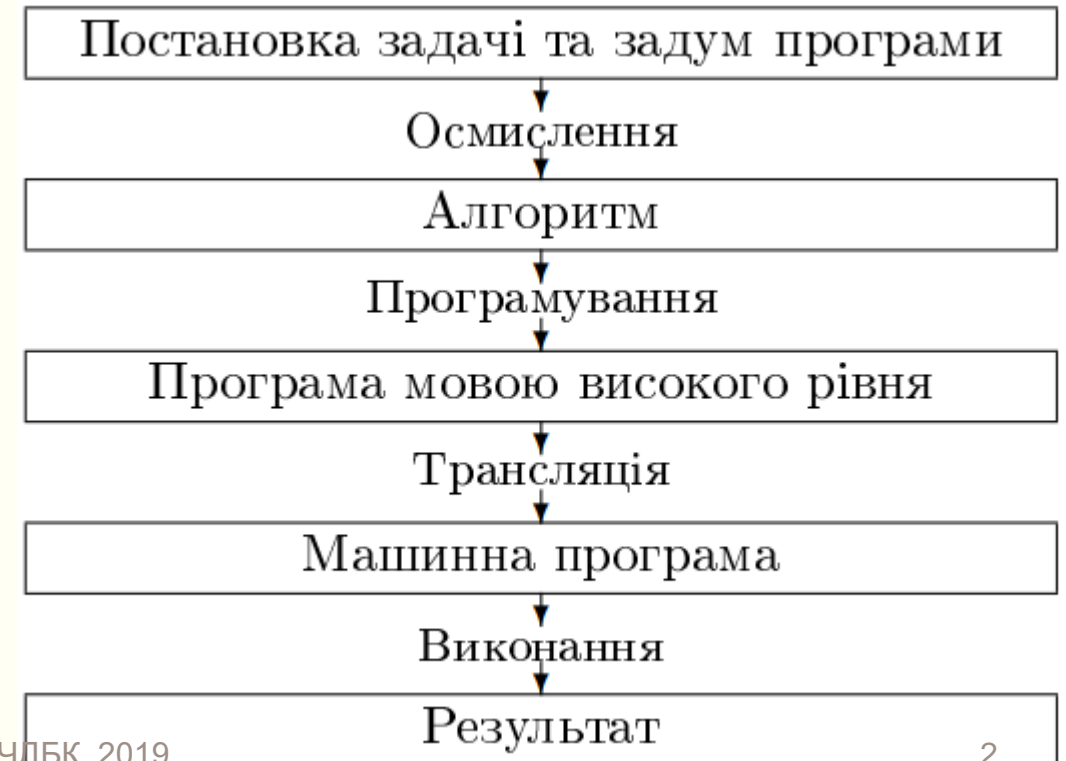
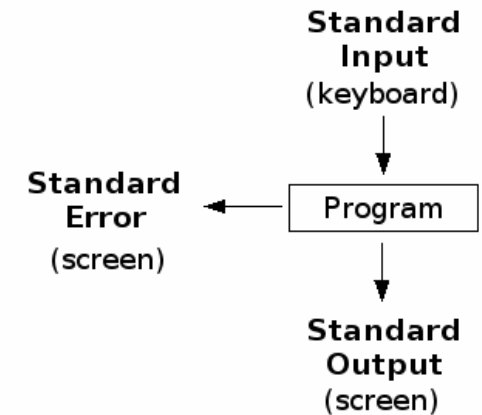




МІСЦЕ АЛГОРИТМІВ У ПРОГРАМУВАННІ

Суть програмування

- Основна задача – навчити комп'ютер розв'язувати певні задачі, які для людини будуть складними або монотонними.
 - *Computer programming (programming, coding)* – це процес написання, тестування, налагодження та підтримки первинного коду комп'ютерних програм.
 - Програма – послідовність інструкцій, які вказують, як проводити обчислення.



