**Лекція 01. Алгоритми і алгоритмізація. Принципи побудови алгоритму вирішення задач.**

Слово «*алгоритм*» походить від algorithm - латинського написання імені аль-Хорезмі, під яким у середньовічній Європі знали видатного математика з Хорезму (місто в сучасному Узбекистані) Мухаммеда бен Мусу, що жив в 783-850 рр.., Який сформулював правила виконання 4 арифметичних дій над багатозначними числами.

***Алгоритм*** — це кінцева послідовність однозначних розпоряджень, виконання яких дозволяє за допомогою кінцевого числа кроків отримати рішення завдання, однозначно визначається вихідними даними.. Програму, призначену для виконання комп‘ютером можна розглядати як різновид алгоритму.

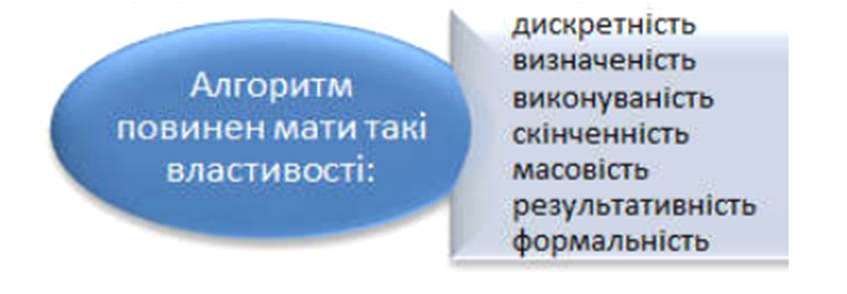
***Програма*** — алгоритм, записаний мовою програмування.

***Проект*** — сукупність файлів, які забезпечують запуск і роботу програмного застосунка.

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.

Поняття алгоритму в інформатиці є фундаментальним, тобто таким, котре не визначається через інші ще більш прості поняття.

**Властивості алгоритму**



**Дискретність** означає, що алгоритм повинен складатися з окремих завершених дій, він задає послідовність дій, чітко відокремлених одна від одної. Дії, задані командою мають починатися лише після закінчення дій за попередньою командою. Виконання кожної команди повинне займати обмежений проміжок часу. Інакше кажучи, алгоритм повинен представляти процес вирішення задачі як послідовність простих кроків. При цьому для виконання кожного кроку алгоритму потрібно деякий кінцевий відрізок часу, тобто перетворення вихідних даних у результат здійснюється в часі дискретно.

**Визначеність / однозначність –** однозначність тлумачення правил виконання дій і порядку їхнього виконання. Алгоритм не повинен містити команди, які можуть сприйматися виконавцем неоднозначно, наприклад, «Порівняти числа А і В». Окрім того, після виконання кожної команди виконавець повинен точно знати, що робити далі.

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

**Масовість** – придатність до використання для великої кількості варіантів вхідних даних. Конкретні об‘єкти, до яких застосовуються дії під час виконання алгоритму, визначають конкретні задачі, що часто називаються екземплярами задачі. 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 можливих екземплярів задачі, хоча існують і алгоритми, що задають тільки один процес. Тобто алгоритм вирішення завдання розробляється не для одного конкретного завдання, а для цілого класу однотипних завдань, що розрізняються лише вихідними даними.

**Результативність** означає, що виконання послідовності вказівок алгоритму повинне приводити до цілком конкретного результату. Наприклад, алгоритм розв’язування рівняння повинен містити перевірку на випадок, коли коренів не існує, і передбачати видання повідомлення про відсутність коренів.

**Формальність / зрозумілість** означає, що будь-який виконавець, здатний сприймати і виконувати вказівки алгоритму (навіть не розуміючи їх змісту), діючи за алгоритмом, може виконати поставлене завдання. Команди завжди записуються за допомогою певної системи позначень, тобто мови. Виконавець повинен розуміти мову запису алгоритму.

