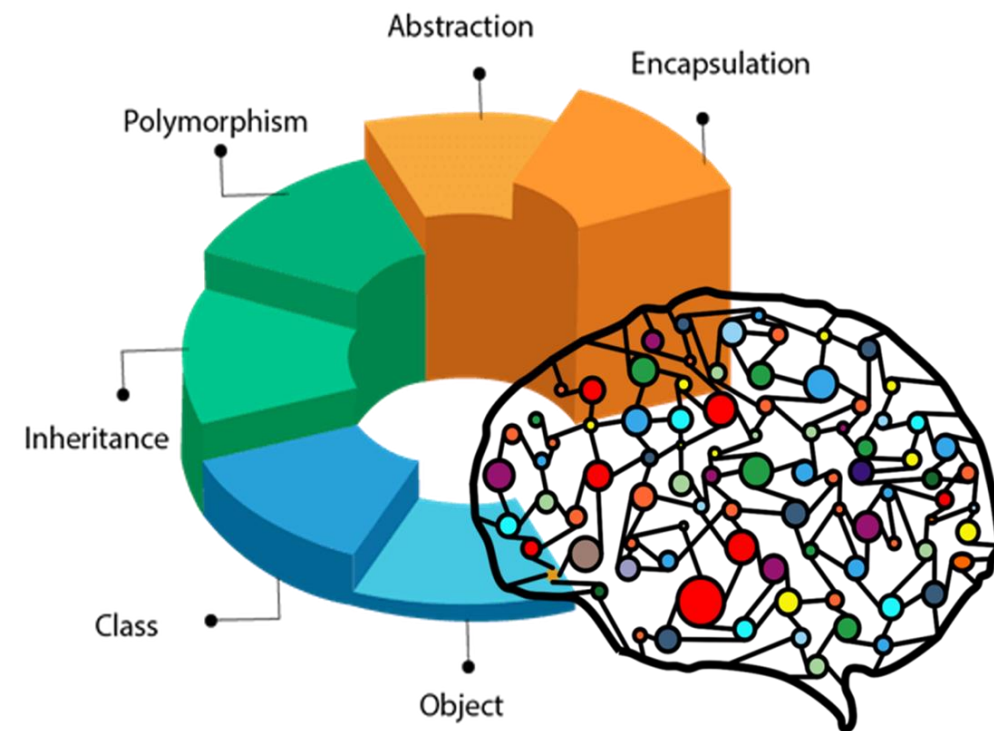


KỸ THUẬT LẬP TRÌNH

ĐẶNG VĂN NGHĨA
0975079414
nghiadv@donga.edu.vn

1. Khái niệm kỹ thuật lập trình
2. Tổng quan về lập trình
3. Mô thức lập trình
4. Chu trình phát triển phần mềm



1. KHÁI NIỆM KỸ THUẬT LẬP TRÌNH



“Kỹ thuật lập trình là kỹ thuật thực thi một giải pháp phần mềm (cấu trúc dữ liệu + giải thuật) dựa trên nền tảng một phương pháp luận (methodology) và một hoặc nhiều ngôn ngữ lập trình phù hợp với yêu cầu đặc thù của ứng dụng”

1. KHÁI NIỆM KỸ THUẬT LẬP TRÌNH

❖ Kỹ thuật lập trình

Algorithms + Data Structures = Programs

= Tư tưởng thiết kế + Kỹ thuật mã hóa

= Cấu trúc dữ liệu + Giải thuật + Ngôn ngữ lập trình

➤ Kỹ thuật lập trình \neq Phương pháp phân tích & thiết kế (A&D)

❖ Chương trình (Programs)

= Giải thuật + Cấu trúc dữ liệu

= Algorithms + Data Structures

❖ Chương trình máy tính

- Là một tập hợp những câu lệnh được viết bằng một ngôn ngữ lập trình theo một trật tự xác định nhằm tự động thực hiện một số nhiệm vụ hoặc chức năng hoặc giải quyết một bài toán nào đó.
- Là một biểu hiện cụ thể của một giải thuật trên một ngôn ngữ lập trình nào đó cùng với những mô tả về cấu trúc dữ liệu mô tả đầu vào, đầu ra của bài toán cần giải quyết.

1. KHÁI NIỆM KỸ THUẬT LẬP TRÌNH

❖ Thế nào là lập trình?

- Viết chương trình tính giai thừa của 10
- Viết chương trình in ra 100 số tự nhiên
- Giải bài toán cổ: “Một người cha có ba mươi sáu con, bố lại cho tròn, và anh chẵn”

KHÔNG PHẢI LÀ LẬP TRÌNH

❖ Thế nào là lập trình?

- Viết chương trình tính giai thừa
- Viết chương trình in ra **n** số nguyên tố đầu tiên
- Giải bài toán cổ “Vừa gà vừa chó, gồm có **X** con, bó lại cho tròn, đủ **Y** chân chẵn”

1. KHÁI NIỆM KỸ THUẬT LẬP TRÌNH

❖ Khái niệm lập trình

- Với mỗi bài toán [vấn đề] đặt ra, cần:
 - ✓ Thiết kế giải thuật để giải quyết bài toán
 - ✓ Cài đặt giải thuật bằng một chương trình máy tính



1. KHÁI NIỆM KỸ THUẬT LẬP TRÌNH

❖ Thế nào là lập trình tốt?

▪ Đúng/chính xác

- ✓ Thỏa mãn các nhiệm vụ
- ✓ Được khách hàng chấp nhận

▪ Ổn định

- ✓ Ổn định
- ✓ Ít lỗi hoặc lỗi nhẹ có thể chấp nhận được

▪ Tương thích

- ✓ Thích ứng tốt các môi trường khác nhau

▪ Khả năng nâng cấp

- ✓ Dễ dàng chỉnh sửa
- ✓ Dễ dàng nâng cấp khi điều kiện bài toán thay đổi

▪ Tái sử dụng

- ✓ Tái sử dụng hoặc kế thừa cho bài toán khác

▪ Hiệu suất

- ✓ Chương trình nhỏ gọn, ít bộ nhớ
- ✓ Tốc độ nhanh, sử dụng ít CPU

1. KHÁI NIỆM KỸ THUẬT LẬP TRÌNH

❖ Thế nào là lập trình tốt?

▪ Hiệu quả

- ✓ Thời gian lập trình ngắn
- ✓ Khả năng bảo trì dễ dàng
- ✓ Giá trị sử dụng lại lớn
- ✓ Sử dụng đơn giản, thân thiện
- ✓ Nhiều chức năng tiện ích

▪ Hiệu quả

- ✓ Thời gian lập trình ngắn
- ✓ Khả năng bảo trì dễ dàng
- ✓ Giá trị sử dụng lại lớn
- ✓ Sử dụng đơn giản, thân thiện
- ✓ Nhiều chức năng tiện ích

1. KHÁI NIỆM KỸ THUẬT LẬP TRÌNH

❖ Làm thế nào để lập trình tốt?

