Лекція 1

Алгоритми та основні поняття

Лектор: Матвіїв Дмитро Сергійович

Зміст

Поняття алгоритму

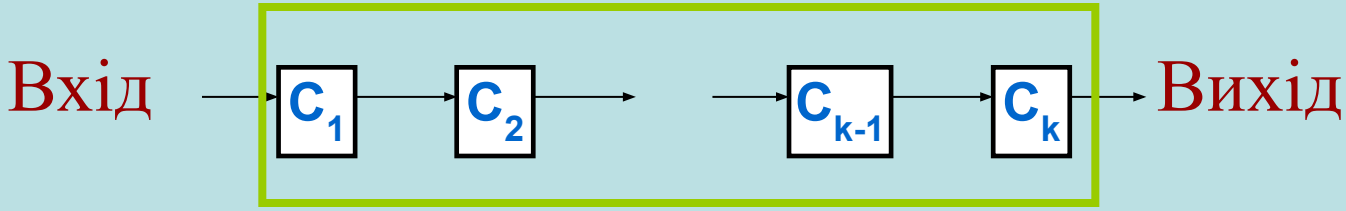
Термін «***алгоритм***» уперше був використаний середньовічними вченими, які перекладали на латинь твори узбецького вченого Аль Хорезмі.

Алгоритмами вони називали правила арифметичних дій над багаторозрядними числами.

Точне математичне визначення алгоритму і вивчення цього поняття - предмет спеціальної математичної дисципліни - теорії алгоритмів

Поняття алгоритму

Алгоритм – це скінчена послідовність інструкцій (команд), виконання яких приводить до результату. Кожна команда алгоритму містить точний опис деякої елементарної дії (операції), а також, у явному або неявному вигляді, вказівку на команду, яку необхідно виконати наступною..



Інформацію, вхідну для алгоритму, прийнято називати його ***входом***, а результат виконання – ***виходом***.

Виконавець алгоритму

Виконавець (Інтерпретатор) алгориму – деякий фізичний або абстрактний пристрій, що однозначно розпізнає і точно виконує (інтерпретує) кожну команду алгоритму.

Виконавця характеризують:

- середовище;
- елементарні дії;
- система команд;
- відмова.

Виконавець алгоритму

Кожний алгоритм описується з врахуванням можливостей конкретного виконавця. У кожного виконавця є набір команд, які він може виконувати.

Система команд виконавця – сукупність команд, які можуть бути використані виконавцем

Властивості алгоритмів

- ***Елементарність*.**

  Кожна команда з набору команд Виконавця містить вказівку виконати деяку елементарну (не деталізовану) дію, яку розуміє, однозначно і точно виконує Виконавець.

- ***Визначеність*.**

  Виконання алгоритму суворо визначене. Це означає, що на кожному кроці Виконавець не тільки точно виконує команду, але й однозначно визначає наступну команду. Тому повторне виконання алгоритму для тих же вхідних даних у точності повторює перше його виконання.

Властивості алгоритмів

- ***Масовість***.

  Алгоритми, як правило, описують хід розв'язування не однієї-єдиної задачі, а цілого класу однотипних задач.

- ***Результативність***.

  Виконання будь-якого алгоритму повинно бути закінчене через скінченну кількість кроків (тобто виконання скінченного числа команд) з деяким результатом.

Властивості алгоритмів

- ***Скінченність***

  Програміст має бути упевненим, що складений ним алгоритм завжди завершує роботу

- ***Правильність***

  Програміст має бути упевненим, що складений ним алгоритм працює правильно, тобто видає правильну відповідь

Саме вираження "властивості алгоритму" некоректно. Властивостями володіють об'єктивно існуючі реальності. Можна говорити, наприклад, про властивості якої-небудь речовини. Алгоритм - штучна конструкція, що ми споруджуємо для досягнення своїх цілей. Щоб алгоритм виконав своє призначення, його необхідно будувати за певними правилами. Тому потрібно говорити не про властивості алгоритму, а про правила побудови алгоритму, або про вимоги, пропонованих до алгоритму.

Перше правило - при побудові алгоритму насамперед необхідно задати безліч об'єктів, з якими буде працювати алгоритм. Формалізоване (закодоване) подання цих об'єктів зветься даних. Алгоритм приступає до роботи з деяким набором даних, які називаються вхідними, і в результаті своєї роботи видає дані, які називаються вихідними. Таким чином, алгоритм перетворить вхідні дані у вихідні.