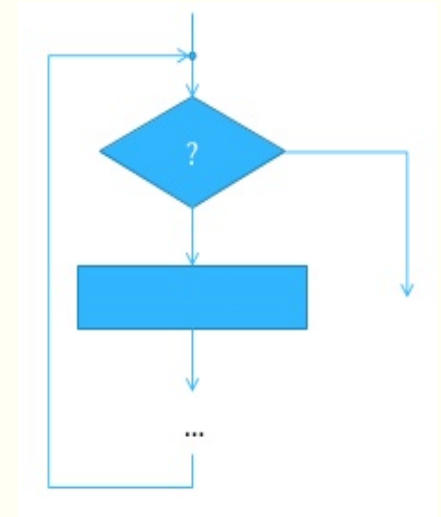
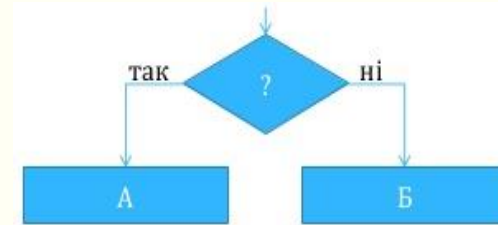
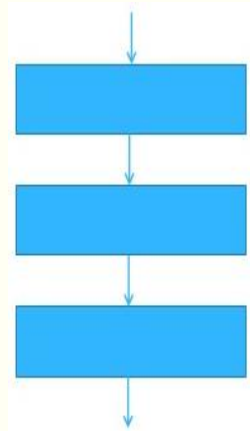
Поняття алгоритму

- **Алгоритм**— *точно визначена послідовність дій, що ведуть від варіацій початкових даних до шуканого результату.*
- Властивості алгоритмів:
 - **Скінченність** — алгоритм повинен завершуватися за скінченну кількість кроків.
 - **Масовість** - застосування конкретного алгоритму для розв'язання цілого класу однотипних задач із різними наборами вхідних даних.
 - **Дискретність** - можливість розчленування процесу виконання алгоритму на окремі кроки.
 - **Елементарність** - крок алгоритму має бути простим, елементарним, можливість виконання якого людиною або машиною не викликає сумнівів.
 - **Детермінованість** - однозначність процесу виконання алгоритму. Результат роботи алгоритму однозначний при однакових наборах вхідних даних.
 - **Результативність** - алгоритм повинен завжди приводити до певного результату.
 - **Формальність** - виконавець алгоритму повинен отримати результат, не вникаючи в його суть.
 - **Ефективність** - алгоритм повинен бути по можливості простим і виконуватися з мінімальними витратами машинного часу.

Яких дій достатньо для представлення алгоритму?



- Основна теорема структурного програмування (Бойм-Якопіні, 1966)
- Програма для розв'язку будь-якої задачі може бути складена з комбінації операторів
 - слідкування, розгалуження, циклу.



Основні способи запису алгоритмів

- **словесна форма алгоритму**, словесно-формульний запис алгоритму
 - алгоритм задається в довільному викладі на природній мові
- **графічний запис алгоритму**
 - алгоритм зображається у вигляді послідовності зв'язаних між собою функціональних блоків, кожен з яких відповідає виконанню одного або декількох дій.
- **псевдокоди**
 - напівформалізовані описи алгоритмів на умовній алгоритмічній мові, що включають як елементи мови програмування, так і фрази природної мови, загальноприйняті математичні позначення і ін.
- **програмний**
 - тексти на мовах програмування

Словесна форма алгоритму: соковитий і легкий у приготуванні шашлик з курки

■ Інгредієнти:

- куряче м'ясо - 1 кг 500 г
- цибуля ріпчаста - 2 шт
- спеції для шашлику - за смаком
- соєвий соус - 2-3 ст. л
- оцет бальзамічний - 3-4 ст. л
- сіль за смаком
- вода - 100 мл

■ Приготування (алгоритм дій):

- 1) Куряче м'ясо звільнити від кісток. Порізати на частини.
- 2) Ріпчасту цибулю почистити і порізати кільцями, додати до курки.
- 3) Влити соєвий соус, воду, посипати щедро спеціями для шашлику. Додати бальзамічний оцет.
- 4) Все добре перемішати. Якщо потрібно - додати сіль.
- 5) Залишити курку для маринування на 4-5 годин при кімнатній температурі.
- 6) Нанизати курку на шампури разом з кільцями цибулі.
- 7) Смажити шашлик з курки над розжареним вугіллям до готовності. Перевірити це можна, розрізавши шматочок м'яса ножом. Якщо виділяється прозорий сік - шашлик готовий!
- 8) Подати шалено смачний шашлик з курки гарячим, з овочами, зеленню і улюбленим соусом.



