

HUST

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

ET2100

ONE LOVE. ONE FUTURE.



Chương I : Tổng quan về CTDL và Giải thuật

Giảng viên: TS. Đỗ Thị Ngọc Diệp
Khoa Kỹ thuật Truyền thông – Trường Điện Điện Tử

ONE LOVE. ONE FUTURE.

1. Mục đích và nội dung của CTDL
2. Các khái niệm cơ bản về CTDL và giải thuật
3. Ngôn ngữ diễn đạt giải thuật
4. Thiết kế và Đánh giá giải thuật

1. Mục đích và nội dung của CTDL

- Mục đích:
 - củng cố và nâng cao kiến thức cơ bản về cấu trúc dữ liệu và giải thuật của ngành khoa học máy tính.
 - Tăng cường khả năng phân tích, thiết kế và cài đặt các chương trình cho máy tính.
 - Nâng cao khả năng tư duy trừu tượng và sự khái quát khi giải quyết các bài toán thực tế bằng máy tính
- Yêu cầu
 - Kiến thức cơ bản về lập trình
 - Thành thạo ít nhất một ngôn ngữ lập trình cơ bản: C, C++,...

1. Mục đích và nội dung của CTDL

- Nội dung:
 - Các phương pháp phân tích và thiết kế một chương trình.
 - Các cấu trúc dữ liệu và các thao tác cơ bản tương ứng
 - Cấu trúc tuyến tính: mảng, danh sách
 - Cấu trúc phi tuyến: cây, đồ thị
 - Các giải thuật
 - Sắp xếp, tìm kiếm,
 - Giải thuật nâng cao : đệ quy, giải thuật trên CTDL cây, đồ thị.

- Bài toán: Viết một chương trình quản lý sinh viên của một lớp. Mỗi sinh viên gồm các thuộc tính: Mã số, Họ tên, Ngày sinh, Địa chỉ, Tên lớp, Môn thi, Điểm thi.

Chương trình cần thực hiện các công việc sau:

- Cập nhật thông tin cho từng sinh viên trong lớp: bổ sung, loại bỏ, hay cập nhật các thuộc tính
- Sắp xếp sinh viên trong lớp theo một trật tự nhất định
- Tìm kiếm một sinh viên theo một tiêu chuẩn nào đó
- In thông tin các sinh viên trong lớp
-

- Phân tích yêu cầu:
 - 2 nhiệm vụ chính trước khi xây dựng được chương trình:
 - Hiểu được cách tổ chức và cài đặt cấu trúc dữ liệu biểu diễn cho *sinh viên và lớp*
 - cần nắm được cấu trúc dữ liệu
 - Hiểu được ý tưởng và cách cài đặt cho các thao tác cơ bản như tìm kiếm, sắp xếp trên cấu trúc dữ liệu đã chọn
 - cần nắm được giải thuật

Program = Data structure + Algorithm

2. Các khái niệm cơ bản về CTDL và giải thuật

- 2.1. Giải thuật (*Algorithms*)
- 2.2. Dữ liệu (*Data*)
 - 2.2.1. Cấu trúc dữ liệu (*Data structures*)
 - 2.2.2. Cấu trúc lưu trữ (*Storage structures*)
- 2.3. Mối liên quan giữa DL và giải thuật (*Relationship between DS and algorithms*)

2.1. Giải thuật (Algorithm)

- **Giải thuật** là một **đặc tả** chính xác và không nhập nhằng về một chuỗi các bước có thể được thực hiện một cách tự động, để cuối cùng ta có thể thu được các kết quả mong muốn.
 - Đặc tả (specification): bản mô tả chi tiết và đầy đủ về một đối tượng hay một vấn đề
- Tại sao phải học về Giải thuật ?
 - Quan trọng đối với tất cả các ngành của khoa học máy tính
 - Ví dụ:
 - Bài toán định tuyến mạng truyền thông trên các thuật toán tìm đường đi ngắn nhất
 - Đồ họa máy tính sử dụng các thuật toán hình học
 - Chỉ số hóa/Tìm kiếm trên cơ sở dữ liệu dựa vào cấu trúc dữ liệu cây tìm kiếm cân bằng
 - Sinh trắc học tính toán sử dụng các thuật toán lập trình động
 - Search engine sử dụng các thuật toán tìm kiếm
 - v.v.

