**ЗМІСТ**

С.

[1 Технологія розробки комп’ютерних програм 1](#_Toc129987700)

[1.1 Архітектура цифрового комп’ютера 1](#_Toc129987701)

[1.2 Подання даних 4](#_Toc129987702)

[1.2.1 Десяткова система числення 7](#_Toc129987703)

[1.2.2 Двійкова система числення 7](#_Toc129987704)

[1.2.3 Шістнадцяткова система числення 8](#_Toc129987705)

[1.2.4 Переведення цілого числа з однієї системи числення до іншої 8](#_Toc129987706)

[1.2.5 Особливості переведення вісімкових і шістнадцяткових чисел 9](#_Toc129987707)

[1.2.6 Внутрішнє машинне представлення цілих та дійсних чисел 9](#_Toc129987708)

[1.2.6.1 Подання цілих від’ємних чисел 11](#_Toc129987709)

[1.2.6.2 Представлення дійсних чисел 12](#_Toc129987710)

[1.2.7 Обчислення машинного порядку 13](#_Toc129987711)

[2 Алгоритм розробки програм 15](#_Toc129987712)

[2.1 Поняття алгоритму. 15](#_Toc129987713)

[2.1.1 Властивості алгоритмів 16](#_Toc129987714)

[2.1.2 Способи написання алгоритмів 16](#_Toc129987715)

[2.2 Машинна програма 19](#_Toc129987716)

[2.3 Мови програмування 20](#_Toc129987717)

[2.4 Особливості мови програмування Python 25](#_Toc129987718)

# 

# ТЕХНОЛОГІЯ РОЗРОБКИ КОМП’ЮТЕРНИХ ПРОГРАМ

## Архітектура цифрового комп’ютера

ЕОМ – електронна обчислювальна машина, що здатна виконувати обчислювальні задачі будь-якої складності за наявності відповідної програми.

Усе забезпечення комп’ютера ділиться на дві частини: апаратну та програмну.

До апаратного забезпечення відносяться пристрої і прилади, що утворюють апаратну конфігурацію. Сучасні комп’ютери й обчислювальні комплекси мають блоково-модульну конструкцію. Основний блок персонального комп’ютера (ПК) – системний блок. Він містить блок електроживлення, кріпильні елементи для материнської (системної) плати, електронних плат і дисководів.

Архітектурою комп’ютера називають склад і взаємозв’язок основних пристроїв комп’ютера. Найбільш поширеною є архітектура зображена на рис 1.1, що відповідає принципам, які були закладені Джоном фон Нейманом (прінстонська архітектура). За цією концепцією щоб комп’ютер був ефективним і універсальним інструментом він має включати такі компоненти:

* арифметико-логічний пристрій;
* пристрій управління;
* запам’ятовуючий пристрій чи пам’ять;
* пристрої уведення-виведення інформації.

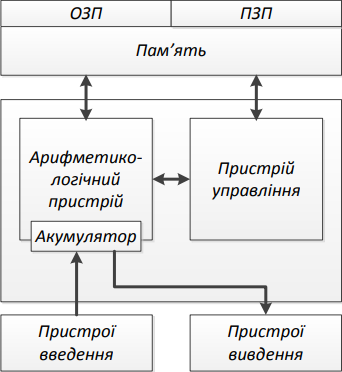


Рисунок 1.1 – Архітектура комп'ютера (фон Неймана)

Арифметико-логічний пристрій виконує арифметичні та логічні перетворення даних, що надходять до нього. Наприклад, зчитування команди з комірки пам’яті A, зчитування даних з комірки пам’яті В, складення 2 чисел, віднімання, ділення, запис результату в комірку С та інші.

Пристрій управління автоматично керує процесом оброблення інформації, посилаючи всім іншим пристроям сигнали про виконання тих чи інших дій.

Сукупність арифметико-логічного пристрою та пристрою управління називають процесором або CPU (Central Processing Unit).

Акумулятор (або регістр) також є частиною процесора і слугує для збереження проміжних результатів обчислень, щоб не звертатися кожного разу до основної пам’яті.

Крім того, процесор містить тактовий генератор, який продукує сигнали для синхронізації роботи всіх мікросхем ("гігагерци", за якими вимірюється швидкодія комп’ютера).

Тобто, коли кажуть, що процесор має частоту 2 GHz – це значить, що він генерує 2 мільярди тактових сигналів в секунду і теоретично виконує 2 мільярди елементарних операцій в секунду.

Але більшість команд є складнішими і можуть потребувати по декілька тактів, тому така характеристика швидкодії є приблизною.

Пам’ять зберігає програми та дані, що передані з інших пристроїв (зокрема, пристроїв уведення), і видає інформацію іншим пристроям комп’ютера, включаючи пристрої виведення.

Там же зберігаються всі проміжні дані, необхідні для виконання програми. Пам’ять складається з великої кількості комірок, кожна з яких має адресу і містить 1 байт (тобто 8 біт інформації). Це дозволяє у будь-який момент часу звернутися до будь-якої комірки і зчитати або записати туди дані. Але дані зберігаються лише поки наявне живлення, а при знеструмленні комп’ютера всі данні стираються. Саме тому ми знаємо її як оперативну пам’ять або RAM (random access memory).