Це правило дозволяє відразу відокремити алгоритми від "методів" й "способів". Поки ми не маємо формалізованих вхідних даних, ми не можемо побудувати алгоритм.

Друге правило - для роботи алгоритму потрібна пам'ять. У пам'яті розміщаються вхідні дані, з якими алгоритм починає працювати, проміжні дані й вихідні дані, які є результатом роботи алгоритму. Пам'ять є дискретної, тобто складається з окремих осередків. Пойменована комірка пам'яті зветься змінної. У теорії алгоритмів розміри пам'яті не обмежуються, тобто вважається, що ми можемо надати алгоритму будь-який необхідний для роботи обсяг пам'яті.

У шкільній "теорії алгоритмів" ці два правила не розглядаються. У той же час практична робота з алгоритмами (з) починається саме з реалізації цих правил. У мовах програмування розподіл пам'яті здійснюється декларативними операторами (з опису змінних). У мові Бейсик не всі змінні описуються, звичайно з тільки масиви. Але однаково при запуску програми транслятор мови аналізує всі ідентифікатори в тексті програми й відводить пам'ять під відповідні змінні.

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

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

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

Отже, алгоритм - невизначуване поняття теорії алгоритмів. Алгоритм кожному певному набору вхідних даних ставить у відповідність деякий набір вихідних даних, тобто обчислює (реалізує) функцію. При розгляді конкретних питань у теорії алгоритмів завжди мається на увазі якась конкретна модель алгоритму.

Будь-яка робота на комп'ютері - це є обробка інформації.

Види алгоритмів як логіко-математичних засобів відбивають зазначені компоненти людської діяльності й тенденції, а самі алгоритми залежно від мети, початкових умов завдання, шляхів її рішення, визначення дій виконавця підрозділяються в такий спосіб:

Механічні алгоритми, або інакше *детерміновані*, тверді (наприклад,алгоритм роботи машини, двигуна й т.п.);

Гнучкі алгоритми, наприклад стохастичні, тобто імовірнісні й евристичні.

Механічний алгоритм задає певні дії, позначаючи їх у єдиній і достовірній послідовності, забезпечуючи тим самим однозначний необхідний або шуканий результат, якщо виконуються ті умови процесу, завдання, для яких розроблений алгоритм.

Імовірнісний (стохастический) алгоритм дає програму рішення завдання декількома шляхами або способами, що приводять до ймовірного досягнення результату.

Евристичний алгоритм (від грецького слова "эврика") - це такий алгоритм, у якому досягнення кінцевого результату програми дій однозначно не визначено, так само як не позначена вся послідовність дій, не виявлені всі дії виконавця. До евристичних алгоритмів відносять, наприклад, інструкції й приписання. У цих алгоритмах використаються універсальні логічні процедури й способи прийняття рішень, засновані на аналогіях, асоціаціях і минулому досвіді рішення схожих завдань.

Лінійний алгоритм - набір команд (вказівок), виконуваних послідовно в часі один за одним. алгоритм, Що Розгалужується, - алгоритм, що містить хоча б одна умова, у результаті перевірки якого ЕОМ забезпечує перехід на один із двох можливих кроків.

Циклічний алгоритм - алгоритм, що передбачає багаторазове повторення того самого дії (тих самих операцій) над новими вихідними даними. До циклічних алгоритмів зводиться більшість методів обчислень, перебору варіантів.

Цикл програми - послідовність команд (серія, тіло циклу), що може виконуватися багаторазово (для нових вихідних даних) до задоволення деякої умови.

Допоміжний (підлеглий) алгоритм (процедура) - алгоритм, раніше розроблений і цілком використовуваний при алгоритмізації конкретного завдання. У деяких випадках при наявності однакових послідовностей вказівок (команд) для різних даних з метою скорочення запису також виділяють допоміжний алгоритм.

На всіх етапах підготовки до алгоритмізації завдання широко використається структурне подання алгоритму.

Структурна (блок-, граф-) схема алгоритму - графічне зображення алгоритму у вигляді схеми зв'язаних між собою за допомогою стрілок (ліній переходу) блоків - графічних символів, кожний з яких відповідає одному кроку алгоритму. Усередині блоку дається опис відповідної дії.