2.1. Giải thuật

- Một số yêu cầu của giải thuật
 - Đúng đắn:
 - với mọi tập dữ liệu vào thì giải thuật phải luôn đưa ra được kết quả đúng như mong đợi
 - Rõ ràng (không nhập nhằng):
 - số bước của giải thuật phải rõ ràng
 - chi tiết từng bước cụ thể: dữ liệu vào/ra/trung gian, thao tác, v.v.
 - Khả dừng:
 - phải kết thúc sau một số hữu hạn bước thực hiện
 - Hiệu quả:
 - Thời gian thực hiện hợp lý
 - Tài nguyên sử dụng (bộ nhớ, phần cứng, v.v.) hợp lý

- **Dữ liệu (data):**

- Là các đối tượng mà thuật toán sẽ sử dụng để đạt được kết quả mong muốn.
- Được dùng để biểu diễn cho các thông tin của bài toán
 - Thông tin vào
 - Thông tin ra (kết quả)
 - Thông tin trung gian (nếu cần)

2.2. Dữ liệu

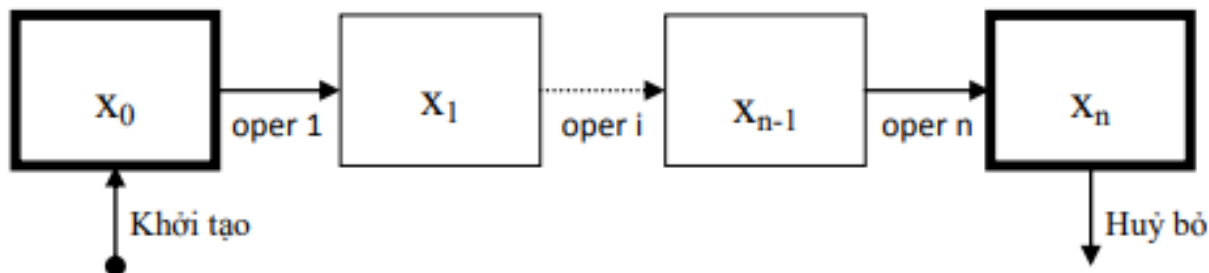
- Dữ liệu gồm có hai mặt:
 - **Mặt tĩnh** (static):
 - Kiểu dữ liệu (data type)
 - Cách tổ chức dữ liệu
 - Miền giá trị của dữ liệu
 - **Mặt động** (dynamic):
 - Trạng thái của dữ liệu
 - Nếu dữ liệu đang tồn tại thì mặt động của nó còn thể hiện ở giá trị cụ thể của dữ liệu tại từng thời điểm.
 - Trạng thái hay giá trị của dữ liệu sẽ bị thay đổi khi xuất hiện những sự kiện, thao tác tác động lên nó.
 - Dữ liệu có thể tồn tại hay không tồn tại, sẵn sàng hay không sẵn sàng

2.2. Dữ liệu

- Hai mặt tĩnh và động của dữ liệu có quan hệ chặt chẽ với nhau, không thể tách rời