Графічний спосіб запису алгоритму



- **Термінатор** – відображає вхід у зовнішнє середовище (початок) або вихід з нього (кінець)

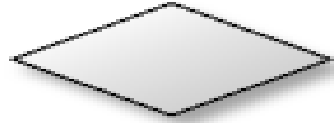


- **Процес** – відображає одну або кілька операцій, обробку даних будь-якого виду



- **Дані** - відображає перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення).

Графічний спосіб запису алгоритму



- **Рішення** – 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 елементу.

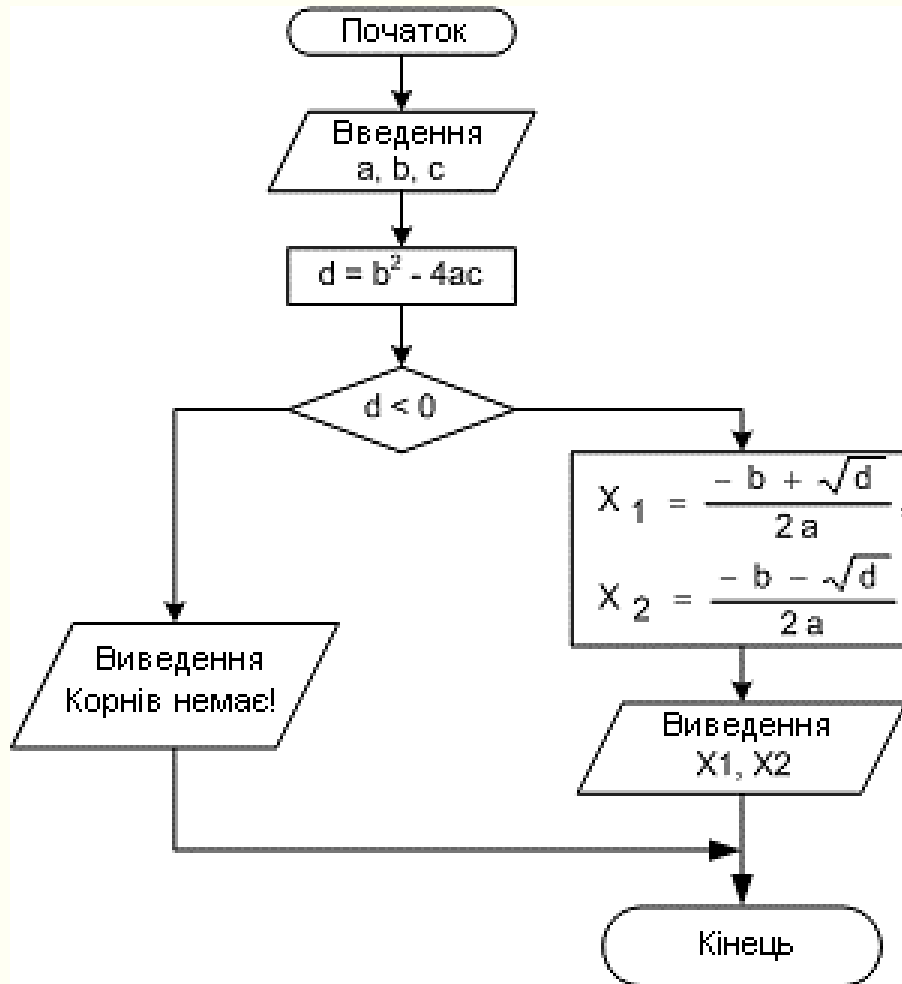


- **Цикл з параметром** – відображає заголовок циклу з параметром. У ньому через крапку з комою вказуються ім'я змінної (параметра) з початковим значенням, граничне значення параметра (або умова виконання циклу), крок зміни параметра.



- **Зумовлений процес** - відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі).