- Tư duy và phương pháp lập trình
- Hiểu sâu về máy tính
- Nắm vững ngôn ngữ
- Rèn luyện

1. KHÁI NIỆM KỸ THUẬT LẬP TRÌNH

❖ **Kỹ thuật lập trình:** có 3 kỹ thuật lập trình thường được sử dụng

- Thiết kế từ trên xuống (Top-down design)
- Thiết kế từ dưới lên (Bottom-up design)
- Thiết kế mô đun (Modular design)

1. KHÁI NIỆM KỸ THUẬT LẬP TRÌNH

▪ Cách tiếp cận từ trên xuống (Top-down Approach)

- ✓ Bắt đầu bằng cách xác định các thành phần chính của hệ thống, phân rã chúng thành các thành phần ở mức thấp hơn và lặp lại cho đến khi đạt được mức độ chi tiết mong muốn
- ✓ Bắt đầu từ thành phần ở mức cao nhất của hệ thống phân cấp và tiến tới các mức thấp hơn
- ✓ Chỉ phù hợp nếu các đặc tả được biết một cách rõ ràng

▪ Cách tiếp cận từ dưới lên (Bottom-up Approach)

- ✓ Bắt đầu với việc thiết kế các thành phần cơ bản hoặc nguyên thủy nhất và tiến tới các thành phần ở mức cao hơn trên cơ sở sử dụng các thành phần thấp hơn
- ✓ Bắt đầu với thành phần ở mức thấp nhất của hệ thống phân cấp và tiếp tục chuyển qua các mức cao hơn
- ✓ Phù hợp nếu một hệ thống được xây dựng từ hệ thống đã có

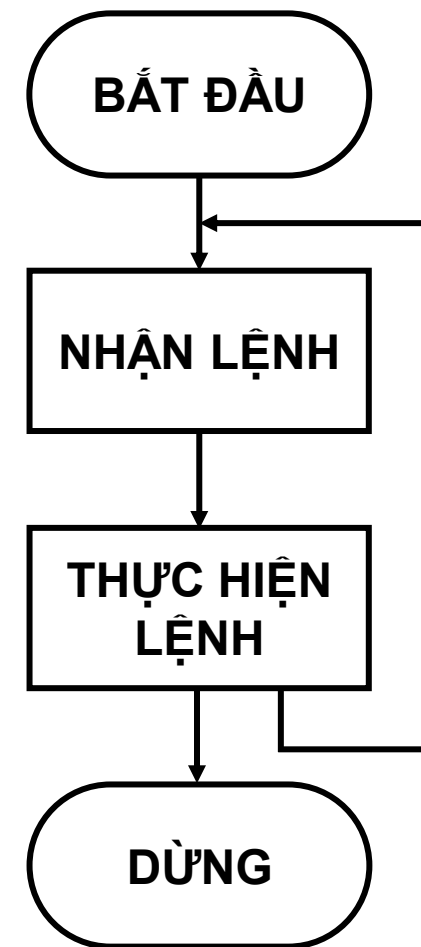
❖ Thiết kế mô đun (Modular design)

- Lập trình mô đun (Modular programming) là lập trình trong đó chương trình lớn được chia thành các module nhỏ để giải quyết

2. TỔNG QUAN VỀ LẬP TRÌNH

❖ Hoạt động của chương trình máy tính

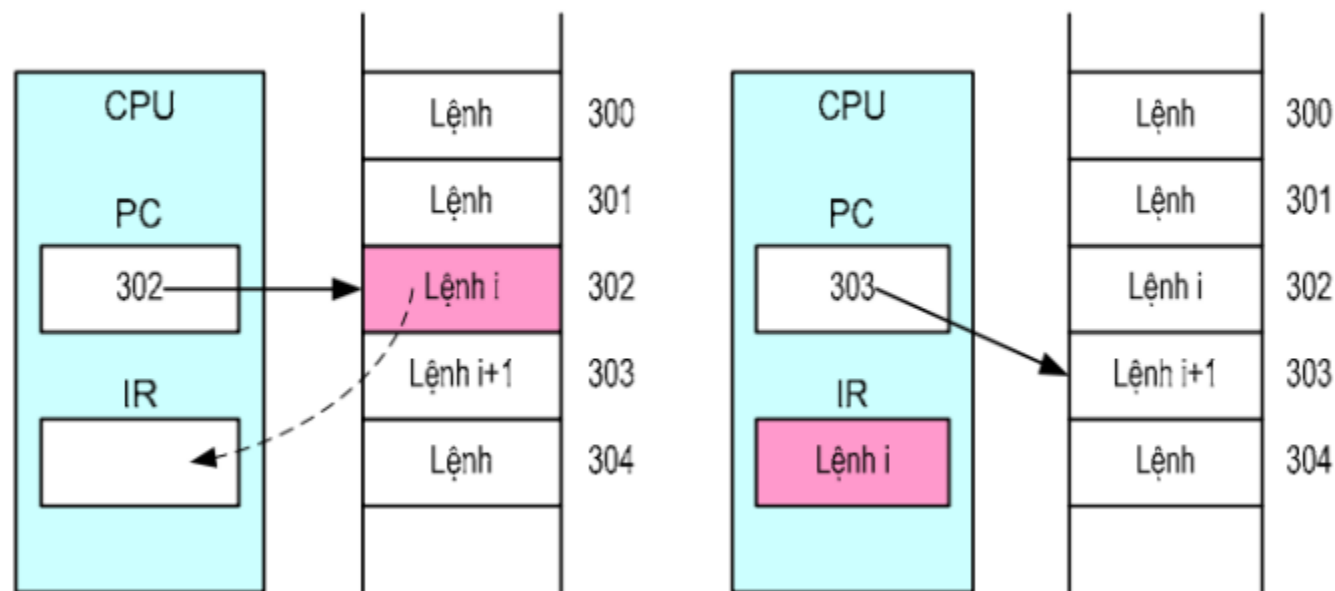
- Chương trình máy tính được nạp vào bộ nhớ chính (Primary Memory) như là một tập các lệnh viết bằng ngôn ngữ mà máy tính hiểu được, tức là một dãy tuần tự các số nhị phân (binary digits)
- Tại thời điểm bất kỳ, máy tính sẽ ở một trạng thái (state) nhất định. Đặc điểm cơ bản của trạng thái là con trỏ lệnh (instruction pointer) trỏ tới lệnh tiếp theo để thực hiện
- Thứ tự thực hiện các nhóm lệnh được gọi là luồng điều khiển (flow of control)



2. TỔNG QUAN VỀ LẬP TRÌNH

❖ Hoạt động của chương trình máy tính

- Bắt đầu mỗi chu trình lệnh, CPU nhận lệnh từ bộ nhớ chính
 - ✓ PC (Program Counter): bộ đếm chương trình là thanh ghi quản lý địa chỉ bộ nhớ của lệnh sẽ được nhận/thực hiện tiếp theo
 - ✓ Lệnh được nạp vào thanh ghi lệnh IR (Instruction Register)
- Sau khi lệnh được nạp vào, nội dung PC tự động tăng để trở sang lệnh kế tiếp



Trước khi nhận Lệnh *i*

Sau khi nhận Lệnh *i*

2. TỔNG QUAN VỀ LẬP TRÌNH

- ❖ Ngôn ngữ lập trình là một hệ thống các ký hiệu dùng để liên lạc, trao đổi với máy tính nhằm thực thi một nhiệm vụ tính toán.
- ❖ Có rất nhiều ngôn ngữ lập trình [>1000], phần lớn là các ngôn ngữ hàn lâm, có mục đích riêng hay phạm vi ứng dụng hạn chế.



