибутів. Значення всіх атрибутів є атомарними. Це випливає з визначення домену як потенційної множини значень простого типу даних, тобто серед значень домену не можуть міститися множини значень (відношення).

Найбільш поширеним трактуванням реляційної моделі даних є те, що реляційна модель складається з трьох частин, що описують різні аспекти реляційного підходу: *структурної частини, маніпуляційної частини й цілісної частини*; коротко зупинимося на кожній з них.

У **структурній** частині моделі фіксується, що єдиною структурою даних, що використовується в реляційних БД, є нормалізоване n-арное відношення. В теорії реляційних баз даних звичайно виділяється наступна послідовність нормальних форм (• перша нормальна форма (1NF);• друга нормальна форма (2NF);• третя нормальна форма (3NF);• нормальна форма Бойса-Кодда;• четверта нормальна форма (4NF);• п'ята нормальна форма, або нормальна форма проекції з'єднання (5NF або PJ / NF).

Кожній нормальній формі відповідає деякий певний набір обмежень, і відношення знаходиться в деякій нормальній формі, якщо задовольняє властивому їй набору обмежень. Прикладом є обмеження першої нормальної форми: значення всіх атрибутів відношенні повинні бути атомарними.

Основні властивості нормальних форм:

- кожна наступна нормальна форма в деякому сенсі покращує властивості попередньої;
- при переході до наступної нормальної форми властивості попередніх нормальних форм зберігаються.

Оскільки вимога першої нормальної форми (1NF) є базовою вимогою класичної реляційної моделі даних, ми будемо вважати, що вихідний набір відношень вже відповідає цій вимозі.

Відношення R знаходиться у другій нормальній формі (2NF) в тому і тільки в тому випадку, коли перебуває в 1NF, і кожен неключовий атрибут повністю залежить від первинного ключа. Неключовим атрибутом називається будь-який атрибут відношення, не входить до складу первинного ключа.

Відношення R знаходиться в третій нормальній формі (3NF) в тому і тільки в тому випадку, якщо всі неключові атрибути R взаємно незалежні і повністю залежать від первинного ключа.У наших прикладах, що описують реляційні моделі фірми-постачальника і складу, всі відношенні знаходяться в INF, 2NF і 3NF.

Відношення R знаходиться в нормальній формі Бойса-Кодда (BCNF) в тому і тільки в тому випадку, якщо кожен детермінант є ключем. Детермінантом називається будь-який атрибут, від якого повністю функціонально залежить деякий інший атрибут. Наприклад, відношенні ПОСТАВКИ і ТОВАР в моделі складу, ПРАЦІВНИКИ, ПОСТАВКИ і ДОГОВОРИ в моделі фірми-постачальника. Зауваження. Легко помітити, що якщо відносно є тільки один можливий ключ (який є первинним ключем), то це визначення стає еквівалентним визначенню третій нормальній форми.

На практиці Третя нормальна форма схем відношень є достатньою в більшості випадків, і приведенням до третьої нормальної форми процес проектування реляційної бази даних зазвичай закінчується. Однак іноді процес нормалізації може бути продовжений.

Маніпуляційна частина реляційної моделі складається з операцій запам'ятовування і пошуку даних. Ці операції діляться на дві групи: операції на множинах (об'єднання, перетин, різниця, добуток) і реляційні операції (вибрати, спроектувати, з'єднати, розділити). Будь-яка мова маніпулювання даними, що забезпечує всі ці операції, є реляційно повною. Залежно від способу формування виразів мови його називають або реляційної алгеброю, або реляційних обчисленням. Мови маніпулювання даними, які можуть використовуватися кінцевими користувачами в діалоговому режимі (тобто не є вкладеними в мову програмування головної системи), часто називають мовами запитів.

У **цілісній частині** реляційної моделі даних фіксуються дві базові вимоги цілісності, які повинні підтримуватися в будь-якій реляційної СУБД. Перша вимога називається вимогою **цілісності сутностей.** Об'єкту або сутності реального світу в реляційних БД відповідають кортежі відношень. Для дотримання цілісності суті досить гарантувати відсутність в будь-якому відношенні кортежів з одним і тим же значенням первинного ключа. Друга вимога називається вимогою цілісності за посиланнями і є трохи більш складною. Очевидно, що при дотриманні нормалізоване відношень складні сутності реального світу представляються в реляційній БД у вигляді кількох кортежів декількох відношень. Атрибут, значення якого однозначно характеризують сутності, представлені кортежами деякого іншого відношення (тобто описують їх первинного ключа), називається зовнішнім ключем. Кажуть, що відношення, в якому визначено зовнішній ключ, посилається на відповідне відношення, в якому такий же атрибут є первинним ключем. Вимога **цілісності по посиланнях,** або вимога зовнішнього ключа, полягає в тому, що **для кожного значення зовнішнього ключа повинен знайтися кортеж з таким самим значенням первинного ключа в відношенні**, на яке ведеться посилання,

або значення зовнішнього ключа повинне бути повністю невизначеним (тобто ні на що не вказувати).

Перший підхід полягає в тому, що забороняється проводити видалення кортежу, на який існують посилання (тобто спочатку потрібно або видалити посилання кортежі, або відповідним чином змінити значення їх зовнішнього ключа).При другому підході при видаленні кортежу, на який є посилання, у всіх посилаються кортежу значення зовнішнього ключа автоматично стає невизначеним.

Третій підхід (каскадне видалення) полягає в тому, що при видаленні кортежу з відношення, на яке веде посилання, з посилається відношенні автоматично видаляються всі посилаються кортежі.

Лекція 7. Системи керування базами даних.

1. *Розробка проекту бази даних, створення бази даних, зміна проекту бази даних.*
2. *Робота з даними в таблиці.*
3. *Створення БД за допомогою СКБД.*

1. Розробка проекту бази даних, створення бази даних, зміна проекту бази даних.

Створення БД починається з проектування. ***Етапи проектування БД:***

- Дослідження предметної області.
- Аналіз даних (тем і їх атрибутів).
- Визначення відношень між даними і визначення первинних і вторинних (зовнішніх) ключів.

В процесі проектування визначається структура реляційної БД (склад таблиць, їх структура і логічні зв'язки). Структура таблиці визначається складом стовпців, типом даних і розмірами стовпців, ключами таблиці.

До базових понятть моделі БД «тема – зв'язок» відносяться: теми, зв'язки між ними та їх атрибути (властивості).

Тема – будь-який конкретний або абстрактний об'єкт в даній предметній області. Тема – це базовий тип інформації, який зберігається в БД (у реляційній БД кожній темі призначається таблиця). До тем можуть відноситися: студенти, клієнти, підрозділи і т.д. Екземпляр (примірник) теми і тип теми – це різні поняття. Поняття тип теми відноситься до набору однорідних осіб, предметів або подій, промовців як ціле (наприклад, студент, клієнт і так далі). Екземпляр (примірник) теми відноситься, наприклад, до конкретної особи в наборі. Типом теми може бути студент, а екземпляром – Петренко, Сидоренко і т. д..

Атрибут – це властивість теми в предметній області. Його ім’я має бути унікальним для конкретного типу теми. Наприклад, для теми студент можуть бути використані наступні атрибути: прізвище, ім'я, по батькові, дата і місце народження, паспортні дані і т. д.. У реляційній БД атрибути зберігаються в полях (стовпцях) таблиць.

Зв'язок – взаємозв'язок між темами в предметній області. Зв'язки є з'єднаннями між частинами БД (у реляційній БД – це з'єднання між записами таблиць).

Тема – це дані, які класифікуються за типом, а зв'язки показують, як ці типи даних співвідносяться один з одним. Якщо описати деяку предметну область в термінах тема – зв'язок, то отримаємо модель тема - зв'язок для цієї БД.

Розглянемо предметну область: Деканат (Успішність студентів). В БД «Деканат» повинні зберігатися дані про студентів, групи студентів, про оцінки студентів з різних дисциплін, про викладачів, про стипендії і т. д.. Обмежимося даними про студентів, групи

студентів і про оцінки студентів з різних дисциплін. Визначимо теми, атрибути тем і основні вимоги до функцій БД з обмеженими даними.

Основними наочно значущими темами БД «Деканат» є:

- Студенти,
- Групи студентів,
- Дисципліни,
- Успішність.

Основні наочно значущі атрибути тем:

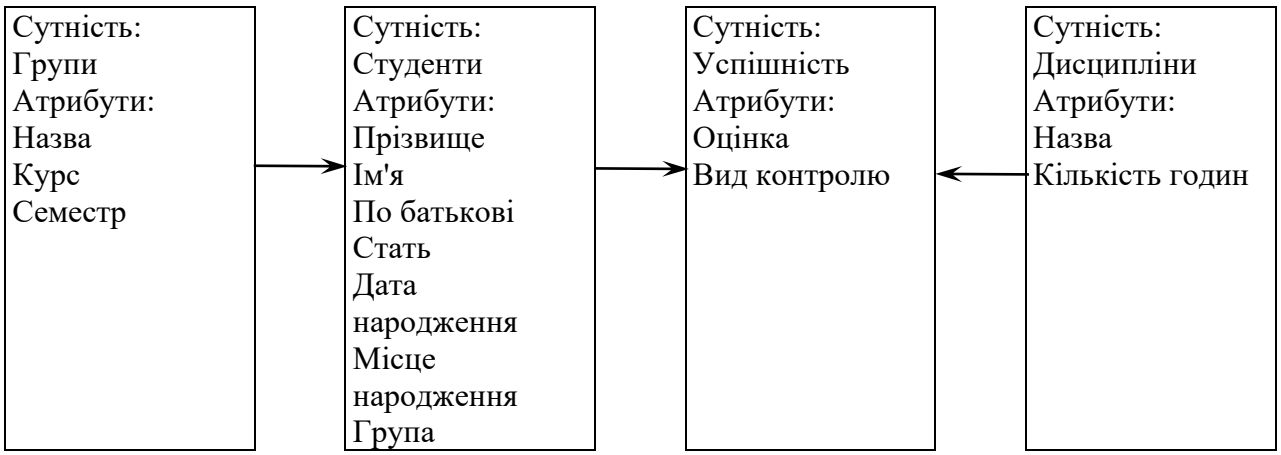
- студенти – прізвище, ім'я, по батькові, стать, дата і місце народження, група студентів;
- групи – назва, курс, семестр;
- дисципліни – назва, кількість годин
- успішність – оцінка, вид контролю.

Основні вимоги до функцій БД:

- вибрати успішність студента з різних дисциплін із вказуванням загальної кількості годин та виду контролю;
- вибрати успішність студентів по групах і дисциплінах;
- вибрати дисципліни, що вивчаються групою студентів на певному курсі або у певному семестрі.

З аналізу даних предметної області виходить, що кожній темі необхідно призначити просту двовимірну таблицю (відношення). Далі необхідно встановити логічні зв'язки між таблицями. Між таблицями Студенти і Успішність необхідно встановити такий зв'язок, щоб кожному запису з таблиці Студенти відповідало кілька записів в таблиці Успішність, тобто один-до-багатьох, оскільки у кожного студента може бути кілька оцінок. Логічний зв'язок між темами Групи – Студенти визначений як один-до-багатьох виходячи з того, що в групі є багато студентів, а кожен студент входить до складу однієї групи. Логічний зв'язок між темами Дисципліни – Успішність визначений як один-до-багатьох тому, що з кожної дисципліни може бути виставлено декілька однакових оцінок різним студентам.

На основі викладеного вище складаємо модель тема – зв’язок для БД «Деканат»



⟶ - стрілка є умовним позначенням зв'язку: один – до – багатьох.

Для створення БД необхідно застосувати одну з відомих СУБД, наприклад OpenOffice.org Base.

Система управління базами даних OpenOffice.org Base та її основні можливості. Додаток OpenOffice.org Base – це настільна система управління реляційними базами даних (СУБД), призначена для роботи на автономному персональному комп'ютері (ПК) або в локальній обчислювальній мережі.

СУБД OpenOffice.org Base володіє потужними, зручними і гнучкими засобами візуального проектування об'єктів за допомогою Майстрів, що дозволяє користувачеві при мінімальній попередній підготовці досить швидко створити повноцінну інформаційну систему на рівні таблиць, запитів, форм і звітів.

До основних можливостей СУБД OpenOffice.org Base можна віднести наступні:

- Проектування базових об'єктів – двовимірних таблиць з полями різних типів даних.
- Створення зв'язків між таблицями з підтримкою цілісності даних, каскадного оновлення полів і каскадного видалення записів.
- Введення, зберігання, перегляд, сортування, зміна і вибірка даних з таблиць з використанням різних засобів контролю інформації, індексування таблиць і апарату алгебри логіки.
- Створення, модифікація і використання похідних об'єктів (запитів, форм і звітів).

Користувацький інтерфейс OpenOffice.org Base. При запуску Base починає виконуватись Майстер баз даних, за допомогою якого можна відкрити існуючу БД чи створити нову БД. При виборі команди **Створити нову базу даних** відкриється вікно діалогу, в якому необхідно вибрати диск і папку для зберігання БД, а також задати ім'я БД.

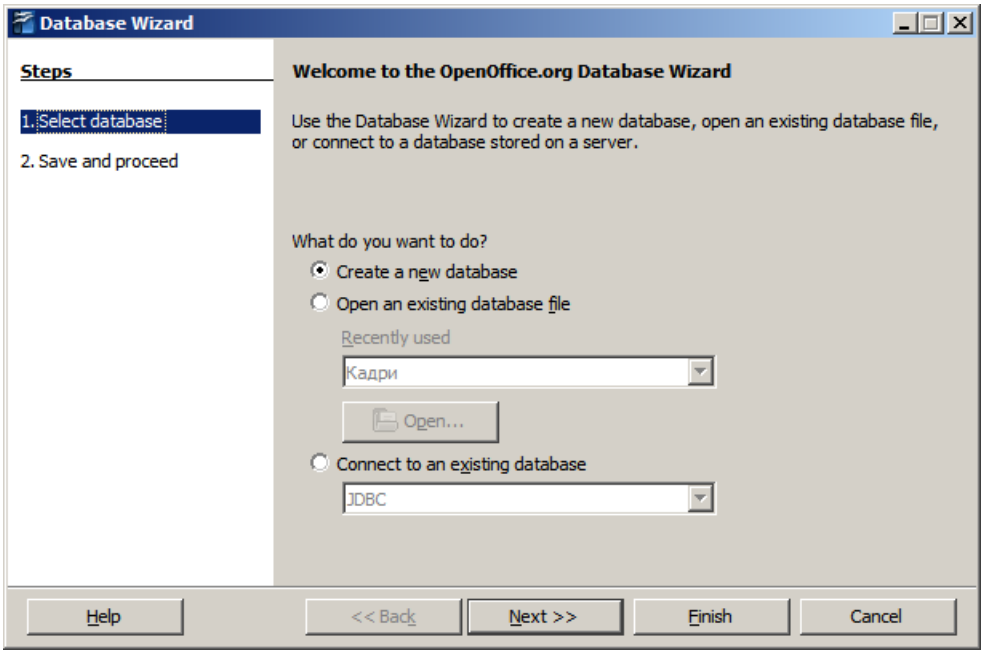
Головне вікно **додатку** Microsoft Access **складається з наступних областей:**

- рядок заголовка;
- рядок меню;
- панель інструментів;
- вікно бази даних;
- рядок стану.

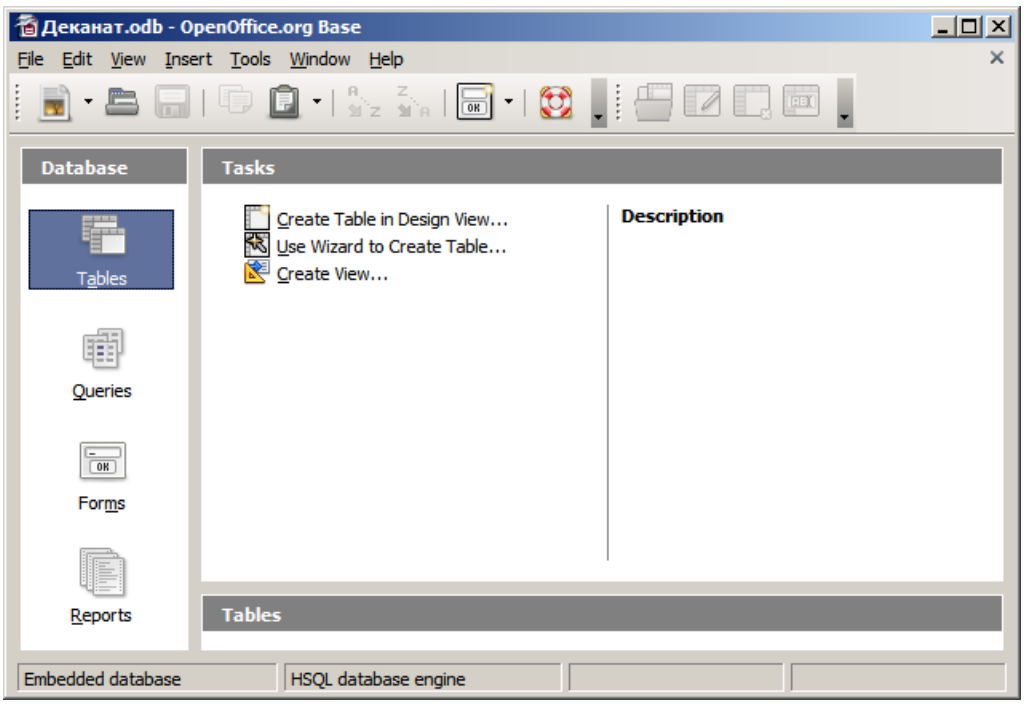
Рядок меню містить групи команд об'єднаних за функціональною ознакою: Файл, Правка, Вигляд, Вставка, Сервіс, Вікно, Довідка.

Рядок стану знаходиться внизу головного вікна і призначений для виведення короткої інформації про поточний режим роботи.

Вікно бази даних **містить вкладки з об’єктами бази даних:** Tables (Таблиці), Queries (Запити), Forms (Форми), Reports (Звіти).



У головному вікні з'явиться вікно БД з призначеним ім'ям, наприклад «Деканат».



В одній базі даних може бути кілька об’єктів однакового типу.

Таблиця – основний об’єкт бази даних, в якому зберігається інформація. Кожен рядок таблиці називається ***записом***, кожен стовпчик – ***полем*** запису. Запис містить набір даних про один об’єкт (наприклад, назва дисципліни, кількість годин), а поле – однакові дані про всі об’єкти (наприклад, назви всіх занесених у таблицю дисциплін).

Запит – об’єкт бази даних, за допомогою якого можна виконати вибірку з таблиць на основі заданих критеріїв, модифікувати таблиці, виконати обчислення.

Форма - бланк (або маска), який накладається на таблицю з даними. Форма спрощує процес заповнення таблиці. За допомогою форми можна обмежити обсяг інформації, доступної користувачу, який звертається до бази.

Звіт – об'єкт для відображення підсумкових даних у зручному вигляді.

2. Робота з даними в таблиці.

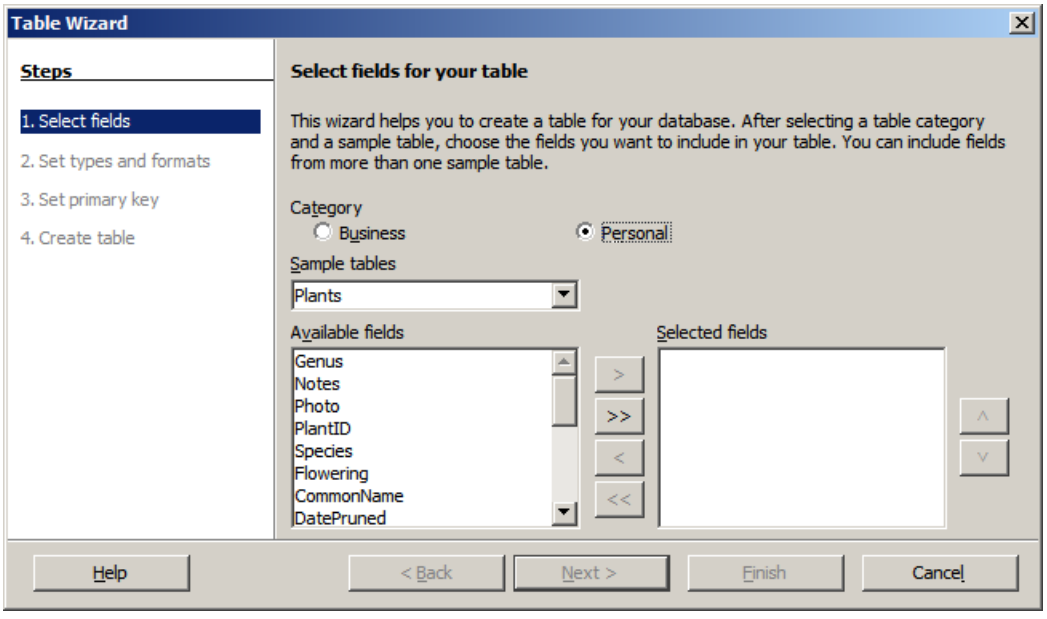
При першому відкриванні вікна бази даних Base завжди активізує вкладку Таблиці і виводить на екран список режимів створення таблиць:

- Створення таблиці в режимі конструктора;
- Створення таблиці за допомогою майстра;
- Створення вигляду

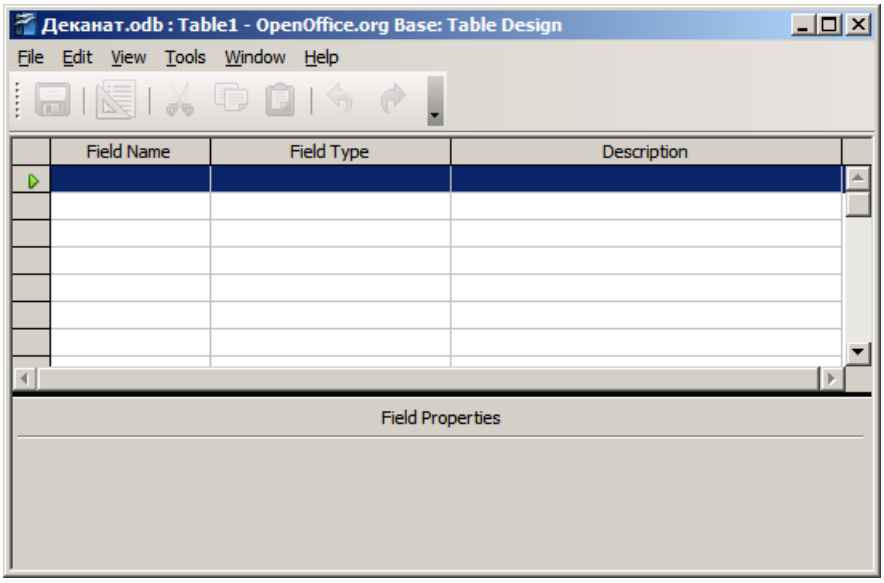
Для створення нової таблиці можна вибрати Майстер таблиць для визначення полів таблиці за допомогою списків зразків таблиць і полів. Для створення довільної таблиці доцільно користуватися режимом Конструктора.

Для вибору необхідного режиму створення таблиць можна двічі клацнути на одному з них в списку режимів, відкриється необхідний режим.

При виборі режиму Майстер таблиць відкриється вікно, в якому за допомогою зразків таблиць і полів легко сформувати поля нової таблиці.



Проте якщо у Майстрі немає необхідного зразка таблиці, то необхідно вибрати режим Конструктора, відкриється вікно Конструктора таблиць.



Склад (структура) таблиці визначається в області проекту таблиці, яка складається з трьох колонок:

- Ім'я поля;
- Тип даних;
- Опис.

Типи даних необхідно вибрати зі списку, що розкривається:

Tiny Integer [TYNYINT] – дуже коротке ціле (8 біт). Використовується найчастіше для нумерації при невеликій кількості позицій.

Bigint [BIGINT] – довге ціле. Розрядність удвічі більша за прийняту в системі.

Image [LONGVARBINARY] – двійковий об’єкт розміром до сотень Кбайт.

Binary [VARBINARY] – двійковий об’єкт змінного розміру. Економить пам’ять, якщо в різних записах поле має різний розмір.

Binary (fix) [BINARY] – двійковий об’єкт фіксованого розміру.

Memo [LONGVARCHAR] – великий текст (до 64 Кбайт).

Text (fix) [CHAR] – рядок з фіксованою кількістю символів. Економії пам'яті не відбувається.

Number [NUMERIC] – видиме, натуральне число.

Decimal [DECIMAL] – десяткове число.

Integer [INTEGER] – ціле 32 біти.

Small Integer [SMALLINT] – коротке ціле (16 біт)

Float [FLOAT] – число з плаваючою крапкою, мало чим відрізняється від DECIMAL, але призводить до похибок заокруглення, тому не використовується при фінансових обчисленнях.

Real [REAL] – дійсне число, представляється як 32-розрядні мантиса й порядок.

Double [DOUBLE] – аналогічно REAL, але розрядність подвоюється.

Text [VARCHAR] – рядок до 256 знаків з економією пам’яті на більш коротких екземплярах.

Text [VARCHAR_IGNORECASE] – не розрізняються малі і великі букви.

Yes/No [BOOLEAN] – логічне значення.

Date [DATE]

Time [TIME]

Date/Time [TIMESTAMP] – так званий “UNIX timestamp” – кількість мілісекунд, що пройшли з початку “ери UNIX”.

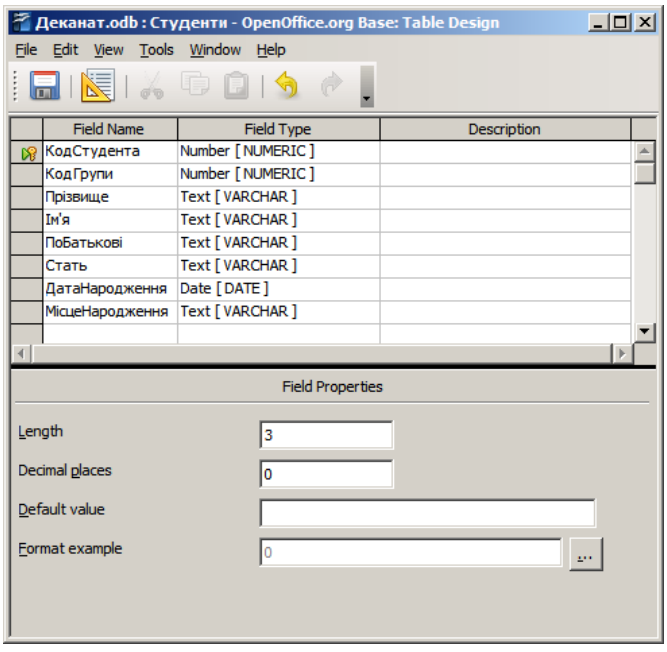
OTHER [OTHER]

В області **Властивості поля** призначають властивості для кожного поля (наприклад, розмір, формат, індексоване поле і так далі).

