

The slide features the logo of the University of Information Technology (UIT) on the left, which consists of a blue circular emblem with a stylized 'U' and 'I' and the text 'UIT TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN' below it. On the right is a stylized 'CS' logo in blue, where the 'C' is a circle with a screwdriver handle and the 'S' is a circle with a screwdriver head, suggesting computer science.

## **Nội dung**

---

1. Vai trò của CTDL& GT.
2. Các tiêu chuẩn đánh giá CTDL
3. Kiểu dữ liệu.
4. Thuật toán và độ phức tạp.

## Các khái niệm cơ bản

- Chương trình máy tính
- Lập trình (Programming)
- Ngôn ngữ lập trình (Programming language)
- Trình thông dịch (Interpreter)
- Trình biên dịch (compiler)

## Khái niệm cơ bản

➤ **Chương trình máy tính, phần mềm (Computer Software):** Danh sách **các câu lệnh, chỉ thị (Instruction)** để máy tính **thực hiện một chức năng** nào đó.

```
graph TD; Software[Software] --> SystemSoftware[System Software]; Software --> ApplicationSoftware[Application Software]; SystemSoftware --> OperatingSystems[Operating Systems]; SystemSoftware --> SystemSupport[System Support]; SystemSoftware --> SystemDevelopment[System Development]; ApplicationSoftware --> GeneralPurpose[General Purpose]; ApplicationSoftware --> ApplicationSpecific[Application Specific];
```


## TƯ DUY LẬP TRÌNH ?

```
graph TD; INPUT((INPUT)) -- blue arrow --> PROCESS((PROCESS)); PROCESS -- green arrow --> OUTPUT((OUTPUT)); OUTPUT -- orange arrow --> INPUT;
```


Phương pháp giải quyết một bài toán trên máy tính

5

# NGÔN NGỮ LẬP TRÌNH VS NGÔN NGỮ MÁY?

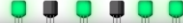


**MicroTransistors**



**Binary!**

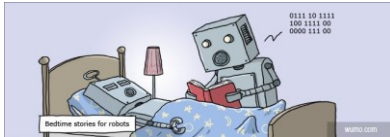
1 0 1 0 1 1



6

### NGÔN NGỮ MÁY?

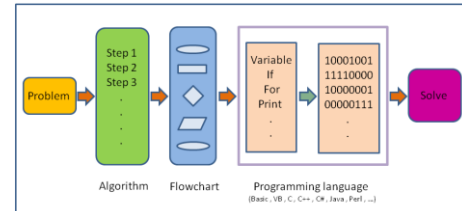
- **Ngôn ngữ máy (machine language)** là các chỉ thị dưới dạng nhị phân, can thiệp trực tiếp vào trong các mạch điện tử.
- Chương trình được viết bằng ngôn ngữ máy thì **có thể được thực hiện ngay không cần qua bước trung gian** nào.



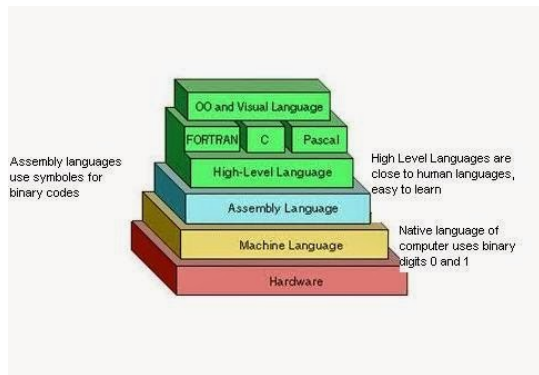
7

### NGÔN NGỮ LẬP TRÌNH?

- Programming language là **ngôn ngữ dùng để viết các chương trình cho máy tính**.
- Cũng như các ngôn ngữ thông thường, NNLT cũng có từ vựng, cú pháp và ngữ nghĩa.



8



9

### NGÔN NGỮ MÁY

- **Ngôn ngữ máy (machine language)** là các chỉ thị dưới dạng nhị phân, can thiệp trực tiếp vào trong các mạch điện tử.
- Chương trình được viết bằng ngôn ngữ máy thì **có thể được thực hiện ngay không cần qua bước trung gian** nào.