2. TỔNG QUAN VỀ LẬP TRÌNH

❖ Các thành phần cơ bản của ngôn ngữ lập trình

- **Mô thức:** Language Paradigm, nguyên tắc chung cơ bản của NNLT
- **Cú pháp:** Syntax, xác định tính hợp lệ
- **Ngữ nghĩa:** Semantic, ghép các ký hiệu thành câu lệnh

2. TỔNG QUAN VỀ LẬP TRÌNH

❖ Mã máy (Machine code)

- Máy tính chỉ nhận các tín hiệu điện tử (có/không), tương ứng với các dòng bit.
- Một chương trình ở dạng đó gọi là mã máy.



2. TỔNG QUAN VỀ LẬP TRÌNH

❖ Hợp ngữ (Assembly)

- Là bước đầu tiên của việc xây dựng cơ chế viết chương trình tiện lợi hơn thông qua các ký hiệu, từ khóa và cả mã máy.
- Để chạy được các chương trình này thì phải chuyển thành mã máy.

```
01 .MODEL SMALL
02 .STACK 100H
03 .CODE
04
05 MOV AX, 0x3C
06 MOV BX, 00000000000001010B
07 ADD AX, BX
08 MOV BX, 14
09 SUB AX, BX
10
11 MOV AH, 04CH
12 INT 21H
```

```
01 .MODEL SMALL
02 .STACK 100H
03 .CODE
04
05 MOV AX, 0x3C
06 MOV BX, 00000000000001010B
07 ADD AX, BX
08 MOV BX, 14
09 SUB AX, BX
10
11 MOV AH, 04CH
12 INT 21H
```

2. TỔNG QUAN VỀ LẬP TRÌNH

❖ Ngôn ngữ lập trình bậc cao

- Thay vì dựa trên phần cứng (machine-oriented) cần tìm cơ chế dựa trên vấn đề (problem-oriented) để tạo chương trình.
- Gần gũi với ngôn ngữ tự nhiên, thường sử dụng các từ khóa giống tiếng Anh.

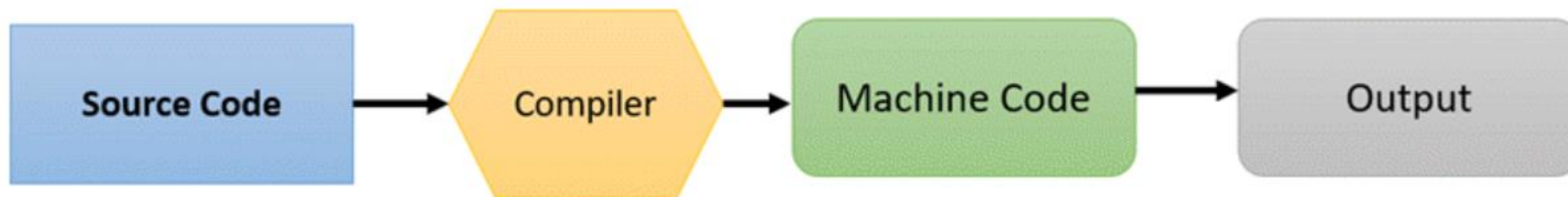
```
32 self.file = None
33 self.fingerprints = set()
34 self.logdupes = True
35 self.debug = debug
36 self.logger = logging.getLogger(__name__)
37 if path:
38     self.file = open(os.path.join(path, 'requests.log'),
39                     'a')
40     self.file.seek(0)
41     self.fingerprints.update(self.request_fingerprint(request))
42
43 @classmethod
44 def from_settings(cls, settings):
45     debug = settings.getbool('DEBUG', False)
46     return cls(job_dir(settings), debug)
47
48 def request_seen(self, request):
49     fp = self.request_fingerprint(request)
50     if fp in self.fingerprints:
51         return True
52     self.fingerprints.add(fp)
53     if self.file:
54         self.file.write(fp + os.linesep)
55
56 def request_fingerprint(self, request):
57     return request_fingerprint(request)
```

```
32 self.file = None
33 self.fingerprints = set()
34 self.logdupes = True
35 self.debug = debug
36 self.logger = logging.getLogger(__name__)
37 if path:
38     self.file = open(os.path.join(path, 'requests.log'),
39                     'a')
40     self.file.seek(0)
41     self.fingerprints.update(self.request_fingerprint(request))
42
43 @classmethod
44 def from_settings(cls, settings):
45     debug = settings.getbool('DEBUG', False)
46     return cls(job_dir(settings), debug)
47
48 def request_seen(self, request):
49     fp = self.request_fingerprint(request)
50     if fp in self.fingerprints:
51         return True
52     self.fingerprints.add(fp)
53     if self.file:
54         self.file.write(fp + os.linesep)
55
56 def request_fingerprint(self, request):
57     return request_fingerprint(request)
```


2. TỔNG QUAN VỀ LẬP TRÌNH

❖ Trình biên dịch (Compiler)

- Là chương trình máy tính biên dịch toàn bộ chương trình nguồn thành mã máy trước khi chương trình chạy (tạo file exe).



2. TỔNG QUAN VỀ LẬP TRÌNH

❖ Trình thông dịch (Interpreter)

- Là chương trình máy tính dịch từng dòng lệnh chương trình nguồn thành mã máy khi chương trình chạy.



3. MÔ THỨC LẬP TRÌNH

- ❖ Imperative paradigm
- ❖ Functional paradigm
- ❖ Logical paradigm
- ❖ Object-oriented paradigm

- ❖ Visual paradigm
- ❖ Parallel paradigm
- ❖ Concurrent paradigm
- ❖ Distributed paradigm
- ❖ Service-oriented paradigm

3. MÔ THỨC LẬP TRÌNH

❖ Mô thức lập trình – hướng mệnh lệnh (Imperative)

first **do this** and next **do that**

- Ý tưởng: Công nghệ số hóa phần cứng (by Von Neumann)
- Che giấu các lệnh trong chương trình con, coi chương trình con là 1 lệnh
- *Tương tự cách mô tả các công việc hằng ngày như là trình tự nấu ăn hay sửa chữa xe cộ*

❖ Mô thức lập trình – hướng mệnh lệnh (Imperative)

▪ Thành phần

- ✓ **Declarative statements**, các lệnh khai báo: cung cấp các tên cho biến. Các biến này có thể thay đổi giá trị trong quá trình thực hiện Chương trình.
- ✓ **Assignment statements**, lệnh gán: gán giá trị mới cho biến.
- ✓ **Program flow control statements**, các lệnh điều khiển cấu trúc chương trình: Xác định trình tự thực hiện các lệnh trong chương trình.
- ✓ **Module**, chia chương trình thành các chương trình con: Functions & Procedures.