Для того, щоб дані нікуди не зникали після завершення роботи комп’ютера, використовуються додаткові запам’ятовуючі пристрої з нижчою швидкістю доступу до інформації. Найбільш розповсюдженим варіантом є жорсткий диск – на якому зберігаються всі аудіо-, відео- та ін. файли та записана операційна система і всі інші програми. Під час запуску програми вона копіюється у оперативну пам’ять і далі виконується звідти. Всі дані, з якими працює програма, також завантажуються з жорсткого диска та назад за вимогою. Саме тому, при вимкненні комп’ютера слід зберегти всі відкриті документи – в цей час всі зміни існують лише в оперативній пам’яті і зникають з неї разом із струмом.

Жорсткий диск, як і всі інші пристрої, що підключаються до комп’ютера, відноситься до пристроїв вводу-виводу. Кожен з них має власний набір команд для взаємодії з комп’ютером і потребує для роботи спеціальну програму, що називається драйвером. Якщо в операційній системі встановлений відповідний драйвер, вона здатна "спілкуватися" з підключеним пристроєм, що дозволяє його використовувати. Найбільш розповсюджені пристрої мають стандартизовані набори команд і драйвери для них вже включені в операційну систему.

Взаємодію між усіма компонентами забезпечує так звана системна шина. До неї підключається все обладнання, і вона передає сигнали між окремими пристроями. Фізично вона розташовується на материнській платі, навкруги якої і збирається сучасний комп’ютер.

Джон фон Нейман також відзначав, що комп’ютер має працювати з двійковими числами, бути електронним, а не механічним пристроєм, і виконувати операції послідовно одну за одною.

## **Подання даних**

Зазвичай вхідні і вихідні дані подаються у формі, зручній для людини. Числа люди звикли зображати в десятковій системі числення. Для комп’ютера зручніше двійкова система. Це пояснюється тим, що технічно набагато простіше реалізувати пристрої (наприклад, запам’ятовуючий елемент) з двома, а не з десятьма стійкими станами (є електричний струм – немає струму, намагнічений – не намагнічений і т.п.). Можна вважати, що один з двох станів означає одиницю, інший – нуль.

Будь-які дані (числа, символи, графічні та звукові образи) в комп’ютері представляються у вигляді послідовностей з нулів і одиниць. Ці послідовності можна вважати словами в алфавіті {0,1}, так що оброблення даних всередині комп’ютера можна сприймати як перетворення слів з нулів і одиниць за правилами, зафіксованим в мікросхемах процесора.

Під час зчитування документів, текстів програм та інших матеріалів введені букви кодуються відповідними числами, а у разі виведення їх для читання людиною (на монітор, принтер та інше) за кожним числом будується зображення символу. Відповідність між набором символів і їх кодів називають кодуванням символів (symbolic coding).

Зазвичай код символу зберігається в одному байті. Код символу розглядається як число без знаку і, отже, може набувати значень від 0 до 255. Такі кодування називають однобайтовими; вони дозволяють використовувати до 256 різних символів. Тепер дедалі більшого поширення набуло двобайтове кодування Unicode, за якого коди символів можуть набувати значень від 1 до 65535. У цьому кодуванні є номери для майже всіх застосовуваних символів (букв та ієрогліфів різних мов, математичних, декоративних символів тощо.

Система числення – це сукупність прийомів та правил для зображення чисел за допомогою цифрових символів (цифр), що мають визначені кількісні значення (числовий еквівалент).

Залежно від способів зображення чисел цифрами та способу визначення числового еквіваленту системи числення поділяють на дві групи: непозиційні і позиційні.

Непозиційна система числення – це така система, в якій значення символу (значення числового еквіваленту) не залежить від його позиції (розряду) в записі числа, а залежить лише від самого числа.

Прикладом такої системи є римська система числення, у якій цифри позначаються буквами латинського алфавіту.

Число XXX (тридцять у десятковій системі) містить у всіх розрядах один і той самий символ X, що має числовий еквівалент 10 одиниць у десятковій системі числення незалежно від його позиції у запису цього числа.

Кажуть, що цифра не змінює свою вагу залежно від позиції. Для запису проміжних значень застосовується правило: кожне менше значення символу, поставлене праворуч від більшого, додається до більшого значення, а поставлене ліворуч від більшого – віднімається від нього.

Позиційна система числення – це така система, в якій значення символу (числовий еквівалент) залежить від його місця в записі числа.

Будь-яка позиційна система числення характеризується основою.

Основа або базис d натуральної позиційної системи числення – це впорядкована послідовність скінченого набору знаків або символів, які використовуються для зображення числа у цій системі, у якій значення кожного символу залежить від його позиції (розряду) у зображенні числа.

Не існує максимальної основи системи числення, проте існує мінімальна основа системи числення, що дорівнює 2. Отже, якщо за основу можна прийняти будь-яке число (крім одиниці), то можна створити нескінченну множину позиційних систем числення.

Позиційні системи числення поділяють на змішані та однорідні.

Змішана позиційна система числення – це така система, в якій кількість допустимих цифр для різних розрядів (позицій) різна. Вага кожного розряду визначається як добуток ваги попереднього розряду на вагу цього розряду.

Вага розряду числа у позиційній системі числення – це відношення(1.1):

, (1.1)

де – вага розряду числа;

– номер розряду справа наліво;

– це перший розряд ліворуч від коми і його номер дорівнює 0, а значення дорівнює 1.

