Міністерство освіти і науки України
Рівненський державний гуманітарний університет
Кафедра інформаційних технологій та моделювання

Бабич Степанія Михайлівна
Шліхта Ганна Олександрівна

Конспект лекцій до курсу

Програмне забезпечення обчислювальних систем

Рівне – 2026

УДК 004.4(075.8)
Б 12

Затверджено на засіданні кафедри інформаційних технологій та моделювання
Протокол від «_27_»_____січня_____2026 року № _1_

Схвалено навчально-методичною комісією факультету математики та інформатики
Протокол від «_27_»_____січня_____2026 року № _1_

Укладачі:

Бабич С. М., кандидат технічних наук, доцент кафедри інформаційних технологій та моделювання РДГУ;
Шліхта Г. О., доктор педагогічних наук, професор кафедри інформаційних технологій та моделювання РДГУ.

Рецензенти:

Ляшук Т. Г., кандидат фізико-математичних наук, старший викладач кафедри інформаційних технологій та моделювання РДГУ**;**

Шевцова Н. В., кандидат фізико-математичних наук, старший викладач кафедри інформаційних технологій та моделювання РДГУ**.**

Бабич С. М., Шліхта Г. О. Конспект лекцій до курсу «Програмне забезпечення обчислювальних систем» [Електронний ресурс]. Рівне : РДГУ, 2026. 80 с.

Призначається для студентів різних спеціальностей, які вивчають основи організації, функціонування та використання програмного забезпечення сучасних обчислювальних систем.

© Бабич С. М., Шліхта Г. О., 2026
© РДГУ, 2026

Зміст

Передмова

Сучасний розвиток інформаційних технологій визначає ключову роль програмного забезпечення у функціонуванні обчислювальних систем, комп'ютерних мереж та цифрових сервісів. В умовах динамічних змін у галузі ІТ зростає потреба у фахівцях, які володіють цілісним розумінням структури, принципів функціонування та практичного використання програмних засобів різного призначення.

Курс «Програмне забезпечення обчислювальних систем» спрямований на формування у студентів системних знань про види програмного забезпечення, його архітектуру, принципи розроблення, експлуатації та інтеграції. Особливу увагу приділено практичним аспектам роботи з прикладними програмами та інструментами для обробки текстових, табличних і графічних даних, а також сучасним тенденціям розвитку програмного середовища.

Пропонований посібник містить конспекти лекцій, що висвітлюють теоретичні основи курсу, тестові завдання для самоконтролю та підготовки до підсумкового оцінювання, а також список рекомендованих джерел, який допоможе студентам поглибити знання з окремих тем.

Матеріали посібника можуть бути використані як під час аудиторних занять, так і для самостійної роботи студентів. Вони орієнтовані на здобувачів спеціальностей «Інженерія програмного забезпечення», «Комп'ютерні науки», а також усіх, хто прагне систематизувати знання про сучасне програмне забезпечення.

Тема. Основи програмного забезпечення: класифікація, функції, еволюція

- ✓ Поняття програмного забезпечення (ПЗ), його структура.
- ✓ Класифікація: прикладне, системне, інструментальне, службове.
- ✓ ПЗ за ліцензією.
- ✓ Життєвий цикл програмного забезпечення.
- ✓ Етапи розвитку ПЗ
- ✓ Основи безпеки програмного забезпечення та інформаційна безпека.

Конспект лекції

Вступ

Обчислювальна система – це сукупність технічних та програмних засобів, які забезпечують збирання, зберігання, обробку і передачу інформації. Такі системи виконують різноманітні обчислювальні, аналітичні та керуючі функції в усіх сферах діяльності людини.

Приклади обчислювальних систем: персональний комп'ютер (ПК), серверна система, суперкомп'ютер, мобільний пристрій, вбудовані (embedded) системи, промислові контролери, комп'ютерні мережі тощо.

Комп'ютерна система зазвичай розглядається як конкретний різновид обчислювальної системи загального призначення – наприклад, персональний комп'ютер або ноутбук. Таким чином, комп'ютерна система є підмножиною ширшого поняття обчислювальних систем.

Будь-яка обчислювальна система складається з апаратного забезпечення (hardware)**,** програмного забезпечення (software) та організаційного або інформаційного забезпечення. Якщо апаратна частина виконує фізичні операції, то саме **програмне забезпечення** визначає логіку їхньої роботи, забезпечує взаємодію між користувачем і технічними засобами, а також реалізує алгоритми обробки даних. Отже, програмне забезпечення є ключовим компонентом обчислювальної системи, який забезпечує її функціональність, гнучкість і ефективність.

Мета лекції: сформувати в студентів цілісне уявлення про програмне забезпечення, його структуру та класифікацію; ознайомити з основними видами ліцензій та принципами відкритого і комерційного ПЗ; розкрити особливості життєвого циклу програмного забезпечення та основні підходи до його розвитку; сформувати базові знання з безпеки ПЗ та інформаційної безпеки.

Поняття програмного забезпечення (ПЗ), його структура

Програмне забезпечення (ПЗ) – це сукупність програм, процедур і супровідної документації, необхідних для функціонування комп'ютерної системи. –

Структура ПЗ:

- *системне ПЗ* – забезпечує роботу апаратного забезпечення та взаємодію з користувачем;
- *прикладне ПЗ* – виконує конкретні завдання користувача (офісні пакети, браузери, редактори);
- *інструментальне ПЗ* – призначене для розробки та тестування програм (компілятори, IDE, бібліотеки);
- *службове ПЗ* – утиліти для обслуговування й налаштування системи (архіватори, антивіруси).

Класифікація ПЗ

Системне ПЗ

- Операційні системи (Windows, Linux, macOS).
- Драйвери пристроїв.
- Утиліти для підтримки системи.

Прикладне ПЗ

- Загального призначення (текстові процесори, електронні таблиці).
- Спеціалізоване (CAD-системи, бухгалтерські програми).

Інструментальне ПЗ

- Засоби програмування (мови, компілятори).
- Середовища розробки (Visual Studio, Eclipse).

Службове ПЗ

- Антивіруси, системні оптимізатори, архіватори.

ПЗ за ліцензією

- *Комерційне* – продається за ліцензією, користувач платить за використання (Microsoft Office).
- *Умовно-безкоштовне (shareware)* – дається для ознайомлення на певний час чи з обмеженим функціоналом.
- *Вільне (freeware)* – безкоштовне для використання, але без відкритого коду.
- *Open Source* – програмне забезпечення з відкритим вихідним кодом (Linux, Firefox).

- *GNU / GPL* – ліцензії, що гарантують свободу використання, модифікації та поширення програмного забезпечення.

Життєвий цикл програмного забезпечення

Життєвий цикл ПЗ – сукупність процесів від ідеї до завершення використання. Основні етапи:

1. *Аналіз вимог* – визначення завдань, які має виконувати програма.
2. *Проєктування* – створення архітектури та структури системи.
3. *Реалізація (програмування)* – написання коду.
4. *Тестування* – перевірка працездатності та усунення помилок.
5. *Впровадження та експлуатація* – передача користувачу, підтримка.
6. *Супровід і модернізація* – оновлення, виправлення помилок.
7. *Завершення використання* – заміна новим ПЗ або відмова від використання.

Еволюція ПЗ

Основні етапи:

1. початковий період (1940–1950-ті роки);
2. ера високорівневих мов (1950–1960-ті);
3. операційні системи (1960–1970-ті);
4. персональні комп'ютери (1980-ті);
5. мережеві технології (1990-ті);
6. хмарні сервіси та мобільні додатки (2000–2010-ті).

Основи безпеки програмного забезпечення та інформаційна безпека.

Безпека програмного забезпечення (Software Security)

Це властивість самої програми бути захищеною від несанкціонованого доступу, помилок і атак, які можуть вплинути на її роботу (здатність програми забезпечувати цілісність, конфіденційність і доступність даних під час роботи).

Мета – зробити програму стійкою до загроз.

Основні уразливості: помилки коду, шкідливі програми, хакерські атаки.

Основний акцент: на тому, як створити і підтримувати безпечне ПЗ.

Методи захисту: шифрування даних, автентифікація користувачів, контроль доступу, оновлення та виправлення уразливостей.

Приклади:

- код написано без уразливостей (SQL-ін'єкції, переповнення буфера);
- доступ до функцій програми обмежений правами користувачів;

- програма правильно перевіряє дані, введені користувачем;
- оновлення усувають вразливості.

Отже, безпека ПЗ ≈ “якість і захищеність самої програми”.

Інформаційна безпека (Information Security)

Це ширше поняття за безпеку ПЗ, що охоплює захист усіх інформаційних ресурсів організації чи системи: даних, мереж, пристроїв, користувачів, програм тощо.

Мета – захистити інформацію від несанкціонованого доступу, змін, втрат чи знищення (забезпечити цілісність, конфіденційність та надійність програмних систем).

Основний акцент: на тому, як захистити дані, незалежно від того, де вони знаходяться – у ПЗ, на сервері чи в мережі.

Приклади:

- використання антивірусів, міжмережевих екранів (фаєрволів);
- політика доступу до даних;
- резервне копіювання;
- захист мережевих з’єднань (VPN, HTTPS).

Отже, інформаційна безпека ≈ “захист усієї інформаційної системи”.

Таблиця. *Порівняння безпеки ПЗ та інформаційної безпеки*

Ознака	**Безпека програмного забезпечення**	**Інформаційна безпека**
Об’єкт захисту	Програма, її код, логіка, функції	Дані, користувачі, мережі, системи
Мета	Захист програми від уразливостей	Захист інформації від витоку чи втрат
Рівень	Технічний (код, архітектура, оновлення)	Організаційний, технічний і правовий
Засоби	Безпечне програмування, тестування, патчі	Антивіруси, шифрування, політики доступу
Приклади	Виправлення помилки, що дозволяє зламати систему	Використання паролів, бекапів, фаєрволів

Отже, безпека ПЗ – частина інформаційної безпеки, яка зосереджена саме на захисті програм від уразливостей**,** а не всієї інформаційної інфраструктури.

Висновки

Програмне забезпечення є невід'ємним компонентом обчислювальної системи, що забезпечує реалізацію її функцій і взаємодію між апаратною частиною та користувачем. Воно має складну структуру і класифікується за призначенням (системне, прикладне, інструментальне, службове) та за типом ліцензування (вільне, комерційне, Open Source тощо).

Еволюція програмного забезпечення відображає розвиток інформаційних технологій – від простих програм до масштабних інтегрованих систем і хмарних сервісів. Важливою характеристикою сучасного ПЗ є його безпека, яка визначає надійність, цілісність і захищеність даних.

Отже, знання основ класифікації, життєвого циклу та принципів безпеки програмного забезпечення є необхідним для ефективного проєктування, розроблення й експлуатації обчислювальних систем.

Питання для самоконтролю

1. Поняття ПЗ

1. Що таке програмне забезпечення?
   a) Лише операційні системи
   b) Програми, документація і процедури для роботи комп'ютера
   c) Лише прикладні програми
   d) Тільки антивіруси

2. Класифікація ПЗ

2. До системного програмного забезпечення належить:
   a) MS Word
   b) Windows
   c) AutoCAD
   d) Chrome
3. Який вид ПЗ використовується для створення та тестування програм?
   a) Прикладне
   b) Системне
   c) Інструментальне
   d) Службове
4. До службового ПЗ можна віднести:
   a) Архіватори
   b) Операційну систему
   c) Текстовий редактор
   d) IDE

3. ПЗ за ліцензією

5. Яке ПЗ є безкоштовним, але не має відкритого коду?
   a) Open Source
   b) Freeware
   c) Shareware
   d) GNU
6. Яка ліцензія гарантує свободу використання, модифікації та поширення?
   a) GPL
   b) Demo
   c) Commercial
   d) Shareware

4. Життєвий цикл ПЗ

7. Який етап життєвого циклу йде після Проєкттування?
   a) Аналіз вимог
   b) Реалізація (програмування)
   c) Тестування
   d) Впровадження
8. На якому етапі відбувається усунення помилок і перевірка працездатності програми?
   a) Проєкттування
   b) Тестування
   c) Впровадження
   d) Супровід

Тема. Системне програмне забезпечення. Операційні системи як базове системне ПЗ.

- ✓ Системне програмне забезпечення і його основні компоненти.
- ✓ Призначення та функції ОС.
- ✓ Архітектура ОС (ядро, оболонка, драйвери).
- ✓ Порівняння ОС (Windows, Linux, macOS).
- ✓ Типи ОС: настільні, мережеві, мобільні, вбудовані.

Конспект лекції

Вступ

У сучасному світі інформаційних технологій програмне забезпечення виступає основою функціонування будь-якої комп'ютерної системи. Особливе місце в його структурі займає системне програмне забезпечення, яке забезпечує взаємодію між апаратною частиною комп'ютера та прикладними програмами. Саме воно створює середовище, у якому користувач може ефективно працювати з технічними ресурсами системи.

Ключовим елементом системного програмного забезпечення є операційна система (ОС) – комплекс програм, що керує апаратними ресурсами, організовує зберігання даних, виконання програм та забезпечує інтерфейс взаємодії користувача з комп'ютером. Розуміння принципів роботи операційних систем є основою підготовки фахівців у сфері інформатики, програмування та комп'ютерних технологій.

У межах цієї теми розглядаються основні поняття та складові системного ПЗ, структура та функції операційних систем, їх архітектура (ядро, оболонка, драйвери), а також здійснюється порівняльний аналіз найпоширеніших ОС – Windows, Linux та macOS. Окрему увагу приділено класифікації операційних систем за призначенням: мережеві, мобільні та вбудовані системи, які відіграють ключову роль у сучасному цифровому середовищі.

Мета лекції: пояснити роль і призначення системного програмного забезпечення в обчислювальних системах; розкрити структуру та функції операційних систем, їх архітектуру та взаємодію з апаратними ресурсами; ознайомити студентів з різними типами ОС і порівняти популярні платформи (Windows, Linux, macOS).

Системне програмне забезпечення – це сукупність програм, які керують роботою комп'ютера та створюють операційне середовище для

виконання інших програм. Воно забезпечує базову функціональність комп'ютера, а також виконує допоміжні функції, такі як керування пристроями, діагностика, копіювання та архівація файлів. До системного ПЗ належать операційні системи, драйвери, антивіруси та утиліти.

Основні компоненти системного програмного забезпечення

Базова система введення/виведення (BIOS, БІОС) – це набір мікропрограм, розташованих на материнській платі комп'ютера, який ініціалізує апаратне забезпечення при увімкненні, виконує самотестування (POST) і завантажує операційну систему. Це найперше програмне забезпечення, що запускається після натискання кнопки живлення, забезпечуючи взаємодію між "залізом" та ОС. Сучасною заміною БІОС є UEFI, який пропонує додаткові функції, швидше завантаження та розширені можливості налаштування.

Хоч це не звичайне програмне забезпечення, а вбудоване (firmware, прошивка), його відносять до системного програмного забезпечення, оскільки воно працює на базовому рівні між апаратурою і ОС.

Операційні системи (ОС) – основний вид системного ПЗ, який керує всіма ресурсами комп'ютера, від апаратного забезпечення до запуску програм.

Драйвери – програми, що дозволяють операційній системі взаємодіяти з конкретними апаратними пристроями, наприклад, принтером або відеокартою.

Утиліти – програми, які виконують допоміжні функції, такі як архівування, антивірусний захист, форматування дисків або оптимізація їх роботи.