**Виконавець алгоритму**

Алгоритм може бути призначений для виконання його людиною або автоматичним пристроєм. Створення алгоритму, нехай навіть самого простого, - процес творчий. Він доступний виключно живим істотам, а довгий час вважалося, що тільки людині. Інша справа - реалізація вже наявного алгоритму. Її можна доручити суб'єкту або об'єкту, який не зобов'язаний вникати в суть справи, а можливо, і не здатний її зрозуміти. Такий суб'єкт або об'єкт прийнято називати ***формальним виконавцем****.*

Кожний алгоритм створюється з розрахунку на конкретного виконавця, тому можна сказати, що алгоритм – це точні розпорядження (указівки, команди, операції, інструкції) виконавцеві здійснити послідовність дій, спрямованих на розв’язання поставленої задачі.

Під **виконавцем** будемо розуміти деяку абстракцію, яка характеризується :

* набором дій (операцій), які виконавець здатний виконати;
* набором об’єктів (даних), над якими виконуються дії;
* набором інструкцій (команд), які виконавець розуміє і реалізує;
* мовою спілкування з виконавцем (алгоритмічною мовою).

Алгоритм складається із **команд** – окремих вказівок виконавцеві виконати деякі конкретні дії. Команди алгоритму виконуються одна за одною, і на кожному кроці відомо, яка команда повинна виконуватися. Почергове виконання команд за кінцеве число кроків приводить до розв’язання задачі. Для того щоб виконавець міг розв’язати задачу за заданим алгоритмом, він повинен уміти виконувати кожну з дій, що вказується командами алгоритму.

Кожен алгоритм створюється з розрахунку на цілком конкретного виконавця. Ті дії, які може здійснювати виконавець, називаються ***допустимими діями***. Сукупність допустимих дій утворює ***систему команд виконавця****.* Алгоритм повинен містити тільки ті дії, які допустимі для даного ***виконавця****.*

Об'єкти, над якими виконавець може вчиняти дії, утворюють так звану ***середу виконавця***.

При побудові алгоритму необхідно дотримуватися наступних п'яти правил.

**Правила побудови алгоритмів**

***Перше правило***- при побудові алгоритму перш за все необхідно задати безліч об'єктів, з якими буде працювати алгоритм. Формалізоване (закодування) представлення цих об'єктів має назву *даних*. Алгоритм приступає до роботи з певним набором початкових даних, які називаються вхідними, і в результаті роботи видає дані, що називаються вихідними. Таким чином, алгоритм перетворює вхідні дані у вихідні.

***Друге правило***- для роботи алгоритму займають пам'ять комп'ютера. У пам'яті розміщуються вхідні дані, з якими алгоритм починає працювати, а також проміжні дані і вихідні дані. Пам'ять є дискретної, тобто що складається з окремих осередків. Пойменована комірка пам'яті має назву змінної. У теорії алгоритмів розміри пам'яті не обмежуються, тобто вважається, що ми можемо надати алгоритму будь-який необхідний для роботи обсяг пам'яті.

Практична робота з алгоритмами (програмування) починається з реалізації перших двох правил. У мовах програмування розподіл пам'яті здійснюється декларативними операторами (операторами опису змінних).

***Третє правило***- дискретність. Алгоритм будується з окремих кроків (дій, операцій, команд). Безліч кроків, з яких складено алгоритм, звичайно.

***Четверте правило*** - детермінованість. Після кожного кроку необхідно вказувати, який крок виконується таким, або давати команду зупинки.

***П'яте правило*** - збіжність (результативність). Алгоритм повинен завершувати роботу після кінцевого числа кроків . При цьому необхідно зазначити, що вважати результатом роботи алгоритму.

**Алгоритми як логіко-математичні засоби відображають певні компоненти людської діяльності і тенденції, а самі алгоритми в залежності від мети, початкових умов завдання, шляхів її вирішення, визначення дій виконавця підрозділяються в такий спосіб:**