#### Machine Language

Language directly understood by the computer

binary code

• VD:

```

1 00000000 00000100 0000000000000000
2 01011110 00001100 11000010 0000000000000010
3 11101111 00010110 000000000000000101
4 11101111 10011110 0000000000000001011
5 11111000 10101101 11011111 0000000000010010
6 01100010 11011111 000000000000010101
7 11101111 00000010 11110111 0000000000010111
8 11110100 10101101 11011111 0000000000011110
9 00000011 10100010 11011111 000000000100001
10 11101111 00000010 11110111 000000000100100
11 01111110 11110100 10101101
12 11111000 10101110 11000101 000000000101011
13 00000110 10100010 11110111 000000000110001
14 11101111 00000010 11110111 000000000110100
15 01010000 11010100 000000000111011
16 00000100 000000000111011
  
```

Tuy nhiên chương trình viết bằng ngôn ngữ máy **đễ sai sót, cồng kềnh và khó đọc, khó hiểu** vì toàn những con số 0 và 1.

### NGÔN NGỮ CẤP THẤP – HỢP NGỮ

- Hợp ngữ (assembly language) được thiết kế để máy tính trở nên thân thiện hơn với người sử dụng.
- Các câu lệnh bao gồm hai phần: **phần mã lệnh (viết tựa tiếng Anh) chỉ phép toán cần thực hiện và phần tên biến chỉ địa chỉ chứa toán hạng của phép toán đó**.



#### Machine Language

Language directly understood by the computer

binary code

#### Symbolic Language

English-like abbreviations representing elementary computer operations

assembly language

• VD

```

1      entry    main, 'm<r2>
2      subl2   #12, sp
3      jsb     CSMAIN_ARGS
4      movab   $CHAR_STRING_CON
5
6      pushal  -8(fp)
7      pushal  (r2)
8      calls   #2, SCANF
9      pushal  -12(fp)
10     pushal  3(r2)
11     calls   #2, SCANF
12     mull3   -8(fp), -12(fp), -
13     pusha   6(r2)
14     calls   #2, PRINTF
15     crrl    r0
16     ret

```

Để máy thực hiện được một chương trình viết bằng hợp ngữ thì chương trình đó **phải được dịch sang ngôn ngữ máy**. Công cụ thực hiện việc dịch đó được gọi là Assembler

### NGÔN NGỮ CẤP THẤP – HỢP NGỮ

- Ngôn ngữ cấp cao (High level language): là ngôn ngữ được tạo ra và phát triển nhằm **phản ánh cách thức người lập trình nghĩ và làm**.
- Ngôn ngữ cấp cao rất gần với ngôn ngữ con người (Anh ngữ) nhưng chính xác như ngôn ngữ toán học.

**Machine Language**  
Language directly understood by the computer

*binary code*

**Symbolic Language**  
English-like abbreviations representing elementary computer operations

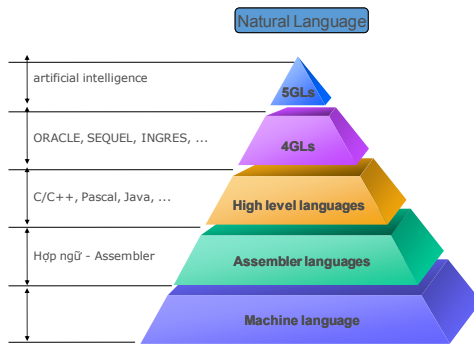
*assembly language*

**High-level Language**  
Close to human language.

Example:  $a = a + b$   
[add values of  $a$  and  $b$ , and store the result in  $a$ , replacing the previous value]

*C, C++, Java, Basic*

### Các lớp ngôn ngữ lập trình



9/15/2018

thuonght@ut.edu.vn

15

### Khái niệm cơ bản

- Lập trình (programming)**: viết chương trình cho máy tính.
- Lập trình viên (programer)**: người **sử dụng ngôn ngữ lập trình để biên soạn các chương trình**.



