

## Поняття алгоритму та правила його складання

Поняття алгоритму так само фундаментально для інформатики, як і поняття інформації. Існує багато різних визначень алгоритму, так як це поняття досить широке і використовується в різних областях науки, техніки і повсякденному житті.

**Алгоритм** – зрозуміла і чітка послідовність дій, що описує процес перетворення об'єкта з початкового стану в кінцеве.

**Виконавцем** алгоритму може бути як людина (кулінарні рецепти, різні інструкції, алгоритми математичних обчислень), так і технічний пристрій. Різні машини (комп'ютери, промислові роботи, сучасна побутова техніка) є формальними виконавцями алгоритмів. Від формального виконавця не потрібно розуміння сутності розв'язуваної задачі, але потрібне точне виконання послідовності команд.

Алгоритм можна записувати різними способами (словесний опис, графічне опис – блок схема, програма на одній з мов програмування і т.д.).

**Програма** – це алгоритм, записаний на мові програмування.

Для створення алгоритму (програми) необхідно знати:

- повний набір вихідних даних завдання (початковий стан об'єкта);
- мету створення алгоритму (кінцевий стан об'єкта);
- систему команд виконавця (тобто набір команд, які виконавець розуміє і може виконати).

Отриманий алгоритм (програма) повинен мати наступний набір властивостей:

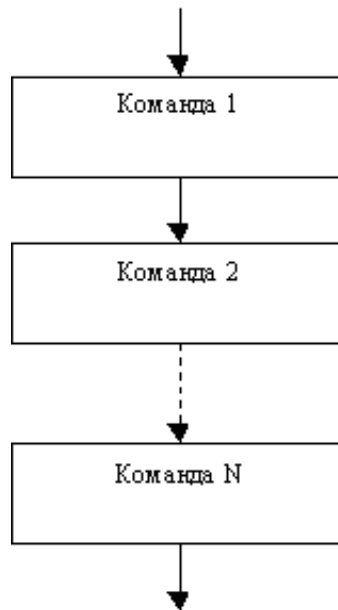
- дискретність (алгоритм розбитий на окремі кроки – команди);
- однозначність (кожна команда визначає єдино можлива дія виконавця);
- зрозумілість (всі команди алгоритму входять в систему команд виконавця);
- результативність (виконавець повинен вирішити задачу за кінцеве число кроків).

Велика частина алгоритмів має також властивість масовості або універсальності (за допомогою одного і того ж алгоритму можна вирішувати безліч однотипних завдань).

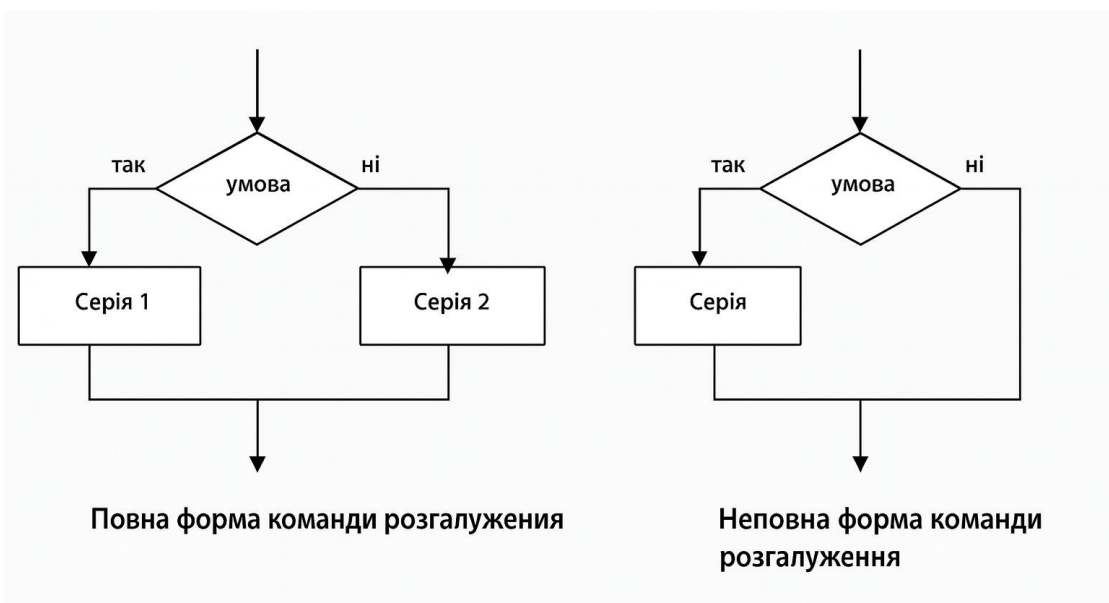
При складанні алгоритму керуються теоремою Дейкстри, яка стверджує, що будь який алгоритм можна представити 3 типами процесів:

- послідовним;
- розгалуженим;
- циклічним

**Лінійним(послідовним)** називається такий обчислювальний процес, при якому всі етапи рішення задачі виконуються в природному порядку проходження записи цих етапів.



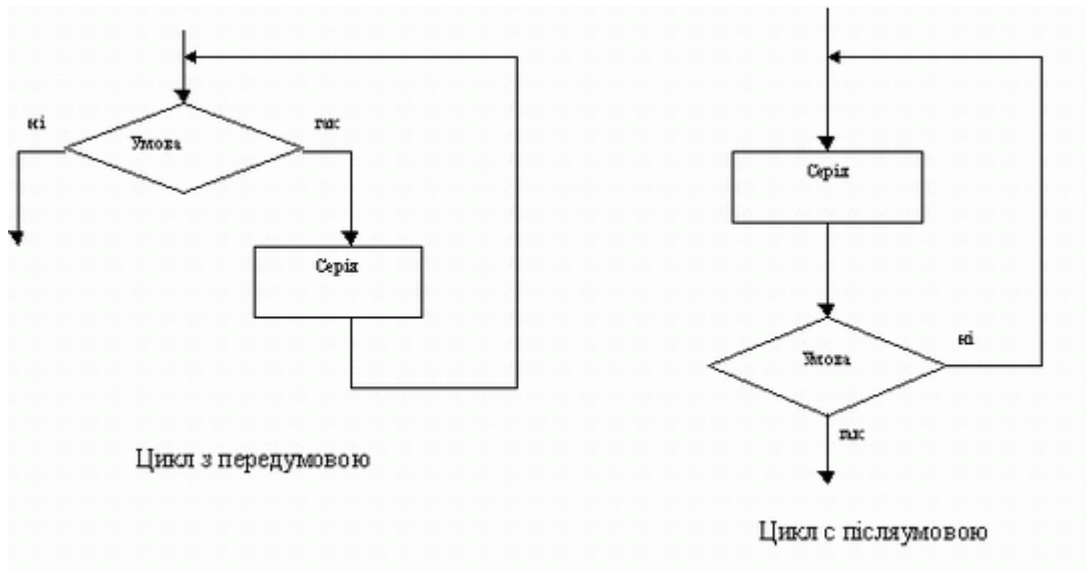
**Розгалуженням** називається такий обчислювальний процес, в якому вибір напрямку обробки інформації залежить від вихідних або проміжних даних (від результатів перевірки виконання будь-якої логічної умови).



**Циклом** називається багаторазово повторювана ділянка обчислень. Обчислювальний процес, що містить один або кілька циклів, називається циклічним.

- За кількістю виконання цикли поділяються на
  - цикли з певним (заздалегідь заданим) числом повторень і
  - цикли з невизначеним числом повторень.

Кількість повторень останніх залежить від дотримання деякої умови, що задає необхідність виконання циклу. При цьому умова може перевірятися на початку циклу – тоді мова йде про цикл з передумовою, або в кінці – тоді це цикл з післяумовою.