* ***механічні алгоритми***, або, інакше, детерміновані, жорсткі (наприклад, алгоритм роботи машини, двигуна і т.п.). Механічний алгоритм задає певні дії, позначаючи їх в єдиній і достовірної послідовності, забезпечуючи тим самим однозначний потрібний чи потрібний результат, якщо виконуються ті умови процесу, завдання, для яких розроблено алгоритм;
* ***гнучкі алгоритми***:
* ***ймовірні*** (стохастичні) алгоритми дають програму рішення задачі декількома шляхами або способами, що приводять до ймовірного досягнення результату;
* ***евристичні******алгоритми*** (від грец.еврика) - це алгоритми, в яких досягнення кінцевого результату програми дій однозначно не визначено, так само як не позначена вся послідовність дій, не виявлено всі дії виконавця. До евристичним алгоритмам відносять, наприклад, інструкції та приписи. У цих алгоритмах використовуються універсальні логічні процедури і способи прийняття рішень, засновані на аналогії, асоціаціях і минулому досвіді вирішення схожих завдань.
* ***лінійні алгоритми*** - набори команд (вказівок), що виконуються послідовно в часі один за одним;
* ***розгалужуються алгоритми*** - алгоритми, що містять хоча б одна умова, в результаті перевірки якого електронно-обчислювальна машина (далі – ЕОМ) забезпечує перехід на один із двох можливих кроків;
* ***циклічні алгоритми*** - алгоритми, що передбачають багаторазове повторення одного і того ж дії (одних і тих же операцій) над новими вихідними даними. До циклічним алгоритмам зводиться більшість методів обчислень, перебору варіантів;
* ***допоміжні (підлеглі) алгоритми*** (процедури) - алгоритми, раніше розроблені і цілком використовувані при алгоритмізації конкретного завдання. У деяких випадках за наявності однакових послідовностей вказівок (команд) для різних даних з метою скорочення запису також виділяють допоміжні алгоритми.

**Способи записів алгоритмів**

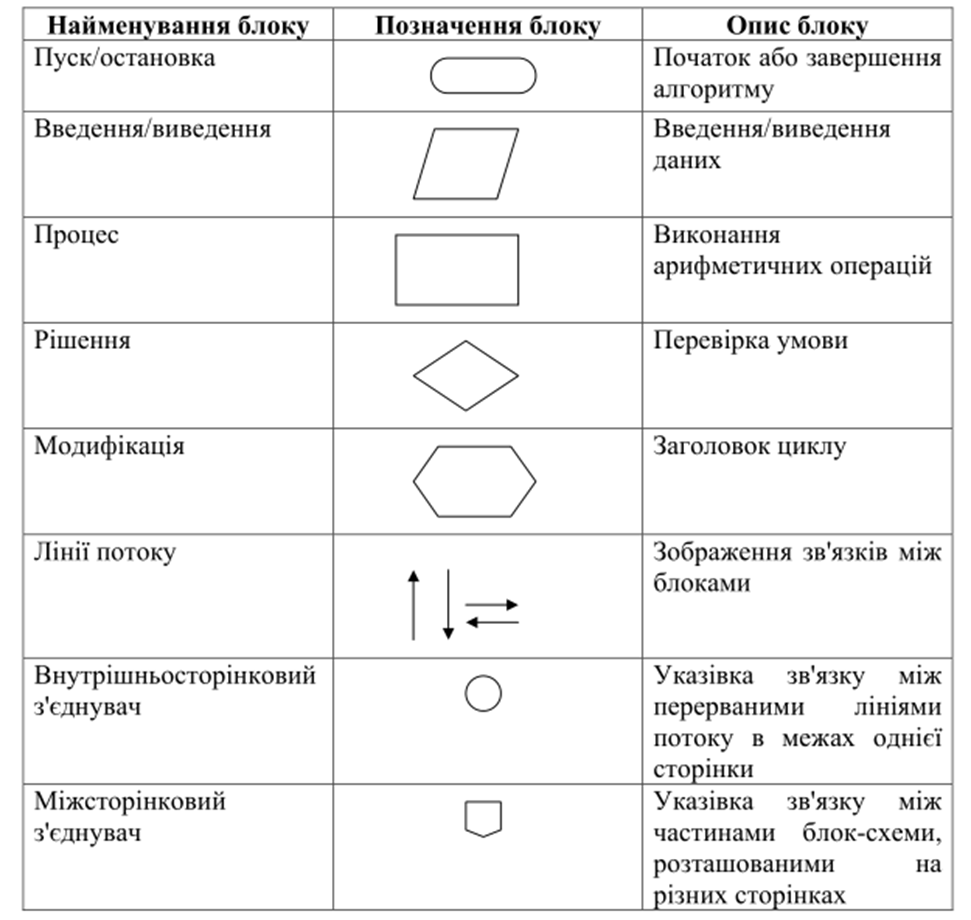
Існують такі основні *способи запису алгоритму*:

1. *Словесно-формульний опис* - це опис природною мовою із використанням математичних формул. Даний спосіб запису алгоритму складається з переліку дій (кроків), кожен з яких має порядковий номер. Алгоритм повинен виконуватися послідовно крок за кроком. Якщо в тексті алгоритму написано «перейти до кроку з номером L», то це означає, що виконання алгоритму продовжиться з зазначеного кроку з номером L. Словесний опис алгоритмів застосовують при вирішенні нескладних завдань, але воно малопридатні для представлення складних алгоритмів через відсутність наочності.

2. *Графічний опис* у вигляді блок-схеми (блок-схема - це набір пов'язаних між собою геометричних фігур). Для позначення кроків рішення у вигляді схеми алгоритму використовуються спеціальні позначення (символи). Для створення блок-схеми використовуються надані нижче позначки (див. табл..1).

3. *Алгоритмічною мовою або мовою програмування*

Таблиця 1. Символи для створення блок-схеми.



Графічна схема (блок-схема) алгоритму  –  це графічне зображення алгоритму у вигляді спеціальних блоків з необхідними словесними поясненнями. Кожний етап алгоритму представляється у вигляді геометричної фігури (блоку), що має певну форму в залежності від характеру операції. Блоки на схемі з’єднуються стрілками (лініями зв’язку), які визначають послідовність виконання операцій та утворюють логічну структуру алгоритму.

Важливою особливістю базових структур алгоритмів є те, що вони мають один вхід і один вихід, що дозволяє при відносній незалежності конструювати окремі блоки алгоритмів, а потім окремо розроблені структури з’єднувати між собою (вихід однієї базової структури сполучається із входом іншої). Весь алгоритм представляє лінійну послідовність базових структур.

Найчастіше використовуються графічні (блок-схеми) засоби та псевдокод. Мова програмування також є засобом зображення, але вона використовується  зазвичай для кодування розробленого алгоритму.

***Псевдокод***. Комбінованим способом представлення алгоритму є написання псевдокоду. Псевдокодом будемо називати словесно-формульний засіб зображення структур керування й алгоритмів.

Наприклад**, з**апис алгоритму обчислення коренів квадратного рівняння за допомогою псевдокоду буде мати вигляд.

  початок

       введення (А, В, С);

       D := В2 - 4 А\*С.

        якщо D <0

                то вивести ('Дійсних коренів немає')

                інакше

                    початок

                       D: =;

                       вивести (Х1, Х2)

                 кінець

   кінець

**Програма** **-** описалгоритму на якій-небудь мові програмування, тобтоформа подання алгоритму для виконання його машиною. В результаті обробки програма*, записана мовою програмування, перетворюється на* набір машинних команд, який слід виконати комп'ютера для реалізації того чи іншого алгоритму.

Для перетворення програми, написаної на одній з мов високого рівня, в програму, що складається з машинних команд, використовується програма-перекладач, так званий *транслятор* (англ*. translator* - перекладач).

Будь-транслятор вирішує наступні основні завдання:

- аналізує транслюються програму, зокрема визначає, чи містить вона синтаксичні помилки;

- генерує вихідну програму (її часто називають *об'єктною*, або *робочою*) мовою машинних команд (в деяких випадках транслятор генерує вихідну програму на проміжному мовою, наприклад, на мові асемблера);

- розподіляє пам'ять для об'єктної програми (у простому випадку це полягає в призначенні кожного фрагменту програми, змінним, констант, масивів і іншим об'єктам адрес пам'яті).

Перекладачі реалізуються у вигляді компіляторів чи інтерпретаторів. З точки зору виконання роботи компілятор та інтерпретатор істотно розрізняються.

Компілятор (англ. *compiler* - упорядник, збирач) читає всю програму цілком, робить її переклад і створює закінчений варіант програми на машинній мові, який потім і виконується.