Отже, кожне число у позиційній системі числення з основою d може бути записане у вигляді дискретної суми степенів основи системи з відповідними коефіцієнтами(1.2):

, (1.2)

де – довільне число у системі числення з основою d;

– коефіцієнти ряду або цифри системи числення;

n – номер розряду цілої частини числа;

m – номер розряду дробової частини числа.

### **Десяткова система числення**

Основу десяткової системи становлять символи 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Отже, = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, основа системи d = 10. Кожна позиція оцінюється значенням або вагою (одиниці, десятки, сотні тощо). Так, у десятковій системі числення у числі перша цифра праворуч означає кількість одиниць, друга цифра – кількість десятків, третя цифра – означає кількість сотень і т.д.

Розгорнута форма запису десяткового числа 7268 можна записати у вигляді 7268 = 7•103+2•102+6•101+8•100.

Якщо десяткове число має дробову частину, то її відокремлюють від цілої частини комою або крапкою. Розгорнута форма запису дробової частини числа буде мати, наприклад, для числа 0,243, вигляд: 0,243 = 2•10–1+4•10–2+3•10–3.

### Двійкова система числення

Двійкова система числення у комп’ютерах є основною, у якій здійснюються арифметичні і логічні перетворення інформації у пристроях комп’ютера. Вона має тільки дві цифри 0 і 1, а всяке двійкове число зображується у вигляді комбінації нулів і одиниць. Будь-яке двійкове число може бути зображене за допомогою формули розкладу десятковим еквівалентом.

Щоб не плутати числа, складені з тих самих цифр, але таких, що належать до різних систем числення рекомендують зазначати основу системи числення у вигляді підрядкового індексу. Наприклад, запис 101,012 означає, що розглядається двійкове число один нуль один кома нуль один, а запис 101,0110 відповідає десятковому числу сто один і одна десята.

### Шістнадцяткова система числення

Шістнадцяткова система числення має основу d = 16 і аі = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A. B. C, D, E, F.

Для запису чисел у системі числення з основою більше ніж 10 арабських цифр вияляється недостатньо і доводиться додатково вводити символи, що однозначно представляють цифри від 10 до 15. У цій системі числення застосовують великі латинські (англійські) символи для позначення цифр від 10 до 15.

Будь-яке число з шістнадцяткової системи числення також може бути зображене десятковим числом за допомогою формули розкладу.

Команди і дані, записані у шістнадцятковій формі, у чотири рази коротше ніж записані у двійковій формі.

### Переведення цілого числа з однієї системи числення до іншої

Для того щоб ціле число перевести з однієї системи числення у іншу, число послідовно ділять на основу нової системи числення, записану у початковій системі числення, до отримання частки, що дорівнює нулю. Число у новій системі числення записується як послідовність залишків від ділення, починаючи з останнього.

Операцію ділення виконують у початковій системі числення, тому її зручно використовувати для переведення десяткових чисел до інших систем числення.

Щоб виконати переведення правильного дробу, його множать на основу нової системи числення, записаної в початковій системі числення. Далі дробові частини від множення послідовно множать на основу нової системи числення. Операцію множення виконують в початковій системі числення. Правильний дріб у новій системі числення записується як послідовність цілих частин від множення, починаючи з першого.

Переведення закінчується, коли проміжний добуток дорівнює 0 у всіх розрядах або досягнута необхідна точність, тобто отримана необхідна кількість розрядів результату після коми. У разі переведення змішаних чисел з однієї системи числення у іншу необхідно окремо перевести його цілу і дробову частини за відповідними правилами, а потім обидва результати об’єднати у одне число у новій системі числення.

### Особливості переведення вісімкових і шістнадцяткових чисел

Будь-яке однорозрядне вісімкове число можна записати у вигляді трирозрядного двійкового, а будь-яке трирозрядне двійкове число можна записати у вигляді однорозрядного вісімкового.

Для переведення вісімкових чисел до двійкової системи числення необхідно кожну вісімкову цифру замінити еквівалентною їй двійковою тріадою (для шістнадцяткових чисел – тетрадою).

Для переведення двійкових чисел до вісімкової системи числення необхідно двійкове число розділити на тріади праворуч і ліворуч від коми (для шістнадцяткових чисел – на тетради). Якщо останні ліворуч і праворуч тріади (тетради) будуть неповні, їх потрібно доповнити нулями. Потім кожну двійкову тріаду (тетраду) замінити одною еквівалентною їй вісімковою (шістнадцятковою) цифрою.

### Внутрішнє машинне представлення цілих та дійсних чисел

Для подання чисел використовують один чи декілька послідовно розміщених байтів. Групи байтів утворюють двійкові слова, що, у свою чергу, можуть бути як фіксованої, так і змінної довжини.

Формати даних фіксованої довжини (півслово, слово і подвійне слово) складаються відповідно з одного, двох і чотирьох послідовно розміщених байтів. Звернення до цих даних виконується за адресою крайнього лівого байта числа, що для слова має бути кратним числу 2, а для подвійного слова – числу 4.

Формат даних змінної довжини складається з групи послідовно розміщених байтів від 1 до 256. Адресація таких даних виконується, як і у форматах фіксованої довжини, за адресою найлівішого байта.

Залежно від характеру інформації у сучасних комп’ютерах застосовують дві форми подання чисел: з фіксованою точкою (комою) і з плаваючою точкою (комою). Так, у форматах даних фіксованої довжини зазвичай подаються двійкові числа, команди і деякі логічні дані, а у форматах даних змінної довжини – десяткові числа, алфавітно-цифрова і деяка логічна інформація.