3. MÔ THỨC LẬP TRÌNH

❖ Mô thức lập trình – hướng chức năng (Functional)

evaluate an **expression** and
use the **resulting value** for something

- Nguồn gốc: lý thuyết hàm số → đơn giản và rõ ràng hơn mô thức lập trình mệnh lệnh.
- Ngôn ngữ lập trình chức năng miêu tả:
 - ✓ *Tập hợp các kiểu dữ liệu có cấu trúc;*
 - ✓ *Tập hợp các hàm định nghĩa trên các kiểu dữ liệu đó.*

❖ Mô thức lập trình – hướng chức năng (Functional)

▪ Thành phần

- ✓ Tập hợp các cấu trúc dữ liệu và các hàm liên quan
- ✓ Tập hợp các hàm cơ sở
- ✓ Tập hợp các toán tử

▪ Đặc trưng cơ bản: *Module hóa chương trình*

- ✓ Chức năng là biểu diễn của một biểu thức
- ✓ Giải thuật thực hiện theo từng bước
- ✓ Giá trị trả về là không thể biến đổi
- ✓ Không thể thay đổi CTDL của giá trị nhưng có thể sao chép các thành phần tạo nên giá trị đó
- ✓ Tính toán bằng cách gọi các chức năng

3. MÔ THỨC LẬP TRÌNH

❖ Mô thức lập trình – hướng logic (Logical)

answer a **question** via searching for a **solution**

- Ý tưởng: Tự động kiểm chứng trong trí tuệ nhân tạo
- Dựa trên các tiên đề - axioms, các quy luật suy diễn – inference rules, và các truy vấn – queries
- *Chương trình thực hiện từ việc tìm kiếm có hệ thống trong 1 tập các sự kiện, sử dụng 1 tập các luật để đưa ra kết luận.*

3. MÔ THỨC LẬP TRÌNH

❖ Mô thức lập trình – hướng đối tượng (Object-oriented)

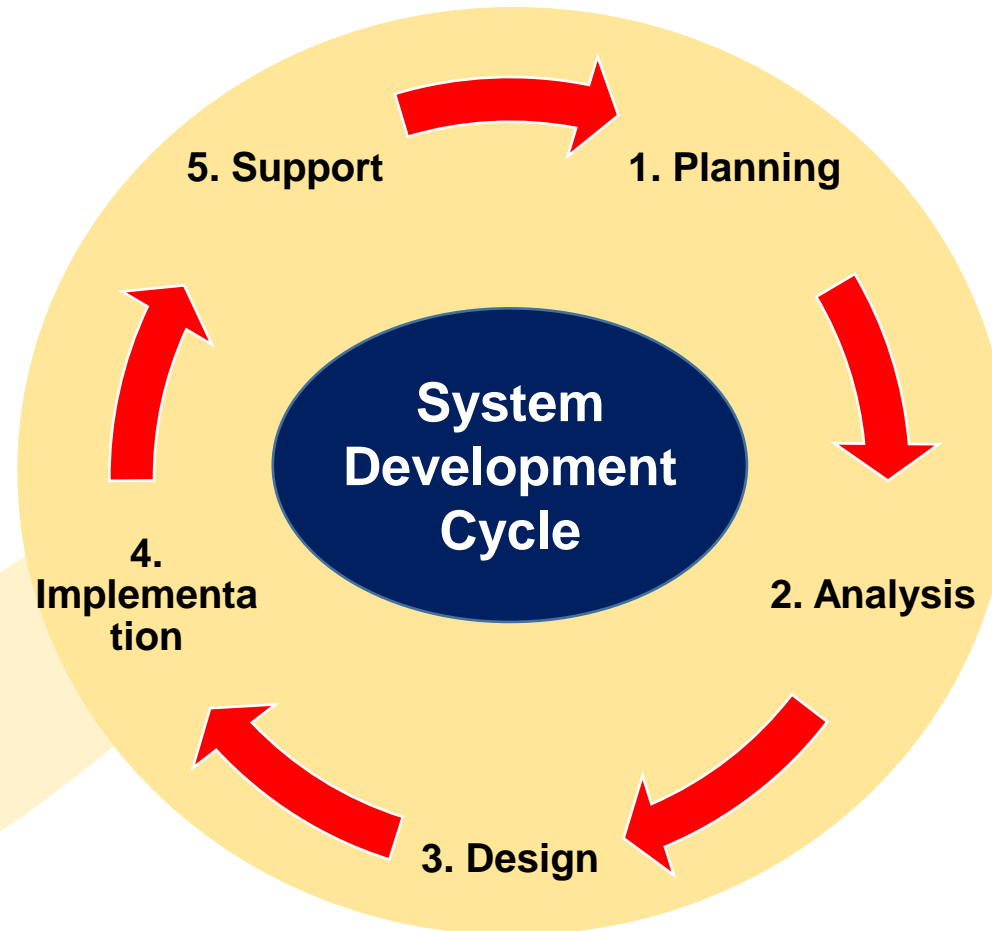
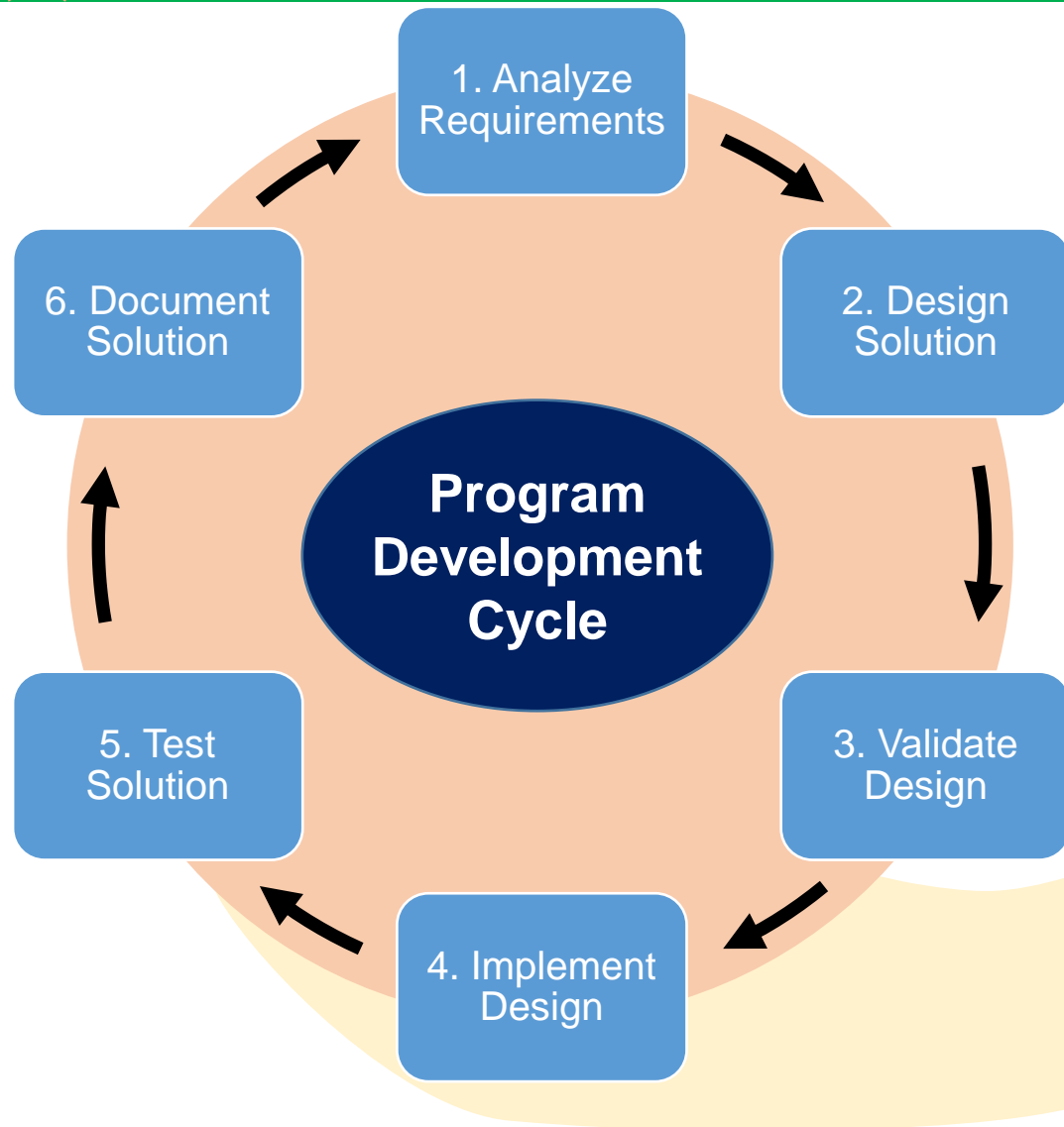
send messages between **objects** to simulate a temporal evolution of a set of **real world phenomena**