При створенні структури таблиці в першу колонку вводять Ім'я поля, потім необхідно вибрати тип даних (за замовчуванням призначається текстовий тип даних, якщо цей тип даних не підходить, то виберіть самостійно зі списку, що розкривається). Потім можна ввести у третю колонку опис поля.

3. Створення БД за допомогою СКБД

Створення БД за допомогою СКБД Base починається зі створення структури таблиць і встановлення зв'язків між таблицями. При виборі режиму Конструктор відображатиметься вікно Конструктора таблиць, в якому необхідно визначити структуру нової таблиці для теми Студенти.



У перший рядок колонки Ім'я поля вводимо КодСтудента, у колонці Тип даних вибираємо тип даних – Number (Числовий).

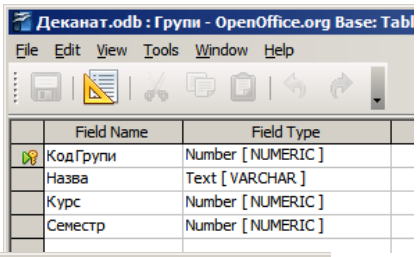
Визначаємо поле **КодСтудента** як поле первинного ключа, для цього виділяємо його і з контекстного меню

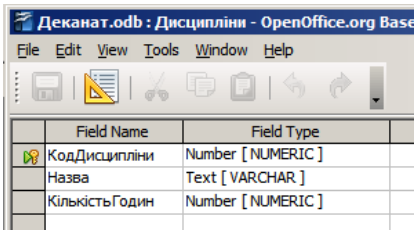
вибираємо команду **Primary Key (Первинний ключ)**, зліва від імені поля з'явиться зображення ключа.

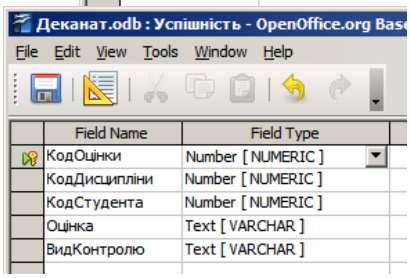
Далі в другий рядок вводимо назву поля **КодГрупи** і вибираємо тип даних **Number (Числовий)**. Потім створюється решта полів відповідно до даних, представлених в моделі "тема - зв'язок".

Після створення структури таблиці необхідно зберегти її під іменем **Студенти**.

Далі створюються структури решти таблиць: Групи, Дисципліни, Успішність.

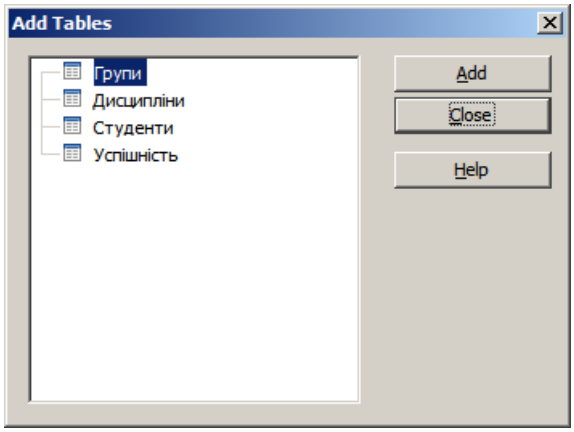




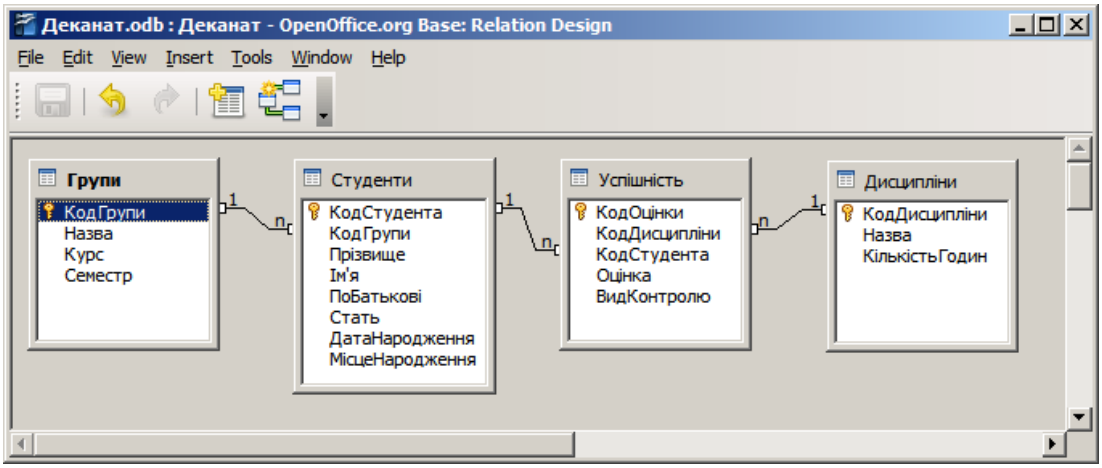


Після створення структури таблиць, що входять до БД "Деканат", необхідно встановити зв'язок між ними.

Установка зв'язків між таблицями. Після створення структури таблиць (Студенти, Групи, Дисципліни, Успішність) для бази даних "Деканат" необхідно встановити зв'язки між таблицями. Зв'язки між таблицями в БД використовуються при формуванні запитів, розробці форм, при створенні звітів. Для створення зв'язків необхідно закрити всі таблиці і вибрати команду **Tools → Relationships… (Сервіс → Зв'язки…)**, з'явиться активне діалогове вікно додавання таблиць на на фоні неактивного вікна створення зв'язків.

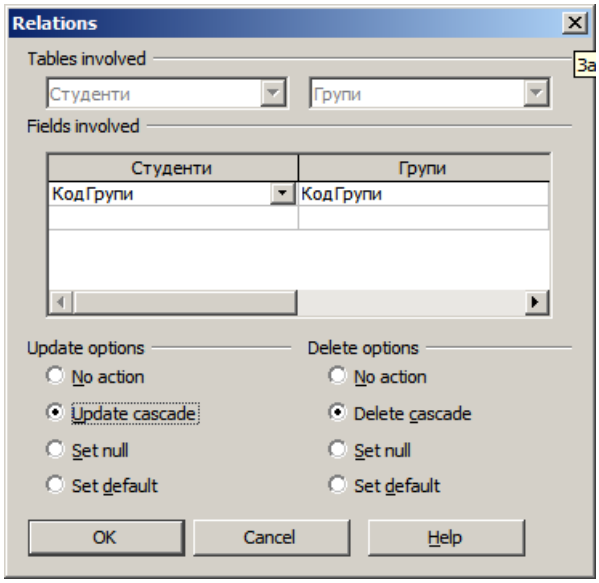


У вікні додавання таблиць необхідно виділити імена таблиць і натиснути кнопку **Додати**, при цьому у вікно створення зв'язків додаються таблиці. Після появи всіх таблиць слід закрити вікно додавання таблиць.



Наступний крок - це установка зв'язків між таблицями. Для цього необхідно перемістити поле КодГрупи з таблиці Групи на відповідне поле таблиці Студенти, в результаті цієї операції встановиться зв'язок.

Двічі клацнувши на графічному зображенні зв’язку, відкриваємо вікно діалогу **Зв'язки**. Необхідно активізувати прапорці: "Каскадне оновлення зв'язаних полів" і "Ккаскадне видалення зв'язаних записів".



Аналогічним чином треба зв'язати поля КодСтудента в таблицях Студенти і Успішність, а потім поля КодДисципліни в таблицях Успішність і Дисципліни. У результаті отримаємо базу даних зі встановленими зв’язками.

Далі необхідно здійснити заповнення всіх таблиць. Заповнення таблиць можна починати з заповнення довільної таблиці, наприклад, таблиці Студенти. У вікні Бази даних виділяємо потрібну таблицю, потім виконуємо клацання на кнопці Відкрити. На екрані з'явиться структура таблиці БД в режимі таблиці. Нова таблиця складається з одного порожнього рядка.

Заповнення проводиться по записах, тобто вводиться інформація для всього рядка цілком. Після введення першого запису порожній запис зміщується в кінець таблиці. Перехід до наступного поля здійснюється натисненням клавіші Tab.

Студенти - Деканат - OpenOffice.org Table Data View

File Edit View Insert Tools Window Help

КодСтудента	КодГрупи	Прізвище	Ім'я	ПоБатькові	Стать	ДатаНародження	МісцеНародження
1	1	Стецько	Роман	Іванович	чол	02/02/93	м. Львів
2	3	Зарубан	Ганна	Степанівна	жін	10/11/92	м. Тернопіль
3	2	Кандиба	Світлана	Андріївна	жін	18/12/92	м. Київ
4	4	Миронов	Антон	Ігорович	чол	30/12/99	м. Одеса
5	1	Зарубан	Микола	Степанович	чол	10/11/92	м. Тернопіль
6	3	Будзь	Сергій	Кирилович	чол	02/09/92	м. Городок
7	1	Іваненко	Семен	Семенович	чол	27/10/93	м. Конотоп
8	4	Карабін	Тамара	Сергіївна	жін	05/05/93	м. Жовква
9	3	Зима	Петро	Петрович	чол	15/03/90	м. Дрогобич
10	2	Шовкова	Ольга	Михайлівна	жін	12/09/92	м. Стрий

Аналогічним чином заповнюється решта таблиць: Групи, Успішність, Дисципліни.

Групи - Деканат - OpenOffice.org Table Data

File Edit View Insert Tools Window Help

КодГрупи	Назва	Курс	Семестр
1	СЕ-11	1	2
2	ЕП-11	1	2
3	ОА-11	1	2
4	ОА-12	1	2

Дисципліни - Деканат - OpenOffice.org Table Data View

File Edit View Insert Tools Window Help

КодДисципліни	Назва	КількістьГодин
1	Мікроекономіка	254
2	Інформатика	160
3	Математика	140
4	Іноземна мова	156
5	Фізкультура	72

Успішність - Деканат - OpenOffice.org Table Data View

File Edit View Insert Tools Window Help

КодОцінки	КодДисципліни	КодСтудента	Оцінка	ВидКонтролю
1	1	1	5	іспит
2	2	5	4	залік
3	1	7	4	іспит
4	3	2	5	іспит
5	1	3	3	залік
6	2	10	5	іспит
7	4	5	5	залік
8	1	9	3	іспит
9	4	4	5	залік
10	5	4	4	іспит
11	3	2	3	іспит
12	1	5	4	залік

Лекція 8. Системи керування базами даних. Запити на вибірку даних.

1. *Запити на вибірку даних.*
2. *Створення запитів за допомогою конструктора.*

1. Запити на вибірку даних

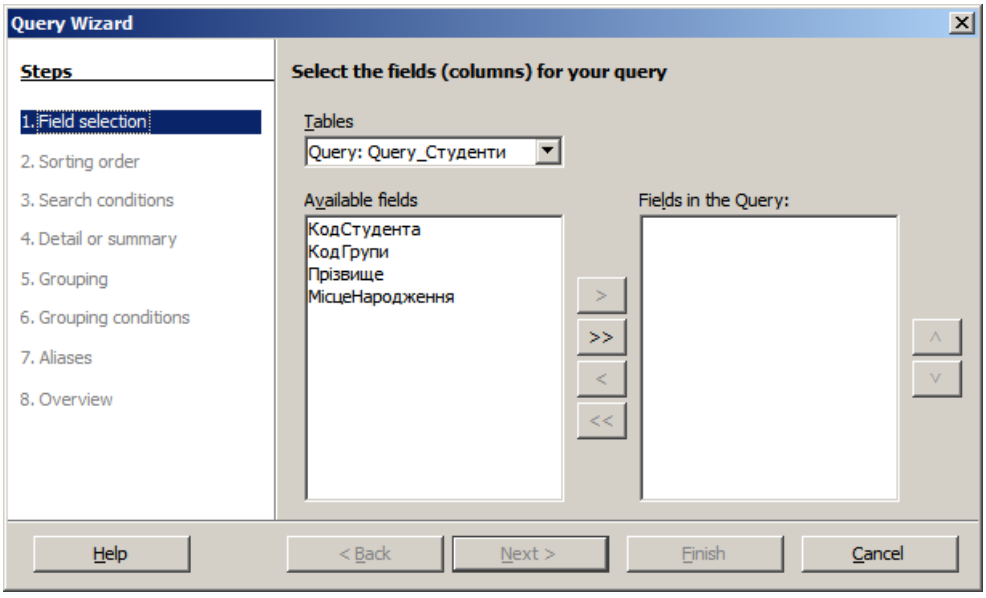
Запит (query) – це засіб вибору необхідної інформації з бази даних. Питання, сформоване для пошуку даних з бази даних, і є запитом. Застосовуються два типи запитів: за зразком (QBE – Query by example) і структурована мова запитів (SQL – Structured Query Language).

QBE - запит за зразком – засіб для відшукання необхідної інформації в базі даних. Він формується не на спеціальній мові, а шляхом заповнення бланка запиту у вікні Конструктора запитів.

SQL – запити – це запити, які складаються (програмістами) з послідовності SQL – інструкцій. Ці інструкції задають необхідні дії з вхідним набором даних для генерації вихідного набору. Всі запити Base будує на основі SQL – запитів.

Існує декілька типів запитів. Найбільш поширеним є запит на вибірку.

Створення запиту на вибірку за допомогою Майстра. Майстри різних об’єктів БД дозволяють автоматизувати процес їх створення і звести його до комфортного діалогу з користувачем. У вікні бази даних вибрати вкладку Запити і двічі клацнути на піктограмі Створення запиту за допомогою майстра, з'явиться вікно.



У вікні майстра вибрати необхідну таблицю (таблицю-джерело) з опції Таблиці і запити і вибрати поля даних. Якщо запит формується на основі кількох таблиць, необхідно повторити дії для кожної таблиці-джерела.

Потім, на одному із кроків роботи Майстра треба задати умови, за якими буде здійснюватись вибірка.

На останньому кроці отримається готовий запит.

2. Створення запиту на вибірку за допомогою Конструктора

За допомогою конструктора можна створити наступні види запитів:

- Простий
- За умовою
- Параметричний
- Підсумковий
- З обчислювальними полями

Щоб викликати Конструктор запитів, необхідно перейти у вікно бази даних. У вікні бази даних необхідно вибрати вкладку Запити і двічі клацнути на піктограмі Створення запиту в режимі конструктора. З'явиться вікно додавання таблиць на фоні неактивного вікна конструктора запитів.

У вікні додавання таблиць слід вибрати таблицю-джерело або кілька таблиць, на основі яких проводитиметься вибір даних, і клацнути на кнопці Додати. Після цього закрити вікно додавання таблиць, вікно конструктора запитів стане активним.

Вікно Конструктора складається з двох частин – верхньої і нижньої. У верхній частині вікна розміщується схема даних запиту, яка містить список таблиць-джерел і відображає зв'язки між ними.

У нижній частині вікна знаходиться Бланк побудови запиту QBE (Query by Example), в якому кожен рядок виконує певну функцію:

Field (Поле) – вказує імена полів, які беруть участь в запиті

Alias (Псевдонім) – дозволяє іменам полів присвоювати псевдоніми

Table (Таблиця) – ім'я таблиці, з якої вибране це поле

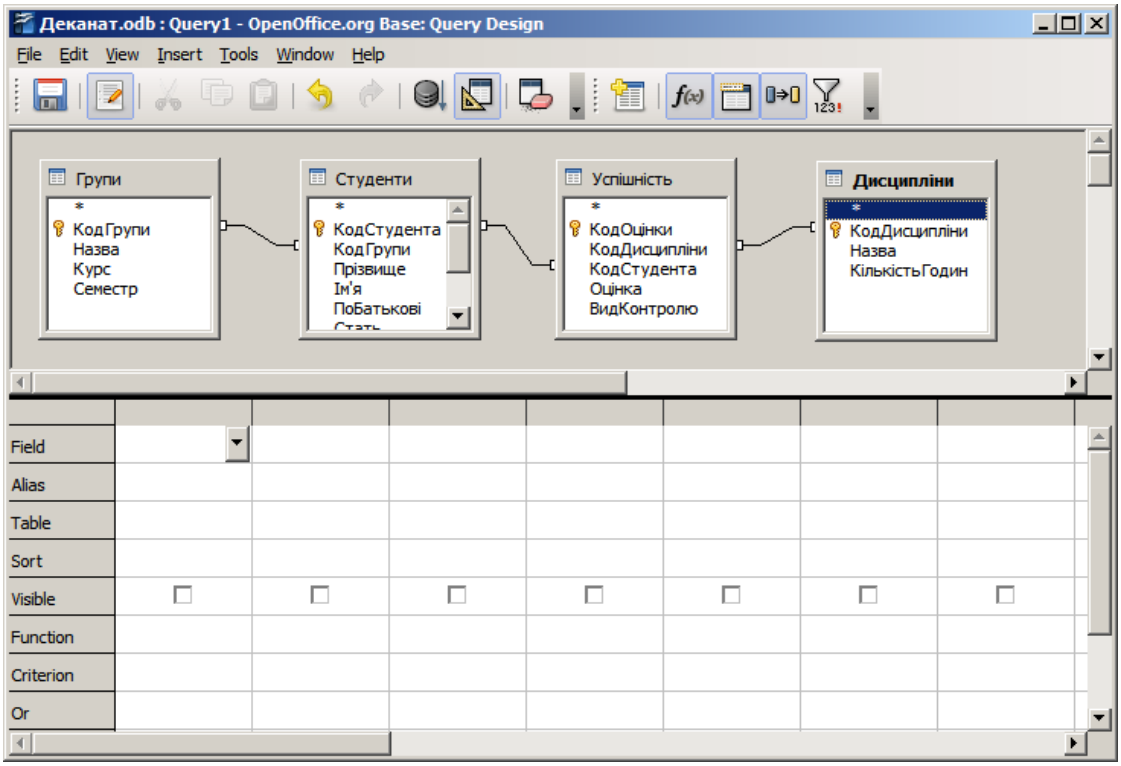
Sort (Сортування) –вказує тип сортування

Visible (Видимий) – встановлює прапорець проглядання поля на екрані

Function (Функції) – задаються функції для обчислень над відібраними даними (Сума, Кількість і т.д.)

Criterion (Умови відбору) – задаються критерії пошуку

Or (Або) – задаються додаткові критерії відбору



Формування запиту:

Вибрати таблицю-джерело, з якої проводиться вибірка записів.

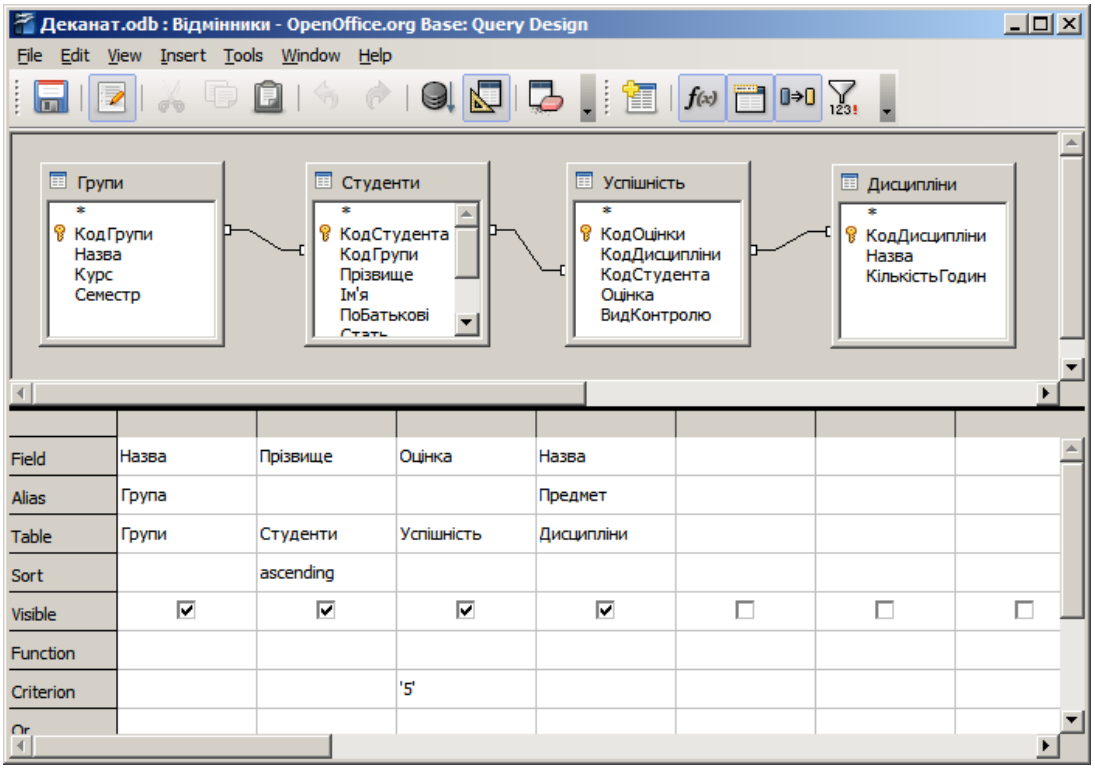
Перемістити імена полів з джерела в Бланк запиту. Наприклад, з таблиці Групи перетягнути поле Назва в перше поле Бланку запитів, з таблиці Студенти відбуксирувати поле Прізвища в друге поле Бланку запитів, а з таблиці Успішність відбуксирувати поле Оцінка в третє поле і з таблиці Дисципліни відбуксирувати поле Назва в четверте поле Бланка запитів.

Задати принцип сортування. Курсор миші перемістити в рядок Сортування для будь-якого поля, з'явиться кнопка відкриття списку режимів сортування: за збільшенням і за зменшенням. Наприклад, встановити в полі Прізвище режим сортування – за збільшенням.

У рядку виведення на екран автоматично встановлюється прапорець проглядання знайденої інформації в полі.

У рядку Умови відбору і рядку Або необхідно ввести умови обмежня пошуку – критерії пошуку. Наприклад, в полі Оцінка ввести 5, тобто відображати всі прізвища студентів, які отримали відмінні оцінки.

Після завершення формування запиту закрити вікно конструктора. Відкриється вікно діалогу Зберегти, в якому надати ім’я створеному запиту. Наприклад, Відмінники.



Щоб відкрити запит з вікна бази даних, необхідно виділити ім'я запиту і клацнути кнопку Відкрити, на екрані з'явиться вікно з результатами роботи запиту. Щоб внести зміни до запиту, його необхідно відкрити у режимі конструктора.

Запити, які є варіантами базового запиту і трохи відрізняються один від одного, називаються параметричними. У параметричному запиті вказується критерій, який може змінюватися за бажанням користувача.

Послідовність створення параметричного запиту:

- Створити запит в режимі конструктора або відкрити існуючий запит в режимі конструктора.
- У бланк запиту в рядку Умови відбору ввести умову відбору у вигляді запрошення після двокрапки, наприклад, :Прізвище.
- Виконати запит. У вікні діалогу, що з'явилося на екрані, «Введіть значення параметра» треба ввести, наприклад, прізвище студента, інформацію про успішність якого необхідно отримати.

Лекція 9 Системи керування базами даних. Мова структурованих запитів (SQL).

1. *Основні типи запитів на SQL.*
2. *Конструкція SQL-запитів на вибірку даних.*
3. *Операція JOIN — об'єднання таблиць*
4. *Вкладені підзапити*
5. *Додаткові засоби мови SQL*

1. Основні типи запитів на SQL.

Команди мови SQL за функціями можна поділити на наступні групи:

DQL (Data Query Language) — мова запиту даних, яка використовується для пошуку та вибору даних із БД та їх відображення **(SELECT);**

DDL (Data Definition Language) **– мова** опису даних, яка включає оператори, що описують структуру таблиць і режим відображення даних, команди для зміни структури таблиць, створення та вилучення індексів **(CREATE TABLE, CREATE VIEW, тощо);**

DML (Data Manipulation Language) – мова маніпулювання даними, команди якої дозволяють додавати та вилучати рядки з таблиць, вносити зміни значень полів **(INSERT, DELETE, UPDATE);**

TPL (Transaction Processing Language) – мова обробки транзакцій, що включає команди, які об'єднують декілька команд DML;

CCL (Cursor Control Language) – мова управління курсором, яка містить команди, що дозволяють виділити для обробки фрагмент результатного набору записів;

DCL (Data Control Language) – мова управління даними, яка забезпечує виконання адміністративних функцій надання та відміни прав доступу до всієї бази, набору таблиць тощо.

2. Конструкція SQL-запитів на вибірку даних.

Оператор SELECT – дозволяє проводити вибірки даних з однієї чи декількох таблиць і перетворювати до потрібного вигляду отримані результати які, як правило, представляються у вигляді таблиці.

Синтаксис оператора:

```
SELECT
 [ ALL | DISTINCT ] * |
 <ім'я таблиці>.*     |
 [ <ім'я таблиці>.]<назва поля> [AS <псевдонім>] [, …] |
 <вираз> [AS <псевдонім>] [, …]
[ FROM
  <ім'я таблиці> [, …] ]
[WHERE …]
[GROUP BY …]
  [HAVING …]
[ORDER BY …] ;
```

ALL | **DISTINCT** – предикати, які використовуються для обмеження числа записів, що повертаються. За замовчуванням використовується **ALL** (повертає усі значення), якщо предикати відсутні. **DISTINCT** – виключає записи, що містять значення, які повторюються у всіх вибраних полях. Впливає на результат тоді, коли у запиті аналізуються не всі поля з таблиці. Символ **«*»** означає, що вибрані всі поля заданої таблиці або таблиць. Параметр **<ім'я таблиці>** – назва таблиці, з якої потрібно вибрати записи, **<назва поля>** – назва поля, з якого вибирають дані; якщо ж задано декілька полів, то дані вибираються у визначеному порядку, **AS** – задає нову назву заголовка стовпця у запиті, **<псевдонім>** – назва, що буде заголовком стовпця замість початкового.

Даний оператор не змінює дані у БД, де відбувається вибір. У найпростішому випадку дію оператора SELECT можна сформулювати так: вибрати <елемент> із <джерела>.

Фраза ***FROM*** міститься у операторі SELECT і є обов'язковою у випадку вибірки даних хоча б з однієї чи кількох таблиць. Дана фраза може бути опущена у вибірках результатів із деяких виразів чи функцій, наприклад **SELECT 2+3;**

Порядок таблиць у виразі не важливий — більшість сучасних СУБД проводять попередній аналіз та планування оптимізації виконання запиту.

Синтаксис:

FROM <ім'я таблиці> [, …];

Тут <ім'я таблиці> – назва таблиці, з якої потрібно вибрати записи.

Для отримання переліку даних без повторень, використовують оператор **DISTINCT**.

В команді **SELECT** можуть міститись не тільки імена стовпців таблиці чи таблиць, але і обчислювальні вирази.

Фраза ***WHERE*** є необов'язковою. Дана фраза визначає умову вибірки результуючих рядків. Якщо фраза WHERE не включена у запит, то результатом будуть усі результуючі рядки. Якщо в запиті використовуються декілька таблиць і фраза WHERE відсутня, то результатом буде скалярний добуток таблиць.

Синтаксис:

[WHERE <умова>]

де **<умова>** – умова або умови вибору значень із поля чи полів які включають операції відношення, логічні операції, функції тощо.

