Data structure and Algorithms

Thanh-Hai Tran

Electronics and Computer Engineering
School of Electronics and Telecommunications

Hanoi University of Science and Technology
1 Dai Co Viet - Hanoi - Vietnam

Outline

- Objectives and contents of the course
- Introduction to data structures and algorithms
- Algorithm flowchart

Objectives

- To provide basic knowledge of data structures and algorithms
- To improve abilities of design, analysis and implementation of computer programs.
- To improve abstraction and generalization thoughts in resolving real problems by computers.

Contents of the course

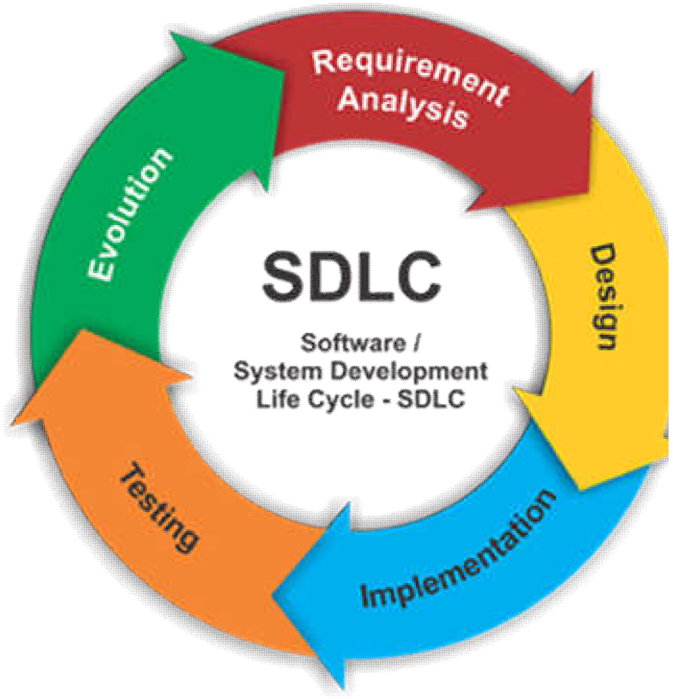
- **Basic data types**
  - Structures, arrays, strings, pointers, files,…
- **Data structures**
  - Linked lists
  - Stacks, queues
  - Trees, Graphs
- **Algorithms**
  - Sorting
  - Searching
  - Hashing
  - Mapping
  - String pattern matching

Example

- Requirements: Writing a program managing list of students in a class. Each student has attributes such as: ID, full name, dob, address, class name, subject, marks.
- The program needs to do the following operations:
  - Updating the information of each student. It means that each attribute can be inserted, removed or updated its value.
  - Sorting the list by some order (like full-name or ID)
  - Searching students in the list by some conditions (like full-name, marks)
  - Printing the list
  - ….

SDLC Model

- **A framework that describes the activities performed at each stage of a software development project**
- **Requirement analysis:**
  - Understand organization and implementation method for student list structure → Understand data structures
  - Understand ideas and implementation methods for operations like sorting, searching → Understand algorithms



Outline

- Objectives and contents of the course
- Introduction to data structures and algorithms
- Algorithm flowchart

Data

- **Data:** Objects used in algorithms to describe the information of related problems (like input, expected output), and storage of intermediate results.
- **Data have two perspectives:**
  - **Static** (Mặt tĩnh): data type that defines way data is organized and its value domain.
  - **Dynamic** (Mặt động): defining states of data such as existent or not, ready or not, current value or value at some time. State of data changes when some action or event happens.