- Ý tưởng: Các khái niệm và mô hình tương tác trong thế giới thực;
- Dữ liệu cũng như các thao tác trên dữ liệu được bao gói trong các đối tượng;
- Cơ chế che giấu thông tin nội bộ được sử dụng để tránh những tác động từ bên ngoài.

❖ Mô thức lập trình – hướng đối tượng (Object-oriented)

- Các đối tượng tương tác với nhau qua việc truyền thông điệp, đó là phép ẩn dụ cho việc thực hiện các thao tác trên 1 đối tượng;
- Trong phần lớn các NNLT HĐT, đối tượng được phân loại thành các lớp:
 - ✓ Đối tượng trong các lớp có chung các thuộc tính, cho phép lập trình trên lớp thay vì lập trình trên từng đối tượng riêng lẻ;
 - ✓ Lớp đại diện cho các khái niệm, còn đối tượng đại diện cho thể hiện;
 - ✓ Lớp có tính kế thừa, cho phép mở rộng hay chuyên biệt hóa.

4. CHU TRÌNH PHÁT TRIỂN PHẦN MỀM



❖ Bước 1. Phân tích yêu cầu (Analyse requirements)

▪ Phân tích hệ thống

- ✓ Dựa trên các hệ thống có thực (do con người vận hành hoặc hệ thống tự động)
- ✓ Do các nhà phân tích hệ thống tiến hành, sẽ hiệu quả hơn nếu phỏng vấn người dùng

▪ Mục tiêu: Xác định xem hệ thống hiện tại đã làm được những gì, làm như thế nào, còn tồn tại các vấn đề nào

➤ *Quyết định xem có nên thực hiện bước tiếp theo hay không (Return-on-Investment – ROI estimation)*

❖ Bước 1. Phân tích yêu cầu (Analyse requirements)

▪ Các công việc chính

- ✓ Thiết lập các requirements
- ✓ Gặp các nhà phân tích hệ thống và users
- ✓ Xác định input, output, processing, và các thành phần dữ liệu

IPO CHART

Input	Processing	Output
Regular Time Hours Worked	Read regular time hours worked, overtime hours worked, hourly pay rate.	Gross Pay
Overtime Hours Worked	Calculate regular time pay.	
Hourly Pay Rate	If employee worked overtime, calculate overtime pay.	
	Calculate gross pay.	
	Print gross pay.	

❖ Bước 2. Thiết kế giải pháp (Design solution)

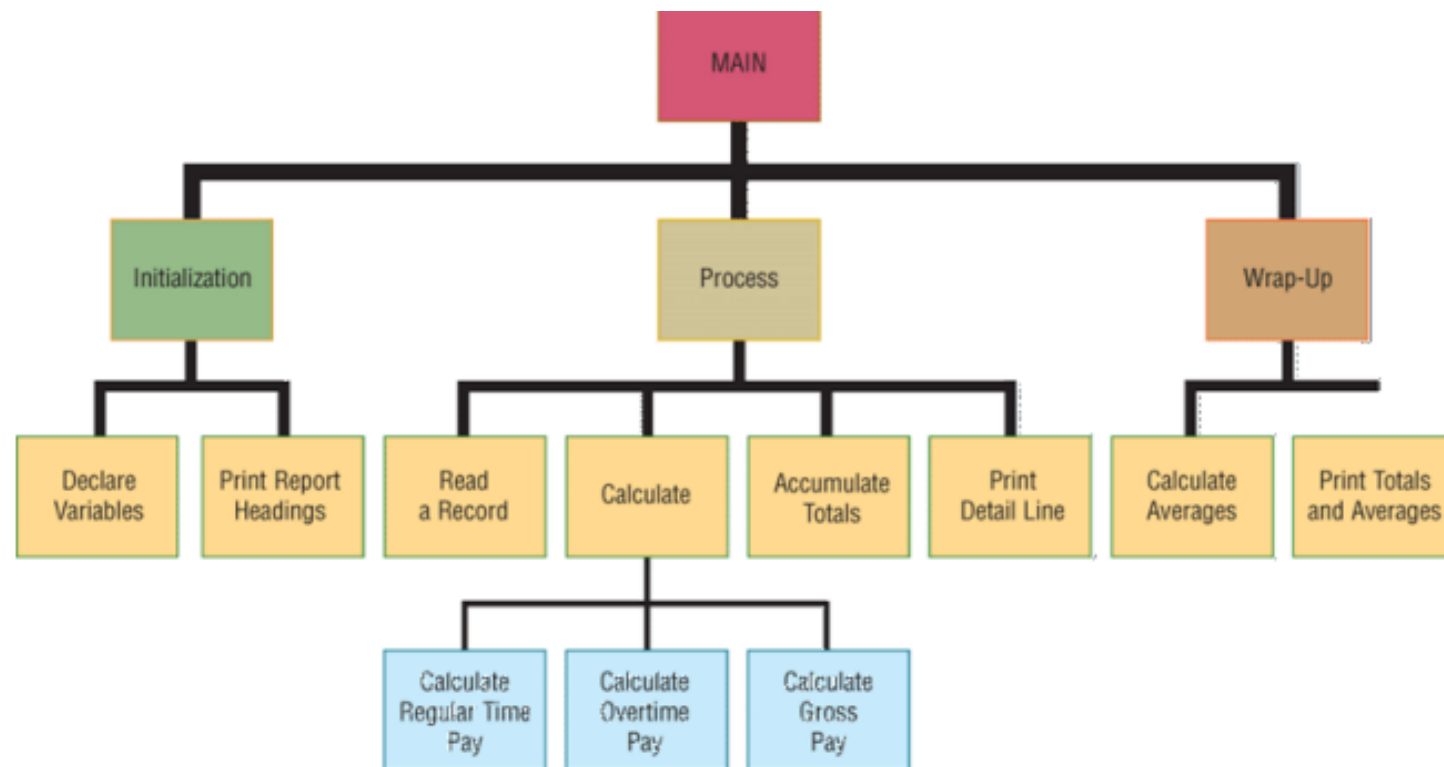
▪ Hai hướng tiếp cận

- ✓ Thiết kế hướng đối tượng (Object-oriented design)
- ✓ Thiết kế hướng cấu trúc (Top-down design)
- ✓ Phân chia hệ thống từng bước thành các thủ tục để giải quyết vấn đề
- ✓ LTV bắt đầu với thiết kế tổng thể rồi đi đến thiết kế chi tiết

