Частина I
Основи програмування

Розділ 1. Базові поняття програмування

Розділ 2. Алгоритмічні структури

Розділ 3. Розробка програм

Розділ 4. Програмування у візуальному середовищі

1 Базові поняття програмування

Повторення

1. Які інформаційні процеси ви знаєте?
2. Що таке інформаційна система та які складові вона має?
3. Як функціонує технічна інформаційна система?
4. Що таке об'єкт і які властивості він має?

Сучасна людина успішно користується численними технічними системами, не замислюючись над їхньою внутрішньою будовою. Не становлять винятку і програми, які є різновидом інформаційних технічних систем. І хоча досконало розуміти роботу сотень засобів і механізмів непотрібно та неможливо, мати загальні уявлення про їхню структуру та принципи функціонування корисно і цікаво. Наприклад, водієві не обов'язково знати, як функціонує двигун внутрішнього згорання чи система подачі палива, однак він повинен принаймні відрізняти бензобак від бачка для омивача скла, мати уявлення про коробку передач та вміти виміряти рівень масла у двигуні.

У цій частині підручника у загальних рисах буде описано, як програми влаштовані «зсередини». Ви зазирнете на «внутрішню кухню» програмного забезпечення, зрозумієте, що відбувається у програмі, коли користувач виконує ту чи іншу дію, наприклад клацає кнопку. Більше того, ви створите власну найпростішу програму. А почнемо ми з розгляду базових понять програмування.

Що таке програма

Та перш ніж розпочинати дослідження внутрішньої будови програм, варто відповісти на питання: «Що таке програма? Чим вона відрізняється від непрограмних об'єктів?». Зробити це достатньо легко, якщо пригадати склад та принципи функціонування технічних інформаційних систем, які ви вивчали у 9 класі. Схему такої системи зображено на рис. 1.1.

Нагадаємо, що відмінною ознакою інформаційної системи є те, що в ній або за її допомогою виконуються певні інформаційні процеси. А програмне забезпечення (сукупність програм), як видно з рис. 1.1, відповідає насамперед за процес обробки даних: дані надходять до програм, обробляються і передаються апаратному забезпеченню. Звичайно, програми можуть брати участь не лише в обробці даних, а й у інших інформаційних процесах: отриманні, передаванні, пошуку, захисті даних тощо, однак обробка даних — це той процес, що ніяк не може відбуватися без участі програм.