### Khái niệm cơ bản

- Mã nguồn**: chương trình được thể hiện bằng NNLT.
- Mã máy**: chương trình bằng ngôn ngữ Máy

#### Source code

```

if (!PREACTION(gm)) {
    void* mem;
    size_t nb;
    if (Bytes <= MAX_SMALL_REQUEST)
        bindmap_t idx;
        binmap_t smallbits;
        nb = (Bytes < MIN_REQUEST)? M
        idx = small_index(nb);
        smallbits = gm->smallmap >> i
    if ((smallbits & 0x30) != 0)
        mchunkptr b, p;
        idx += -smallbits & 1;
        b = smallbin_at(gm, idx);
        p = b->fd;
        assert(chunksize(p) == smal
        unlink_first_small_chunk(gm

```



compiler reordering

#### Machine code

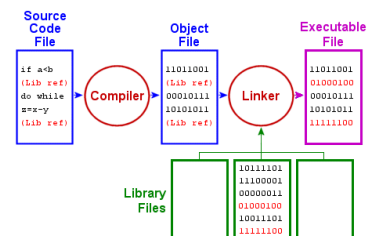
```

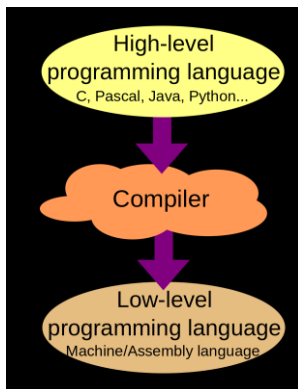
cmp     esi,0F4h
ja      dimalloc+1AEh (777CEh)
cmp     esi,00h
jbe     dimalloc+20h (77640h)
mov     esi,10h
jmp     dimalloc+26h (77646h)
and     esi,00h
and     esi,0FFFFFFFh
mov     ecx,dword ptr [__fmode+2
edi,esi
shr     edi,3
mov     ecx,edi
shl     ecx,cl
shr     test
dimalloc+5Ah (776BAh)
je      not
not     eax
and     eax,1
add     esi,ecx
mov     esi,dword ptr fmode+50

```

### Khái niệm cơ bản

- Chương trình đích**: dịch chương trình từ mã nguồn thành các ngôn ngữ ở cấp thấp hơn- thường là ngôn ngữ máy

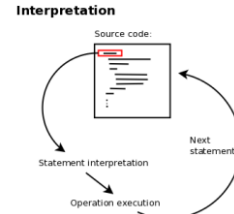




19

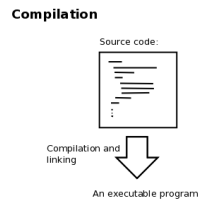
### THÔNG DỊCH(interpreted)

- Khi chương trình chạy đến dòng lệnh nào sẽ chuyển thành mã máy đến đó để máy tính có thể thực thi.
- Bộ thông dịch thực hiện quá trình thông dịch gọi là interpreter.

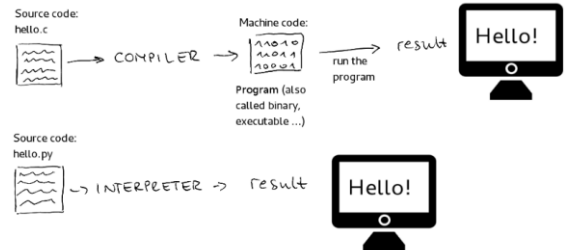


### BIÊN DỊCH (compiled)

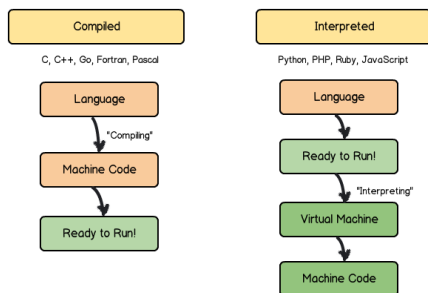
- Chương trình sẽ dịch toàn bộ thành mã máy rồi mới tiến hành thực thi.
- Bộ biên dịch thực hiện quá trình biên dịch được gọi là compiler