Інтерпретатор (англ. *interpreter* - тлумач, усний перекладач) перекладає і виконує програму *рядок за рядком.*

Після того, як програма відкомпілювати, ні сама вихідна програма, ні компілятор більше не потрібні. У той же час програма, що обробляється інтерпретатором, повинна заново переводитися на машинний мова при кожному черговому запуску програми.

*Відкомпілювалися* програми працюють швидше, але інтерпретуються простіше виправляти і змінювати.

**Практичні аспекти застосування алгоритмів**.

Алгоритми є основою комп'ютерних наук і програмування. Реальна продуктивність програмної системи визначається рівно двома факторами:

*(1) обраними для неї алгоритмами і*

*(2) адекватністю і ефективністю різних рівнів реалізації.*

Тому хороше проектування алгоритмів є вирішальним для продуктивності будь-яких програмних систем. Більш того, вивчення алгоритмів забезпечує проникнення в глибинну природу розв'язуваної задачі і вибір методів її вирішення, незалежних від мови програмування, парадигми програмування, архітектури комп'ютера і інших аспектів реалізації.

Важливою частиною обчислювальної науки є вміння вибирати і застосовувати алгоритми, що відповідають конкретним цілям, а також враховувати можливість того, що таких алгоритмів може не існувати. Це вміння грунтується на вивченні алгоритмів, використовуваних для вирішення відомого кола чітко визначених завдань, усвідомленні їх сили і слабкості, а також застосування в конкретних контекстах. Лейтмотивом даної сфери знань є ефективність.

Аналіз алгоритму полягає в тому, щоб передбачити необхідні для його виконання ресурси. Іноді оцінюється потреба в таких ресурсах, як пам'ять, пропускна здатність мережі або необхідне апаратне забезпечення, але частіше за все визначається час обчислення. Шляхом аналізу декількох алгоритмів, призначених для вирішення однієї і тієї ж задачі, можна без напруги вибрати найбільш ефективний. У процесі такого аналізу може також опинитися, що кілька алгоритмів приблизно рівноцінні, а все решта потрібно відкинути.

**Алгоритмізація** - це процес побудови алгоритму розв'язання задачі.

Найбільш загальне поняття алгоритмізації – це процес формування алгоритмів, програм. Передбачається систематичний підхід до складання послідовності, що дозволяє вирішити певну прикладну задачу. Якщо необхідно створити програму для комп'ютера, вирішити за допомогою такого продукту чітко визначену задачу, необхідно попередньо скласти алгоритм цього рішення – цей крок вважається обов'язковим.

Алгоритмізація – це детермінований підхід до вирішення завдання, він виключно значимий для алгоритмів, програм прикладного класу. Результат рішення повинен бути масовим, відповідь повинна бути ефективно розрахованою. Правильно сформований алгоритм – запорука правильного вирішення сформульованого питання.

За словниковими визначеннями, алгоритмізація – це етап роботи над завданням, під час якого формулюють алгоритм, що дозволяє вирішити проблему. Альтернативна думка – область інформатики, присвячена методикам, способам створення алгоритмів. Крім того, алгоритмізація розглядає властивості алгоритмів. Іноді цю науку називають алгоритмікой.

У відповідності з іншими поняттями алгоритмізація – описовий процес, що дає уявлення про черговість дій, виконуваних для вирішення завдання, а суть алгоритму - точний опис заданого процесу і формулювання інструкцій, у відповідності з якими можна його виконати. Створення алгоритму трудомісткий і складний процес, а алгоритмізація – техніка, що дозволяє сформулювати дійсно ефективний, оптимізований комплекс послідовних операцій, що реалізуються за допомогою ЕОМ.

**Процеси і етапи алгоритмізації.**

Алгоритмізація – така описова робота, яка дає уявлення про події, що відбуваються всередині заданих процесів. Описують їх при застосуванні математичних символів. Це дозволяє отримати алгоритм, в якому містяться всі елементарні акти завдання, зв'язки та послідовності, що існують між ними, а також причини і наслідки. Сформовані в ході алгоритмізації алгоритми в загальному випадку розробляються саме для електронно-обчислювальної техніки. Алгоритм і алгоритмізація – два дуже важливих поняття для будь-якого, хто змушений працювати з пошуком шляхів вирішення різних складних завдань. Формування ефективної послідовності дій, яка відображала б що відбуваються в реальності виконання процесу, в більшості випадків передбачає послідовне знаходження відповідей на два питання:

* Які системи інформаційної обробки будуть ефективними в конкретному випадку?
* Які математичні методики функціонування стосуються до великих систем?

