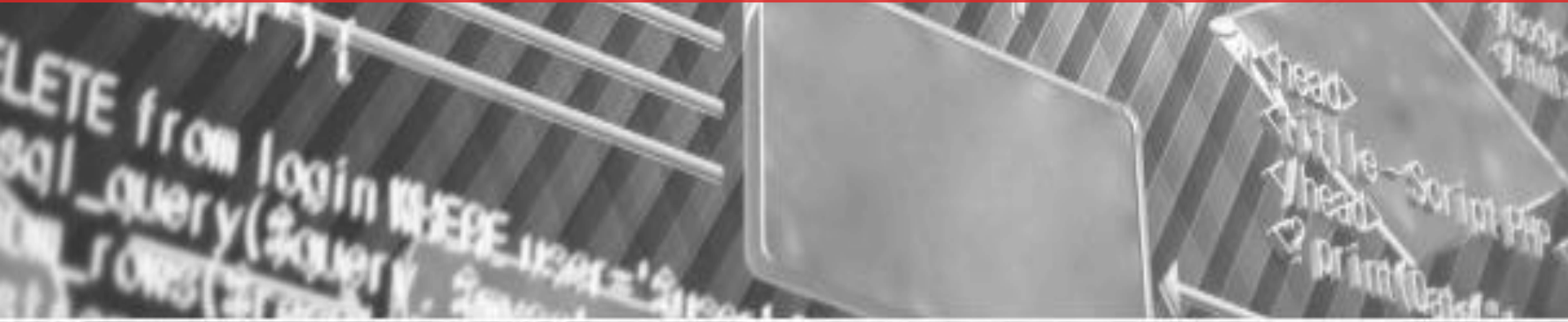


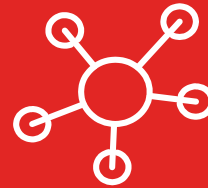
Trịnh Thành Trung (ThS)
trungtt@soict.hust.edu.vn

Bài 1

TỔNG QUAN



Nội dung



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

Các khái niệm cơ bản về 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*

Tổng quan

Kỹ thuật lập trình

- Kỹ thuật lập trình

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)

Thế nào là lập trình

- Viết chương trình tính giai thừa của 100

- Viết chương trình tính giai thừa của 100 số

- Giải bài toán cổ

«*Vừa gà vừa chó, ba mươi sáu con, bó lại cho tròn, một trăm chân chẵn*»

- Viết chương trình tính giai thừa

- Viết chương trình in ra **n số nguyên tố** đầu tiên

- Giải bài toán cổ

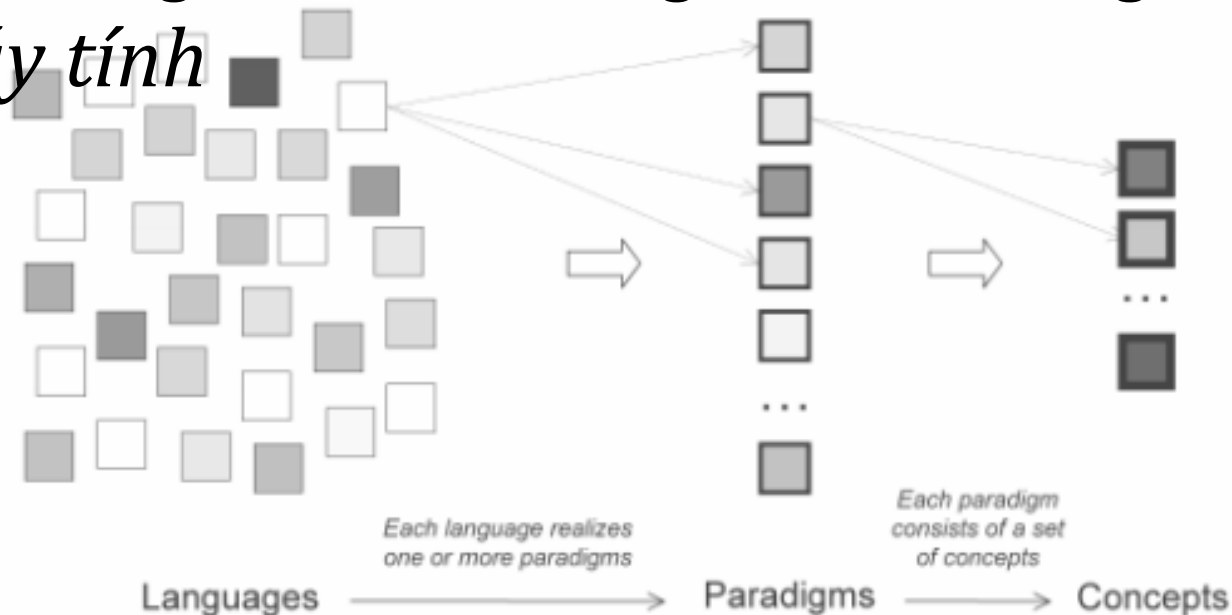
«*Vừa gà vừa chó, vừa vạy **X** con, bó lại cho tròn, đủ **Y** chân chẵn*»