Графічне зображення алгоритму широко використається перед програмуванням завдання внаслідок його наочності, тому що зорове сприйняття звичайно полегшує процес написання програми, її коректування при можливих помилках, осмислювання процесу обробки інформації.

Можна зустріти навіть таке твердження: "Зовні алгоритм являє собою схему - набір прямокутників й інших символів, усередині яких записується, що обчислюється, що вводиться в машину й що видається на печатку й інші засоби відображення інформації ".

Тут форма подання алгоритму змішується із самим алгоритмом.

Принцип програмування "зверху вниз" вимагає, щоб блок-схема поетапно конкретизувалася й кожен блок "розписувався" до елементарних операцій. Але такий підхід можна здійснити при рішенні нескладних завдань. При рішенні скільки-небудь серйозного завдання блок-схема "розповзеться" настільки , що її неможливо буде охопити одним поглядом.На всіх етапах підготовки до алгоритмізації завдання широко використається структурне подання алгоритму.

Блок-схеми алгоритмів зручно використати для пояснення роботи вже готового алгоритму, при цьому як блоки беруться дійсно блоки алгоритму, робота яких не вимагає пояснень. Блок-схема алгоритму повинна служити для спрощення зображення алгоритму, а не для ускладнення.

Тут форма подання алгоритму змішується із самим алгоритмом.

Принцип програмування "зверху вниз" вимагає, щоб блок-схема поетапно конкретизувалася й кожен блок "розписувався" до елементарних операцій. Але такий підхід можна здійснити при рішенні нескладних завдань. При рішенні скільки-небудь серйозного завдання блок-схема "розповзеться" настільки , що її неможливо буде охопити одним поглядом.На всіх етапах підготовки до алгоритмізації завдання широко використається структурне подання алгоритму.

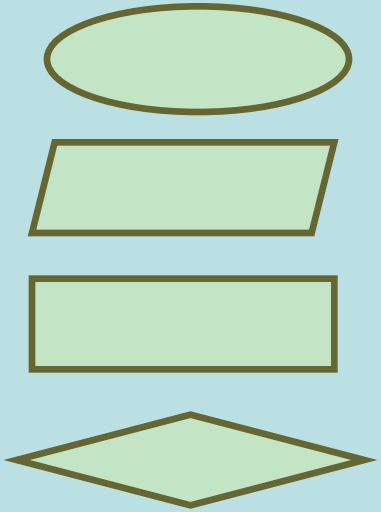
Блок-схеми алгоритмів зручно використати для пояснення роботи вже готового алгоритму, при цьому як блоки беруться дійсно блоки алгоритму, робота яких не вимагає пояснень. Блок-схема алгоритму повинна служити для спрощення зображення алгоритму, а не для ускладнення.

Форми запису алгоритмів

- Блок-схема
- Словесна
- Таблична форма запису
- Алгоритмічна мова

Форми запису алгоритмів

- ***Блок-схема*** – це таке графічне зображення алгоритму, в якому кожна елементарна дія представляється у вигляді спеціального графічного знаку (блоку), який доповнено елементами словесної форми запису.



початок та кінець алгоритму

введення та виведення даних

обчислювальний блок

логічний блок, перевірка умови

Форми запису алгоритмів

- ***Словесна*** - алгоритм записується у вигляді пронумерованих етапів його виконання

Алгоритм додавання двох чисел (a і b)

1. Опитати, чому дорівнює число a.
2. Опитати, чому дорівнює число b.
3. Додати a і b, результат присвоїти c.
4. Повідомити результат c.

Форми запису алгоритмів

- ***Таблична форма запису*** - це запис алгоритму у вигляді таблиці. Таблиці, що використовуються, можуть бути різними

Алгоритм обчислення R=2a+3b

№ дії	дія	величина		результат
		1	2	
1	*	2	a	k
2	*	3	b	u
3	+	k	u	R

Форми запису алгоритмів

✔ ***Алгоритмічна мова*** **-** це запис алгоритму на спеціальній мові (у тому числі і на мові програмування).