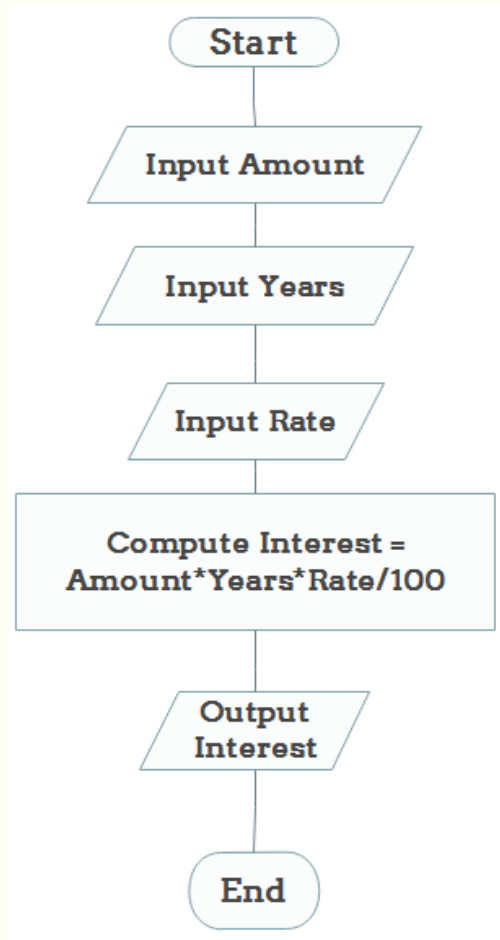
Приклад блок-схеми



■ **Блок-схема** - це покроковий опис алгоритму за допомогою геометричних фігур, що з'єднуються стрілками (які вказують напрям руху).

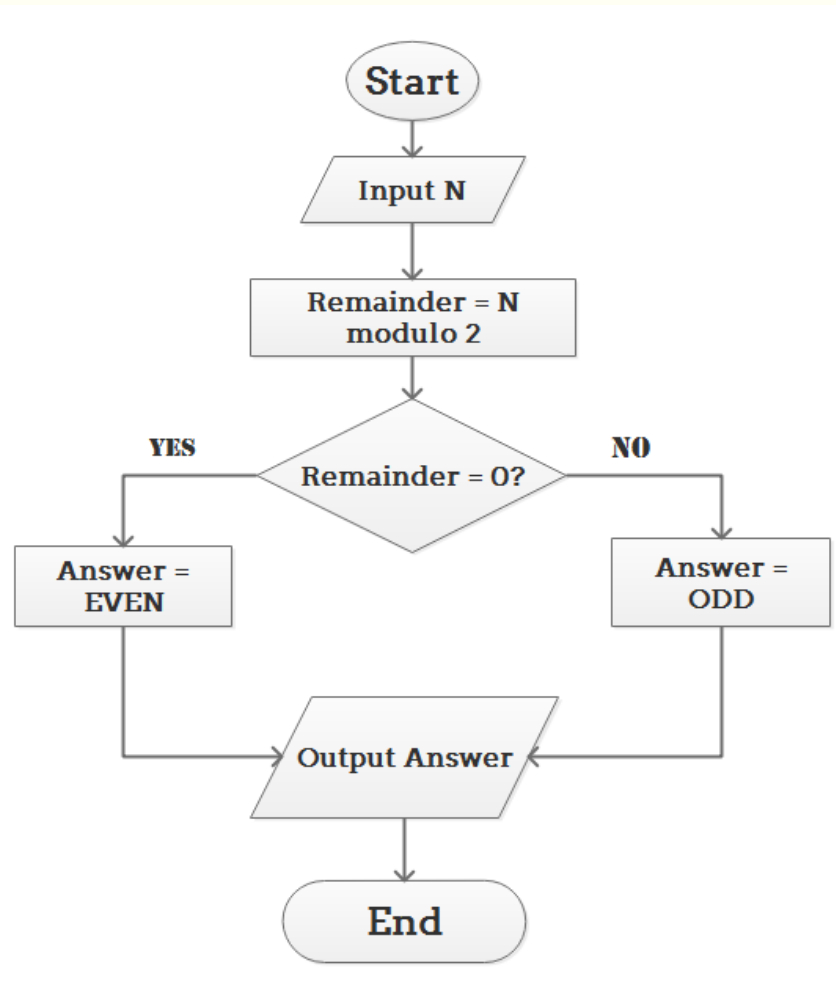
- Усі логічно пов'язані блоки з'єднані стрілками
- Блок-схеми будуються згори донизу
- Блок-схеми починаються і закінчуються термінаторами.