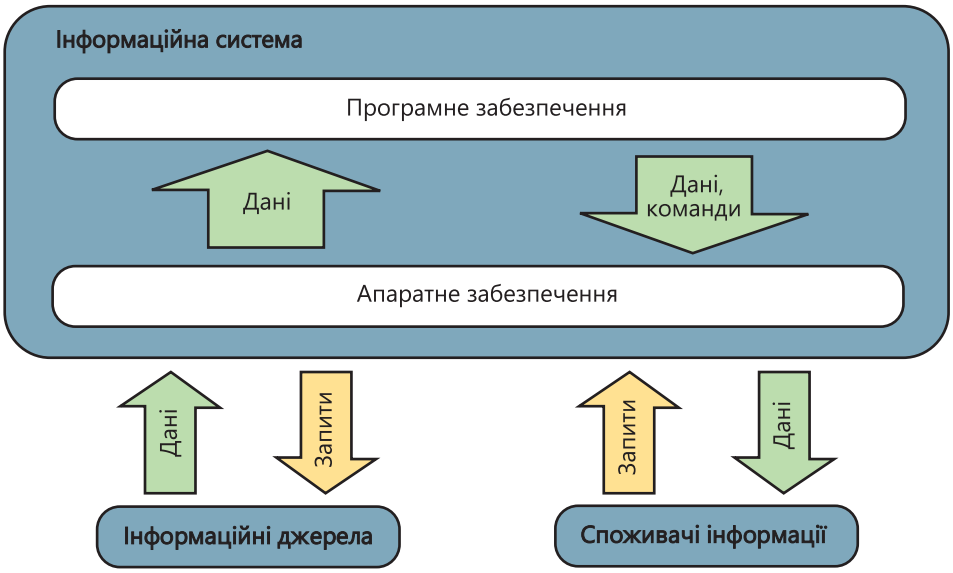


Рис. 1.1. Схема технічної інформаційної системи

Крім того, програмне забезпечення визначає поведінку системи, адже подає команди, що виконуються апаратним забезпеченням. Зауважимо, що окрема програма визначає поведінку системи не завжди, а лише в той час, коли *виконується*, тобто коли апаратне забезпечення здійснює вказівки саме цієї програми.

Програма — це складова інформаційної системи, що виконує обробку даних та може визначати поведінку системи.

Образно кажучи, програмне та апаратне забезпечення технічної інформаційної системи взаємодіють так само, як розум і тіло людини. Ця аналогія тим більше доречна, що основна мета створення технічних інформаційних систем — замінити людину в окремих видах діяльності, і тому будь-яка така система в чомусь наслідує людину або імітує її дії. Зокрема, комп'ютер імітує (хоча й дуже недосконало) розумову діяльність людини. Та властивість програм, що вони протягом деякого часу виконуються, а решту часу неактивні, теж має аналогію зі способом життя людини, а саме з періодами її сну та активності.

Складові програми

В означенні програми вжито доволі незвичне, якщо йдеться про технічні системи, слово «поведінка». Адже зазвичай ми говоримо про поведінку лише живих істот і розуміємо під цим словом типові схеми дій у тих чи інших ситуаціях, способи реагування на зовнішні події. Річ у тім, що програмовані системи теж діють за тією чи іншою схемою залежно від ситуації і теж реагують на певні події, наприклад натискання клавіш чи переміщення миші. Створюючи спрощений аналог людського розуму, винахідники комп'ютера «навчили» цей пристрій аналізувати дані

й робити прості логічні висновки. Наприклад, коли під час роботи в Microsoft Word ми натискаємо клавішу **Del**, комп'ютер аналізує, чи виділено у цей момент фрагмент тексту. Якщо ні, то буде видалено лише один символ, якщо фрагмент виділений, то видалений буде він увесь. Таким чином, комп'ютер реалізує певну логіку дій, **і ця логіка, звичайно**, «записана» у програмах. *Програмна логіка* — одна з трьох головних складових програмного забезпечення.

Іншою важливою складовою програм є *структури даних*. Дані, які програма обробляє, надходять до неї і зберігаються не в хаотичному вигляді, а в добре організованих структурах. Від того, якими саме будуть ці структури, значною мірою залежить і програмна логіка, і можливості програми в цілому.

Як уже зазначалося, програми керують роботою апаратного забезпечення. А хто керує самими програмами, хто створює ті ситуації та події, від яких залежить поведінка програм? Відповідь на це запитання така: вказівки програмам дають або користувачі, або інші програми. Засоби, за допомогою яких з програмою взаємодіє користувач або інші програми, називаються *інтерфейсом* — це третя важлива складова програмного забезпечення. Слід відрізняти *інтерфейс користувача*, через який з програмою взаємодіє користувач, від *інтерфейсу прикладного програмування* (англ. API — Application Programming Interface), за допомогою якого програми обмінюються даними з іншими програмами. Зазначимо, що деякі програми, наприклад драйвери, не мають інтерфейсу користувача, однак частка таких програм серед всього програмного забезпечення незначна.

Основні складові програми — **логіка**, **структури даних** та **інтерфейс**. Логіка визначає поведінку програми, структури даних — спосіб зберігання даних, а інтерфейс є засобом взаємодії програми з користувачем та іншими програмами.

