**Лекція 1**

**Вступ**

**План**

1. Програми, дані, моделі, мови

2. Алгоритми та їхні властивості

3. Мови програмування

4. Види діяльності зі створення програми

5. Кроки роботи з програмою

**1. Програми, дані, моделі, мови**

Сучасна людина використовує комп'ютер для розв'язання різноманітних задач – від виконання простих арифметичних дій до моделювання атмосферних явищ і керування польотами в космос. Насправді все, що "вміє" комп'ютер – це *виконувати програми*. Щоб комп'ютер розв'язав потрібну задачу, необхідно спочатку створити відповідну програму, а потім запустити її на виконання.

**Програма** (*program*) – це опис тих дій, які має виконати комп'ютер, щоб розв'язати деяку задачу. **Програмування** – це діяльність, яка полягає у створенні програм. Мета програмування – забезпечити, щоб замість людини ту чи іншу роботу виконував комп'ютер.

Усі дії комп'ютера пов'язані з **обробкою даних** – числових, символьних, текстових тощо. Згідно з В. М. Глушковим, **дані** (*data*) зображують (позначають) деякий **зміст**, або **інформацію**. Поняття змісту та інформації не мають чіткого означення або тлумачення. Будемо розуміти їх як знання, відомості про щось.

Отже, зображення об'єктів реального світу за допомогою даних є основою будь-якої взаємодії, у тому числі й комп'ютера із зовнішнім світом.

Комп'ютер має розв'язувати задачі для людини. У цих задачах фігурують різноманітні об'єкти реального світу – картинки, фізичні явища, особи, технологічні процеси тощо. Щоб запрограмувати обробку даних, пов'язаних із цими об'єктами, треба спочатку *зобразити об'єкти у вигляді даних*.

Зображення об'єкта, явища або процесу за допомогою даних про нього є його *моделлю*. Узагалі, **модель** – це спрощене зображення об'єкта або явища, що відображає його необхідні суттєві властивості. В обробці даних модель є сукупністю властивостей і співвідношень між ними, що відображають суттєві риси досліджуваного об'єкта, явища або процесу. Модель зазвичай містить опис не лише даних, а також їх обробки, яка відтворює реальні процеси, пов'язані з об'єктом.

Існує безліч різновидів моделей. Один і той самий об'єкт реального світу може мати кілька різних моделей, що виражають властивості, потрібні з різних поглядів на нього.

Власне дані – це деякі позначення, тобто записи, що позначають властивості об'єкта. Однак для правильного розуміння запису (відновлення позначеного ним змісту) необхідно знати, що саме позначають його окремі елементи.

Система позначень деякого змісту називається **мовою**. Мова включає елементарні позначення, правила утворення складніших позначень із простіших і правила, за якими зміст і позначення відповідають одне одному. Правила утворення позначень визначають **синтаксис** мови, а правила, що задають зміст позначень, – **семантику**.

Для спілкування й мислення людина використовує *природні мови* (українську, російську, англійську тощо). Проте в науці й техніці природні мови неефективні або недостатні, тому розроблено також спеціальні *штучні мови*. Їх застосовують насамперед для обміну інформацією між користувачем і/або прикладними процесами. Одним із класів штучних мов є *мови програмування*, призначені для запису програм.

**2. Алгоритми та їхні властивості**

Програму, призначену для виконання комп'ютером, можна розглядати як різновид алгоритму, що є загальнішим поняттям.

**Алгоритм** – це опис послідовності дій, які треба виконати, щоб розв'язати деяку задачу. Позначення дій у алгоритмі називаються **командами** або **інструкціями** (*statement*).

Зазвичай у алгоритмі вказано деякі **вхідні**, **результатні** (**вихідні**) та **проміжні дані**, що не є ні вхідними, ні вихідними.

