TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

TIN HỌC ĐẠI CƯƠNG

Phần 1. Tin học căn bản

Chương 3: Thuật toán

Nội dung

Nội dung

3.1. Bài toán (problem)

- **"Bài toán" hay "Vấn đề"**
  - Vấn đề có nghĩa rộng hơn bài toán
  - Bài toán là một loại vấn đề mà để giải quyết phải liên quan ít nhiều đến tính toán: bài toán trong vật lý, hóa học, xây dựng, kinh tế…
- **Hai loại vấn đề (Pitago)**
  - Theorema: là vấn đề cần được khẳng định tính đúng sai.
  - Problema: là vấn đề cần tìm được giải pháp để đạt được một mục tiêu xác định từ những điều kiện ban đầu nào đó.

3.1. Bài toán (2)

- **Biểu diễn vấn đề-bài toán**
  - A → B
    - A: Giả thiết, điều kiện ban đầu
    - B: Kết luận, mục tiêu cần đạt
- **Giải quyết vấn đề-bài toán**
  - Từ A dùng một số hữu hạn các bước suy luận có lý hoặc hành động thích hợp để đạt được B
  - Trong Tin học, A là đầu vào, B là đầu ra

Ví dụ

1. Cho một dãy số, tìm số lớn nhất

A: dãy số cho trước

B: số lớn nhất trong dãy số

2. Cho một dãy số bất kỳ và số a, kiểm tra dãy số có chứa số a hay không?

A: dãy số cho trước, số a

B: kết luận dãy số có chứa số a hay không?

3. Cho một dãy số, sắp xếp theo chiều tăng dần

A: dãy số cho trước

B: dãy số đã được sắp xếp theo chiều tăng dần

Nội dung

3.2. Giải quyết bài toán bằng máy tính

- Máy tính không thể dùng để giải quyết các vấn đề liên quan đến hành động vật lý hoặc biểu thị cảm xúc
- Máy tính chỉ làm được những gì mà nó được bảo phải làm. Máy tính không thông minh, nó không thể tự phân tích vấn đề và đưa ra giải pháp.
- Lập trình viên là người phân tích vấn đề, tạo ra các chỉ dẫn để giải quyết vấn đề (chương trình), và máy tính sẽ thực hiện các chỉ dẫn đó

3.2. Giải quyết bài toán bằng máy tính (2)

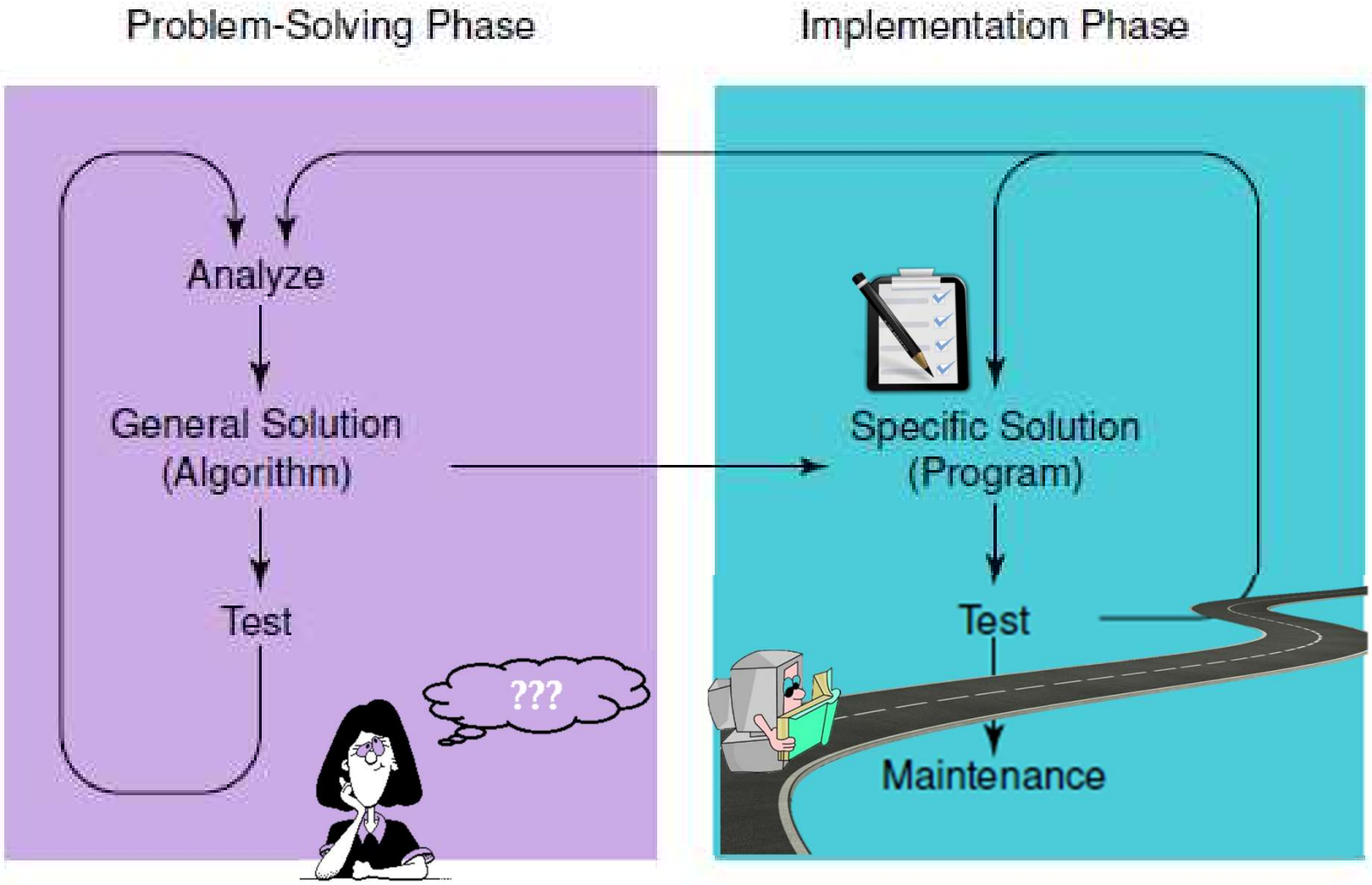
- **Phương án giải quyết bài toán được gọi là thuật toán/giải thuật (algorithm) trong tính toán**
- **Một thuật toán là:**
  - một dãy **hữu hạn** các thao tác và **trình tự** thực hiện các thao tác đó sao cho sau khi thực hiện dãy thao tác này theo trình tự đã chỉ ra, với **đầu vào** (input) ta thu được kết quả **đầu ra** (output) mong muốn.

3.2. Giải quyết bài toán bằng máy tính (3)

- **Không chỉ đơn giản là lập trình**
- **Phức tạp, gồm nhiều giai đoạn phát triển**
- **Các giai đoạn quan trọng**

Bước 1.	Xác định yêu cầu bài toán	
Bước 2.	Phân tích và thiết kế bài toán	Lựa chọn phương án Xây dựng thuật toán
Bước 3.	Lập trình	
Bước 4.	Kiểm thử và hiệu chỉnh chương trình	
Bước 5.	Triển khai và bảo trì	

Hai giai đoạn chính để hiện thực hóa bài toán



Giai đoạn giải quyết vấn đề

Giai đoạn thực hiện

Nội dung

3.1. Bài toán (problem)

3.2. Giải quyết bài toán bằng máy tính

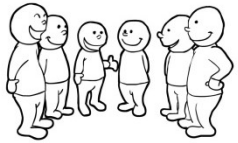
3.3. Biểu diễn thuật toán

3.4. Một số thuật toán thông dụng

3.3. Biểu diễn thuật toán

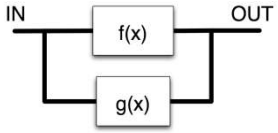
Cách 1:

- Ngôn ngữ tự nhiên



Cách 2:

- Ngôn ngữ lưu đồ (lưu đồ/sơ đồ khối)



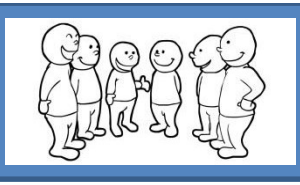
Cách 3:

- Mã giả (pseudocode) gọi là ngôn ngữ mô phỏng chương trình PDL (Programming Description Language).