У разі подання чисел з фіксованою точкою (в першій формі) положення точки фіксується у певному місці відносно розрядів числа. У перших комп’ютерах точка фіксувалася перед старшим розрядом числа, тому подані числа за абсолютною величиною були менші від одиниці. У сучасних комп’ютерах точка фіксується праворуч від наймолодшого розряду і тому можуть подаватися тільки цілі числа. При цьому використовують два варіанти подання цілих чисел: зі знаком і без знаку.

Для числа зі знаком крайній розряд ліворуч потрапляє під знак числа. У цьому розряді записується нуль для додатних чисел і одиниця – для від’ємних.

Нумерація розрядів числа зазвичай ведеться справа наліво.

У комп’ютерах числа з фіксованою точкою мають три основні формати – один байт (півслово), 16-розрядне слово (короткий формат) і 32-розрядне подвійне слово (довгий формат).

В даній комірці можна зберігати цілих чисел. Діапазон допустимих значень залежить також від того, чи будуть в комірці зберігатися числа зі знаком чи без знаку.

Додатковий код цілого додатнього числа збігається з його прямим кодом. Для його отримання необхідно:

* перевести число N в двійкову систему числення;
* отриманий результат доповнити зліва незначущими нулями до k розрядів;
* при необхідності перевести число в стислу шістнадцяткову форму.

#### Подання цілих від’ємних чисел

Для представлення знакових цілих чисел та спрощення арифметичних операцій використовуються три способи:

* прямий код;
* зворотний код;
* додатковий код.

Прямий код двійкового числа містить цифрові розряди, ліворуч від яких записується знаковий розряд. Додавання в прямому коді чисел, що мають однакові знаки, виконується досить просто. Цифрові розряди чисел додаються за правилами арифметики, і сумі присвоюється код знаку доданків. Значно складніше реалізовується в прямому коді операція алгебричного додавання, тобто додавання чисел, що мають різні знаки. У цьому разі доводиться визначати більше за модулем число, вираховувати числа і присвоїти різниці знак більшого за модулем числа.

За допомогою оберненого і додаткового кодів операція віднімання (чи алгебричного додавання) зводиться до арифметичного додавання, спрощується визначення знаку результату операції, а також полегшується вироблення ознак переповнення результату (коли в результаті арифметичних операцій число стає більшим від максимально допустимого для цієї форми значення).

Обернений код від’ємного числа одержується за таким правилом: у знаковий розряд числа записується одиниця, у цифрових розрядах нулі замінюються одиницями, а одиниці – нулями.

Під час виконання операції алгебричного додавання з використанням оберненого чи додаткового коду додатні числа подаються прямим кодом, а від’ємні – оберненим або додатковим кодом. Потім виконується арифметичне підсумовування цих кодів, включаючи знакові розряди, що при цьому розглядаються як старші. У разі використання оберненого коду виникла одиниця перенесення зі знакового розряду циклічно додається до молодшого розряду суми кодів, а у разі використання додаткового коду ця одиниця вилучається.

Алгоритм отримання внутрішнього представлення цілого від’ємного числа N, що зберігається у k-розрядному машинному слові:

* отримати внутрішнє представлення додатнього числа N;
* отримати зворотний код цього числа заміною 0 на 1 і 1 на 0, тобто значення всіх біт інвертувати;
* до отриманого числа додати 1 (отримати доповняльний код);
* при необхідності записати стисле внутрішнє машинне подання.

#### Представлення дійсних чисел

Для подання дійсних чисел у пам’яті комп’ютера був розроблений стандарт IEEE 754 (Institute of Electrical and Electronics Engineers – Інститут інженерів з електротехніки й електроніки) (табл. 1.1). Він використовується багатьма мікропроцесорами і програмними засобами.

Таблиця 1.1 – Подання дійсних чисел у пам’яті комп’ютера за стандартом IEEE 754

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Формат точності | Розмір | Мантиса | Експонента | Діапазон значень | Точність |
| Половинна | 2 Б | 11 | 5 | 10-4..104 | 3 |
| Одинарна | 4 Б | 24 | 8 | 10-4..1038 | 7 |
| Подвійна | 8 Б | 53 | 11 | 10-307..10307 | 16 |

Дійсні числа в пам’яті комп’ютера представлені в форматі з плаваючою десятковою комою (експоненційній формі).

Будь-яке число А може бути представлене в експоненційній формі(1.3):

, (1.3)

де m – мантиса числа;

q – основа системи числення;

N – порядок числа.

Так як варіантів представлення одного і того ж числа в експоненційній формі безліч, то для внутрішнього машинного представлення домовилися представляти числа у нормалізованій або денормалізованій формі.

У денормалізованій формі мантиса повинна відповідати такій умові: вона повинна бути правильним дробом і мати після коми цифру, відмінну від нуля, тобто .

Наприклад, 555,55 – звичайна форма, – денормалізована форма. Це стосується і від’ємних чисел, тому що мантиса в умові взята по модулю.

Нормалізована мантиса містить свій старший біт зліва від точки. У загальному випадку мантиса повинна задовольняти умові 1<=|m|<10. Так як числа представлені в двійковому вигляді, то цей біт завжди дорівнює 1. Іншими словами нормалізована мантиса належить інтервалу . У пам’яті машини цей біт не зберігати, тобто є "прихованим".