## Блок-схеми алгоритмів

Блок схема – це графічне представлення алгоритму за допомогою стандартних позначень. Блок схеми складаються відповідно до державних стандартів і алгоритмів. На схемах алгоритмів їх дії зображуються у вигляді окремих блоків, які з'єднуються між собою лініями зв'язку в порядку виконання дій. На лініях зв'язку можуть ставитися стрілки, причому, якщо напрямок зв'язку зліва направо або зверху вниз, то стрілки не ставляться. Блоки нумеруються. У середині блоку дається інформація про виконувані дії.

## Основні елементи схем алгоритму

Найменування	Позначення	Функція
Початок(кінець)		Елемент відображає вхід у зовнішнє середовище або вихід з нього (найчастіше застосування - початок і кінець програми). Всередині фігури записується відповідна дія.
Процес		Елемент відображає одну або кількох операцій, обробку даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.
Умова		Елемент відображає обробку умови, рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елемента. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елемента. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижньої). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижньої) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.
Функція (процедура)		Елемент відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

<b>ввід/вивід</b>		Елемент відображає перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).
<b>Цикл з параметром</b>		Елемент відображає заголовок циклу з параметром. У ньому через крапку з комою вказуються ім'я змінної (параметра) з початковим значенням, граничне значення параметра (або умова виконання циклу), крок зміни параметра.
<b>Межа циклу</b>		Елемент складається з двох частин - відповідно, початок і кінець циклу - операції, що виконуються всередині циклу, розміщуються між ними. Умови циклу і збільшення записуються всередині символу початку або кінця циклу - в залежності від типу організації циклу. Часто для зображення на блок-схемі циклу замість цього символу використовують символ рішення, вказуючи в ньому умову, а одну з ліній виходу замикають вище в блок-схемі (перед операціями циклу).
<b>З'єднувач</b>		Елемент відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.
<b>Коментар</b>		Елемент використовується для детальнішої інформації про кроки, процесу або групи процесів. Опис поміщається з боку квадратної дужки і охоплюється нею по всій висоті. Пунктирна лінія йде до описуваного елемента, або групи елементів (при цьому група виділяється замкнутою пунктирною лінією). Також символ коментаря слід використовувати в тих випадках, коли обсяг тексту в будь-якому іншому символі (наприклад, символ процесу, символ даних та ін) перевищує його обсяг.

Таблиця 1 - Основні блоки, які використовуються при складанні алгоритмів

### Приклади складання блок-схеми алгоритму

Приклад 1. Обчислити площу трикутника(s), якщо відомі довжина кожної сторони(a, b, c) .

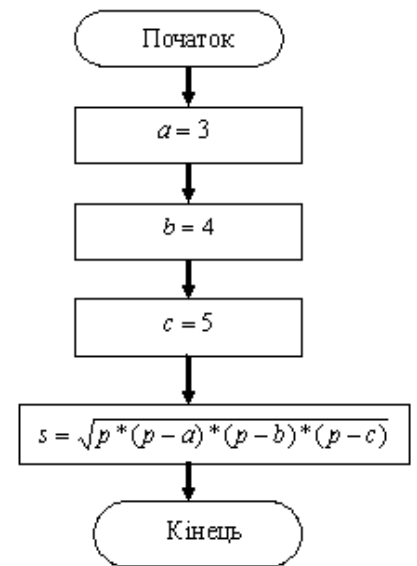
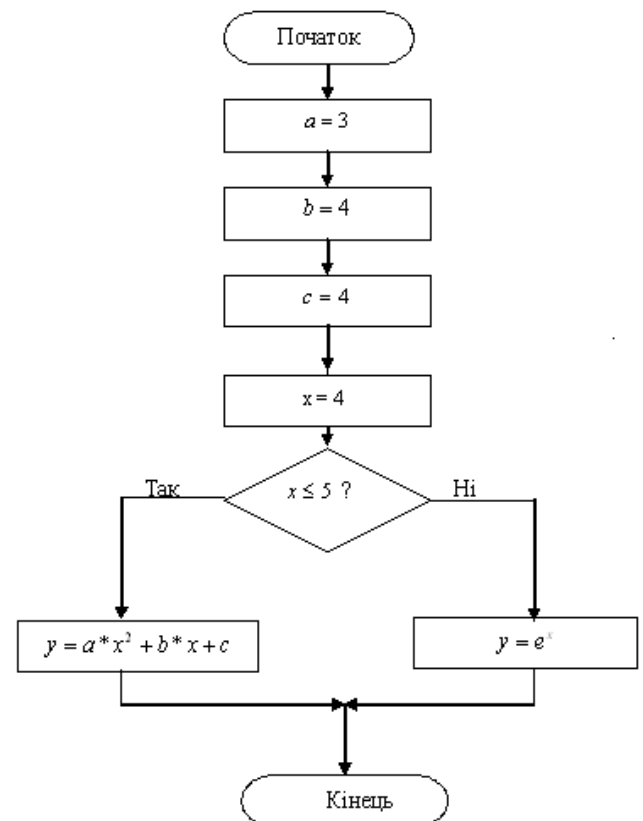