Cách 4:

- Các ngôn ngữ lập trình như Pascal, C/C++ hay Java. uy nhiên, không nhất thiết phải sử dụng đúng ký pháp của các ngôn ngữ đó mà có thể được bỏ một số ràng buộc.

3.3.1. Ngôn ngữ tự nhiên



- **Sử dụng một loại ngôn ngữ tự nhiên để liệt kê các bước của thuật toán**
- **Ưu điểm**
  - Đơn giản
  - Không yêu cầu người viết và người đọc phải có kiến thức nền tảng
- **Nhược điểm**
  - Dài dòng
  - Không làm nổi bật cấu trúc của thuật toán
  - Khó biểu diễn với những bài toán phức tạp

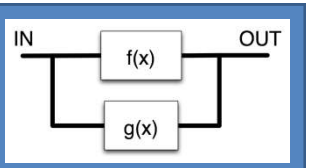
Ví dụ 1

- **Bài toán: Đưa ra kết luận về tương quan của hai số a và b (>, < hay =).**
  - Đầu vào: Hai số a và b
  - Đầu ra: Kết luận a>b hay a<b hay a=b.
- **Ý tưởng:**
  - So sánh a và b rồi đưa ra kết luận

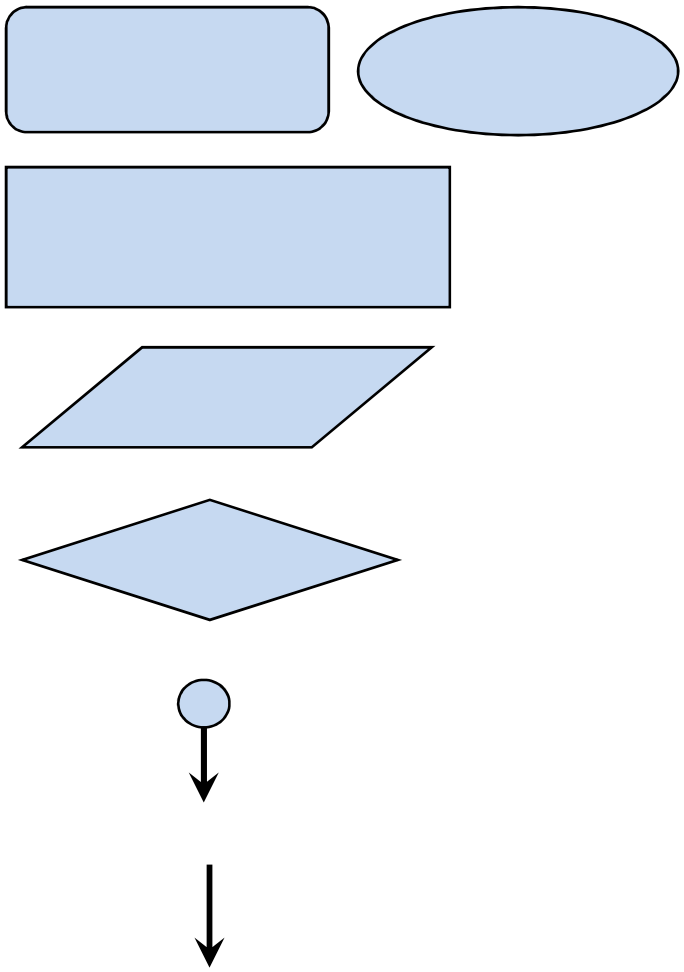
VD1 - Ngôn ngữ tự nhiên (tuần tự các bước)

- B1: Nhập số a và số b.
- B2: Nếu a> b, hiển thị “a>b” và kết thúc
- B3: Nếu a=b, hiển thị “a=b” và kết thúc
- B4: (a<b) Hiển thị “a<b” và kết thúc.