### Обчислення машинного порядку

Для зберігання дійсних чисел в пам’яті комп’ютера виділяються наступні розряди:

* знак числа (старший біт);
* машинний порядок числа;
* мантиса.

Щоб спростити операції з порядками, їх зводять до дій над цілими додатними числами використанням зміщеного порядку, що завжди додатний. Отже, щоб не зберігати знак порядку використовується так званий машинний порядок. Машинний порядок зміщений відносно математичного порядку і має тільки додатні значення. Зсув вибирається так, щоб мінімальному математичному значенню порядку відповідав нуль (рис.1.2).

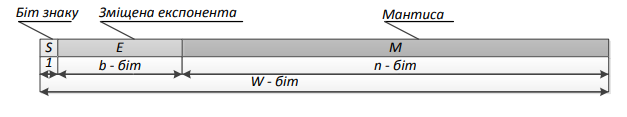


Рисунок 1.2 – Представлення дійсних чисел в стандарті IEEE 754

Для 64-бітного представлення чисел (double) на порядок відводиться 11 біт, для 16-бітного (real) – 5 біт, 32-бітного (single) – 8 біт, для 80-бітного (extended) – 15 біт.

# Алгоритм Розробки програм

## Поняття алгоритму.

В 1936 англійський математик Алан Тьюрінг запропонував математичну модель обчислювальної машини, відому як машина Тюрінга. Крім того, довів, що ця машина здатна виконати будь-які обчислення, і ввів поняття алгоритму – як послідовності дій, необхідних машині для досягнення результату. Сама ж комп’ютерна програма являє собою записаний набір команд, які задають алгоритм у формі, зрозумілій для машини.

Для того, щоб навчити комп’ютер щось робити, потрібно попередньо скласти алгоритм.

Визначень поняття алгоритму дуже багато. Найбільш загальним, простим та неформальним є: алгоритм – це набір інструкцій, що описує, як деяке завдання може бути виконане.

Алгоритм (algorithm) :

* скінченна послідовність точно визначених дій або операцій, спрямованих на досягнення поставленої мети;
* система формальних правил, що визначає зміст і порядок дій над вхідними даними і проміжними результатами, необхідними для отримання кінцевого результату при розв’язуванні задачі;
* будь-яка коректно визначена обчислювальна процедура, на вхід (input) якої подається деяка величина або набір величин, і результатом виконання якої є вихідна (output) величина або набір значень.

При розв’язуванні будь-якої задачі та побудові алгоритму її розв’язку звичайно беруть до уваги наявність деяких вхідних даних і мають уявлення про результат, що необхідно отримати.

### Властивості алгоритмів

Можна навести загальні риси алгоритму:

* дискретність інформації, кожний алгоритм має справу з даними, ці дані представляються у вигляді скінченних слів деякого алфавіту;
* дискретність роботи алгоритму, алгоритм виконується по кроках та при цьому на кожному кроці виконується тільки одна операція;
* детермінованість алгоритму, система величин, які отримуються в кожний (не початковий) момент часу, однозначно визначається системою величини, які були отримані в попередні моменти часу;
* елементарність кроків алгоритму, закон отримання наступної системи величин з попередньої повинен бути простим та локальним;
* виконуваність операцій, в алгоритмі не має бути не виконуваних операцій, наприклад, не можна в програмі призначити значення змінній "нескінченність", така операція була би не виконуваною, кожна операція опрацьовує певну ділянку у слові, яке обробляється;
* скінченність алгоритму, опис алгоритму повинен бути скінченним;
* спрямованість алгоритму, якщо спосіб отримання наступної величини з деякої заданої величини не дає результату, то має бути вказано, що треба вважати результатом алгоритму;
* масовість алгоритму, початкова система величин може обиратись з деякої потенційно нескінченної множини, тобто, можливість виконання алгоритмів для рішення цілого класу конкретних задач, що відповідають загальній постановці задачі.

### Способи написання алгоритмів

Існує чотири способи написання алгоритмів:

* вербальний (словесний);
* алгебраїчний (за допомогою літерно-цифрових позначень виконуваних дій);
* графічний;
* з допомогою алгоритмічних мов програмування.

Словесна форма запису алгоритмів використовується в різних інструкціях, призначених для виконання їх людиною.

Алгебраїчна форма найчастіше використовується у теоретичних дослідженнях фундаментальних властивостей алгоритмів.

Графічна форма відповідно до державних стандартів (ГОСТ) на оформлення документації прийнята як основна для опису алгоритмів.

Алгоритм записаний за допомогою алгоритмічної мови програмування називається програмою. Алгоритм у такій формі може бути введений у ЕОМ і після відповідного оброблення виконаний з метою отримання шуканого результату.

Графічний спосіб подання алгоритмів

Блок-схеми – найбільш зручний спосіб візуального представлення алгоритмів. Для того, щоб не заплутатися в численних подробицях, є сенс складати блок-схему алгоритму в декілька ітерацій: почати з найбільш загального і поступово його уточнювати.

Блок-схема – наочне графічне зображення алгоритму, коли окремі його дії (етапи) зображуються за допомогою різних геометричних фігур (блоків), а зв’язки між етапами указуються за допомогою стрілок, що сполучають ці фігури.

Блок-схеми відображають кроки, які повинні виконуватися комп’ютером, і послідовність їх виконання.