❖ Bước 2. Thiết kế giải pháp (Design solution)

- Thiết kế Sơ đồ phân cấp chức năng (hierarchy chart)

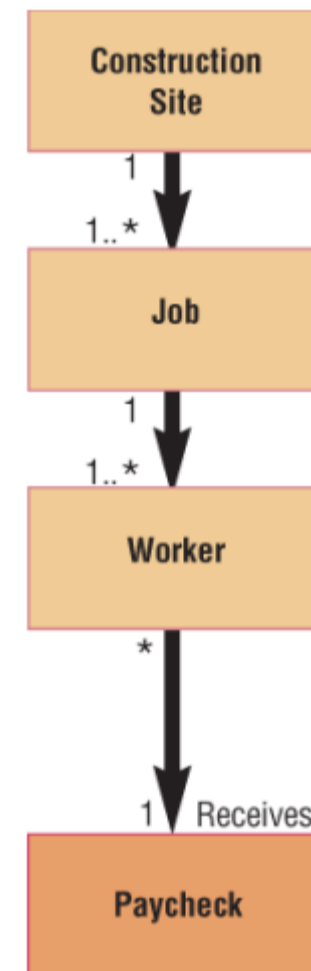
- ✓ Còn gọi là sơ đồ cấu trúc
- ✓ Trực quan hóa các module chương trình



❖ Bước 2. Thiết kế giải pháp (Design solution)

▪ Thiết kế hướng đối tượng

- ✓ LTV đóng gói dữ liệu và các thủ tục xử lý dữ liệu trong đối tượng (object)
- ✓ Các đối tượng được phân loại thành các lớp (classes)
- ✓ Thiết kế các biểu đồ lớp thể hiện trực quan các quan hệ phân cấp quan hệ của các lớp

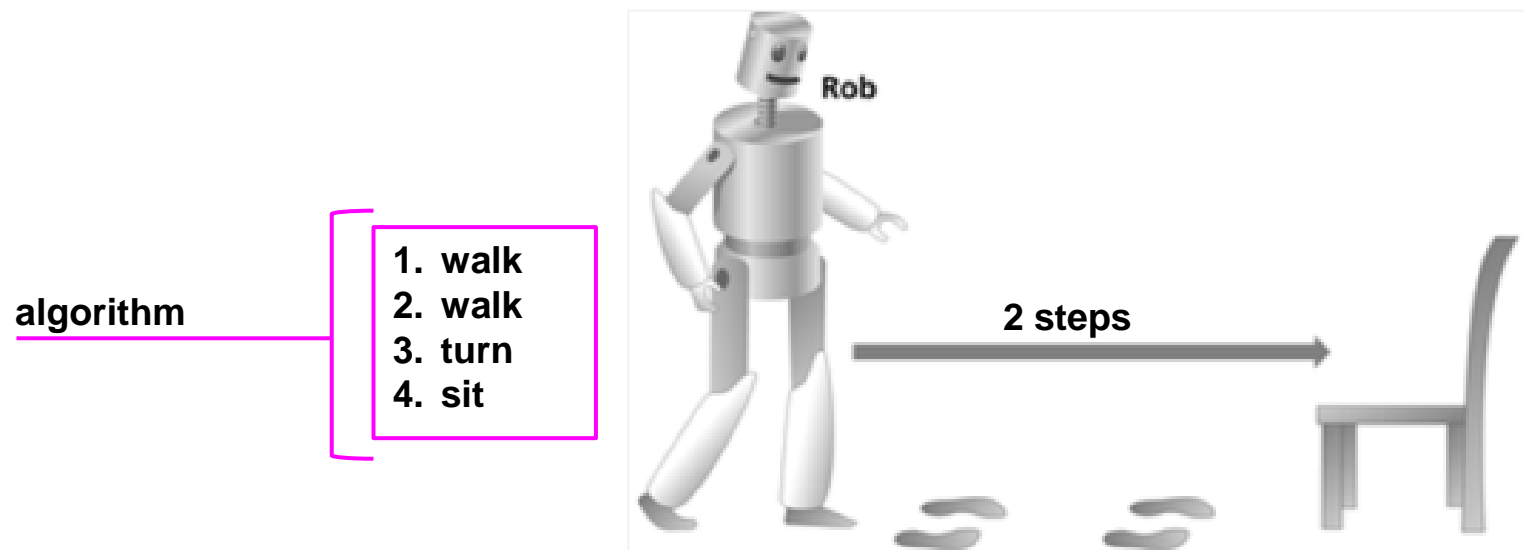


❖ Bước 2. Thiết kế giải pháp (Design solution)

- Thiết kế giải thuật (thiết kế thuật toán)
 - ✓ Máy tính không thể tự nghĩ ra hoặc tự quyết định một sơ đồ hoạt động
 - ✓ Máy tính chỉ có thể làm chính xác những gì được yêu cầu, theo cách được yêu cầu, chứ không phải làm những gì con người muốn máy tính thực hiện
 - ✓ Giải thuật là một tập các chỉ thị miêu tả cho máy tính nhiệm vụ cần làm và thứ tự thực hiện các nhiệm vụ đó.
 - ✓ Giải thuật được thiết kế theo ba cấu trúc suy luận cơ bản: tuần tự (Sequential), lựa chọn (Selection), lặp (Repeating)

❖ Cấu trúc tuần tự (Sequential)

- Xử lý lần lượt các lệnh (statement) của chương trình theo thứ tự được chỉ ra trong chương trình



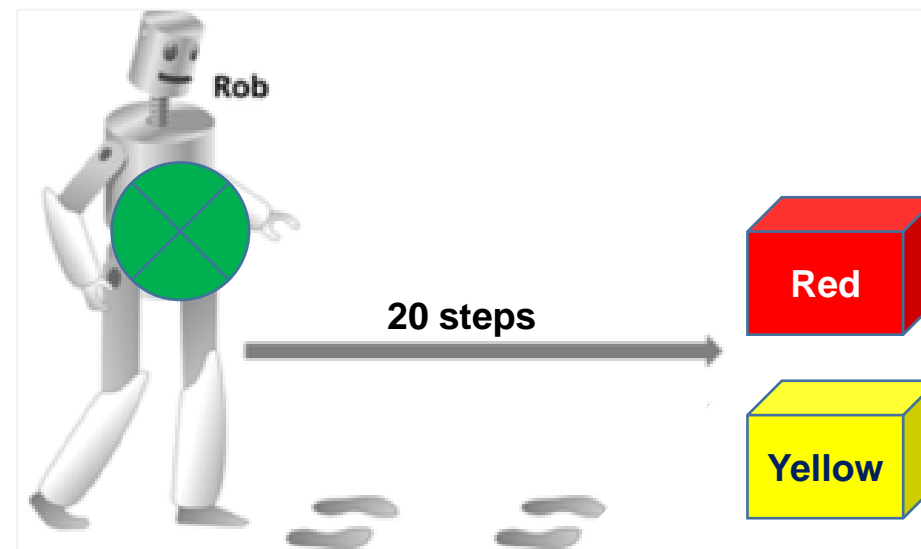
❖ Cấu trúc chọn (Selection)

- Phải chỉ ra được các hành động có khả năng được thực hiện sau khi có quyết định. Quyết định phụ thuộc vào các điều kiện

algorithm

Indent the instructions within the if and otherwise selections of a selection structure