### Compiled vs interpreted



### Compiled vs interpreted

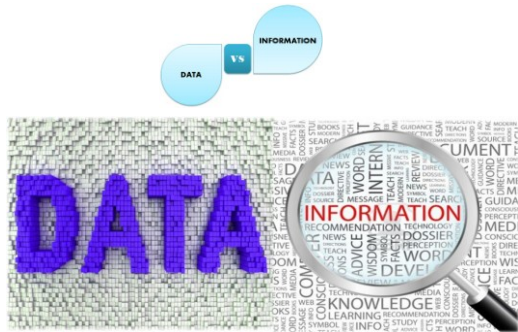


### Giải Bài Toán Trên Máy Tính



20

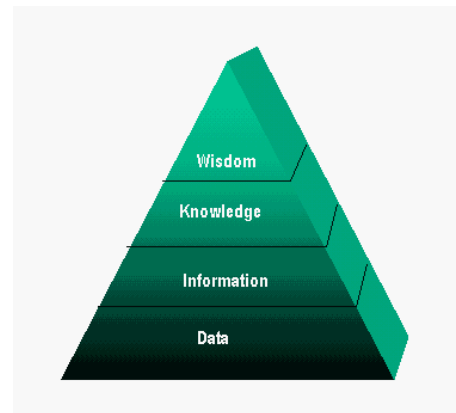
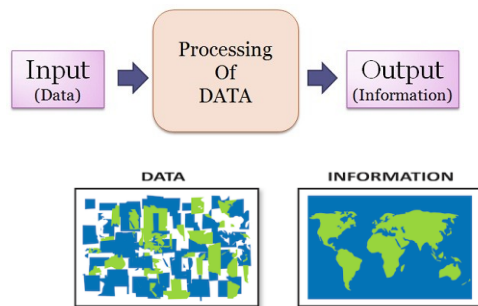
## DỮ LIỆU VS THÔNG TIN



## DỮ LIỆU (DATA)

- Theo *từ điển Tiếng Việt*: số liệu, tư liệu đã có, được dựa vào để giải quyết vấn đề
- *Tin học*: Biểu diễn các thông tin từ thế giới thực cần thiết cho bài toán được đưa vào máy tính.

## THÔNG TIN (INFORMATION)



## Vấn đề - Bài Toán

### • Biểu diễn vấn đề-bài toán

- $A \rightarrow B$
- 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 (input)**, B là **đầu ra (output)**

29

## VAI TRÒ CỦA CTDL & GT



### VAI TRÒ CỦA CTDL & GT

1. Tổ chức biểu diễn các đối tượng thực tế
  - ❖ Nhận định: Dữ liệu của con người đa dạng nhưng dữ liệu trên máy thì hạn chế và đơn giản  
→ **biểu diễn dữ liệu của con người lên trên máy tính?**
  - ❖ Người LT phải thực hiện việc thiết kế, xây dựng các cấu trúc **thích hợp nhất** để biểu diễn dữ liệu  
→ **xây dựng CTDL cho bài toán**
2. Xây dựng thao tác xử lý dữ liệu
  - ❖ Từ yêu cầu xử lý thực tế, xác định **trình tự các thao tác** máy tính phải thi hành để cho ra kết quả mong muốn → **xây dựng giải thuật cho bài toán**

Ngoài CTDL & GT cũng đóng **vai trò thể hiện giải pháp**

### MỐI QUAN HỆ GIỮA CTDL & GT

CTDL + Giải thuật = Chương trình



- **Cấu trúc dữ liệu:** (có thể hiểu là) cách tổ chức dữ liệu, cách mô tả bài toán dưới dạng NNLT
- **Giải thuật:** một quy trình để thực hiện một công việc xác định.

### CTDL Là gì ?

Cấu trúc dữ liệu

*Các mô hình dữ liệu, tổ chức dữ liệu (khai báo, lưu trữ dữ liệu) để biểu diễn dữ liệu trừu tượng hóa.*

Mô hình:

- Diễn đạt toán học
- Diễn đạt bằng các sơ đồ, biểu đồ

### Các Tiêu chuẩn của CTDL

1. **Phản ánh đúng thực tế:** phải biểu diễn đầy đủ thông tin, thể hiện chính xác đối tượng thực tế
2. **Hiệu quả lưu trữ:** tiết kiệm tài nguyên hệ thống
3. **Hiệu quả xử lý:** đáp ứng việc thiết kế hiệu quả, phải phù hợp với các thao tác trên đó, phù hợp với điều kiện cho phép của NNLT.

### Biểu diễn kiểu dữ liệu

$T = \langle V, O \rangle$   
 $V = \{\text{Tập các giá trị}\}$   
 $O = \{\text{Tập các thao tác xử lý}\}$

Ví dụ: Kiểu dữ liệu số nguyên **int** trong ngôn ngữ C

$T = \text{int}$   
 $V = \{-32768, 32767\}$   
 $O = \{+, -, *, /, \%\}$

35

### Các thuộc tính của kiểu dữ liệu

- Tên KDL
- Miền giá trị
- Kích thước lưu trữ
- Tập các toán tử tác động lên KDL

### Các Kiểu Dữ Liệu Cơ Bản

Kiểu dữ liệu phần cứng

Kiểu dữ liệu cơ bản

Kiểu dữ liệu có cấu trúc

Kiểu dữ liệu trừu tượng

### Kiểu dữ liệu phần cứng (hardware data type)

Ở cấp phần cứng máy tính:

- Xây dựng sẵn 1 tập kiểu dữ liệu phần cứng (sơ cấp): bit, byte, kiểu số nhị phân, kiểu ký tự, ...
- Cơ chế điều khiển: tuần tự và rẽ nhánh.
- Các tác vụ phần cứng: như cộng 2 số ở 2 địa chỉ bộ nhớ khác nhau rồi lưu kết quả tại 1 địa chỉ khác, dời 1 byte, ...