**KHÔNG PHẢI LÀ
LẬP TRÌNH**

Khái niệm lập trình

Với mỗi bài toán (vấn đề) đặt ra, cần:

- *Thiết kế giải thuật để giải quyết bài toán đó*
- *Cài đặt giải thuật bằng một chương trình máy tính*



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

Khả năng nâng cấp

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

Tái sử dụng

- Tái sử dụng hoặc kế thừa cho bài toán khác

Thế nào là lập trình **tốt**

Tương thích

- Thích ứng tốt các môi trường khác nhau

Hiệu suất

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

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

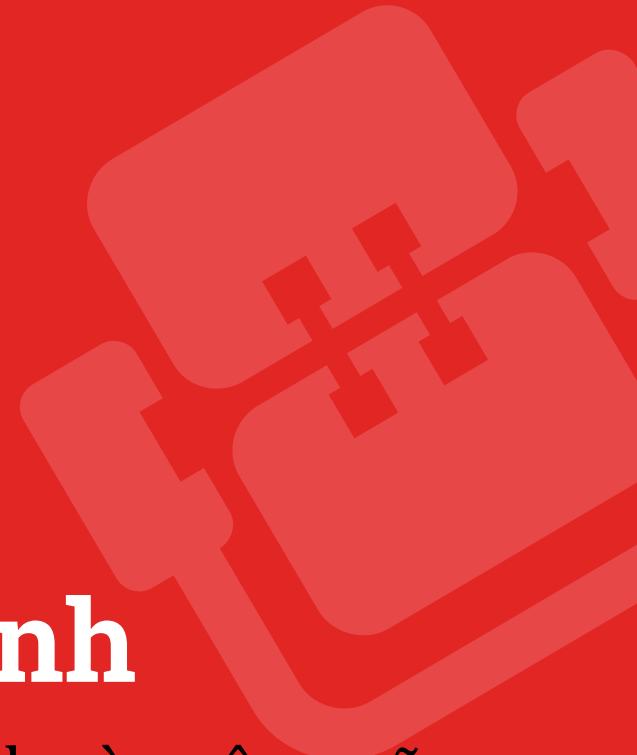
Làm thế nào để lập trình tốt

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

2.