- Ví dụ:

```
int n ;  
n = x0;  
n = x1;  
...
```

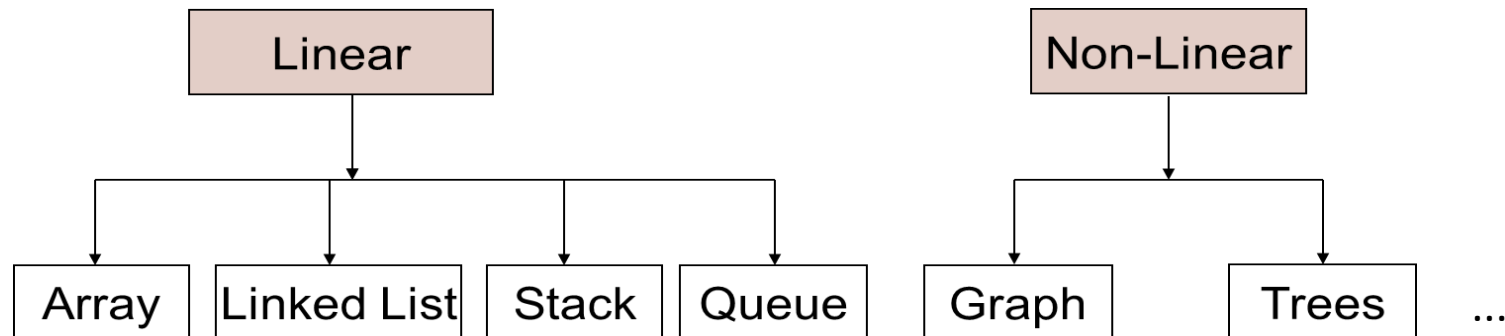


2.2.1. Cấu trúc dữ liệu

- Dữ liệu cơ bản:
 - Dữ liệu vô hướng, dữ liệu đơn.
 - Số nguyên, số thực, ký tự, con trỏ
- **Dữ liệu có cấu trúc:**
 - Là kiểu dữ liệu mà bên trong nó có chứa nhiều thành phần dữ liệu đơn/có cấu trúc và các thành phần dữ liệu này được tổ chức theo một cấu trúc nào đó.
 - Dùng để biểu diễn cho các *thông tin có cấu trúc* của bài toán.
 - Cấu trúc dữ liệu thể hiện khía cạnh logic của dữ liệu.

2.2.1. Cấu trúc dữ liệu

- Dữ liệu có cấu trúc :
 - **Cấu trúc tuyến tính:** các phần tử bên trong nó luôn được bố trí theo một trật tự tuyến tính hay trật tự trước sau.
 - Đây là loại cấu trúc dữ liệu đơn giản nhất.
 - Ví dụ: mảng, danh sách.
 - **Cấu trúc phi tuyến:** các thành phần bên trong không được bố trí theo trật tự tuyến tính mà theo các cấu trúc khác.
 - Ví dụ: tập hợp (không có trật tự), cấu trúc cây (cấu trúc phân cấp), đồ thị (cấu trúc đa hướng).

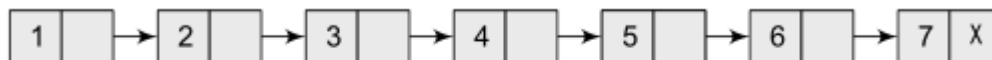


Một số cấu trúc dữ liệu cơ bản

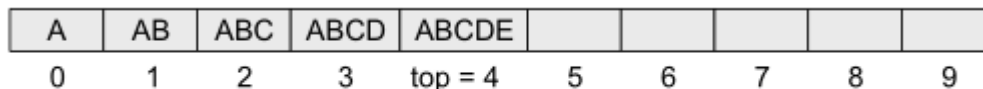
1 st element	2 nd element	3 rd element	4 th element	5 th element	6 th element	7 th element	8 th element	9 th element	10 th element
----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	----------------------------	-----------------------------

marks[0] marks[1] marks[2] marks[3] marks[4] marks[5] marks[6] marks[7] marks[8] marks[9]

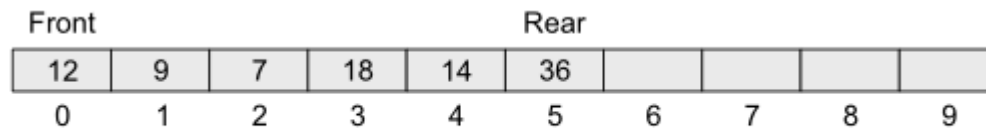
Memory representation of an array of 10 elements



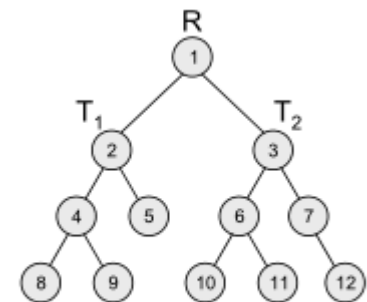
Simple linked list



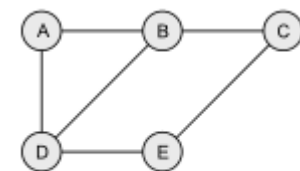
Array representation of a stack



Array representation of a queue



Binary tree



Graph

- Khởi tạo, Hủy
- Chèn mới
- Sửa đổi phần tử
- Xóa phần tử
- Tìm kiếm
- Duyệt phần tử
- Sắp xếp
- Trộn