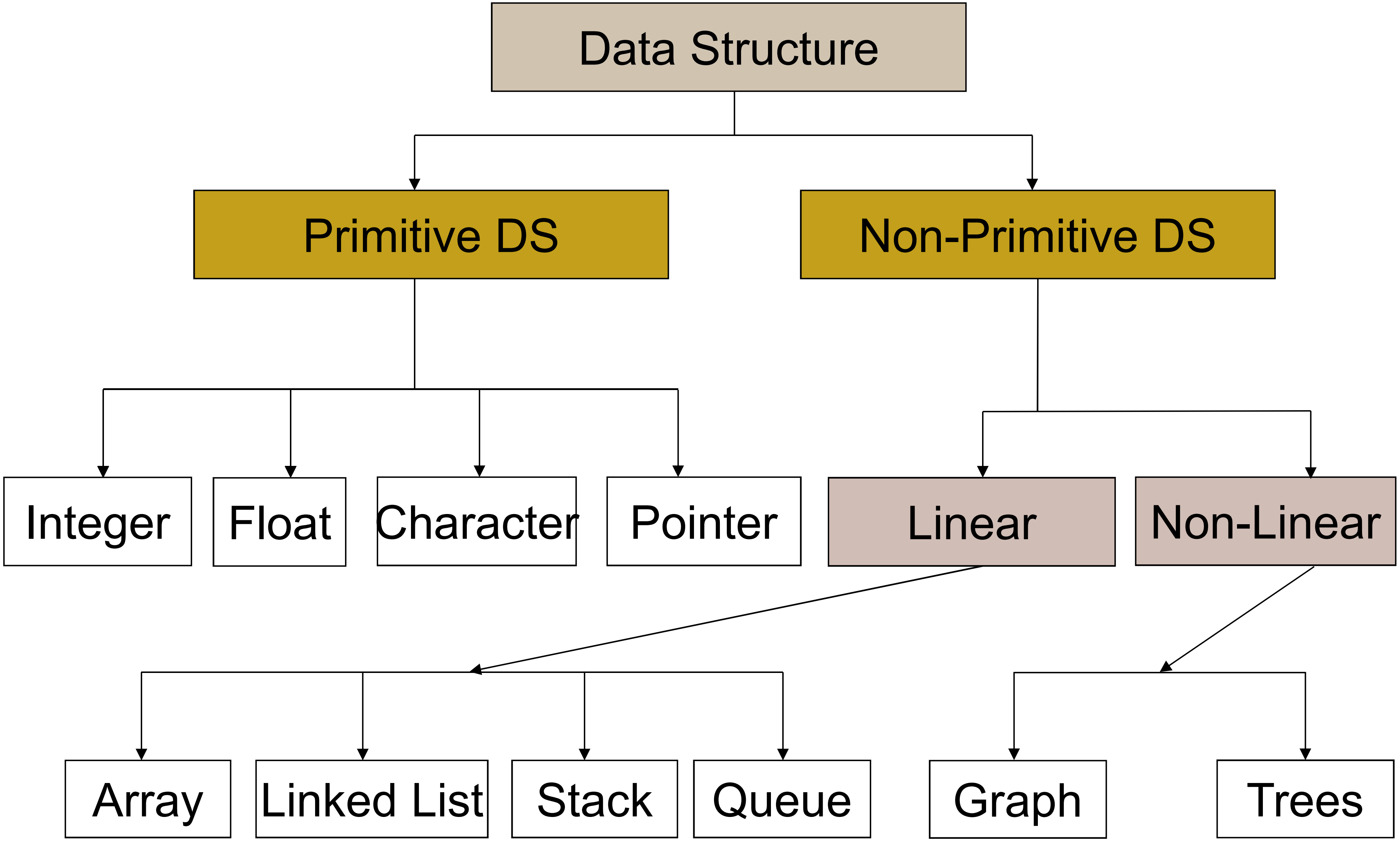
Data structure

- **Data structure (Cấu trúc dữ liệu) :** A data type which contains another data organized by some structure.
- **Scalar (dữ liệu vô hướng) or Simple Data Types (dữ liệu đơn giản):** opposite to data structures. For example: integer, real, logic (boolean).
- **There are 2 classes:**
  - **Linear structures** (Cấu trúc tuyến tính): its components are organized by linear order (predecessor-successor). It is also called **simple structure**.

    For example: **arrays**, **lists**.
  - **Non-linear structures** (Cấu trúc phi tuyến): its components are organized by non-linear order.

    For example: **set** (no order), **trees** (hierarchical structure), **graphs** (network structure).

Data structure classification



Storage structures (Cấu trúc lưu trữ)

- Storage structure of a data structure is a way to organize and implement for the data structure in some computer program.
- In principle, storage structures is related closely to memory storage organization in computers.
- However in reality, a storage structure of a data structure is usually other data structure (lower level data structure) supported by the programming language used to implement the data structure.
- *For example*: the data structure array can be used to implement the list in many programming languages.

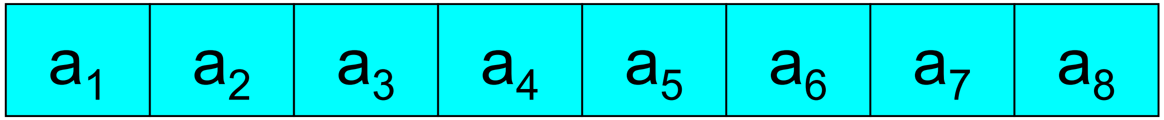
Storage structures

- In computers, there are 2 types of storage structures
- **Internal storage structures** (Cấu trúc lưu trữ trong): located in internal memories of computers (also called primary memories like RAM, ROM). (The course focuses only on these storage structures)
  - *Advantages*: simple structure, quick access speed
  - *Disadvantages*: not persistent, limited storage space, high costs
- **External storage structures** (Cấu trúc lưu trữ ngoài): located in external memories (also called secondary memories like HDD, SSD, CD, …).
  - *Advantages*: persistent, large storage space, low costs
  - *Disadvantages*: complicated structures, low access speed