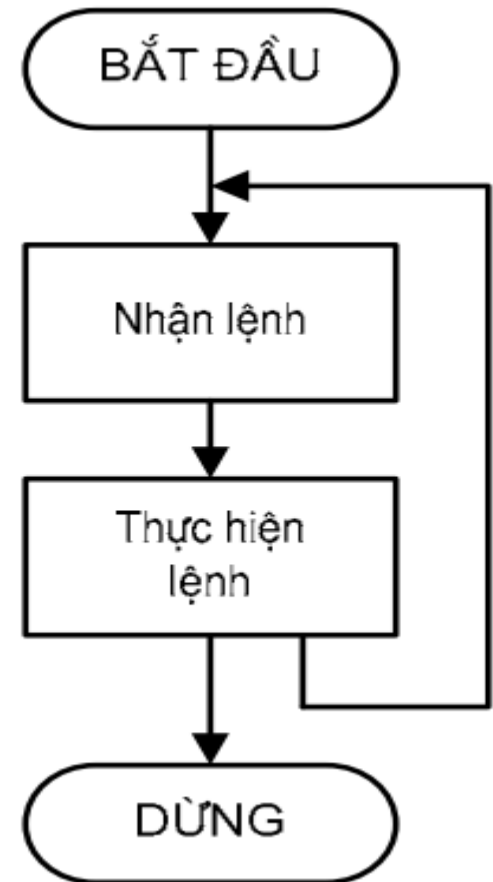
Tổng quan về lập trình

Hoạt động của chương trình máy tính và ngôn ngữ lập trình



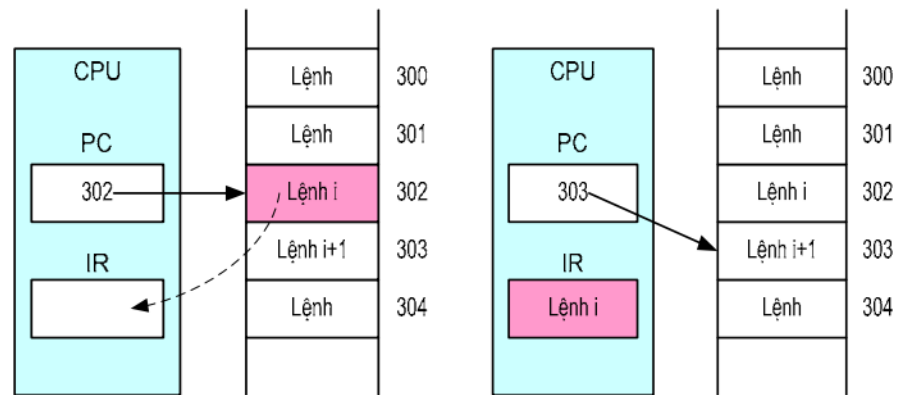
Hoạt động của chương trình máy tính

- 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 bất cứ một thời điểm nào, máy tính sẽ ở một trạng thái (state) nào đó. 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).



Hoạt động của chương trình máy tính

- Bắt đầu mỗi chu trình lệnh, CPU nhận lệnh từ bộ nhớ chính.
 - *PC (Program Counter): thanh ghi giữ địa chỉ của lệnh sẽ được nhận*
 - *Lệnh được nạp vào thanh ghi lệnh IR (Instruction Register)*
- Sau khi lệnh được nhận vào, nội dung PC tự động tăng để trở sang lệnh kế tiếp



Trước khi nhận Lệnh i

Sau khi nhận Lệnh i

Ngôn ngữ 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 (khoảng hơn 1000), phần lớn là các ngôn ngữ hàn lâm, có mục đích riêng hay phạm vi ứng dụng hạn chế

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 cái gì là hợp lệ

Ngữ nghĩa

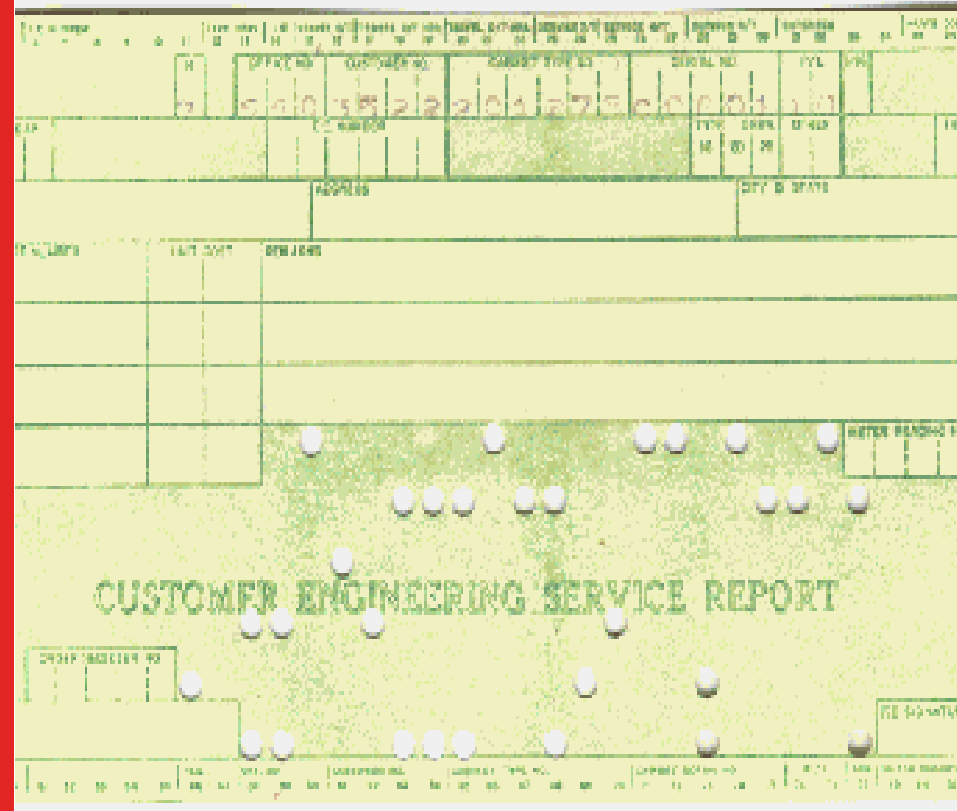
- Semantic, ghép các ký hiệu thành câu lệnh

Mã máy

Machine code

Máy tính chỉ nhận các tín hiệu điện tử - có, không có - tương ứng với các dòng bits.