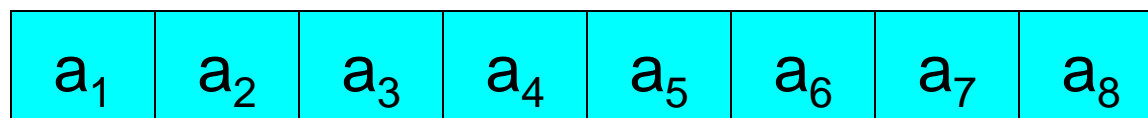
2.2.2. Cấu trúc lưu trữ

- Cấu trúc lưu trữ của một cấu trúc dữ liệu thể hiện khía cạnh vật lý (cài đặt) của cấu trúc dữ liệu đó.
 - cách tổ chức bộ nhớ của máy tính để có thể lưu trữ cho một cấu trúc dữ liệu
- Hoặc là cấu trúc kiểu dữ liệu mà một ngôn ngữ lập trình hỗ trợ
 - số lượng các cấu trúc lưu trữ là số lượng các kiểu dữ liệu của ngôn ngữ lập trình đó

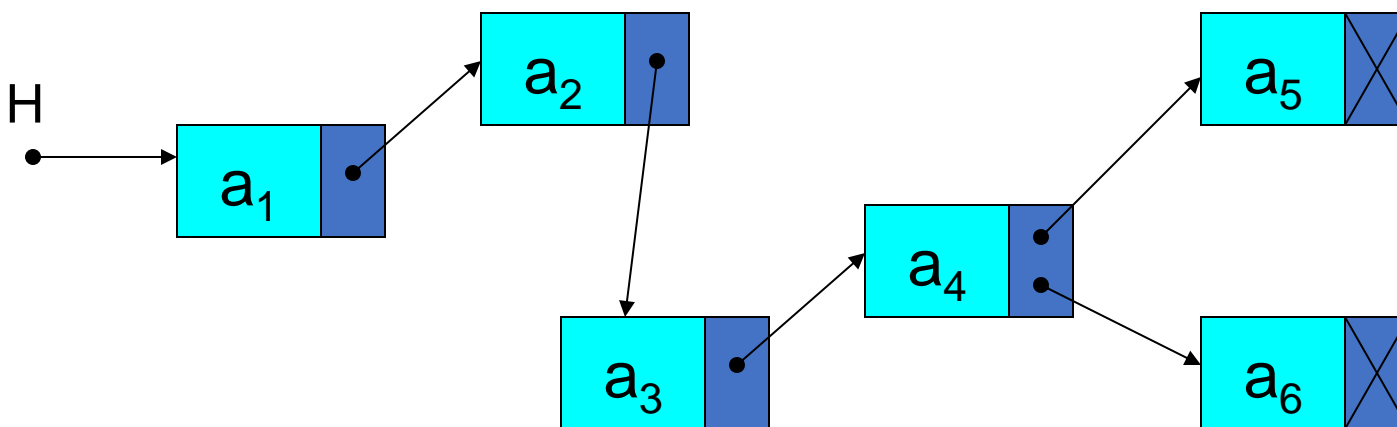
- Vị trí lưu trữ:
 - **Cấu trúc lưu trữ trong:**
 - nằm ở bộ nhớ trong (bộ nhớ chính) của máy tính.
 - Đơn giản, dễ tổ chức và tốc độ thao tác rất nhanh.
 - Không có tính lưu tồn (persistence), kích thước khá hạn chế.
 - **Cấu trúc lưu trữ ngoài:**
 - nằm ở bộ nhớ ngoài (bộ nhớ phụ).
 - Cấu trúc phức tạp và tốc độ thao tác chậm
 - Có tính lưu tồn và cho phép lưu trữ các dữ liệu có kích thước rất lớn.

2.3.1. Cấu trúc lưu trữ trong

- Minh họa 2 loại CTLT trong



Cấu trúc tĩnh



Cấu trúc động

2.3.1. Cấu trúc lưu trữ trong

- Cách thức tổ chức:

- **Cấu trúc lưu trữ tĩnh (CTLT tuần tự):**

- Các ngăn nhớ đứng liền kề nhau thành một dãy liên tục trong bộ nhớ
 - Số lượng và kích thước mỗi ngăn là cố định => kích thước dữ liệu luôn cố định
 - Có thể truy nhập trực tiếp vào từng ngăn nhờ chỉ số => tốc độ truy nhập vào các ngăn là đồng đều
 - Dùng cho các CTDL tuyến tính có kích thước cố định hoặc ít thay đổi như cấu trúc mảng, danh sách nhỏ