Internal storage structure

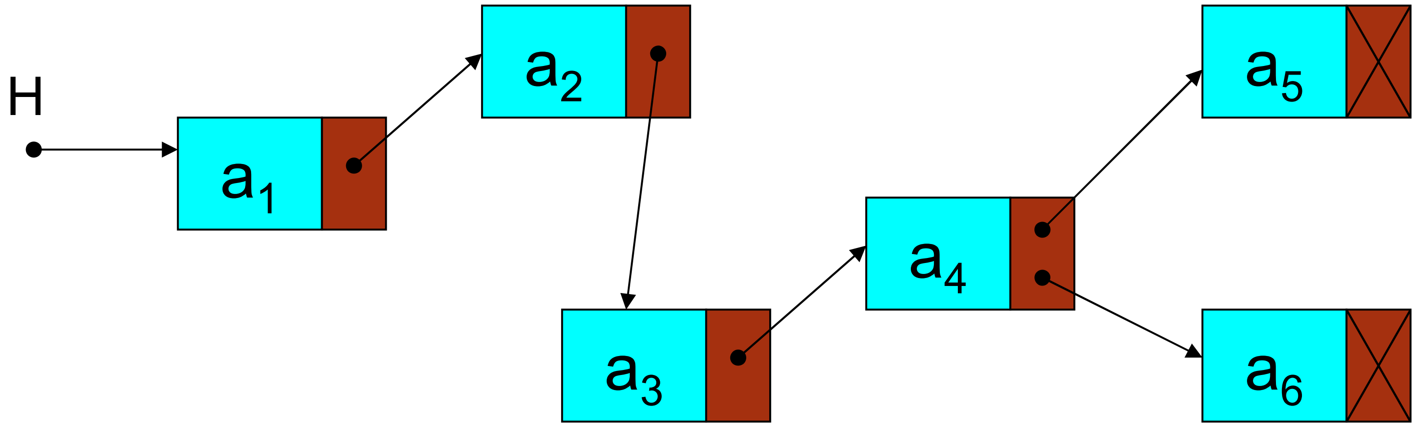
- **Static storage structure (Cấu trúc lưu trữ tĩnh):** it usually has fixed (static) space, and also called *sequential storage structure*.
- **Dynamic storage structure (Cấu trúc lưu trữ động):**
  - It has changeable (dynamic) space
  - Using self-reference (linked) and dynamic allocation of memory.

Example: two types of ISS



a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8

a. Static storage structure



b. Dynamic storage structure

(linked structure)

Characteristics of ISS

- **Static storage structure:**
  - Consisting of memory cells that adjacent to each others → sequential structure of cells
  - Fixed number of cells and fixed size of each cell
  - Each cell can be accessed directly by its index → quick and identical access speed for each cell

- **Dynamic storage structure:**
  - Consisting of memory cells that are normally not adjacent
  - Variable number of cells and variable size of each cell
  - The number of cells that can be accessed directly is limited (only one or two terminal cells). Most of cells mus be accessed sequentially (one by one access).

Steps of building a data structure

- **Step 1: identify all characteristics of the DS such as:**
  - Data elements in the DS,
  - Relationships among data elements.
- **Step 2: identify basic operations of the DS.**
- **Step 3: identify suitable storage structure for the DS, so that the implementation of the DS is efficient in both aspects: runtime and used memory space.**
- **Step 4: Implementation of basic operations by following principles:**
  - Reuse: using functions/procedures.
  - Independence as much as possible: choosing suitable parameters.
  - Efficiency: try to optimize codes.

Algorithm

- **Definition:** is a clear and unambiguous specification about a sequence of steps that can be run automatically on computers, in order to achieve expected results.
- **Specification:** a detail description about some object or problem.
- **Requirements for algorithms**
  - Truth (đúng đắn),
  - Unambiguity (rõ ràng, không nhập nhằng),
  - Terminal (finished after finite steps)
  - Description about used data such as input data, output data and possible intermediate data,
  - Reasonable running time.