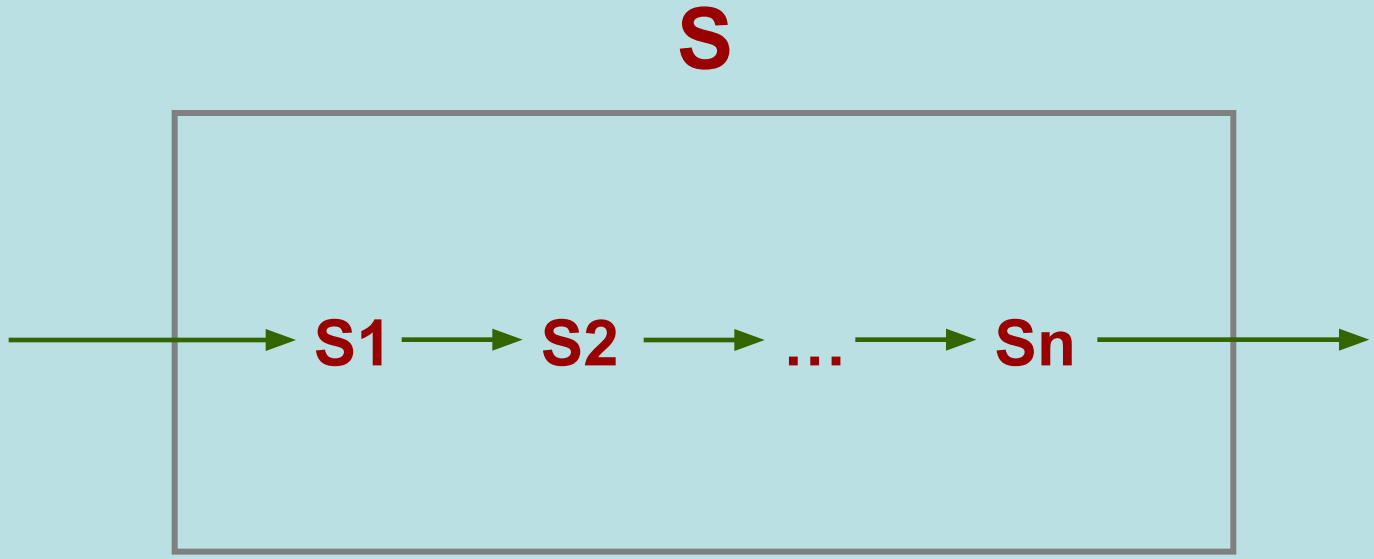
Алгоритм обчислення значення виразу

Y=z-a+2b

алг ОЗВ Y=z-a+2b	**назва алгоритму**
арг z,a,b	**початкові дані (аргументи)**
рез Y	**результат**
поч	**початок алгоритму**
Y:=z-a+2*b	**тіло алгоритму**
Кін	**кінець алгоритму**

Базові структури алгоритмів

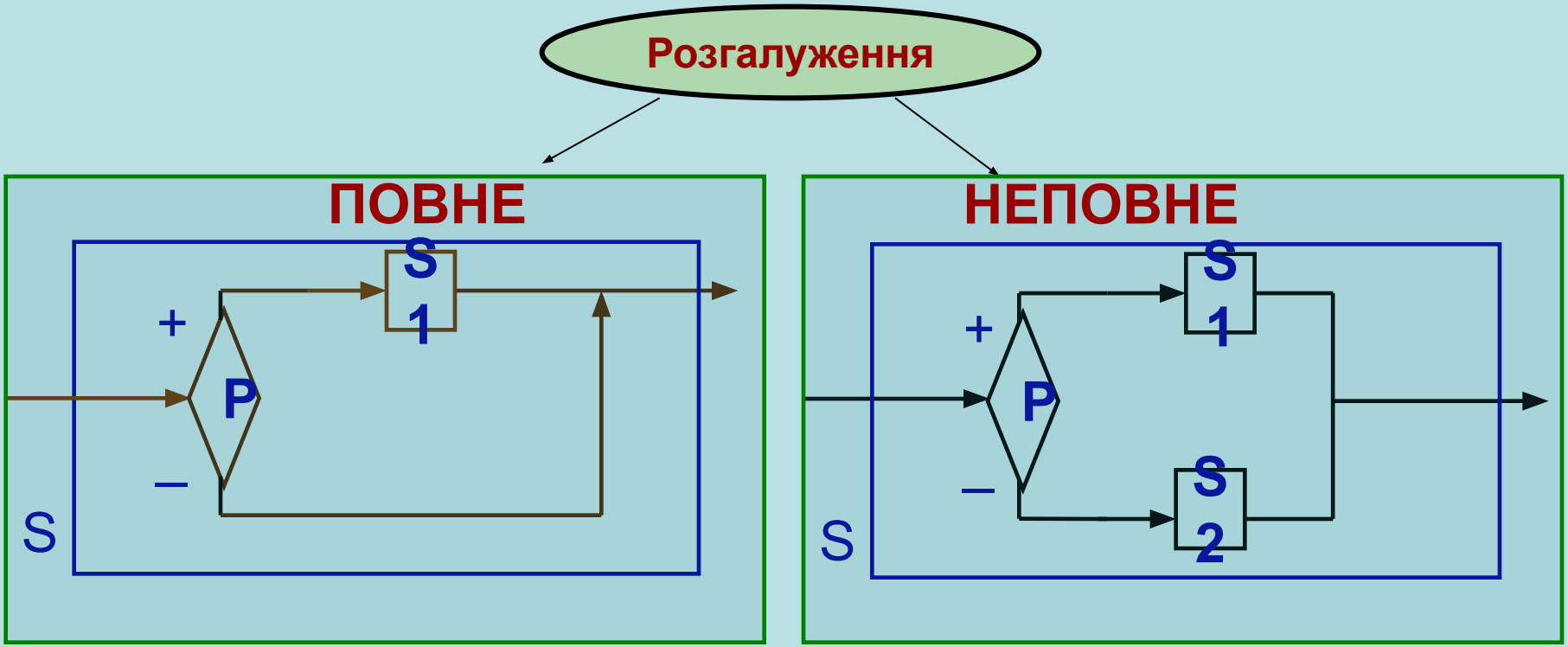
Слідування – вказівка S подається у вигляді послідовності двох (або більше) простих вказівок S1, S2, …,Sn, що виконуються одна за одною