Розміри символів розраховуються до наступних правил:

* менший геометричний розмір символу слід обирати з ряду 10, 15, 20, … мм (тобто a = {10, 15, 20, …} мм;
* співвідношення більшого та меншого розмірів має становити 1.5 (тобто b = 1.5 a).

Існують такі правила графічного запису алгоритмів:

* блоки алгоритмів з’єднуються лініями потоків інформації;
* лінії потоків не повинні перетинатися;
* будь-який алгоритм може мати лише один блок початку і один блок кінця.

Будь-який, навіть найскладніший, алгоритм може бути поданий у вигляді комбінацій кількох елементарних фрагментів, які називають базовими структурами. До них належать послідовне проходження (рис. 2.1), розгалуження “якщо-то-інакше” (рис. 2.2), цикл з передумовою (рис. 2.3).

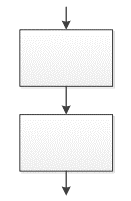


Рисунок 2.1 – Послідовне проходження

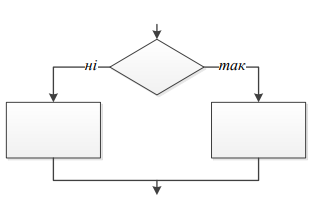


Рисунок 2.2 – Розгалуження “якщо-то-інакше”

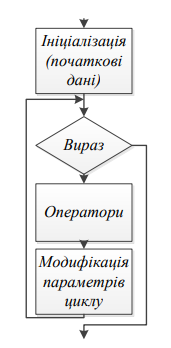


Рисунок 2.3 – Цикл з передумовою

## Машинна програма

Отже, сам по собі комп’ютер не вміє нічого робити. Все на що він здатний це виконувати деякий набір елементарних команд, наприклад зчитати з комірки, додати значення, записати в іншу комірку. Всі можливості комп’ютера забезпечуються програмами.

Величезна кількість комп’ютерних програм складають програмне забезпечення процесу оброблення інформації. Програмне забезпечення можна розділити на три основні частини: системне, інструментальне і прикладне.

Системне програмне забезпечення призначене для:

* управління роботою комп’ютера;
* розподілу його ресурсів;
* підтримки діалогу з користувачем – операційні системи;
* автоматизації процесу розробки та відлагодження програм;
* перекладу мов високого рівня програмування на коди комп’ютера;
* архівація файлів тощо – утиліти;
* забезпечення роботи периферійних пристроїв – драйвери.

Інструментальне забезпечення слугує для розробки різних пакетів програм, що застосовуються в різних областях знання.

В групу інструментальних програм входять:

* транслятори з різних алгоритмічних мов, які переводять текст програми на машинну мову;
* налагоджувачі з допомогою яких знаходять і виправляють помилки, які були допущені при написанні програми;
* інтегровані середовища розробки, які об’єднують вказані вище компоненти в єдину, зручну для розробки систему.

Прикладне програмне забезпечення поділяють на прикладне програмне забезпечення загального призначення, до якого відносяться ті програми, які широко використовуються різними категоріями користувачів (текстові редактори, електронні таблиці, системи управління базами даних, графічні редактори) та прикладне програмне забезпечення спеціального призначення, до якого відносяться програми, які мають специфічне призначення (засоби розробки програм, статистичні обчислення, мережеві застосунки тощо).

У категорію засобів розробки програм відносять всі програми, що використовуються для розробки нових програм. Спектр засобів підготовки програм містить редактори вихідних текстів (зазвичай забезпечують підсвітку (виділення деяких елементів тексту, що мають значення для користувача: дужки, службові слова та ін.) і деяку поверхневу перевірку синтаксису конструкцій що вводяться), транслятори (дозволяють запускати програми), налагоджувачі (призначені для пошуку помилок в програмах) і в деяких випадках ще тести (профайлери), що дозволяють, наприклад, визначити найбільш повільний або вимогливий до ресурсів блок програми.

## Мови програмування

Для подання алгоритму у вигляді, зрозумілому комп’ютеру, служать мови програмування. Спочатку завжди розробляється алгоритм дій, а потім він записується однією з таких мов.

У підсумку виходить текст програми – повний, закінчений і детальний опис алгоритму мовою програмування.

Потім цей текст програми спеціальними службовими додатками, які називаються трансляторами, або переводиться в машинний код, або виконується.

Мови програмування – штучні мови. Від природних вони відрізняються обмеженою кількістю "слів", значення яких зрозуміло транслятору, і дуже строгими правилами запису команд-операторів. Сукупність подібних вимог формує синтаксис мови програмування, а сенс кожної команди та інших конструкцій мови – його семантику.

Отже, програма, з якою працює процесор, являє собою послідовність чисел, яку називають машинним кодом. Такий запис містить лише номери команд процесора, необхідні дані та адреси комірок пам’яті. Наприклад (для зручності двійкові дані найчастіше записуються у шістнадцятковій формі, де 2 символи відповідають 1 байту даних – шістнадцяткова система числення є досить популярною у програмуванні).

Для написання таких програм застосовуються мови асемблера, які дозволяють записувати команди замість числової форми у текстовій (MOV, ADD, IN, OUT) та містять деякі найпростіші засоби для полегшення написання програми. Тим не менше, навіть в такому "прикрашеному" вигляді програма залишається надзвичайно близькою до машинного коду і тому мови асемблера відносять до низькорівневих мов програмування (тобто таких, що близькі до рівня машинного коду).