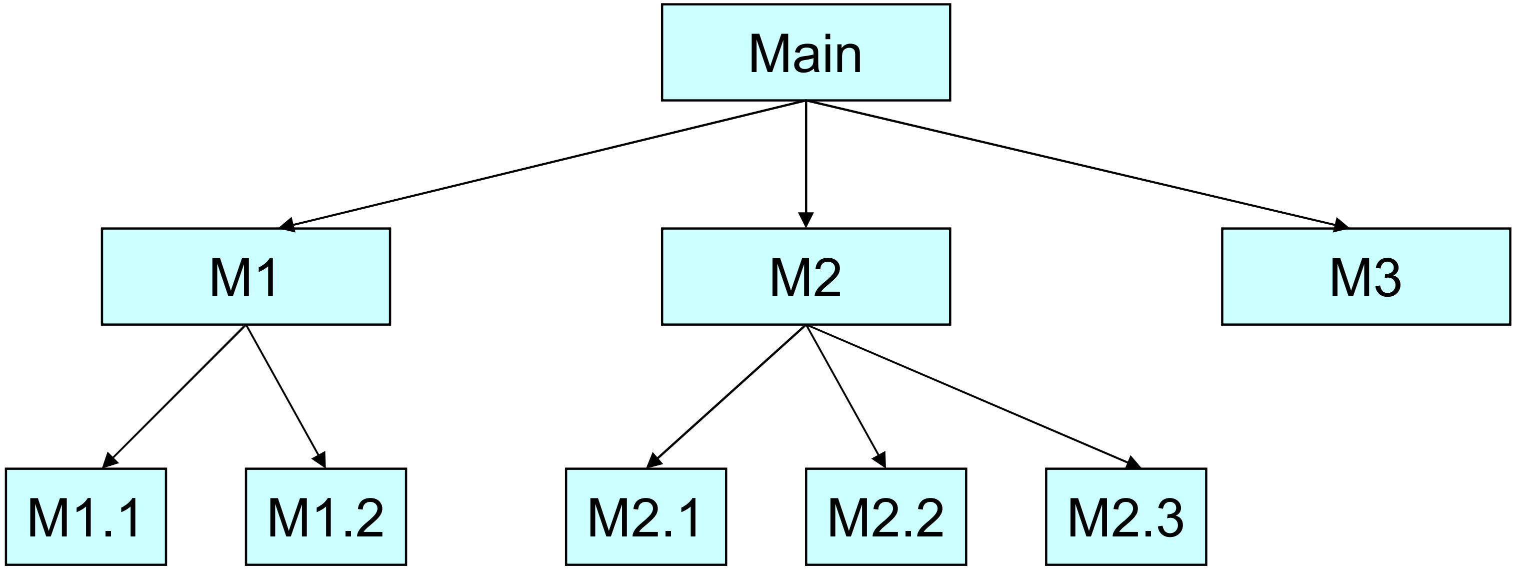
Design and Analysis of algorithms

- **Design of algorithms:** The process of transforming specification of algorithm into structure of the program that implements the algorithm
- In general, it includes two steps (phases):
  - **General design (Thiết kế sơ bộ):** this step needs to identify clearly components (also called modules) of the algorithm. The method is normally used in this step is top-down design which helps to identify functionalities of each module, and their relationships.
  - **Detail design (Thiết kế chi tiết):** this step begins to implement (coding) modules, one by one. Then all implementations need to be combined into a complete program. This step usually uses the design method called stepwise refinement method (phương pháp *tinh chỉnh từng bước*).

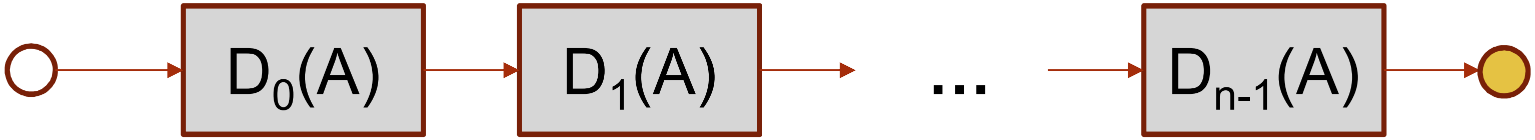
Top-down design method

- Also called *modularization* (mô dul hóa), based on *divide and conquer* (chia để trị) principle, original algorithm will be divided into modules (sub-algorithms, sub-modules), each one will take a task of the original algorithm (including many tasks).
- The process may be repeated for modules until all generated modules are small enough to resolve all tasks.
- The finish of the process will create *functional hierarchy diagram* (sơ đồ phân cấp chức năng)

Functional Hierarchy Diagram



Stepwise refinement method



D_0(A): first description of algorithm A (in natural language or algorithm diagam)