Приклади використання функцій **between** і **in** у запитах.

Select * from Students where Age between 17 and 19;

Select * from Students where Age in (17,19);

Фраза ***GROUP BY*** є необов'язковою. Використовується при формуванні підсумкових запитів і об'єднує записи з однаковими значеннями у вказаному списку полів в один запис. Дозволяє застосовувати агрегатні функції до кожної групи, яка визначається загальним значенням поля або полів, вказаних в цій фразі.

Синтаксис:

[GROUP BY [<таблиця>.]<назва поля> [,…]]

Параметр <таблиця> можна не використовувати у тих випадках, коли користувач працює з однією таблицею.

Параметр <назва поля> – поле, по якому відбувається групування. Йому може передувати ім'я таблиці, якщо вибір виконується більше ніж з однієї таблиці.

Фраза ***HAVING*** є необов'язковою. Після того, як записи будуть згруповані за допомогою **GROUP BY**, фраза **HAVING** відбере ті значення з отриманих записів, що задовольнять умовам вибірки, які вказані у **HAVING.**

Синтаксис:

[HAVING <умова>]

Параметр **<умова>** – умова або умови відбору значень із отриманих записів.

Фраза ***ORDER BY*** є необов'язковою. Дана фраза сортує результуючі записи, що отримані після виконання запиту за зростанням або спаданням указаних полів чи поля. Його необхідно вказувати для сортування результату запиту. За замовчуванням задано порядок сортування за зростанням.

Синтаксис:

[ORDER BY { <назва поля> [ASC | DESC] } [,…]]

<назва поля> – поле, за яким відбувається сортування. **ASC (DESC)** – зареєстровані слова для розташування елементів у зростаючому (спадному) порядку.

3. Операція JOIN — об'єднання таблиць

Для задання об'єднання кількох таблиць у фразі **FROM** виконують операцію **JOIN.** Якщо в результат вибірки необхідно включити всі рядки з обох таблиць, що задовольняють умові вибірки, використовується операція **INNER JOIN**.

Синтаксис:

```
FROM
 <таблиця1> [AS <,псевдонім>]
 [ INNER | LEFT | RIGHT | FULL ] JOIN
 <таблиця2> [AS ,псевдонім>]
 ON
 {<таблиця1>.<поле1>} <операція> {<таблиця2>.<поле2>}
```

Тут **<таблиця1>**, **<таблиця2>** – імена таблиць, записи яких підлягають об'єднанню; **<поле1>**, **<поле2>** – імена полів, що об'єднуються; якщо поля не є числовими, то повинні мати однаковий тип даних і містити дані одного виду; поля можуть мати різні імена, **<операція>** – будь-яка операція порівняння.

Нехай дано дві таблиці T1 і T2:

T1

F1	***F2***
A	100
B	101
C	102
D	103

T2

F3	***F4***
B	200
C	201
F	202
G	203

Задамо запит:

```
SELECT *
FROM T1 join T2 on T1.F1=T2.F3 ;
```

Результатом такого запиту буде таблиця T12:

Таблиця T12

F1	***F2***	***F3***	***F4***
B	101	B	200
C	102	C	201

Операція **LEFT JOIN** використовується для створення лівого зовнішнього об'єднання, при якому всі записи з першої (лівої) таблиці включаються в динамічний набір, навіть якщо в другій (правій) таблиці немає відповідних до них записів. Якщо у лругій таблиці, з якою виконується з'єднання, не має відповідних рядків, то замість значень її полів додається значення Null.

Задамо запит:

```
SELECT *
FROM  T1 left join T2 on T1.F1=T2.F3
```

Результат – табл.T13

Таблиця Т13

F1	***F2***	***F3***	***F4***
A	100	Null	null
B	101	B	201
C	102	C	202
D	103	Null	null

Операція **RIGHT JOIN** використовується для створення правого зовнішнього об'єднання, при якому всі записи з другої (правої) таблиці включаються в динамічний набір, навіть якщо в першій (лівій) таблиці немає відповідних до них записів. Якщо у першій таблиці, з якою виконується з'єднання, не має відповідних рядків, то замість значень її полів додається значення Null.

Операції JOIN можуть бути вкладеними. Синтаксис:

```
SELECT <поля>
FROM
 <таблиця1> INNER JOIN ( <таблиця2> INNER JOIN
  [ ( ] <таблиця3> [INNER JOIN
   [ ( ] <таблицяX> [INNER JOIN…)]
   [ ON <таблиця3>.<поле3> <операція> <таблицяX>.<полеX> ]
  [ ON <таблиця2>.<поле2> <операція> <таблиця3>.<поле3>) ]
 ON <таблиця1>.<поле1> <операція> <таблиця2>.<поле2>;
```

4. Вкладені підзапити

SQL дозволяє використовувати одні запити всередені інших запитів, тобто вкладати один запит у інший. При цьому верхнього рівня оператор **SELECT** використовує результат внутрішнього оператора **SELECT** для визначення отримання кінцевого результату всього запиту. Внутрішні оператори можуть бути розміщені у реченнях **WHERE** і **HAVING** і в такому разі отримують назву підзапитів, або вкладених запитів. Крім того, внутрішні оператори **SELECT** можуть використовуватись в операторах **INSERT**, **UPDATE** і **DELETE**.

Існує три типи підзапитів, а саме:

- ***скалярний підзапит*** який повертає значення вибране із перетину одного стовпця і одного запису чи результату виразу;
- ***рядковий підзапит*** який повертає значення декількох стовпців таблиці у вигляді одного рядка;
- ***табличний підзапит*** який повертає значення одного чи більше стовпців таблиці розміщених у більше ніж одному рядку.

При побудові підзапитів потрібно дотримуватись наступних правил і обмежень:

У підзапитах не повинна використовуватись фраза ORDER BY, хоча вона може бути присутня у зовнішньому запиті.

Список у визначенні SELECT підзапиту повинні складатись із імен окремих стовпців або складених із них виразів за виключенням того випадку, коли у підзапиті

використовується ключове слово EXIST. Дане слово використовується тільки разом із підзапитом і результатом його дії є логічне значення True чи False. Істина буде в тому випадку коли у результуючій таблиці підзапиту присутній хоча би один рядок.

За замовчуванням імена стовпців у підзапиті відносяться до таблиці, ім'я якої вказано у фразі FROM.

Якщо підзапит є одним із двох операндів, які беруть участь в операції порівняння, то підзапит повинен бути вказаний у правій частині даної операції.

У вкладених підзапитах використовується предикат IN (Select … From … Where … IN).

У підзапитах, які повертають один стовпець числових значень, можуть використовуватись ключові слова ALL і ANY.

Якщо підзапиту буде передувати ключове ключове слово ALL, то умова порівняння вважається виконаною тільки в тому випадку, якщо воно виконується для всіх значень у результуючому стовпці підзапиту.

Якщо запиту, підзапиту передує ключове слово ANY, то умова порівняння буде вважатись виконаною, якщо вона виконується хоча би для одного із значень у результуючому стовпці підзапиту. Якщо в результаті виконання підзапиту буде отримано порожнє значення, то для слова ALL умова порівняння буде вважатись виконаною, а для ANY ні.

У мові SQL можна використовувати стандартні операції над множинами, а саме: об'єднання, перетин і різниця, які дозволяють комбінувати результати виконання двох і більше запитів в одну результуючу таблицю.

На таблиці, які можуть комбінуватись з допомогою операцій над множинами накладаються певні обмеження, а саме:

- вони повинні мати одну і ту ж структуру, тобто одну і ту ж кількість стовпців;
- у відповідних стовпцях повинні міститись дані однакового типу і довжини.

Операція UNION. Дана операція створює запит на об'єднання, що поєднує результати кількох незалежних запитів.

Синтаксис:

<запит1> [UNION [ALL] <запит2> […]];
<запит1>, <запит2> – інструкція SELECT.
Всі запити, включені в операцію UNION, повинні відбирати однакове число полів.

Приклад 13. Об'єднати два запити, один з яких вибирає прізвище та ім'я із таблиці Students, а другий із відповідних двох полів P1і P2 із таблиці NTab:

```
SELECT Surname, Name
FROM  Students
UNION SELECT P1, P2
FROM  Ntab;
```

5. Додаткові засоби мови SQL

До них можна віднести:

- системи забезпечення безпеки, що попереджує несанкціонований доступ до БД зі сторони користувачів;
- системи підтримки цілісності даних, що забезпечує непротирічивий стан збереження даних;
- системи управління паралельною роботою додатків, які контролюють процеси їх сумісного доступу до БД;
- системи відновлення, які дозволяють відновити БД до попереднього непротирічивого стану, який був порушений в результаті збою апаратного чи програмного забезпечення;
- доступного користувачам каталогу, який містить опис інформації, що зберігається в БД.

При роботі з БД кінцевим користувачам не важливо, наскільки проста чи складна внутрішня організація системи, оскільки знання всієї структури БД і самих даних які в ній зберігаються може тільки ускладнити роботу з ними за рахунок їх надлишковості (тобто даних які зберігаються в БД, але які не потрібні конкретному користувачеві в процесі його роботи). Для розв'язання такої проблеми в СУБД присутній механізм створення **представлень (view),** які дозволяють довільному користувачеві мати свій власний погляд на БД. Мова DDL містить засоби визначення представлень, кожне із яких є деякою підмножиною БД.

Крім спрощення роботи за рахунок надання користувачам тільки дійсно потрібних даних, представлення володіють, зокрема, такими можливостями:

- забезпечують додатковий рівень безпеки даних за рахунок вилучення тих даних, які не повинні бачити окремі користувачі.

- представляють механізм налагодження зовнішнього інтерфейсу БД.

- дозволяють зберігати зовнішній інтерфейс БД непротирічивим і незмінним навіть при внесенні змін у її структуру. Наприклад якщо у таблиці додаються поля, які не використовуються у жодному представленні, то всі ці зміни ніяк не відобразяться на цих представленнях. Тим самим представлення забезпечує повну незалежність програм від реальної структури даних.

В реляційній моделі подання даних слово **представлення** характеризує не всю зовнішню модель, але тільки деяке використовуване ним **віртуальне відношення (virtual relation)**, тобто іншими словами відношення, яке насправді саме по собі не існує, але яке динамічно будуються на основі одного чи декількох базових відношень, які реально існують в БД. Таким чином, зовнішня модель може складатись одночасно як із базових (на концептуальному рівні) відношень, так і з представлень, які створені на основі базових відношень.

Лекція 10. Системи керування базами даних. Форми та звіти.

1. *Побудова форм за допомогою майстра форм. Переміщення по формі і робота з даними. Елементи управління.*
2. *Створення звітів за допомогою майстра звітів та конструктора звітів.*

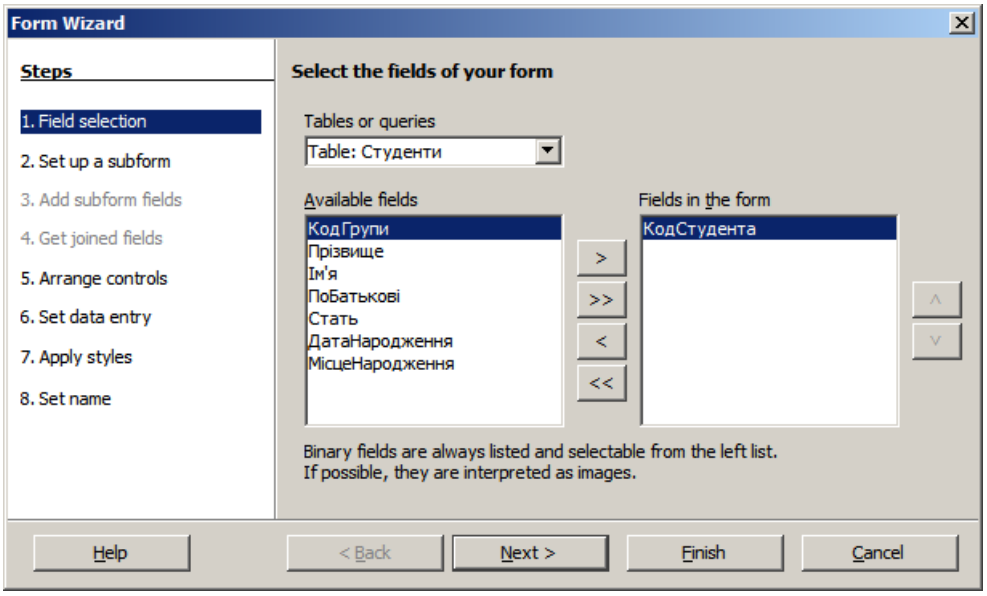
1. Побудова форм за допомогою майстра форм. Переміщення по формі і робота з даними. Елементи управління.

Форма в БД – це структуроване вікно. Форма є засобом розробки призначеного для користувача інтерфейсу. Зовнішній вигляд форми вибирається залежно до того, з якою метою вона створюється. Джерелом даних для форми є записи таблиці або запиту.

Форма надає можливості для:

- введення і перегляду інформації бази даних,
- зміни даних,
- друку.

Створення форми за допомогою Майстра. У вікні бази даних вибрати вкладку **Forms (Форми)**, а в правій частині вибрати режим **UseWizard to Create Form… (Використати майстер для створення форм…)** і запустити Майстер форм.



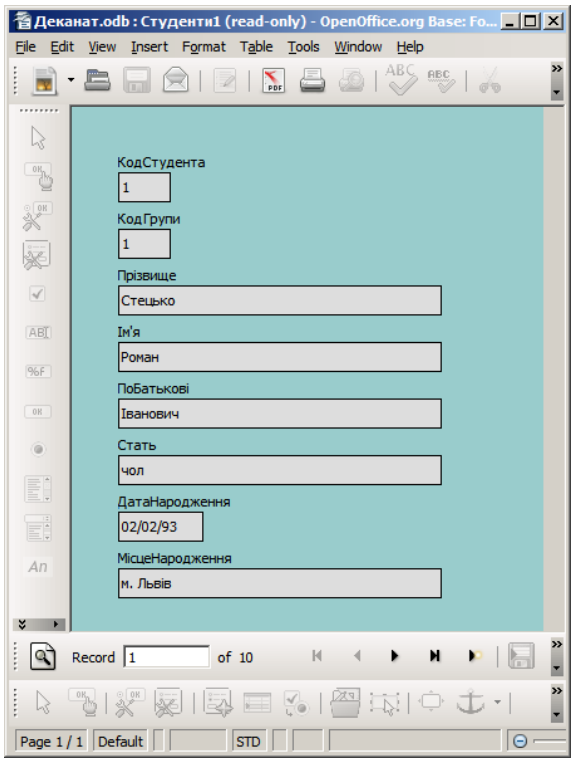
На першому етапі потрібно вибрати таблицю і поля, які будуть відображатись у формі. На наступному кроці Майстер пропонує визначити наявність субформ. Наступний крок – компонування форми. Можна вибрати одну з чотирьох компоновок:

- Columnar – Labels Left (Стовпчикова – підписи зліва).

- Columnar – Labels on Yop (Стовпчикова – підписи зверху)
- As Data Sheet (Таблична)
- In Blocks – Labels above (Блочна – підписи зверху)

На передостанньому кроці треба вибрати зовнішній вигляд форми.

У останньому вікні Майстра потрібно ввести ім'я форми і вказати подальші дії: Відкрити форму для перегляду і введення даних; Змінити макет форми.



2. Створення звіту як об'єкту бази даних

Звіт – це форматоване представлення даних, яке виводиться на екран, в друк або файл. Він дозволяють вибирати з бази потрібні відомості і представляти їх у вигляді, зручному для сприйняття, а також надає широкі можливості для узагальнення і аналізу даних.

При друці таблиць і запитів інформація видається практично в тому вигляді, в якому зберігається. Часто виникає необхідність представити дані у вигляді звітів, які мають традиційний вигляд і легко читаються. Докладний звіт включає всю інформацію з таблиці або запиту, але містить заголовки і розбитий на сторінки зі вказівкою верхніх і нижніх колонтитулів.

Структура звіту в режимі Конструктора. В звіті відображаються дані з запиту або таблиці, але до них можна додавати текстові елементи, які спрощують його сприйняття.

До таких *елементів* належать:

- **Заголовок**. Цей розділ друкується лише у верхній частині першої сторінки звіту. Використовується для виведення даних, таких як текст заголовка звіту, дата або констатуюча частина тексту документа, які слід надрукувати один раз на початку звіту.
- **Верхній колонтитул**. Використовується для виведення даних, таких як заголовки стовпців, дати або номери сторінок, що друкуються зверху на кожній сторінці звіту.
- **Область даних**, розташована між верхнім і нижнім колонтитулами сторінки. Містить основний текст звіту. У цьому розділі розміщені дані, що роздруковуються для кожного з тих записів в таблиці або запиті, на яких заснований звіт. Для розміщення в області даних елементів управління використовують список полів і панель елементів.

- **Нижній колонтитул**. Цей розділ розміщений в нижній частині кожної сторінки. Використовується для виведення даних, таких як підсумкові значення, дати або номери сторінки, що друкуються знизу на кожній сторінці звіту.
- **Примітка**. Використовується для виведення даних, таких як текст висновку, загальні підсумкові значення або підпис, які слід надрукувати один раз в кінці звіту.

Способи створення звіту. У Base можна створювати звіти такими способами:

- Конструктор
- Майстер звітів

Майстер дозволяє створювати звіти з групуванням записів і є простим способом створення звітів. Він поміщає вибрані поля в звіт і пропонує кілька стилів його оформлення. Після завершення роботи Майстра отриманий звіт можна доопрацьовувати в режимі Конструктора.

Звіти служать для роздруківки необхідної інформації з бази даних. Цим вони схожі на запити. Звіти створюються на основі таблиць бази даних або запитів. Вони можуть містити всі поля таблиці або запиту, або тільки обрані групи полів. Звіти можуть бути статичними або динамічними. Статичні звіти містять дані з обраних полів, що існують у них на момент створення звіту. Динамічні звіти можуть змінюватися, щоб показувати самі останні дані.

Створимо динамічний звіт про річниці весіль у заданому місяці. Запит Запит_Весілля буде основою для нашого звіту: Річниці весіль місяця. Редагуючи запит для заданого місяця, ми внесемо в нього зміни й одночасно запам'ятаємо їх у звіті.

Крок 1: Виклик майстра створення звіту одним із двох способів.

- Натисніть значок Звіти у вікні Information - OpenOffice.org Base і потім клацніть по рядку Використати майстер для створення звіту. або
- Клацніть правою кнопкою по запиті або таблиці й виберіть із контекстного меню **Майстер звітів**.

Крок 2: Майстер звітів

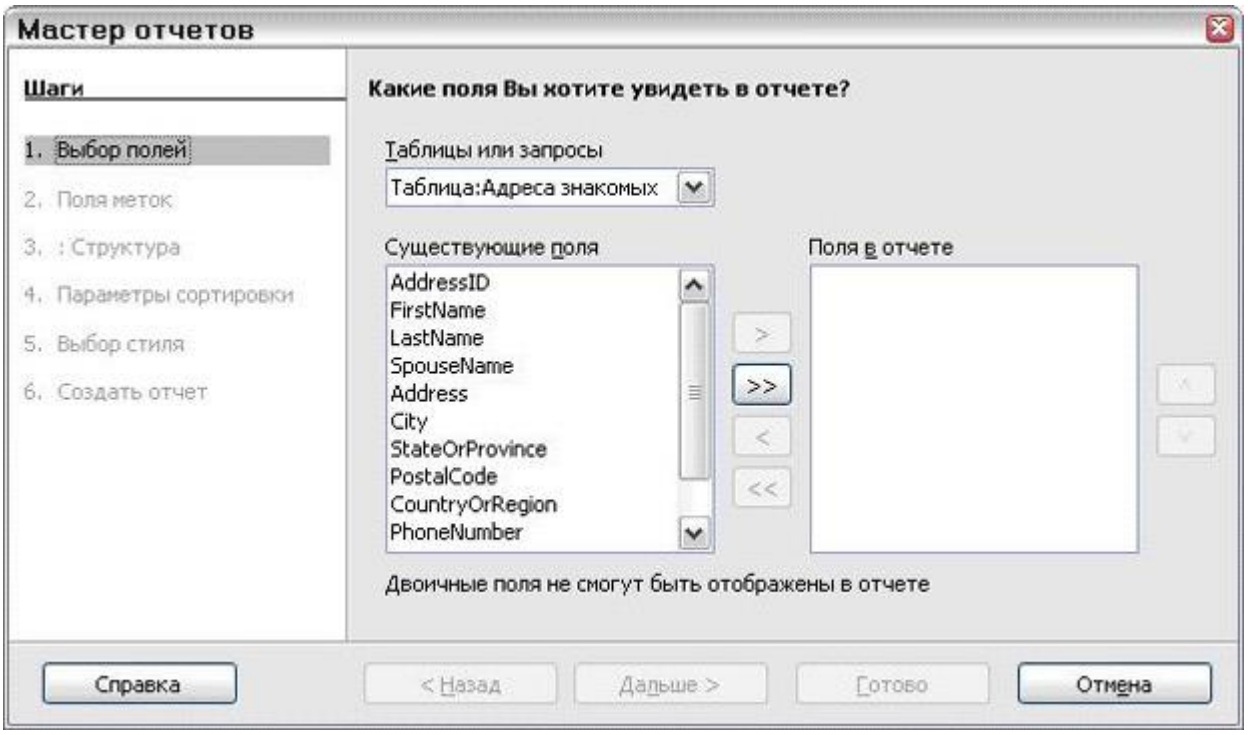
1. У списку, що випадає, Таблиці або запити виберіть запит: Запит_Весілля. ¶- За допомогою кнопки з подвійною стрілкою (>) перемістіть всі поля зі списку Існуючі поля в список Поля у звіті.¶- Натисніть кнопку **Далі**.

2. Змініть написи для деяких полів міток.

- - Для написів, що складаються більш ніж з одного слова, помістіть пробіл між словами, а також переведіть їх на рідну мову. (Наприклад, FirstName стане Ім'я, LastName стане Прізвище й CountryOrRegion стане Країна.)
- - Змініте PostalCode на Поштовий код, WedDate на Місяць, WedDate на День й WedDate на Рік.
- - Натисніть кнопку Далі.

3. Структура. Згрупуємо записи звіту по полю LastName.

- - Клацніть по рядку Прізвище в списку Поля й за допомогою кнопки зі стрілкою (>) перемістіть її в список Угруповання.
- - Натисніть кнопку **Далі**.



Перша сторінка майстра звітів

4. Вибір стилю: Використаємо установки за замовчуванням. Вони передбачають альбомну орієнтацію в нижній частині **Майстра звітів.** Натисніть кнопку **Далі**.

5. Створити звіт:

- Назвіть звіт Звіт_Весілля.
- Який тип звіту ви хочете створити? Виберіть Динамічний звіт.
- Що Ви збираєтеся робити після створення звіту? Виберіть **Модифікувати шаблон звіту**.
- Натисніть кнопку **Готово**.

6. Модифікація звіту. Звіт містить таблицю з інформацією із Запиту. Він може містити дивні слова. Ми змінимо вертикальне вирівнювання другого рядка.

- Клацніть по осередку, розташованої нижче напису Ім'я, і переміщайте курсор вправо, щоб виділити другий рядок.
- Клацніть правою кнопкою в будь-якому місці виділеного осередку. Виберіть **Осередок** > по центрі для установки правильного вирівнювання.
- Якщо хочете, ви можете зараз змінити ширину будь-якого осередку.
- Виберіть **Файл > Зберегти** й потім **Файл > Закрити** у вікні Звіт_Весілля - OpenOffice.org Writer.

Лекція 11. Мережні технології. Огляд комп'ютерних мереж. Пошук інформації в Internet.

1. *Огляд комп'ютерних мереж.*
2. *Технологія роботи в локальній мережі.*
3. *Основні послуги глобальної мережі.*
4. *Види ресурсів Internet.*
5. *Пошук інформації в Internet.*

1. Огляд комп'ютерних мереж.

Комп'ютерна мережа – це сукупність комп'ютерів, які можуть здійснювати інформаційну взаємодію один з одним за допомогою комунікаційного обладнання і програмного забезпечення.

Комп'ютерні мережі створюються з метою доступу до загальносистемних ресурсів (інформаційних, програмних і апаратних), розподілених (децентралізованих) в цій мережі. За територіальною ознакою розрізняють мережі локальні і територіальні (регіональні і глобальні).

Слід розрізняти комп'ютерні і термінальні мережі. Комп'ютерні мережі зв'язують комп'ютери, кожен з яких може працювати і автономно. Термінальні мережі зазвичай пов'язують потужні комп'ютери (мейнфрейми) з терміналами (пристроями введення-виведення інформації). Прикладом термінальних пристроїв і мереж може служити мережа банкоматів або кас продажу квитків.

Комп'ютерну мережу можна уявляти як багатошарову модель, що складається з шарів:

- комп'ютери;
- комунікаційне обладнання;
- операційні системи;
- мережеві додатки.

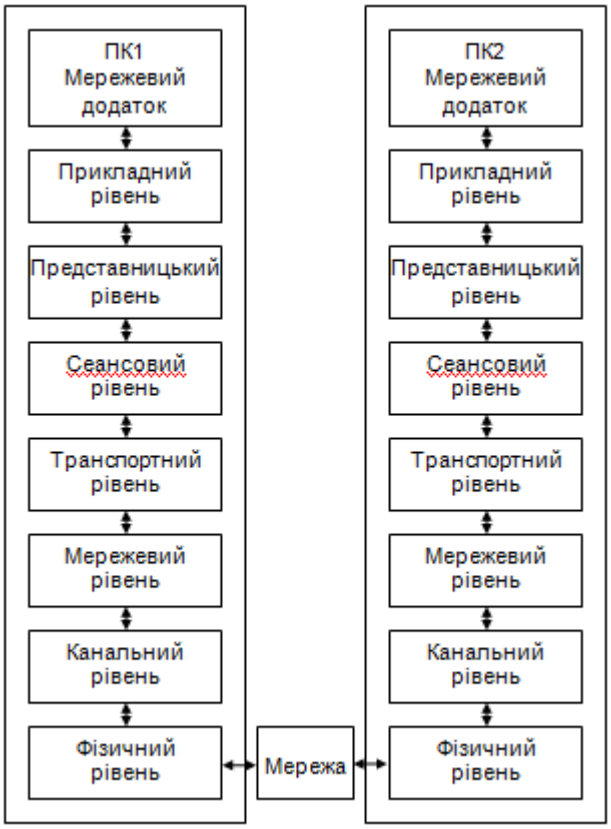
У комп'ютерних мережах використовуються різні типи і класи комп'ютерів. Комп'ютери і їх характеристики визначають можливості комп'ютерних мереж.

До комунікаційного устаткування відносяться: модеми, мережеві карти, мережеві кабелі і проміжна апаратура мереж. До проміжної апаратури відносяться: прийомопередавачі або трансивери (transceivers), повторювачі або репітери (repeaters), концентратори (hubs), мости (bridges), комутатори, маршрутизатори (routers), шлюзи (gateways).