Мережеві операційні системи – забезпечують роботу комп'ютерних мереж.

Оболонки – додають до операційної системи користувацький інтерфейс або розширюють її функціональність.

Призначення та функції ОС

Операційна система (ОС) – це комплекс програм, що забезпечує взаємодію користувача з апаратною частиною комп’ютера та організовує виконання програм.

Основні функції ОС:

- *керування процесами* – планування та розподіл процесорного часу;
- *керування пам’яттю* – виділення, захист і оптимальне використання оперативної пам’яті;

- *керування файлами* – створення, збереження, доступ і захист даних у файловій системі;
- *керування пристроями* – робота з периферією через драйвери;
- *інтерфейс користувача* – командна оболонка або графічний інтерфейс (GUI);
- *забезпечення безпеки* – контроль доступу, автентифікація, шифрування.

Архітектура ОС

Структура операційної системи складається з кількох ключових частин:

- *ядро (kernel)* – "серце" ОС, що керує апаратними ресурсами та основними процесами;
- *драйвери пристроїв* – спеціальні модулі для роботи з конкретним обладнанням (принтери, відеокарти тощо);
- *системні бібліотеки* – набір готових функцій для програм;
- *оболонка (shell)* – інтерфейс користувача:
  - командний (CLI, приклад – Bash),
  - графічний (GUI, приклад – Windows Explorer).

Драйвери – це спеціальні програми, які забезпечують взаємодію операційної системи з апаратним забезпеченням (пристроями комп'ютера). Вони є проміжною ланкою між «залізом» та ОС, дозволяючи використовувати пристрої коректно.

Основні типи драйверів:

- *драйвери відеокарт* – забезпечують коректне відображення графіки та відео;
- *драйвери принтерів* – дозволяють ОС надсилати друковані завдання до принтера;
- *драйвери мережевих адаптерів* – забезпечують підключення до інтернету;
- *драйвери аудіокарт* – відповідають за правильне відтворення звуку.

Без оновлених драйверів пристрої можуть працювати некоректно або не працювати **взагалі.**

Порівняння ОС:

Характеристика	**Windows**	**Linux**	**macOS**
Інтерфейс	Графічний, простий у користуванні	CLI + GUI, гнучкий	Графічний, інтуїтивний

Відкритість коду	Закритий	Відкритий (Open Source)	Закритий
Безпека	Вразливіша до вірусів	Висока стійкість, регулярні оновлення	Висока, інтегрована із системою
Сфера застосування	Масовий сегмент (офіси, навчання, ігри)	Сервери, розробка, наукові обчислення	Креативні індустрії (дизайн, монтаж відео)

Типи операційних систем

- *Мережеві ОС* – підтримують роботу комп'ютерних мереж, адміністрування користувачів і ресурсів (Windows Server, Linux Server).
- *Мобільні ОС* – оптимізовані для смартфонів і планшетів (Android, iOS).
- *Вбудовані ОС* – використовуються у спеціалізованих пристроях (автомобілі, банкомати, побутова техніка; приклади: FreeRTOS, QNX).
- *Настільні ОС* – загального призначення для ПК (Windows, Linux, macOS).

Висновок

Системне програмне забезпечення – це комплекс програм, який управляє роботою комп'ютера та забезпечує взаємодію між користувачем і апаратними ресурсами. Основними компонентами системного ПЗ є операційні системи, драйвери пристроїв, інтерфейсні оболонки та сервісні утиліти. Операційна система виконує ключові функції: управління апаратними ресурсами, організацію файлової системи, запуск і координацію інших програм, забезпечення інтерфейсу користувача. Архітектура ОС включає ядро (core), що керує ресурсами, оболонку для взаємодії з користувачем та драйвери, які забезпечують роботу пристроїв. Популярні ОС – Windows, Linux, macOS – відрізняються архітектурою, призначенням та застосуванням. Існують різні типи ОС: настільні для персональних комп'ютерів, мережеві для серверів, мобільні для портативних пристроїв та вбудовані для спеціалізованих пристроїв. Системне ПЗ є фундаментом для роботи всього програмного середовища й забезпечує ефективне використання обчислювальних ресурсів.

Питання для самоконтролю

1. Призначення та функції ОС

1. Основне призначення операційної системи:
   а) Вирішення математичних задач

b) Забезпечення взаємодії користувача з апаратним забезпеченням
c) Запуск лише прикладних програм
d) Створення графіки

2. Яка з наведених функцій НЕ належить до ОС?
   a) Керування пам’яттю
   b) Керування процесами
   c) Керування пристроями
   d) Компоновка тексту в документі

2. Архітектура ОС

3. Як називається центральна частина ОС, що керує апаратними ресурсами?
   a) Драйвер
   b) Ядро
   c) Оболонка
   d) Файлова система
4. Що таке драйвер у складі ОС?
   a) Програма для створення текстів
   b) Модуль для керування апаратним пристроєм
   c) Засіб для компіляції програм
   d) Інтерфейс користувача

3. Порівняння ОС

5. Яка ОС є відкритою за вихідним кодом?
   a) Windows
   b) macOS
   c) Linux
   d) Android
6. Яка ОС найчастіше використовується у сфері дизайну та відеомонтажу?
   a) Windows
   b) Linux
   c) macOS
   d) FreeRTOS

4. Типи ОС

7. Яка з ОС належить до мобільних?
   a) Windows Server
   b) Linux Ubuntu
   c) Android
   d) QNX

8. Яка ОС використовується у вбудованих системах (автомобілі, банкомати)?
   a) Windows 10
   b) FreeRTOS
   c) Linux Mint
   d) macOS

Тема. Службове програмне забезпечення

- ✓ Поняття службового програмного забезпечення
- ✓ Програми обслуговування дисків (дефрагментація, очищення, перевірка).
- ✓ Архіватори, антивіруси, фаєрволи.
- ✓ Засоби резервного копіювання та відновлення.

Конспект лекції

Вступ

Службове програмне забезпечення є важливою складовою системного середовища сучасного комп'ютера, адже саме воно забезпечує підтримку стабільної, безпечної та ефективної роботи операційної системи і користувацьких програм. Якщо операційна система виступає основою функціонування комп'ютера, то службові програми виконують роль «інструментів технічного обслуговування», що допомагають підтримувати оптимальний стан системи.

До службового програмного забезпечення належать різноманітні утиліти, призначені для діагностики, оптимізації, захисту та відновлення даних. Зокрема, це програми обслуговування дисків (дефрагментація, очищення, перевірка файлової системи), архіватори, що забезпечують стиснення та пакування файлів, антивірусні засоби і фаєрволи, які захищають комп'ютер від шкідливого програмного впливу та несанкціонованого доступу.

Важливу роль відіграють також засоби резервного копіювання та відновлення даних, які гарантують збереження інформації у випадку технічних збоїв або кібератак. Розуміння призначення, принципів роботи та класифікації службового програмного забезпечення є необхідним для ефективного управління інформаційними ресурсами і забезпечення надійності комп'ютерних систем.

Мета лекції: надати студентам знання про призначення та різновиди службового ПЗ; ознайомити з інструментами обслуговування дисків, архівації, антивірусного захисту, фаєрволів, резервного копіювання та відновлення даних; сформувати розуміння ролі службового ПЗ у підтримці стабільної та безпечної роботи систем.

Службове програмне забезпечення – це сукупність програм, які забезпечують ефективну роботу комп'ютерної системи, її діагностику, обслуговування та захист від збоїв. Основні завдання службового ПЗ:

- оптимізація роботи комп'ютера;
- виявлення та усунення помилок;

- підвищення продуктивності;
- захист від шкідливого програмного забезпечення.

Програми обслуговування дисків

1. Дефрагментація

Це процес упорядкування фрагментованих файлів на жорсткому диску для прискорення доступу до даних.

Основні програми:

Windows Defragmenter (вбудований в Windows) - базовий інструмент для HDD;

Defraggler від Piriform – дозволяє дефрагментувати окремі файли та папки;

Auslogics Disk Defrag – швидка дефрагментація з оптимізацією системних файлів;

O&O Defrag – професійний інструмент з різними методами де фрагментації;

Зауваження: SSD-диски НЕ потребують дефрагментації. Для них це навіть шкідливо, оскільки скорочує термін служби.

2. Очищення диска

Це видалення непотрібних файлів для звільнення місця на диску.

Основні програми:

Disk Cleanup (вбудований в Windows) **–** видаляє тимчасові файли, кеш, старі оновлення

CCleaner – популярний інструмент для очищення системи та реєстру

BleachBit – безкоштовна альтернатива з акцентом на приватність

TreeSize Free – візуалізує використання простору на диску

WinDirStat – показує графічне представлення файлів за розміром

Очищуються тимчасові файли, кеш браузерів, корзина, старі логи, залишки оновлень, дублікати файлів.

3. Перевірка диска

Це діагностика та виправлення помилок файлової системи, пошкоджених секторів.

Основні програми:

CHKDSK (Windows Command Line) - вбудована утиліта для перевірки та виправлення помилок.

CrystalDiskInfo – моніторинг стану здоров'я диска через S.M.A.R.T.

HD Tune – комплексна перевірка продуктивності та помилок.

Victoria – потужний інструмент для низькорівневої діагностики.

HDDScan – безкоштовна утиліта для тестування поверхні диска.

Перевіряються логічні помилки файлової системі, фізичні пошкодження секторів, загальний стан здоров'я диска.

Рекомендації по обслуговуванню

Для HDD:

дефрагментація: раз на 1-3 місяці;

очищення: щомісяця або при необхідності;

перевірка: раз на 3-6 місяців або при підозрах на проблеми.

Для SSD:

дефрагментація: ніколи;

очищення: регулярно, але без перезапису;

перевірка: моніторинг S.M.A.R.T. щомісяця;

TRIM: переконайтеся, що він увімкнений в Windows

Фаєрвол – це програмне забезпечення або апаратний пристрій (міжмережевий екран або брандмауер), який контролює та фільтрує мережевий трафік, захищаючи комп'ютерні мережі та пристрої від несанкціонованого доступу та кіберзагроз. Він працює як "охоронець", що перевіряє кожен пакет даних і пропускає лише той, що відповідає заданим правилам безпеки.

Як працює фаєрвол:

- *фільтрація трафіку***:** аналізує дані, що надходять до мережі або пристрою, і ті, що відправляються;
- *перевірка на основі правил:* використовує встановлені правила для визначення, чи безпечний трафік. Це може ґрунтуватися на джерелі та призначенні трафіку, його вмісті, типі з'єднання тощо;
- *блокування та дозвіл:* якщо трафік відповідає правилам, він пропускається. Якщо ж він підозрілий або шкідливий (наприклад, вірус, спроба злому), фаєрвол блокує його.

Типи фаєрволів

- **Програмні:**

Це програмне забезпечення, яке встановлюється безпосередньо на комп'ютер або сервер (наприклад, Брандмауер Windows). Воно захищає пристрій, на якому встановлено.

- **Апаратні:**

Це фізичні пристрої, які розміщуються між локальною мережею та Інтернетом. Вони забезпечують вищий рівень безпеки для всієї мережі та можуть обробляти великі обсяги трафіку без втрати продуктивності.

- **Програмно-апаратні комплекси:**

Це поєднання програмного та апаратного забезпечення, які є автономними системами безпеки.

Фаєрвол як сервіс (FWaaS): хмарний сервіс, який надає захист для кількох клієнтських інфраструктур, працюючи на стороні провайдера.

Архівування й відновлення даних

Архівування – це стискання файлів для зменшення їхнього розміру. Відновлення – процес повернення даних у разі їхньої втрати. Популярні утиліти:

- **WinRAR, 7-Zip** – створення архівів ZIP, RAR.
- **Acronis True Image** – резервне копіювання й відновлення системи.
- **Windows Backup** – стандартний засіб резервного копіювання у Windows.

Антивірусні програми

Антивіруси – це спеціальні програми, що захищають комп'ютер від вірусів, троянів, шпигунського ПЗ та інших загроз. Вони працюють у фоновому режимі та перевіряють файли на наявність шкідливого коду.

Основні функції антивірусів:

- сканування системи на віруси;
- видалення або ізоляція шкідливих файлів;
- захист у реальному часі;
- перевірка файлів у хмарі.

Популярні антивірусні програми:

- **Windows Defender** – вбудований у Windows, забезпечує базовий захист;
- **Avast, Kaspersky, Bitdefender** – потужні рішення з додатковими функціями;
- **Malwarebytes** – спеціалізується на виявленні шкідливих програм.

Висновок

Службове програмне забезпечення є важливим компонентом операційної системи. Драйвери забезпечують роботу пристроїв, утиліти допомагають оптимізувати систему та зберігати дані, а антивіруси гарантують безпеку комп'ютера. Регулярне використання та оновлення службового ПЗ сприяє стабільній і безпечній роботі комп'ютера.

Питання для самоконтролю

Тестові питання (Multiple Choice)

1. Яке з наведених програм належить до службового програмного забезпечення?
    - a) Microsoft Word
    - b) WinRAR
    - c) Photoshop
    - d) Chrome
2. Яке призначення антивірусного програмного забезпечення?
    - a) Оптимізація роботи процесора
    - b) Виявлення та видалення шкідливих програм
    - c) Архівування файлів
    - d) Керування мережею
3. Яка утиліта використовується для **дефрагментації диска**?
    - a) Disk Cleanup
    - b) Task Manager
    - c) Defragment and Optimize Drives
    - d) Registry Editor
4. Який тип службових програм відповідає за **архівування даних**?
    - a) Файлові менеджери
    - b) Утиліти безпеки
    - c) Компресори
    - d) Емулятори
5. Яке службове ПЗ дозволяє контролювати процеси, що виконуються в системі?
    - a) File Explorer
    - b) Task Manager
    - c) Disk Management
    - d) Control Panel

Тестові питання (True/False)

6. Службове програмне забезпечення – це прикладні програми, призначені для виконання конкретних задач користувача (текстові редактори, браузери).
   Так/Ні
7. Утиліти для резервного копіювання належать до службового програмного забезпечення.
   Так/Ні

8. Програми-емулятори дозволяють відтворювати роботу однієї системи в іншій.
   Так/Ні

Відкрите запитання

9. Наведіть приклади службових програм, які ви використовуєте у своїй операційній системі, та поясніть їхнє призначення.

Тема. Інструментальне програмне забезпечення для розробки

- ✓ Поняття інструментального ПЗ.
- ✓ Системи програмування.
- ✓ IDE, компілятори, інтерпретатори.
- ✓ Приклади: Visual Studio, Code::Blocks, JetBrains, GCC.
- ✓ Системи контролю версій: Git, SVN.
- ✓ Пакетні менеджери (npm, pip, apt).

Конспект лекції

Вступ

Інструментальне програмне забезпечення є ключовою складовою сучасної програмної індустрії, оскільки саме воно забезпечує створення, тестування, налагодження та супровід програмних продуктів. Цей тип ПЗ використовується переважно розробниками та інженерами-програмістами і формує основу процесу розробки програмного забезпечення на всіх його етапах.

До інструментального програмного забезпечення належать інтегровані середовища розробки (IDE), які об’єднують у собі редактор коду, компілятор, засоби відлагодження та інші утиліти; компілятори та інтерпретатори, що перетворюють програмний код у форму, зрозумілу комп’ютеру; а також системи контролю версій, які дають змогу координувати спільну роботу над проєктами, відстежувати зміни та зберігати історію розробки.