- **Cấu trúc lưu trữ động (cấu trúc liên kết hay móc nối):**

- Chiếm các ngăn nhớ thường không liên tục
 - Số lượng và kích thước các ngăn có thể thay đổi trong khi chạy chương trình
 - Việc truy nhập trực tiếp vào từng ngăn rất hạn chế, mà thường sử dụng cách truy nhập tuần tự, bắt đầu từ một phần tử đầu, rồi truy nhập lần lượt qua các con trỏ móc nối (liên kết)
 - Dùng cho các cấu trúc dữ liệu phi tuyến, hay cấu trúc dữ liệu tuyến tính nhưng có kích thước luôn thay đổi như cấu trúc danh sách.

2.4. Mối liên quan giữa DL và giải thuật

- Mối quan hệ khăng khít

- Giải thuật đặc tả các thao tác được thực hiện trên một loại CTDL nhất định ... CTDL thay đổi, thì giải thuật cũng phải có những thay đổi cho phù hợp.
- Một giải thuật thường chỉ thích hợp với một hay một số CTDL. Không có giải thuật nào thích hợp cho tất cả các loại CTDL.
- Sử dụng CTDL luôn gắn liền với các giải thuật trên CTDL đó.

3. Ngôn ngữ diễn đạt giải thuật

- 3.1. Nguyên tắc khi sử dụng ngôn ngữ
- 3.2. Các loại ngôn ngữ diễn đạt giải thuật

3.1. Nguyên tắc khi sử dụng ngôn ngữ

Ngôn ngữ diễn đạt = cách thể hiện giải thuật

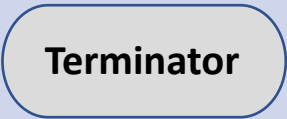
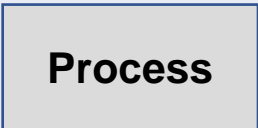


- **Tính độc lập của giải thuật:**
 - Ngôn ngữ được chọn phải làm sáng tỏ tinh thần của giải thuật, giúp người đọc dễ dàng hiểu được logic của giải thuật.
 - VD: ngôn ngữ tự nhiên và ngôn ngữ hình thức (như các lưu đồ thuật toán, các ký hiệu toán học).
- **Tính có thể cài đặt được của giải thuật:**
 - Ngôn ngữ được chọn phải thể hiện được khả năng có thể lập trình được của giải thuật, và giúp người đọc dễ dàng chuyển từ mô tả giải thuật thành chương trình
 - VD: ngôn ngữ lập trình





!