Рисунок 2 - Схема алгоритму прикладу 1

Приклад 2. Обчислити функцію із заданими значеннями параметрів a, b, c і змінної x.

$$y = \begin{cases} ax^2 + bx + c, & x \leq 5, \\ e^x, & x > 5; \end{cases}$$

де значення a, b, c, x задається.



### Застосування циклів з лічильником

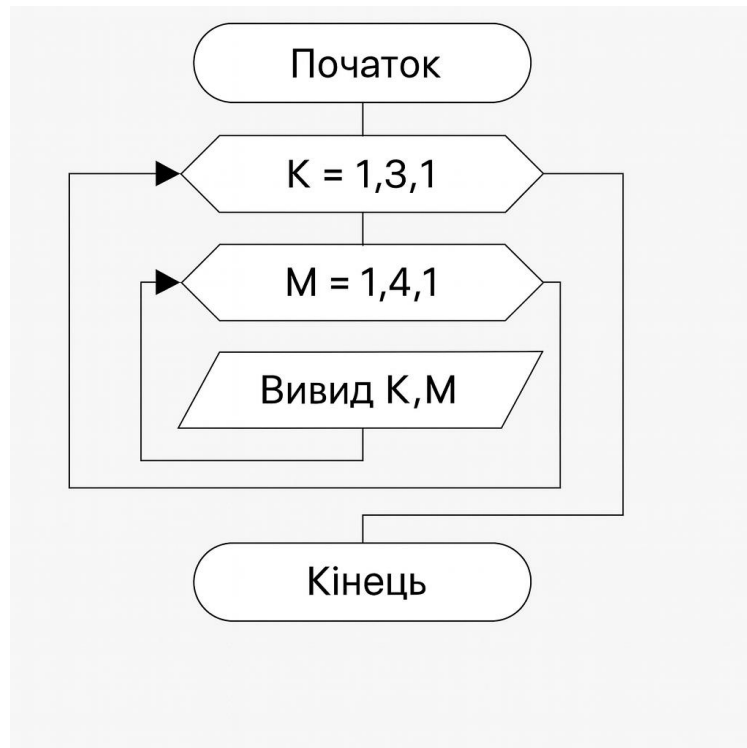
Можна організувати виконання одного циклу всередині іншого. В цьому випадку розрізняють зовнішній і внутрішній цикли - наприклад, коли при кожному значенні лічильника зовнішнього циклу потрібно кілька разів виконати якусь дію (внутрішній цикл). Лічильник зовнішнього циклу змінюється повільніше, ніж лічильник внутрішнього.