Серед поширених прикладів інструментального ПЗ можна назвати Visual Studio, Code::Blocks, GCC, продукти JetBrains (IntelliJ IDEA, PyCharm тощо). Важливу роль у сучасній розробці відіграють Git та платформа GitHub, що забезпечують ефективну командну взаємодію, а також пакетні менеджери (наприклад, npm, pip, apt), які автоматизують установлення й оновлення необхідних бібліотек та залежностей.

Мета лекції: розкрити поняття інструментальних засобів розробки та їхнє значення для програмування; познайомити студентів з IDE, компіляторами та інтерпретаторами; навчити принципам роботи систем контролю версій і підкреслити важливість керування змінами під час розробки ПЗ.

Інструментальне програмне забезпечення – це сукупність програмних засобів, що використовуються для створення, налагодження, тестування та супроводу програм. Воно становить основу сучасної розробки

програмного забезпечення, значно підвищуючи продуктивність розробників та якість кінцевого продукту.

Основні компоненти інструментального ПЗ:

- ✓ **редактори коду** (Notepad++, Visual Studio Code, Sublime Text) – забезпечують зручне написання та редагування вихідного коду з підсвічуванням синтаксису, автодоповненням та іншими функціями;
- ✓ **компілятори та інтерпретатори** (GCC, Clang, Python Interpreter, Node.js) – перетворюють код, написаний мовою програмування, у формат, який може виконуватися комп'ютером;
- ✓ **системи управління версіями** (Git, SVN, Mercurial) – відстежують зміни в коді, забезпечують співпрацю між розробниками та дозволяють повертатися до попередніх версій;
- ✓ **налагоджувачі** (GDB, WinDbg, LLDB) – допомагають знаходити та виправляти помилки в програмах шляхом покрокового виконання коду та аналізу стану програми;
- ✓ **інтегровані середовища розробки (IDE)** (Eclipse, Visual Studio, IntelliJ IDEA, PyCharm) – об'єднують усі вищезазначені інструменти в єдиному зручному інтерфейсі.

Ключові функції інструментального ПЗ:

- автоматизація рутинних операцій у процесі написання програм;
- перетворення коду в машинний формат для виконання на цільовій платформі;
- виявлення синтаксичних, логічних та семантичних помилок у коді;
- оптимізація програм для підвищення швидкодії та ефективності використання ресурсів;
- тестування та забезпечення якості програмного продукту;
- управління залежностями та бібліотеками.

Системи програмування: детальний огляд

Системи програмування – це комплексні програмні засоби, що забезпечують повний цикл розробки програм від написання коду до його виконання. Вони містять редактор коду, компілятор або інтерпретатор, налагоджувач та набір допоміжних утиліт.

1. Класифікація систем програмування

За рівнем програмування:

Низькорівневі системи:

- Мови асемблера (MASM, NASM, FASM) працюють безпосередньо з машинними командами процесора

- Забезпечують максимальний контроль над апаратним забезпеченням
- Використовуються для розробки драйверів, операційних систем, критичних за швидкодією ділянок коду
- Вимагають глибокого розуміння архітектури процесора
- Код залежить від конкретної апаратної платформи

Високорівневі системи:

- Мови програмування (C, C++, Java, Python, C#) використовують абстракції, зрозумілі для людини
- Забезпечують портативність коду між різними платформами
- Пришвидшують розробку завдяки потужним бібліотекам та фреймворкам
- Автоматизують управління пам'яттю (в деяких мовах)
- Мають багатий екосистему інструментів та спільноту розробників

<u>За способом реалізації:</u>

Компілятори:

- Перекладають весь вихідний код у машинний формат перед виконанням
- Створюють окремий виконуваний файл
- Забезпечують високу швидкість виконання програми
- Виявляють помилки на етапі компіляції
- Приклади: GCC (для C/C++), Clang, компілятор Go
- Процес компіляції включає: лексичний аналіз, синтаксичний аналіз, семантичний аналіз, оптимізацію, генерацію коду

Інтерпретатори:

- Виконують програму рядок за рядком без створення окремого виконуваного файлу
- Забезпечують більшу гнучкість та швидшу розробку
- Помилки виявляються під час виконання
- Нижча швидкість виконання порівняно з компільованими програмами
- Приклади: Python Interpreter, Ruby, PHP
- Ідеальні для скриптування, швидкого прототипування, інтерактивної розробки

Гібридні системи:

- Поєднують переваги компіляції та інтерпретації
- Java: компіляція в байт-код, виконання через віртуальну машину JVM
- C#: компіляція в IL (Intermediate Language), виконання через CLR (Common Language Runtime)
- Забезпечують портативність та прийнятну швидкість виконання
- Дозволяють JIT-компіляцію (Just-In-Time) для оптимізації під час виконання

<u>За середовищем використання:</u>

Локальні системи:

- Встановлюються та працюють безпосередньо на комп'ютері розробника
- Забезпечують повний контроль над середовищем розробки
- Не потребують постійного інтернет-з'єднання
- Приклади: Visual Studio, IntelliJ IDEA, Eclipse
- Можуть споживати значні ресурси системи

Хмарні системи:

- Доступні через веб-браузер
- Не потребують встановлення на локальному комп'ютері
- Забезпечують співпрацю в реальному часі
- Приклади: Replit, AWS Cloud9, GitHub Codespaces, CodeSandbox
- Ідеальні для навчання, швидкого прототипування, роботи з різних пристроїв

2. Архітектура та компоненти систем програмування

Редактор коду:

- Підсвічування синтаксису для різних мов програмування
- Автодоповнення коду (IntelliSense, Code Completion)
- Рефакторинг коду (перейменування, витягування методів)
- Навігація по коду (перехід до визначення, пошук використань)
- Інтеграція з системами контролю версій
- Підтримка сніпетів (шаблонів коду)

Компілятор/Інтерпретатор:

- Лексичний аналізатор (розбиває код на токени)
- Синтаксичний аналізатор (перевіряє правильність структури програми)
- Семантичний аналізатор (перевіряє логічну коректність)
- Оптимізатор (покращує ефективність коду)
- Генератор коду (створює машинний код або байт-код)

Налагоджувач (Debugger):

- Встановлення точок зупинки (breakpoints)
- Покрокове виконання коду (step-by-step execution)
- Перегляд значень змінних у реальному часі
- Аналіз стеку викликів функцій
- Умовні точки зупинки
- Моніторинг пам'яті та ресурсів

Засоби тестування:

- Модульне тестування (unit testing) для перевірки окремих компонентів
- Інтеграційне тестування для перевірки взаємодії компонентів
- Функціональне тестування для перевірки відповідності вимогам
- Автоматизовані тести для регресійного тестування

- Фреймворки тестування: JUnit, pytest, Jest, NUnit

Бібліотеки та фреймворки:

- Стандартні бібліотеки мов програмування
- Сторонні бібліотеки для розширення функціональності
- Фреймворки для веб-розробки (Django, Spring, ASP.NET)
- Бібліотеки для роботи з базами даних, мережами, графікою
- Прискорюють розробку завдяки готовим перевіреним рішенням

Інтегровані середовища розробки (IDE)

Інтегроване середовище розробки (IDE) – це програмний комплекс, що об'єднує всі необхідні інструменти для розробки в єдиному зручному інтерфейсі.

Це програмне забезпечення, яке забезпечує зручне середовище для написання, компіляції, тестування та налагодження коду. Основні функції: редактор коду, компілятор, дебагер, система підказок.

Основні переваги IDE:

- Єдине середовище для всіх етапів розробки
- Підвищення продуктивності завдяки автоматизації
- Інтеграція з системами контролю версій
- Вбудовані інструменти профілювання та аналізу коду
- Підтримка різних мов програмування та технологій

Популярні IDE:

Visual Studio:

- Розроблена Microsoft для екосистеми .NET
- Підтримує C#, C++, Python, JavaScript та інші мови
- Потужні інструменти налагодження та профілювання
- Інтеграція з Azure для хмарної розробки
- Розширюваність через плагіни

Visual Studio Code:

- Легкий та швидкий редактор коду з можливостями IDE
- Кросплатформенність (Windows, macOS, Linux)
- Величезна екосистема розширень
- Вбудована підтримка Git
- Ідеальний для веб-розробки та скриптування

IntelliJ IDEA (JetBrains):

- Провідна IDE для Java-розробки
- Інтелектуальне автодоповнення та рефакторинг
- Вбудовані інструменти для роботи з базами даних

- Підтримка фреймворків (Spring, Hibernate)
- Родина продуктів: PyCharm (Python), WebStorm (JavaScript), PhpStorm (PHP)

Eclipse:

- Відкрите кросплатформенне середовище
- Спочатку для Java, тепер підтримує багато мов
- Модульна архітектура на основі плагінів
- Використовується для розробки Android-додатків

Code::Blocks:

- Вільна кросплатформенна IDE для C/C++
- Підтримка різних компіляторів (GCC, Clang, MSVC)
- Легка та швидка
- Ідеальна для навчання програмуванню

Компілятори та інтерпретатори:

Компілятор GCC (GNU Compiler Collection):

- Один з найпопулярніших відкритих компіляторів
- Підтримує C, C++, Objective-C, Fortran, Ada та інші мови
- Кросплатформенність та портативність
- Високий рівень оптимізації коду
- Використовується в операційних системах Linux та багатьох інших проєктах
- Команди компіляції: gcc program.c -o program, g++ program.cpp -o program

Інтерпретатор Python:

- Стандартна реалізація CPython написана на C
- Альтернативні реалізації: PyPy (JIT-компіляція), Jython (на JVM), IronPython (.NET)
- Інтерактивний режим для швидкого тестування коду
- Підтримка динамічної типізації
- Величезна стандартна бібліотека

Віртуальні машини:

- JVM (Java Virtual Machine) для виконання Java-програм
- CLR (Common Language Runtime) для .NET-додатків
- V8 (JavaScript engine у Chrome та Node.js)
- Забезпечують портативність та безпеку виконання

Системи контролю версій

Git:

- Розподілена система контролю версій

- Створена Лінусом Торвальдсом для розробки ядра Linux
- Основні команди: git init, git add, git commit, git push, git pull
- Гілкування (branching) для паралельної розробки
- Злиття (merging) змін з різних гілок
- Локальний репозиторій на кожному комп'ютері розробника

GitHub:

- Веб-платформа для хостингу Git-репозиторіїв
- Інструменти для співпрацю: pull requests, code review, issues
- GitHub Actions для CI/CD (безперервна інтеграція/доставка)
- Соціальні функції: stars, forks, following
- Безкоштовні приватні репозиторії

Альтернативи:

- GitLab: самохостинг, вбудований CI/CD
- Bitbucket: інтеграція з продуктами Atlassian
- Azure DevOps: екосистема Microsoft

Основні концепції:

- Коміт (commit): збереження змін у локальному репозиторії
- Гілка (branch): окрема лінія розробки
- Злиття (merge): об'єднання змін з різних гілок
- Конфлікт (conflict): суперечливі зміни, що потребують ручного вирішення
- Pull request: запит на включення змін у основну гілку

SVN (Subversion) – це система контролю версій, яка дозволяє відстежувати зміни у файлах проєкту з часом, повертатися до попередніх версій та працювати в команді над спільним проєктом. SVN належить до централізованих систем, де всі дані та історія змін зберігаються в одному центральному репозиторії.

Основні функції та можливості:

– *відстеження змін*: записує історію всіх внесених змін, хто і коли їх зробив;

– *повернення до попередніх версій*: дозволяє відновити проєкт до будь-якої попередньої версії, якщо щось пішло не так;

– *спільна робота*: дає можливість кільком розробникам одночасно працювати над одним проєктом, не заважаючи один одному, завдяки централізованому сховищу;

– *створення гілок (branches)*: дозволяє створювати окремі "гілки" для розробки нових функцій або виправлення помилок, не впливаючи на основну версію коду;

– *надійне збереження даних*: забезпечує захист даних від втрати.

Принцип роботи:

центральний репозиторій: це єдине сховище коду та історії змін, яке доступне для всіх розробників;

клієнт: кожен розробник має локальну копію проєкту на своєму комп'ютері, з якою він безпосередньо працює;

синхронізація: після внесення змін, розробник "фіксує" (commit) їх до центрального репозиторію, де вони стають доступними для інших. Інші розробники можуть "витягувати" (checkout) останні зміни з репозиторію, щоб оновити свої локальні копії.

CVS (Concurrent Versions System) – це система контролю версій, яка дозволяла командам розробників спільно працювати над проєктами, відстежуючи зміни у вихідному коді та інших файлах. Вона була популярна в 1990-х – на початку 2000-х років, зберігаючи історію змін і полегшуючи спільну роботу, але вважається застарілою, оскільки її активна розробка припинилася наприкінці 2000-х.

Пакетні менеджери

npm (Node Package Manager):

- менеджер пакетів для JavaScript та Node.js;
- найбільший реєстр програмних бібліотек у світі;
- файл package.json описує залежності проекту;
- команди: npm install, npm update, npm run;
- дозволяє легко ділитися та використовувати код.

pip (Package Installer for Python):

- стандартний менеджер пакетів для Python;
- встановлення з PyPI (Python Package Index);
- команди: pip install package, pip freeze > requirements.txt;
- віртуальні середовища (venv) для ізоляції залежностей;
- pip3 для Python 3.

apt (Advanced Package Tool):

- менеджер пакетів для Debian-based дистрибутивів Linux (Ubuntu, Mint);
- управління системними пакетами та залежностями;
- команди: sudo apt update, sudo apt install, sudo apt upgrade;
- автоматичне розв'язання залежностей;
- центральні репозиторії для безпечного встановлення ПЗ.

Інші пакетні менеджери:

- yarn: альтернатива npm з кращою швидкістю;

- composer: менеджер залежностей для PHP;
- Maven/Gradle: інструменти збірки та управління залежностями для Java;
- NuGet: менеджер пакетів для .NET;
- Homebrew: менеджер пакетів для macOS;

Переваги використання пакетних менеджерів:

- автоматичне управління залежностями;
- легке встановлення та оновлення бібліотек;
- контроль версій залежностей;
- спрощення розгортання проєктів;
- стандартизація процесу розробки.

Висновок

Сучасна розробка програмного забезпечення неможлива без використання інструментального ПЗ. IDE, компілятори, інтерпретатори, системи контролю версій та пакетні менеджери становлять екосистему, що дозволяє розробникам ефективно створювати якісні програмні продукти. Розуміння цих інструментів та вміння з ними працювати є ключовою компетенцією сучасного програміста.

Питання для самоконтролю

1. Вибір однієї правильної відповіді

1.1. Яка з програм є прикладом інтегрованого середовища розробки (IDE)?

A) GCC
B) Git
C) Visual Studio
D) pip

1.2. Яка з наведених програм є компілятором мови C/C++?

A) JetBrains Rider
B) GCC
C) Node.js
D) GitHub

1.3. Що таке Git?

A) Онлайн-платформа для зберігання коду
B) Середовище розробки
C) Система контролю версій
D) Компілятор Java

2. Вибір кількох правильних відповідей

2.1. Які з наступних інструментів є IDE?

a) Visual Studio
b) Code::Blocks
c) JetBrains PyCharm
d) pip
e) GitHub

2.2. Які функції виконує система контролю версій Git?

a) Відстеження змін у коді
b) Робота з гілками
c) Збереження історії проєкту
d) Компіляція програм

2.3. Які з наведених прикладів є пакетними менеджерами?

a) npm
b) pip
c) apt
d) VS Code

3**. Так / Ні**

3.1. Інтерпретатор виконує програму построково, без попередньої компіляції.
3.2. JetBrains – це назва мови програмування.
3.3. GitHub є онлайн-сервісом для розміщення репозиторіїв Git.
3.4. pip використовується для встановлення пакетів на Java.