Một chương trình ở dạng đó gọi là mã máy (machine code).



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.

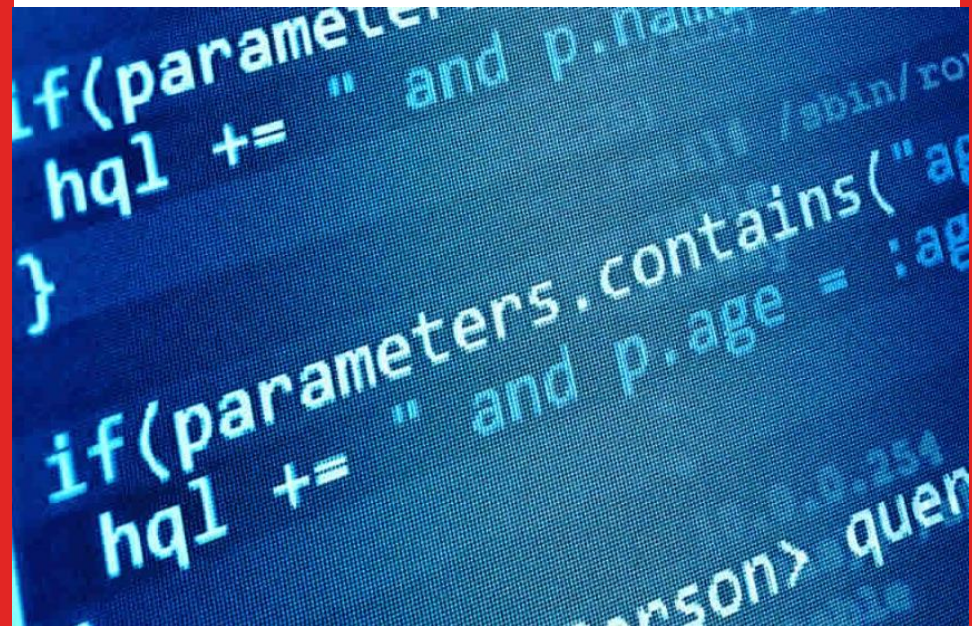
Tất nhiên, để chạy được các chương trình này thì phải chuyển thành machine code.

```
8B4D 10          MOV     ECX, 10
81E1 FFFF0000   AND     ECX, 0
3BC1           CMP     ECX, 1
74 21          JE      SHORT 00000000
68 78594200    PUSH   OFFSET 00000000
6A 00         PUSH   0
68 7F020000    PUSH   0
68 EC584200    PUSH   OFFSET 00000000
6A 02         PUSH   2
E8 B1CBFFFF    CALL   00000000
83C4 14        ADD     ESP, 14
83F8 01        CMP     EAX, 1
75 01         JNE    SHORT 00000000
CC           INT3
> 33D2         XOR     EDI, EDI
85D2         TEST   EDI, EDI
> ^ 75 C1        JNE    SHORT 00000000
> 837D 1C 00   CMP     DWORD PTR [7D1C], 0
74 25         JE      SHORT 00000000
8B45 0C        MOV     EAX, DWORD PTR [450C]
83C0 24        ADD     EAX, 24
50           PUSH   EAX
004D 50        MOV     EAX, DWORD PTR [4D50]
```

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 hơn, thường sử dụng các từ khóa giống tiếng Anh



Tương lai của ngôn ngữ lập trình?



AI



neural
network



?