Лінійна блок-схема



- Обчислити відсоток банківського депозиту:
- Алгоритм:
 - Крок 1: зчитати первинний внесок (amount).
 - Крок 2: зчитати кількість років.
 - Крок 3: зчитати відсоткову ставку (rate).
 - Крок 4: обчислити відсоток за формулою $\text{Interest} = \text{Amount} * \text{Years} * \text{Rate} / 100$
 - Крок 5: вивести відсоток (interest)

Блок-схема з умовою



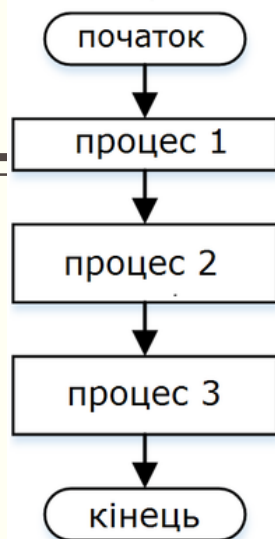
- **Завдання:** визначити, чи є число N парним або непарним.
- **Алгоритм:**
 - Крок 1: зчитати число N ,
 - Крок 2: визначити остачу від ділення N на 2,
 - Крок 3: якщо остача дорівнює 0, число N – парне (even), інакше – непарне (odd),
 - Крок 4: вивести відповідь.

Блок-схема з циклом

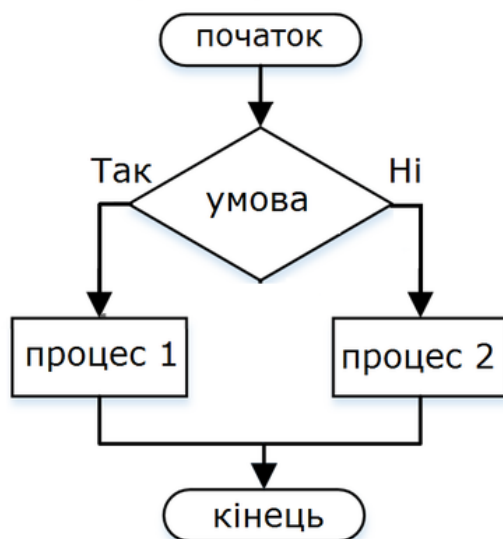


- Цикл використовується за потреби повторювати одну або кілька дій декілька разів.
 - Після кожного кроку (ітерації) буде перевірятись певна умова.