1. repeat 20 times:
walk
2. if the balloon is red, do this:
drop the balloon in the red box
otherwise, do this:
drop the balloon in the yellow box
3. turn
4. repeat 20 times:
walk
5. turn



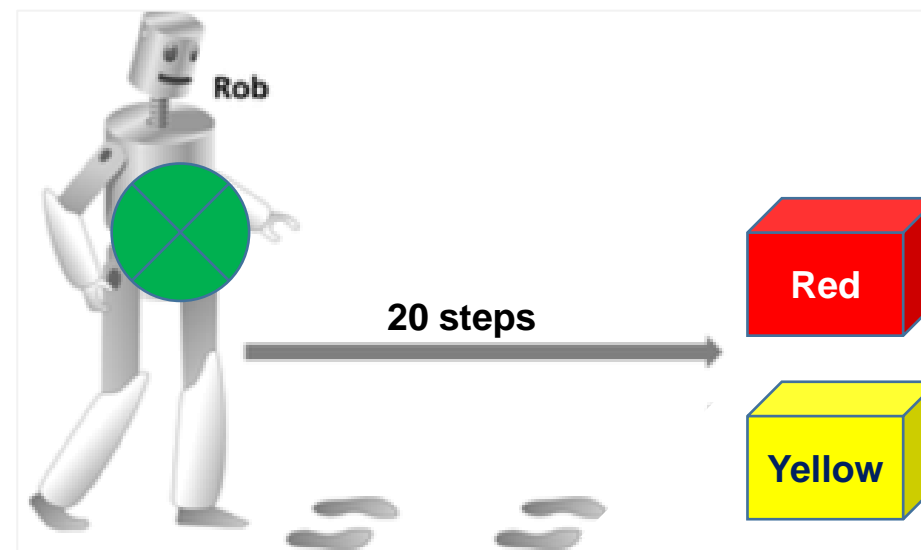
❖ Cấu trúc lặp (Repeating)

- Thực hiện lặp đi lặp lại một hoặc nhiều lệnh, cho đến khi thỏa mãn điều kiện. Có 2 loại: Lặp xác định; Lặp không xác định

algorithm

Indent the instructions within the if and otherwise selections of a selection structure

1. repeat 20 times:
 walk
2. if the balloon is red, do this:
 drop the balloon in the red box
 otherwise, do this:
 drop the balloon in the yellow box
3. turn
4. repeat 20 times:
 walk
5. turn



❖ Ngôn ngữ tự nhiên

- Là ngôn ngữ của chúng ta đang sử dụng
- Sử dụng ngôn ngữ tự nhiên để mô tả giải thuật

Ví dụ: Ta có giải thuật giải phương trình bậc nhất dạng $ax + b = 0$ như sau:

Bước 1: Nhận giá trị của các tham số a, b

Bước 2: Xét giá trị của a xem có bằng 0 hay không?

Nếu $a = 0$ thì làm bước 3, nếu a khác không thì làm bước 4.

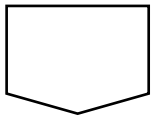
Bước 3: (a bằng 0) Nếu b bằng 0 thì ta kết luận phương trình vô số nghiệm, nếu b khác 0 thì ta kết luận phương trình vô nghiệm.



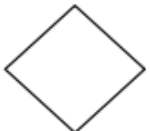


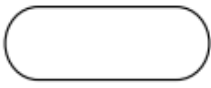
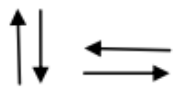
Bước 4: (a khác 0) Ta kết luận phương trình có nghiệm $x = -b/a$

4. CHU TRÌNH PHÁT TRIỂN PHẦN MỀM

❖ Biểu đồ luồng (Flowchart)

- Mô tả giải thuật một cách trực quan
- Sử dụng sơ đồ khối, mỗi khối quy định một hành động như hình bên

Symbol	Name	Function
	Off-page Connector	Connects two parts of a flowchart which are spread over different pages.

Symbol	Name	Function
	Process	Indicates any type of internal operation inside the Processor or Memory
	input/output	Used for any Input / Output (I/O) operation. Indicates that the computer is to obtain data or output results
	Decision	Used to ask a question that can be answered in a binary format (Yes/No, True/False)
	Connector	Allows the flowchart to be drawn without intersecting lines or without a reverse flow.
	Predefined Process	Used to invoke a subroutine or an Interrupt program.
	Terminal	Indicates the starting or ending of the program, process, or interrupt program
	Flow Lines	Shows direction of flow.

4. CHU TRÌNH PHÁT TRIỂN PHẦN MỀM

❖ Ví dụ (Flowchart): Vẽ lưu đồ tính giá trị trung bình của 2 số

- ✓ **Bước 1.** Bắt đầu
- ✓ **Bước 2.** Nhập số thứ 1, số thứ 2
- ✓ **Bước 3.** Kiểm tra có phải là số?
*Nếu đúng là số thực hiện **Bước 4**
Ngược lại quay lại **Bước 2***
- ✓ **Bước 4.** Tính giá trị trung bình
- ✓ **Bước 5.** Hiển thị kết quả
- ✓ **Bước 6.** Kết thúc