Характерні особливості сучасних прикладних програм

Вище ми розглянули призначення та складові програм узагалі, а зараз звернімо увагу на окремий різновид програм, яким ми користуємось найчастіше, а саме сучасні прикладні програми. Працюючи з такими програмами, ви виконували дії над певними об'єктами: командами меню, кнопками, прапорцями, клітинками електронної таблиці, фрагментами тексту тощо. Можна сказати, що об'єкт — це основна структура даних у сучасних програмах, які тому і називають *об'єктно-орієнтованими*.

Нагадаємо, що *об'єктом* прийнято називати єдине ціле, яке можна відрізнити від іншого цілого. Кожен об'єкт має певний набір параметрів (наприклад, якщо йдеться про кнопку у програмі, то це розмір, колір тла, тип шрифту, яким зроблено напис на кнопці, тощо). *Стан об'єкта* — це сукупність значень його параметрів у певний момент часу. Зокрема

стан кнопки можна описати так: розмір 50×100 пікселів, колір — сірий, шрифт — Arial, ненатиснута.

Крім стану об'єкт має *поведінку* — сукупність дій, які він може виконувати. Зауважимо, що не лише сам об'єкт здатен виконувати дії, а й над ним можуть виконувати дії інші об'єкти. Так, користувач може клацнути кнопку або перемістити фрагмент тексту — це дії користувача над кнопкою та фрагментом тексту.

Одні дії можуть спричиняти інші. Наприклад, клацання кнопки  приводить до того, що виділений фрагмент тексту копіюється в буфер обміну. У програмуванні ті дії, які спричиняють інші, називають *подіями*. Програмні об'єкти виконують дії не самі по собі, не раптово, а лише у відповідь на настання тієї чи іншої події — у цьому полягає основний принцип *подійно-орієнтованого програмування*. Переважна більшість програм, з якими ви працюєте, є не лише об'єктно-орієнтованими, а й подійно-орієнтованими — це друга характерна особливість сучасного програмного забезпечення.

Головні особливості сучасних прикладних програм — це зберігання даних у вигляді об'єктів та виконання дій у результаті настання тих чи інших подій. Таку властивість програм називають **об'єктною** та **подійною орієнтованістю**.

Зауважимо, що у програмуванні слово «об'єкт» має значно вужче значення, ніж у загальнофілософському розумінні. Програмний об'єкт — це один із різновидів структур даних, а всі інші елементи об'єктно-орієнтованих програм об'єктами не вважаються.

Структура та принцип роботи сучасних програм

У результаті настання події виконується не вся програма, а лише певний її фрагмент, що його називають *обробником події*. Зауважте, що події завжди зв'язані з якимись об'єктами. Так, подія клацання кнопки зв'язана з певною кнопкою, розкриття списку — з розкривним списком тощо. Така подія, як наведення на об'єкт вказівника миші, може бути зв'язана майже з будь-яким програмним об'єктом, а подія настання певного моменту часу завжди зв'язана зі спеціальним об'єктом «Таймер».

Отже, структуру та загальний принцип функціонування сучасної прикладної програми можна зобразити так, як показано на рис. 1.2. Програма — це набір об'єктів, з кожним із яких пов'язані певні події. У результаті настання подій виконуються фрагменти програми, що отримали назву обробників подій.

Оскільки події зв'язані з об'єктами, то логічно було б зв'язати з об'єктами і обробники подій. Власне, так і є: в об'єктно-орієнтованій програмі кожен обробник події зв'язано з якимось об'єктом (щоправда, не завжди з тим, з яким зв'язано подію). Більше того, з об'єктами зв'язано не лише

обробники подій, а й фрагменти програми, що виконують будь-які дії взагалі. Такі фрагменти називають *методами об'єктів*. Отже, підіб'ємо підсумки:

- об'єкти містять не тільки дані, а й методи;
- кожен метод призначено для виконання певної дії;
- усі функції програми «розкладено», як у скриньки, в окремі методи.