→ chỉ phù hợp cho máy chứ không phù hợp với con người

### Các kiểu dữ liệu cơ bản (basic data type)

- Loại dữ liệu đơn giản, không có cấu trúc, giá trị dữ liệu là **đơn nhất**.
- Các NNLT xây dựng sẵn như một thành phần của ngôn ngữ (kiểu dữ liệu định sẵn).
- Tùy NNLT, các kiểu dữ liệu có thể khác nhau đôi chút.
- Bao gồm:
  - Kiểu có thứ tự rời rạc: số nguyên, ký tự, boolean, liệt kê, ...
  - Kiểu không rời rạc: số thực

### Các kiểu dữ liệu cơ bản (basic data type)

- Cơ chế điều khiển:
  - Tuần tự
  - Chọn lựa/rẽ nhánh (if, if/else, switch)
  - Lặp (for, while, do ...while)
- Các tác vụ cài đặt trong NNLT: hàm thư viện, chương trình con (hàm/thủ tục)
- Khi dịch chương trình viết bằng NNLT, các kiểu dl cơ bản và tác vụ của nó chuyển dịch về các kiểu dl sơ cấp và tác vụ phần cứng để máy tính có thể thực thi được

### Ví dụ basic data type

Data type	Size	Value range
char	1 byte	-128 đến 127 hoặc 0 đến 255 (Ký tự dạng mã ASCII)
unsigned char	1 byte	0 đến 255
signed char	1 byte	-128 đến 127
int	2 hoặc 4 bytes	-32,768 đến 32,767 hoặc -2,147,483,648 đến 2,147,483,647
unsigned int	2 hoặc 4 bytes	0 đến 65,535 hoặc 0 đến 4,294,967,295
short	2 bytes	-32,768 đến 32,767
unsigned short	2 bytes	0 đến 65,535
long	4 bytes	-2,147,483,648 đến 2,147,483,647
unsigned long	4 bytes	0 đến 4,294,967,295

### Kiểu dữ liệu có cấu trúc

- Do người dùng định nghĩa.
- Gồm nhóm/liên kết các thành phần dữ liệu có kiểu dữ liệu đã được định nghĩa thành **một kiểu dữ liệu phức hợp nhiều thành phần**.

### Kiểu dữ liệu có cấu trúc



### Kiểu dữ liệu trừu tượng (Abstract data type - ADT)

- Thông tin ngoài trạng thái tĩnh (dữ liệu) còn ngầm định các hoạt tính của nó (tác vụ).  
VD: Phân/số, Tác vụ nhân
- Để xây dựng những kiểu dữ liệu mới phức tạp hơn người ta dùng 1 công cụ gọi là **trừu tượng hóa**. **Kết quả của quá trình trừu tượng hóa** là hình thành một kiểu dữ liệu mới gọi là kiểu dữ liệu trừu tượng.

### Kiểu dữ liệu trừu tượng (Abstract data type - ADT)

#### Trừu tượng hóa

Ý niệm về sự vật, hiện tượng sau khi thu thập chất lọc những thông tin có nghĩa, loại bỏ những thông tin không cần thiết, không quan trọng

- Trừu tượng hóa dữ liệu
- Trừu tượng hóa chức năng

### Kiểu dữ liệu trừu tượng (Abstract data type - ADT)

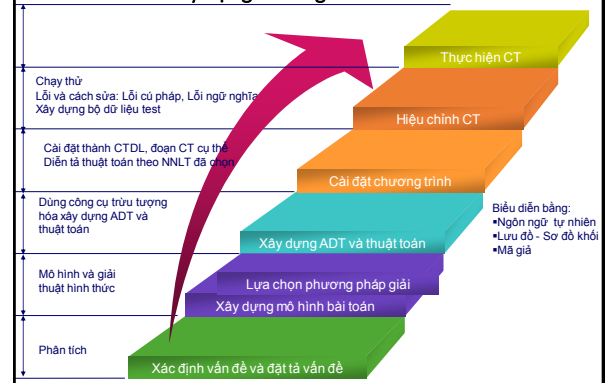
- Mỗi kiểu dữ liệu trừu tượng có *mô tả dữ liệu* và các *tác vụ liên quan*.
- Vấn đề tiếp theo là cài đặt nó thành CTDL và các đoạn chương trình.
- Các kiểu dữ trừu tượng và CTDL thông dụng:
 