Bước 1. Start



Bước 2. Input num1, num2



Bước 3. Check num1 & num2 are number



Yes

Bước 4. Average = $(\text{num1} + \text{num2}) / 2$

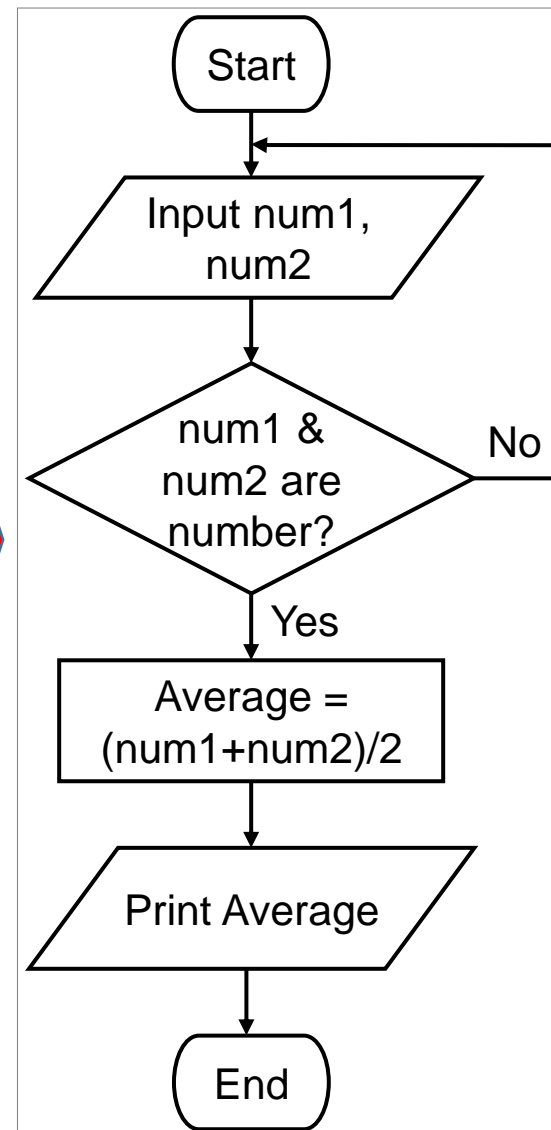


Bước 5. Print Average



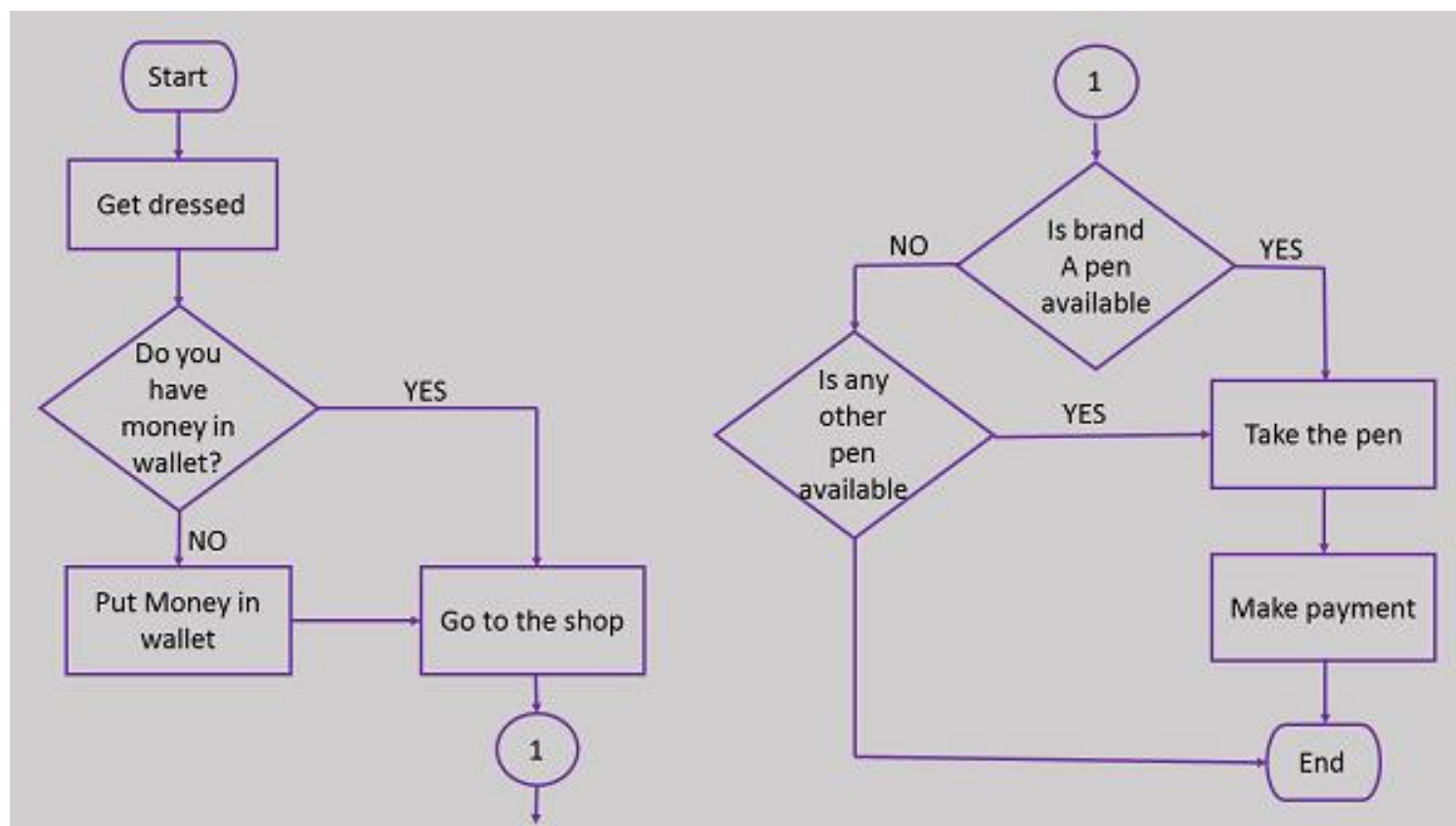
Bước 6. End

No



4. CHU TRÌNH PHÁT TRIỂN PHẦN MỀM

❖ **Ví dụ (Flowchart):** Cho lưu đồ thực hiện việc mua bút. Hãy giải thích các bước thực hiện?



❖ Giải mã (Mã giả: Pseudocode)

- một bản mô tả giải thuật ngắn gọn và không chính thức
- sử dụng những quy ước có cấu trúc của một số ngôn ngữ lập trình (thường là Pascal)
- bỏ đi những chi tiết không cần thiết để giúp hiểu rõ giải thuật hơn

Ví dụ: Giải phương trình bậc 2

Vào: a, b, c

Ra: Kết luận về nghiệm

BEGIN

Delta: = $b*b - 4*a*c$;

If Delta = 0 Then

Phương trình có nghiệm kép $x = -$

$b/(2*a)$;

else

begin

if Delta < 0 then

Phương trình Vô nghiệm

Else

Begin

Phương trình có 2 nghiệm

$x1 = (-b + \sqrt{\text{Delte}})/(2*a)$

$x2 = (-b + \sqrt{\text{Delte}})/(2*a)$

end

end

END

❖ Bước 3. Kiểm chứng thiết kế (Validate design)

- Kiểm tra độ chính xác của chương trình
- Desk check: LTV dùng các dữ liệu thử nghiệm (Test data: các dữ liệu thử nghiệm giống như số liệu thực mà chương trình sẽ thực hiện) để kiểm tra chương trình
- LTV kiểm tra tính đúng đắn bằng cách tìm các lỗi logic (Logic Error: các sai sót khi thiết kế gây ra những kết quả không chính xác)
- Structured Walkthrough: LTV mô tả logic của thuật toán trong khi đội lập trình duyệt theo logic chương trình

❖ Bước 4. Cài đặt thiết kế (Implement design)

- Viết mã nguồn chương trình (coding): dịch từ thiết kế thành chương trình
 - ✓ *Cú pháp (Syntax): Quy tắc xác định cách viết các lệnh*
 - ✓ *Chú thích (Comments): Tài liệu chương trình (tài liệu trong)*
- Lập trình nhanh (Extreme programming - XP): Viết mã nguồn và kiểm thử ngay sau khi các yêu cầu được xác định

❖ Bước 5. Kiểm thử giải pháp (Test solution)

- Đảm bảo chương trình chạy thông và cho kết quả chính xác
- Debugging: Tìm và sửa các lỗi syntax và logic errors (Kiểm tra phiên bản beta, giao cho Users dùng thử và thu thập phản hồi)

❖ **Bước 6. Viết tài liệu cho giải pháp (Document solution)**

- Rà soát lại program code – loại bỏ các dead code, tức các lệnh mà chương trình không bao giờ gọi đến
- Rà soát, hoàn thiện documentation

❖ Câu hỏi

- Trình bày khái niệm Kỹ thuật lập trình?
- Phân biệt Kỹ thuật lập trình với phân tích & thiết kế?
- Trình bày các thành phần cơ bản của ngôn ngữ lập trình?
- Phân biệt giữa trình biên dịch với trình thông dịch?
- Phân biệt giữa ngôn ngữ lập trình bậc thấp với ngôn ngữ lập trình bậc cao?
- Phân biệt giữa các mô thức lập trình?
- Trình bày chu trình phát triển phần mềm?