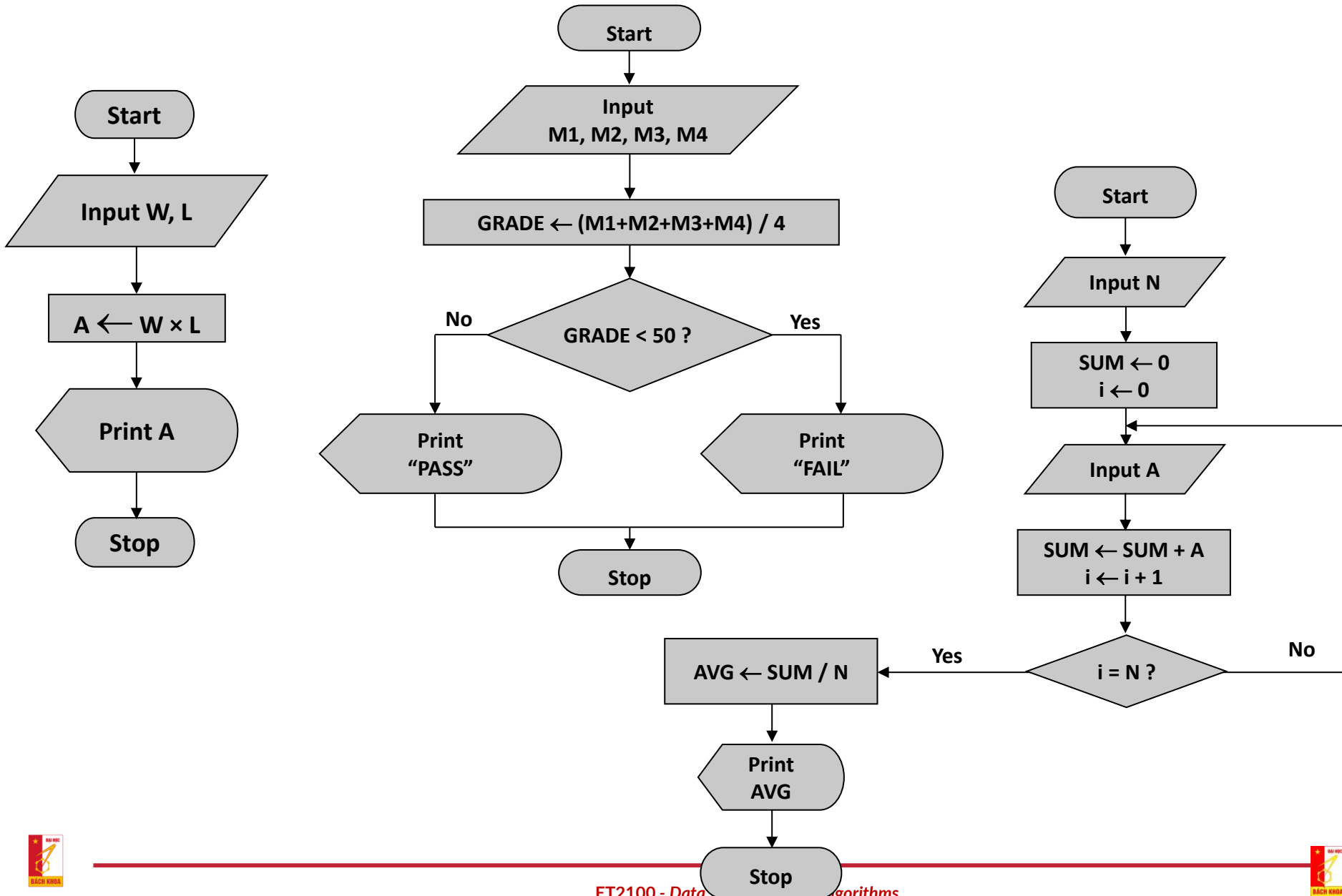
3.2. Các loại ngôn ngữ diễn đạt giải thuật

- Ngôn ngữ tự nhiên
 - + : dễ đọc, dễ viết, dễ hiểu, khả năng biểu diễn không hạn chế.
 - - : dài dòng, không đủ rõ ràng để cài đặt được giải thuật.
- Lưu đồ giải thuật: Sử dụng các hình vẽ, biểu tượng để biểu diễn cho các thao tác của giải thuật
 - + : rõ ràng các bước, các thao tác, khả năng có thể lập trình
 - - : chú trọng vào chức năng hơn là dữ liệu, không thích hợp với giải thuật phức tạp
- Ngôn ngữ lập trình
 - + : cú pháp rõ ràng, giúp hiểu sâu về giải thuật và cách cài đặt
 - - : cồng kềnh, chi tiết, cụ thể

=> Có thể Sử dụng NNLT dạng giản lược để diễn đạt giải thuật
(Ngôn ngữ tựa lập trình/ giả mã/ pseudo code)

Element	Description
 Terminator	Beginning or end of a process
 Process	A step (task, action) in a process
 Arrow	Directional execution flow
 Decision	Conditional decision, this-or-that choice branch

Element	Description
 Data	Process input/output
 Display	Displayed output
 On-page connector	Flow continues at a target on the same chart (to avoid long arrows)
 Off-page connector	Flow continues at a target on another page



- Ngôn ngữ nhân tạo và không chính thức giúp lập trình viên phát triển các thuật toán
- Có thể là sự kết hợp của ngôn ngữ máy tính và ngôn ngữ nói
- Giả mã thường gồm:
 - Khai báo kiểu mới
 - Khai báo đối tượng dữ liệu
 - Truy cập các thành phần đối tượng dữ liệu
 - Định nghĩa các thao tác trên kiểu dữ liệu
 - Các thao tác sử dụng, gọi, v.v.