- Ngăn xếp (stack)	hàng đợi (queue)
- danh sách (list)	cây (tree)
- bảng băm (hash table)	đồ thị (graph)

### Kiểu dữ liệu trừu tượng (Abstract data type - ADT)

- ADT là 1 kiểu dữ liệu do ta định nghĩa ở mức khái niệm, chưa được cài đặt cụ thể bằng NNLT.
- Khi cài đặt 1 ADT trên 1 NNLT cụ thể, phải làm 2 động tác:
  1. **Biểu diễn kiểu dữ liệu** = 1 CTDL hoặc ADT khác đã được cài đặt
  2. **Viết các CT con** thực hiện các phép toán tên kiểu dữ liệu này

### Các bước xây dựng chương trình





### Thuật toán - Algorithm

- Là **tập hợp (dãy) hữu hạn các chỉ thị** (hành động) được **định nghĩa rõ ràng** nhằm **giải quyết một bài toán cụ thể** nào đó.
- Thuật toán để giải một bài toán là một **dãy hữu hạn các thao tác được sắp xếp theo một trình tự xác định** sao cho sau khi thực hiện dãy thao tác đó, **từ Input của bài toán, ta nhận được Output cần tìm**.

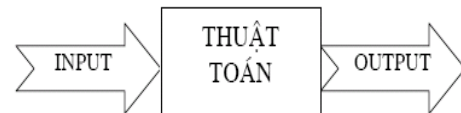
### Các tính chất - yêu cầu của thuật toán

- **Tính chính xác/đúng:**
  - Quá trình tính toán hay các thao tác máy tính thực hiện là chính xác.
  - Khi kết thúc, giải thuật phải cung cấp kết quả đúng đắn.
- **Tính phổ dụng/tổng quát:**
  - Có thể áp dụng cho một lớp các bài toán có đầu vào tương tự nhau.
- **Tính kết thúc/hữu hạn:**
  - Thuật toán phải dừng sau một số bước hữu hạn.

### Các tính chất - yêu cầu của thuật toán

- **Tính rõ ràng/hiệu quả:**  
Các câu lệnh minh bạch được sắp xếp theo thứ tự nhất định. Tối ưu về mặt thời gian và không gian.
- **Tính khách quan/xác định:**
  - Được viết bởi nhiều người trên máy tính nhưng kết quả phải như nhau.
  - Trong cùng một điều kiện hai bộ xử lý cùng thực hiện, thuật toán phải cho những kết quả giống nhau.

### Đặc tả một thuật toán



### Đặc tả một thuật toán

- Dữ liệu vào.
- Điều kiện ràng buộc (nếu có).
- Các sản phẩm ,kết quả (xuất).
- Các yêu cầu trên sản phẩm, kết quả.

### Các phương pháp biểu diễn thuật toán

1. Dùng ngôn ngữ tự nhiên.
2. Dùng mã giả (pseudocode)
3. Dùng lưu đồ - sơ đồ khối (flowchart)
4. Ngôn ngữ lập trình (chương trình)

### 1. Dùng ngôn ngữ tự nhiên

Đầu vào:  $a, b$  thuộc  $\mathbb{R}$

Đầu ra: nghiệm phương trình  $ax + b = 0$

1. Nhập 2 số thực  $a$  và  $b$ .
2. Nếu  $a = 0$  thì
  - 2.1. Nếu  $b = 0$  thì
    - 2.1.1. Phương trình vô số nghiệm
    - 2.1.2. Kết thúc thuật toán.
  - 2.2. Ngược lại
    - 2.2.1. Phương trình vô nghiệm.
    - 2.2.2. Kết thúc thuật toán.
3. Ngược lại
  - 3.1. Phương trình có nghiệm.
  - 3.2. Giá trị của nghiệm đó là  $x = -b/a$
  - 3.3. Kết thúc thuật toán.

