

Алгоритм — це базове поняття інформатики, тому єдиного визначення цього поняття немає. Проте спроби сформулювати таке визначення здійснювалися не раз і тривають донині.

Слово «алгоритм» походить від імені великого середньоазійського вченого Мухаммеда аль-Хорезмі, який жив у першій половині IX ст. (точні роки його життя невідомі, але вважається, що він народився близько 780 р., а помер близько 850 р.). «Аль-Хорезмі» означає «з Хорезму» (історична область у нинішньому Узбекистані, центром якої було місто Хіва).

Наведемо лише два визначення алгоритму.

«Алгоритм — це набір правил, що визначає послідовність операцій для розв'язування конкретної множини задач і має п'ять важливих рис: скінченність, визначеність, введення, виведення, ефективність» (Д. Е. Кнут, автор відомого багатомника «Мистецтво програмування»).

Алгоритм – это точное и безотказное предписание действий, которые должны быть выполнены.



«Алгоритм — це точний припис, що визначає обчислювальний процес, який іде від варійованих вихідних даних до шуканого результату» (А. Марков).

Існують також інші визначення алгоритму, і ці визначення явно чи неявно задають загальні риси, властиві алгоритмам:

- *дискретність* — процес розв’язування задачі подається як послідовне виконання простих кроків;
- *визначеність* (детермінованість) — кожен наступний крок однозначно обумовлений станом програми (системи) в теперішній момент;
- *зрозумілість* — усі команди алгоритму однозначно «зрозумілі» обчислювальній машині (виконавцеві);
- *масовість* — алгоритм може бути виконаний для різних наборів допустимих вхідних даних;
- *результативність* — виконання алгоритму має привести до очікуваного результату;
- *скінченність* — результат може бути отриманий за обмежене число кроків.

Розглянемо визначення кожної властивості алгоритму.

Дискретність. Процес розбивається на ряд послідовних приписів:

$$1 \rightarrow 2 \rightarrow 3.$$

Визначеність. Кожен виконавець, якому зрозумілий алгоритм, одержить один і той самий результат.

Зрозумілість. Алгоритм подається у вигляді тексту, зрозумілого для виконавця.

Масовість. Алгоритм справджується для різних наборів даних.

Результативність. Після точного виконання всіх запропонованих дій неодмінно має бути отримано результат.

Скінченність. Кількість послідовних дій не є нескінченною, їх можна порахувати:

$$1. \dots \quad 2. \dots \quad 3. \dots \quad N_{\text{останньої}}$$

Розглянемо ці властивості на прикладах алгоритмів у словесній формі.

Дискретність

Вдалий опис алгоритму: «*Махнула красуня правою рукою — виникло озеро пречудове, потім лівою рукою махнула — попливли озером лебеді...*» (Опис подано крок за кроком: спершу озеро, потім лебеді, потім буде ще палац.)

Невдалий опис алгоритму: «*Цап меле, цап меле, коза насипає, а маленьке козенятко на скрипочку грає*». (З опису не ясно, коли починається черговий крок і що відбувається по його завершенні.)

Зрозумілість

Вдалий опис алгоритму: *«Інструкція українською мовою з картинками»*.

Невдалий опис алгоритму: *«Інструкція на дощечках, написаних мовою ронгоронго»* (до речі, досі не розшифрованих).

Визначеність

Вдалий опис алгоритму: *«На дубі — скринька, у скриньці — качка, у качці — яйце, у яйці — голка, а в голці — Коцієва смерть»*.

Невдалий опис алгоритму: *«Піди туди, не знаю куди, принеси те, не знаю що»*.

Масовість

Вдалий опис алгоритму: *«Кожній дочці батько привіз по дорогому подарунку»*.

Невдалий опис алгоритму: *«Принц міг одружитися тільки зі справжньою принцесою»*. (У казці не визначено правило, до яких вихідних даних можна застосувати зазначену дію, лише наприкінці виробляється критерій — з периною й горошиною, але він підходить до єдиної вхідної змінної.)

Результативність

Вдалий опис алгоритму: *«Мишка бігла, хвостиком махнула — яйце впало й розбилось»*.

Невдалий опис алгоритму: *«Баба яйце біла-біла — не розбила, дід бив-бив — не розбив...»*

Скінченність

Вдалий опис алгоритму: *«Мама зварила смачну кашу в горщику»*.