D_1(A), D_2(A), …, D_{n-2}(A): intermediate descriptions of A

D_{n-1}(A): final description is a complete program in some programming language

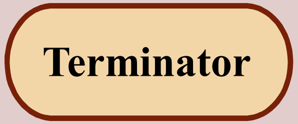
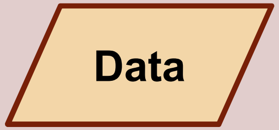
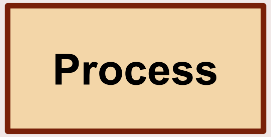
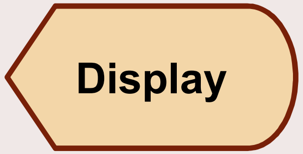
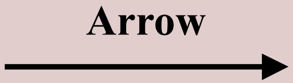
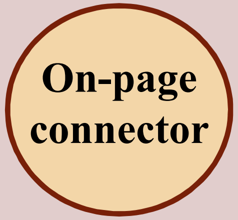
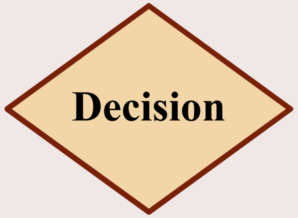
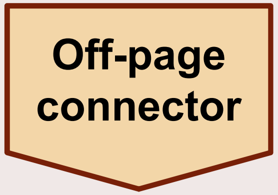
Outline

- Objectives and contents of the course
- Introduction to data structures and algorithms
- Algorithm flowchart

Flowchart

- A graphical representation of the sequence of operations in an algorithm
- Show the sequence of instructions in a program
- Helps visualize and understand how a program works

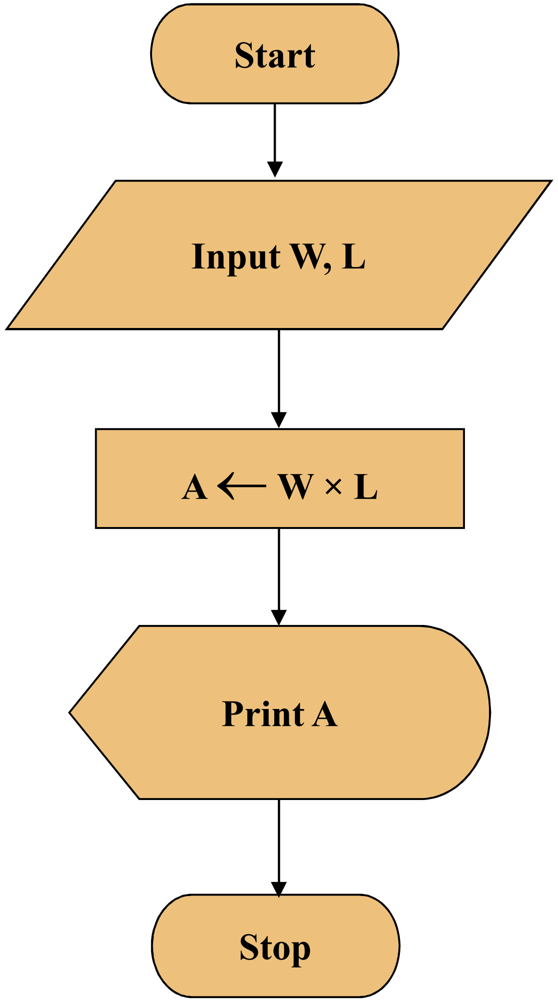
Basic Elements (*)

Element	Description	Element	Description
 Terminator	Beginning or end of a process	 Data	Process input/output
 Process	A step (task, action) in a process	 Display	Displayed output
 Arrow	Directional execution flow	 On-page connector	Flow continues at a target on the same chart (to avoid long arrows)
 Decision	Conditional decision, this-or-that choice branch	 Off-page connector	Flow continues at a target on another page