4. Встановлення відповідності

4.1. Встановіть відповідність між інструментом і його призначенням:

Інструмент	**Призначення**
A) Git	2) Система контролю версій
B) pip	4) Менеджер пакетів для Python
C) GCC	1) Компілятор для C/C++
D) Code::Blocks	3) IDE для C/C++

5. Відкрите запитання

5.1. Що таке IDE і які його основні функції?
5.2. Назвіть два приклади пакетних менеджерів та вкажіть, для яких мов/систем вони призначені.

Тема. Прикладне програмне забезпечення: функції, види, сфери застосування

- ✓ Поняття прикладного ПЗ та його роль у діяльності користувачів.
- ✓ Основні функції та класифікація прикладних програм.
- ✓ Приклади сфер застосування.
- ✓ Тенденції розвитку прикладного програмного забезпечення.

Конспект лекції

Вступ

У структурі програмного забезпечення комп'ютерних систем прикладне програмне забезпечення (ППЗ) посідає особливе місце, оскільки саме воно безпосередньо забезпечує виконання завдань користувачів у різних сферах діяльності. Якщо системне та службове ПЗ створюють умови для функціонування комп'ютера, то прикладні програми є інструментами, за допомогою яких користувач реалізує свої професійні, навчальні чи творчі потреби.

Прикладне програмне забезпечення охоплює широкий спектр програм – від офісних пакетів для роботи з документами та таблицями до графічних редакторів, мультимедійних систем і спеціалізованих програмних комплексів, що застосовуються у медицині, освіті, бізнесі, інженерії чи наукових дослідженнях.

Основними функціями прикладного ПЗ є автоматизація процесів, оптимізація праці користувача, обробка та аналіз даних, а також створення інформаційних продуктів різного типу. У сучасних умовах особливого значення набувають тенденції розвитку прикладного програмного забезпечення – зокрема, поширення хмарних сервісів, мобільних додатків, штучного інтелекту та веборієнтованих платформ, що змінюють способи взаємодії користувачів із цифровими системами.

Мета лекції: сформувати розуміння ролі прикладного програмного забезпечення у діяльності користувачів; розглянути класифікацію та приклади застосувань прикладних програм у різних сферах; ознайомити з сучасними тенденціями розвитку прикладного ПЗ і їх впливом на професійну діяльність.

Прикладне програмне забезпечення (ПЗ) – це сукупність програм, призначених для виконання конкретних завдань користувача, які безпосередньо не пов'язані з функціонуванням комп'ютера як системи.

Основні функції прикладного ПЗ

1. **Автоматизація діяльності користувача**
   – Обробка текстів, таблиць, зображень, відео, тощо
   – Ведення документації, облік, аналітика
2. **Підтримка прийняття рішень**
   – Аналітичні системи, системи підтримки рішень
3. **Організація дозвілля та комунікацій**
   – Ігри, мультимедійні плеєри, месенджери, соціальні мережі
4. **Навчання та розвиток**
   – Освітні платформи, тренажери, системи тестування

Класифікація прикладного ПЗ

1. За способом використання:

- **Інтерактивне (інтерфейс користувача)** – наприклад, текстові редактори, графічні програми
- **Пакетне (без участі користувача)** – наприклад, автоматизовані звіти, резервне копіювання

2. За сферою застосування:

- **Універсальне ПЗ**
  – Підходить для широкого кола користувачів
  – **Приклади:** Microsoft Office, Adobe Photoshop, VLC Player
- **Спеціалізоване ПЗ**
  – Створене для виконання професійних чи галузевих завдань
  – **Приклади:** AutoCAD (інженерія), 1С:Підприємство (бухгалтерія), SPSS (статистика)
- **Наукове та інженерне ПЗ**
  – Моделювання, розрахунки, обробка експериментальних даних
  – **Приклади:** MATLAB, MathCAD, ANSYS
- **Навчальне ПЗ**
  – Електронні підручники, тренажери, тести
  – **Приклади:** Duolingo, Moodle, Khan Academy
- **Ігрове та розважальне ПЗ**
  – Комп’ютерні ігри, програми для створення музики, відео

Сфери застосування прикладного ПЗ

Сфера	**Приклади програм**
Освіта	Google Classroom, Moodle, Zoom
Бізнес і бухгалтерія	1С, SAP, CRM-системи

Медицина	MEDOC, Doctor Eleks
Інженерія та дизайн	AutoCAD, SolidWorks
Медіа	Adobe Premiere, Audacity
Фінанси	QuickBooks, Financier
Повсякденне використання	Браузери, месенджери, офісні пакети

Тенденції розвитку прикладного ПЗ

- **Хмарні технології** – доступ до програм через інтернет (Google Docs, Microsoft 365)
- **Мобільні додатки** – більшість ПЗ має мобільні версії
- **Інтеграція з ШІ** – автоматизація, персоналізація функцій
- **Підвищення кібербезпеки** – захист персональних даних користувача

Висновки

Прикладне програмне забезпечення є невід'ємною частиною сучасного життя. Його розвиток тісно пов'язаний з потребами суспільства, новітніми технологіями та цифровою трансформацією всіх сфер діяльності. Розуміння класифікації та функцій прикладного ПЗ дозволяє ефективно обирати інструменти для виконання професійних та повсякденних завдань.

Питання для самоконтролю

1. Що таке прикладне програмне забезпечення?

A. Програми, що забезпечують взаємодію апаратних компонентів
B. Програми для роботи користувача з конкретними завданнями
C. Операційна система
D. Системи управління базами даних

2. До якого виду прикладного ПЗ належить Microsoft Excel?

A. Спеціалізоване
B. Системне
C. Універсальне
D. Інструментальне

3. Яка з наведених програм є прикладом спеціалізованого ПЗ?

A. Google Chrome
B. Adobe Photoshop

C. AutoCAD
D. WinRAR

4. Яке ПЗ виконується без участі користувача за заздалегідь заданим сценарієм?
A. Системне
B. Пакетне прикладне
C. Інтерактивне прикладне
D. Клієнт-серверне

5. Яка функція не є основною для прикладного програмного забезпечення?
A. Редагування тексту
B. Відтворення відео
C. Обробка сигналів пристроїв введення
D. Створення презентацій

6. Яка з програм призначена для автоматизації бухгалтерського обліку?
A. VLC Media Player
B. 1С:Підприємство
C. AutoCAD
D. Google Meet

7. До якого виду ПЗ належить система Moodle?
A. Ігрове
B. Системне
C. Навчальне
D. Графічне

8. Яка тенденція розвитку прикладного ПЗ є найактуальнішою?
A. Зменшення функціональності
B. Перехід на текстові інтерфейси
C. Інтеграція з хмарними сервісами
D. Відмова від автоматизації

Тема. Інструменти для роботи з текстом та документами

- ✓ Розмітка документів: Markdown, LaTeX, XML.
- ✓ Редактори: Typora, Overleaf, VS Code, Notepad++.
- ✓ Формати документів, автоматизація форматування, шаблони.

Конспект лекції

Вступ

У сучасних інформаційних технологіях розмітка текстів є одним із ключових засобів структурування та автоматизації роботи з документами. Використання мов розмітки дозволяє відокремити зміст документа від його представлення, що полегшує редагування, повторне використання та форматування текстових матеріалів.

Серед найпоширеніших систем розмітки – Markdown, який забезпечує просте та зручне форматування текстів, LaTeX, орієнтований на наукові й технічні документи з чіткою структурою, та XML (eXtensible Markup Language), який застосовується для опису структури даних, обміну інформацією між системами та збереження складних документів у машиночитному форматі.

Інструменти, такі як MarkText, Overleaf або редактори з підтримкою XML, сприяють автоматизації створення, редагування й форматування документів. Завдяки використанню шаблонів, тегів і правил структурування, мови розмітки значно підвищують ефективність роботи з великими обсягами текстової та технічної інформації.

Мета лекції: надати студентам уявлення про розмітку текстів та її значення для автоматизації оформлення документів; ознайомити з основами Markdown, LaTeX і XML; продемонструвати інструменти для роботи з розміткою та принципи використання шаблонів і структурованих документів.

Розмітка документів

Розмітка тексту – це використання спеціальних команд, або тегів, для визначення структури та форматування тексту, що дозволяє комп'ютерним програмам (наприклад, браузерам) зрозуміти, як його відображати. Це штучна мова, яка використовує анотації для надання інструкцій стосовно структури тексту, наприклад, заголовків, абзаців, списків чи посилань, а також для вставки зображень та інших об'єктів.

Приклади мов розмітки

HTML (HyperText Markup Language): Стандартизована мова розмітки для створення веб-сторінок. Браузери інтерпретують її код, щоб відобразити контент на екрані.

XML (Extensible Markup Language): Використовується для опису структурованих даних. Це дозволяє машині розуміти структуру інформації, наприклад, при передачі даних між системами.

XHTML (Extensible HyperText Markup Language): Розширена версія HTML, яка поєднує в собі переваги HTML та XML, забезпечуючи суворішу структуру.

TeX: Мова для набору складних математичних і наукових текстів, яка широко використовується в академічному середовищі.

SGML (Standard Generalized Markup Language): Основа для інших мов розмітки, таких як HTML і XML. Це потужна, але складна мова для опису структури документів.

PDF (Portable Document Format): Хоча це не класична мова розмітки, PDF використовується для фіксації форматування документа, зберігаючи його вигляд незалежно від пристрою чи програми.

Markdown: Полегшена мова розмітки, яка дозволяє легко писати та читати текст, а також конвертувати його в HTML для подальшого відображення.

Теги: Це команди, які починаються знаком < і закінчуються знаком > (наприклад, <p> для абзацу).

Анотації: Текст, що розміщується між тегами, може бути позначений, наприклад, як заголовок, або оточений тегами, які надають йому певних властивостей.

LaTeX – це система підготовки та верстки документів, яка використовується для створення високоякісних наукових, технічних та інших текстів. Вона відокремлює зміст документа від його оформлення, дозволяючи авторам зосередитися на змісті, а програмі – на верстці. LaTeX широко використовується в академічних колах, особливо для математичних та технічних текстів, і є стандартом де-факто для наукових публікацій.

Мова розмітки: це система команд, яка дозволяє структурувати документ, додавати формули, таблиці та інші елементи.

Система верстки: автоматично форматує текст, забезпечуючи професійний вигляд документів відповідно до заданих правил.

Стандарт для науки: вважається стандартом для підготовки наукових статей, дисертацій та технічних звітів.

Текстові файли: вхідні файли LaTeX є звичайними текстовими файлами, що робить їх легкими для редагування та обміну.

Був розроблений Леслі Лампортом на основі системи TeX.

1. Розмітка

1.1 Markdown

Markdown – легка мова розмітки, призначена для швидкого створення форматованого тексту з мінімальними зусиллями.

- Основне застосування:
  - ✓ Документація
  - ✓ README-файли на GitHub
  - ✓ Статті в блогах
- Основні елементи:
  - ✓ Заголовки (#, ##, ### тощо)
  - ✓ Списки (нумеровані: 1., марковані: -, *)
  - ✓ Виділення тексту (*курсив*, **жирний,** підкреслений**,** ~~закреслений~~)
  - ✓ Посилання та зображення
  - ✓ Вставка коду (інлайново або блоками)

Переваги:

- Простота синтаксису
- Широка підтримка
- Легкість конвертації в HTML, PDF

1.2 LaTeX

LaTeX – потужна система розмітки для наукових та технічних документів.

- Використовується у:
  - ✓ Академічному письмі
  - ✓ Математичних публікаціях
  - ✓ Курсових/дипломних роботах
- Основні можливості:
  - ✓ Форматування тексту, таблиць, списків
  - ✓ Розмітка математичних формул
  - ✓ Створення бібліографій
  - ✓ Автоматична генерація змісту, списку рисунків тощо

Переваги:

- висока якість типографіки;
- підтримка складних математичних формул;
- масштабованість та автоматизація оформлення.

1.3 XML

XML (eXtensible Markup Language) – розширювана мова розмітки, призначена для **структурування, зберігання та обміну даними** між різними

системами та програмами. На відміну від Markdown і LaTeX, XML не орієнтований на візуальне форматування тексту, а на **логічну організацію інформації**.

Основне застосування:

- зберігання структурованих даних;
- обмін інформацією між програмами та вебсервісами;
- опис форматів документів (наприклад, DOCX, SVG, RSS);
- конфігураційні файли програм.

Основні елементи:

- **Теги (елементи)** – визначають структуру даних, наприклад `<name>Іван</name>`
- **Атрибути** – уточнюють властивості елементів, наприклад `<book genre="novel">`
- **Декларація документа** – вказує версію XML і кодування: `<?xml version="1.0" encoding="UTF-8"?>`
- **Ієрархічна структура** – дані організовані у вигляді дерева (батьківські та дочірні елементи)

Переваги:

- незалежність від платформи та мови програмування;
- гнучкість структури (можна створювати власні теги);
- зручність для зберігання й передачі даних;
- підтримка багатьма технологіями (HTML5, JSON, SOAP, SVG тощо).

Порівняльна таблиця мов розмітки

Мова розмітки	Призначення	Особливості та переваги	Основні сфери використання	Приклади інструментів
Markdown	Спрощене форматування тексту для швидкого створення документів	- Простий синтаксис, легко читати навіть без обробки; - Швидке створення форматованих текстів (заголовки, списки, посилання, таблиці); - Підтримується багатьма онлайн-платформами (GitHub, Notion, Obsidian).	- Документація проєктів; - Блоги, вебсторінки; - Освітні та технічні тексти; - README-файли у розробці.	MarkText, Typora, Obsidian, VS Code
LaTeX	Професійне оформлення	- Потужна система форматування;	- Наукові статті та дисертації;	Overleaf, TeXstudio,

Мова розмітки	Призначення	Особливості та переваги	Основні сфери використання	Приклади інструментів
	наукових і технічних документів	- Висока якість типографіки; - Підтримка формул, таблиць, бібліографій; - Використання шаблонів і пакетів.	- Звітність, презентації; - Академічні публікації та книги.	TeXworks
XML	Структурування та обмін даними між програмами і системами	- Гнучка мова опису даних; - Теги задаються користувачем; - Може описувати складну ієрархічну структуру; - Основою для багатьох сучасних форматів (SVG, RSS, DOCX).	- Вебтехнології та обмін даними; - Зберігання конфігурацій; - Документи та бази даних; - Інтеграційні системи.	XML Notepad, Oxygen XML Editor, Visual Studio Code

2. Редактори

2.1 Typora

- Markdown-редактор з **візуальним попереднім переглядом** (WYSIWYG).
- Підтримує:
  - ✓ Синтаксис Markdown
  - ✓ Вставку формул через LaTeX
  - ✓ Таблиці, діаграми, код
- Можливість експорту в PDF, HTML, Word
- Простий у використанні інтерфейс, зручний для початківців

2.2 Overleaf

- Онлайн-редактор для роботи з LaTeX-документами.
- Особливості:
  - ✓ Спільна робота над документами (collaboration)
  - ✓ Автоматичне збереження та компіляція
  - ✓ Шаблони для статей, звітів, резюме
  - ✓ Можливість імпорту з GitHub або Dropbox
- Не потребує встановлення LaTeX на ПК

2.3.XML Notepad

Це безкоштовний редактор з відкритим кодом від Microsoft для перегляду та редагування XML-документів. Він відображає структуру XML як дерево, що

дозволяє легко створювати та змінювати дані, а також підтримує такі функції, як IntelliSense, пошук за регулярними виразами, XPath та перевірка схеми XML.