Для забезпечення взаємодії різних програмних і апаратних засобів в комп'ютерних мережах були прийняті єдині правила або стандарт, який визначає алгоритм передачі інформації в мережах.

Як стандарт були прийняті мережеві протоколи, які визначають взаємодію устаткування в мережах. Слід зазначити, що в обчислювальних мережах здійснюється обмін даними не лише між вузлами як фізичними пристроями, але і між програмними модулями.

Розглянемо функції, що виконуються кожним функціональним рівнем семирівневої моделі взаємодії відкритих систем при передачі пакету даних від мережевого додатка одного комп'ютера до мережевого додатка, що працює на іншому комп'ютері.



Механізм передачі повідомлення між Пк1 і Пк2 можна представити у вигляді послідовної пересилки цього повідомлення зверху вниз від прикладного рівня до фізичного рівня. Потім фізичний рівень Пк1 забезпечує пересилку повідомлення (даних) по мережі фізичному рівню Пк2. Далі повідомлення передається знизу доверху від фізичного рівня до прикладного рівня Пк2.

- Прикладний рівень – управляє спільним доступом до мережі, потоком даних і обробкою помилок. Прикладний рівень отримує запит (повідомлення) від мережевого додатка, що працює на комп'ютері Пк1, який потрібно передати мережевому додатку, що працює на Пк2.
- Представницький рівень визначає формат, який використовується для обміну даними між Пк1 і Пк2. На Пк1 дані, що поступили від прикладного рівня, на представницькому рівні переводяться в проміжний формат.
- Сеансовий рівень дозволяє двом додаткам на Пк1 і Пк2 встановлювати, використовувати і завершувати з'єднання, яке називається сеансом.
- Транспортний рівень здійснює контроль даних і гарантує доставку пакетів без помилок.
- Мережевий рівень служить для утворення єдиної транспортної системи, яка об'єднує декілька мереж, що можуть мати різні принципи передачі повідомлень.
- Канальний рівень забезпечує пересилку пакетів між будь-якими двома ПК локальної мережі. Функції канального рівня реалізуються мережевими адаптерами і їх драйверами.
- Фізичний рівень забезпечує фізичний шлях для електричних сигналів, що несуть інформацію. Як правило, функції фізичного рівня реалізуються мережевим адаптером або портом.

В обчислювальних мережах, як правило, застосовуються набори протоколів, а не всі функціональні рівні моделі взаємодії відкритих систем. Набір протоколів, достатній для організації взаємодії обладнання в мережі, називається стеком комунікаційних протоколів. Найбільш популярним є стек TCP/IP. Цей стек використовується для зв'язку комп'ютерів в мережі Internet і в корпоративних мережах.

Протоколи реалізуються автономними і мережевими операційними системами (комунікаційними засобами, які входять до ОС), а також пристроями телекомунікаційного обладнання (мостами, комутаторами, маршрутизаторами, шлюзами).

2. Технологія роботи в локальній мережі.

Сьогодні широко застосовуться LAN (Local Area Network), основне призначення яких – забезпечити доступ до загальномережевих (інформаційних, програмних і апаратних) ресурсів. Крім того, LAN дозволяють співробітникам підприємств оперативно обмінюватися один з одним інформацією. LAN застосовуються для вирішення таких проблеми:

- Розподіл та колективне використання даних;
- Розподіл та колективне використання ресурсів;
- Розподіл та колективне використання програм.

Локальна обчислювальна мережа (LAN) є з'єднанням кількох ПК за допомогою відповідного апаратного і програмного забезпечення. У локальних мережах швидкість передачі даних висока, протоколи порівняно з протоколами глобальних мереж відносно прості, відсутня надмірність каналів зв'язку.

Локальні мережі залежно від адміністративних взаємин між ЕОМ розділяються на:

- ієрархічні або централізовані;
- однорангові.

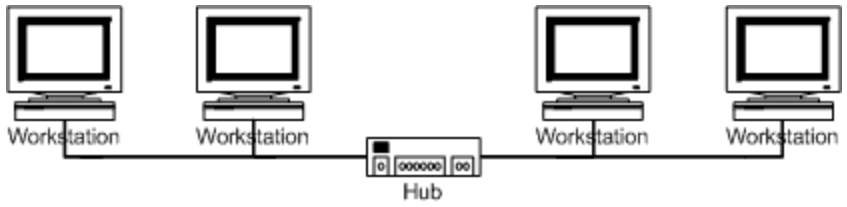
Локальні мережі залежно від фізичних і логічних взаємин між ЕОМ відрізняються архітектурою (Ethernet, TokenRing, FDDI і т.д.) і топологією (шинна, кільце, зірка і т.д.).

У локальних мережах реалізується технологія «клієнт-сервер». Сервер – це об'єкт (комп'ютер або програма), який надає сервісні послуги, а клієнт – це об'єкт (комп'ютер або програма), який просить сервер надати ці послуги.

В однорангових мережах сервер може бути одночасно і клієнтом, тобто використовувати ресурси іншого ПК або того ж ПК, якому він сам надає ресурси.

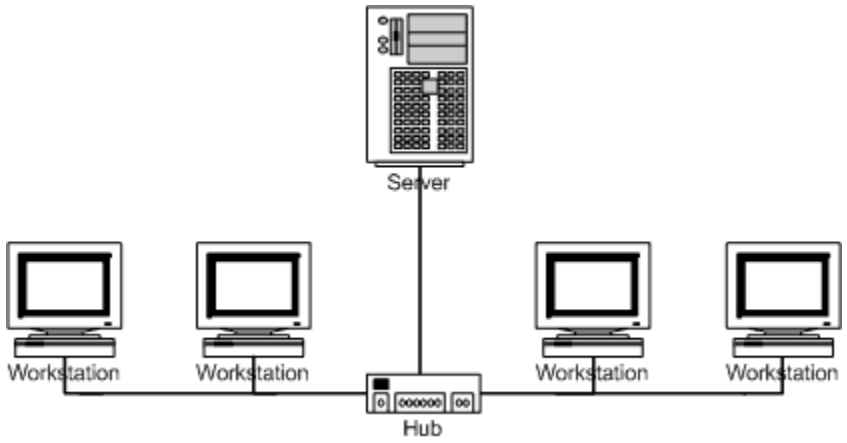
Сервер в ієрархічних мережах може бути клієнтом лише сервера більш високого рівня ієрархії. Ієрархічні мережі називаються мережами з виділеним сервером. Комп'ютери, що складають локальну мережу, прийнято називати вузлами. Кожен вузол може бути сервером або робочою станцією.

Однорангова мережа – це мережа рівноправних комп'ютерів (робочих станцій), кожен з яких має унікальне ім'я і пароль для входу в комп'ютер. Однорангова мережа не має центрального ПК.



У одноранговій мережі кожна робоча станція може розділити всі ресурси з іншими робочими станціями мережі. Робоча станція може розділити частину ресурсів, а може взагалі не надавати жодних ресурсів іншим станціям. Наприклад, деякі апаратні засоби (сканери, принтери, вінчестери, приводи CD-ROM та ін.), підключені до окремих ПК, використовуються спільно на всіх робочих місцях.

Ієрархічні локальні мережі – локальні мережі, в яких наявний один або декілька спеціальних комп'ютерів – серверів, на яких зберігається інформація, що спільно використовується різними користувачами. Ієрархічні локальні мережі – це, як правило, LAN з виділеним сервером, але існують мережі і з невиділеним сервером. У мережах з невиділеним сервером функції робочої станції і сервера поєднані. Робочі станції, що входять в ієрархічну мережу, можуть одночасно організувати між собою однорангову локальну мережу.



Виділені сервери зазвичай є високопродуктивними комп'ютерами, з вінчестерами великої ємності. На сервері встановлюється мережева операційна система, до нього підключаються всі зовнішні пристрої (принтери, сканери, жорсткі диски, модеми і так далі). Надання ресурсів сервера в ієрархічній мережі проводиться на рівні користувачів.

Кожен користувач має бути зареєстрований адміністратором мережі під унікальним ім'ям (логіном) і користувачі повинні призначити собі пароль, під яким входитимуть в ПК і мережу. Крім того, при реєстрації користувачів адміністратор мережі виділяє їм необхідні ресурси на сервері і права доступу до них. Комп'ютери, з яких здійснюється доступ до інформації на сервері, називаються робочими станціями, або клієнтами. На них встановлюється автономна операційна система і клієнтська частина мережевої операційної системи.

Залежно від способів використання сервера в ієрархічних LAN розрізняють сервери наступних типів:

- ***Файловий сервер.*** В цьому випадку на сервері знаходяться спільно оброблювані файли і спільно використовувані програми.
- ***Сервер баз даних.*** На сервері розміщується мережева база даних. База даних на сервері може поповнюватися з різних робочих станцій і видавати інформацію за запитами з робочих станцій.
- ***Сервер доступу*** – виділений комп'ютер в локальній мережі для виконання віддаленої обробки завдань. Сервер виконує завдання, отримане з віддаленої робочої станції, і результати направляє на віддалену робочу станцію. Іншими

словами, сервер призначений для віддаленого доступу (наприклад, з мобільного ПК) до ресурсів локальної мережі.

- ***Сервер-друк.*** До комп'ютера невеликої потужності підключається достатньо продуктивний принтер, на якому може бути роздрукована інформація відразу з декількох робочих станцій. Програмне забезпечення організовує чергу завдань на друк.
- ***Поштовий сервер.*** На сервері зберігається інформація, що відправляється і отримується як по локальній мережі, так і ззовні по модему. Користувач може проглянути інформацію, що поступила на його ім'я, або відправити через поштовий сервер свою інформацію.

До програмних компонентів мереж відносяться: операційні системи і мережеві додатки або мережеві служби. Мережева операційна система – це основа будь-якої обчислювальної мережі. Мережева операційна система необхідна для управління потоками повідомлень між робочими станціями і серверами. Вона може дозволити будь-якій робочій станції працювати з мережевим диском або принтером, які фізично не підключені до цієї станції.

3. Основні послуги глобальної мережі.

Глобальні мережі Wide Area Networks (WAN), які відносяться до територіальних комп'ютерних мереж, призначені, як і локальні мережі, для надання послуг, проте значно більшій кількості користувачів, що знаходяться на великій території.

У мережах існує три принципово різні схеми комутації:

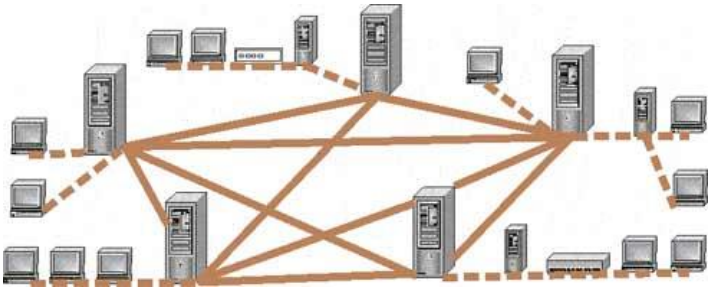
- Комутація каналів – процес, який за запитом здійснює з'єднання двох або більшої кількості станцій даних і забезпечує монопольне використання каналу передачі даних до тих пір, поки не станеться роз'єднання.
- Комутація повідомлень – процес пересилання даних, що включає прийом, зберігання, вибір початкового напряму і подальшу передачу повідомлень без порушення їх цілісності. Використовуються в тих випадках, коли не очікується негайної реакції на повідомлення. Повідомлення передаються між транзитними комп'ютерами мережі з тимчасовою буферизацією їх на дисках кожного комп'ютера. Повідомленнями називаються дані, що з'єднані смисловим змістом, мають певну структуру і придатні для обробки, пересилання або використання.
- Комутація пакетів – це комутація повідомлень, що представлені у вигляді пакетів, які адресуються, коли канал передачі даних зайнятий лише під час передачі пакету і по її завершенні звільняється для передачі інших пакетів.

Перевагою мереж комутації каналів є простота реалізації (утворення безперервного складеного фізичного каналу), а недоліком – низький коефіцієнт використання каналів, висока вартість передачі даних, підвищений час очікування інших користувачів. При комутації повідомлень передача даних (повідомлення) здійснюється після звільнення каналу, поки воно не дійде до адресата. Кожен сервер проводить прийом, перевірку, збір, маршрутизацію і передачу повідомлення. Пакетна комутація має на увазі обмін невеликими пакетами (частина повідомлення) фіксованої структури, які не дають можливості утворення черг у вузлах комутації. Переваги: швидке з'єднання, надійність, ефективність використання мережі.

Internet – всесвітня інформаційна комп'ютерна мережа, що є об'єднанням багатьох регіональних комп'ютерних мереж і комп'ютерів, що обмінюють один з одним інформацією по каналах суспільних телекомунікацій (виділених телефонних аналогових і цифрових лініях, оптичних каналах зв'язку і радіоканалах, зокрема супутникових лініях зв'язку). Інформація в Internet зберігається на серверах. Сервери мають свої адреси і управляються спеціалізованими програмами. Вони дозволяють пересилати пошту і файли, проводити пошук в базах даних і виконувати інші завдання.

Доступ окремих користувачів до інформаційних ресурсів Internet зазвичай здійснюється через провайдера або корпоративну мережу. Провайдер – постачальник мережевих послуг – особа або організація, яка надає послуги з підключення до комп'ютерних мереж. Як провайдер виступає деяка організація, що має модемний пул для з'єднання з клієнтами і виходу у всесвітню мережу.

Основними комірками глобальної мережі є локальні обчислювальні мережі. Якщо деяка локальна мережа безпосередньо підключена до глобальної, то і кожна робоча станція цієї мережі може бути підключена до неї. Існують також комп'ютери, які безпосередньо підключені до глобальної мережі. Вони називаються хост-комп'ютерами (host – господар). Хост – це будь-який комп'ютер, що є постійною частиною Internet, тобто з'єднаний по Internet–протоколу з іншим хостом, який в свою чергу, з'єднаний з іншим, і так далі.



Структура глобальної мережі Internet. Практично всі послуги Internet побудовані на принципі клієнт-сервер. Вся інформація в Інтернет зберігається на серверах. Обмін інформацією між серверами здійснюється по високошвидкісних каналах зв'язку або магістралях. Сервери, з'єднані високошвидкісними магістралями, складають базову частину мережі Інтернет.

Окремі користувачі підключаються до мережі через комп'ютери місцевих постачальників послуг Інтернету, Internet-провайдерів (Internet Service Provider – ISP), які мають постійне підключення до Інтернет. Регіональний провайдер підключається до крупнішого провайдера національного масштабу, що має вузли в різних містах країни. Мережі національних провайдерів об'єднуються в мережі транснаціональних провайдерів або провайдерів першого рівня. Об'єднані мережі провайдерів першого рівня складають глобальну мережу Internet.

Передача інформації в Інтернет забезпечується завдяки тому, що кожен комп'ютер в мережі має унікальну адресу (IP-адресу), а мережеві протоколи забезпечують взаємодію різнотипних комп'ютерів, що працюють під управлінням різних операційних систем.

В основному в Інтернет використовується сімейство мережевих протоколів (стек) TCP/IP. На канальному і фізичному рівні стек TCP/IP підтримує технологію Ethernet, FDDI і інші технології. Основою сімейства протоколів TCP/IP є мережевий рівень, представлений протоколом IP, а також різними протоколами маршрутизації. Цей рівень забезпечує переміщення пакетів в мережі і управляє їх машрутизацією. Розмір пакету, параметри передачі, контроль цілісності здійснюється на транспортному рівні TCP.

Прикладний рівень об'єднує всі служби, які система надає користувачеві. До основних прикладних протоколів відносяться: протокол віддаленого доступу telnet, протокол передачі файлів FTP, протокол передачі гіпертексту HTTP, протоколи електронної пошти: SMTP, POP, IMAP, MIME.

В даний час відомі наступні способи доступу в Інтернет:

- Dial-Up – комутований доступ по аналоговій телефонній мережі
- DSL (Digital Subscriber Line) – сімейство цифрових абонентських ліній, призначених для організації доступу по аналоговій телефонній мережі, використовуючи кабельний модем. Ця технологія (ADSL, VDSL, HDSL, ISDL, SDSL, SHDSL, RADSL під загальною назвою xDSL) забезпечує високошвидкісне з'єднання. Користувач дістає доступ в мережу Інтернет зі збереженням звичайної роботи телефонного зв'язку;
- ISDN – комутований доступ по цифровій телефонній мережі. Головна особливість використання ISDN – це висока швидкість передачі інформації в порівнянні з Dial-Up доступом. ;
- Доступ в Інтернет по виділених лініях (аналогових і цифрових). Доступ по виділеній лінії – це такий спосіб підключення до Інтернет, коли комп'ютер користувача з'єднаний з сервером провайдера за допомогою кабелю (витої пари) і це з'єднання є постійним, тобто некомутованим. В цьому головна відмінність від звичайного телефонного зв'язку. Швидкість передачі даних до 100 Мбіт/с.
- Доступ в Інтернет по локальній мережі (Fast Ethernet). Підключення здійснюється за допомогою мережевої карти (10/100 Мбіт/с) із швидкістю передачі даних до 1 Гбіт/с на магістральних ділянках і 100 Мбіт/сек для кінцевого користувача. Для підключення комп'ютера користувача до Інтернет в квартиру підводиться окремий кабель (вита пара), при цьому телефонна лінія завжди вільна.
- Супутниковий доступ в Інтернет або супутниковий Інтернет (DIRECPC, Europe Online). Супутниковий доступ в Інтернет буває двох видів - ассиметричний і симетричний. Обмін даними комп'ютера користувача зі супутником двосторонній. Запити від користувача передаються на сервер супутникового оператора через будь-яке доступне наземне підключення, а сервер передає дані користувачеві з супутника. Максимальна швидкість прийому даних до 52,5 Мбіт/с (реальна середня швидкість до 3 Мбіт/с).
- Доступ в Інтернет з використанням каналів кабельної телевізійної мережі, швидкість прийому даних від 2 до 56 Мб/сек. Кабельний Інтернет (“coax a home”). В даний час відомо дві архітектури передачі даних – симетрична і асиметрична архітектура. Крім того, існує два способи підключення: а) кабельний модем встановлюється окремо в кожній квартирі користувачів; б) кабельний модем встановлюється в будинку, де живе відразу декілька користувачів послуг Інтернету. Для підключення користувачів до спільного кабельного модему використовується локальна мережа і встановлюється спільне на всіх обладнання Ethernet.
- Безпровідні технології: WiFi ;WiMax; RadioEthernet; MMDS; LMDS; Мобільний GPRS-Інтернет; Мобільний CDMA-Internet, 3G-інтернет..
- В даний час для доступу в Internet застосовуються технології Home PNA (HPNA) і HomePlug. Доступ в Інтернет по виділених лініях Home PNA або HPNA (телефонних лініях) і доступ через побутову електричну мережу напругою 220 вольт (HomePlug, Plug — це штепсель).
- Технологія Home PNA застосовується в основному для організації домашньої мережі за допомогою мережевих адаптерів. Підключення до глобальної мережі можна здійснити за допомогою роутера через мережі спільного доступу. Крім того, технологія HPNA призначена для організації колективного доступу в Інтернет

(наприклад, для підключення житлового будинку або під'їзду будинку до Інтернет по існуючій телефонній проводці). Телефонну лінію при цьому можна використовувати для ведення переговорів.

- Стандарт HomePlug 1.0 забезпечує доступ до Інтернет через побутову електричну мережу, підтримує швидкість передачі до 14 Мбіт/с. і максимальну протяжність між вузлами до 300 м.
- Технологія PLC (Power Line Communication) дозволяє передавати дані по високовольтних лініях електропередач, без додаткових ліній зв'язку. Комп'ютер підключається до електричної мережі і виходить в Інтернет через одну і ту ж розетку. Для підключення до домашньої мережі не потрібні жодні додаткові кабелі. До домашньої мережі можна підключити різне обладнання: комп'ютери, телефони, охоронну сигналізацію, холодильники і т.д.

4. Види ресурсів Internet.

Основним протоколом мережі Інтернет є мережевий протокол TCP/IP. Кожен комп'ютер в мережі TCP/IP має свою унікальну IP-адресу або IP-номер. Адреси в Інтернеті можуть бути представлені як послідовністю цифр, так і ім'ям, побудованим за певними правилами. Комп'ютери при пересилці інформації використовують цифрові адреси, а користувачі в роботі з Інтернетом використовують в основному імена.

Цифрові адреси в Інтернеті складаються з чотирьох чисел, кожне з яких не перевищує 256. При записі числа відділяються крапками, наприклад: 195.63.77.21. Такий спосіб нумерації дозволяє мати в мережі більше чотирьох мільярдів комп'ютерів.

Спочатку в мережі Internet застосовувалися IP-номери, але коли кількість комп'ютерів в мережі стала більшою за 1000, то був прийнятий метод зв'язку імен і IP-номерів, який називається сервер імені домена (Domain Name Server, DNS). Сервер DNS підтримує список імен локальних мереж і комп'ютерів та відповідних їм IP-номерів.

В Інтернеті застосовується так звана доменна система імен. Кожен рівень в такій системі називається доменом. Типове ім'я домена складається з декількох частин, розташованих в певному порядку і розділених крапками. Домени відділяються один від одного крапками, наприклад: www.lessons-tva.info або tva.jino.ru.

Доменна система імен використовує принцип послідовних уточнень (як і в звичайних поштових адресах – країна, місто, вулиця і будинок, в який слід доставити лист).

Домен верхнього рівня розташовується в імені правіше, а домен нижнього рівня – лівіше. У нашому прикладі домени верхнього рівня info і ru вказують на те, що мова йде про приналежність сайту www.lessons-tva.info до тематичного домена верхнього рівня info, а сайту tva.jino.ru до російської (ru) частини Інтернету. Але в Росії багато користувачів Інтернету, тому наступний рівень визначає організацію, якій належить дана адреса. У нашому випадку це компанія jino.

Інтернет-адреса цієї компанії – jino.ru. Всі комп'ютери, підключені до Інтернету в цій компанії, об'єднуються в групу, що має таку адресу. Ім'я окремого комп'ютера або мережі кожна компанія вибирає для себе самостійно, а потім реєструє його в тій організації Інтернет, яка забезпечує підключення.

Це ім'я в межах домена верхнього рівня має бути унікальним. Далі слідує ім'я хоста tva, таким чином, повне ім'я домена третього рівня: tva.jino.ru. В імені може бути довільна кількість доменів, але найчастіше використовуються імена з кількістю доменів від трьох до п’яти.

Доменна система утворення адрес гарантує, що у всьому Інтернеті більше не знайдеться іншого комп'ютера з такою ж адресою. Для доменів нижніх рівнів можна використовувати будь-які адреси, але для доменів самого верхнього рівня існує угода.

У системі адрес Інтернету прийняті домени, представлені географічними регіонами. Вони мають ім'я, що складається з двох літер, наприклад: Україна – ua; Франція – fr; Канада – ca; США – us; Росія – ru.

Існують і домени, розділені за тематичними ознаками, наприклад:

- навчальні заклади – edu,
- урядові установи – gov,
- комерційні організації – com.

Останнім часом додані нові зони, наприклад: biz, info, in, cn і т. д..

При роботі в Internet використовуються не доменні імена, а універсальні покажчики ресурсів, які називаються URL (Universal Resource Locator). URL – це адреса будь-якого ресурсу (документа, файла) в Internet, він вказує, за допомогою якого протоколу слід до нього звертатися, яку програму слід запустити на сервері і до якого конкретного файла слід звернутися на сервері. Загальний вид URL: протокол://хост-комп’тер/ім’я файла (наприклад: http://lessons-tva.info/book.html).

Реєстрація домена здійснюється у вибраній користувачем зоні ua, ru, com, net, info і т. д.. Залежно від призначення сайту вибирається його зона реєстрації. Для реєстрації сайту бажано вибрати домен другого рівня, наприклад lessons-tva.info, хоча можна працювати і з доменом третього рівня, наприклад tva.jino.ru.

Домен другого рівня реєструється у реєстратора – організації що займається адмініструванням доменних імен, наприклад http://www.imhoster.net/domain.htm. Домен третього рівня отримується, як правило, разом з хостингом в хостинговій компанії. Ім'я сайту вибирають, виходячи з виду діяльності, назви компанії або прізвища власника сайту.

З метою класифікації усього розмаїття представленої в мережі Інтернет інформації можна розглядати способи її передачі та збереження на серверах, програмне забезпечення, що використовується для її опрацювання, чи способи споживання (сприйняття) користувачами.

Служби (сервіси) - це види послуг, які надаються серверами мережі Інтернет. В історії Інтернет існували різні види сервісів , одні з яких в даний час вже не використовуються , інші поступово втрачають свою популярність , в той час як треті переживають свій розквіт. Така класифікація умовна, оскільки, наприклад, сервери лише одного пошукового сервісу google.com на основі технології www (яка в поданій класифікації є окремим сервісом) надає на сьогоднішній декілька сотень різноманітних послуг.

В традиційних класифікаціях можна знайти такі види сервісів:

- 1. World Wide Web (www) - Всесвітня павутина - служба пошуку та перегляду гіпертекстових документів, що включають в себе графіку, звук і відео.
- 2. E-mail - . Електронна пошта - служба передачі електронних повідомлень.
- 3. Usenet, News - . Телеконференції , групи новин - різновид мережевої газети або дошки оголошень.
- 4. FTP - служба передачі файлів.
- 5. ICQ - . Служба для спілкування в реальному часі за допомогою клавіатури.
- 6. Telnet - служба віддаленого доступу до комп'ютерів.
- 7. Gopher - Служба доступу до інформації за допомогою ієрархічних каталогів.

Серед цих служб можна виділити служби , призначені для комунікації, тобто для спілкування , передачі інформації, а також служби , призначення яких - це зберігання інформації та забезпечення доступу до цієї інформації користувачів. Серед останніх служб лідируюче місце за обсягом збереженої інформації займає служба WWW, а сервіси, що інтенсивно розвиваються на базі неї поступово заміняють перелічені тут служби.