Послідовність дій, що виконується за алгоритмом, називається **процесом**. Алгоритм зазвичай визначає не один, а деяку множину процесів. Алгоритми мають кілька загальних властивостей: зрозумілість, результативність, однозначність, дискретність, масовість і виконуваність. Розглянемо їх.

**Зрозумілість**. Для виконання алгоритму завжди потрібен *виконавець*. Це може бути людина або деяка технічна система, зокрема комп'ютер. Наприклад, виконувати арифметичні дії, розв'язуючи квадратне рівняння (і не тільки), може людина. Однак вона може перекласти цю роботу на комп'ютер, якщо створить відповідну програму та примусить комп'ютер її виконати.

Так само збирати прилади може спеціальна автоматична лінія, якщо виконує відповідну програму.

Зрозумілість алгоритму полягає в тому, що виконавець може правильно зрозуміти й виконати команди, записані в алгоритмі. Команди завжди записуються за допомогою певної системи позначень, тобто мови. Отже, виконавець повинен розуміти мову запису алгоритму.

**Результативність**. Виконання будь-якого алгоритму має приносити його виконавцю або іншій особі відчутні результати. Наприклад, "корені рівняння визначено", "прилад зібрано" тощо.

**Однозначність**. В алгоритмі не допускаються команди, зміст яких можна сприйняти неоднозначно. Наприклад, якби в алгоритмі розв'язання квадратного рівняння була команда "обчислити *x* або написати, що коренів немає", то виконавець не знав би, що саме йому робити. Окрім того, після виконання кожної команди виконавець повинен точно знати, що робити далі.

**Дискретність**. Дискретність алгоритму полягає в тому, що він задає послідовність дій, чітко відокремлених одна від одної. Отже, дії, задані командою, мають починатися лише після закінчення дій за попередньою командою. Окрім того, виконання кожної команди повинне займати обмежений проміжок часу.

**Масовість**. Конкретні об'єкти, до яких застосовуються дії під час виконання алгоритму, визначають конкретні задачі, що часто називаються **екземплярами задачі**. Наприклад, конкретна трійка чисел 3, 10, 2 відповідає квадратному рівнянню, яке треба розв'язати. Масовість алгоритму полягає в тому, що він застосовний до різних наборів вхідних даних, тобто до різних екземплярів задач. Найчастіше алгоритм описує не один, а деяку *множину процесів*, які відбуваються при розв'язанні всіх можливих екземплярів задачі, хоча існують і алгоритми, що задають тільки один процес.

**Виконуваність і скінченність**. Алгоритм має бути таким, щоб на кожному екземплярі задачі його можна було *виконати до кінця* (й отримати результат). Кожен процес, заданий алгоритмом, має бути *скінченним* і тривати скінченний час. Окрім того, процес *не повинен обриватися* без отримання результату.

Комп'ютерна програма є послідовністю команд, основний зміст яких – *обробка даних*. Центральний процесор зчитує дані й команди програми з оперативної пам'яті та виконує їх. Команди задають зчитування даних із пам'яті, створення нових даних і запис їх у пам'ять. Є також команди, за якими дані надходять до зовнішніх пристроїв або зчитуються з них.

**3. Мови програмування**

Під **мовами програмування** розуміють штучні мови, призначені для запису програм. Першими мовами програмування були машинні мови. Однак для людини вони дуже незручні, тому в 1950-х рр. було розпочато винайдення мов програмування, які за формою мали бути ближчими до мови математичних формул і людських мов, а також вільними від машинних подробиць. Водночас ці мови мали бути такими, щоб записані ними програми можна було *перекласти* у відповідні машинні програми *за допомогою самої машини*. Такі мови невдовзі було розроблено.

Дії комп'ютера в цих мовах мали високий рівень абстракції, тому їх назвали **мовами високого рівня**.

Програмувати мовою високого рівня можна, якщо є **транслятор** – спеціальна програма, що здійснює переклад високорівневої програми на машинну мову. Цей переклад називається **трансляцією** програми.

Протягом кількох десятиліть було створено тисячі мов програмування й трансляторів, і кількість їх постійно зростає.