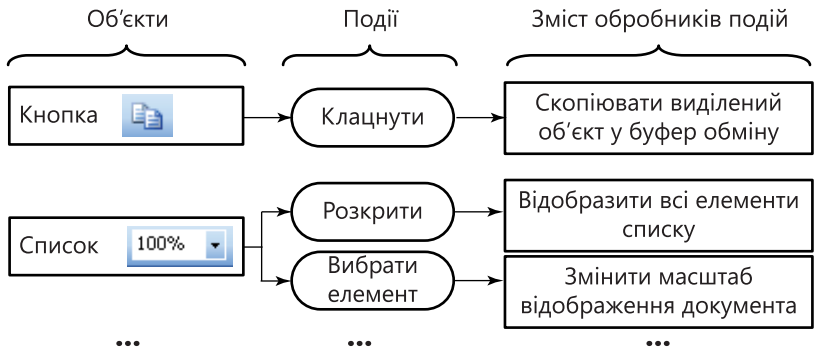


Рис. 1.2. Структура сучасної прикладної програми

Поняття алгоритму

На рис. 1.2 зображена лише загальна, крупноблочна структура програми. Спробуємо перейти на глибший рівень деталізації і зазирнути всередину обробників подій. Поміркуйте над тим, з чого вони можуть складатися. Звичайно, в обробнику події не може бути записано українською мовою текст на кшталт «змінити масштаб відображення документа», адже комп'ютер українською не читає і такого тексту не зрозуміє. Відповідь полягає в тому, що в кожному обробнику події описано певний *алгоритм*.

Алгоритм — це послідовність інструкцій, виконання яких дозволяє розв'язати певну задачу за скінченний час.

Алгоритм завжди розраховано на певного *виконавця*, для якого інструкції алгоритму мають бути зрозумілими настільки, щоб він міг їх виконувати автоматично, не застосовуючи розумову діяльність. Останнє положення важливе, адже виконавцем може бути не тільки людина, а й не наділена розумом істота — механізм або інформаційна система. Зокрема, виконавцем алгоритмів, що містяться у програмах, є комп'ютер.

Алгоритми можна подавати в найрізноманітніших формах: текстовій, графічній, звуковій тощо. Оскільки алгоритм — це саме послідовність інструкцій, а не якийсь невпорядкований набір, то подавати його потрібно так, щоб виконавцеві було зрозуміло, у якому порядку ці інструкції виконувати. Тому, записуючи алгоритми в текстовій формі, його інструкції зазвичай нумерують, а коли алгоритм подано графічно, порядок виконання інструкцій вказують стрілками. Так, на рис. 1.3 у текстовій та графічній формах зображено алгоритм переходу вулиці на перехресті зі світлофором.

1. Підійти до пішохідного переходу.
2. Якщо на світлофорі для пішоходів зелений сигнал, перейти вулицю.
3. Інакше дочекатися зеленого сигналу і перейти вулицю.

а

б

Рис. 1.3. Алгоритм переходу вулиці: *а* — текстова форма; *б* — графічна форма

Алгоритми, за якими працюють програми (а точніше, методи об'єктів), найчастіше записують у текстовій формі, але не природною людською мовою, а *мовою програмування*, призначеною спеціально для опису дій програми. Складати алгоритми та записувати їх мовою програмування — основний професійний обов'язок *програмістів*.

Для допитливих. Сьогодні слово «програміст» — це назва цілої низки професій, така ж загальна, як, скажімо, «лікар». Власне складанням алгоритмів та записом їх мовою програмування займаються *програмісти-кодувальники*. Загальну архітектуру програм проектують *програмісти-архітектори*. *Програмісти-тестувальники* перевіряють створювані прототипи програм та окремих програмних модулів. Загалом програмістських професій існує кілька десятків.

Приклад 1.1

Розглянемо структуру і принцип дії однієї з найпростіших програм — калькулятора, що може виконувати чотири арифметичні дії. Інтерфейс такої програми зображено на рис. 1.4.