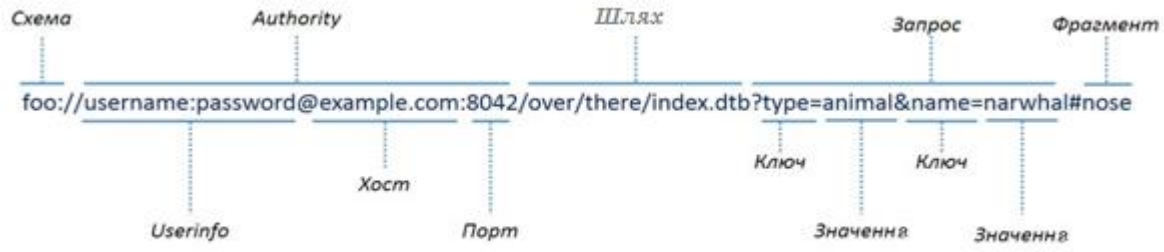
Види інформації в мережі Інтернет можна систематизувати за способами її передачі, тобто протоколами передачі даних верхнього (прикладного) рівня, що визначаються відповідним програмним забезпеченням та способом кодування даних. Основні протоколи прикладного рівня:

- FTP (File Transfer Protocol) – протокол передачі файлів, використовується для обміну файлами між комп’ютерами та серверами в Internet, наприклад, FTP-серверами. Зазвичай для передачі файлів використовуються спеціальні програми FTP-клієнти, наприклад FileZilla.
- TFTP (Trivial File Transfer Protocol) - використовується головним чином для початкового завантаження бездискових робочих станцій. TFTP, на відміну від FTP, не має можливостей аутентифікації;
- DNS (Domain Name System) – використовується для перетворення імен хостів в мережеві адреси одноіменною службою з підтримки доменних імен;
- SMTP (Simple Mail Transfer Protocol), IMAP (Interactive Mail Access Protocol), POP3 (Post Office Protocol version 3) – різні протоколи обміну поштовими повідомленнями;
- NNTP – протокол передачі новин між відповідними ресурсами новин та клієнтським ПЗ;
- HTTP (Hyper Text Transfer Protocol) – протокол обміну гіпертекстовою інформацією в Word Wide Web. Передана інформація з боку клієнта опрацьовуються і відображається за допомогою відповідної програми – Інтернет-переглядача (браузера).
- HTTPS (Hyper Text Transfer Protocol Secure) – теж, що HTTP але в захищеному від сторонніх втручань режимі за рахунок шифрування даних, що передаються.

З розвитком мережі Інтернет, в міру необхідності, створюються і розвиваються інші протоколи передачі даних. Наприклад, BitTórrent – пірінговий (P2P) протокол для кооперативного обміну файлами між комп’ютерами.

Кожен файл чи сервіс, розташований на серверах мережі Інтернет вважається ресурсом і має унікальну назву серед інших ресурсів мережі, яка називається URI (англ. Uniform Resource Identifier) - уніфікований (однаковий) ідентифікатор ресурсу. На

англійський манер вимовляється як [ю-ар-ай], у нас говорять [урі]. URI - це послідовність символів, що ідентифікує абстрактний або фізичний ресурс. При цьому URI може вказувати як місце розташування ресурсу (URL), так і його ім'я (URN). А може містити і те й інше. Тобто URL і URN - це окремі випадки URI. Структуру запису URI можна продемонструвати таким прикладом:



5. Пошук інформації в Internet

Пошукові системи дозволяють із величезної кількості інформації в мережі практично моментально одержати саме те, що потрібно. Найпопулярніша пошукова система на сьогоднішній день у світі Google. Компанія Google прийшла у світ Інтернету досить пізно в 1996 році. Тоді вже існували монстри пошуку, такі як Yahoo! (http://www.yahoo.com). Але завдяки унікальній технології пошуку компанія незабаром завоювала заслужену популярність і стала найкращим пошуковим сервісом. Пошукова система Яндекс найпопулярніша для пошуку по російськомовних ресурсах.

Щоб виконати пошук достатньо просто ввести у рядок пошуку те, що вас цікавить, натиснути клавішу Enter або кнопку Пошук, і Google виконає в Інтернеті пошук змісту, що ставиться до вашого запиту.

У більшості випадків можна знайти потрібну інформацію за допомогою простого запиту (слова або фрази, які ввели). Однак наступні рекомендації допоможуть якнайкраще скористатися наявними можливостями. Надалі будемо використовувати квадратні дужки [ ] для позначення пошукового запиту, тобто [чорно^-білі] - один запит, а [чорний] і [білий] є двома різними запитами. Кілька простих фактів:

- Кожне слово має значення. Як правило, у запиті використовується кожне слово.
- При пошуку ніколи не враховується регістр символів. Пошуковий запит [new york times] нічим не відрізняється від пошукового запиту [New York Times].
- Розділові знаки, як правило, не враховуються, як і спеціальні символи, такі як @#$%^&*()=+[]\@@@

Запити повинні бути простими. Якщо ви шукаєте якесь підприємство, просто введіть його назву або хоча б ту частину назви, що ви помнете напевно. Якщо ви шукаєте конкретне поняття, місце або продукт, почніть із його назви або ім'я. Якщо ви шукаєте піцерію, просто введіть слово "піцерія" і назву свого міста або поштовий індекс. Для більшості запитів зовсім не потрібні оператори або витончений синтаксис. Чим простіше, тим краще.

Подумайте, які слова присутні на сторінці, що ви шукаєте. Пошукова система не людина. Це програма, що порівнює слова, які ви вводите, і слова, які є на веб-сторінках.

Використовуйте слова, які з найбільшою ймовірністю можуть бути присутнім на шуканій сторінці. Наприклад, замість [у мене болить голова] уведіть [головний біль], тому що саме цей термін буде використовуватися на сторінці, присвяченій медицині. Запит [у якій країні кажани вважаються гарною прикметою?] зрозумілий людині, але в документі, що містить відповідь, може не бути цих слів. Уведіть краще [кажани вважаються гарною прикметою в] або навіть [кажани гарна прикмета], оскільки ці слова швидше за все присутні на потрібній сторінці.

Опишіть, що вам потрібно, використовуючи якнайменше слів. Кожне слово в запиті служить для звуження й уточнення області пошуку. Оскільки використовуються всі слова, кожне додаткове слово обмежує коло результатів. Якщо ввести занадто багато обмежень, можна пропустити корисну інформацію. Почніть пошук з декількох ключових слів. Навіть якщо ви не знайшли те, що потрібно, переглянете знайдені результати, і ви зрозумієте, які додаткові слова потрібно включити в наступний запит, щоб одержати більше релевантні результати. Наприклад, простий запит [погода мінськ] дасть кращі результати, чим більше довгий запит [прогноз погоди для мінська білорусь].

Підбирайте більш інформативні слова. Чим більш інформативне слово використовується, тим більше ймовірність, що результати будуть релевантними. Такі слова, як "документ", " веб-сайт", "компанія" або "інформація" звичайно зайві. При цьому варто пам’ятати, що навіть якщо ви використовуєте правильне слово, але більшість людей рідко ним користується, це слово може не виявитися на потрібній сторінці. Наприклад, запит [популярні рингтони] більш інформативний і конкретний, ніж [популярні мелодії].

Розширений пошук в Google:

Цей вид пошуку використовується рідко, менш 5% від загального числа разів. Простого базового пошуку часто буває цілком достатньо.

Пошук по словосполученню (""). Якщо ви містите набір слів у подвійні лапки, то тим самим даєте команду розглядати зазначені слова саме в такому порядку, без змін. Google уже використовує цей порядок слів і просто так від нього відхилятися не буде, тому лапки в більшості випадків зайві. Наполягаючи на пошуку по словосполученню, ви можете випадково пропустити важливі результати. Наприклад, при пошуку ["Тарас Шевченко"] (у лапках) будуть пропущені сторінки, у яких згадується Тарас Григорович Шевченко.

Пошук у межах певного веб-сайту (site:). Google дозволяє вказувати, що результати пошуку повинні бути із зазначеного веб-сайта. Наприклад, запит [ірак site:kommersant.ru] поверне сторінки про Ірак, але тільки із сайту kommersant.ru. Більш прості запити [ірак kommersant.ru] або [ірак Комерсант] звичайно бувають так само ефективні, хоча можуть повернути результати з інших сайтів, що згадують "Комерсант". Можна вказати цілий клас веб-сайтів. Наприклад, [ірак site:.ru] поверне результати тільки з домена .ru, а [ірак site:.iq] поверне результати тільки з іракських веб-сайтів.

Пошукові слова, які потрібно виключити (-). Додавання знака мінуса прямо перед словом приведе до того, що сторінки, що містять це слово, не будуть з'являтися у ваших результатах пошуку. Указуйте знак мінуса безпосередньо перед словом, а перед мінусом ставте пробіл. Наприклад, у запиті [кисло-солодкий соус]знак мінуса використовується як дефіс і не буде вважатися символом виключення. А в запиті [кисло^-солодкий -соус] буде виконаний пошук слів "кисло^-солодкий", але будуть виключені посилання на соус.

Можна виключити будь-яке число слів, ставлячи знак - перед кожним з них, наприклад [ягуар -автомобілі -футбол -ос]. Знак - можна використовувати для виключення не тільки слів. Наприклад, поставте його перед оператором "site:" (без пробілу), щоб виключити певний сайт із результатів пошуку.

Заповнення порожніх місць (*). Знак *, або підставний знак, – це маловідома функція, що може бути дуже дієвою. Якщо в запиті вказати зірочку (*), то вона буде означати будь-яке невідоме пошукове слово, що дозволяє знайти найкращі відповідності. Наприклад, запит [Google *] видасть результати про багатьох продуктах Google (на декількох сторінках). Запит [Верховна рада проголосувала * по * законопроекту] видасть результати про різні голосування по різних законопроектах. Зверніть увагу, що оператор * працює тільки із цілими словами, а не частинами слів.

Пошук точної відповідності (+). Google автоматично включає синоніми. У такий спосіб він знаходить сторінки, що містять, наприклад, слово "легкодоступний" по запиті [легко доступний] (із пробілом), або історію Російської Федерації по запиті [історія рф]. Але іноді допомога Google буває зайвою, і ви одержуєте синоніми там, де вони не потрібні. Якщо поставити знак + безпосередньо перед словом (без пробілу після +), Google знайде точні відповідності саме цьому слову. Якщо укласти одне слово в подвійні лапки, ефект буде той же.

Оператор OR. За замовчуванням Google ураховує всі слова в запиті. Якщо ви хочете дозволити яке-небудь із декількох слів, то можете використовувати оператор OR (зверніть увагу, що варто набирати "OR" ЗАГОЛОВНИМИ БУКВАМИ). Наприклад, запит [Динамо 2004 OR 2005] видасть результати про одній із цих дат, у той час як [Динамо 2004 2005] (без OR) поверне результати, у яких зазначені обидва роки на одній сторінці. Замість OR можна використовувати символ |. (Зверніть увагу, що оператор AND використовується за замовчуванням, тому його вказувати не потрібно.)

Лекція 12. Мережні технології. Поняття гіпертексту як основного засобу подання інформації в Internet.

1. *Поняття гіпертексту і подання інформації в Internet.*
2. *Структура веб-документа.*
3. *Основні конструкції (теги) мови гіпертекстової розмітки (HTML).*
4. *Програми для редагування html-документів*
5. *Інші види гіпертекстової розмітки, технологія Wiki-проектів.*

1. Поняття гіпертексту і подання інформації в Internet.

Серед перелічених у попередній лекції сервісів мережі Інтернет (***World Wide Web (www), E-mail, Usenet, News, FTP, ICQ*** і т.п.) найбільшого поширення і розвитку, в силу своєї зручності, набула www. Більшість рядових користувачів мережі сьогодні отожнюють понятя Інтернет і World Wide Web. Це пов'язано з тим, що служба www у своїй технології має достатньо засобів для створення сервісів, що частково, чи пвністю заміняють аналогічні служби мережі. Наприклад, сучасні поштові сервери мають засоби для доступу користувача до електронної пошти через web-інтерфейс безпосередньо із вікна браузера, позбавляючи його потреби встановлювати і налаштовувати спеціальні програми для роботи з електронною поштою. Подібну ситуацію можемо спостерігати і з іншим службами інтернету.

Попри розмаїття сучасних web-технологій в основі функціонування World Wide Web лежить простий механізм передачі гіпертексту від сервера, на якому розміщено Інтернет-ресурс до web- браузера на комп'ютері користувача через протокол http (чи https). ***Гіпертекст*** — це така форма організації тексту, при якій його одиниці представлені не в лінійній послідовності, а як система явно вказаних можливих переходів, зв'язків між ними. Слідуючи цим зв'язкам, можна читати матеріал в будь-якому порядку, утворюючи різні лінійні тексти. Найпростіший приклад гіпертексту «доінтернетовської епохи» — це будь-який словник чи енциклопедія, де кожна стаття має посилання до інших статей цього ж словника.

У www використовується спеціальна мова розмітки гіпертексту web-сторінок - HTML (англ. ***HyperText Markup Language*** — Мова розмітки гіпертекстових документів). Браузер отримує html-код web-сторінки за її url в мережі і відображає на екрані за правилами, вказаними в цьому коді. Дані у форматі HTML нагадують звичайні текстові файли за винятком того, що деякі символи в них (так звані теги (tag)) інтерпретуються як розмітка. Розмітка надає документу деяку, визначену тегами, структуру: параграфи, розділи, абзаци, списки, малюнки, таблиці, колонтитули, індекси, зміст тощо. Головною особливістю HTML є спроможність використовувати гіперзв'язки (links), завдяки яким можливі посилання та переходи з поточної веб-сторінки на інші документи. url сторінки, яку має відобразити браузер можна вказувати у відповідному полі (адресній стрічці) його вікна, але, переважно, перехід на потрібну web-сторінку здійснюється саме за рахунок гіперзв'язків на інших сторінках. В тому числі зі сторінок спеціальних служб мережі www, які називаються пошуковими сервісами (google.com, yandex.ru, bing.com і т.п.). Сучасні браузери мають вбудовані інструменти пошуку що використовують можливості згаданих пошукових сервісів.

Зазвичай web-сайти фізично розташовуються на серверах спеціальних хостингових компаній (хостинг-провайдерів, HSP - Hosting Service Provider) що підтримують роботу серверів у мережі і надають користувачам доступ до інформації сайту. Послуги host-провайдерів не завжди є платними, часто провайдери з різних маркетингових міркувань надають базові пакети послуг хостингу на безкоштовних умовах. З іншого боку, для підтримки доменної адресації ресурсів в мережі функціонують спеціальні DNS - сервери (служба DNS - Domain Name System), що транслюють ім'я домену в IP-адресу хоста, де фізично розташований сайт. Щоб користувач зміг переглянути web-сторінку за відповідною адресою (url), його браузер повинен відправити до сервера запит відповідної структури (по мережі через систему доменних імен). Сервер, у відповідь на запит, пересилає браузеру гіпертекстовий файл, що відповідає вказаному url. Цей файл може як постійно зберігатися за вказаною адресою (статичний html), так і формуватися на сервері за допомогою відповідного ПЗ (наприклад інтерпретатора мови PHP) з даних збережених насайті (у базі даних та інших файлах). Браузер інтерпретує отриманий гіпертекст і відображає web- сторінку відповідно до специфікації мови html.

2. Структура веб-документа.

HTML (англ. HyperText Markup Language — **Мова розмітки гіпертекстових документів**), призначена для структурування документів, що містять текст, зображення, гіперпосилання, тощо. Разом із видимою інформацією, HTML-документи містять додаткові метадані, такі як, наприклад, мова тексту, автор документа, стислий підсумок. Мова розмітки розроблялась консорціумом W3C (World Wide Web Consortium – офіційний сайт http://www.w3.org/) очікується, що HTML буде замінена розширюваною мовою розмітки гіпертексту (XHTML).

Команди **Html** називають **ТЕГ**-ами (англ. ***tag***). Теги бувають парними і одиничними. Парні теги обмежують деякий фрагмент гіпертексту початкови тегом виду **<...>**, и кінцевим тегом **</...>** з однакив іменем (наприклад, текст курсивом). Парні теги призначені для застосування до фрагментів гіпертексту певних параметрів форматування, чи розбиття документа на деякі частини. Так, наприклад, параними є теги ***<html>...</html>***, що обмежують текст HTML-документа, ***<body>...</body>***, що обмежують його змістовну частину та ***<head>...</head>***, що відділяють заголовкову частину. Одиничні теги, переважно, використовують для вставки в гіпертекст якихось його спеціальних елементів (наприклад, тег
 вставляє в гіпертекст спецсимвол переходу на нову стрічку без створення нового абзацу). HTML регістронезалежна, тобто теги ****, **<STRONG>**, чи **<StrOnG>** діятимуть на відображення гіпертексту браузером одинаково.

При використанні вкладених парних тегів слід закривати їх в зворотному порядку (Тут текст буде **неправильним**, а Тут текст правильний).

Документи **Html**, як правило, мають таку стрктуру:

```
<html>
 <head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Сторінка студента</title>
</head>
<body>
<p>Текст документа.</p>
</body>
</html>
```

В розділі заголовку ***<head>...</head>*** за допомогою тегу ***<title> </title>*** можна задати заголовок(назву) сторінки. Елементи ***<meta>*** у заголовку дозволяють задавати додаткову «метаінформацію» про сторінку, таку як спосіб кодування тексту ("Content-Type"), автора, ключові слова, програму, що згенерувала гіпертекст і т.п.

3. Основні конструкції (теги) мови гіпертекстової розмітки (HTML).

Теги в HTML, зазвичай мають не тільки назву, але і атрибути, яким можна надавати певних значень, наприклад текст . Значення атрибутів слід записувати після імені тегу (початкового, для парних тегів) за правилом: назва атрибута = “значення атрибута” через пропуск в довільному порядку. Зазвичай, для тегів визначена деяка кількість атрибутів, які на практиці не завжди записуються і їх значення вважається рівним значенню за замовчуванням. Наприклад текст документа поміщений в `<body bgcolor="gold" text="0000ff">…</body>` буде відображатися блакитними літерами на золотистому фоні, тоді як текст між `<body> та </body>` без атрибутів буде написаний за замовчуванням чорним по білому. Стосовно тега ***<body>*** можна задавати такі атрибути:

- **bgcolor -** встановлює колір фону документа;
- **text -** встановлює колір символів;
- **link -** колір, яким відображаються гіперзв’язки з іншими документами;
- **vlink -** колір гіперзв’язків з документами, що вже були переглянуті;
- **alink –** колір гіперзв’язків під час натискання мишою;
- **background –** адреса фонового зображення.

Для атрибутів, що задають колір (не тільки стосовно тега ***<body>***) значення можна задавати:

англійським іменем відповідного кольору, наприклад “red”, “green”, “gold” і т.п.

в форматі #RGB, де R (red - красный), G (green - зеленый), B (blue - синий)- рівень насиченості складових кольорів заданий двоцифровими шістнадцятковими числами від 00 (0) до FF (255), наприклад, зелений колір без домішок червоного і синього задається числом #00FF00, коричневий колір середньої насиченості – #7F7F00 і т.п. Добирати значення складових кольорів можна за допомогою більшості графічних редакторів, в т. ч. Paint.

Для форматування тексту можна використовувати теги ... , для поділу на абзаци і <h1> - <h6> для виділяння заголовків. Для задання елементів форматування фрагмента тексту можна використовувати <big>...</big>, <small>...</small>, _{...}, ^{...}, ..., ..., ..., ...,

<u>...</u>, <s>...</s>, <strike>...</strike>. Для встановлення таких параметрів як розмір і колір символів та тип шрифта слід використовувати атрибути **face, size** та **color** тега **<font>**.

Для створення списків використовують теги **
... (**маркований список**)** та **
...
** (нумерований список) всередині яких елементи списку розмежовують парним тегом **
....**

Для вставки малюнків використовується одиничний тег <img>, основні атрибути:

- **src –** адреса файла з малюнком, можна задавати шлях відносно поточного документа чи URL малюнка в Internet
- **alt** - задає текст, що буде відображатися замість малюнка, якщо він не завантажився;
- **lowsrc** – джерело зображення низької якості для швидкого попереднього завантаження;
- **align** - вирівнювання зображення відносно рядка тексту, може набувати значеню *top, middle, bottom, left, right;*
- **width** та **height** - атрибути задают розмір зображення по горизонталі та вертикалі відповідно;
- **border** - товщина рамки, довкола малюнка.

Таблиці в гіпертекст вставляють з допомогою парного тега **<table>…</table>,** йому можна задавататрибути: **align, width, border**, **bgcolor, background** (фонове зображення, підтримують не всі браузери). Комірки таблиці розграфлюються за допомогою тег **<tr>…</tr> та <td>…</td>, або <th>…</th>** ().Кількість комірок в кожному рядку таблиці повинна бути однаковою.

Тег **<tr>** створює черговий рядок таблиці (можна не закривати) і має атрибути: **align, valign, bgcolor**.

Комірки таблиці визначають тегами **<td>…</td>**(комірка з даними), або **<th>…</th>**(комірка з заголовком). Атрибути ALIGN, VALIGN, WIDTH, HEIGHT, COLSPAN (кількість стовців, які займає комірка, т.з. об'єднані комірки) ROWSPAN (кількість рядків, що займає комірка), BGCOLOR, BACKGROUND. Межі комірок відображаються тільки тоді, коли вони щось містять, якщо потрібна порожня комірка, то в неї записують спеціальний символ .

Значення атрибутів, що задають розміри елементів (наприклад, **width** та **height**) можна задавати в різних одиницях вимірювання. Вони бувають

- абсолютними: in - Дюйм (2,54 см), cm - Сантиметр, mm - Миллиметр, pt - Пункт (1/72 дюйма);
- та відносними: em - висота поточного елемента, px - піксел, % - процент.

Одна з переваг гіпер текстів – можливість встановлення гіперзв'язків (гіперпосилань) між документами. Для для цього використовується тег **<a>…** в який можна поміщати і текст і малюнки. Атрибути:

- **href** – задає URL ресурсу на який веде посилання. Відповідно до правил запису URL може вказувати на www-документ (http://), файл (ftp://), запускати поштову програму (mailto:) і т. п.

- **name** – дозволяє створювати закладки всередині вмісту документа.
- **target** – визначає вікно (фрейм), в якому відкривається посилання, можливі значення:
- _self – поточне вікно; _parent – батьківське вікно фрейма; _top – батьківське вікно усієї фреймової структури; _blank –в новому вікні.
- **title** – текст підказки, що з'являється при наведенні на посилання вказівника миші.

Ще два парних теги … та … використовуються для обмеження фрагментів тексту та прямокутних областей сторінки відповідно і є засобом для використання CSS-стилів та опрацювання документа з допомогою скриптів.

Способи застосування CSS. Під способами застосування CSS ми в даному розділі розуміємо форму декларування стилю на HTML - сторінці і форму зв'язування опису стилю відображення елемента розмітки з самим елементом . Мова йде про те , де і в якій формі автор сторінки (або дизайнер) описує стиль , і як і в якій формі на нього посилається. Розрізняють чотири способи застосування стилів:

- перевизначення стилю в елементі розмітки;
- розміщення опису стилю в заголовку документа в елементі STYLE;
- розміщення посилання на зовнішній опис через елемент LINK ;
- імпорт опису стилю в документ.

Перевизначення стилю:

```
<H1 STYLE="font-weight:normal;
font-style:italic;
font-size:10pt;">
Заголовок первого уровня
</H1>
```

Атрибут стиль можна застосувати усередині будь-якого елемента розмітки. Очевидно , що не всі параметри стилю можна встановити для конкретного елемента розмітки. Потрібно відзначити що стилі розроблені в першу чергу для управління відображенням тексту. Не слід захоплюватися стилями при управлінні відображенням нетекстових елементів HTML - розмітки. Атрибут стиль стосовно різних елементів може мати найрізноманітніші параметри, наприклад, margin, ***padding, border, background-color, color, font-family, font-size, float*** і т.п. Параметріи атрибута STYLE часто дублюють аналогічні елементи HTML – розмітки, але CSS має набагато ширші можливості щодо форматування гіпертекстів. Детальний опис CSS-параметрів та їх допустимих значень можна знайти в мережі, наприклад, в довіднику http://css.manual.ru/.

Елемент STYLE

Застосування елемента - це впровадження каскадних таблиць стилів в структуру HTML -документа. Крім управління відображенням елементів розмітки , елемент STYLE дозволяє описувати стильові властивості елементів , які можна змінювати при програмуванні на JavaScript.

Елемент STYLE дає можливість визначити стиль відображення для :

- стандартних елементів HTML - розмітки;

- довільних класів (селектор КЛАС) ;
- HTML - об'єктів (селектор ID).

Наприклад, стандартні елементи розмітки описуються в елементі STYLE так:

```
<HEAD>
<STYLE>
p { color:darkred;text-align:justify;
   font-size:8pt; }</STYLE>
</HEAD>
<BODY>
...
<P>
Приклад тексту з визначеним стилем HTML-розмітки.
</P>
...
</BODY>
```

Тепер все параграфи документа будуть відображатися стилем з елемента STYLE , якщо тільки стиль не буде якимось способом перевизначений . У STYLE можна визначити стиль будь-якого елементу розмітки.

Посилання на зовнішній опис

Посилання на опис стилю , розташованого за межами документа, здійснюється за допомогою елемента LINK, який розміщують в елементі HEAD . Зовнішный опис може бути файлом, що містить опис стилів. Опис стилів в цьому файлі за синтаксисом відповідає опису стилів в тегу STYLE з попереднього способу. Переваги зовнішніх CSS очевидні – можна окремо редагувати CSS –файл, не втручаючись в HTML-розмітку, крім того, одну таблицю стилів можна використовувати для всіх сторінок сайту. З цих міркувань, в основному, використовують зовнішні каскадні таблиці стилів.

Приклад посилання на зовнішній опис стилів:

```
<LINK TYPE="text/css" REL="stylesheet"
    HREF="http://intuit.ru/my_css.css">
```

4. Програми для редагування html-документів

На сьогодні існує безліч всіляких систем створення веб-додатків, які підрозділяють на різні категорії:

- Текстові редактори (Emacs, gedit, Notepad, TextEdit, UltraEdit, vi і т.п.);
- Текстові редактори HTML-коду (т.з. «text-based HTML editors»: Alleycode HTML Editor, Aptana, Arachnophilia, BBEdit, Bluefish, Crimson Editor, CoffeeCup HTML Editor, EditPlus, Evrsoft 1st Page, HateML Pro, HotHTML, HTML-Kit, HTMLPad, Macromedia HomeSite, Notepad++, NoteTab, PSPad, Quanta Plus, SCREEM, Siteaid, skEdit, Taco HTML Edit, TextMate, TopStyle, WebTide і т.п.);
- Редактори WYSIWYG **(**«What You See Is What You Get» або «що ви бачите, то ви і получите»-редактори; досить могутні і зручні засоби створення і обробки веб-додатків: Adobe Contribute "Dreamweaver Lite", Adobe Dreamweaver – раніше називався Macromedia Dreamweaver, Adobe GoLive, Amaya, Blockstar, Bluevoda, HotDog, iWeb, Media Lab SiteGrinder, Microsoft Expression Web,

Microsoft SharePoint Designer, Microsoft Visual Studio / ASP.NET Web Matrix, Microsoft Visual Web Developer, NetObjects Fusion, Nvu, Quanta Plus, Sandvox, SeaMonkey Composer, Softpress Freeway, RapidWeaver, WorldWideWeb і т.п.).

5. Інші види гіпертекстової розмітки, технологія Wiki-проектів.

Для створення веб-додатків використовуються різноманітні технології і мови програмування, наприклад:

- PHP
- ASP
- Java Script
- Java
- CGI
- Perl
- Python
- Ruby on Rails

і інші. Ряд з них (**PHP**, Perl, Python) мають відкритий код, розповсюджуються вільно і можуть використовуватися практично на будь-яких веб-серверах, інші (ASP, Java) — прив'язані до конкретних веб-серверів та інтегрованих систем розробки програмних продуктів.