**4. Види діяльності** **зі створення програми**

Процес створення програми в найзагальніших рисах вимагає кількох видів діяльності:

- аналіз задачі й уточнення її постановки;

- проектування програми;

- розробка програми (кодування);

- перевірка програми (тестування);

- передача замовнику (упровадження).

**Аналіз задачі й уточнення її постановки**. Спочатку за­мовник формулює задачу, яку йому необхідно розв'язати. Задачу зазвичай сформульовано недостатньо точно, навіть може бути, що замовник чітко не уявляє, що саме йому насправді потрібно. Саме тому робота над програмою починається з аналізу зада­чі й уточнення її постановки. Для цього потрібно заглибитися в предметну область замовника й у діалозі з ним уточнити деякі питання. Зазвичай за результатами аналізу задачі будується спрощена модель предметної області, у термінах якої уточню­ється задача. Необхідно також з'ясувати вхідні дані майбутньої програми й результат її роботи над ними. Якщо розв'язання по­ставленої задачі можливе не за всіх вхідних даних, то слід *ви­значити поведінку програми на некоректних вхідних даних.*

**Приклад**. Розглянемо задачу: написати програму ділення двох чисел. Вхідними даними є пара чисел: перше - ділене, дру­ге - дільник. Проте дані з дільником 0 є некоректними. Отже, уточнення постановки полягає в тому, що для коректних вхід­них даних програма має виводити частку від ділення першого числа на друге, а для некоректних - повідомлення про неможли­вість ділення.

В умовах навчального процесу замовником виступає викла­дач. Проте умова задачі все рівно не обов'язково формулюється цілком точно й однозначно (див. попередній приклад).

Якщо не провести аналіз задачі й на його підставі не уточнити її постановку, то можна розв'язати не ту задачу. Аналіз задачі дозволяє визначити, якими саме засобами (математичними та програмними) розв'язувати задачу, а уточнена постановка - що саме й у яких ситуаціях має робити програма.

**Проектування програми**. Між написанням твору та про­грами є певна аналогія. Писати твір починають із плану, який далі розкривають у творі. Так само з програмою: спочатку фор­мулюють її план у вигляді проекту, а потім втілюють цей план у життя - пишуть код (текст) програми.

Під час проектування з'ясовують структуру програми, її скла­дові частини та взаємодію між ними. Тут поступово *уточнюють дії* з розв'язання задачі *та їх опис.* З'ясовують і *уточнюють дані,* потрібні для розв'язання задачі. Дуже часто в задачі можна виді­лити кілька *підзадач* і описати їх розв'язання окремо. Тоді й алго­ритм складається зі зв'язаних і узгоджених між собою частин *(до­поміжних алгоритмів),* що описують розв'язання підзадач.

Одночасно з проектуванням зазвичай відбувається подальше уточнення постановки задачі й моделі предметної області. На практиці замовник може вирішити дещо змінити умову задачі (причому будь-коли!), і тоді доводиться знов уточнювати поста­новку та аналізувати задачу.

Результатом проектування є модель (проект) програми, що далі поступово перетворюється на текст програми. Ця модель може бути записана, наприклад, у вигляді алгоритму з достатньо абстрактними кроками.

У багатьох програм є чотири основні частини: отримати вхідні дані, обробити некоректні вхідні дані, обробити коректні вхідні дані, вивести результати обробки. Для складніших задач під час проектування потрібно визна­чати не лише загальний алгоритм роботи програми, але й необ­хідні структури даних і, можливо, програмні засоби для ство­рення програми та її окремих частин.

**Розробка програми (кодування)**. Коли дії та дані уточ­нено до вигляду, в якому їх можна виразити мовою програму­вання, починають розробку програми. Найчастіше програму за­писують *мовою високого рівня* (інколи окремі її частини - різ­ними мовами).