Trình dịch compiler

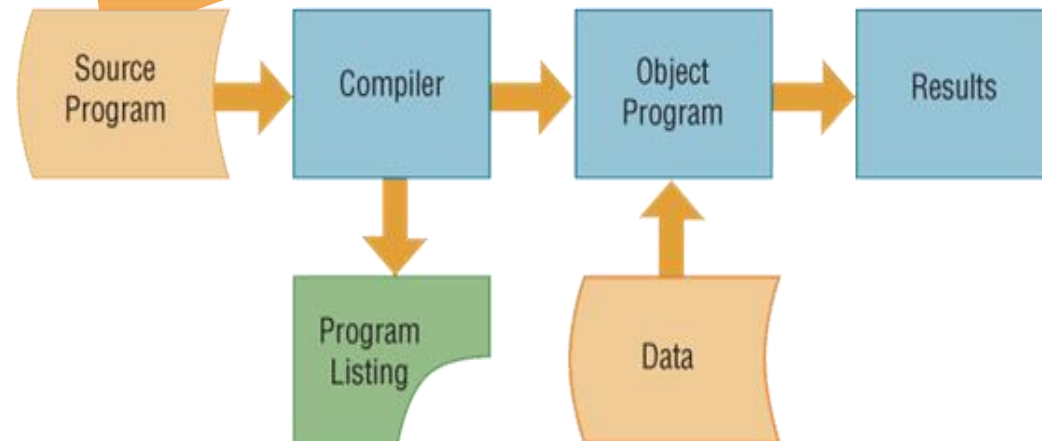
- Chương trình thực hiện biên dịch toàn bộ chương trình nguồn thành mã máy trước khi thực hiện

```
* COMPUTE REGULAR TIME PAY
MULTIPLY REGULAR-TIME-HOURS BY HOURLY-PAY-RATE
GIVING REGULAR-TIME-PAY.

* COMPUTE OVERTIME PAY
IF OVERTIME-HOURS > 0
  COMPUTE OVERTIME-PAY = OVERTIME-HOURS * 1.5 * HOURLY-PAY-RATE
ELSE
  MOVE 0 TO OVERTIME-PAY.

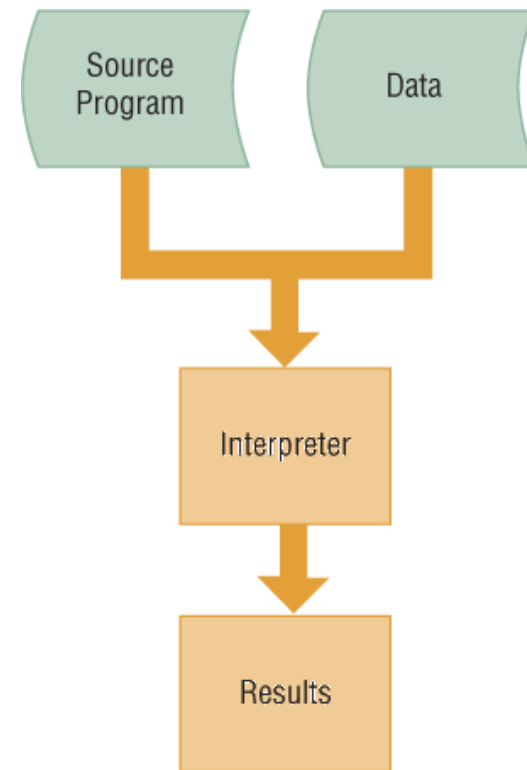
* COMPUTE GROSS PAY
ADD REGULAR-TIME-PAY TO OVERTIME-PAY
GIVING GROSS-PAY.

* PRINT GROSS PAY
MOVE GROSS-PAY TO GROSS-PAY-OUT.
WRITE REPORT-LINE-OUT FROM DETAIL-LINE
AFTER ADVANCING 2 LINES.
```



Thông dịch interpreter

- Chương trình dịch và thực hiện từng dòng lệnh của chương trình cùng lúc
- Dịch từ ngôn ngữ này sang ngôn ngữ khác, không tạo ra chương trình dạng mã máy hay assembly



3.

Các mô thức lập trình

Programming paradigm



Các mô thức lập trình

- Imperative paradigm
- Functional paradigm
- Logical paradigm
- Object-oriented paradigm
- Visual paradigm
- Parallel paradigm
- Concurrent paradigm
- Distributed paradigm
- Service-oriented paradigm

Mô thức lập trình hướng mệnh lệnh

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 hàng ngày như là trình tự nấu ăn hay sửa chữa xe cộ*

Mô thức lập trình hướng mệnh lệnh

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

Mô thức lập trình hướng chức năng

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 hướng mệnh lệnh
- Ngôn ngữ lập trình hướng 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