3.3.2. Sơ đồ khối



Một số khối trong sơ đồ khối dùng biểu diễn thuật toán



Bắt đầu hoặc ***kết thúc***

Thao tác tính toán hoặc xử lý

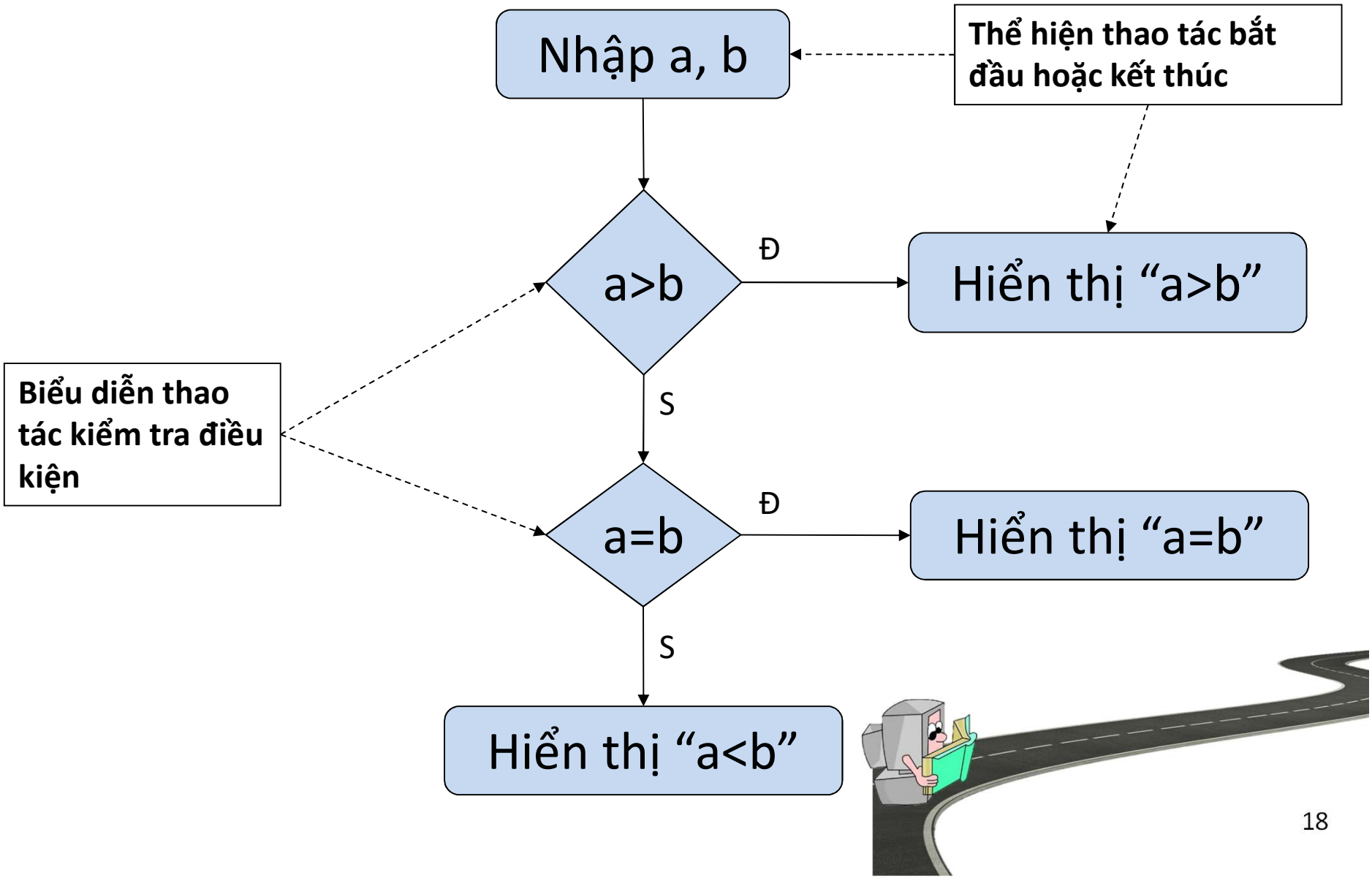
Lệnh vào, lệnh ra *(read hoặc write)*

Kiểm tra điều kiện

Nối tiếp đoạn lệnh

Luồng thực hiện

Ví dụ 1 - Mô tả bằng lưu đồ thuật toán



3.3.2. Sơ đồ khối (2)

- **Ưu điểm**
  - Trực quan, dễ hiểu, dễ thiết kế
  - Cung cấp toàn cảnh, tổng quan về thuật toán
- **Nhược điểm**
  - Cồng kềnh, đặc biệt với bài toán phức tạp

3.3.3. Mã giả (pseudocode)

- **Ngôn ngữ tựa (gần giống) với ngôn ngữ lập trình được gọi là mã giả**
  - Mệnh đề có cấu trúc
  - Ngôn ngữ tự nhiên
- **Ưu điểm**
  - Tiện lợi, đơn giản
  - Dễ hiểu, dễ diễn đạt

Ví dụ 2

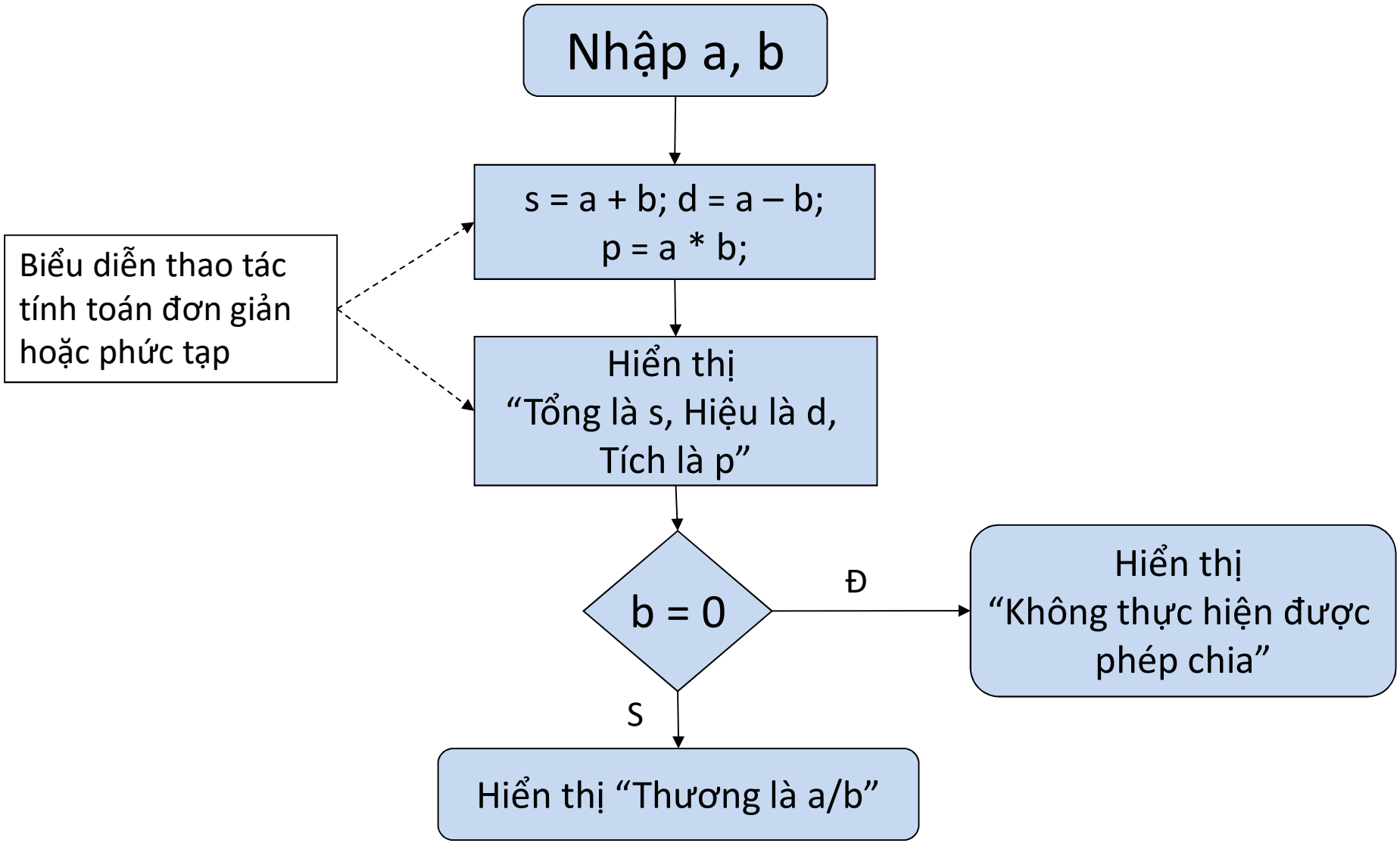
- **Bài toán: Đưa ra tổng, tích, hiệu, thương của hai số a và b.**
  - Đầu vào: Hai số a và b
  - Đầu ra: Tổng, tích, hiệu và thương của a và b.
- **Ý tưởng:**

  - Tính tổng, tích, hiệu của a và b
  - Nếu b khác 0, đưa ra thương
  - Nếu b bằng 0, đưa ra thông báo không thực hiện được phép chia

VD2 - Ngôn ngữ tự nhiên (tuần tự các bước)

- **B1: Nhập số a và số b.**
- **B2: Gán s = a + b, d = a – b, p = a * b**
- **Hiển thị**
  - Tổng là s
  - Hiệu là d
  - Tích là p
- **B3: Nếu b = 0, hiển thị "Không thực hiện được phép chia" và kết thúc**
- **B4: (b<>0) Hiển thị "Thương là a/b" và kết thúc**