Невдалий опис алгоритму: *«Каша вже заповнила всі вулиці, а горщик усе варив і варив»*.

➡ Основні відомості про професію програміста

Програміст — одна з найпопулярніших професій останніх десятиліть. Кількість обчислювальних машин невинно зростає, у зв'язку з чим виникає потреба в численних фахівцях, які мають обслуговувати й програмувати ці машини, оформлювати одержувані результати. Ця сфера діяльності належить до інформаційних технологій (*information technology*). Цим терміном називають технології керування та оброблення інформації з використанням обчислювальної техніки.

То чим займається програміст? У 1950–60-х рр., коли обчислювальних машин було мало і вони сильно відрізнялись одна від одної, люди, які працювали в обчислювальних центрах, займалися всім — від контролю працездатності техніки та її ремонту до програмування й налагодження завдань, а також оформлення результатів. Відтоді минуло всього 50–60 років, але для інформатики вони стали цілою епохою. Зараз у світі безліч обчислювальних машин. Абсолютна більшість із них — персональні комп'ютери й ноутбуки. Машини схожі одна на одну як зовнішніми пристроями (наприклад клавіатурою), так

і встановленими програмами. Щоб не перелічувати, які програми встановлено на конкретному комп'ютері, всі разом ці програми називають програмним забезпеченням.

Загалом створення нової програми складається з кількох етапів:

- визначення завдання (на цьому етапі вирішується, що робитиме розроблювана програма, за яких вхідних даних і як можна буде нею керувати);
- розробки алгоритму (на цьому етапі програма розбивається на окремі кроки, які будуть виконуватись один за одним; одні з них дуже прості, інші ж вимагають серйозних знань у математиці, графіці, у предметній галузі розроблюваної програми);
- вибір програмних засобів (тут розробник програми вирішує, на якому програмному забезпеченні належить вести програмування завдання, при цьому часто доводиться використовувати декілька програм: мови програмування, спеціальні бібліотеки, графічні, анімаційні й музичні редактори);
- великі проекти рідко робить одна людина, тому важливим етапом є поділ програми на окремі незалежні модулі, що їх можуть писати й випробовувати декілька людей незалежно один від одного, а потім збирати в загальну програму;
- окремим питанням у розробленні є інтерфейс програми — зовнішній вигляд запущеної програми на екрані монітора з кнопками, картинками, елементами введення-виведення, що дозволяють користувачеві (який не завжди є програмістом) застосовувати написану програму;
- важливими частинами сучасних програм є звуковий супровід, графіка, елементи мультиплікації, система допомоги (Help) і супровідна документація;
- коли всі складники програми готові, настає один із найважчих етапів — її остаточне збирання з тих складників і тестування одержаної програми в цілому; іноді тестування буває настільки складним, що програмістські фірми випускають програми для пробного використання майбутніми користувачами (так звані beta-версії програм);
- коли програма готова, з метою її захисту від копіювання програмісти застосовують різні засоби, щоб обмежити доступ до програми (паролі для встановлення на машину), час використання чи кількість запусків програми (trial-версії) або її можливості в разі безкоштовного встановлення (shareware). Іноді в Інтернеті виставляють не саму програму, а її опис із миттєвими знімками екрана (screen-shot) або анімаційними роликами, що показують можливості розробленої програми (trailer).

Для великої програми всі перелічені операції є дуже складним завданням як на одну людину. Тому в наш час програміст спеціалізується тільки на деяких із цих етапів. Традиційно

програмістом називають людину, котра пише програми, програмний код. Однак із появою візуального програмування пов'язувати поняття «робота програміста» з тим, скільки він написав рядків діючих програм, буде не зовсім правильно: великі фрагменти програмного коду вже вписано в компоненти, з яких будується розроблюваний проект. У цьому посібнику як приклади буде наведено хороші, ефективні програми, що, проте, містять мінімальне число рядків програмного коду. І діятимуть вони анітрохи не гірше, ніж якби ви всі дії, що відбуваються на екрані, програмували вручну у вигляді операторів (рядків розробленої програми).



ТЕОРЕТИЧНІ ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1. Наведіть приклади визначення алгоритму.
2. Перелічіть властивості, які має алгоритм. Поясніть їх.
3. У чому полягають особливості професії програміста?
4. Назвіть етапи створення програми.