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*

Mô thức lập trình hướng logic

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*

Mô thức lập trình hướng đối tượng

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

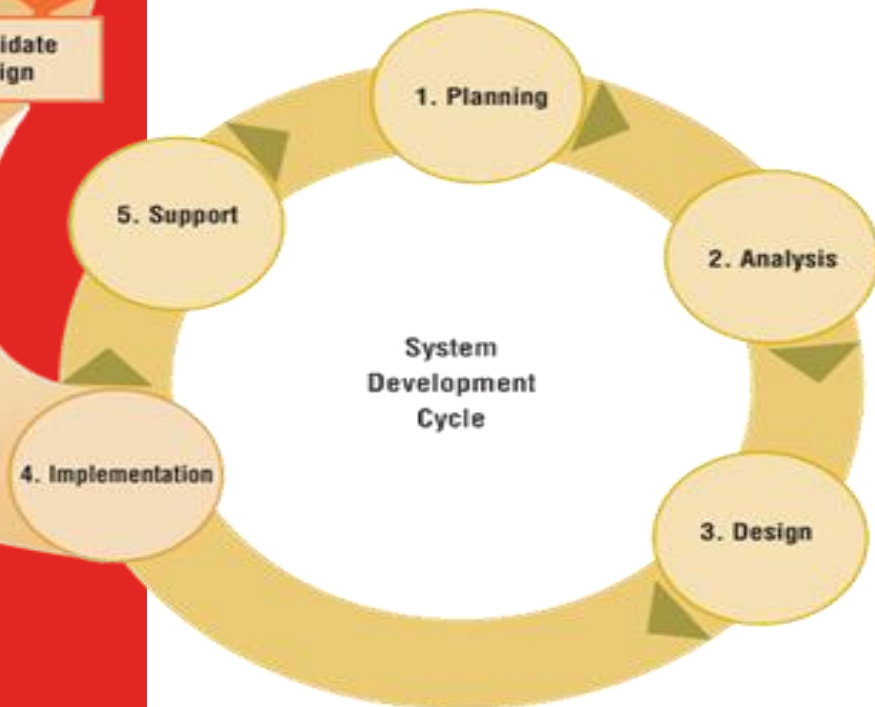
- Các đối tượng tương tác với nhau qua việc truyền thông điệp, đó là phép ẩn dụ cho việc thực hiện các thao tác trên 1 đối tượng
- Trong phần lớn các NNLT HĐT, đối tượng phân loại thành các lớp
 - *Đối tượng trong các lớp có chung các thuộc tính, cho phép lập trình trên lớp, thay vì lập trình trên từng 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



Development cycle



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 gì*

→ 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.	

Ví dụ

IPO chart tính giá trị trung bình



- Viết chương trình cho phép nhập vào 3 số, tính tổng của chúng và tính giá trị trung bình của chúng

Bước 2: Thiết kế giải pháp

Design solution



Object-oriented design

Structured design, còn gọi là top-down design

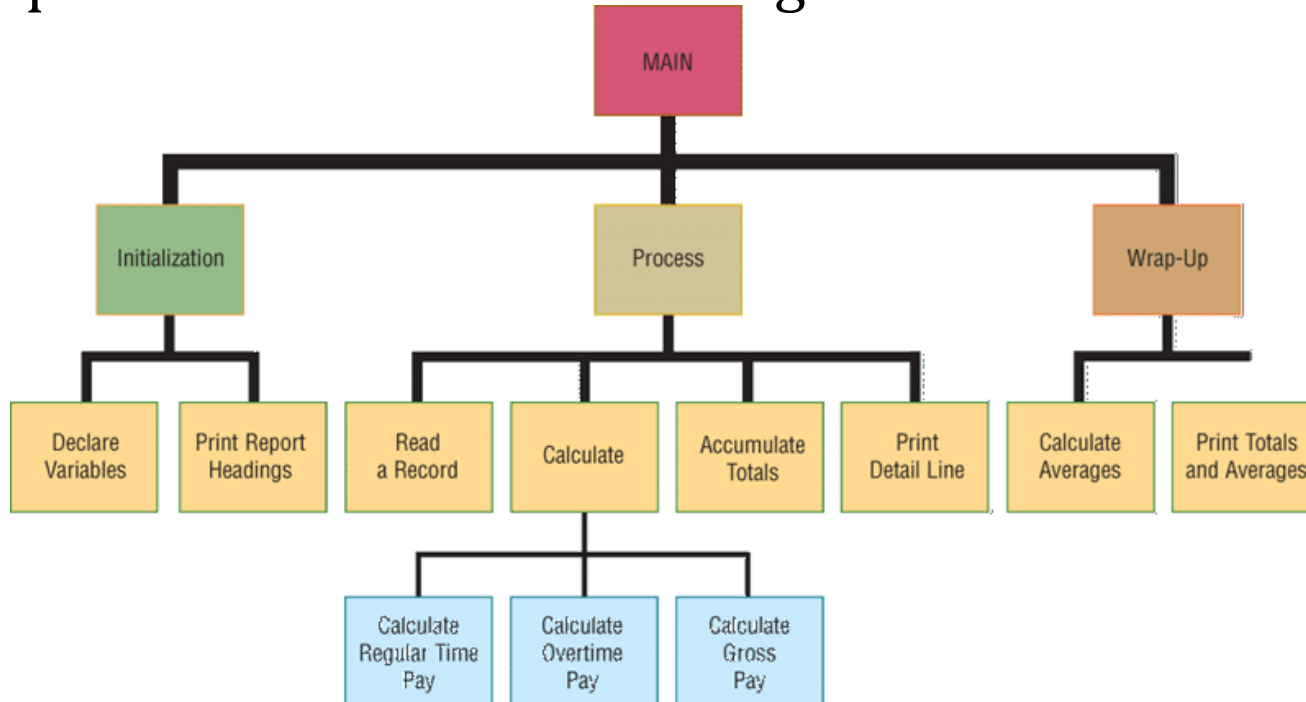
LTV bắt đầu với thiết kế tổng thể rồi đi đến thiết kế chi tiết