Ví dụ 2 - Mô tả bằng lưu đồ thuật toán



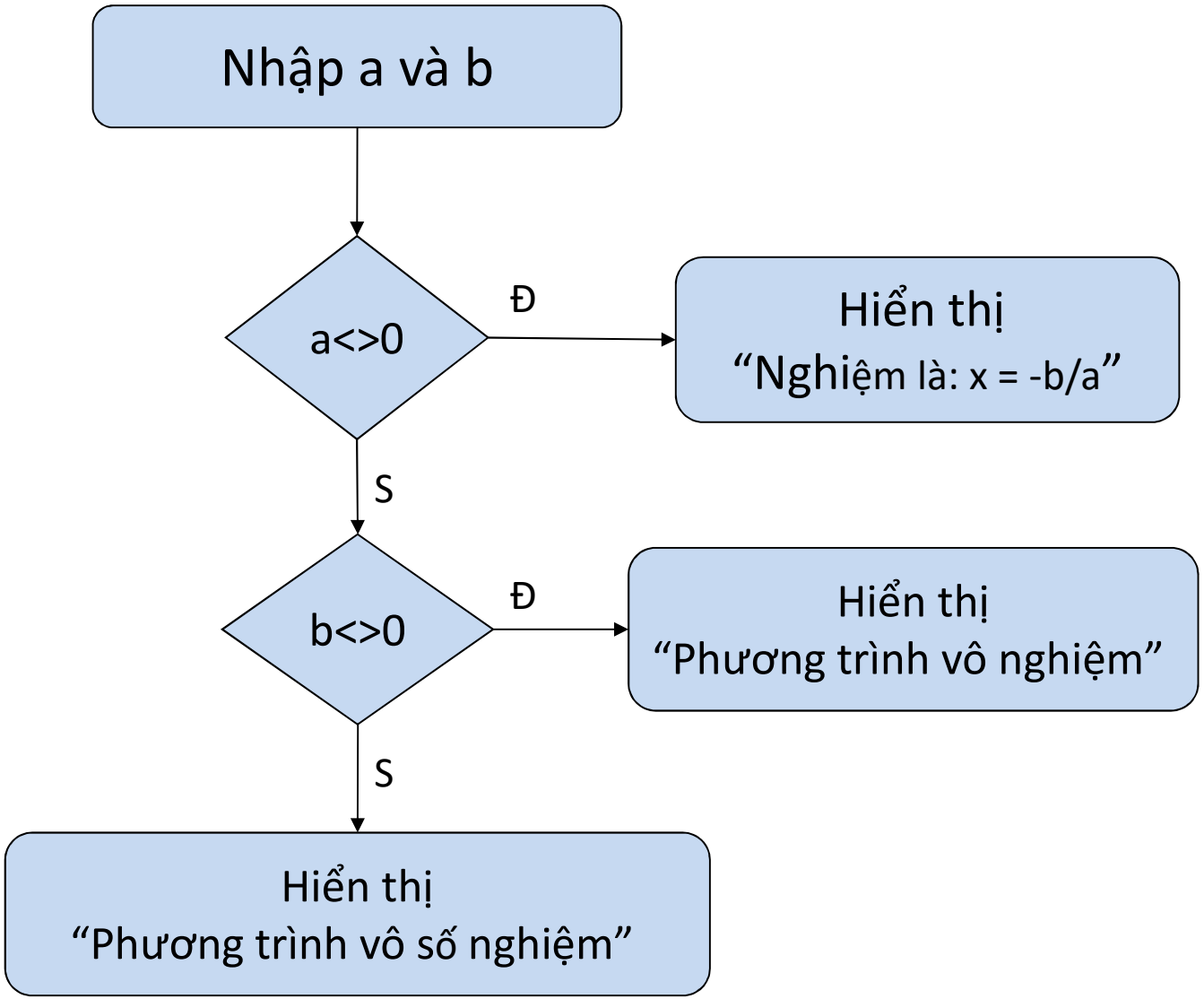
Ví dụ 3

- **Bài toán: Giải phương trình bậc I**
  - Đầu vào: Hai hệ số a, b
  - Đầu ra: Nghiệm của phương trình ax + b = 0
- **Ý tưởng:**
  - Lần lượt xét a = 0 rồi xét b = 0 để xét các trường hợp của phương trình

Ví dụ 3 - Mô tả tuần tự các bước

- B1: Nhập a và b.
- B2: Nếu a<>0 thì hiển thị "Phương trình có 1 nghiệm duy nhất x = -b/a".
- B3: (a=0) Nếu b <> 0 thì hiển thị "Phương trình vô nghiệm" và kết thúc
- B4: (a=0)(b=0) Hiển thị "Phương trình vô số nghiệm" và kết thúc

Ví dụ 3 - Mô tả bằng lưu đồ thuật toán



Ví dụ 4

- **Bài toán: Tìm giá trị lớn nhất của một dãy số nguyên có N số**
  - Đầu vào: Số số nguyên dương N và N số nguyên $a_1, a_2, \ldots, a_N$
  - Đầu ra: số nguyên lớn nhất của dãy a_k, k trong khoảng [1…N]
- **Ý tưởng:**
  - Khởi tạo giá trị max = a_1
  - Lần lượt so sánh max với a_i với i=2, 3, …, N; nếu a_i > max ta gán giá trị mới cho max

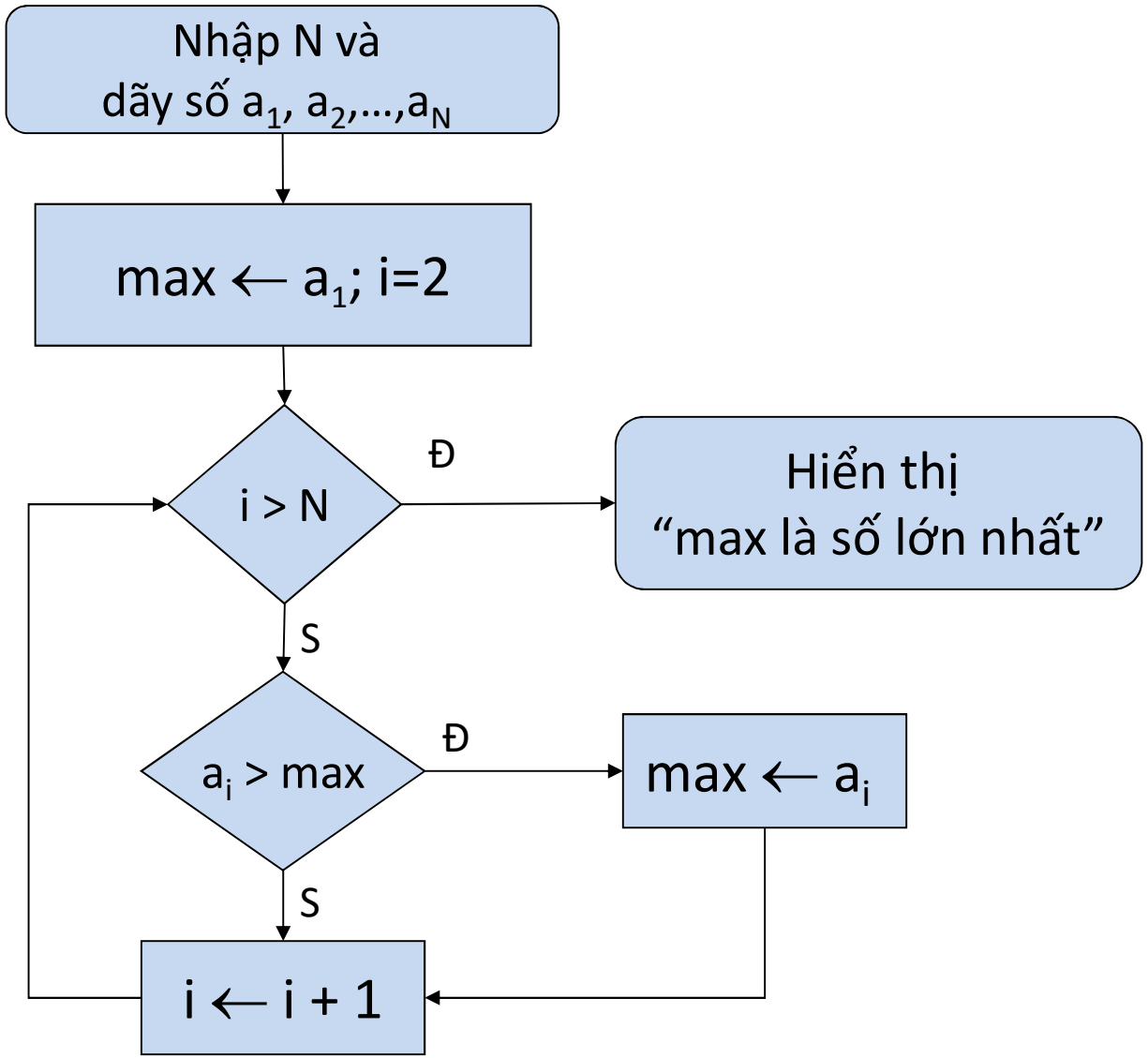
Ví dụ 4 – Ý tưởng

max	3	5	7	9	2	8
max = 3	3					
max < 5		5				
max < 7			7			
max < 9				9		
max > 2					9	
max > 8						9
Kết quả	9					