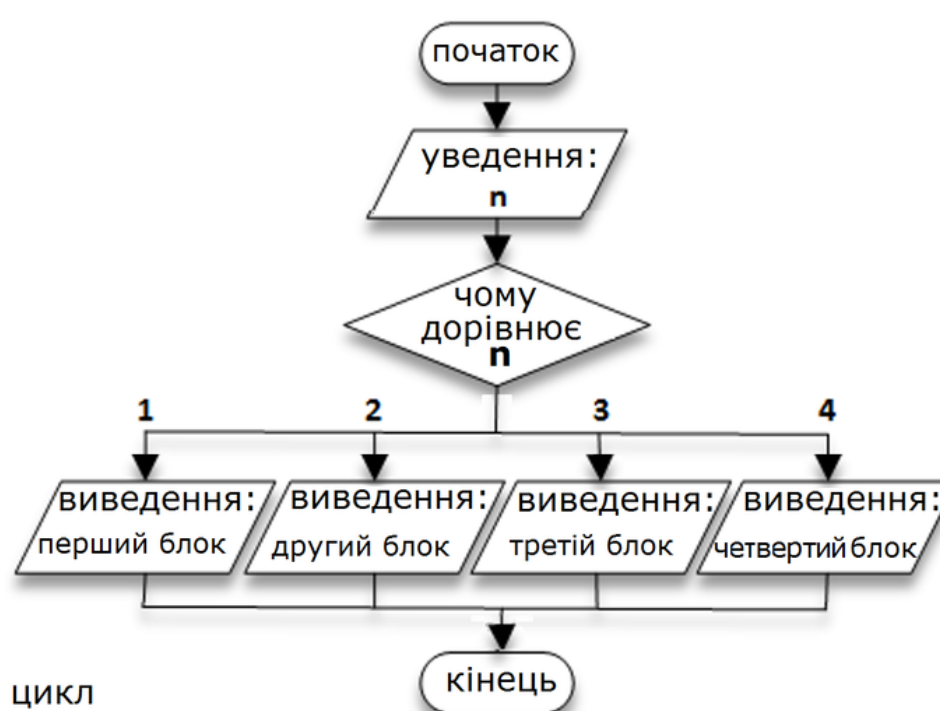
лінійний алгоритм



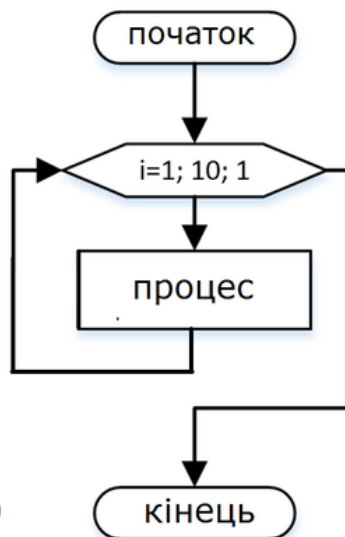
алгоритм з розгалуженням



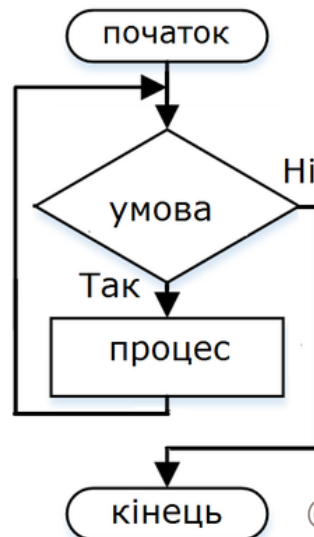
алгоритм з множинним вибором



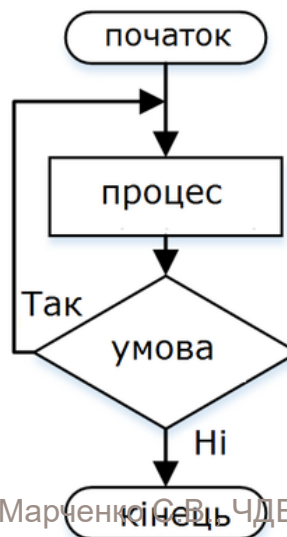
цикл з параметром



цикл з передумовою



цикл з післяумовою



Задача: знаходження найбільшого спільного дільника

- Знайти НСД(gcd) двох невід'ємних цілих чисел m і n , які одночасно не дорівнюють нулю. Використовується для
 - операцій з дробами
 - перевірки взаємної простоти та факторизації чисел у криптографії (наприклад, RSA)
- Розглянемо 3 способи розв'язку:
 - Алгоритм Евкліда
 - Метод послідовного перебору
 - «Шкільний» метод

1. Алгоритм Евклида - ідея

- Базується на рекурентному співвідношенні

$$\gcd(m, n) = \gcd(n, m \bmod n)$$

- Выполнение алгоритма заканчивается, когда выражение $(m \bmod n)$ становится равным нулю.
 - Поскольку $\gcd(m, n) = m$, последнее полученное значение m будет также являться НОД исходных чисел тип.

- Приклад

$$\gcd(60, 24) = \gcd(24, 12) = \gcd(12, 0) = 12.$$

Словесний опис алгоритму

- **Крок 1.** Якщо $n = 0$, повернути m у якості відповіді та закінчити роботу; інакше перейти до кроку 2.
- **Крок 2.** Поділити націло m на n і присвоїти значення залишку змінній r .
- **Крок 3.** Присвоїти значення n змінній m , а значення r — змінній n . Перейти до кроку 1.

Псевдокод алгоритму

АЛГОРИТМ *Euclid* (m, n)

// Алгоритм Евклида вычисляет значение функции $\text{gcd}(m, n)$

// **Входные данные:** два неотрицательных целых числа m и n ,

// которые не могут одновременно быть равны нулю

// **Выходные данные:** наибольший общий делитель чисел m и n

while $n \neq 0$ **do**

$r \leftarrow m \bmod n$

$m \leftarrow n$

$n \leftarrow r$

return m

2. Метод послідовного перебору - ідея

- Базується на підборі найбільшого спільного цілого дільника.
 - Очевидно, що він не перевищує менше число з пари.
- Алгоритм починається з перевірки того, чи діляться обидва числа, m і n , на t без остачі.
 - Якщо так, число t – наша відповідь;
 - Якщо ні, зменшуємо значення t на 1 і знову виконуємо перевірку.
- Для пари (60, 24) перевіряємо спочатку 24, потім 23, потім 22, ... поки $t = 12$.
 - Після цього алгоритм закінчує роботу