З точки зору передачі гіпертексту між сервером та браузером користувача роботу веб-додатків можна пояснити так. Навідміну від статичних HMTL-сайтів, на сервері зберігаються не самі сторінки, а програми, що генерують HMTL-текст у відповідь на запит браузера і повертають його браузеру для відображення вмісту сторінки. При цьому гіпертекст може генеруватися як на сервері (php, asp), так і на комп'ютері користувача (java, java-script). Очевидно, що у першому випадку на сервері має бути встановлений транслятор відповідної мови веб-програмування, в другому –відповідне ПЗ і функції браузера (віртуальна машина java, увімкнена трансляція java-скриптів у браузері) на ПК користувача. Ще одна особливість web-додатків – дані для генерування веб-сторінок обов'язково зберігаються у БД, тобто сервер повинен мати СКБД (My SQL, MS SQL Server, тощо).

MediaWiki - система для створення електронної енциклопедії. Практично не можливо знайти людину, котра б хоч раз не чула про такий проект як Вікіпедія. Саме для створення Вікіпедії було розроблено досить специфічний двигунець, який дозволив задовольнити специфічні потреби проекту, адже структура Вікі є справді іншою. Проте механізми структури були максимально спрямовані на створення інструмента побудови справжньої енциклопедії. І розробникам справді вдалось виконати поставлену задачу, ще й було створену свою розмітку аналогічно до HTML. MediaWiki – це наданий час чудовий інструмент для створення інтернет проектів енциклопедичного типу. Даний механізм дозволяє створювати посилання на ще неіснуючі статті. Це дозволяє спокійно завершити написання поточної статті, а потім натиснувши на попередньо-створене посилання, почати написання нової статті, сторінку для якої буде згенеровано автоматично.

Важливою особливістю MediaWiki є наявність шаблонів та можливість створення нових. Для чого існують шаблони (темплейти) MediaWiki? Тут річ у тім, що у енциклопедіях описуються різноманітні сутності. Кожна сутність має певні характеристики, дані, властивості і так далі. Саме для того щоб полегшити процес опису

сутностей (об'єктів) – споруд, людей, техніки було запроваджено систему шаблонів. Як приклад, можна навести наприклад умовний шаблон для Людини. В ньому можуть бути присутні такі поля як вік, стать, рік народження, місце проживання, освіта, сімейний стан та будь які інші. Відповідно, при написанні статті про якусь персону потрібно спочатку скопіювати шаблон, заповнити його, а далі писати основний текст статті. Система MediaWiki підтримує більшість класичних тегів html, але при ствренні Wiki-статей редактори користуються спеціальними правилами розмітки Wiki-сторінок:

Вирівнювання тексту можна задавати html-тегом з атрибутом style, наприклад<

… Або спеціальним тегом … для вирівнювання до центру.Розділи виділяються знаком «=» :

== Назва разділу ==

=== Підрозділ ===

==== Під-підрозділ ====

Тег
 розділяє стрічки тексту не починаючи новий абзац.

Курсив та потовщений текст поміщають між двома парами, чи, відповідно, трійками одинарних лапок. Так, наприклад, щоб написати слово потовщеним курсивом записують:

''''потовщений курсив''''.

Нумеровані списки позначають «#» а марковані «*», кількість # чи * означає рівень вкладеності пункту списку. Для додаткового відступу пункту списку можна використовувати «:». Можна створювати змішані списки.

Посилання в тексті wiki-сторінок поділяються на внутрішні (в межах сайту) та зовнішні (на сторонні ресурси). Особливість внутрішніх посилань MediaWiki виявляється в тому, що при переході на за внутрішнім посиланням на неіснуючу сторінку браузер не отримує повідомлення про помилку, бо система створює нову wiki-сторінку.

Внутрішні гіпер-посилання в MediaWiki розміщують в подвійних квадратних дужках. Вони можуть містити тільки назву сторінки, назву сторінки і текст посилання, розділені вертикальною рискою «|». Назва сторінки може утворюватися з назви простору імен та назви сторінки через «:». Наприклад

[[Вікіпедія:Довіка|Довідкова інформація]]

Для зовнішнього посилання достатньо вказати url. Можна задавати зовнішні посилання в одинарних квадратних дужках, додаючи після url через пропуск текст, що має відображати посилання.

Для задання кольору тексту чи фону використовуються CSS-атрибути параметра style відповідних html-тегів , , тощо.

Для створення таблиць використовують код що розпочинається з {| та закінчується |}. Рядки відділяються “|-”, а комірки в рядках просто “|”. Дозволяється використовувати html-параметри rowspan, colspan, border тощо.

Лекція 13. Мережні технології. Застосування мережних технологій в професійній діяльності.

1. *Огляд сучасних технологій створення мережних додатків.*
2. *Хмарні технології, Web 2.0.*
3. *Огляд деяких мережевих засобів для інженерних розрахунків задач та моделювання.*

1. Огляд сучасних технологій створення мережних додатків.

Для створення веб-додатків використовуються різноманітні технології і мови програмування, які забезпечують динамічне генерування html-сторінок інтернет-ресурсу у режимі взаємодії з користувачем. Починаючи від технологій спеціально розроблених для web-програмування, таких як:

- реалізація CGI-скиптів на мові програмування ***Perl***;
- генерування динамічних вебсторінок на стороні сервера за допомогою ***PHP***;
- розробка сторінок з динамічною поведінкою на стороні web-браузера клієнта, за допомогою технологій, що виникли на основі надзвичайно популярної сьогодні ***JavaScript***.

Практично кожна сучасна система програмування має засоби для розробки web-додатків для роботи в мережі www, до прикладу:

- створення Java Web Application засобами найрейтинговішої мови програмування Java;
- технологія ASP.Net від компанії Microsoft, яка функціонує на платформі .net з використанням декількох мов програмування, в тому числі C# та Visual Basic;
- технологія Ruby on Rails на основі мови програмування Ruby, щонабуває популярності.

З точки зору передачі гіпертексту між сервером та браузером користувача роботу веб-додатків такого типу можна пояснити так. Навідміну від статичних HMTL-сайтів, на сервері зберігаються не самі сторінки, а програми, що генерують HMTL-текст у відповідь на запит браузера і повертають його браузеру для відображення вмісту сторінки. При цьому гіпертекст може генеруватися як на сервері (php, asp), так і на комп'ютері користувача (java, java-script). Очевидно, що у першому випадку на сервері має бути встановлений транслятор відповідної мови веб-програмування, в другому – відповідне ПЗ і функції браузера (віртуальна машина java, увімкнена трансляця java-скриптів у браузері) на ПК користувача. Ще одна особливість web-додатків – дані для генерування веб-сторінок обов'язково зберігаються у БД, тобто сервер повинен мати СКБД (My SQL, MS SQL Server, тощо).

Окремо чи у поєднанні згадані технології призвели до появи найрізноманітнішого програмного забезпечення включно з інформаціними системами комерційного характеру, онлайновими аналогами популярного програмного забезпечення для настільних ПК. А також програмними рішеннями для автоматичнї підтримки типових ресурсів мережі WWW – сайтів, блогів, чатів, онлайн-конференцій, систем дистанційного навчання і т.п.,

які загалом можна окреслити абревіатурою ***CMS.*** Система управління вмістом або система управління контентом (англ. **Content management system**, CMS) — комп'ютерна програма, що використовується для управління вмістом чого-небудь (звичайно цей вміст розглядається як неструктуровані дані наочної задачі в протилежність структурованим даним, звичайно що знаходяться під управлінням СУБД (Система Управління Базами Даних)). Звичайно такі системи використовуються для зберігання і публікації великої кількості документів, зображень, музики або відео.

Окремим випадком такого роду систем є системи управління web-контентом (вмістом). Подібні **CMS** дозволяє управляти текстовим і графічним наповненням **веб-сайта**, надаючи користувачу зручні інструменти зберігання і публікації інформації.

Зараз існує безліч готових систем управління вмістом **сайту**, у тому числі і безкоштовних. Їх можна розділити на три типи, за способом роботи:

генерація сторінок за запитом. Системи такого типу працюють на основі зв'язки «Модуль редагування > База даних > Модуль уявлення». Модуль уявлення генерує сторінку із змістом при запиті на нього, на основі інформації з бази даних. Інформація в базі даних змінюється за допомогою модуля редагування. Сторінки наново створюються сервером при кожному запиті, а це створює навантаження на системні ресурси. Навантаження може бути багато разів понижено при використовуванні засобів кешування, які є в сучасних веб-серверах;

генерація сторінок при редагуванні. Системи цього типу суть програми для редагування сторінок, які при внесенні змін в зміст **сайту** створюють набір статичних сторінок. При такому способі жертвується інтерактивність між відвідувачем і вмістом сайту;

змішаний тип. Як зрозуміло з назви, поєднує в собі переваги перших двох. Може бути реалізований шляхом кешування — модуль уявлення генерує сторінку один раз, надалі вона в рази швидше підвантажується з кеша. Кеш може обновлятися як автоматично, по закінченню деякого терміну часу або при внесенні змін в певні розділи **сайту**, так і уручну по команді адміністратора. Інший підхід — збереження певних інформаційних блоків на етапі редагування сайту і збірка сторінки з цих блоків при запиті відповідної сторінки користувачем.

Термін контент-менеджер позначає рід людської діяльності — редактор **сайту**.

Велика частина сучасних систем управління вмістом реалізується за допомогою візуального (**WYSIWYG**) редактора — програми, яка створює **HTML**-код із спеціальної спрощеної розмітки, що дозволяє користувачу простіше форматувати текст.

CMS Drupal для створення повнофункціональних інтернет-проектів

Друпал є потужною безкоштовною платформою для створення сайтів та керування ними, яка починає своє історію з далекого 2000 року. А почалося все із невеличкої програми для керування університетським сайтом, які створив бельгієць Dries Buytaer для спілкування студентів Антверпенського університету із використанням PHP. Що цікаво, то те що із звичайного сайту спілкування проект поступово переріс у спільноту, учасники якої захоплювалися технологіями створення сайтів.

Drupal чудово зарекомендував себе серед розробників сайтів як надійна система, яка стабільно активно розвивається, має досить хороший захист від взлому, а також і широкі гнучкі можливості проектування та розвитку проекту, що й сприяло росту популярності системи.

Значний вплив на ріст популярності системи мали зручні засоби структуризації інформації, яка розміщалась на сайтах створених на основі Друпал, а також відображенню (подання) її користувачам. Тут було розроблено новітню своєрідну методику побудови структури, яка носить назву Таксономії, або простими словами – словники та терміни. При їх створенні їм можна надавати найрізноманітніші властивості, задавати способи відображення завдяки ще одному потужному механізму створеному саме для цього. Цей механізм називається Views. Це найпопулярніший модуль системи для якісної візуалізації матеріалів сайту. Завдяки гнучкості налаштування одну і ту ж саму інформацію можна подавати на різних сторінках сайту по-різному. Views дозволяє відображати поля матеріалів проекту в залежності від прав доступу відвідувача сайту, категорії розміщення контенту, розділу до якого він належить чи іншого критерію.

Також цей інструмент дозволяє створювати фільтри для відображення тих чи інших матеріалів. Що стосується створення матеріалів сайту, то для цього передбачена система різних типів контенту. В базовому варіанті ми можемо створювати Звичайні базові сторінки та Статті. Проте ми маємо можливість як редагувати існуючі типи контенту додаючи або видаляючи певні поля, так і створювати власні нові, які отримуватимуть саме потрібні поля для конкретного опису чи типу вмісту, що додається на сайт.

Як і для багатьох інших ЦМС, для Друпалу створено багато різноманітних тем шаблонів, які можна безкоштовно завантажити та використовувати, а при бажанні чи потребі – модифікувати їх адаптовуючи до особливостей свого проекту. Слід відмітити, що для Drupal існує надзвичайно велика кількість модулів, які виконують найрізноманітніші задачі, що значно полегшує роботу розробників. Проте, якщо Вам не вдається знайти необхідний модуль, то Ви можете створити його самостійно.

Завантажити останню версію Drupal можна безкоштовно з офіційного сайту drupal.org.

CMS для створення блогівWordpress. На сьогоднішній ця день безкоштовна система керуванням сайтів є однією з найбільш популярних систем створення та ведення блогів серед блогерів різного рівня. Своєю популярністю Wordpress завдячує надзвичайною простотою при встановленні та подальшому використанні, публікації нових матеріалів на сайт. Часто цю платформу крім блогерів використовують і сайти новин. Хоча інструментарій наявний у системі дозволяє створювати доволі різноманітні сайти, такі як сайти-візитки для компаній чи приватних осіб, фотогалереї для фотографів та інші.

Wordpress не вимагає глибокого знання мов програмування, проте маючи елементарні знання початкового рівня в HTML, PHP, CSS власник проекту може без зайвих зусиль змінювати внутрішні механізми та зовнішній вигляд сайт, тобто може унікалізувати структуру та дизайн сайту. Звісно, для даної СКС (системи керування сайтами) існує надзвичайно велика кількість платних та безкоштовних тем шаблонів, які достатньо всього лиш завантажити та активувати використовуючи інтерфейс керування адміністратора та насолоджуватись красивим дизайном новоствореного сайту, негайно починати наповнення цікавими матеріалами, статтями чи фотографіями.

Для Вордпресу існує і величезна кількість додаткових модулів, які дозволяють суттєво суттєво розширити функціонал сайту. Як і шаблони тем, додаткові модулі встановлюються із панелі керування адміністратора. Деякі з них потребують певного налаштування, яке здійснюється цілком інтуїтивно без зайвих труднощів. Саму інсталяцію Wordpress можна завантажити із офіційного сайту wordpress.org, а українську версію із uk.wordpress.org.

Що стосується типів публікацій, які можна здійснювати на Wordpress сайті, то після встановлення адміністратор чи інший користувач, який має на це право може створювати матеріали типу Допис блогу та сторінка. Як правило, для додавання нових публікацій використовується механізм додавання записів блогу – статті. Сторінки ж зазвичай відображають певну статичну інформацію, як не змінюється, наприклад, інформація про сайт, інформація про автора, координати і таке інше. Також тут слід відмітити, що вигляд головної сторінки також залежить від того, який контент вона буде відображати. Тут можна налаштувати так, що на ній буде відображатись стрічка нових доданих матеріалів, або ж можна створити статичну (не оновлювану автоматично) сторінку, яка і буде відображатись як головна. По замовчуванню сайт відображатиме нові додані матеріали. Для зміни вигляду потрібно спершу створити статичну сторінку, а тоді вказати в налаштуваннях відображення саме її.

MediaWiki - система для створення електронної енциклопедії. Практично не можливо знайти людину, котра б хоч раз не чула про такий проект як Вікіпедія. Саме для створення Вікіпедії було розроблено досить специфічний двигунець, який дозволив задовольнити специфічні потреби проекту, адже структура Вікі є справді іншою. Проте механізми структури були максимально спрямовані на створення інструмента побудови справжньої енциклопедії. І розробникам справді вдалось виконати поставлену задачу, ще й було створену свою розмітку аналогічно до HTML.

MediaWiki – це наданий час чудовий інструмент для створення інтернет проектів енциклопедичного типу. Даний механізм дозволяє створювати посилання на ще неіснуючі статті. Це дозволяє спокійно завершити написання поточної статті, а потім натиснувши на попередньо-створене посилання, почати написання нової статті, сторінку для якої буде згенеровано автоматично. Важливою особливістю MediaWiki є наявність шаблонів та можливість створення нових. Для чого існують шаблони (темплейти) MediaWiki? Тут річ у тім, що у енциклопедіях описуються різноманітні сутності. Кожна сутність має певні характеристики, дані, властивості і так далі. Саме для того щоб полегшити процес опису сутностей (об’єктів) – споруд, людей, техніки було запроваджено систему шаблонів. Як приклад, можна навести наприклад умовний шаблон для Людини. В ньому можуть бути присутні такі поля як вік, стать, рік народження, місце проживання, освіта, сімейний стан та будь які інші. Відповідно, при написанні статті про якусь персону потрібно спочатку скопіювати шаблон, заповнити його, а далі писати основний текст статті. Система MediaWiki підтримує більшість класичних тегів html, але при ствренні Wiki-статей редактори користуються спеціальними правилами розмітки Wiki-сторінок:

Вирівнювання тексту можна задавати html-тегом з атрибутом style, наприклад<

```
<p style="text-align:right;">…</p>
```

Або спеціальним тегом … для вирівнювання до центру.Розділи виділяються знаком «=» :

```
== Назва розділу ==
=== Підрозділ ===
==== Під-підрозділ ====
```

Тег
 розділяє стрічки тексту не починаючи новий абзац.

Курсив та потовщений текст поміщають між двома парами, чи, відповідно, трійками одинарних лапок. Так, наприклад, щоб написати слово потовщеним курсивом записують:

```
‘’’’’потовщений курсив’’’’’.
```

Нумеровані списки позначають «#» а марковані «*», кількість # чи * означає рівень вкладеності пункту списку. Для додаткового відступу пункту списку можна використовувати «:». Можна створювати змішані списки.

Посилання в тексті wiki-сторінок поділяються на внутрішні (в межах сайту) та зовнішні (на сторонні ресурси). Особливість внутрішніх посилань MediaWiki виявляється в тому, що при переході на за внутрішнім посиланням на неіснуючу сторінку браузер не отримує повідомлення про помилку, бо система створює нову wiki-сторінку.

Внутрішні гіпер-посилання в MediaWiki розміщують в подвійних квадратних дужках. Вони можуть містити тільки назву сторінки, назву сторінки і текст посилання, розділені вертикальною рискою «|». Назва сторінки може утворюватися з назви простору імен та назви сторінки через «:». Наприклад

```
[[Вікіпедія:Довіка | Довідкова інформація]]
```

Для зовнішнього посилання достатньо вказати url. Можна задавати зовнішні посилання в одинарних квадратних дужках, додаючи після url через пропуск текст, що має відображати посилання.

Для задання кольору тексту чи фону використовуються CSS-атрибути параметра style відповідних html-тегів , , тощо.

Для створення таблиць використовують код що розпочинається з {| та закінчується |}. Рядки відділяються “|-”, а комірки в рядках просто “|”. Дозволяється використовувати html-параметри rowspan, colspan, border тощо.

2. Хмарні технології, Web 2.0.

Web 2.0 — друге покоління мережних сервісів, що останнім часом стали основою розвитку мережі Інтернет. Термін Web 2.0 було запроваджено у 2004 видавництвом O'Рейлі (англ. O'Reilly Media) та комерційним організатором серії конференцій під назвою «Web 2.0», — МедіаЛайв (англ. MediaLive, сьогодні англ. CMP Technology).для означення сукупності прогресивних тенденцій у розвитку веб-технологій .

Принциповою відмінністю технології Web 2.0 від технологій Web 1.0 (першого покоління сервісів мережі Інтернет), є те, що її використання дає змогу не лише переглядати веб-ресурси мережі, а й завантажувати власні, здійснювати обмін цими ресурсами з іншими користувачами, діяти спільно з метою їхнього накопичення, брати участь в обговореннях та ін.

Сьогодні термін Web 2.0 швидше означає не стільки сукупність певних конкретних технологій, а філософію представлення інформації у веб-орієнтованому середовищі та побудову інформаційних відношень.

Технології Web 2.0 справедливо називають соціальними сервісами мережі Інтернет, оскільки їх використання зазвичай здійснюється спільно в межах відповідної групи користувачів. Групи користувачів можуть утворювати цілі мережні співтовариства, які об'єднують свої зусилля для досягнення відповідної мети. Прикладом такої групи може бути створення мережного співтовариства біюблдіотекарів для спільного використання веб-ресурсів та обміну досвідом.

Найбільш використовуваними у практиці стали такі соціальні сервіси:

- створення блогів (веб-журналів) та сайтів;
- вікі-енциклопедії;
- соціальні пошукові системи;
- зберігання мультимедійних веб-ресурсів;
- зберіганя закладок.

Варто зауважити, що використання соціальних сервісів web 2.0 не є складним процесом, оскільки не вимагає знань мови програмування або умінь створювати html-сторінки. Простота і зручність використання соціальних сервісів дає змогу економити час і не витрачати його на довгі пояснення технології функціонування веб-систем.

Наприклад, у сфері хостингу вебсайтів беззаперечним лідером довший час був web 2.0 сервіс ***ucoz***. Деякі з них активно розвиваються, ростуть та закріплюються на ринку, інші відходять у забуття. uCoz-у пощастило зайняти одне з лідируючих місць на ринку умовно-безкоштовних хостингів, які надають свою систему керування сайтами. До умовно-безкоштовних відносимо Укоз через те, що система самовільно примусово встановлює рекламні банери на ваш сайт, а також свій копірайт. Проте, для тих, хто створює свій сайт і не бажає бачити на ньому зайвих рекламних банерів, може заплатити суму еквівалентну від 3 до 9 американських доларів придбавши преміям аканут певного рівня. Це дозволить забрати укозівську рекламу з Вашого сайту, а також отримати певні додаткові вигоди, наприклад, більше простору для розміщення даних та інші.

Для створення сайту потрібно придумати та зареєструвати доменне ім'я для сайту. Сам uCoz надає можливість зареєструвати домен безкоштовно, проте він буде мати закінчення типу ucoz.ru, ucoz.ua, at.ua та інші. Для українських сайтів досить часто обирають at.ua. Загалом, такого доменного імені цілком достатньо, проте можна придбати і будь-яке інше, наприклад у реєстратора доменних імену Ucraine.Com.Ua та прикріпити його до свого сайту.

Для створення нового проекту uCoz пропонує надзвичайно широкий набір інструментарію, що дозволяє створити сайт будь-якого типу. Таким чином можна організувати свою фотогалерею, блог, форум каталог файлів чи статтей. Для роботи в цій системі керування сайтом не обов'язково знати будь-які мови програмуванні чи мати інші знання програміста, проте маючи елементарні знання по HTML/CSS буде досить легко вносити корекцію у зовнішній вигляд сайту та деякий його функціонал чи навіть створювати свої теми оформлення сайту, яких система надає досить багато і безкоштовно. Але мінус тут є в тому, що можна зустріти схожі сайти із практично однаковим дизайном. Тому краще вносити якісь зміни, щоб Ваш проект був справді унікальним та цікавим

іншим користувачам. Отже, uCoz чудово підходить як варіант для початківців у галузі сайтобудування, який дозволяє цілком безкоштовно розібратись в основних питаннях створення сайтів.

3. Огляд деяких мережевих засобів для інженерних розрахунків задач та моделювання.

Серед розмаїття технологій розробки вебдодатків та продуктів їх застосування, очевидно, можна знайти інструментарій на будь-який смак застосовний у будь-якій сфері професійної діяльності. Стрімкий розвиток технологій у цьому напрямку не дозволяє навести фіксований список корисних інтернет-сервісів усфері прикладної, який був би постійно актуальним. Тому, залишивши право опису найновіших ресурсів за пошуковими системами, тут на ведемо лише декілька типових прикладів.

Серед інтернет-ресурсів тематичного спрямування особливий інтерес викликає сайт ***phet.colorado.edu***, що надає безкоштовний доступ до цікавих, інтерактивних, науково-обгрунтованих симуляцій явищ та процесів, які досліджуються в природничих науках. Моделі написані на Java, Flash або HTML5, їх можна досліджувати безпосередньо за допомогою браузера, або завантажувати на комп'ютер. Всі моделі з відкритим кодом, тобто, спеціалісти з досвідом програмування з використанням зазначених техлогій можуть завантажити їх програмний код з метою їх модифікації чи вдосконалення.

Для прикладу наведемо лише декілька популяних фізичних симуляторів, доступних на сайті:

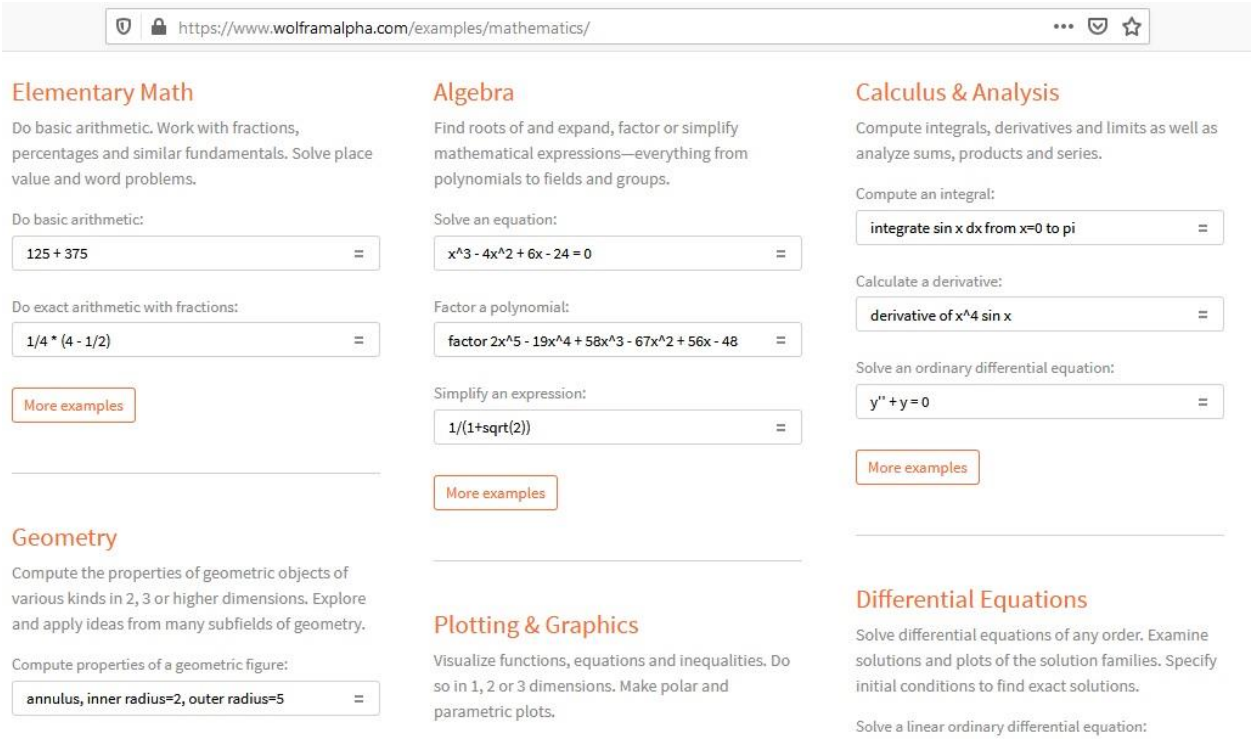
- Кульки і статична електрика - імітація для вивчення статичної електрики, передачі заряду, притягування, відштовхування та індукованих зарядів.
- Тертя - модель для дослідження понять, пов'язаних з тертям, що використовує хіміні властивості модельованих матеріалів.
- Лабораторія гравітаційних сил - візуалізація гравітаційних сил на прикладі притягування двох об'єктів один до одного.
- Закон Ома - керування повзунками напруги та опору дозволяє спостерігати за змінами рівняння та ланцюга.
- Опір провідників - спостереження за змінами, які відбуваються в провіднику під час керування повзунками опору, довжини та площі.
- Резерфордівське розсіювання - для дослідження потоку альфа-частинок в експерименті Резерфорда.