- **Графічний інтерфейс:** показує структуру XML-документа у вигляді дерева, що полегшує навігацію та редагування.
- **Швидке редагування:** дозволяє додавати елементи, атрибути та текст у два окремих вікна: одне для структури (дерева) та одне для значень.
- **Розширені можливості:** включає IntelliSense для автодоповнення коду, підтримку XPath, XInclude та пошук за регулярними виразами.
- **Висока продуктивність:** добре справляється з великими XML-документами.
- **Інтерактивна перевірка:** проводить перевірку XML-схем в режимі реального часу.
- **Перегляд XSLT:** має вбудований засіб для перегляду HTML-результатів перетворення XSLT.
- **Безкоштовний та відкритий код:** програма є безкоштовною і має відкритий вихідний код.

Visual Studio Code (VS Code) є потужним і універсальним редактором коду, який чудово підходить для розмітки XML та інших завдяки своїй багатій екосистемі розширень. VS Code підтримує вбудовані функції для багатьох мов, а розширення дозволяють додавати специфічні можливості для роботи з XML, такі як перевірка синтаксису, форматування коду та автодоповнення.

Робота у VS Code з XML

Універсальність: завдяки модульній архітектурі, VS Code підтримує різні мови програмування та розмітки, включаючи XML.

Розширення: існує велика кількість розширень, які можна знайти в бібліотеці VS Code. Шукайте розширення, пов'язані з XML, щоб отримати такі можливості:

Підсвічування синтаксису: кольорове виділення тегів, атрибутів та значень для кращої читабельності.

Перевірка синтаксису: автоматичне виявлення помилок у вашому XML-коді.

Форматування коду: автоматичне форматування для забезпечення узгодженого стилю.

Автодоповнення: пропозиції для завершення тегів та атрибутів під час набору коду.

Перегляд структури: зручний навігатор по структурі документа.

***Таблиця**. Порівняння засобів редагування мов розмітки*

Інструмент	Для якої мови	Ліцензія / Вартість	Коментар
MarkText	Markdown	Безкоштовний, з відкритим кодом (open source)	Дуже зручний і простий інтерфейс; підтримує live preview (попередній перегляд форматування).
Typora	Markdown	Умовно платний (є пробна версія)	Красивий, інтуїтивний інтерфейс; популярний серед студентів і блогерів.
Overleaf	LaTeX	Безкоштовний (базова версія онлайн) / Платна (розширений функціонал)	Працює в браузері, не потребує встановлення; зручно для спільної роботи над документами.
TeXstudio / TeXworks	LaTeX	Безкоштовні	Потужні офлайн-середовища для написання документів у LaTeX; зручні для навчання.
XML Notepad	XML	Безкоштовний (від Microsoft)	Простий редактор для створення та перегляду XML-файлів; підходить для базового рівня.
Visual Studio Code (VS Code)	Markdown, LaTeX, XML	Безкоштовний, open source	Універсальний редактор із великою кількістю розширень для всіх трьох мов.
Oxygen XML Editor	XML	Платний (є безкоштовна академічна/тестова версія)	Професійний інструмент для роботи зі складними XML-документами, XSLT, DTD, схемами.

3. Формати документів, автоматизація форматування, шаблони

3.1 Формати документів

- **Текстові формати:** .txt, .md, .tex, .docx
- **Документи для друку:** .pdf
- **Презентації:** .pptx, .pdf (у разі використання Beamer у LaTeX)

3.2 Автоматизація форматування

- Автоматичне форматування полегшує підготовку великих документів.
- У LaTeX можна автоматично:
  - ✓ Нумерувати сторінки, розділи, формули
  - ✓ Створювати список літератури (через BibTeX)
  - ✓ Формувати зміст

- Markdown + Pandoc дозволяє:
- Конвертувати .md → .pdf, .docx, .html
- Використовувати шаблони для уніфікації вигляду

3.3 Шаблони

- Використання шаблонів забезпечує єдине оформлення документації:
  - ✓ шаблони звітів, курсових, резюме, статей;
  - ✓ в Overleaf доступна велика бібліотека шаблонів;
  - ✓ Typora також підтримує кастомні CSS-шаблони.

Висновки

- Markdown – простий, зручний для щоденної документації.
- LaTeX – стандарт для наукових документів з математикою та складною структурою.
- Typora – легкий Markdown-редактор для локальної роботи.
- Overleaf – потужне онлайн-середовище для роботи з LaTeX.

Автоматизація форматування і шаблони економлять час і забезпечують стандартизований вигляд документів.

Питання для самоконтролю

1. Яка мова розмітки використовується переважно для написання наукових текстів з математичними формулами?

A. HTML
B. Markdown
C. LaTeX
D. XML

2. Який з перелічених редакторів є онлайн-сервісом для роботи з LaTeX?

A. Typora
B. Sublime Text
C. Overleaf
D. Notepad++

3. Що з наведеного є прикладом синтаксису для заголовка в Markdown?

A. <h1>Заголовок</h1>
B. ## Заголовок
C. \section{Заголовок}
D. **Заголовок**

4. Який з форматів найчастіше використовується для збереження результату компіляції LaTeX-документа?

A. .docx
B. .html
C. .pdf
D. .txt

5. Який інструмент дозволяє писати Markdown з миттєвим візуальним відображенням результату?

A. Overleaf
B. Typora
C. TeXmaker
D. MS Word

6. Установіть відповідність між інструментом і його призначенням:

Інструмент	**Призначення**
A. Typora	1. Онлайн-редактор LaTeX
B. Overleaf	2. Markdown-редактор з попереднім переглядом
C. LaTeX	3. Система розмітки наукових текстів

7. Markdown дозволяє створювати складні математичні формули на рівні LaTeX.

Так / Ні

8. Overleaf дозволяє одночасну роботу декількох користувачів над одним документом.

Так / Ні

9. Шаблони документів дозволяють автоматично форматувати текст за встановленим стилем.

Так / Ні

Тема. Текстові процесори

- ✓ Призначення та основні функції текстових процесорів.
- ✓ Можливості створення, редагування й форматування текстів.
- ✓ Приклади: Microsoft Word, LibreOffice Writer, Google Docs.
- ✓ Спільна робота над документами, хмарні можливості.

Конспект лекції

Вступ

Текстовий процесор – це прикладне програмне забезпечення, призначене для створення, редагування, форматування та збереження текстових документів.

Роль у сучасних системах: основний інструмент роботи з текстовою інформацією для бізнесу, освіти, науки та побуту.

Відмінність від текстових редакторів: текстові процесори надають розширені можливості (оформлення, автоматизація, інтеграція з іншими системами), тоді як редактори (наприклад, Notepad) забезпечують лише базове введення та редагування. Основна мета: підвищення ефективності підготовки документів і забезпечення їх професійного вигляду.

Мета лекції: пояснити призначення текстових процесорів і їхні основні можливості; навчити студентів працювати зі стилями, таблицями, списками та графічними елементами; ознайомити з можливостями хмарних сервісів і спільної роботи над документами.

Призначення текстових процесорів

- Створення різноманітних текстових документів: звітів, листів, наукових робіт, інструкцій, офіційних бланків тощо.
- Забезпечення зручного **редагування** тексту, **оформлення** за стандартами та **автоматизації** рутинних операцій.
- **Інтеграція** з іншими офісними додатками (таблиці, презентації, бази даних).

Основні функції текстових процесорів

1. **Створення та редагування тексту**
   - введення, копіювання, вирізання, вставка;
   - перевірка орфографії, граматики, стилю.
2. **Форматування**
   - шрифти, абзаци, заголовки, стилі;

- створення списків, колонок, таблиць.

3. **Автоматизація**
   - шаблони документів;
   - автоматична нумерація сторінок, зміст, колонтитули;
   - злиття пошти (mail merge).
4. **Робота з графікою та мультимедіа**
   - вставка зображень, діаграм, об’єктів.
5. **Інтеграція**
   - імпорт/експорт у різні формати (DOCX, ODT, PDF, HTML тощо);
   - взаємодія з іншими програмами (Excel, PowerPoint, бази даних).
6. **Спільна робота**
   - коментування, рецензування, контроль версій;
   - хмарні сервіси (Google Docs, Office 365).

Приклади текстових процесорів

- **Комерційні**: Microsoft Word, Apple Pages, Corel WordPerfect.
- **Вільні та відкриті**: LibreOffice Writer, OpenOffice Writer, AbiWord.
- **Онлайнові**: Google Docs, Zoho Writer.

***Таблиця**. Характеристика сучасних текстових процесорів*

Назва	Тип	Коротка характеристика
Microsoft Word	Комерційний	Потужний процесор із великим набором інструментів, інтеграція з Office 365.
LibreOffice Writer	Вільний, відкритий	Альтернатива Word із підтримкою відкритого формату ODT.
Google Docs	Хмарний	Підтримує спільну роботу онлайн, автоматичне збереження, доступ з будь-якого пристрою.

Архітектура текстового процесора

- **Інтерфейс користувача** (меню, панелі інструментів, вікна форматування).
- **Ядро програми** (обробка тексту, збереження у форматах, шрифтовий рендеринг).
- **Модулі розширення** (словники, плагіни, макроси).
- **Система збереження даних** (власні формати + стандарти обміну).

Спільна робота та хмарні можливості

- **Хмарні сервіси** (Google Docs, Microsoft 365, OnlyOffice):
  - спільне редагування документів кількома користувачами одночасно;
  - коментування, рецензування, контроль версій;
  - автоматичне збереження в реальному часі;
  - доступ до документів із будь-якого пристрою.
- **Переваги хмарних технологій**:
  - командна робота без потреби пересилати файли електронною поштою;
  - безпека й резервне копіювання даних;
  - сумісність форматів і кросплатформність.

Переваги та недоліки

Переваги:

- висока продуктивність при створенні документів;
- зручні інструменти форматування;
- автоматизація рутинних процесів;
- підтримка мультимедіа й різних форматів.

Недоліки:

- вимогливість до ресурсів (особливо сучасні комерційні рішення);
- складність освоєння для початківців;
- залежність від закритих форматів (наприклад, DOCX у Microsoft Word).

Перспективи розвитку текстових процесорів

- Розвиток **хмарних технологій** і спільної роботи онлайн.
- Використання **штучного інтелекту** для допомоги в написанні та редагуванні текстів.
- Уніфікація форматів і підтримка **міжплатформності.**
- Тісна інтеграція з іншими видами офісного ПЗ (таблиці, презентації, БД).

Висновки

Текстові процесори – невід'ємний інструмент сучасного користувача комп'ютера. Вони поєднують функції створення, редагування та професійного оформлення документів із можливостями автоматизації та спільної роботи. Сучасний розвиток відбувається в напрямку інтелектуалізації, онлайн-доступу та інтеграції з іншими офісними додатками.

Питання для самоконтролю

Варіант А (закриті з вибором однієї правильної відповіді)

1. Що відрізняє текстовий процесор від простого текстового редактора?
   a) Можливість змінювати шрифти і форматування
   b) Робота тільки з plain-text файлами
   c) Неможливість вставки зображень
   d) Відсутність шаблонів документів
2. Який із перелічених не є текстовим процесором?
   a) Microsoft Word
   b) LibreOffice Writer
   c) Google Docs
   d) Adobe Photoshop
3. Який формат є відкритим стандартом для текстових документів?
   a) DOCX
   b) ODT
   c) PDF
   d) RTF
4. Яка функція текстового процесора відповідає за автоматичне створення змісту?
   a) Стилі абзаців
   b) Колонтитули
   c) Злиття пошти
   d) Вставка зображень
5. Який модуль у текстовому процесорі відповідає за корекцію орфографії?
   a) Інтерфейс користувача
   b) Ядро програми
   c) Модулі розширення (словники)
   d) Система збереження даних

Варіант В (відкриті короткі відповіді)

1. Поясніть різницю між форматами DOCX і ODT.
2. Назвіть три приклади безкоштовних текстових процесорів.
3. Які можливості автоматизації роботи надають сучасні текстові процесори?
4. Чому хмарні сервіси (Google Docs, Office 365) набувають популярності?
5. Які переваги інтеграції текстових процесорів з іншими офісними додатками?

Тема. Табличні процесори та обробка даних

- ✓ Призначення та функції табличних процесорів.
- ✓ Поглиблена робота з Excel / Google Sheets: формули, діаграми, умовне форматування.
- ✓ Обробка даних CSV.

Конспект лекції

Вступ

У сучасному світі вміння працювати з табличними процесорами є однією з ключових компетенцій фахівця будь-якої галузі. Від бухгалтерів та економістів до інженерів та науковців, від менеджерів до аналітиків даних – усі ці професіонали щоденно використовують електронні таблиці для вирішення різноманітних завдань.

За останні роки роль табличних процесорів значно еволюціонувала. Якщо раніше вони використовувались переважно для простих розрахунків та ведення обліку, то сьогодні це потужні інструменти для аналізу великих масивів даних, візуалізації інформації, автоматизації бізнес-процесів та прийняття управлінських рішень на основі даних.

Навички роботи з табличними процесорами мають широке практичне застосування:

- у бізнесі: створення фінансових звітів, планування бюджетів, аналіз продажів, управління проєктами, ведення баз даних клієнтів;
- в освіті та науці: обробка результатів досліджень, статистичний аналіз, побудова графіків та діаграм, організація навчальних матеріалів;
- в особистому житті: планування сімейного бюджету, ведення обліку витрат, організація особистих проєктів, аналіз інвестицій;
- у державному секторі: обробка статистичних даних, складання звітності, планування ресурсів, моніторинг показників.

Мета лекції: навчити студентів ефективно використовувати табличні процесори для розрахунків, аналізу та візуалізації даних; розкрити принципи роботи з формулами, діаграмами та умовним форматуванням; ознайомити з можливостями обробки CSV та елементами базового аналізу даних у таблицях.

Поняття табличного процесора

Табличний процесор – це програмне забезпечення, яке призначене для створення, редагування, аналізу й візуалізації даних, організованих у вигляді таблиць.

Найпоширеніші представники:

- **Microsoft Excel**
- **Google Sheets**
- LibreOffice Calc
- Apple Numbers

Основні завдання:

- зберігання великих обсягів числової й текстової інформації;
- виконання обчислень за допомогою **формул і функцій**;
- **аналіз** та **візуалізація** даних через діаграми й графіки;
- **експорт та імпорт** даних у різних форматах (зокрема CSV);
- **автоматизація** обчислень (макроси, сценарії).

Призначення та функції табличних процесорів

Призначення:

- облік (фінансовий, статистичний, навчальний тощо);
- планування (графіки, бюджети, розклади);
- аналітика (порівняння показників, тренди, прогнозування).

Основні функції:

1. **Обчислювальні:**
   виконання арифметичних операцій, використання вбудованих функцій (SUM, AVERAGE, IF тощо).
2. **Логічні:**
   аналіз умовних виразів (IF, AND, OR).
3. **Статистичні та фінансові:**
   MEDIAN, STDEV, NPV, PMT тощо.
4. **Пошук і сортування:**
   фільтри, сортування, пошук значень.
5. **Візуалізація:**
   створення діаграм і графіків для наочного представлення даних.
6. **Умовне форматування:**
   зміна вигляду клітинки залежно від значення (наприклад, підсвічування від'ємних чисел червоним).