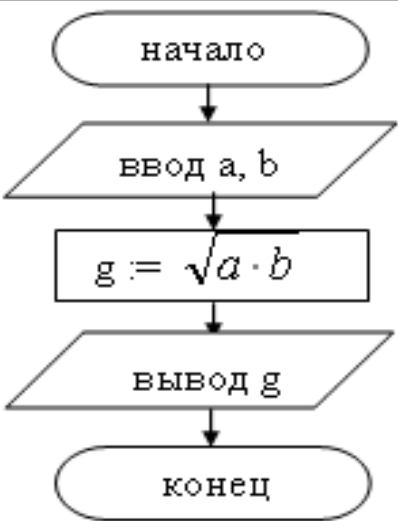
Словесний опис алгоритму

- **Крок 1.** Присвоїти значення функції $\min\{m, n\}$ змінній t .
- **Крок 2.** Розділити m на t . Якщо остача дорівнює нулю, перейти до кроку 3; інакше перейти до кроку 4.
- **Крок 3.** Розділити n на t . Якщо остача дорівнює нулю, повернути t в якості відповіді та завершити роботу; інакше перейти до кроку 4.
- **Крок 4.** Відняти 1 від t . Перейти до кроку 2.
 - На відміну від алгоритму Евкліда, даний алгоритм не буде працювати коректно при наявності нульового параметру.

3. “Шкільний” метод – словесний опис алгоритму

- **Крок 1.** Розкласти на прості множники число m .
- **Крок 2.** Розкласти на прості множники число n .
- **Крок 3.** Для простих множників чисел m і n , знайдених на кроках 1 та 2, виділити їх спільні дільники.
 - Якщо p є спільним дільником чисел m і n та зустрічається в їх розкладанні на прості множники, відповідно, p_m та p_n разів, то при виділенні потрібно повторити це $\min\{p_m, p_n\}$ разів.
- **Крок 4.** Обчислити добуток усіх виділених спільних дільників та повернути його в якості результату пошуку НСД двох вказаних чисел.
- Для пари чисел (60,24) отримаємо:
 - $60 = 2 \cdot 2 \cdot 3 \cdot 5$
 - $24 = 2 \cdot 2 \cdot 2 \cdot 3$
 - $\gcd(60, 24) = 2 \cdot 2 \cdot 3 = 12$.
- Метод набагато складніший та повільніший за алгоритм Евкліда.
 - Прості множники не визначені однозначно (немає списку простих чисел), що не дає написати програму.
 - Окреме питання, як виділяти спільні елементи зі списків?

Способы запису алгоритму

Блок-схема	Псевдокоды	Паскаль
 <pre>graph TD; A([начало]) --> B[/ввод a, b/]; B --> C[g := √(a * b)]; C --> D[/вывод g/]; D --> E([конец]);</pre>	<p><u>алг</u> среднее геометрическое <u>вещ</u> a, b, g <u>нач</u> <u>ввод</u> a, b $g := (a * b) ^ (1/2)$ <u>вывод</u> g <u>кон</u></p>	<pre>program Srednee_geometr; var a, b, g: real; begin readln (a, b); s := sqrt(a * b); writeln (g) end.</pre>

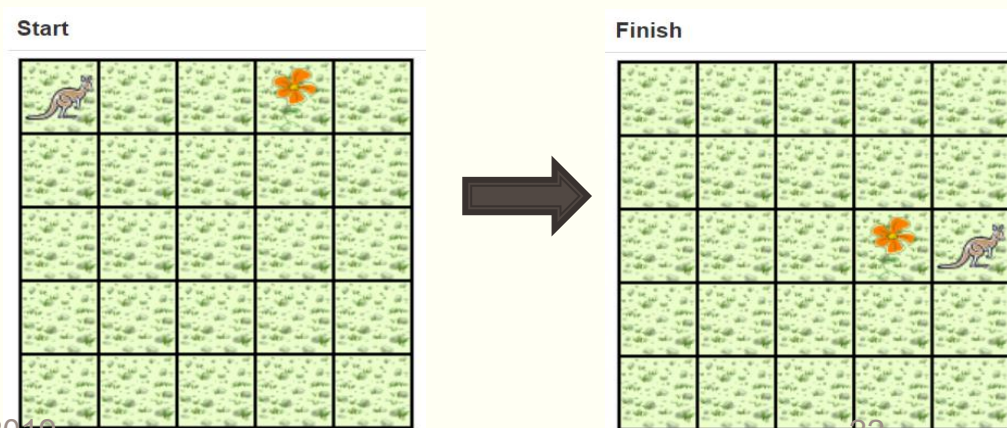
Як спроектувати свій алгоритм?