(*) ANSI/ISO compliant

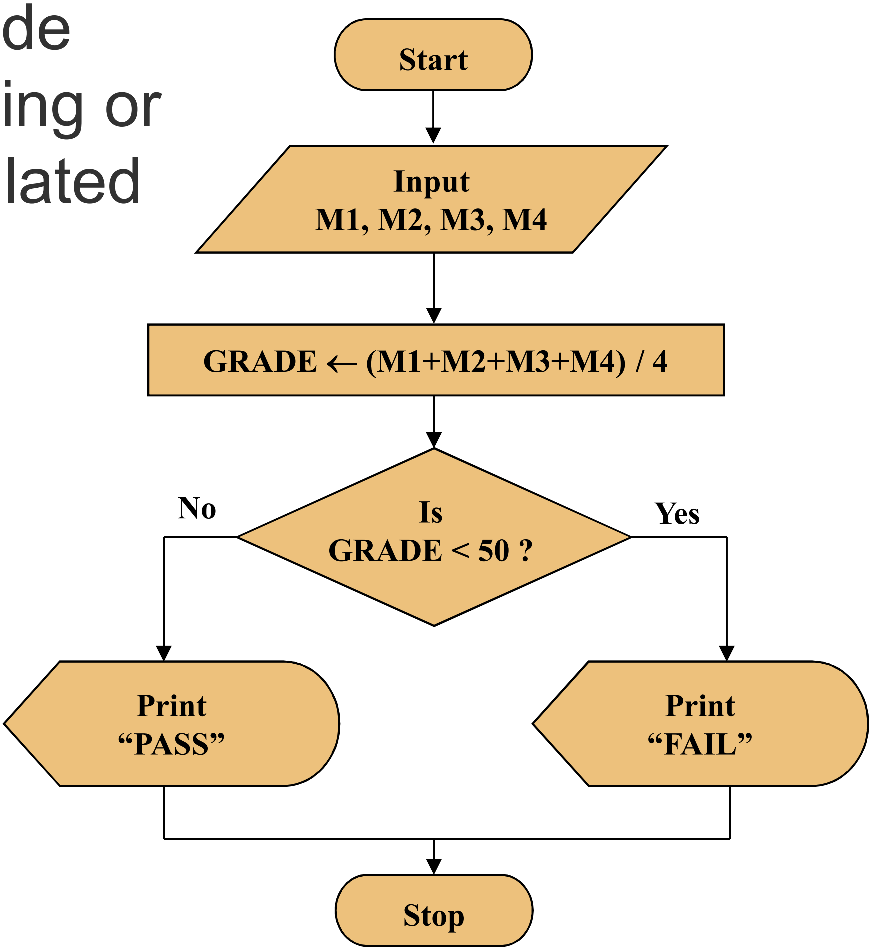
Example 1

- **Requirement:** Write an algorithm and draw a flowchart that will read the two sides of a rectangle and calculate its area.
- **Input: W, L**
- **Output: A (area)**
- **Steps:**
  - Enter W
  - Enter L
  - Compute A = L x W
  - Display A



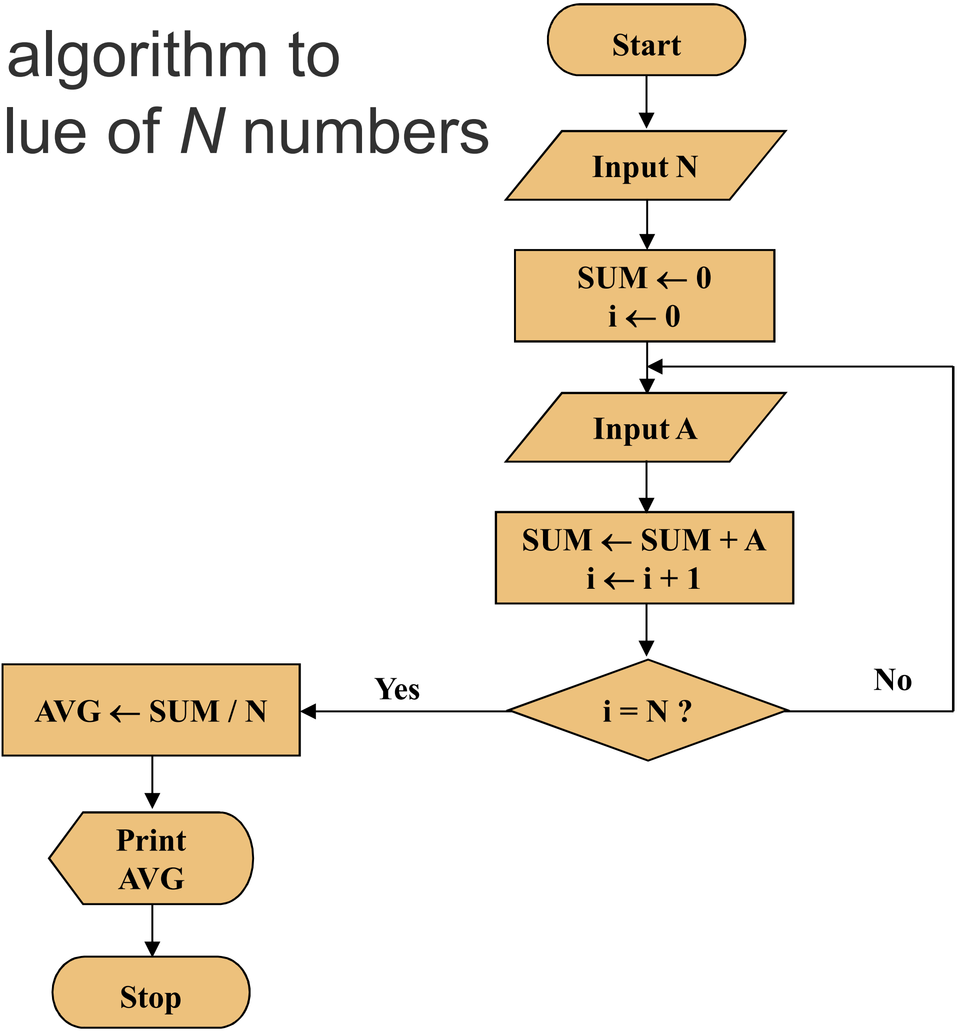
Example 2: Conditional Branch

- **Requirement:** Write an algorithm to determine a student’s final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks.
- Input: M1, M2, M3, M4
- Output: FAIL/PASS
- Steps:
    - Enter M1, M2, M3, M4
    - Compute GRADE = (M1+M2+M3+M4)/4
    - Check if GRADE < 50
        - Yes: FAIL => Display FAIL
        - No: PASS => Display PASS