Поглиблена робота з Microsoft Excel

- **Формули:** починаються зі знаку =. Можуть містити посилання на клітинки (A1, B2).
  Приклади:
  - =A1+B1
  - =AVERAGE(C1:C10)

    - =IF(D2>100;"Високий рівень";"Низький рівень")
- **Абсолютні й відносні посилання:**
    - Відносне: A1 (змінюється при копіюванні формули).
    - Абсолютне: A1 (не змінюється).
- **Діаграми:**
  Типи – стовпчикові, кругові, лінійні, комбіновані.
  Використовуються для швидкого порівняння й аналізу.
- **Умовне форматування:**
  Використовується для автоматичного підсвічування даних (наприклад, усі оцінки нижче 60%).

Google Sheets: спільна робота та автоматизація

- Доступ через браузер, автоматичне збереження.
- Можливість **спільного редагування** у реальному часі.
- Формули та функції подібні до Excel.
- Інтеграція з **Google Forms**, **Google Data Studio**, **Google Drive**.
- Використання **Apps Script** для автоматизації обробки даних.

LibreOffice Calc: офлайн-табличний процесор з відкритим кодом

• Безкоштовний додаток із пакету **LibreOffice**, доступний для Windows, Linux та macOS.
• Підтримує **офлайн-режим роботи** без підключення до Інтернету.
• Має **аналогічний до Excel інтерфейс** і сумісний із форматами `.xls` та `.xlsx`.
• Підтримує **формули, функції, зведені таблиці, діаграми та фільтри**.
• Можливість **створення макросів** мовою **LibreOffice Basic** або **Python** для автоматизації завдань.
• Підтримує **експорт у формати PDF, CSV, HTML**, а також збереження у відкритому форматі `.ods`.
• Має **активну спільноту користувачів** і постійні оновлення, що робить його стабільною альтернативою комерційним продуктам.

Робота з даними у форматі CSV

CSV (Comma-Separated Values) – текстовий формат, де дані розділені комами або крапками з комою.

- Використовується для перенесення даних між програмами (наприклад, з Excel у базу даних).
- Під час імпорту важливо вказати правильний **роздільник** (кома, крапка з комою, табуляція).

- CSV не зберігає формул, кольорів і форматування – лише **текст та числа.**

Основи аналізу даних у таблицях

1. **Сортування та фільтрування:** для виділення потрібних записів.
2. **Зведені таблиці (Pivot Tables):**
   дозволяють агрегувати дані, підраховувати підсумки за категоріями.
3. **Діаграми:** відображають тенденції та зв'язки.
4. **Описова статистика:** середнє значення, медіана, стандартне відхилення.
5. **Пошук аномалій:** умовне форматування допомагає виявити нетипові значення.

Висновки

Табличні процесори – це універсальні інструменти для роботи з даними. Вони поєднують обчислення, аналіз, візуалізацію та автоматизацію, роблячи обробку інформації ефективною й доступною.

Питання для самоконтролю

І. Завдання на вибір правильної відповіді

1. Табличний процесор призначений для:
   a) редагування зображень
   b) створення презентацій
   c) роботи з числовими та текстовими даними у вигляді таблиць
   d) програмування на Python
2. Яка програма є табличним процесором?
   a) Microsoft Word
   b) Microsoft Excel
   c) Adobe Photoshop
   d) PowerPoint
3. Формула в Excel завжди починається зі знаку:
   a) +
   b) -
   c) =
   d) :
4. У якому форматі зберігаються дані, розділені комами?
   a) PDF
   b) DOCX
   c) CSV
   d) XLSX

5. Що робить функція =AVERAGE(A1:A10)?
   a) Рахує суму всіх чисел
   b) Знаходить середнє арифметичне
   c) Виводить найбільше значення
   d) Об’єднує текстові значення
6. Який тип діаграми краще використовувати для відображення частки кожного елемента в загальному обсязі?
   a) Лінійну
   b) Кругову
   c) Стовпчикову
   d) Точкову
7. Яке твердження **правильне** для умовного форматування?
   a) Змінює колір клітинки за певної умови
   b) Видаляє формули з таблиці
   c) Впорядковує дані за алфавітом
   d) Перетворює числа у текст
8. У Google Sheets кілька користувачів можуть одночасно працювати з файлом.
   a) Так
   b) Ні

II. Завдання на відповідність

Співвіднесіть функцію з її призначенням:

№	Функція	Призначення
1	SUM(A1:A10)	A. Визначає середнє значення діапазону
2	MAX(B1:B20)	B. Підраховує кількість числових значень
3	AVERAGE(C1:C5)	C. Знаходить найбільше число
4	COUNT(D1:D15)	D. Підсумовує всі значення

III. Питання відкритого типу (коротка відповідь)

1. У чому різниця між абсолютним і відносним посиланням у формулах?
2. Які переваги має формат CSV для зберігання даних?
3. Для чого використовують зведені таблиці (Pivot Tables)?
4. Наведи приклад ситуації, коли умовне форматування допомагає проаналізувати дані.

Тема. Інструменти для створення діаграм і схем

- ✓ Призначення та функції програм для створення блок-схем, організаційних діаграм, майнд-карт.
- ✓ Приклади інструментів: Microsoft Visio, draw.io, Lucidchart, XMind.
- ✓ Використання схем для моделювання бізнес-процесів, алгоритмів та проєктної документації.

Конспект лекції

Вступ

У сучасних обчислювальних системах важливе місце займають наочні засоби представлення інформації. Діаграми та схеми допомагають:

- структурувати великі обсяги даних;
- пояснити складні процеси простою візуальною мовою;
- підтримувати колективну роботу над проєктами;
- формувати документацію, яку легко сприймати користувачам і фахівцям.

Використання діаграм – це не лише зручність, а й інструмент ефективного управління інформацією та процесами.

Мета лекції: сформувати розуміння призначення програм для створення схем, блок-схем і діаграм; навчити основам моделювання бізнес-процесів і алгоритмів; ознайомити з популярними інструментами та їх застосуванням у проєктній документації.

Призначення діаграм і схем

- **Блок-схеми** – для відображення алгоритмів і послідовностей дій.
- **Організаційні діаграми** – для опису структури підприємств, відділів, команд.
- **Майнд-карти (карти думок)** – для візуалізації ідей, мозкового штурму, планування.
- **Діаграми процесів (BPMN, UML)** – для моделювання бізнес-процесів та системного аналізу.

Таким чином, діаграми дозволяють не тільки документувати, але й аналізувати та оптимізувати діяльність організацій чи алгоритми роботи програмного забезпечення.

Основні функції програм для створення діаграм

- Набір готових шаблонів і елементів (фігури, стрілки, блоки).

- Підтримка різних типів діаграм (блок-схеми, майнд-карти, ER-діаграми, UML).
- Інтеграція з іншими офісними й професійними програмами.
- Підтримка спільної роботи (онлайн-редагування, коментарі, доступ кількох користувачів).
- Експорт у різні формати (PDF, PNG, SVG, DOCX).

Приклади інструментів

- **Microsoft Visio** – професійне рішення для створення різних типів діаграм (корпоративний рівень, інтеграція з Microsoft 365).
- **draw.io (тепер diagrams.net)** – безкоштовний вебінструмент, зручний для швидкого створення блок-схем, UML, ER-діаграм.
- **Lucidchart** – онлайн-сервіс для командної роботи, має інтеграцію з Google Workspace, Slack, Jira.
- **XMind** – популярний інструмент для створення майнд-карт, зручний у плануванні, навчанні, організації ідей.

Використання в реальних завданнях

- **Моделювання бізнес-процесів** (наприклад, схема обслуговування клієнтів у банку).
- **Опис алгоритмів роботи програм** (блок-схема сортування даних, UML-діаграма для проєкту).
- **Проєктна документація** (ER-діаграма бази даних, організаційна структура компанії).
- **Освітні цілі** (схема історичного процесу, структура курсу чи лекції, карта понять).

Висновки

Інструменти для створення діаграм і схем є невід'ємною частиною сучасної роботи з інформацією. Вони:

- забезпечують **наочність**;
- сприяють **оптимізації процесів**;
- підвищують **ефективність командної роботи**;
- допомагають **швидко зрозуміти складні явища**.

У майбутньому зростатиме роль інтерактивних та інтелектуальних інструментів, що автоматично будуватимуть схеми на основі даних.

Питання для самоконтролю

1. Яке основне призначення блок-схем?

a) Візуалізація бізнес-процесів
b) Створення презентацій
c) Відображення алгоритмів і послідовностей дій
d) Збереження даних у таблицях

2. Який із наведених інструментів призначений для створення майнд-карт?

a) XMind
b) Excel
c) PowerPoint
d) MarkText

3. Виберіть правильний варіант: організаційна діаграма використовується для…

a) моделювання бізнес-процесів;
b) структурування команди, відділів, ієрархій;
c) створення інфографіки;
d) опису алгоритмів.

4. Які функції зазвичай мають сучасні інструменти для створення діаграм? ***(кілька** правильних відповідей)*

a) Набір шаблонів і фігур
b) Інтеграція з іншими сервісами
c) Автоматичне тестування програмного коду
d) Підтримка командної роботи

5. В якому форматі зазвичай можна експортувати створені схеми та діаграми?

a) JPG, PNG, PDF, SVG
b) EXE, BAT, CMD
c) MP3, WAV
d) DOCX, PPTX, XLSX

6. Який із інструментів є безкоштовним і зручним для онлайн-створення блок-схем?

a) Microsoft Visio
b) Lucidchart

c) draw.io (diagrams.net)
d) Figma

7. Яка з тенденцій розвитку інструментів для створення діаграм є найбільш актуальною?
a) Використання low-code/no-code платформ
b) Відмова від хмарних сервісів
c) Перехід до повністю текстового інтерфейсу
d) Заміна діаграм лише на текстові документи

Тема. Програмне забезпечення для візуалізації та презентації даних

- ✓ Інструменти: PowerPoint, Canva, Prezi, Figma.
- ✓ Візуалізація даних: інфографіка, діаграми.
- ✓ Засоби інтерактивних презентацій.

Конспект лекції

Вступ

У сучасному інформаційному середовищі важливо не лише мати дані, а й ефективно їх **подавати та візуалізувати**. Зрозуміла та приваблива візуалізація полегшує сприйняття інформації, сприяє прийняттю рішень і покращує взаємодію з аудиторією. Існує низка програмних інструментів, які дозволяють створювати презентації, інфографіку, діаграми та навіть інтерактивні звіти.

Мета лекції: познайомити студентів із сучасними засобами створення презентацій і візуалізації інформації; навчити принципам ефективного подання даних за допомогою діаграм, інфографіки та інтерактивних елементів; показати можливості сучасних онлайн-платформ для створення динамічних, наочних презентацій.

1. Інструменти для створення презентацій

PowerPoint

- **Тип**: Класичний інструмент для створення презентацій.
- **Можливості**:
  - Слайди з текстом, зображеннями, відео, графіками.
  - Анімація та переходи.
  - Інтеграція з Excel для вставки діаграм.
- **Переваги**:
  - Широке використання в освіті й бізнесі.
  - Зручність друку та експорту.
- **Недоліки**:
  - Менше сучасних інтерактивних функцій.

Canva

- **Тип**: Онлайн-сервіс для графічного дизайну та презентацій.
- **Можливості**:
  - Шаблони презентацій, постерів, інфографіки.
  - Спільна робота над дизайнами.

- Простий інтерфейс drag-and-drop.
- **Переваги**:
  - Величезна бібліотека графіки.
  - Не потребує встановлення.
- **Недоліки**:
  - Безкоштовна версія має обмеження.
  - Обмежений контроль над складними анімаціями.

Prezi

- **Тип**: Хмарна платформа для створення динамічних, **нелінійних** презентацій.
- **Особливості**:
  - Презентація у вигляді "карти", з наближенням/віддаленням до окремих фрагментів.
  - Висока візуальна привабливість.
- **Переваги**:
  - Ідеально для розповідей (storytelling).
  - Добре запам’ятовується аудиторією.
- **Недоліки**:
  - Не завжди підходить для строгої структури.
  - Потрібне стабільне інтернет-з'єднання.

Figma

- **Тип**: Інструмент для створення UI/UX-дизайну, прототипів і презентацій.
- **Можливості**:
  - Створення макетів, візуалізацій, прототипів.
  - Інтерактивність (натискання, переходи між екранами).
  - Колаборація в реальному часі.
- **Переваги**:
  - Потужна підтримка командної роботи.
  - Векторна графіка та компоненти.
- **Недоліки**:
  - Вища крива навчання порівняно з Canva чи PowerPoint.

2. Візуалізація даних

Це **графічне подання інформації**, яке допомагає краще зрозуміти дані, виявити закономірності, тенденції та аномалії.

Основні типи візуалізації:

- **Діаграми** (секторні, стовпчикові, лінійні, гістограми)
- **Інфографіка** (комбінація тексту, іконок, графіки для пояснення даних)
- **Теплові карти**, **bubble charts**, **timeline** тощо.

Інструменти для візуалізації:

- **Excel / Google Sheets** – базова побудова діаграм.
- **Canva / Figma** – створення графічної інфографіки.
- **Tableau / Power BI** *(не обов'язково в рамках цієї лекції)* – професійна аналітика.

3. Засоби інтерактивних презентацій

Інтерактивність у презентаціях дозволяє залучити аудиторію, створити діалог або адаптувати контент залежно від реакції слухачів.

▪ Приклади інтерактивних елементів:

- Гіперпосилання між слайдами (нелінійна навігація)
- Кнопки дій (наприклад, "перейти до розділу")
- Інтерактивні графіки (в Figma, Power BI)
- Інтеграція опитувань (Mentimeter, Slido)

▪ Інструменти для інтерактивності:

- **Prezi** – нелінійна структура подачі.
- **Figma** – створення клікабельних прототипів.
- **Mentimeter** – опитування, тести в реальному часі.
- **Google Slides + Add-ons** – інтеграції для голосувань, питань.

Висновки

- Існує багато інструментів для створення презентацій – від простих до професійних.
- Важливо обирати інструмент згідно з ціллю: для формальної доповіді підійде PowerPoint, для дизайну – Canva чи Figma, для візуального впливу – Prezi.
- Якісна візуалізація (графіки, діаграми, інфографіка) значно покращує сприйняття даних.
- Інтерактивні елементи роблять презентації живими та залучаючими.

Питання для самоконтролю

1. Вибір однієї правильної відповіді

1.1. Який інструмент дозволяє створювати нелінійні презентації з ефектами масштабування?

A) PowerPoint

B) Canva
C) Prezi
D) Figma

1.2. Яка програма найчастіше використовується для створення UI/UX-прототипів і має інтерактивні компоненти?

A) Google Slides
B) Figma
C) Prezi
D) Excel

1.3. Що з наведеного є прикладом інфографіки?

A) Презентація з 10 слайдів
B) Таблиця Excel з формулами
C) Візуальний постер з іконками, даними та графікою
D) Відеоінструкція

2. Вибір кількох правильних відповідей

2.1. Які з перелічених інструментів можна використовувати для створення презентацій?