Проект Phet розвивається за рахунок пожертв спонсорів, що дозволяє зробити ці ресурси безкоштовними і вільними для використання для всіх учнів, студентів і викладачів. Редактори сайту ретельно перевіряють і тестуємо кожну модель для забезпечення її ефективного використання у навчанні.

Окрім тематичних ресурсів з фізики і у професійні діяльності можуть знадобитися також ресурси, що допомогають проводити різноманітні математичні розрахунки. До ресурсів такого типу можна віднести ***exponent.ru***, сайт який розпочинався майже на зорі інтернету зі збору російськомовної документації для системи інженерних розрахунків MATLAB, аналогічних програмних продуктів та суміжної літератури з математики та фізики. Сьогодні проект перетворився на справжній онлайновий навчальний центр, тут прводять тренінги, конференції та навчальні курси з прикладних математичних пакетів та їх використання у прфесіній діяльності.

В мережі функціонують також полегшені веб-версії популярних прикладних математичних пакетів, наприклад ***wolframalpha.com***. Система Mathematica створена американською компанією Wolfram Research, Inc., голова і засновник якої –відомий фізик і математик Стефан Вольфрам (Stephen Wolfram) –є основним автором розробки. Ще в 70-х роках молодий дослідник (С. Вольфрам народився в 1959 році), працюючи в різних галузях фізики, звернув увагу на те, що вченим дуже часто зустрічаються схожі комплекси громіздких математичних викладок, які віднімають багато часу. Поставивши собі за мету забезпечити вчених продуктивним математичним інструментом, Вольфрам зібрав колектив розробників для визначення архітектури нової (як тепер кажуть, повністю ексклюзивної) комп'ютерної системи. Далекоглядна концепція системи Mathematica полягало у створенні раз і назавжди такої системи символьної математики, в якій можна було б обробляти найрізноманітніші аспекти технічних обчислень і не тільки, когерентним і єдиним чином.Ключовим інтелектуальним досягненням, завдяки якому це стало можливо, стало створення нового виду символічної комп'ютерної мови, яка вперше змогла маніпулювати найширшим діапазоном об'єктів, необхідних для досягнення універсальності, обов'язкових для технічних обчислень, використовуючи при цьому лише невелику кількість примітивів.У серпні 1987 року була заснована Wolfram Research, а наступного року –у червні 1988 року –офіційно вийшла перша версія системи Mathematica на платформі Macintosh.

Сама система Mathematica розповсюджується за комерційною ліцензією з досить викою вартістю і не є доступною для широкого загалу користувачів. Проте її розробники з ознайомчою метою надають безкоштовний доступ до полегшеної web-версії програмного продукту Wolfram Alpha. Ця версія дозволяє виконувати не надто складні символьні розрахунки з різних розділів математики.



Лекція 14. Основи алгоритмізації та програмування.

1. *Етапи розв'язання прикладних задач з використанням обчислювальної техніки.*
2. *Поняття та властивості алгоритму.*
3. *Способи опису алгоритмів.*
4. *Базові алгоритмічні конструкції.*
5. *Огляд сучасних засобів, мов та технологій програмування.*

1. Етапи розв'язання прикладних задач з використанням обчислювальної техніки.

Основні етапи розв'язання прикладних задач із використанням комп'ютера Розв'язання задач із використанням комп'ютера характеризується декількома етапами, частина з яких виконуються безпосередньо людиною, решта — людиною і машиною:

- Постановка задачі. Опис початкових даних, формулювання мети задачі.
- Побудова інформаційної моделі. Опис реального об'єкта дослідження в припустимих для реалізації задачі термінах, щоб звести дослідження реального об'єкта до розв'язання задачі на моделі.
- Вибір програмного забезпечення. Визначення необхідного прикладного програмного забезпечення або розробка нового програмного забезпечення.
- Аналіз отриманих результатів. Аналіз результатів, отриманих на моделях та на реальних об'єктах, для виправлення помилок і доопрацювання розробленої прикладної програми, що пройшла тести на моделі.

Будь-яка програма формується в процесі програмування, що є складовою частиною загального процесу проектування обробки даних. Даний процес складається з послідовності етапів, що виконуються людьми різної кваліфікації, деякі з етапів виконуються автоматично на ЕОМ, в інших приймають участь людина і машина. Технологія розробки програм складається з декількох етапів:

- Аналіз задачі і постановка задачі для машинного вирішення;
- Розробка алгоритму (алгоритмізація);
- Складання блок-схеми;
- Складання програми на початковій мові програмування;
- Перенесення початкової програми і даних на технічний носій для введення в ЕОМ;
- Трансляція початкової програми на машинну мову;
- Редагування програми;
- Завантаження машинної програми в пам'ять і її виконання;
- Відлагодження програми;
- Тестування програми;
- Введення в експлуатацію.

До програмного забезпечення відносять програмні засоби (оформлені, як стандартні, бібліотечні або каталогізовані процедури), що дозволяють реалізувати слідуючі етапи:

- Програмування задачі.
- Підготовка інформації на машинному носії.
- Завантаження і відлагодження програми.
- Експлуатація готової програми.

Під програмним забезпеченням розуміють сукупність машинних і алгоритмічних мов програмування, трансляторів, алгоритмів, тестових, службових, прикладних (робочих) програм, а також описи і інструкції по їх застосуванню, що забезпечують технічну експлуатацію і використання машин. В залежності від функціонального призначення розрізняють загальне і спеціальне програмне забезпечення. До першого відносять мови програмування, трансляція, а також службові і тестові (випробовчі) програми; до спеціального – прикладні програми.

Система загального ПЗ включає комплекс випробовчих програм, системи програмування і програми операційної системи. Випробовчі програми (тестові чи технічного обслуговування) використовуються для контроля роботи, відлагодження і технічної експлуатації окремих пристроїв і ЕОМ в цілому. Це програми контрольних і діагностичних тестів.

Спеціальне програмне забезпечення є додатком до загального програмного забезпечення. Працюють ці програми під керівництвом операційної системи. В спеціальному програмному забезпеченні розрізняють пакети програм, що розширюють можливості ОС (забезпечення роботи комплексів), прикладні програми для вирішення наукових, інженерних, економічних і інших задач. ППП – це набір програмних засобів, розрахованих на певне коло задач і конкретного користувача. Пакети можуть бути з простою (набір залежних і незалежних) і складною структурою (включає провідну програму, тіло пакета, обслуговуючі програми).

Для того, щоб вибрати найоптимальніше програмне забезпечення, чи розробити його, необхідно користуватись слідуючими принципами:

продуктивність програми. При виборі програми передусім необхідно врахувати “розмір” об’єкту, тобто той обсяг інформації, з яким необхідно працювати програмі. Рекомендується вибирати програму з деяким запасом по продуктивності;

співвідношення затрат й ефекту. Ефект, що принесе куплена (створена) програма повинен переважати над затратами, що пов’язані з її купівлею, доробкою, адаптацією;

контроль і безпека даних. Бажано, щоб всі дані, що введені в програму, були надійно захищені і від несанкціонованого доступу до них, і від випадкового їх знищення. Програма повинна здійснювати контроль над введенням інформації і доступом до неї, а також створювати архівні копії файлів, що містять економічну інформацію;

сумісність. Програма, що купують (розробляють) повинна бути сумісною з організаційною структкрою конкретної фірми (підприємства) чи повинна мати иожливість пристосування до неї;

гнучкість. При виборі програми необхідно орієнтуватись на те, щоб вибрана програма мала можливість змінюватись при зміні зовнішнього середовища;

авторський супровід документацією. Програма повинна мати детальну і легку для використання документацію. Для складної комплексної програми важлива наявність "гарячої" телефонної лінії, завдяки якій в будь-який момент можна звернутись за консультацією до фірми-розробника і отримати необхідну інформацію.

2. Поняття та властивості алгоритму.

Розв'язування задач на комп'ютері засноване на понятті алгоритму. Алгоритм – це точне і однозначне розпорядження, що визначає обчислювальний процес, який веде від варійованих початкових даних до вихідного результату.

Алгоритм означає точний опис деякого процесу, інструкцію по його виконанню. Розробка алгоритму є складним і трудомістким процесом. Алгоритмізація – це техніка розробки (складання) алгоритму для вирішення завдань на ЕОМ.

Слово алгоритм походить від імені видатного вченого середньовічного Сходу **Мухаммеда ібн Муси аль-Хорезмі** (783 — 850), який в своїх наукових працях з математики, астрономії та географії описав і використовував індійську позиційну систему числення, а також сформулював у загальному вигляді правила виконання чотирьох основних арифметичних дій: додавання, віднімання, множення і ділення. Європейські вчені ознайомились з його працями завдяки їхнім перекладам на латину. При перекладі ім'я автора було подано як **Algorithmus**. Звідси й пішло слово алгоритм. А розроблені ним правила виконання арифметичних дій вважають першими алгоритмами.

Властивості алгоритму:

Дискретність алгоритму означає, що його виконання зводиться до виконання окремих дій (кроків) у певній послідовності. Причому, кожна команда алгоритму повинна виконуватися за скінченний проміжок часу.

Визначеність (або детермінованість) алгоритму означає, що для заданого набору значень початкових (вхідних) даних алгоритм однозначно визначає порядок дій виконавця і результат цих дій. Алгоритм не повинен містити команди, які можуть сприйматися виконавцем неоднозначно, наприклад, «Узяти дві-три ложки цукру», «Трохи підігріти молоко», «Вимкнути світло через кілька хвилин», «Поділити число x на одне з двох даних чисел a або b» тощо. Крім того, в алгоритмах недопустимі ситуації, коли після виконання чергової команди виконавцю неясно, яку команду він повинен виконувати наступною.

Виконуваність алгоритму означає, що алгоритм, призначений для певного виконавця, може містити тільки команди, які входять до системи команд цього виконавця. Так, наприклад, алгоритм для виконавця «Учень першого класу» не може містити команду «Побудуй бісектрису даного кута», хоча така команда може бути в алгоритмі, який призначений для виконавця «Учень восьмого класу». Зазначимо, що виконавець повинен лише вміти виконувати кожну команду зі своєї системи команд, і не важливо, розуміє він її, чи ні. Говорять, що виконання алгоритмів виконавцем носить формальний характер: виконавець може не розуміти жодну з команд, може не знати мети виконання алгоритму, і все одно отримає результат. Так, наприклад, верстат з програмним керуванням не розуміє жодної з команд, яку він виконує, але завдяки своїй конструкції успішно виготовляє деталі.

Скінченність алгоритму означає, що його виконання закінчиться після скінченної (можливо, досить великої) кількості кроків і за скінченний час при будь-яких допустимих значеннях початкових даних. Наведені вище послідовності команд є скінченними, а наступна послідовність команд — нескінченна:

- Взяти число 2.
- Помножити взяте число на 10.
- Додати до одержаного числа 5.
- Якщо одержане число додатне, то виконати команду 3, якщо ні, то припинити виконання алгоритму.

Результативність алгоритму означає, що після закінчення його виконання обов'язково одержуються результати, які відповідають поставленій меті. Результативними вважаються також алгоритми, які визначають, що дану задачу не можна розв'язати, або дана задача не має розв'язків при заданому наборі початкових даних.

Масовість алгоритму означає, що алгоритм може бути застосований до цілого класу однотипних задач, для яких спільними є умова та хід розв'язування, і які відрізняються тільки початковими даними. Таким, наприклад, є алгоритм розв'язування квадратного рівняння, який дозволяє знайти дійсні корені квадратного рівняння з довільними дійсними коефіцієнтами, або визначити, що при певних значеннях коефіцієнтів рівняння не має дійсних коренів. Масовим також є, наприклад, алгоритм побудови бісектриси довільного кута з використанням циркуля і лінійки. Однак, крім масових алгоритмів, складаються і застосовуються алгоритми, які не є масовими. Таким, наприклад, є алгоритм розв'язування конкретного квадратного рівняння.

3. Способи опису алгоритмів.

Для запису алгоритму рішення задачі застосовуються наступні способи їх зображення:

- Словесно-формульний опис
- Блок-схема (схема графічних символів)
- Алгоритмічні мови
- Операторні схеми
- Псевдокод

Для запису алгоритму існує загальна методика:

- Кожен алгоритм повинен мати ім'я, яке розкриває його зміст.
- Необхідно позначити початок і кінець алгоритму.
- Необхідно описати вхідні і вихідні дані.
- Вказати команди, які дозволяють виконувати певні дії над виділеними даними.

Загальний вигляд алгоритму

- Назва алгоритму
- Опис даних
- Початок
- Команди

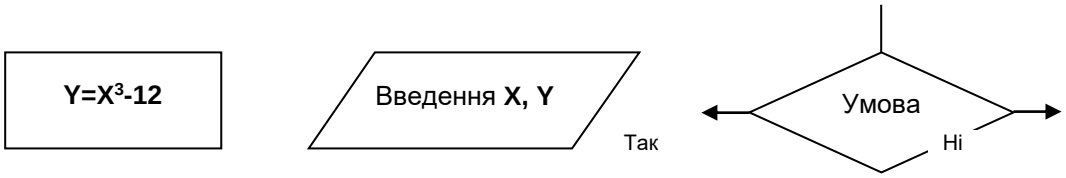
- Кінець

Формульно-словесний спосіб запису алгоритму характеризується тим, що опис здійснюється за допомогою слів і формул. Зміст послідовності етапів виконання алгоритмів записується на природній професійній мові предметної області в довільній формі.

Графічний спосіб опису алгоритму (блок-схема) набув найширшого поширення. Для графічного опису алгоритмів використовуються схеми алгоритмів або блочні символи (блоки), які з'єднуються між собою лініями зв'язку.

Кожен етап обчислювального процесу представляється геометричними фігурами (блоками). Вони діляться на арифметичні або обчислювальні (прямокутник), логічні (ромб) і блоки введення-виведення даних (паралелограм).

Схеми алгоритмів:



Порядок виконання етапів вказується стрілками, що з'єднують блоки. Геометричні фігури розміщуються зверху вниз і зліва направо. Нумерація блоків проводиться в порядку їх розміщення в схемі.

Алгоритмічні мови – це спеціальний засіб, призначений для запису алгоритмів в аналітичному вигляді. Алгоритмічні мови близькі до математичних виразів і до природних мов. Кожна алгоритмічна мова має свій словник. Алгоритм, записаний на алгоритмічній мові, виконується за строгими правилами цієї конкретної мови.

Операторні схеми алгоритмів. Суть цього способу опису алгоритму полягає в тому, що кожен оператор позначається літерою (наприклад, А – арифметичний оператор, Р – логічний оператор і так далі). Оператори записуються зліва направо в послідовності їх виконання, причому, кожен оператор має індекс, що вказує порядковий номер оператора. Алгоритм записується в один рядок у вигляді послідовності операторів.

Псевдокод – система команд абстрактної машини. Цей спосіб запису алгоритму за допомогою операторів близький до алгоритмічних мов.

4. Базові алгоритмічні конструкції.

За структурою виконання алгоритми і програми діляться на три види:

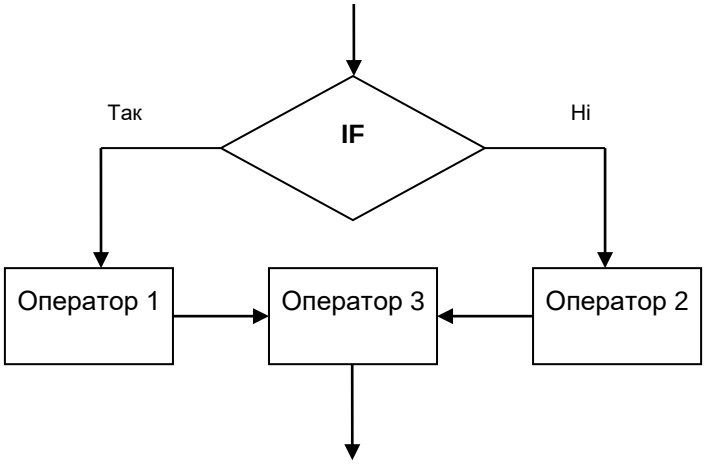
- Лінійні
- Розгалужені
- Циклічні

Лінійний алгоритм (лінійна структура) – це такий алгоритм, в якому всі дії виконуються послідовно один за одним і лише один раз. Схема є послідовністю блоків, які

розташовуються зверху вниз в порядку їх виконання. Початкові та проміжні дані не впливають на напрям процесу обчислення.

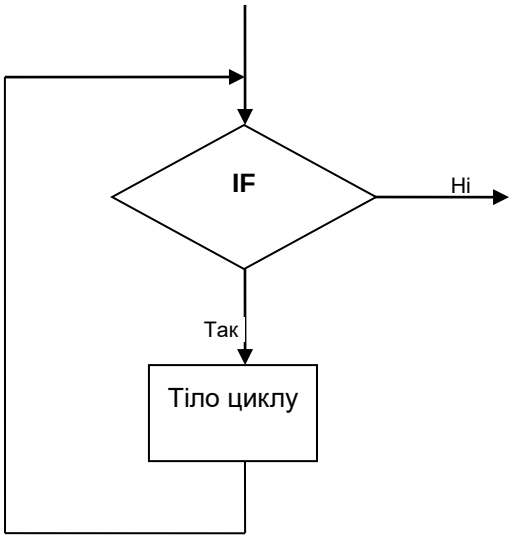
Розгалужені структури на практиці часто зустрічаються в задачах, в яких залежно від початкових умов або проміжних результатів необхідно виконати обчислення за тими чи іншими формулами.

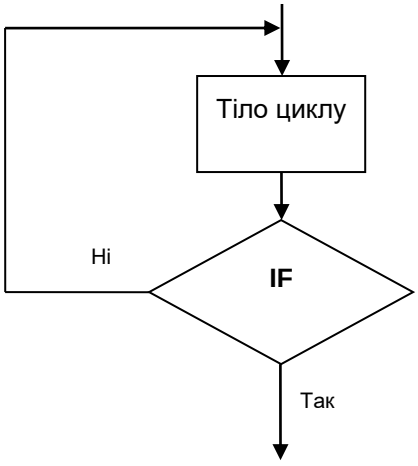
Такі задачі можна описати за допомогою алгоритмів розгалуженої структури. У таких алгоритмах вибір напряму продовження обчислення здійснюється за підсумками перевірки заданої умови. Розгалужені процеси описуються за допомогою оператора IF (умовного оператора).



Циклічні обчислювальні процеси. Для вирішення багатьох задач характерні багатократні повторення окремих ділянок обчислень. В таких випадках застосовуються алгоритми циклічної структури (циклічні алгоритми). Цикл – послідовність команд, яка повторюється до тих пір, поки не буде виконано задану умову. Циклічний опис багато разів повторюваних процесів значно знижує трудомісткість написання програм.

Існують дві схеми циклічних обчислювальних процесів.





Особливістю першої схеми є те, що перевірка умови виходу з циклу проводиться до виконання тіла циклу. В тому випадку, якщо умова продовження циклу не виконується,

тіло циклу не виконається жодного разу. Особливістю другої схеми є те, що цикл виконується хоча б один раз, оскільки перша перевірка умови виходу з циклу здійснюється після того, як тіло циклу виконане. Існують цикли з відомим числом повторень та ітераційні цикли. При ітераційному циклі вихід з тіла циклу, як правило, відбувається при досягненні заданої точності обчислення.

5. Огляд сучасних засобів, мов та технологій програмування.

Складовою частиною загального (системного) програмного забезпечення є системи програмування з відповідними алгоритмічними мовами. Системи програмування призначені для вдосконалення процесу розробки й налагодження програм. Система програмування включає у свій склад:

- вхідну мову системи програмування (званий також вихідною мовою);
- транслятор, що забезпечує переклад (трансляцію) програми з вхідної мови системи на внутрішній (машинний) мову;
- бібліотеку стандартних, найбільш часто використовуваних підпрограм (наприклад, сортування інформації, різного роду вбудованих функцій і т.п.), що підключаються в процесі підготовки програм до виконання, а також відповідну документацію.

Мови програмування, або алгоритмічні мови, класифікуються:

- за ступенем їхньої залежності від обчислювальної машини;
- по орієнтації на сферу застосування;
- по специфіці організаційної структури мовних конструкцій і т.п.

З урахуванням залежності від ЕОМ мови програмування поділяються на: ***машинно-залежні і машинно-незалежні***. Структура і засоби машинно-залежних мов відображають (враховують) специфіку функціонування певного класу ЕОМ. При програмуванні задач за допомогою таких мов потрібне знання не тільки суті реалізованого алгоритму рішення задачі, але і технічних особливостей конкретної ЕОМ і специфіки способів написання для неї програм.

До машинно-залежних мов в першу чергу відносяться машинні мови. Машинний мова є внутрішньою мовою ЕОМ і являє собою систему інструкцій і даних, які не вимагають трансляції і можуть безпосередньо інтерпретуватися і виконуватися апаратними засобами ЕОМ. Програмування на цих мовах здійснювалося на ЕОМ першого і частково другого поколінь.

До машинно-залежних мов програмування також відносяться машинно-орієнтовані мови, основні конструктивні засоби яких також дозволяють враховувати особливості архітектури та принципів роботи певної ЕОМ або ряду ЕОМ, тобто володіють тими ж можливостями і вимогами до програмістів, що і машинні мови, але на відміну від останніх вимагають попередньої трансляції на машинну мову програм, складених з їх допомогою. До даного виду мов програмування відносяться: автокодом, мови символічного кодування і асемблери. На відміну від програмування на машинних мовах програмування на машинно-орієнтованих мовах (асемблерах) характерно і для сучасних ПК. Це пояснюється тим, що в мові асемблера допускається використання засобів, властивих мовам високого рівня. Використання мови асемблера, як правило, обмежується областю системного програмування, тобто програмуванням мікропроцесорів, розробкою операційних систем

або їх компонентів, розробкою драйверів - програм обміну інформацією між центральними і периферійними пристроями, програм ув'язки взаємодії окремих компонентів прикладних програм і т.д.

Той факт, що мови даного класу враховують специфіку організації та принципів роботи конкретних ЕОМ і допускають при програмуванні вказівку конкретних режимів роботи фізичних засобів ЕОМ, розподіл пам'яті, явне визначення зовнішніх пристроїв і т.п., відносить їх до мов "низького рівня", або мовам рівня 1:1 (тобто до мов, для яких одному операторові вхідної мови програмування відповідає один оператор машинного мови).

Машинно-незалежні мови (або мови високого рівня) не вимагають від користувача повного знання специфіки ЕОМ, на якій реалізується програма рішення задачі. Інструментальні засоби цих мов програмування дозволяють записувати програму у вигляді, що допускає її реалізацію на ЕОМ з різними типами машинних операцій, прив'язка до яких цілком покладається на відповідний транслятор. Рішення задачі на цих мовах описується в наочному, досить легко сприйманому вигляді. Для них характерні: можливість написання виразів, символічна ідентифікація змінних, виклик функцій по іменах і т.п. Завдяки цьому продуктивність програміста при складанні вихідних програм на мовах високого рівня приблизно в 10 - 15 разів вище, ніж на мові асемблера. Однак одержувані в результаті трансляції машинні програми, як правило, в 2 - 5 разів об'ємніше у порівнянні з такою ж програмою, але написаною на асемблері, і працюють в 2 - 5 разів повільніше. Швидке зростання продуктивності ЕОМ, з одного боку, і хронічна недостача програмістських кадрів, з іншого боку, послужили причиною бурхливого розвитку і застосування високорівневих мов програмування.

Відокремлений, проміжне положення між машинно-незалежними і машинно-залежними мовами займає мова С, створення якої стало результатом спроби об'єднання достоїнств, властивих мовам обох класів:

- в плані максимального використання можливостей конкретної обчислювальної архітектури (що властиво мовам низького рівня), завдяки чому програми на мові Сі компактні і працюють ефективно;
- в плані максимального використання потужних виразних можливостей сучасних мов високого рівня.

Результат такого компромісу зумовив досить складний синтаксис мови С. Мова Сі та його модифікації в даний час використовуються головним чином для створення системних і прикладних програмних продуктів, в яких вирішальне значення відводиться факторам швидкодії і мінімізації обсягів пам'яті. На мові Сі написано ядро операційної системи Unix, внаслідок чого її легко можна було змінювати і модернізувати (а це спрощує процес її перенесення з однієї обчислювальної системи на іншу, при цьому 95% вихідного програмного тексту операційної системи залишається незміненим).

Машинно-незалежні мови класифікуються на процедурно-орієнтовані та проблемно-орієнтовані. Процедурно-орієнтовані (універсальні) мови ефективні для опису алгоритмів вирішення широкого класу задач. З мов цього класу найбільш відомі: Фортран, Кобол, ПЛ / 1, Бейсик, Паскаль, Ада.

Проблемно-орієнтовані мови призначені для опису процесів обробки інформації в більш вузької, специфічної області. Найбільш відомими мовами цієї групи є: РПГ, Лісп, АПЛ, GPSS.

Основна перевага алгоритмічних мов високого рівня - можливість опису програм вирішення задач у формі, максимально зручній для сприйняття людиною. Але так як кожне сімейство ЕОМ має свій власний, специфічний внутрішній (машинний) мову і може виконувати лише ті команди, які записані на цій мові, то для перекладу вихідних програм на машинну мову використовуються спеціальні програми-транслятори. Робота всіх трансляторів будується по одному з двох принципів: ***інтерпретація*** чи ***компіляція***.

Інтерпретація передбачає пооператорну трансляцію і наступне виконання трансльованого оператора вихідної програми. У зв'язку з цим можна відзначити два недоліки методу інтерпретації: по-перше, інтерпретує програма повинна знаходитися в пам'яті ЕОМ протягом усього процесу виконання вихідної програми, тобто займати певний об'єм пам'яті, по-друге, процес трансляції одного і того ж оператора повторюється стільки разів, скільки разів повинна виконуватися ця команда в програмі, що різко знижує продуктивність роботи програми. Незважаючи на зазначені недоліки, транслятори-інтерпретатори отримали достатнього поширення, оскільки вони підтримують діалоговий режим, що особливо зручно при розробці та налагодженні вихідних програм. Крім того, інтерпретатори легше розробляти, і вони обходяться дешевше, ніж компілятори з того ж мови.

У разі багаторазового розв'язування задачі, коли швидкодія роботи обчислювальної системи має істотне значення, доцільно використовувати інший принцип - компіляцію. При компіляції процеси трансляції і виконання розділені в часі: спочатку вихідна програма повністю перекладається на машинний мову (після чого наявність транслятора в оперативній пам'яті стає непотрібним), а потім відтранслювати програма може багаторазово виконуватися. Отже, для однієї і тієї ж програми трансляція методом компіляції забезпечує більш високу продуктивність обчислювальної системи при скороченні необхідної оперативної пам'яті. Велика складність в розробці компілятора в порівнянні з інтерпретатором з того ж самого мови пояснюється тим, що компіляція програми включає дві дії: аналіз, тобто визначення правильності запису вихідної програми відповідно до правил побудови мовних конструкцій вхідної мови, і синтез - генерування еквівалентної програми в машинних кодах. Трансляція методом компіляції вимагає неодноразового "перегляду" трансльованій програми, тобто транслятори-компілятори є багатопрохідних: при першому проході вони перевіряють коректність синтаксису мовних конструкцій окремих операторів незалежно один від одного, при наступному проході - коректність синтаксичних взаємозв'язків між операторами і т.д. Отримана в результаті трансляції методом компіляції програма називається об'єктним модулем, який являє собою еквівалентну програму в машинних кодах, але не "прив'язану" до конкретними адресами оперативної пам'яті. Тому перед виконанням об'єктний модуль повинен бути оброблений спеціальною програмою операційної системи (редактором зв'язків) і перетворений в завантажувальний модуль, тобто програмний модуль з відносними адресами.