Розглядаючи методи оброблення інформації, потрібно спершу створити алгоритм, який би детально описував, як система працює. Потім формується послідовність дій, яка дозволяє визначити оптимальні рішення, а також алгоритмізується управлінський процес. В деяких випадках вимагається створення послідовності для виявлення значень, що характеризують управління. Останнє зазвичай стосується великої системи, в якій можна одночасно проводити не тільки якісні, але і кількісні дослідження. Це дозволяє оцінювати ключові особливості системи – надійність, результативність.

Етапи алгоритмізації передбачають послідовне виділення елементарних актів. Кожен з них має бути такого рівня, щоб вдалося описати його математичними функціями, застосовуючи підходи алгебри логіки. Користь при побудові алгоритму принесуть також теорії кінцевих автоматів, випадкових процесів, масового обслуговування. При цьому виявляються співвідношення, які описують взаємні зв'язки між елементарними актами. На підставі таких даних формулюється система, яка і стає повноцінним алгоритмом, придатним для подальшої роботи. Процедури, операції, включені в опис процесу через алгоритм, найбільш зручно фіксувати, застосовуючи спеціальні мови програмування. Особливо це актуально, якщо процес побудови алгоритму необхідний для подальшого втілення коду на ЕОМ. Створений людиною код потім обробляється транслятором і переводиться в машинний код.

Таким чином, послідовність дій для формування алгоритму включає в себе:

* аналіз вихідних даних;
* виявлення найбільш значущих аспектів;
* формалізацію ключових моментів;
* подання даних символами;
* формування цілісної послідовності операцій.

В процесі розроблення алгоритмів використовують прийоми декомпозиції та інтеграції. Декомпозиція має на меті розкладання складної проблеми (задачі) на простіші, інтеграція – реалізацію складної проблеми (задачі) вищого рівня за допомогою проблем (задач) нижчого рівня. Кроків декомпозиції може бути декілька, тобто проблема (задача) деякого рівня декомпозиції може підлягати подальшій декомпозиції. Цей процес породжує ієрархію проблем (задач, виконавців).

Декомпозиція при розробці програми породжує ієрархічну структуру програми. Ієрархічній структурі програми відповідає ієрархія абстрактних виконавців – ієрархія дій, даних, керування. Для кожного абстрактного виконавця ієрархії будується алгоритм відповідного рівня абстракції. При побудові такого алгоритму треба використовувати структури керування й структури абстрактних даних. При розробці програми для структур керування й структур даних використовуються певні засоби зображення.

При розробці алгоритму (програми) застосовуються різні методи. Найпоширенішими є методи: *зверху-вниз* – складна проблема розкладається поступово на простіші; *знизу-вверх* – виділяються простіші елементи, які  об’єднуються поступово у складніші; *розширення ядра* – спочатку виділяється основна (головна) частина алгоритму (скелет алгоритму), яка поступово доповнюється рештою елементів.

Фактично алгоритмізація – складний процес, сам по собі в деякій мірі визначається алгоритмом. Важлива особливість – чіткість, математичність, логічність підходу і результату.

**Приклади алгоритмізації на практиці.**

Алгоритмізація – це ефективний метод автоматизації широкого спектра завдань, робочих процесів, в яких беруть участь люди. Формування програм, алгоритмів в першу чергу використовується для спрощення обчислювальних завдань, які раніше можна було вирішити тільки вручну. Також алгоритмізація дозволяє створити послідовність дій управління машинами. Алгоритмізація дозволяє ефективно переформулювати вихідний (найчастіше досить хаотичний) обсяг інформації в алгоритмічний вигляд, чіткий, упорядкований і структурований. При цьому виділяють всі об'єкти, які беруть участь в операціях, ідентифікують їх, визначають виконавців і задають алгоритм послідовних дій. Важлива умова – обов'язкова однозначність тлумачення будь-якого етапу. Після А завжди "потрібно", а не «може, В, а може, Д, ви вже вирішите самі, як краще». Це правило – основа алгоритмізації.