Принцип дії програми надзвичайно простий: у поля **Число 1** та **Число 2** користувач вводить два числа, після чого натискає кнопку однієї з арифметичних операцій і отримує результат її застосування до введених чисел. Отже, програма реагує на чотири події — клацання чотирьох кнопок арифметичних операцій. Зміст обробників трьох подій очевидний: вони мають просто записати в поле результату суму,

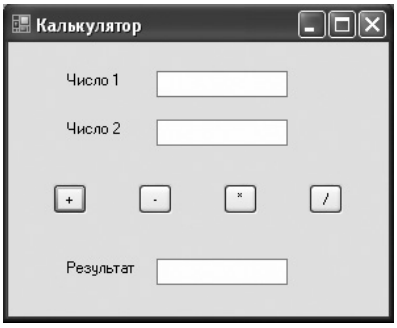


Рис. 1.4. Інтерфейс програми Калькулятор

різницю чи добуток двох чисел. Обробник клацання кнопки ділення працює за дещо цікавішим алгоритмом: він має перевірити, чи не дорівнює друге число нулю, і, якщо дорівнює, вивести повідомлення про помилку, а інакше виконати ділення і відобразити результат. Подійно-орієнтовану архітектуру програми-калькулятора зображено на рис. 1.5.

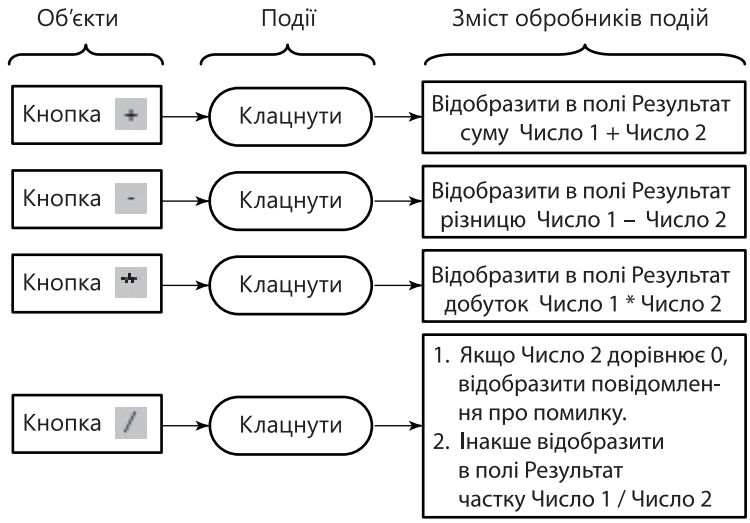
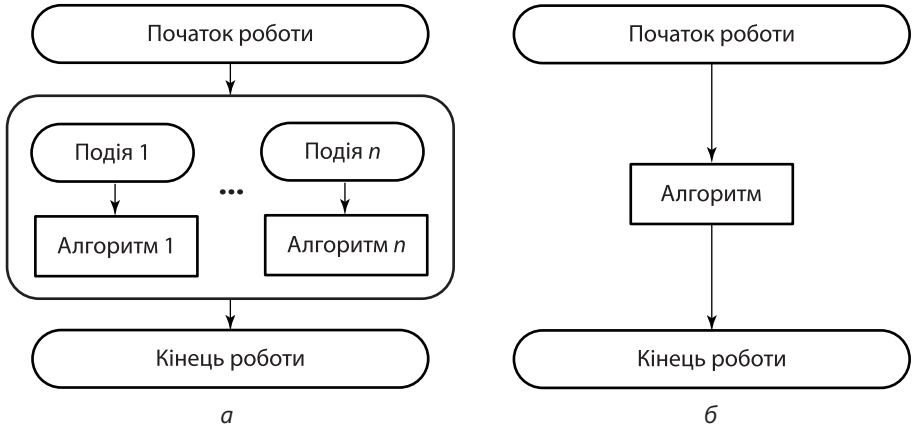


Рис. 1.5. Подійно-орієнтована архітектура програми Калькулятор

Екскурс в історію програмного забезпечення

Отже, подійно-орієнтована програма реалізує багато алгоритмів, кожен з яких виконується за настання певної події (рис. 1.6, *а*). Однак подійно- та об'єктно-орієнтоване програмне забезпечення домінувало не завжди. До кінця 80-х років XX століття переважна більшість програм працювала за простішим принципом: вони реалізовували лише один алгоритм, а подією, за настання якої він виконувався, був запуск програми (рис. 1.6, *б*).