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

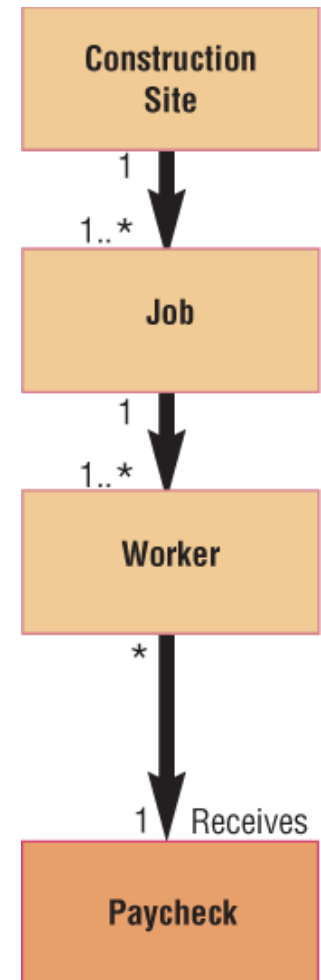


Bước 2: Thiết kế giải pháp

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

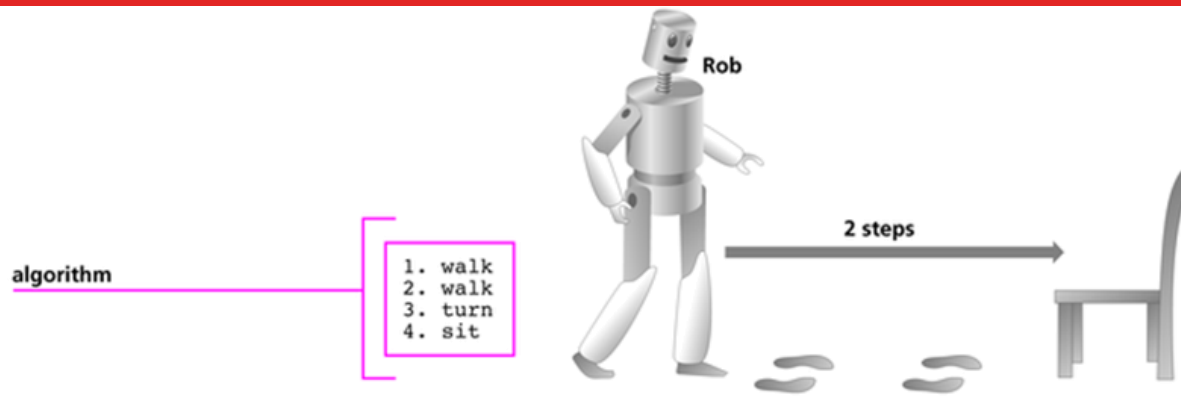
Design solution

Thiết kế giải thuật

- Máy tính không thể tự nghĩ ra hay 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 làm
- 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.