Представлені в алгоритмічної формі дані – це дані, що продукуються алгоритмізацією. Для них неможливі багатозначні інтерпретації. Алгоритм інформатики, математики, логіки - це така послідовність, яку виконавець може зрозуміти, маючи перед собою лише цей документ і ніяких сторонніх джерел, умов, пояснень операціями. В алгоритмі завжди вказується порядок дій. Без цієї інформації система не може вважатися повноцінною та застосовується на практиці. Алгоритмізація та мови програмування були розроблені людьми, але не лише для себе. Виконувати готовий результат може і машина, причому не тільки високопродуктивний і складно організований комп'ютер, але й простий автоматизований пристрій. В алгоритмі застосовуються наступний типи послідовності операцій:

* лінійні;
* циклічні;
* розгалуження;
* змішані.

*Лінійна* передбачає наявність чіткої послідовності по кроках: є перша операція, друга і так далі. Відхилення від схеми не допускаються, варіантів коригування не передбачено.

*Розгалуження* – можливість дещо коригувати послідовність. Для цього формулюються умови, які вирішуються в ході попередніх операцій (однієї або декількох), які визначають вибір одного з шляхів продовження послідовності дій.

*Цикл* практично ідентичний розгалуженню, але дозволяє повертатися до операції, вже пройденої в ході виконання алгоритму.

*Змішаний* варіант послідовності алгоритмізованих дій містить фрагменти лінійні, циклічні, розгалуження – усі можливі форми. Якщо програма, алгоритм є складними, можна з упевненістю говорити, що вони належать саме до такої форми, її просто неможливо уникнути. Причому складність – поняття дуже широке. Те, що для звичайної людини здається елементарним завданням, при формулюванні її у вигляді алгоритму може перетворитися тривалу послідовність дій різного плану і характеру. Завдання алгоритміста – враховувати всі можливі стани всіх включених в систему об'єктів.

Фактично з алгоритмізацією ми стикаємося в повсякденному житті, наприклад, технологічні інструкції – це класичний зразок алгоритму. Виконавчі інструкції зазвичай складаються стосовно до різноманітних об'єктів – клапанів, агрегатів, витяжок, двигунів. В інструкції описуються фізичні операції – взяти, підняти, закрити. Коли мова йде про обчислювальній машині, об'єкти в алгоритмі будуть математичні дії, відповідно, такі ж. Алгоритм може бути присвячений формулам, таблицям, які скомпоновані значення, а дії бувають самими різними – від найпростіших обчислень до досить складних для людини матричних табличних операцій. Інструкція зазвичай містить умову, що відповідає правилам логіки. Якщо вдалося досягти необхідного показника – можна продовжувати рух за алгоритмом або завершити його, в іншому випадку доведеться пройти ще один цикл. Також алгоритми в нормі мають «запасний вихід» на випадок позаштатної ситуації. Стосовно до людської повсякденності можна знайти аналог у вигляді «Повідомити керівництву про неполадки».

Деякі вважають, що алгоритмізація – це в першу чергу процес переформатування даних в більш упорядкований вигляд. Спершу досліджується вихідна ситуація, аналізується супроводжує її інформація, документація, особливості, побажання. Одночасно з цим алгоритмізація – це цілком чітка і обмежена за масштабом завдання створення інструкцій. Вона має свої складності й особливості.

**Об'єкт алгоритмізації**

Прийнято говорити про такі об'єкти, які можуть вчиняти дії, а також ті, над якими вони виконуються. Для кожного об'єкта характерні деякі певні стани і можливість переходу між ними. Знання повного набору атрибутики дозволяє створити коректний і точний алгоритм, який буде працювати, не вимагаючи додаткових дій.