За рейтингом популярності мов програмування IEEE Spectrum лідером є Python (табл. 2.1).

Таблиця 2.1 – Рейтинг популярності мов програмування

|  |  |  |
| --- | --- | --- |
| Місце у рейтингу | Назва мови | Кількість балів |
| 1 | Python | 100 |
| 2 | Java | 95.4 |
| 3 | C | 94.7 |

Продовження таблиці 2.1

|  |  |  |
| --- | --- | --- |
| Місце у рейтингу | Назва мови | Кількість балів |
| 4 | C++ | 92.4 |
| 5 | JavaScript | 88.1 |
| 6 | C# | 82.4 |
| 7 | R | 81.7 |

Незважаючи на незручність написання, такий код виконується з найвищою швидкістю і може бути максимально оптимізований, так як програміст має доступ буквально до кожного біту у ньому. Тому найчастіше низькорівневі мови застосовуються для програмування мікросхем та окремих дій у прикладних програмах, де швидкодія є критичною.

Асемблер – це програма, яка перетворює код, написаний мовою асемблера, остаточно у машинний код. Але часто і саму мову називають скорочено "асемблером".

Проте, більшість програм пишеться на високорівневих алгоритмічних мовах програмування. Такі мови зазвичай мають складний синтаксис, використовують слова-оператори близькі до людської мови і – що найголовніше – реалізують алгоритмічні структури для простого і зрозумілого запису програми.

З іншого боку, для виконання комп’ютером така програма має бути перетворена на машинний код або хоча б переписана низькорівневою мовою (а далі вже асемблер забезпечить її розуміння машиною). Причому команди високорівневої мови програмування можуть бути досить складними і відповідати кільком або навіть кільком десяткам машинних команд. Цей процес перетворення називається трансляцією, а програми, які його виконують – трансляторами.

Існує два типи трансляторів, що перетворюють вихідний код програм в машинні команди: інтерпретатори та компілятори.

Інтерпретатор зчитує вихідний код програми по одній інструкції і, в найпростішому випадку, одразу намагається їх "перекладати" та виконувати. Це дозволяє програмісту швидше перевіряти виконання програми та знаходити помилки в коді. Крім того, така програма може бути легко перенесена на іншу машину і, якщо там є потрібний інтерпретатор, виконана ним – незалежно від операційної системи та процесора. А різницю між особливостями різних комп’ютерів покриває сам інтерпретатор, який, звичайно, буде трохи відрізнятися.

Логічно, що в такій схемі виконання програми буде займати трохи більше часу – так як при цьому кожного разу відбувається аналіз коду та його перетворення.

Тому для підвищення швидкодії більшість сучасних інтерпретаторів насправді працює за змішаною схемою, спочатку транслюючи вихідний код програми у деяку проміжну форму – так званий байт-код. Він є кодом нижчого рівня і ближчий до асемблеру, але машинно-незалежний – тому виконується не безпосередньо комп’ютером, а деякою віртуальною машиною, яка входить до складу інтерпретатора. Це дозволяє, за відсутності змін в оригінальній програмі, не перечитувати її повністю, а використовувати байт-код як "напівфабрикат" для роботи. Виконання байт-коду все одно повільніше ніж машинного коду, але такий підхід є компромісом, що намагається поєднати переваги інтерпретації та компіляції.

На відміну від інтерпретаторів, компілятор повністю перетворює вихідний код програми в машинний код, який операційна система може виконати самостійно. Це дозволяє виконувати скомпільовані програми навіть на тих комп’ютерах, на яких немає компілятора.

Проте, скомпільована програма прив’язується до операційної системи і набору команд процесора, тому не завжди може бути перенесена і виконана на іншому комп’ютері.

Крім того, такі програми виконуються швидше за рахунок того, що комп’ютеру не доводиться кожен раз перед запуском програми виконувати її розбір і перетворення в зрозумілий для себе вигляд.

Однак, при сучасних потужностях комп’ютерів і обсягах пам’яті різниця в швидкості виконання програм інтерпретаторами і компіляторами вже майже непомітна, але процес розробки і налагодження програм на інтерпретованих мовах набагато простіший.

Програмування – досить складний процес, і цілком природно, коли програміст припускається помилки. Так повелося, що програмні помилки називають "багами" (від англ. bug – жучок). Процес виявлення і усунення помилок в англомовній літературі прийнято позначати терміном debugging.

Процес пошуку помилок в програмі називається тестуванням (testing), процес усунення помилок – налагодженням (debugging).

Уміння налагоджувати програми є дуже важливим навиком для програміста. Процес налагодження вимагає великих інтелектуальних зусиль і концентрації уваги, проте це одне з найцікавіших занять.

Налагодження дуже нагадує роботу дослідника. Вивчаючи результати свого попереднього експерименту, робляться деякі висновки, потім відповідно до них змінюється програма, запускається і знову аналізується отриманий результат.

Якщо отриманий результат не співпаде з очікуваним, то необхідно знову розбиратися в причинах, які призвели до цієї невідповідності. Якщо ж гіпотеза виявиться правильною, то можна передбачити результат модифікацій програми і на крок наблизитися до завершення роботи над нею або, можливо, ще більше повірити в помилку.

Тому для перевірки працездатності програми мало перевірити її один раз – потрібно придумати всі можливі набори вхідних даних, які можуть якось вплинути на стійкість системи. Такі набори вхідних даних називають граничними значеннями.