Ví dụ 4 - Mô tả tuần tự các bước

- B1: Nhập N và dãy số a_1, a_2, …, a_N.
- B2: Gán max = a_1, i =2
- B3: Nếu i > N, hiển thị max là giá trị lớn nhất của dãy và kết thúc
- B4: Nếu a_i > max, gán max = a_i
- B5: Tăng i lên 1 đơn vị.
- B6: Quay lên B3.
- B7: Kết thúc.

Ví dụ 4 - Mô tả bằng lưu đồ thuật toán



Bài tập

- **Bài toán: Giải phương trình bậc II**
  - Đầu vào: Ba hệ số a, b, c
  - Đầu ra: Nghiệm của phương trình

$$ax^2 + bx + c = 0$$

- **Ý tưởng:**
  - Lần lượt xét a = 0, b = 0 rồi xét c=0 để xét các trường hợp của phương trình
- **Yêu cầu: Vẽ sơ đồ khối biểu diễn thuật toán**

Các tiêu chí giải thuật cần thỏa mãn

- **Tính hữu hạn:** giải thuật phải dừng sau một thời gian hữu hạn.
- **Tính đúng đắn:** Khi kết thúc, giải thuật phải cung cấp kết quả đúng đắn.
- **Tính hiệu quả:**
  - Thời gian tính toán nhanh
  - Sử dụng ít tài nguyên không gian như bộ nhớ, thiết bị,…
  - Mang tính phổ dụng, dễ hiểu, dễ cài đặt và mở rộng cho các lớp bài toán khác.

Nội dung

3.1. Bài toán (problem)

3.2. Giải quyết bài toán bằng máy tính

3.3. Biểu diễn thuật toán

3.4. Một số thuật toán thông dụng

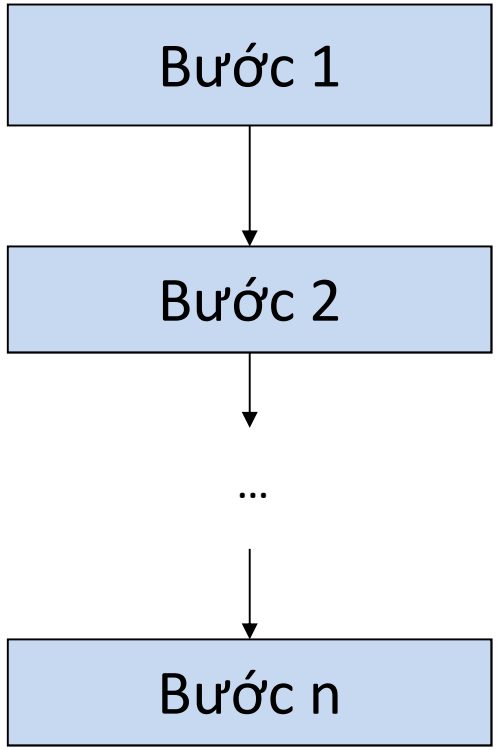
Nội dung

3.4.1. Các cấu trúc cơ bản trong lập trình

- **Cấu trúc tuần tự**
- **Cấu trúc rẽ nhánh**
- **Cấu trúc lặp**

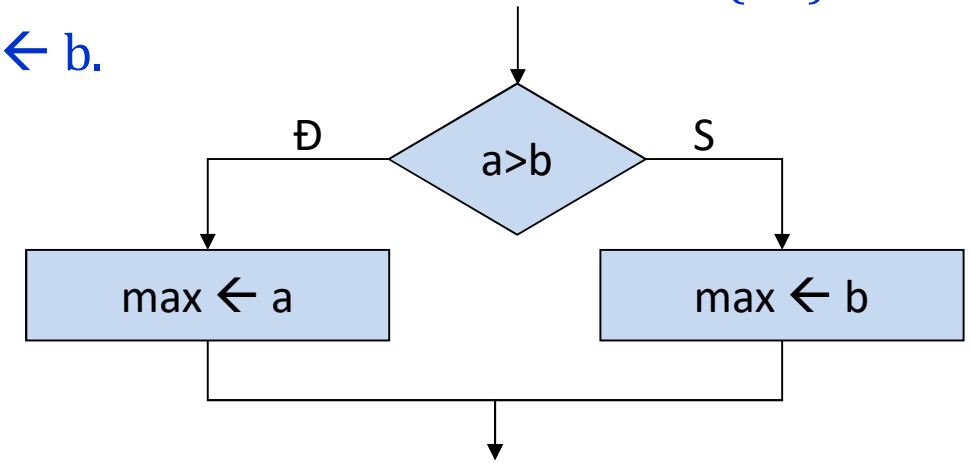
Cấu trúc tuần tự

- **Các bước được thực hiện theo 1 trình tự tuyến tính, hết bước này đến bước khác**



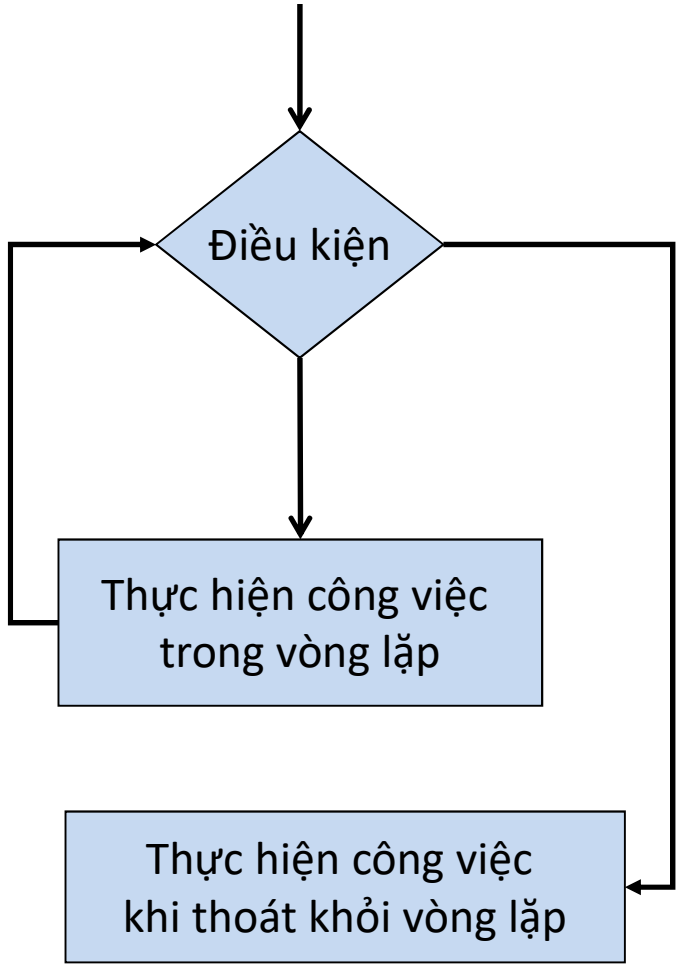
Cấu trúc rẽ nhánh

- **Việc thực hiện bước nào phụ thuộc vào điều kiện xác định.**
- **Ví dụ: Tìm max của 2 số a, b.**
  - Nếu a > b thì max là a, ngược lại max sẽ là b.
  - Diễn giải:
    - B1: Nhập 2 số a, b.
    - B2: Nếu a > b thì max = a và đi đến bước kết thúc (B4).
    - B3: (a <= b) max ← b.
    - B4: Kết thúc.