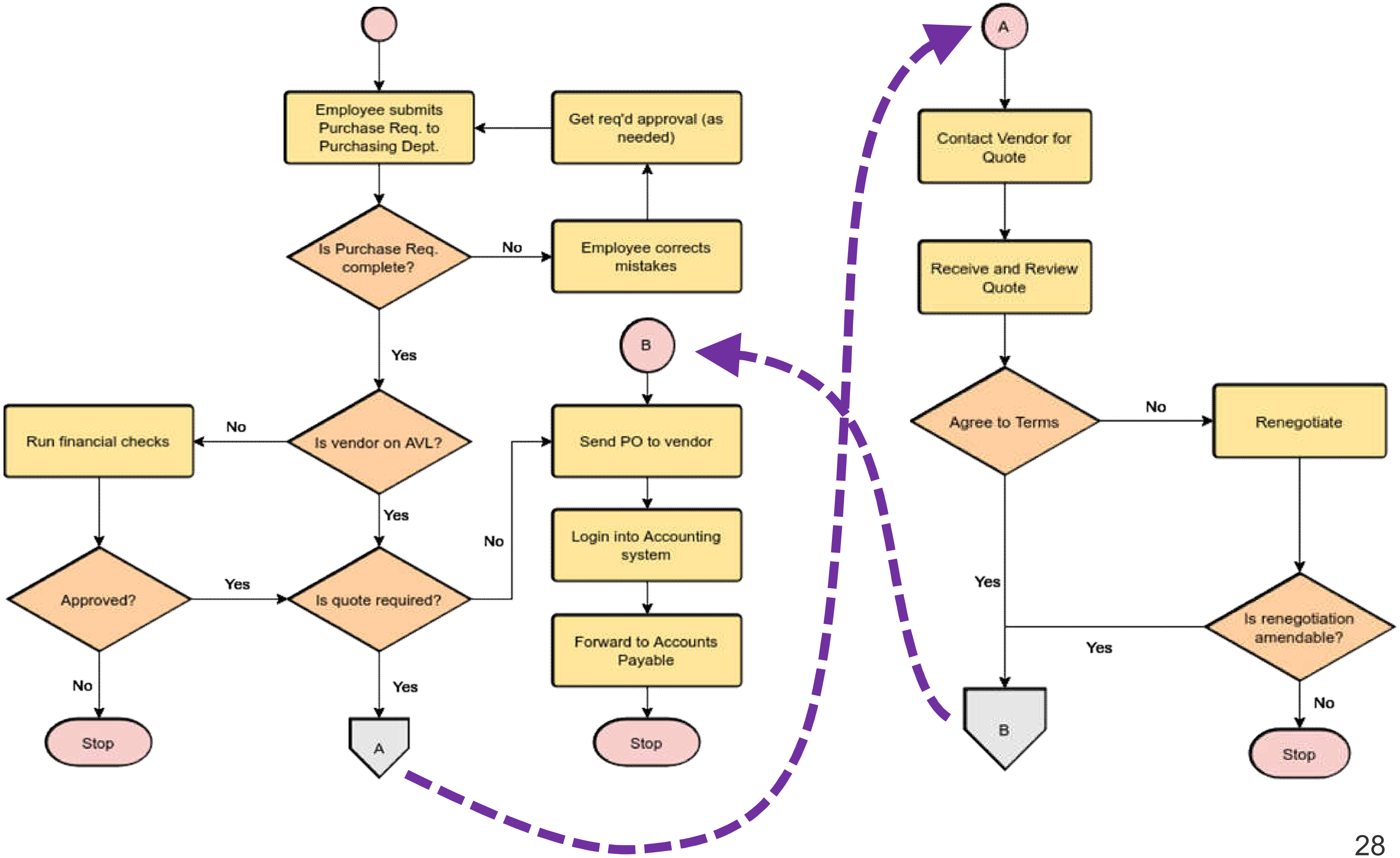


Example 3: Loop

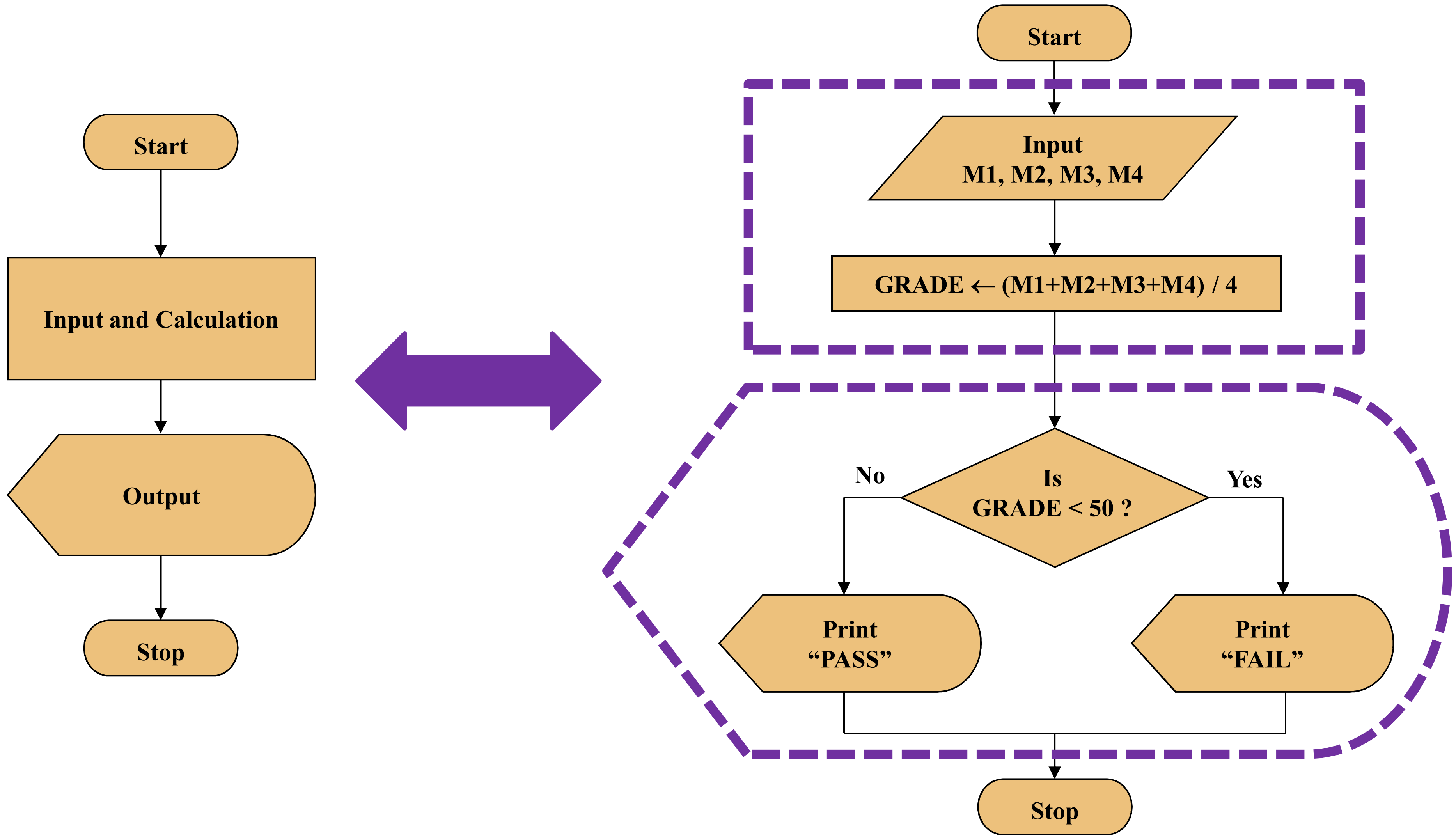
- Requirement: Write an algorithm to determine the mean value of *N* numbers entered by user.
- Input: N
- Output: AVG
- Steps:
  - Enter N
  - SUM = 0; I = 0;
  - **DO**
    - Enter A
    - SUM = SUM + A
    - I = I + 1
  - **UNTIL** I = N
  - AVG = SUM / N



Linking Flowcharts


Employee submits Purchase Req. to Purchasing Dept.
Get req'd approval (as needed)
Is Purchase Req. complete?
No
Employee corrects mistakes
Yes
B
Run financial checks
No
Is vendor on AVL?
Send PO to vendor
Yes
No
Login into Accounting system
Approved?
Yes
Is quote required?
Forward to Accounts Payable
No
Yes
Stop
A
Stop
A
Contact Vendor for Quote
Receive and Review Quote
Agree to Terms
No
Renegotiate
Yes
Is renegotiation amendable?
Yes
B
No
Stop

General and Detailed Flowcharts



Exercises

- Make a flowchart to show how to solve quadratic equations: $ax^2 + bx + c = 0$.
- Given a array of *N* integers, make a flowchart to print all even numbers.
- Make a flowchart to compute the sum of all odd number
- Make a flowchart to check if a number is a prime