- **Крок 1: отримати опис задачі.**

- Через невдалий вибір слів постановка задачі може бути проблемною: містити необґрунтовані припущення, неоднозначно трактуватись, неповно описуватись або мати внутрішні суперечності.
- Спочатку розробник має визначити недоліки постановки задачі!

- **Приклад постановки задачі:**

- Кенгуру знаходиться в позиції (0, 0) та рухається на схід, не маючи квіток у своїй сумці.
- На позиції (3, 0) знаходиться квітка.
- Напишіть програму, яка спрямовує кенгуру взяти квітку та пересадити її на позиції (3, 2).
- Після висадки кенгуру має стрибнути на одну позицію на схід.
- На острові більше немає квіток, кенгуру чи перешкод.



Як спроектувати свій алгоритм?

- **Крок 2: проаналізувати задачу.**
 - Квітка знаходиться рівно через 3 клітинки від кенгуру.
 - Квітку слід висадити рівно на 2 клітинки південніше від її позиції.
 - Кенгуру закінчить рух на одну клітинку східніше, ніж квітка.
 - Немає перешкод руху, про які слід турбуватись.
- **Крок 3: розробити високорівневий алгоритм.**
 - 1) зібрати квітку
 - 2) висадити квітку в іншому місці
 - 3) стрибнути на схід

Як спроектувати свій алгоритм?

▪ Крок 4: деталізувати алгоритм.

- 1) *зібрати квітку*
 - Стрибнути тричі
 - Взяти квітку
- 2) *висадити квітку в іншому місці*
 - Повернутись у напрямку півдня
 - Стрибнути двічі
 - Висадити квітку
- 3) *стрибнути на схід*
 - Повернутись у напрямку сходу
 - Стрибнути один раз

▪ Крок 5: виконати загальний огляд алгоритму.

- Високорівневий алгоритм розбив задачу на три досить прості підзадачі.
- Алгоритм розв'язує дуже конкретну задачу, оскільки маємо справу з кенгуру та квіткою в конкретних розташуваннях.
- Алгоритм є розв'язком для дещо загальнішої задачі: кенгуру може починати рухатись із довільної точки.

Уривок реалізації мовою Python

```
def myProgram() :  
    bobby = Jeroo() ;  
  
    // --- зібрати квітку ---  
    bobby.hop(3) ;           // Стрибнути тричі  
    bobby.pick() ;          // Взяти квітку  
  
    // --- висадити квітку в іншому місці ---  
    bobby.turn(RIGHT) ;     // Повернутись у напрямку півдня  
    bobby.hop(2) ;          // Стрибнути двічі  
    bobby.plant() ;         // Висадити квітку  
  
    // --- стрибнути на схід ---  
    bobby.turn(LEFT) ;      // Повернутись у напрямку сходу  
    bobby.hop() ;           // Стрибнути один раз
```

Що далі?

- Розгляньте питання, як розв'язувати довільні задачі
- Корисно почитати:
 - [Стратегії розв'язування задач](#)
 - Програмісти рекомендують: [Пойа Д. Как решать задачу \(1959\)](#)
 - [Один язык чтобы править всеми](#)
 - [Лео Броуди. Способ мышления – ФОРТ. Язык и философия для решения задач](#)
- Відео:
 - [Алгоритмическое мышление 1](#) (Coursera)
 - [Алгоритмическое мышление 2](#) (Coursera)
 - [Блок-схеми, побудова у Word](#)

Що чекає попереду?

- *Основи програмної інженерії*
 - Формування вимог до програмного забезпечення
- *Алгоритми та структури даних*
 - Популярні алгоритми та стратегії розв'язування задач
- *Математична логіка*

Контрольні запитання

- Що таке алгоритм та що для нього властиво?
- Яких дій достатньо для представлення алгоритму?
- Які форми запису алгоритмів існують?
- Яка форма запису алгоритму за своїм виглядом найближча до реального коду?

-
-
- *„Experience is something you don't get until just after you need it.“*



- *Стивен Паўл*

- <https://proglib.io/p/27-puzzle-websites-to-sharpen-your-skills/>



ДЯКУЮ ЗА УВАГУ!

Наступна тема: принципи створення та виконання програм