Cấu trúc lặp

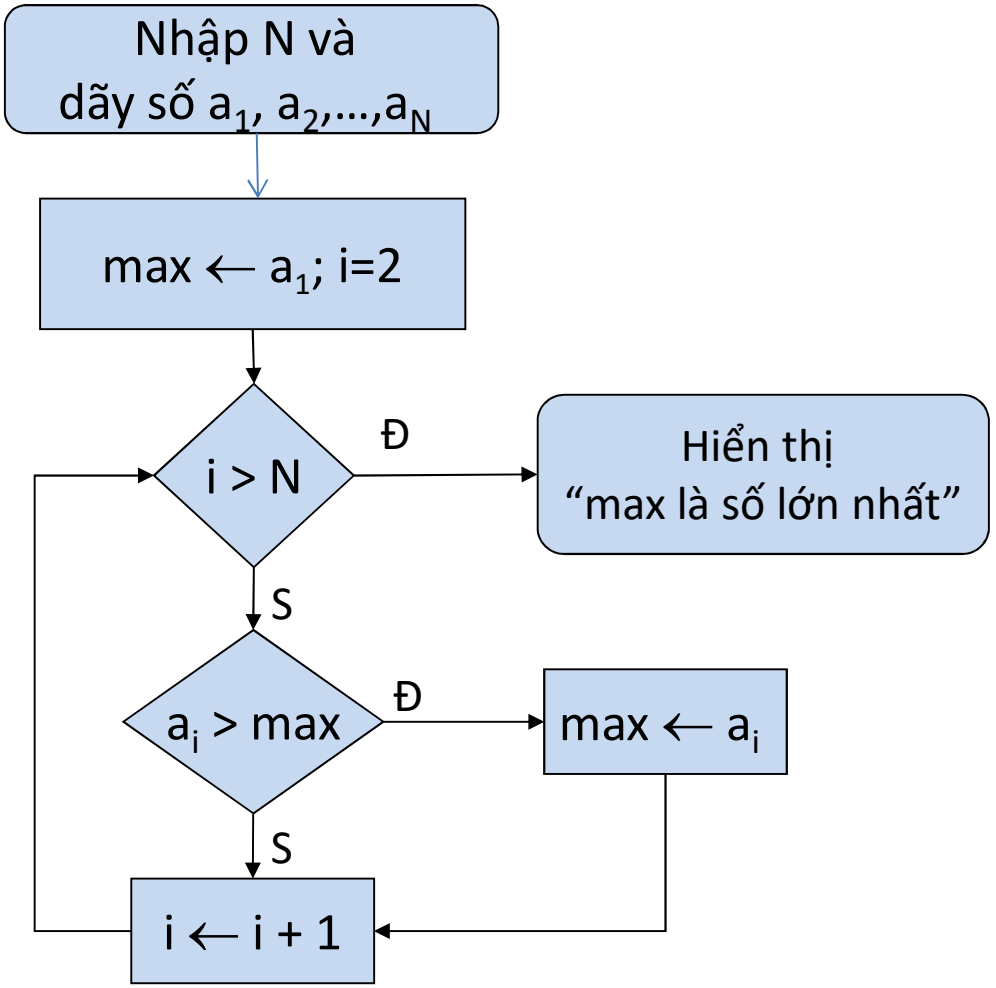
- **Một tác động/nhiệm vụ có thể được thực hiện lặp nhiều lần.**
- **Số lần lặp có thể biết trước hoặc không biết trước. Tuy nhiên số lần lặp phải hữu hạn.**
- **Kết hợp giữa điều kiện và công việc cần thực hiện**



Cấu trúc lặp (2)

Ví dụ: Tìm số lớn nhất của một dãy có n số

- Lần lượt phải so sánh số max tạm thời (lúc đầu max được gán bằng phần tử thứ nhất, a_1) với a_i, với i từ 2, 3,…, n.
- Việc so sánh này được thực hiện lặp nhiều lần giữa max và a_i.
- Khi kết thúc quá trình lặp, ta sẽ thu được max là số lớn nhất của dãy n số.



Nội dung

3.4.1. Các cấu trúc cơ bản trong lập trình

3.4.2. Giả mã (pseudocode)

3.4.3. Thuật toán số học

3.4.4. Thuật toán về dãy

3.4.5. Thuật toán đệ quy

Mã giả (pseudocode)

- **Phép gán: ← hoặc :=**

  Ví dụ: i ← i + 1

  a := b + c

- **Cấu trúc rẽ nhánh**

  if (điều kiện) **then** (hành động)

  hoặc

  if (điều kiện) **then** (hành động 1)

  else (hành động 2)

- **Cấu trúc nhảy goto:**

  – **goto** nhãn x;

Giả mã (2)

- **Cấu trúc lặp:**

  while điều_kiện **do** hành_động

  hoặc

  repeat

  hành_động

  until điều_kiện

  hoặc

  for biến:= gtrị_đầu **to** gtrị_cuối **do** hành_động

  (giá trị đầu < giá trị cuối)

  hoặc

  for biến:= gtrị_đầu **downto** gtrị_cuối **do** hành_động

  (giá trị đầu > giá trị cuối)

Nội dung

3.4.1. Các cấu trúc cơ bản trong lập trình

3.4.2. Giả mã (pseudocode)

3.4.3. Thuật toán số học

3.4.4. Thuật toán về dãy

3.4.5. Thuật toán đệ quy

Thuật toán số học

- **Các bài toán về số học**
  - Xác định một số nguyên có phải là số nguyên tố/hợp số hay không
  - Tìm USCLN, BSCNN của 2 số nguyên
  - ..

Bài toán số nguyên tố

- **Cho một số nguyên dương p. Làm thế nào để biết được p có phải số nguyên tố hay không?**
  - Input: p nguyên dương
  - Output: kết luận về tính nguyên tố của p
- **Ý tưởng?**
  - p = 1? → Không phải số nguyên tố
  - p > 1?
    - Kiểm tra từ 2 đến p-1 có phải là ước số của p không
    - Nếu có thì kết luận p không là số nguyên tố, ngược lại không có số nào thì kết luận p là số nguyên tố

Bài toán số nguyên tố (2)

```
Nhập p
if p=1 then begin
                    Xuất: p không nguyên tố;
                    Dừng thuật toán;
               end
flag := TRUE   //Cờ trạng thái cho biết có tìm được ước nào của p
    không
for k:=2 to p-1 do //Tối ưu duyệt đến [căn bậc 2 của p]
    if (k là ước số của p) then
          begin
              flag:=FALSE
              break //ngắt vòng lặp FOR
           end
if flag=TRUE then
          Xuất: p là số nguyên tố
else
```