PowerPoint
Canva
Prezi
Visual Studio

2.2. Які типи візуалізації даних є поширеними?

Діаграми
Інфографіка
Теплові карти
Компілятори

3. Так / Ні

3.1. Canva підтримує спільну роботу в режимі онлайн.
3.2. PowerPoint дозволяє створювати повноцінні інтерактивні прототипи з кнопками переходу.
3.3. Figma потребує встановлення на комп’ютер.

4. Встановлення відповідності

4.1. Встановіть відповідність між інструментом і його призначенням:

Інструмент	Призначення
A) PowerPoint	2) Класична презентація
B) Prezi	4) Динамічна, масштабована подача
C) Figma	1) UI/UX-дизайн і прототипи
D) Canva	3) Онлайн-дизайн інфографіки

5. Відкрите запитання

5.1. Поясніть різницю між інфографікою та діаграмою.

5.2. Назвіть два приклади інструментів для створення інтерактивних презентацій.

Тема. Інтеграція програмного забезпечення в обчислювальних системах

- ✓ Вступ до проблематики інтеграції програмного забезпечення.
- ✓ Поняття інтеграції програмного забезпечення
- ✓ Архітектури інтеграції.
- ✓ Технології та інструменти інтеграції.
- ✓ Типові проблеми інтеграції.
- ✓ Перспективи та тенденції.

Конспект лекції

Вступ

Сучасні обчислювальні системи вже давно перестали бути ізольованими. Більшість інформаційних технологій працюють у складних середовищах, де взаємодіють різні програми, сервіси й апаратні рішення. Уявімо типовий день людини: смартфон підключається до «розумного дому», додаток банку синхронізується з платіжними сервісами, студент користується навчальною платформою, яка інтегрує відеоконференції, електронні бібліотеки й системи тестування. Усі ці процеси можливі завдяки інтеграції програмного забезпечення.

Мета лекції – з'ясувати, що таке інтеграція ПЗ, які існують її рівні, архітектури та інструменти, а також чому вона є критично важливою для розвитку сучасних ІТ-рішень.

1. Вступ до проблематики інтеграції програмного забезпечення

1.1. Актуальність інтеграції ПЗ у сучасних ІТ-системах

1. **Зростання складності систем.**

Сучасні компанії й організації використовують десятки, а то й сотні різних програмних продуктів – від бухгалтерських систем до CRM і хмарних сервісів. Без інтеграції вони працюють розрізнено, що ускладнює управління та знижує ефективність.

2. **Необхідність обміну даними.**

В епоху цифровізації дані стали основним ресурсом. Їх потрібно швидко передавати між системами: від бази клієнтів до системи аналітики, від державного реєстру до мобільного застосунку («Дія» – яскравий український приклад).

3. **Економічна ефективність.**
   Інтеграція дозволяє уникнути дублювання функціоналу, зменшує витрати на підтримку різних систем і прискорює впровадження нових рішень.
4. **Гнучкість і масштабованість.**
   Бізнес і державні структури потребують рішень, які легко розширювати. Інтегровані системи простіше масштабувати: до них можна додавати нові сервіси без повного перероблення архітектури.
5. **Безпека та стандарти.**
   Інтеграція забезпечує централізований контроль доступу й дотримання міжнародних стандартів обміну даними (наприклад, у фінансах – ISO 20022, у медицині – HL7/FHIR).

Отже, інтеграція ПЗ – це не додатковий «комфорт», а стратегічна необхідність. Вона визначає ефективність роботи бізнесу, зручність для користувачів і конкурентоспроможність IT-рішень на глобальному ринку.

1.2. Приклади інтеграції ПЗ у різних сферах

1. **Бізнес та комерція**
   - **Інтернет-магазин.** Робота магазину неможлива без інтеграції: сайт → система управління товарами (ERP) → платіжні сервіси (LiqPay, PayPal) → служби доставки (Нова пошта, Meest).
   - **CRM-системи.** Наприклад, Salesforce або Bitrix24 об’єднують облік клієнтів, електронну пошту, телефонію, чат-боти в соцмережах і аналітику.
   - **Фінансові сервіси.** Банківські системи інтегруються з мобільними додатками, сервісами Apple Pay/Google Pay, а також міжнародними стандартами платежів.
2. **Освіта**
   - **Електронні платформи.** Moodle чи Google Classroom інтегрують календарі, відеоконференції (Zoom, Meet), електронні бібліотеки, системи тестування.
   - **Університетські інформаційні системи.** Єдина база студентів може інтегруватися з бухгалтерією, модулем розкладу, бібліотечною системою, електронним журналом успішності.
   - **Приклад з України.** «Єдина державна електронна база з питань освіти (ЄДЕБО)» об’єднує заклади освіти, МОН, вступні кампанії та студентські реєстри.
3. **Державні сервіси**
   - **«Дія».** Один із найуспішніших прикладів інтеграції: мобільний застосунок і портал, що об’єднують десятки державних реєстрів (паспорт, водійські права, бізнес, податки, медичні дані).

- **Prozorro.** Система електронних закупівель інтегрує державні органи, бізнес та громадськість у єдиній платформі прозорих торгів.
- **E-health (електронна медицина).** Інтеграція лікарень, аптеки, Національної служби здоров'я України (НСЗУ) та пацієнтів через єдину інформаційну систему.

4. **Інші сфери**
   - **Транспорт.** Інтеграція систем бронювання, мобільних додатків, GPS-моніторингу, електронних квитків.
   - **Енергетика та «розумні міста».** Системи обліку електроенергії, датчики вуличного освітлення, «розумні» лічильники інтегруються з аналітичними платформами.
   - **Охорона здоров'я.** Міжнародний стандарт HL7/FHIR дозволяє обмінюватися медичними даними між лікарнями, лабораторіями та страхових компаніями.

Таким чином, інтеграція ПЗ – універсальна тенденція, яка охоплює всі сфери діяльності: від побутових мобільних застосунків до державних і критичних інфраструктур. Вона підвищує зручність, прозорість і ефективність процесів.

1.3. Мета та завдання інтеграції програмного забезпечення

Мета інтеграції ПЗ – забезпечити узгоджену роботу різнорідних програмних і апаратних компонентів так, щоб система функціонувала як єдине ціле, була ефективною, гнучкою й зручною для користувачів.

Основні завдання інтеграції:

Узгодженість даних

Забезпечити обмін інформацією між різними програмами без дублювання чи втрати даних.

Приклад: автоматична синхронізація бази клієнтів у CRM та системі електронної пошти.

Сумісність компонентів

З'єднати програми, написані на різних мовах чи для різних платформ.

Приклад: веб-застосунок, створений на Java, який взаємодіє з базою даних Oracle і мобільним додатком на Android/iOS.

Автоматизація процесів

Мінімізувати ручні дії користувачів за рахунок автоматичного обміну даними між системами.

Приклад: онлайн-магазин автоматично формує замовлення в обліковій системі та передає дані службі доставки.

Підвищення продуктивності

Завдяки інтеграції зменшується час на виконання бізнес-операцій, зростає швидкість прийняття рішень.

Приклад: інтегровані фінансові системи дозволяють в реальному часі бачити всі рухи коштів.

Масштабованість та розвиток

Легке підключення нових сервісів без перебудови всієї інфраструктури.

Приклад: у мікросервісній архітектурі можна додати новий модуль (наприклад, чат-бот) без зміни ядра системи.

Безпека та контроль доступу

Централізоване управління правами користувачів, шифрування переданих даних.

Приклад: єдина система авторизації (Single Sign-On) для корпоративних сервісів.

Отже, інтеграція ПЗ – це не лише технічне завдання, а й стратегічний інструмент оптимізації бізнес-процесів, підвищення ефективності роботи організацій та якості сервісів для користувачів.

2. Поняття інтеграції програмного забезпечення

2.1. Визначення

Інтеграція програмного забезпечення (ПЗ) – це процес поєднання різних програмних систем, модулів і сервісів в єдину взаємодіючу структуру з метою забезпечення безперервного обміну даними, узгодженості бізнес-процесів і підвищення ефективності використання обчислювальних ресурсів.

У ширшому сенсі інтеграція означає створення «екосистеми» програмних продуктів, які функціонують як одне ціле.

2.2. Рівні інтеграції

1. **Апаратний рівень**
   - Об’єднання різних пристроїв і платформ у спільну систему.
   - Приклад: підключення датчиків «розумного дому» до центрального контролера.
2. **Системний рівень**
   - Узгодження роботи операційних систем, драйверів, системних бібліотек.
   - Приклад: сумісність між Linux-серверами та Windows-клієнтами в корпоративній мережі.
3. **Прикладний рівень**

- Взаємодія між прикладними програмами, базами даних, мобільними застосунками.
- Приклад: інтеграція CRM, ERP та онлайн-магазину в єдине бізнес-рішення.

2.3. Види інтеграції

1. **Горизонтальна інтеграція**
   - Об’єднання систем одного рівня чи типу.
   - Приклад: синхронізація кількох CRM-систем у міжнародній корпорації.
2. **Вертикальна інтеграція**
   - Взаємодія систем різних рівнів (від апаратного до прикладного).
   - Приклад: база даних ↔ серверна логіка ↔ веб-інтерфейс.
3. **Гібридна інтеграція**
   - Поєднання горизонтального й вертикального підходів.
   - Використовується у великих розподілених системах (наприклад, державні електронні сервіси).

2.4. Методи інтеграції

- **Point-to-Point (P2P):** прямі зв’язки між програмами.
- **Через шину даних (Enterprise Service Bus, ESB):** централізований канал для комунікації.
- **API-інтеграція:** використання інтерфейсів прикладного програмування (REST, gRPC).
- **Подієво-орієнтована інтеграція:** системи реагують на події в реальному часі (Kafka, RabbitMQ).

Отже, інтеграція ПЗ може відбуватися на різних рівнях і різними способами. Вибір підходу залежить від масштабу системи, її цілей і ресурсів.

3. Архітектури інтеграції

3.1. Традиційні підходи

1. **Монолітні системи**
   - Усі функції системи реалізуються в одному великому програмному модулі.
   - Переваги:
     - простота початкової розробки,
     - швидкий запуск,

        - менше проблем із сумісністю.
    - Недоліки:
        - складність масштабування,
        - важко оновлювати частини системи окремо,
        - зростання ризиків при збоях (падіння одного модуля може «покласти» всю систему).
    - Приклад: перші корпоративні облікові системи 90-х років.
2. **Інтеграція «точка-точка» (Point-to-Point, P2P)**
    - Кожна програма напряму «підключається» до іншої.
    - Переваги:
        - швидка реалізація для невеликої кількості систем,
        - відносна простота налаштувань.
    - Недоліки:
        - зі збільшенням кількості програм кількість зв’язків росте експоненціально (проблема «спагеті»),
        - важко відслідковувати потоки даних.
    - Приклад: інтеграція бухгалтерської програми з CRM через прямий обмін файлами.

3.2. Сучасні підходи

1. **Сервісно-орієнтована архітектура (SOA)**
    - Система складається з окремих сервісів, які взаємодіють через єдиний протокол.
    - Характерні риси: використання стандартів (SOAP, XML, WSDL), орієнтація на повторне використання.
    - Переваги: модульність, гнучкість, централізоване управління.
    - Недоліки: складність реалізації, залежність від «шини сервісів» (Enterprise Service Bus, ESB).
2. **Мікросервісна архітектура**
    - Система розбивається на невеликі незалежні сервіси, кожен виконує одну чітку функцію.
    - Взаємодія зазвичай через REST API або повідомлення.
    - Переваги:
        - незалежна розробка й оновлення сервісів,
        - легке масштабування,
        - висока стійкість до збоїв.
    - Недоліки:
        - потребує складнішої інфраструктури (Docker, Kubernetes),
        - складніше тестування та моніторинг.

- Приклад: Netflix, Amazon, Uber.

3. **Подієво-орієнтована архітектура (Event-Driven Architecture, EDA)**
   - Сервіси взаємодіють через обмін подіями (повідомленнями), часто в асинхронному режимі.
   - Використовуються брокери повідомлень (RabbitMQ, Apache Kafka).
   - Переваги:
     - висока швидкість обробки даних,
     - зручна для систем у реальному часі (біржі, IoT, онлайн-ігри).
   - Недоліки: складність налагодження й підтримки, ризики втрати подій при неправильних налаштуваннях.

3.3. Порівняння архітектур

Архітектура	Переваги	Недоліки	Приклади використання
Моноліт	Простота, швидкий старт	Важке масштабування, ризик збоїв	Старі ERP, локальні програми
Point-to-Point	Швидке підключення	«Спагеті»-архітектура, хаос	Малий бізнес, тимчасові рішення
SOA	Стандарти, модульність	Складність, залежність від ESB	Корпоративні системи
Мікросервіси	Гнучкість, масштабованість	Складна інфраструктура	Netflix, Amazon, Uber
Подієва інтеграція	Реактивність, швидкодія	Складність налагодження	IoT, фінансові біржі

Таким чином, розвиток архітектур інтеграції – це шлях від простих монолітів і «спагеті»-зв’язків до гнучких, масштабованих та стійких систем на базі мікросервісів і подієвої моделі. Вибір архітектури залежить від масштабу проєкту, ресурсів і стратегічних цілей організації.

4. Технології та інструменти інтеграції

4.1. Middleware (проміжне програмне забезпечення)

- **Визначення:** програмний шар між операційною системою та прикладними застосунками, який забезпечує їхню взаємодію.
- **Приклади:**
  - **Message Brokers (брокери повідомлень):** RabbitMQ, Apache Kafka – передають повідомлення між сервісами.

    - **Системи черг:** ActiveMQ, ZeroMQ – дозволяють керувати асинхронною комунікацією.
    - **Enterprise Service Bus (ESB):** Mule ESB, WSO2 – централізовані шини для корпоративних інтеграцій.
- **Застосування:** інтеграція банківських систем, e-commerce, IoT, державних платформ.

4.2. API та протоколи інтеграції

- **API (Application Programming Interface):** набір методів і протоколів, що дозволяє одній програмі взаємодіяти з іншою.
- **Основні технології:**
    - **REST API:** простий, використовує HTTP та JSON; підходить для більшості веб-додатків.
    - **gRPC:** високопродуктивний протокол від Google, базується на HTTP/2 і Protobuf.
    - **GraphQL:** дозволяє клієнту отримувати лише ті дані, які потрібні (розробка Facebook).
    - **SOAP:** більш старий протокол, орієнтований на строгі стандарти безпеки та формати (XML).
- **Приклад:** ПриватБанк відкрив **публічне REST API** для розробників, завдяки чому з'явилися десятки інтегрованих сервісів у фінансовій сфері.

4.3. Інтеграційні платформи

- **MuleSoft (Anypoint Platform):** корпоративна інтеграційна платформа, що підтримує API-led інтеграцію.
- **WSO2:** відкрите ПЗ для інтеграції, орієнтоване на SOA та мікросервіси.
- **Apache Camel:** легковаговий фреймворк для інтеграції з використанням «маршрутів» даних.
- **Zapier, Integromat (Make):** no-code/low-code інтеграційні сервіси, що дозволяють об'єднувати веб-додатки без програмування.
- **Застосування:** бізнес-процеси, аналітика, хмарні екосистеми, державні сервіси (наприклад, Prozorro інтегрується через API з іншими системами).