Базові структури алгоритмів

Розгалуження (вибір)

Для виконання вказівки **S** треба спочатку визначити, хибне чи істинне деяке твердження **P**. Якщо твердження **P** істинне, то виконується вказівка **S1** і на цьому вказівка **S** закінчується. Якщо ж твердження хибне, то виконується вказівка **S2** і на цьому виконання вказівки **S** закінчується.

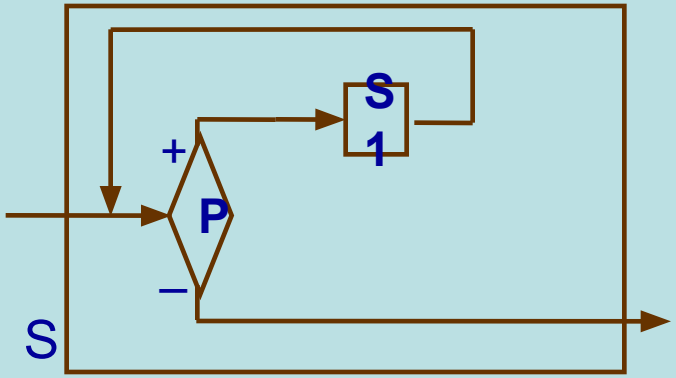


Базові структури алгоритмів

Цикли

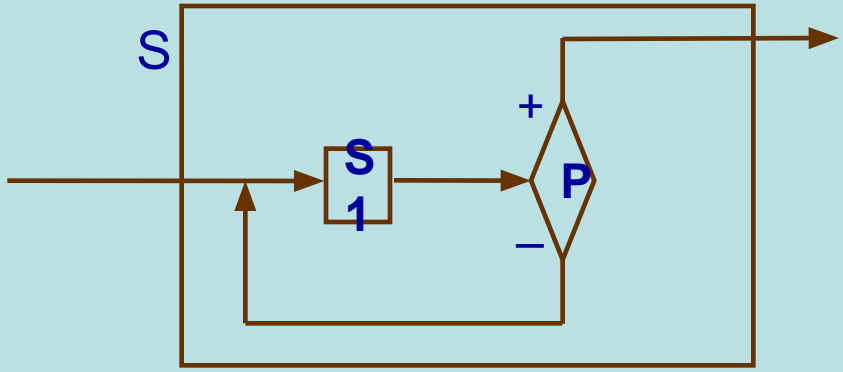
Цикл – ПОКИ

Для виконання вказівки **S** спочатку треба визначити, істинне чи хибне твердження **P**. Якщо **P** істинне, то виконується вказівка **S1** і знову повертаються до визначення істинності твердження **P**. Якщо твердження **P** хибне, то виконання вказівки **S** вважається закінченим



Цикл - ДО

Спочатку виконується вказівка **S1**, а потім визначається істинність твердження **P**. Якщо твердження **P** хибне, то знову виконується вказівка **S1** і визначається істинність твердження **P**. Якщо ж твердження істинне **P**, то виконання вказівки **S** вважається закінченим.