Сьогоднішні технології програмування успішно поєднують обидва підходи. З метою кросплатформності програмного забезпечення процес виконання написаного програмного коду відбувається у два етапи:

- компіляція програмного коду у проміжний код, так званий – байт-код;
- інтерпретація скомпільованого коду засобами спеціального програмного середовища – віртуальної машини.

Ця ідеологія була започаткована наприкінці минулого століття фірмою Sun Microsystems у технології ***Hot Java***, що побачила світ разом із новітньою мовою

програмування ***Java***. Подальшого розвитку цей підхід зазнав з виходом платформи ***.net*** від компанії Microsoft, яка об'єднала декілька мов програмування, включно з C++, C# та Visual Basic.

Поряд з універсальними процедурно-орієнтованими мовами стали створюватися проблемно-орієнтовані мови програмування, призначені для опису процесів обробки інформації в якій-небудь вузькій (специфічної) області, в яких рішення задачі в більшій мірі зосереджувалася на проблемі, що необхідно отримати в результаті, а проблема , як це необхідно зробити, в більшій чи меншій мірі знімалася з програміста. Серед цих мов найбільш відомими є: РПГ - мову для генерації звітів, Лісп - мова для обробки списків, GPSS - мова для моделювання, АПЛ - мову для статистичної обробки масивів.

Особливе місце серед мов програмування займають функціональні мови, зокрема Пролог (PROLOG - PROgram-ming in LOGic - логічне програмування), запропонований А.Калмерое в 1978 р., що є мовою логічного програмування, що відноситься до мов п'ятого покоління. Головне призначення мови - розробка інтелектуальних програм і систем. Пролог - це мова програмування, створена спеціально для роботи з базами знань, заснованими на фактах і правилах (одного з елементів систем штучного інтелекту). У мові реалізований механізм повернення для виконання зворотного ланцюжка міркувань, при якому передбачається, що деякі висновки або укладення істинні, а потім ці припущення перевіряються в базі знань, що містить факти і правила логічного висновку. Якщо припущення не підтверджується, виконується повернення і висувається нове припущення.

Мовні засоби СУБД призначені в першу чергу для розробки прикладних програм розв'язання задач економічного управління, інформація для яких зберігається і підтримується за допомогою баз даних. Синтаксис мови програмування в середовищі СУБД мало чим відрізняється від синтаксису високорівневих мов програмування, в зв'язку з чим зазначені програмно-інструментальні засоби орієнтовані в основному на професійних програмістів, хоча наявність розвинених засобів підказки і допомоги (у вигляді прикладів, які демонструють використання окремих мовних конструкцій) значно полегшує роботу досить широкого кола користувачів. Sequel (Structured English QUEry Language) і його вдосконалений варіант ***SQL*** - мови маніпулювання даними, засновані на обчисленні відносин. Використовуються в реляційних СУБД в якості мови запитів до баз даних і мови програмування задач обробки даних.

З розвитком комп'ютерних мереж, збільшенням обчислювальної потужності комп'ютерів і їх ресурсів виникла потреба в інтерпретуються мовою, що дозволяє отримувати багатоплатформності обчислювальне середовище шляхом перетворення з його допомогою програм, написаних на інших мовах програмування (при незначному зниженні їх продуктивності). Найбільш близько до реалізації подібного мови підійшла технологія мови Java (а точніше її частина - байт-код). Саме його розробка і використання складають принципова відмінність, яке виділяє мову Java серед інших мов програмування високого рівня. Об'єктно-орієнтована мова Java (розроблена на базі мови Сі + +) призначений для створення надійних, переносите, розподілених мережевих програмних додатків, які працюють в різних багатовіконний системах в умовах архітектури клієнт-сервер, а також для адміністраторів мережі, що використовують Java-додатки для поліпшення інтерактивних якостей Web-серверів.

Популярність протоколу http та WWW покликала до життя цілу низку новітніх технологій програмування, разом з використаними в них мовами, розробленими спеціально для Web. Це стосується перелічених у попередній лекції мов ***Perl, PHP, JavaScript, Ruby on Rails.***

Інтенсивна розробка штучного інтелекту на базі нейронних мереж посприяла, окрім бібліотек практично для кожної сучасної мови програмування, виникненню спеціалізованої мови програмування ***Pyton***.

Виникнення нових мов програмування продовжується і далі. Наприклад, компанія Google, випустивши операційну систему для керування мобільними пристроями ***Android,*** забезпечила програмістів засобами розробки програмного забезпечення на популярній мові Java. Проте згодом спеціально для цих потреб компанією JetBrains була розроблена нова мова програмування ***Kotlin***, якій Google надала перевагу у розробці ПЗ для ОС Android. Крім того у самій Google розроблено нову мову програмування Dart, що позиціонується як мова програмування загального призначення.

Лекція 15. Основи алгоритмізації та програмування. Алгоритмічна мова програмування С.

1. *Алгоритмічна мова програмування С, історія виникнення та розвитку.*
2. *Структура найпростішої програми на мові С.*
3. *Основні правила написання програми на С/С++.*
4. *Базові алгоритмічні конструкції С.*

1. Алгоритмічна мова програмування С, призначення та особливості.

Мова програмування С є поширеним інструментом розробки програмних засобів як системного, так і прикладного характеру. Історію її появи пов'язують із співробітником американської фірми Bell Labs Денізом Рітчі, хоча його дітищу - мові С передували розробки та інших системних програмістів (М. Річардс - система BCPL, К. Томпсон - мова В). Поштовхом до появи різних програмних засобів, що полегшували б роботу системних програмістів, стало створення операційної системи Unix для комп'ютера PDP-7, розпочате в 1969 році. Єдиною на той час операційною системою великого комп'ютера GE-645, що обслуговував співробітників лабораторії, була досить громіздка багатокористувальницька система Multics. К. Томпсон (до речі, один з розробників Multics) в свій час написав програму, яка моделювала рух небесних тіл. Кожен її запуск на GE-645 обходився в 75 $, а траєкторії руху видавалися в табличному вигляді.

Тоді невелика група співробітників, очолювана К. Томпсоном, вирішила створити більш зручну однокористувальницької систему на маленькому покинутому комп'ютері PDP-7 з дисплеєм. До складу цієї групи входив і Деніз Рітчі. Система Unix стала дуже популярною серед співробітників лабораторії, тому, що вона суттєво спрощувала процес проходження завдань і не вимагала від користувачів знання численних директив системи Multics. У 1970 році Д. Рітчі допоміг перенести Unix на більш потужний комп'ютер PDP-11. У процесі цієї роботи став у нагоді набір макрокоманд на мові асемблера, який спрощував програмування численних процедур. Цей набір і був покладений в основу мови С, який вдало поєднував специфіку машинних команд з елементами мови високого рівня. У 1973 році Д. Рітчі і К. Томпсон переписали ядро операційної системи Unix на мову С (до цього всі програми були написані на асемблері).

З 1974 року система Unix разом з вихідними текстами на мові С і компілятор цієї мови були передані ряду університетів. Найбільш важливу роль у подальшому розвитку системи Unix, що перетворилася з однокористувальницької в багатокористувацьку, зіграли співробітники університету Берклі. Популярність системи Unix, що збереглася до наших днів і обслуговує сьогодні більше 90% серверів, значною мірою сприяла і популярності мови С, компілятор якої входив до складу Unix.

Наступний крок у розвиток потужності та універсальності мови С в 1983 році вніс співробітник все тієї ж Bell Labs Бьорн Страуструп. Запропоновані ним розширення призвели до появи версії С++ (первинна назва - С з класами). Ці нововведення дозволили користувачам конструювати власні типи даних, включати в мову нові операції над такими даними, агрегувати дані з обробними їх функціями-методами, успадковувати і перевизначати методи в породжуваних класах.

Слід зазначити і істотний внесок в розвиток систем програмування на базі мов С, С++, внесений фірмою Borland, точніше, її засновником - Філіпом Канном. Це створення інтегрованих систем розробки, в яких вдало поєдналися засоби підготовки, зберігання, налагодження та компонування програм, які спочатку з'явилися в системі Turbo Pascal. Всі пізніші системи програмування в тій чи іншій мірі запозичили основні ідеї Ф. Канна.

На початку 90-х років XX століття мова С та її об'єктно-орієнтований «нащадок» С++ набули надзвичайної популярності в середовищі розробників програмного забезпечення. Ця популярність стала причиною наслідування синтаксичних конструкції С іншими популярними сучасними мовами програмування, такими як Java, PHP чи Javascript. Розвиток «лінійки» С-подібних мов програмування успішно піхопила компанія Microsoft випустивши кілька успішних релізів Visual C++ і увінчавши у 2000 році цей процес успішним виходом міжмовної кросплатформенної технології ***.net*** та ще одного «правонаступника» С – сучасної об'єктно-орієнтованої мови програмування С#.

2. Структура найпростішої програми на мові С

Продемонструємо приклад найпростішої програми на мові С++, яка запитує у користувача два цілочисельних значення змінних a і b, аналізує їх і виводить найбільше число.

```
#include <iostream.h>
#include <conio.h>
void main()
 {
   int a,b,max;
   cout << "a=";     //запрошення ввести значення a
   cin >> a;         //ввід значення змінної a
   cout << "b=";     // запрошення ввести значення b
   cin >> b;         // ввід значення змінної b
   if(a>b) max=a;      //якщо a>b то max=a
   else max=b;         //в протилежному випадку max=b
   cout << "max="<<max;  //вивід максимуму
   getch();     }    //зупинка до натискання клавіші
```

Перші два рядки підключають (include - включити) до тексту програми так звані заголвкові (h від header - заголовок) файли системи. У цих файлах описані системні функції і їхні аргументи. Використовуючи ці описи, компілятор перевіряє правильність виклику системних функцій. У нашому прикладі програма використовує системні функції введення (cin >>) і виводу (cout <<), опису яких знаходяться в заголовному файлі iostream.h, а також функцію очікування натискання будь-якої клавіші (getch), опис якої знаходиться в заголовному файлі conio.h. Назви заголовків файлів найчастіше утворюються від будь-яких абревіатур англійських слів, їх корисно навчитися розуміти, а не запам'ятовувати. У нашому прикладі: io - input / output (ввід / вивід), stream (потік), con (console - пульт оператора, тобто клавіатура і дисплей). Крім цих двох файлів ми будемо використовувати бібліотеки stdio.h для фор матованого вводу та виводу даних та math.h – бібліотеку вбудованих математичних функцій.

Третій рядок містить заголовок функції main. Функція з такою назвою повинна бути в кожній програмі на мовах С та С++. Саме з неї починається виконання програми, вона - головна (саме так перекладається службове слово main). Службове слово int (від integer - цілий) повідомляє, що результатом роботи функції main повинно бути ціле число, (за яким

операційна система, запустивши програму main, може "вирішити", правильно чи неправильно завершилася робота програми). За загальноприйнятою угодою нульове значення, що повертається функцією main, свідчить про нормальне завершення роботи програми. Службове слово void (дослівно - порожнеча), зазначене в круглих дужках, повідомляє, що у функції main аргументи відсутні.

Текст програми (тіло функції) розміщується в фигурних дужках (4-та 15-та стрічки). В п’ятій стрічці оголошено три змінні з іменами a, b и max, які можуть набувати цілочисельних значень. Окрім типу int в C/C + + можна використовувати і інші цілі типи даних, що відрізнятимуться діапазонами допустимих значень. Для десяткових дробів у найпростішому випадку використовуватимемо тип float (короткое дійсне, 4 байти) або double.

Шоста стрічка є першою стрічкою програми, яка виконує якусь дію - вона виводить на дисплей повідомлення, що складається з двох символів (a =). Текст повідомлення обмежують подвійними лапками. Рядок 7 організовує зупинку роботи програми до тих пір, поки користувач не набере на клавіатурі якесь число і натисне клавішу Enter. Отримане значення буде сприйняте лише, якщо воно є цілим числом, і його буде направлено в змінну a. Подібним чином в рядках 8 і 9 буде організовано ввід значення числової змінної b.

У десятій стрічці організовано порівняння поточних значень змінних a і b. Якщо значення змінної a більше, то воно присвоюється змінної max, в іншому випадку в змінну max заноситься значення b.

Рядок 12 виводить на дисплей два повідомлення - текстове (max =) і числове (значення змінної max).

Звернення до функції getch (рядок 13) призводить до затримки на екрані повідомлення програми до тих пір, поки користувач не натисне будь-яку клавішу (getch - від get character, дай символ).

Остання виконувана рядок з номером 14 повертає керування операційній системі (return - повернутися) і видає в якості значення функції нульовий результат.

3. Основні правила написання програми на C/C + +

- - Якщо програма звертається до якихось системних функцій, то в перших її рядках обов'язково повинна бути вказівка про підключення відповідних заголовкових файлів.
- - Програма може містити більш ніж одну функцію, але серед них обов'язково має бути присутня функція з іменем main.
- - Кожен рядок програми, що містить оголошення чи виконувану дію, закінчується крапкою з комою.
- - Тіло функції обов'язково обмежуєполягає у фігурні дужки (в Паскалі аналогічні функції виконували операторні дужки - begin і end).

Стандартний вигляд оператора присвоєння. Оператор присвоювання є найбільш поширеним засобом, що дозволяє змінити значення змінної під час роботи програми. У напростішому випадку він має формат:

```
namev = expression;
```

Тут

namev – ім’я змінної
expression – вираз, значення якого буде присвоєно змінній namev.

Якщо змінна namev відноситься до категорії числових даних, то результат обчислення виразу теж повинен бути числовим. Типи змінної namev і значення виразу вираження при цьому можуть не збігатися. Про відповідне перетворенні машинних форматів система подбає сама. Але не забувайте, що при перетворенні речового значення в цілочисельне дробова частина виразу (якою б вона не була) буде відкинута. Таким чином, про округлення ви повинні подбати самі. У тих випадках, коли тип виразу допускає більш широкий діапазон представлення даних, можлива втрата точності або переповнювання діапазону, передбаченого для змінної namev. У таких випадках компілятор видає повідомлення про можливі наслідки, і на такі повідомлення програміст зобов'язаний звернути увагу.

Якщо декільком змінним необхідно присвоїти значення одного і того ж вирази, то оператор присвоювання допускає розширений формат:

```
v1=v2=v3=exp;
```

Використовуючи таку форму множини присвоєння, не намагайтеся писати оператори, результат виконання яких може по різному тлумачитися різними користувачами. наприклад:

```
i=5;
  i=a[i]=4;
  j=6;
  a[j]=j=2;
```

Результат таких дій залежить від алгоритму перегляду рядка програми і послідовності виконання присвоювання - зліва направо або справа наліво. Намагайтеся уникати подібних ситуацій, тому у разі перенесення програми в іншу систему програмування можна отримати несподіваний результат.

Змінні, що використовуються в програмі, обов'язково повинні бути оголошені до їх використання в тих чи інших виконуваних операторах. На відміну від мови Pascal алгоритмічні мови С, С + + дозволяють вводити такі описи не тільки на початку програмних одиниць (функцій), але і по ходу формування програми:

```
int main(void)
{ int i,j;
.....
  s=0;
  for(int k=0; k<10; k++)
.....
```

У наведеному прикладі змінні і і j оголошені на початку функції, а оголошення змінної k зустрілося в операторі циклу. За стандартом мови С + + місце оголошення змінної визначає сферу її дії. Змінні, описані на початку функції (такі як і та j в наведеному вище прикладі), вважаються локалізованими в даній функції і можуть бути використані в будь-якій частині тіла цієї функції. Повторне оголошення змінних з такими ж іменами вважається помилкою (дублювання імен змінних). На відміну від цього дія

змінних, оголошених в деякому внутрішньому блоці функції (у нашому прикладі змінна k оголошені в циклі for), поширюється тільки на час роботи цього блоку. Тобто після виходу з циклу пам'ять, виділена під змінну k, повертається системі, і без повторного переоб'явленія цієї змінної користуватися вже не можна.

Оголошення змінної можна поєднати з присвоєнням їй початкового значення (ініціалізацією):

int x = 18, y = -5;

float a = 5F;

Для оголошення іменованих констант зазвичай використовують наступну конструкцію:

const [tc] namec = value;

тут

• tc - необов'язковий тип константи (за замовчуванням tc = int);

• namec - ім'я константи;

• value - значення константи.

Іноді для задання таких же констант вдаються до механізму найпростішої макропідстановки:

```
#define Nmax 100
#define eps 1e-6
```

Це означає, що перед трансляцією програми компілятор (точніше, прекомпілятор) прогляне її текст і всюди, де буде зустрінуте поєднання символів Nmax, його замінять на число 100, а поєднання символів eps на число 1e-6. Результат буде тим же самим, але робота по макропідстановки пов'язана з більш помітними витратами часу.

4. Базові алгоритмічні конструкції С.

Умовний оператор. За допомогою умовного оператора в програмі можна створити дві альтернативні ланцюжки операторів, виконання кожній з яких відбувається в залежності від істинності чи хибності заданої умови:

if(умова) {S1; S2;...} [else {Q1;Q2;...}]

Якщо умова, зазначена в круглих дужках істинно, то виконується ланцюжок операторів S1, S2, ... В іншому випадку виконується ланцюжок операторів Q1, Q2, ... Альтернативна гілку разом із службовим словом else може бути відсутнім. І тоді мова йде про виконання або обході ланцюжка операторів S1, S2, ...

Прості умови задаються за допомогою операцій порівняння:

```
if(a>b) cout<<a; else cout<<b;
  if(x>=0) y=sqrt(x);
```

Більш складні умови можуть складатися з елементарних співвідношень за допомогою логічних операцій "АБО" (| |), "І" (&&), "НЕ" (!). Наприклад, перевірка належності x діапазону [a, b] виглядає наступним чином:

```
if(a<=x && x<=b) ...
```

В якості логічного умови іноді можна зустріти в круглих дужках звичайне арифметичне вираз:

```
if(a)...
```

У мові С логічні змінні відсутні. Замість цього діє угода про те, що брехня еквівалентна нульового значення числового виразу, а істина - не нульового значення. Тому наведена вище перевірка еквівалентна співвідношенню:

```
if(a != 0)...
```

Оператори циклу. Оператори циклу дозволяють організувати повторне виконання фрагмента програми до тих пір, поки не виконається деякий умова. Звичайно, аналогічні дії можуть бути оформлені з використанням умовних операторів (if. ..) і безумовних переходів (goto) в початок повторюваного фрагменту. Проте використання операторів циклу дозволяє зробити логіку роботи програми більш прозорою і трохи більш ефективною за рахунок застосування спеціальних машинних команд, які вставляє компілятор.

Наиболее часто применяемая конструкция цикла использует оператор for:

```
for(S1; C; S2) { Q1; Q2;...}
```

тут

- S1 - дія, яка виконується перед повторенням групи операторів Q1, Q2, ..., утворюють тіло циклу;
- C - умова, при виконанні якого тіло циклу повторюється. Якщо умова C не виконано, то тіло циклу обходиться і цикл завершується;
- S2 - дія, яка виконується слідом за останнім оператором тіла циклу. Після цього управління передається на перевірку умови завершення циклу.

Одна з форм оператора for дозволяє організувати повторення тіла циклу заданий число раз, змінюючи при кожному повторенні значення деякої керуючої змінної (іноді таку змінну називають лічильником циклу або індексом циклу):

```
s=0;
  for(j=0; j<10; j++)
    s=s+a[j];
```

У наведеному прикладі тіло циклу складається з єдиного оператора, який можна не укладати в фігурні дужки. Перед початком циклу в змінну j заноситься 0 і оскільки це значення менше 10, то до вмісту змінної s додається значення першого елемента масиву a [0]. Після першого завершення тіла циклу до лічильника j додається 1 і тіло циклу

повторюється. Так продовжується до тих пір, поки значення лічильника не збільшиться до 10, при якому умова повторення циклу вже не виконується.

Якщо лічильник циклу використовується тільки в тілі циклу і значення цієї змінної за межами циклу зберігати не потрібно, то стандарт C + + дозволяє визначити таку суперлокальную змінну безпосередньо в операторі for:

```
for(int j=0; j<10; j++)
```

Значення лічильника може не тільки збільшуватися при кожній ітерацій, але і зменшуватися:

```
s=0;
  for(j=9; j>=0; j--)
    s=s+a[j];
```

В цьому прикладі масив a сумується, починаючи з останнього елемента. Цикл з лічильником в зворотньому порядку можна організувати ще й так:

```
count=20;
  for(s=0; count; count--)
    s += a[count];
```

У цьому прикладі умовою повторення циклу є додатне значення лічильника count, яке сприймається як "істина". Як тільки вміст лічильника стане рівним нулю (тобто "брехні"), цикл завершить свою роботу.

На відміну від мови Паскаль, де оператор for допускає тільки збільшення або зменшення лічильника циклу тільки на 1, оператор for в мовах C, C++ дозволяє змінювати значення керуючої змінної довільним чином. Ця змінна може бути дійсного типу, і її значення може змінюватися не тільки за законом арифметичної прогресії, але і будь-яким як завгодно складним способом. Наприклад, можна підсумувати лише елементи масиву з парними індексами:

```
s=0;
  for(j=0; j<10; j+=2)    s=s+a[j];
```

Дія S1, віконуване при вході в цикл, Може складаються НЕ Тільки з єдиного оператора, засілає в Лічильник циклу Початкове значення. ЯКЩО потрібно віконаті декілька Операторів, то їх розділяють комами (список Дій, аналогічній екзотічної формі оператора прісвоювання). У наведенні Вище прикладах очистку змінної s теж можна Було внести в оператор заголовка циклу:

```
for(s=0,j=0; j<10; ++j)
   s=s+a[j];
```

Аналогічний список з декількох операторів можна використовувати і в якості дії S2, виконуваного слідом за останнім оператором тіла циклу. У наведеному нижче прикладі оператор циклу обчислює n-й член послідовності чисел Фібоначчі:

```
for(x=1,y=1,i=2; i<n; z=x+y, x=y, y=z, i++);
```

Друга конструкція оператора циклу починається зі службового слова while (від англ. – поки):

```
while(умова) {Q1; Q2;...}
```

Вхід в такий цикл починається з перевірки умови, заданого в круглих дужках. Якщо ця умова істинно, то тіло циклу (оператори Q1, Q2, ...) виконується. У разі невиконання умови тіло циклу обходиться. За допомогою оператора while теж можна організувати підсумовування елементів масиву:

```
s=0;  i=0;
  while(i<10)
    { s=s+a[i]; i++; }
```

Погодьтеся, що цей варіант виглядає менш витончено, ніж його аналог з оператором for, хоча тут присутні ті ж елементи, які раніше фігурували в заголовку циклу for. Однак в інших ітераційних процесах використання циклу while може виявитися більш доречним. Наприклад, ітераційний схема обчислення кореня квадратного з x (аналогічні схеми можна написати і для коренів інших ступенів) має наступний вигляд:

```
yn+1=0.5*(yn +x/yn)
Для її реалізації можна скористатися циклом:
y=x; //початкове наближення
  while(fabs(x-y*y)>1e-6)
    y=0.5*(y+x/y);
```

Третя конструкція циклу починається з оператора do (від англ. - Виконати) і завершується перевіркою умови продовження циклу while:

```
y=x; // початкове наближення
  do
    y=0.5*(y+x/y);
  while(fabs(x-y*y)>1e-6)
```

Відмінність конструкції do - while від двох попередніх операторів циклу полягає в тому, що тут тіло циклу виконується, принаймні, один раз. Цей цикл іноді супроводжують терміном з постусловієм - тобто перевірка умови продовження циклу проводиться після виконання його тіла. На відміну від цього два попередні циклу розпочинаються з перевірки заданої умови (цикли з передумовою) і в деяких випадках тіло таких циклів може жодного разу не виконатися.

Всі конструкції операторів циклу допускають вкладення інших циклів в свої тіла. Наприклад:

```
for(i=0; i<n; i++)
  for(j=0; j<n; j++)
    {for(d=0,k=0; k<n; k++)
       d += a[i][k]*b[k][j];
     c[i][j]=d;
    }
```

Література

1. Грицунов О.В. Інформаційні системи та технології. Навчальний посібник. Харків, 2010р., 222 с.
2. Клименко О. Ф., Шарапов О. Д., Головко Н. Р. Інформатика та комп'ютерна техніка.- К.: КНЕУ, 2005.- 534 с.
3. Рзаєв Д. О., Шарапов О. Д., Ігнатенко В. М., Дибкова Л. М. Інформатика та комп'ютерна техніка: Навч.-метод. посібник для самост. вивч. дисц.- К.: КНЕУ, 2002.- 486 с.
4. Кучерява Т.О., Сільченко М.В., Шабаліна І.В. Інформатика та комп'ютерна техніка: активізація навчання. Практикум для індивідуальної роботи студентів. - К.: КНЕУ, 2006.- 448 с.
5. Глинський Я.М. Інформатика. Практикум з інформаційних технологій. Навч. посібник – Тернопіль: Підручники і посібники, 2014 – 304 с.
6. Основи програмування на C/C++. Конспект лекцій з курсу «Основи інформатики і програмування, частина 2» спеціальності 105 – «Прикладна фізика та наноматеріали» для першого (бакалаврського) рівня освіти/ Укл.: Ментинський С.М., 2016. – 140 с.
7. І. Демків, П. Каленюк, І. Клюйник, І. Кравець, Р. Петрович. Основи роботи в середрвищах Microsoft Excel 2000 та Microsoft Access 2000. Лекції та завдання до лабораторних робіт. – Львів: Каменяр, 2007, - 262 с.
8. OpenOffice.org: Теория и практика / И. Хахаев, В. Машков,Г. Губкина и др. М. : ALT Linux ; БИНОМ. Лаборатория знаний, 2008. 319с. : (Библиотека ALT Linux).
9. Троценко В.С., Чаленко П.Й., Старовський А.Б. Техніка програмування мовою Сі. К.:Либідь, 1993.

Інформаційні інтернет-ресурси

1. http://vns.lp.edu.ua/
2. http://labs.starbasic.net/
3. http://wiki.openoffice.org/
4. http://wiki.documentfoundation.org
5. http://help.libreoffice.org
6. http://cppstudio.com/uk/
7. https://msdn.microsoft.com/ru-ru/library/60k1461a.aspx

НАВЧАЛЬНЕ ВИДАННЯ

Л.Б. Гнатів, С.М. Ментинський, Я.М. Пелех, Є.М.Федюк

Основи інформатики та інформаційних технологій.

Конспект лекцій

для студентів бакалаврату спеціальності
105 Прикладна фізика та наноматеріали

навчальний посібник
з курсу
«Основи інформатики і програмування, частина 1»