Ключова умова – перше, що перевіряється стосовно до об'єкта – його присутність саме в такому стані, що припускає виконання передбачених алгоритмом функцій. У разі якщо об'єкт не пройшов попередню підготовку, він несправний, не підходить, стан стає непрацездатним, отже, дії, визначені алгоритмом, не можуть виконуватися. У повсякденності алгоритми застосовуються до різних реальних об'єктів – персоналу, обладнання. Стан їх повинен бути таким, щоб проведені у відповідності з програмою операції або функції виконувалися б успішно, якісно, без збоїв. Враховувати це важливо при формулюванні інструкцій. Так, якщо мова йде про якесь обладнання, його потрібно попередньо зібрати, почистити, протестувати, тільки після цього ознайомити персонал з правилами використання і почати застосовувати інструкцію з використання. Стосовно до машинного алгоритму ситуація схожа, хіба що в якості об'єкта будуть виступати пристрої, а самі кроки, зазвичай, мають бути більш детальними, щоб апарат зміг правильно їх інтерпретувати і робити. При цьому послідовність повинна бути гранично чіткою, інакше агрегат просто не зможе здогадатися – адже це не людина, що володіє волею, інтуїцією, здатністю міркувати на прикладі вже отриманого досвіду.

Важливе поняття – алгоритмізація навчання. Воно передбачає складання такої послідовності дій, яка допоможе навчити цільовий об'єкт (машину або людини) виконувати задані операції. В якості початкового етапу розглядається стан повної відсутності знань і уявлень про цільовому об'єкті. Алгоритм навчання повинен містити таку послідовність операцій, яка дозволить отримати об'єкту уявлення про процес, корисну інформацію, релевантну далі на практиці. Формулювання складних і ефективних алгоритмів навчання останнім часом стало особливою областю уваги в силу підвищення інтересу до штучного інтелекту і здатності до навчання машин.

Алгоритм навчання починається з розгляду найпростіших завдань. Якщо має бути робота з людьми, даються доручення, які дозволяють освоїти базові поняття і процеси системи. Поступово завдання ускладнюються, і в якийсь момент об'єкти алгоритму навчання можуть не просто з легкістю вирішувати поставлені перед ними завдання, але і вчити інших – особливо актуально це, звичайно, стосовно до людей.

***Для самостійного вивчення*:** Поглибити матеріал лекції за наданою літературою. Вивчення лекційного матеріалу та додаткових джерел. Розгляд запитань і виконання завдань для самостійної роботи, запропонованих на лекції..

***Література***

1. Ковалюк Т. В. Алгоритмізація та програмування: Підручник. — Львів: «Магнолія 2006», 2013. — 400 с., ил.
2. Вступ до програмування мовою С++. Організація обчислень: навч. посіб. / Ю. А. Бєлов, Т. О. Карнаух, Ю. В. Коваль, А. Б. Ставовський. – К. : Видавничо-поліграфічний центр "Київський університет", 2012. – 175 с. с.: іл. ISBN (укр.) . URL: <http://csc.knu.ua/uk/library/books/belov-24.pdf>
3. Зелковиц М., Шоу А., Геннон Дж. Принципы разработки программного обеспечения. — М.: Мир, 1982. — 368 с. URL: [http://computersbooks.net/index.php?id1=4&category=teoriyaprogramirovaniya& author=zelkovic-m&book=1982](http://computersbooks.net/index.php?id1=4&category=teoriyaprogramirovaniya&%20author=zelkovic-m&book=1982)
4. Джейс Либерти Освой самостоятельно С++ за 21 день: 3-е изд. пер. с англ.: Уч. пос. – М.: Издательский дом „Вильямс”, 2001. – 816 с.: ил..

***Контрольні запитання*.**

1. Дайте визначення алгоритму та перелічите його властивості.
2. Як можна визначити виконавця алгоритму та систему його команд?
3. Перелічите правила побудови алгоритмів.
4. Які види алгоритмів ви можете визначити?
5. В який спосіб можна записати алгоритм?
6. Які позначки використовуються при запису блок-схем?
7. Яким чином програма перетворюється в машинні коди?
8. Дайте визначення алгоритмізації.
9. Які процеси та етапи алгоритмізації?
10. В чому полягає декомпозиція?
11. Коли застосовується інтеграція?
12. Наведіть приклади алгоритму та алгоритмізації в повсякденному житті.