#### Цикл в циклі

Розглянемо задачу виведення послідовності пар чисел:

11 12 13 14 21 22 23 24 31 32 33 34

Блок-схема алгоритму розв'язання задачі показана на рис.:



### Що таке IDE?

IDE (Integrated Development Environment) – це інтегроване середовище розробки, яке поєднує в собі всі основні інструменти, необхідні програмісту для створення, редагування, тестування та відлагодження програмного забезпечення. Його головна мета полягає у підвищенні ефективності та зручності роботи розробників шляхом зосередження всіх можливостей у єдиному інтерфейсі.

Якщо раніше програмісти користувалися окремими інструментами для редагування тексту, компіляції коду чи налагодження, то сучасні IDE об'єднують ці можливості в одному продукті. Завдяки цьому скорочується час на розробку, зменшується кількість помилок та спрощується сам процес програмування.

## Основні компоненти IDE:

1. Редактор коду – головний елемент будь-якої IDE. Він відрізняється від звичайних текстових редакторів тим, що підтримує підсвічування синтаксису, автодоповнення команд, перевірку правильності коду в реальному часі. У багатьох редакторах коду реалізовані функції швидкого пошуку та заміни, можливість переходу до оголошення змінних чи функцій, що робить роботу програміста значно швидшою.

2. Компілятор або інтерпретатор – у більшості мов програмування код потрібно перетворити на виконувану програму. Для цього IDE має вбудовані компілятори або інтерпретатори, які здійснюють перетворення вихідного коду у машинні інструкції. Завдяки інтеграції з редактором цей процес максимально автоматизований: достатньо натиснути одну кнопку, щоб побачити результат.

3. Дебагер (налагоджувач) – навіть найдосвідченіші розробники роблять помилки. Дебагер дозволяє знаходити та виправляти їх, крок за кроком виконуючи програму, відстежуючи значення змінних і логіку виконання алгоритмів. Це значно полегшує пошук логічних та синтаксичних помилок.

4. Інструменти тестування – якісний код має бути перевірений. IDE зазвичай підтримує створення автоматизованих тестів і їх виконання. Це дозволяє вчасно виявляти помилки і переконатися, що зміни у програмі не пошкодили існуючий функціонал.

5. Засоби для створення інтерфейсу користувача (GUI) – багато IDE мають інструменти, що дозволяють створювати графічний інтерфейс за допомогою перетягування елементів (drag-and-drop). Це значно спрощує розробку програм з віконним інтерфейсом і робить процес доступним навіть для початківців.

### Приклади популярних IDE:

- **Visual Studio** — потужне середовище від Microsoft, яке підтримує мови C++, C#, Visual Basic та інші. Використовується для створення настільних, мобільних та веб-застосунків.
- **IntelliJ IDEA** — популярна IDE для Java, також підтримує Kotlin, Scala, Groovy та інші мови. Відоме своїм розумним автодоповненням і широким набором плагінів.
- **PyCharm** – середовище, орієнтоване на мову Python. Має вбудовані інструменти для веб-розробки та роботи з базами даних.
- **Eclipse** – універсальна IDE з відкритим кодом, яка часто використовується для розробки на Java, але може бути розширена для інших мов.
- **Xcode** – офіційна IDE для створення програм під macOS та iOS.
- **Android Studio** – середовище розробки мобільних застосунків для Android.

Переваги використання IDE:

1. Продуктивність: всі інструменти зосереджені в одному місці.
2. Зручність: розробник працює в єдиному середовищі, що економить час.
3. Підтримка сучасних технологій: IDE постійно оновлюються, враховуючи останні тренди у світі програмування.
4. Автоматизація процесів: від компіляції до тестування – усе відбувається за декілька кліків.

Висновок:

IDE – це ключовий інструмент сучасного програміста. Вони дозволяють значно підвищити ефективність роботи, зосередитися на логіці програми, а не на технічних деталях компіляції чи налагодження. Завдяки інтегрованим середовищам розробки процес створення програмного забезпечення стає більш швидким, зручним і доступним навіть для початківців.