Ngôn ngữ tự lập trình (giả mã, pseudo code)

- Thao tác:

Assignment

variable \leftarrow expression
set variable to value

if-then

if condition then
statement

if-then-else

if condition then
statement-#1
else statement-#2

Case-of

case expression of
condition-#1: statement-#1
condition-#2: statement-#2
...
condition-#n: statement-#n
others: default-statement

Loop

repeat

statement
until condition

while condition do
statement

for each element in list do
statement

for variable \leftarrow v1 [to|down to] v2 do
statement

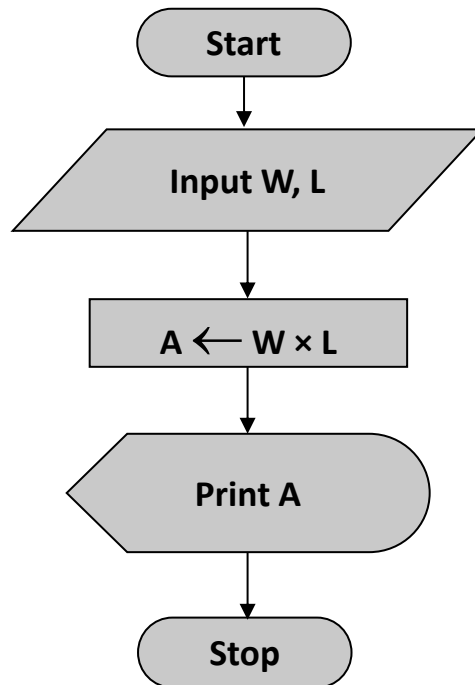
Function Call

call proc

call proc with parameters

call func with parameters returning value

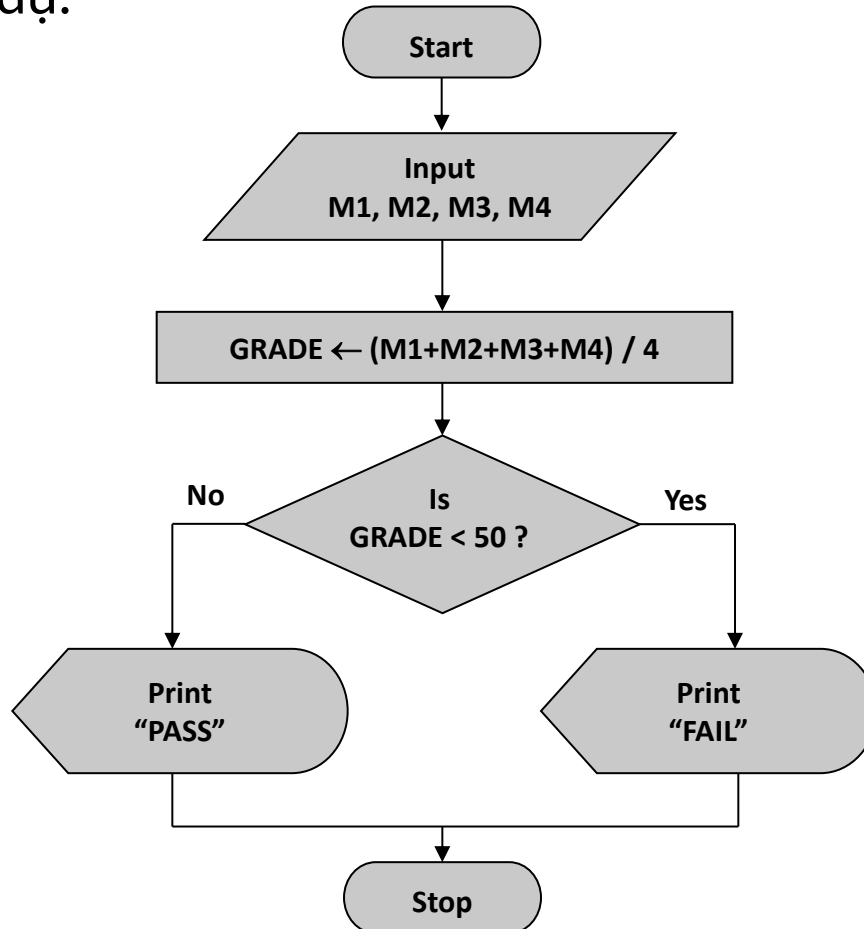
- Ví dụ:



Input W, L
 $A \leftarrow W \times L$
Print A

Ngôn ngữ tựa lập trình (giả mã, pseudo code)

- Ví dụ:



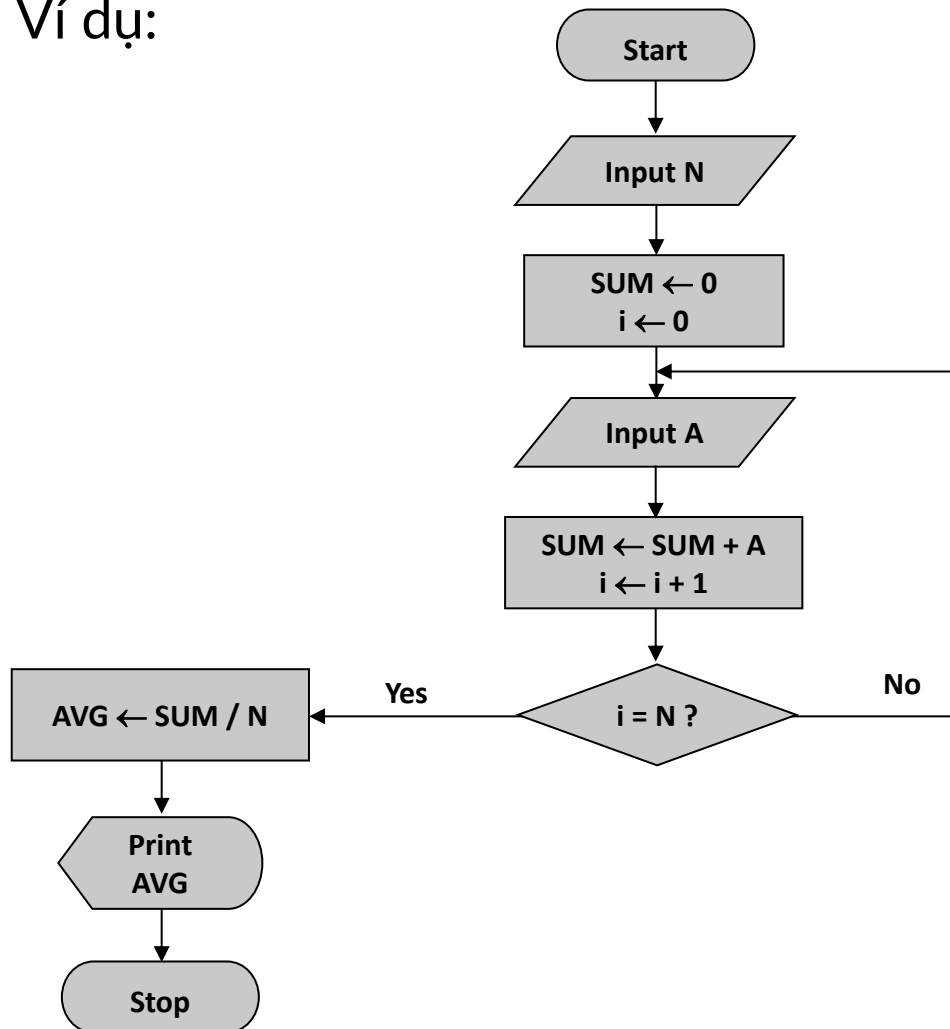
Input $M1, M2, M3, M4$

$GRADE \leftarrow (M1 + M2 + M3 + M4) / 4$

if $GRADE < 50$ then
 Print "FAIL"
else
 Print "PASS"

Ngôn ngữ tự lập trình (giả mã, pseudo code)

- Ví dụ:



Input N

$SUM \leftarrow 0$

$i \leftarrow 0$

repeat

Input A

$SUM \leftarrow SUM + A$

$i \leftarrow i + 1$

until $i = N$

$AVG \leftarrow SUM / N$

Print AVG

- Vẽ lưu đồ giải thuật giải phương trình: $ax^2 + bx + c = 0$. Biểu diễn lại lưu đồ dưới dạng giả mã.
- Cho một mảng gồm N số nguyên, hãy vẽ lưu đồ giải thuật in tất cả các số chẵn. Biểu diễn lại lưu đồ dưới dạng giả mã.
- Vẽ lưu đồ giải thuật bài toán tính cước phí taxi theo khoảng cách và thời gian chờ, với quy ước:
 - Cước phí = cước theo khoảng cách + cước theo thời gian chờ
 - 2km đầu tiên: \$5
 - Mỗi 0.1km tiếp theo: \$0.15
 - 30 phút chờ đầu tiên: \$3
 - Mỗi 10 phút chờ tiếp theo: \$0.5
 - Biểu diễn lại lưu đồ dưới dạng giả mã.