4.4. Контейнеризація та оркестрація

- **Docker:** дозволяє запускати застосунки в ізольованих контейнерах, що робить інтеграцію простішою та передбачуваною.
- **Kubernetes:** система оркестрації контейнерів, що керує масштабуванням, балансуванням навантаження, відмовостійкістю.

- **Helm:** менеджер пакетів для Kubernetes, що полегшує розгортання складних інтеграційних систем.
- **Приклад:** у мікросервісній архітектурі кожен сервіс запускається у власному контейнері Docker, а Kubernetes забезпечує їхню взаємодію.

4.5. Хмарні технології інтеграції

- **iPaaS (Integration Platform as a Service):** інтеграція «як сервіс».
- Приклади: Dell Boomi, Microsoft Azure Logic Apps, Google Cloud Apigee.
- Дозволяють налаштовувати інтеграцію через веб-інтерфейс без складної інфраструктури.
- Використовуються для швидкого з'єднання SaaS-сервісів (CRM, ERP, email-маркетинг, хмарне сховище).

Отже, технології інтеграції різняться за рівнем складності – від простих API та no-code рішень до масштабних корпоративних платформ і контейнеризованих середовищ. Правильний вибір інструментів визначає ефективність роботи всієї системи.

5. Проблеми та виклики інтеграції

5.1. Сумісність систем

- **Гетерогенність технологій:** різні мови програмування, формати даних, протоколи.
- **Версійність:** старі системи (legacy) часто не мають API або працюють із застарілими стандартами.
- **Приклад:** інтеграція сучасного веб-додатку з банківським «мейнфреймом» 80-х років вимагає створення спеціальних адаптерів.

5.2. Масштабованість

- **Проблема:** збільшення кількості користувачів або сервісів може призвести до перевантаження каналів інтеграції.
- **Рішення:** використання мікросервісів, хмарних платформ, балансування навантаження (Load Balancing).
- **Приклад:** у системах онлайн-торгівлі «Чорна п'ятниця» тестує здатність інтеграційної інфраструктури витримувати пікові навантаження.

5.3. Безпека

- **Ризики:**
  - витік даних під час обміну,

    - атаки «людина посередині» (MITM),
    - підробка або несанкціонований доступ через API.
- **Методи захисту:**
    - шифрування (TLS, SSL),
    - аутентифікація (OAuth 2.0, JWT),
    - аудит і логування транзакцій.
- **Приклад:** у медичних системах (E-Health) інтеграція повинна гарантувати захист персональних даних пацієнтів відповідно до GDPR.

5.4. Надійність і відмовостійкість

- **Проблема:** збій одного модуля може паралізувати всю інтегровану систему.
- **Рішення:**
    - резервування каналів,
    - асинхронна обробка даних (черги повідомлень),
    - автоматичний перезапуск контейнерів (Kubernetes).
- **Приклад:** системи авіакомпаній повинні працювати 24/7, інакше будь-який збій зупинить продаж квитків у всьому світі.

5.5. Вартість і складність реалізації

- **Фінансові витрати:** ліцензії на інтеграційні платформи, серверні ресурси, навчання персоналу.
- **Організаційні витрати:** потреба у команді DevOps, архітекторах інтеграції, аналітиках.
- **Приклад:** впровадження SAP ERP з інтеграцією у великій компанії може тривати роками та коштувати мільйони доларів.

5.6. Управління змінами

- **Проблема:** будь-яка зміна в одній системі може «зламати» інтеграцію з іншими.
- **Рішення:**
    - версіонування API,
    - використання контрактного тестування,
    - моніторинг і CI/CD-процеси.
- **Приклад:** у державних платформах, як-от «Дія», зміни в одному сервісі повинні бути узгоджені з усіма підключеними модулями.

Отже, головні виклики інтеграції – це технічна сумісність, масштабованість, безпека, стійкість до збоїв і економічна доцільність. Подолати

їх можна лише комплексним підходом: правильним вибором архітектури, сучасних інструментів і ефективною організацією процесів.

6. Перспективи та майбутні тенденції інтеграції

6.1. Автоматизація інтеграційних процесів

- **iPaaS + RPA:** поєднання платформ інтеграції як сервісу з роботизованою автоматизацією процесів (UiPath, Blue Prism).
- **Тенденція:** системи самостійно визначатимуть оптимальні шляхи обміну даними.
- **Приклад:** автоматичне підключення нового SaaS-додатку до корпоративної екосистеми без ручного налаштування.

6.2. Використання штучного інтелекту (AI)

- **AI-інтеграція:** розпізнавання форматів даних, автоматичне створення конекторів між системами.
- **Інтелектуальний моніторинг:** виявлення аномалій у потоках даних (наприклад, кібератаки).
- **Приклад:** AI-помічники, які інтегрують календар, пошту й CRM, пропонуючи користувачу оптимальні дії.

6.3. Low-code / No-code інтеграції

- **Тренд:** розробка інтеграцій без програмування, за допомогою drag-and-drop інтерфейсів.
- **Інструменти:** Zapier, Make (Integromat), Microsoft Power Automate.
- **Наслідки:** інтеграція стає доступною навіть нефахівцям, що пришвидшує цифрову трансформацію бізнесу.

6.4. Edge Computing

- **Суть:** обробка даних ближче до джерела (на «краю» мережі), а не лише у хмарі.
- **Значення для інтеграції:**
  - зменшення затримок,
  - оптимізація трафіку,
  - підвищення безпеки.
- **Приклад:** «розумне місто», де камери, сенсори та датчики інтегруються локально, а в хмару відправляються лише агреговані дані.

6.5. Глобальна стандартизація API

- **Тенденція:** поступовий перехід до єдиних протоколів і описів API (OpenAPI, AsyncAPI).
- **Результат:** зменшення проблем сумісності та пришвидшення інтеграції між компаніями та державами.
- **Приклад:** ініціатива **Open Banking** у фінансовій сфері, яка дозволяє банкам і фінтех-компаніям швидко інтегрувати сервіси.

6.6. Кібербезпека як ключовий фактор

- Зростання кількості кіберзагроз робить **безпеку інтеграцій** основним пріоритетом.
- Очікуваний розвиток: вбудовані системи захисту (Zero Trust Architecture), автоматичний аудит безпеки API.

Отже**,** майбутнє інтеграції – це **автоматизація, AI, доступність для нефахівців, edge computing і підвищена безпека**. Інтеграція стає стратегічним фактором розвитку будь-якої цифрової екосистеми – від стартапів до державних платформ.

Висновок

Інтеграція програмного забезпечення в обчислювальних системах є ключовим процесом розвитку сучасних IT. Вона забезпечує взаємодію між різними програмами, сервісами та апаратними платформами, створюючи єдині інформаційні екосистеми.

Протягом останніх десятиліть архітектури інтеграції пройшли шлях від простих монолітів і зв’язків «точка-точка» до складних та масштабованих рішень на основі SOA, мікросервісів та подієво-орієнтованих моделей. Кожен підхід має свої переваги й обмеження, а вибір архітектури залежить від конкретних завдань, ресурсів і масштабу проєкту.

Сучасні технології інтеграції – middleware, API, контейнеризація, хмарні та low-code інструменти – значно прискорюють і спрощують процес об’єднання систем. Водночас інтеграція супроводжується низкою викликів: забезпеченням сумісності, масштабованості, безпеки, надійності та економічної доцільності.

Майбутнє інтеграції визначається автоматизацією, використанням штучного інтелекту, розвитком low-code платформ, поширенням edge computing і глобальною стандартизацією API. При цьому питання кібербезпеки стають критично важливими, адже саме від них залежить довіра користувачів та ефективність цифрової інфраструктури.

Таким чином, інтеграція ПЗ є не лише технічним завданням, а й стратегічним чинником цифрової трансформації суспільства, бізнесу й

держави. Від якості її реалізації залежить конкурентоспроможність компаній, зручність користувачів і безпека даних у сучасному цифровому світі.

Питання для самоконтролю

Тестові питання (True/False)

1. Монолітна архітектура легко масштабується й дозволяє незалежно оновлювати окремі модулі.

2. Інтеграція «точка-точка» (Point-to-Point) створює проблему «спагеті»-зв’язків при збільшенні кількості систем.

3. REST API зазвичай використовує JSON як формат передачі даних.

4. Middleware є проміжним шаром, що забезпечує взаємодію між прикладними програмами та користувачами напряму, минаючи ОС.

5. Подієво-орієнтована архітектура (EDA) зручна для обробки даних у реальному часі.

6. Мікросервіси завжди легші в управлінні, ніж монолітні системи.

7. Використання контейнерів (Docker) сприяє передбачуваній інтеграції застосунків.

8. Legacy-системи інтегруються легко, оскільки завжди мають сучасні API.

Тестові питання (закритого типу)

1. Яка з наведених архітектур характеризується проблемою «спагеті»-зв’язків?
   - a) Монолітна
   - b) Point-to-Point
   - c) SOA
   - d) Мікросервісна

2. Яка головна перевага мікросервісної архітектури?
   - a) Простота реалізації
   - b) Централізоване керування всіма модулями
   - c) Незалежне масштабування та оновлення сервісів

- d) Відсутність потреби в оркестрації

3. Який протокол інтеграції найчастіше використовується для веб-застосунків завдяки простоті та підтримці JSON?
   - a) SOAP
   - b) REST
   - c) GraphQL
   - d) gRPC

4. Яке з наведених рішень належить до **middleware**?
   - a) Kubernetes
   - b) RabbitMQ
   - c) Docker
   - d) Git

5. Який виклик інтеграції пов’язаний із ризиком втрати даних при неправильному налаштуванні системи?
   - a) Масштабованість
   - b) Вартість
   - c) Надійність і відмовостійкість
   - d) Управління змінами

Тестові питання (відкритого типу)

1. Які переваги та недоліки має сервісно-орієнтована архітектура (SOA)?
2. Чому інтеграція старих (legacy) систем є особливо складним завданням?
3. Поясніть різницю між **горизонтальною** та **вертикальною інтеграцією**.
4. Які перспективи відкриває використання **AI** в інтеграційних процесах?
5. Як технології контейнеризації (Docker, Kubernetes) полегшують інтеграцію?

Список рекомендованих джерел

1. Авраменко В. С., Авраменко А. С. Основи операційних систем : навч. посібник. Черкаси : ЧНУ імені Богдана Хмельницького, 2018. 524 с. URL: https://eprints.cdu.edu.ua/1480/1/osnovu.pdf (дата звернення: 05.01.2026).
2. Довідка LibreOffice 25.8. URL: https://help.libreoffice.org/latest/uk/text/shared/05/new_help.html?&DbPAR=SHARED&System=WIN (дата звернення: 05.01.2026).
3. Гнатовська Г.А Конспект лекцій з дисципліни «XML-технології» для студентів 4 курсу денної форми навчання спеціальності 122 – «Комп'ютерні науки». Одеса, 2019. – 77 с. URL: http://eprints.library.odeku.edu.ua/id/eprint/7433/7/Конспект-XML-v1.pdf (дата звернення: 02.01.2026)
4. Конспект лекцій з дисципліни «Системне програмне забезпечення». Краматорськ : Донбаська держ. машинобудівна академія, 2020. 98 с. URL: http://www.dgma.donetsk.ua/docs/kafedry/avp/metod/СПЗ%20Конспект%20лекцій.pdf (дата звернення: 05.01.2026).
5. Короткі посібники користувача для програм пакета Office 2013. URL: https://support.microsoft.com/uk-ua/office/короткі-посібники-користувача-для-програм-пакета-office-2013-4a8aa04a-f7f3-4a4d-823c-3dbc4b8672a1 (дата звернення: 05.01.2026).
6. Махней О. В. Практикум з LaTeX : метод. рекомендації. Івано-Франківськ : Голіней, 2018. 36 с. URL: https://kdrpm.pnu.edu.ua/wp-content/uploads/sites/55/2018/03/LaTeX_mv_el.pdf (дата звернення: 05.01.2026).
7. Огляд Canva Docs та посібник із початку роботи. URL: https://www.canva.com/uk_ua/help/about-canva-docs/ (дата звернення: 05.01.2026).
8. Операційна система Windows 10. URL: https://cpto.dp.ua/public_html/posibnyky/OSWin10/zmist.htm (дата звернення: 05.01.2026).
9. Петренко О. Я., Бондаренко В. В. Цифрові інструменти Google : навч. посібник. Київ : ІПДО, 2022. 73 с. URL: https://dspace.nuft.edu.ua/server/api/core/bitstreams/71d1d043-2858-4065-b778-39579a3d45e8/content (дата звернення: 05.01.2026).
10. Посібник зі спеціальних можливостей для користувачів Google Workspace. URL: https://support.google.com/accessibility/answer/1631886?hl=uk#docs (дата звернення: 05.01.2026).
11. Програмне забезпечення обчислювальної техніки. URL: https://sites.google.com/bcsd.ukr.education/metelska/предмети/д-обчислювальна-техніка-та-програмування/апаратне-та-програмне-забезпечення-пк/програмне-забезпечення-обчислювальної-техніки (дата звернення: 05.01.2026).

12. Путівник по Markdown. URL: https://markdown.rozh2sch.org.ua (дата звернення: 05.01.2026).
13. Сервіс для розгалужених презентацій Prezi. URL: https://www.youtube.com/watch?v=N7nDxZYKAoA (дата звернення: 05.01.2026)
14. Як створити діаграму ER у Draw.io за допомогою повних інструкцій. URL: https://www.mindonmap.com/uk/blog/drawio-er-diagram/ (дата звернення: 05.01.2026).
15. Якименко І. З. Опорний конспект лекцій з дисципліни «Безпека програм та даних» для студентів спеціальності «Кібербезпека». Тернопіль, 2019. 50 с. URL: https://dspace.wunu.edu.ua/bitstream/316497/36004/1/Безп_прогр_даних_LEC_2019.pdf (дата звернення: 05.01.2026).
16. Figma Help Center. URL: https://help.figma.com/ (дата звернення: 05.01.2026).
17. Git Documentation. URL: https://git-scm.com/doc (дата звернення: 05.01.2026).
18. Hohpe G., Woolf B. Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Updated ed. Boston : Addison-Wesley Professional, 2023. 736 p.
19. JetBrains Documentation. URL: https://www.jetbrains.com/help/ (дата звернення: 05.01.2026).
20. Lucidchart Tutorials. Lucidchart Help Center. URL: https://www.lucidchart.com/pages/tutorial (дата звернення: 05.01.2026).
21. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. Sebastopol : O'Reilly Media, 2021. 612 p.
22. Notepad++ User Manual. URL: https://npp-user-manual.org (дата звернення: 05.01.2026).
23. Overleaf Documentation. URL: https://www.overleaf.com/learn (дата звернення: 05.01.2026).
24. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 9th ed. New York : McGraw-Hill, 2020. 976 p.
25. Thomas D., Hunt A. The Pragmatic Programmer: Your Journey To Mastery. 20th Anniversary ed. Boston : Addison-Wesley Professional, 2019. 352 p.
26. Visual Studio IDE documentation. URL: https://learn.microsoft.com/uk-ua/visualstudio/ide/?view=vs-2022 (дата звернення: 05.01.2026).
27. Windows 2010: Навчальний посібник / Укладач: С.Ф. Дячук. Тернопіль : Вид-во ТНТУ ім. Івана Пулюя, 2021. 144 с.
28. XML for the Complete Beginner. Birmingham : Packt Publishing, 2024. 350 p.
29. XML Tutorial. W3Schools. URL: https://www.w3schools.com/xml/ (дата звернення: 05.01.2026).