Іноді процес написання і налагодження програм поділяють не тільки в часі, але і між учасниками команди розробників. Останнім часом все більшої популярності набувають так звані гнучкі методології розробки. У них кодування не відділяється від налагодження: програмісти, які пишуть код, також відповідають і за підготовку тестів і виявлення якомога більшої кількості помилок вже в процесі кодування.

Отже, програмування – це процес поступового доопрацювання і налагодження доти, поки програма не робитиме те, що необхідно. Починати варто з простої програми, яка робить щось просте, а потім можна приступати до нарощування її функціональності, роблячи невеликі модифікації і налагоджуючи додані частини коду.

Існує три типи помилок, які можуть виникнути в програмах:

* синтаксичні помилки;
* помилки виконання;
* семантичні помилки.

Будь-який інтерпретатор зможе виконати програму тільки в тому випадку, якщо програма синтаксично правильна. Відповідно компілятор теж не зможе перетворити програму в машинні інструкції, якщо програма містить синтаксичні помилки. Коли транслятор знаходить помилку (тобто доходить до інструкції, яку не може зрозуміти), він перериває свою роботу і виводить повідомлення про помилку.

Другий тип помилок зазвичай виникає під час виконання програми (їх прийнято називати винятковими ситуаціями або, коротко – винятками, англ. exceptions). Такі помилки мають іншу причину. Якщо в програмі виникає виняток, то це означає, що по ходу виконання сталося щось непередбачене: наприклад, програмою було передано некоректне значення, або програма спробувала розділити якесь значення на нуль, що є неприпустимим з точки зору дискретної математики. Якщо операційна система надсилає запит на негайне завершення програми, то також виникає виняток.

Третій тип помилок – семантичні помилки. Першою ознакою наявності у програмі семантичної помилки є те, що вона виконується успішно, тобто без виняткових ситуацій, але робить не те, що від неї очікується.

У таких випадках проблема полягає в тому, що семантика написаної програми відрізняється від того, що ви мали на увазі. Пошук таких помилок – завдання нетривіальне, тому що доводиться переглядати результати роботи програми і розбиратися, що програма робить насправді.

## **Особливості мови програмування Python**

Python – молода сценарна мова, історія якого почалася в 1990 році, коли співробітник голандського інституту CWI, тоді ще мало кому відомий Гвідо ван Росум приймав участь в проекті створення мови АВС. Ця мова була призначена для заміни мови BASIC в навчанні студентів основних концепцій програмування.

Паралельно з роботою над основним проектом Гвідо ван Росум вдома на своєму Macintosh написав інтерпретатор іншої простої мови але деякі принципи мови АВС все ж були запозичені.

Мова почала швидко розвиватися, оскільки з’явилася велика кількість людей, що були зацікавлені та розумілися в розвитку мов програмування. Спочатку це була досить проста мова, невеликий інтерпретатор, незначна кількість функцій, об’єктно-орієнтоване програмування було відсутнім, але дуже швидко все це з’явилося та до сьогоднішнього дня продовжується її розвиток та виходять нові версії, де кожна наступна має декілька суттєвих відмінностей від попередньої.

Python є однією з десяти найпопулярніших мов програмування. Розглянемо загальну інформацію про цю мову за посиланням на наступний перелік:

1. основи Python:
   1. Python - це інтерпретована мова програмування загального призначення;
   2. Python має простий і зрозумілий синтаксис, що робить його легким у використанні для початківців;
   3. Python підтримує багато типів даних, таких як рядки, числа, списки, кортежі та словники;
2. використання Python:
   1. Python використовується для веб-розробки, наукових обчислень, обробки даних, штучного інтелекту, машинного навчання та багато чого іншого;
   2. Python має велику бібліотеку модулів та фреймворків, які допомагають в розробці різних програм та додатків;
   3. Python можна використовувати на різних платформах, таких як Windows, macOS та Linux;
3. навчання Python:
   1. існує багато онлайн-курсів та підручників, які допомагають вивчити Python;
   2. Python має велику спільноту користувачів, яка надає допомогу та поради новачкам у програмуванні;
   3. Python має інтерактивну оболонку, що дозволяє швидко виконувати код та дізнаватися результати.

Можна знайти велику кількість додатків, написаних на Python, наприклад:

* командний рядок на моніторі або у вікні терміналу;
* призначені для користувача інтерфейси, включаючи мережеві;
* веб-додатки, як клієнтські, так і серверні;
* бекенд-сервери, що підтримують великі популярні сайти;
* хмари (сервери, керовані сторонніми організаціями);
* додатки для мобільних пристроїв;
* додатки для вбудованих пристроїв;
* призначені для роботи з xml/html файлами;
* додатки призначені для роботи з http запитами;
* призначені для роботи із зображеннями, аудіо та відео файлами;
* додатки призначені для роботи з математичними та науковими розрахунками.

Хоча Python є досить швидким для більшості застосунків, проте його швидкості може виявитися не завжди недостатньою. Якщо програма проводить більшу частину часу за обчисленнями ("обмежена швидкодією процесора" (CPU-bound)), то мови С, С++ або Java впораються із завданням набагато краще, ніж Python. Але не завжди. Іноді більш якісний алгоритм (покрокове рішення) для Python перевершує за швидкістю неефективний алгоритм для С. Більш висока швидкість розробки для Python дає більше часу для експериментів над альтернативними рішеннями.