а *б*

Рис. 1.6. Схема роботи програми: *а* — подійно-орієнтованої; *б* — не подійно-орієнтованої

Такі програми були призначені для розв'язання однієї задачі або вузького кола подібних задач і надавали користувачеві доволі обмежені можливості для вибору об'єкта, який він бажав опрацьовувати, та дій із цим об'єктом. Образно кажучи, роботу з програмою, яка не є подійно-орієнтованою та об'єктно-орієнтованою, можна порівняти з мандрівкою від пункту А до пункту Б, коли лише зрідка можна трохи відхилитися від маршруту, а роботу з сучасною програмою — з шаховою партією, коли є поле дій і можна на власний розсуд вибирати, якою фігурою і куди походити.

Для допитливих. Подумайте над таким запитанням: коли виконується подійно-орієнтована програма, що вона робить більшу частину часу? Відповідь може видатися несподіваною: програма «простоює», очікуючи на настання чергової події. На перший погляд, це свідчить про неефективність роботи, проте насправді простоювання є дуже суттєвою перевагою, адже під час простоїв можуть виконуватися інші програми. Тому принцип подійної орієнтованості програм тісно пов'язаний з мультизадачністю операційних систем (згадайте матеріал 9 класу).

Висновки

- Програма — це складова інформаційної системи, що виконує обробку даних та може визначати поведінку системи.
- Основні складові програми — логіка, структури даних та інтерфейс. Логіка визначає поведінку програми, структури даних — спосіб зберігання даних, а інтерфейс є засобом взаємодії програми з користувачем та іншими програмами.
- Головні особливості сучасних прикладних програм — це зберігання даних у вигляді об'єктів та виконання дій у результаті настання тих чи інших подій. Ці властивості програм називають об'єктною та подійною орієнтованістю.
- Фрагмент програми, що виконується у результаті настання якоїсь події, називають обробником події.
- Алгоритм — це послідовність інструкцій, виконання яких дозволяє розв'язати певну задачу за скінченний час.
- Алгоритм завжди розраховано на певного *виконавця*, для якого інструкції алгоритму мають бути зрозумілими настільки, щоб він міг їх реалізувати автоматично, не застосовуючи розумову діяльність.
- Кожен обробник події виконує певний алгоритм. Цей алгоритм записують мовою програмування. Складати алгоритми та записувати їх — основний професійний обов'язок програмістів.

Питання для роздумів

1. Спробуйте пояснити, чим мова програмування повинна відрізнятися від природної людської мови.

2. Подумайте, які недоліки мають об'єктно-орієнтовані та подійно-орієнтовані програми порівняно з застарілим програмним забезпеченням, що домінувало до кінця 80-х років XX століття.

Завдання для досліджень

1. Запустіть стандартну гру Сапер (**Пуск ▸ Усі програми ▸ Ігри ▸ Сапер**). Визначте, які об'єкти є в програмі та які події з ними пов'язані. Спробуйте словесно описати алгоритми, що їх виконують обробники цих подій.
2. Припустимо, що в документ **Microsoft Word вставлено полотно з векторним зображенням** і що деякі об'єкти на ньому виділені. Визначте, які події можуть бути пов'язані з полотном і об'єктами на ньому, та опишіть алгоритми, виконувані обробниками цих подій.
3. Знайдіть у Вікіпедії статтю про алгоритм. У ній наведено близько двох десятків досить різноманітних означень цього поняття. З'ясуйте, що є спільного в усіх цих означеннях. Накресліть схему подійно-орієнтованої архітектури програми Сапер, подібну до наведеної на рис. 1.5 для програми Калькулятор.