Cấu trúc tuần tự

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



Cấu trúc chọn

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 sections 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

thực hiện lặp đi lặp lại một hoặc nhiều lệnh, cho đến khi thỏa mãn điều kiện



algorithm









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

indent the instructions
within the if and
otherwise sections of a
selection structure

Biểu đồ luồng Flowchart

Mô tả giải thuật một cách
trực quan
Sử dụng ít ký hiệu để định
nghĩa giải thuật với độ
khó khác nhau

ANSI FLOWCHART SYMBOLS

	PROCESS: program instruction(s) that transforms input(s) into output(s)
	INPUT/OUTPUT: enter data or display information
	ANNOTATION: additional descriptive information about the program
	DECISION: condition that determines a specified path to follow
	TERMINAL: beginning or ending of program
	CONNECTOR: entry from or exit to another part of the flowchart on the same page
	CONNECTOR: entry from or exit to another part of the flowchart on a different page
	PREDEFINED PROCESS: named process containing a series of program steps specified elsewhere

Ví dụ flow chart



- Thiết kế biểu đồ
luồng chuyển từ số
nhị phân sang số
thập phân

Bước 3: Chứng thực thiết kế

Validate design

Kiểm tra độ chính xác của chương trình

Desk check
LTV dùng các dữ liệu thử nghiệm (Test data) để kiểm tra chương trình

Test data
các dữ liệu thử nghiệm giống như số liệu thực mà chương trình sẽ thực hiện

LTV kiểm tra tính đúng đắn bằng cách tìm các lỗi logic (Logic Error)

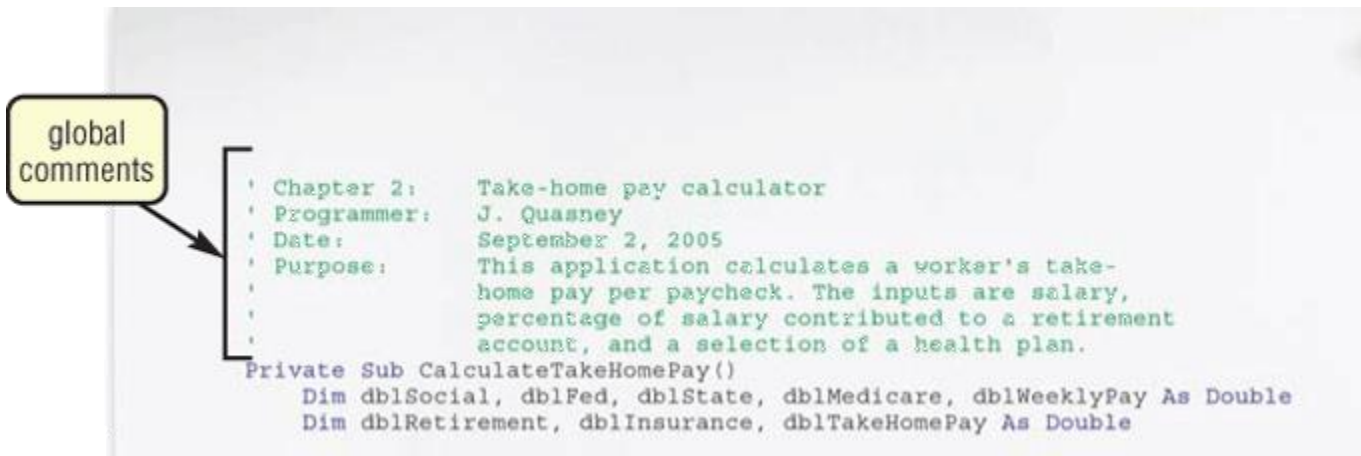
Logic error
các sai sót khi thiết kế gây ra những kết quả không chính xác

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



A screenshot of code with a callout box. The callout box is yellow with a black border and contains the text "global comments". An arrow points from the callout box to the first line of the code block, which is a comment: "Chapter 2: Take-home pay calculator".

```
Chapter 2: Take-home pay calculator
Programmer: J. Quasney
Date: September 2, 2005
Purpose: This application calculates a worker's take-home pay per paycheck. The inputs are salary, percentage of salary contributed to a retirement account, and a selection of a health plan.
Private Sub CalculateTakeHomePay()
    Dim dblSocial, dblFed, dblState, dblMedicare, dblWeeklyPay As Double
    Dim dblRetirement, dblInsurance, dblTakeHomePay As Double
```

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



Thanks!

Any questions?

Email me at trungtt@soict.hust.edu.vn