Розробляючи програму, можна припуститися помилок, що виявляються під час трансляції або виконання програми. Помилки необхідно виявляти й виправляти, тобто налагоджувати програму. **Налагодження програми** полягає в тому, що її багаторазово запускають зі *спеціально підібраними* вхідними даними, які допомагають виявити й відшукати помилки, а потім виправити їх.

Проект програми може залежати від можливостей мови програмування та обладнання. У довготривалих проектах трапляється, що під час розробки змінюються програмні й апаратні засоби, замовник уточнює задачу відповідно до нових можливостей свого обладнання, тому всі процеси починаються наново.

**Перевірка програми (тестування)**. У реальних виробничих процесах програму розробляють програмісти, а перевіряють інші спеціалісти – **тестувальники**. Тестування починається, коли програмісти впевнені, що програма (або деяка її частина) є правильною. Задачею тестування є лише встановлення факту відсутності або наявності помилок. Зазвичай спочатку тестувальники виявляють помилки, і цикл "розробка – тестування" повторюється. В умовах навчання ситуація аналогічна, тільки в ролі програміста виступає студент, а тестувальника – спочатку студент, а потім викладач.

**Передача замовнику (упровадження)**. У найпростішому випадку робота програміста над програмою завершується передачею програми й супроводжувальної документації з її використання замовникові. Для серйозніших програм може знадобитися не просто передати код замовнику, але й установити програму в замовника, зокрема у випадках, коли програма потребує спеціального налаштування параметрів. Інколи потрібно навчити персонал замовника працювати з програмою. Усі ці дії (передавання, установлення, навчання), що дозволяють замовнику користуватися програмою у своїх виробничих процесах, є частиною впровадження програми.

Серйозні програми потребують **супроводження**. Розробник виправляє помилки, виявлені під час експлуатації програми, модернізує її, передає оновлені варіанти користувачам тощо.

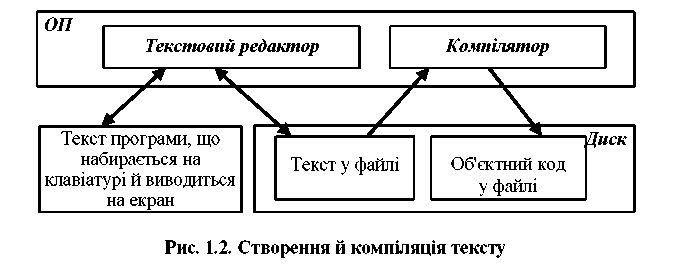
У реальних великих проектах одночасно можуть виконуватися кілька видів робіт. Як тільки постановку задачі більш-менш зрозуміло, можна проектувати програму, обирати програмні засоби для реалізації, писати частини коду й тестувати їх, демонструвати готові частини замовнику, отримувати від нього уточнення й навчати його користуватися програмою. Зазвичай програма та її можливості нарощуються поступово, шляхом багаторазових повернень до аналізу задачі та інших видів роботи.

**5. Кроки роботи з програмою**

Типова послідовність роботи з програмою включає такі кроки: набирання тексту, компіляція, компонування, завантаження й виконання або інтерпретація.

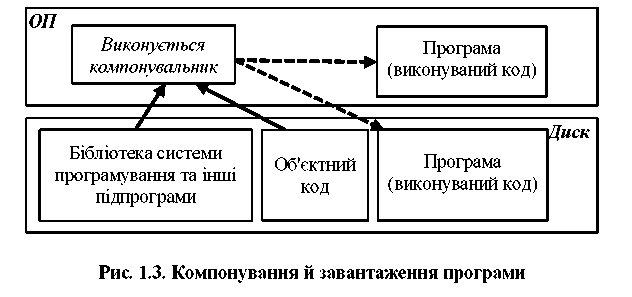
**Набирання тексту**. Текст програми мовою високого рівня (**вхідний текст**) найчастіше набирають за допомогою спеціальної програми (текстового редактора) і зазвичай записують на диск у вигляді вхідного файлу (рис. 1.2). Програма може складатися з кількох файлів – у великих програмах їх можуть бути десятки й сотні.