Bài tập

- **Bài 1. Xây dựng thuật toán tìm USCLN và BSCNN của 2 số a và b.**
- **Bài 2. Xây dựng thuật toán kiểm tra xem số a có phải là một số hoàn hảo hay không.**
- **Bài 3. Xây dựng thuật toán tính tổng các chữ số của số a.**
- **Bài 4. Xây dựng thuật toán tính tổng của n số Fibonaci đầu tiên.**

Nội dung

3.4.1. Các cấu trúc cơ bản trong lập trình

3.4.2. Giả mã (pseudocode)

3.4.3. Thuật toán số học

3.4.4. Thuật toán về dãy

3.4.5. Thuật toán đệ quy

Thuật toán về dãy

- **Làm việc với một dãy số**
- **Các bài toán điển hình**
  - Tìm số lớn nhất, nhỏ nhất trong dãy
  - Kiểm tra dãy có phải là dãy tăng hoặc dãy giảm
  - Sắp xếp dãy tăng dần hoặc giảm dần
  - Tìm trong dãy có phần tử nào bằng một giá trị cho trước
  - Tính trung bình cộng của dãy
  - …

Ví dụ 1 - Tìm số lớn nhất trong dãy

- **Input: dãy số $a_1, a_2, a_3, \ldots a_n$**
- **Output: max là giá trị lớn nhất trong dãy số đã cho**
- **Thuật toán:**

max:=a_1
for i:=2 **to** n **do**
if max < a_i **then** max:= a_i
Xuất: max là giá trị lớn nhất trong dãy số

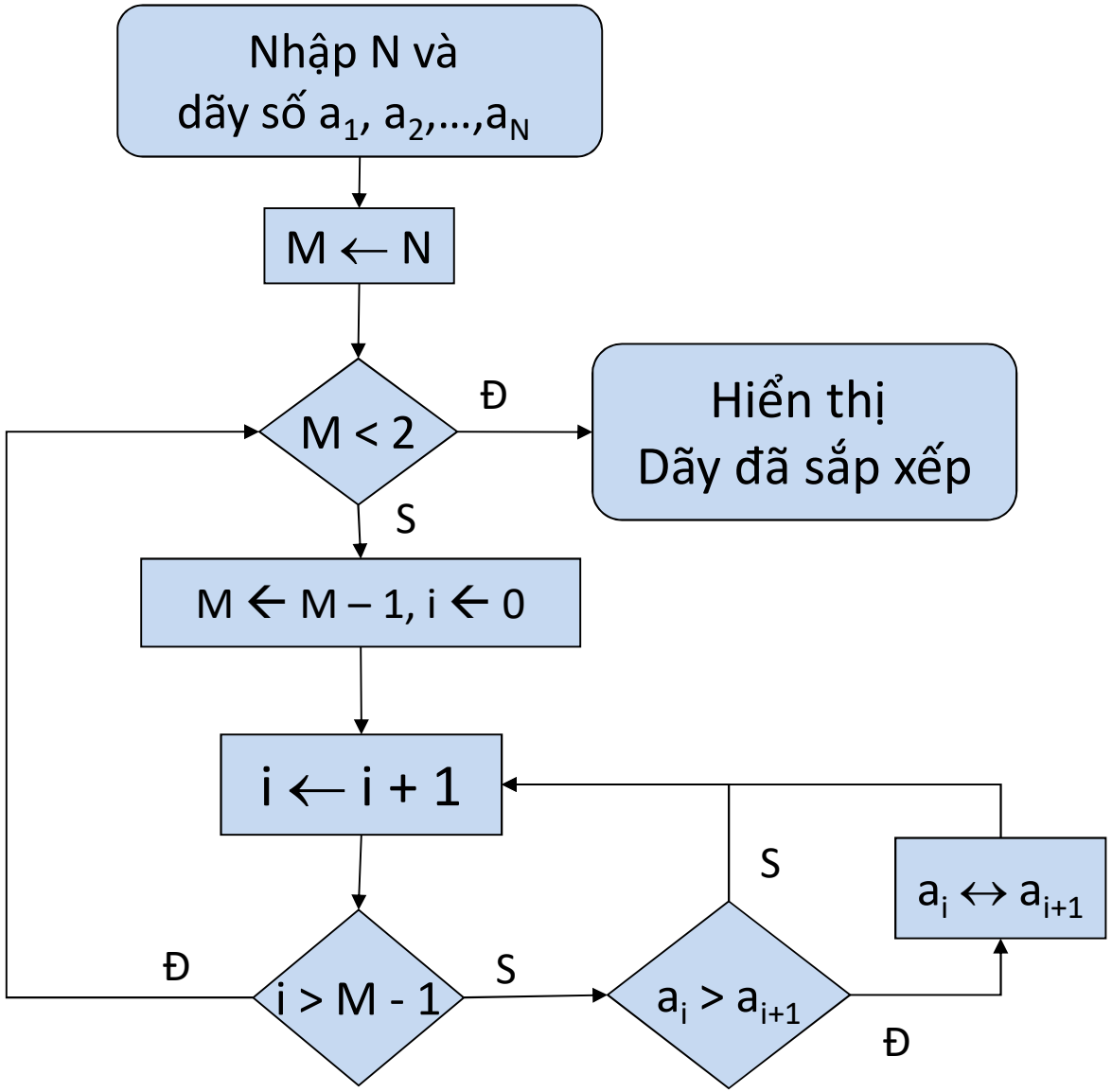
Ví dụ 2. Sắp xếp dãy số

- **Bài toán: Sắp xếp bằng phương pháp nổi bọt (Bubble Sort)**
  - Đầu vào: Dãy a gồm N số nguyên a_1, a_2,..., a_N
  - Đầu ra: Dãy a được sắp lại theo thứ tự không giảm.
- **Ý tưởng:**
  - Với mỗi cặp số liên tiếp trong dãy, nếu số trước lớn hơn số sau ta đổi chỗ chúng cho nhau.
  - Việc đó được lặp cho đến khi không có sự đổi chỗ nào cho nhau

Ví dụ 2 - Mô tả tuần tự các bước

- **B1: Nhập số N và dãy số $a_1,a_2,...,a_N$**
- **B2: M ← N.**
- **B3: Nếu M < 2 thì thuật toán kết thúc và hiển thị dãy đó.**
- **B4: M ← M – 1, i ← 0.**
- **B5: Tăng i lên 1 đơn vị.**
- **B6: Nếu i > M - 1 thì quay lại B3.**
- **B7: Nếu $a_i > a_{i+1}$ thì tráo đổi hai số đó cho nhau**
- **B8: Quay lên B5.**