Метод покрокової деталізації

Кожну задачу можна розуміти як окрему вказівку на виконання однієї операції відносно отримання результатів.

Якщо виконавець не може виконати дану операцію, то виникає необхідність розкласти її на деяку сукупність більш простих операцій.

Такий розклад операцій триває доти, доки не отримаємо сукупність операцій, яка містить в собі тільки операції системи команд виконавця.

Даний метод конструювання алгоритмів називається ***методом покрокової деталізації***

Величини

Величиною називають таку характеристику предмета або явища, значення якої можна виміряти або обчислити.

Величини

Ім'я величини ідентифікує цю величину. Програміст використовує імена для позначення величин. Виконавець алгоритму одержує доступ до даної величини за її ім’ям.

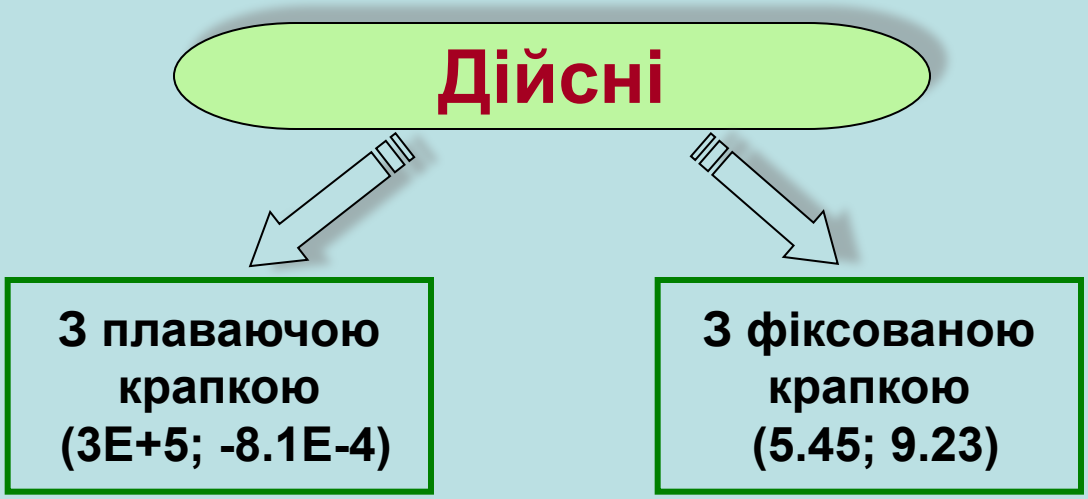
Тип величини визначає набір припустимих операцій над даною величиною, область її визначення і форму запису її значень.

Величини

Типи величин:

- Цілі
- Дійсні
- Літерний тип
- Логічний тип

Величини



Дійсні числа – це десяткові дроби і, в окремому випадку, цілі числа, записані у вигляді десяткового дробу

Величини

Літерний тип

Уявлення про ***рядкові величини*** сформувалося в процесі становлення інформатики як науки. Значенням рядкової величини є слово, тобто ланцюжок букв з деякого алфавіту.

Приклади літерних даних: ‘Інформатика’, ‘Algorithm’, ‘5 травня 2004 року’, ‘триста двадцять дев'ять’. Необхідність розглядати слова як дані виникає в алгоритмах обробки текстової інформації.

Логічний тип

Логічна величина може приймати два логічних значення – Істина або Хибність.

Результат виконання операцій порівняння (=,<,>,>=,<=,<>) над даними одного типу належить до логічного типу.

Величини

Цілі

Арифметичні операції:

a + b – операція додавання
a - b – операція віднімання,
- b – операція «мінус»;
a*b – операція множення,
a div b – операція обчислення неповної частки,
a mod b – операція обчислення залишку.

Логічні операції:

a > b - операція «більше»
a < b - операція «менше»
a <= b - операція «менше або дорівнює»
a >= b - операція «більше або дорівнює»
a = b - операція «дорівнює»
a <> b - операція «не дорівнює»

Дякую за увагу!