**Компіляція**. Компілятор – це програма, під час виконання якої читається вхідний текст і створюється його машинний еквівалент – **об'єктний код** (рис.1.2).

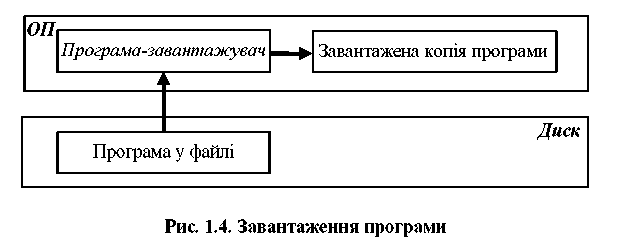


Зазвичай об'єктний код програми містить далеко не всі необхідні команди– програма може складатися з частин; деякі з них є стандартними.

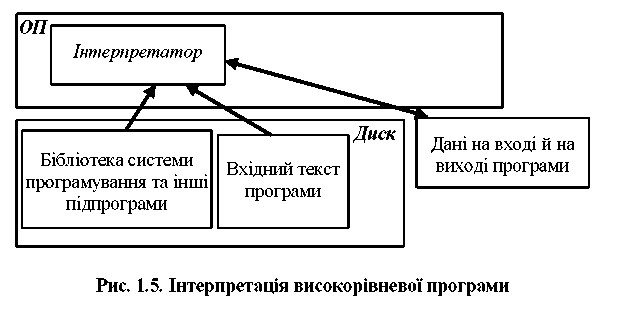
**Компонування**. Об'єктний код обробляє ще одна програма – **компонувальник**. Ця програма "збирає з частин" (компонує) виконуваний код, тобто машинну програму, і записує його або в оперативну пам'ять (**завантажує**), або на диск у вигляді файлу, *готового до виконання* (рис. 1.3). Такий файл можна завантажити пізніше.



**Завантаження й виконання**. Запис машинної програми в оперативну пам'ять називається **завантаженням** (рис. 1.4). Його здійснює спеціальна програма – **завантажувач**, який може входити до складу компонувальника. Якщо завантаження здійснене успішно, то починається процес виконання завантаженої програми.



**Інтерпретація**. Це такий спосіб обробки високорівневої програми, за яким машинна програма не створюється (рис. 1.5). Вхідна високорівнева програма обробляється спеціальною програмою – **інтерпретатором**. При цьому дії вхідної програми, які вона задає, відразу виконуються. Зазвичай інтерпретація вхідної програми відбувається повільніше, ніж виконання відповідної машинної програми.



Інтерпретація програми використовується в такому інструменті, як **налагоджувач**. Він забезпечує інтерпретацію вхідної програми невеликими порціями (кроками) і дає можливість побачити результати виконання після кожного кроку. Це полегшує виявлення помилок у вхідній програмі.

Інколи компіляцію та інтерпретацію узагальнюють словом **трансляція**, називаючи трансляторами всі види програм перетворення вхідних текстів до машинного чи проміжного вигляду.

Описані засоби (текстовий редактор, компілятор, інтерпретатор, компонувальник, завантажувач і налагоджувач) зазвичай утворюють **систему програмування**, або **інтегроване середовище**. Крім них, до складу цієї системи входить **бібліотека стандартних підпрограм**, які можна використовувати під час створення програми.

Що краще розробник програми володіє технологіями, інструментом і бібліотеками, то його робота ефективніша.

**Використана література**

1. Г. Шилдт, Полный справочник по С++, 4-е видання, в-во «Вильямс», 2006
2. Х.М.Дейтел, Как програмировать на С++, 4-е видання, в-во «Бином-Пресс» 2009.
3. Р. Лафоре, Объектно-ориентированное программирование в С++, в-во «Питер», 2004, с 924.