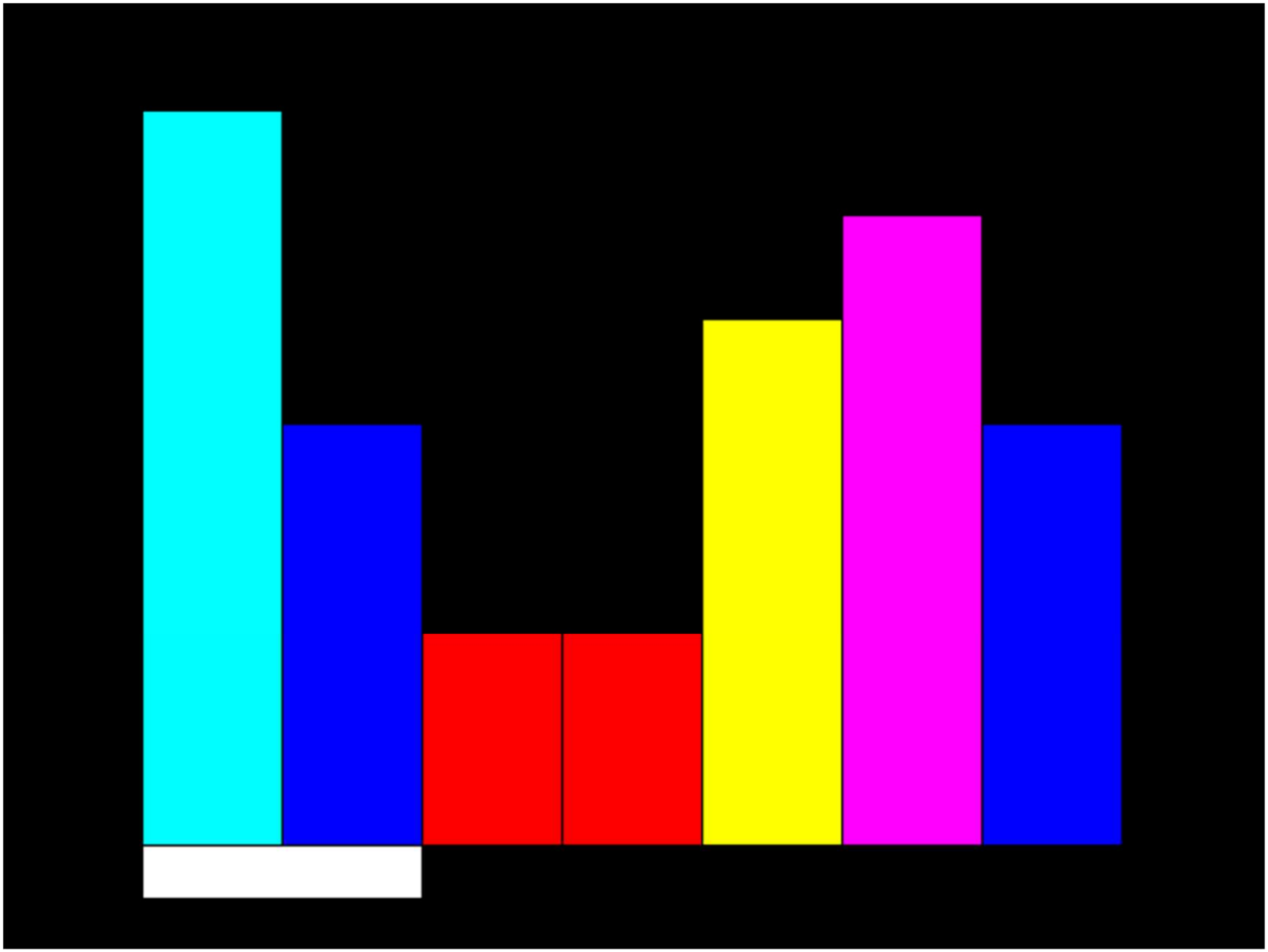
Ví dụ 2 - Mô tả bằng lưu đồ thuật toán



Ví dụ 2 – Mô tả bằng giả mã

Nhập N và dãy số $a_1, a_2, \ldots, a_N$
for M := N **downto** 2 **do**
    for i := 1 **to** M - 1 **do**
        if $a_i > a_{i+1}$ **then**
            Đổi chỗ a_i và a_{i+1}
Xuất ra dãy đã được sắp xếp

Ví dụ 2 – Minh hoạ



Ví dụ 2 – Minh hoạ

Nhập dãy gồm 5 số 6 4 7 2 3

M = 5

i = 1, a1 > a2 => đổi chỗ a1 với a2, dãy số 4 6 7 2 3

i = 2, a2 < a3

i = 3, a3 > a4 => đổi chỗ a3 với a4, dãy số 4 6 2 7 3

i = 4, a4 > a5 => đổi chỗ a4 với a5, dãy số 4 6 2 3 7

M = 4, dãy số 4 2 3 6 7

M = 3, dãy số 2 3 4 6 7

M = 2, dãy số 2 3 4 6 7

Dừng thuật toán, dãy số đã sắp xếp 2 3 4 6 7

Bài tập

- **Bài 1. Xây dựng thuật toán tìm phần tử có giá trị truyệt đối lớn nhất trong dãy gồm n phần tử.**
- **Bài 2. Xây dựng thuật toán tìm tổng của các số chẵn và tổng của các số lẻ trong dãy gồm n phần tử.**
- **Bài 3. Xây dựng thuật toán kiểm tra xem một dãy số gồm n phần tử có phải là dãy số tăng (hoặc giảm) không.**
- **Bài 4. Xây dựng thuật toán tính trung bình cộng của các số dương trong dãy gồm n phần tử.**

Nội dung

3.4.1. Các cấu trúc cơ bản trong lập trình

3.4.2. Giả mã (pseudocode)

3.4.3. Thuật toán số học

3.4.4. Thuật toán về dãy

3.4.5. Thuật toán đệ quy

Thuật toán đệ quy

- **Với bài toán có thể được phân tích và đưa tới việc giải một bài toán cùng loại nhưng cấp độ thấp hơn**
  - độ lớn dữ liệu nhập nhỏ hơn
  - giá trị cần tính toán nhỏ hơn

  → Tự thực hiện lại thuật toán
- **Ví dụ:**
  - Giai thừa: n! = (n-1)! * n
  - Dãy số Fibonacci: 0, 1, 1, 2, 3, 5, 8...
    - F(n) = F(n-1) + F(n-2)

Thuật toán đệ quy (2)

- **Để xây dựng thuật toán đệ quy, cần xác định:**
  - Trường hợp cơ bản: (Các) trường hợp không cần thực hiện lại thuật toán.
  - Phần tổng quát: Có yêu cầu gọi đệ quy
    - Cần xác định nguyên lý đưa trường hợp tổng quát về trường hợp cơ bản
    - Đảm bảo tính dừng của giải thuật đệ quy - chắc chắn từ trường hợp tổng quát sẽ đến được trường hợp cơ bản

Ví dụ

- **Tính giai thừa của n:**
  - Trường hợp cơ bản: 1! = 1
  - Trường hợp tổng quát: n! = (n-1)! * n
- **Xây dựng dãy Fibonacci**
  - Trường hợp cơ bản: F(0) = F(1) = 1
  - Trường hợp tổng quát: F(n) = F(n-1) + F(n-2)

Tính giai thừa - Thuật toán đệ quy

- **Input: số tự nhiên n**
- **Output: GT(n)=n!**
- **Thuật giải:**

```
Nhập n
GT:=1;
if n>0 then
        GT := GT(n-1)*n;
Xuất GT
```

Ví dụ

- **Xây dựng thuật toán cho bài toán tìm số Fibonacci F(n)**

- **Thuật giải:**

```
Nhập n
Fibo:=1;
if n>1 then
   Fibo:= Fibo(n-1) + Fibo(n-2);
Xuất Fibo
```

Bài tập

- **Bài 1. Xây dựng thuật toán đệ quy tính X^n, X là số thực, n là số nguyên.**

  Gợi ý: $X^n = X * X^{n-1}$, $X^1 = X$

- **Bài 2. Xây dựng thuật toán đệ quy tính USCLN của 2 số nguyên dương a và b.**

  Gợi ý:

  - Nếu a = b thì USCLN(a, b) = a
  - Nếu a > b thì USCLN(a, b) = USCLN(a – b, b)
  - Nếu a < b thì USCLN(a, b) = USCLN(a, b – a)

Thuật giải heuristic

- **Dùng "mẹo"**
- **Áp dụng với những bài toán**
  - Chưa tìm được thuật toán và không biết có tồn tại thuật toán không
  - Có thuật toán nhưng thời gian tính toán quá lâu hoặc điều kiện của thuật toán khó đáp ứng