### 1. Dùng ngôn ngữ tự nhiên

- Sử dụng ngôn ngữ thường ngày để liệt kê các bước của thuật toán.
- Phương pháp biểu diễn này **không yêu cầu người viết thuật toán** cũng như người đọc thuật toán phải nắm các quy tắc.
- Tuy vậy, cách biểu diễn này:
  - Thường **dài dòng**,
  - Không thể hiện rõ **cấu trúc của thuật toán**,
  - Đôi lúc **gây hiểu lầm hoặc khó hiểu cho người đọc**.
- Gần như không có một quy tắc cố định nào trong việc thể hiện thuật toán bằng ngôn ngữ tự nhiên.

### 2. Dùng mã giả

Đầu vào:  $a, b$  thuộc  $\mathbb{R}$

Đầu ra: nghiệm phương trình  $ax + b = 0$

```

If a = 0 Then
  Begin
    If b = 0 Then
      Xuất "Phương trình vô số nghiệm"
    Else
      Xuất "Phương trình vô nghiệm"
    End
  Else
    Xuất "Phương trình có nghiệm x = -b/a"
  
```

57

### 2. Dùng mã giả

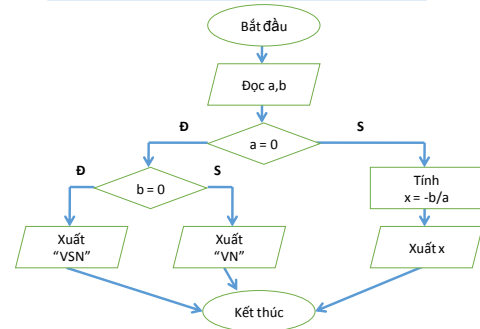
- Ngôn ngữ **tựa ngôn ngữ lập trình**:
  - Dùng **cấu trúc chuẩn hóa**, chẳng hạn tựa Pascal, C.
  - Dùng các **ký hiệu toán học, biến, hàm**.
- Ưu điểm**:
  - Đỡ công kênh
- Nhược điểm**:
  - Không trực quan.

### 3. Dùng lưu đồ

Là phương thức để **biểu diễn thuật toán thông qua các ký hiệu hình học**



### 3. Dùng lưu đồ



60

## Thuật toán - Algorithm

- ❖ Quy trình thiết kế thuật toán
  - Khảo sát, phân tích
  - Thiết kế (CTDL, thuật toán)
  - Mã hóa, viết chương trình
  - Kiểm tra
  - Thực hiện
  - Bảo trì, phát triển
- ❖ Kỹ thuật thiết kế thuật toán
  - Vết cạn
  - Đệ qui
  - Chia để trị
  - Quy hoạch động

## Đánh giá độ phức tạp của thuật toán



## Đánh giá độ phức tạp của thuật toán

- Là công việc ước lượng **thời gian thực hiện của thuật toán** để so sánh tương đối các thuật toán với nhau

→ *Làm sao ước lượng thời gian thực hiện của thuật toán?*

## Đánh giá độ phức tạp của thuật toán

- ❑ Phương pháp thực nghiệm (thời gian)



- ❑ Phương pháp xấp xỉ (số bước thực hiện)



## Đánh giá độ phức tạp của thuật toán

- Cài thuật toán rồi chọn các bộ dữ liệu thử nghiệm.
- Thống kê các thông số nhận được khi chạy các bộ dữ liệu đó.
- **Đơn vị đo:** giờ, phút, giây.
- **Ưu điểm:** Dễ thực hiện.
- **Nhược điểm:**

- Chịu sự hạn chế của **ngôn ngữ lập trình**.
- Ảnh hưởng bởi **trình độ của người lập trình**.
- **Chọn được các bộ dữ liệu thử đặc trưng** cho tất cả tập các dữ liệu vào của thuật toán: **khó khăn và tốn nhiều chi phí**.
- **Phụ thuộc vào phần cứng** (cấu hình máy).



## Đánh giá độ phức tạp của thuật toán

- Đơn vị đo thời gian thực hiện = **số các lệnh được thực hiện trong một máy tính lý tưởng**

VD: Tính tổng các số nguyên dương từ 1 → n

```
int Tong (int n){
    int S=0;
    for (int i = 1; i<=n; i++)
        S = S+i;
    return S;
}
```



*Bao nhiêu lệnh?*

### Đánh giá độ phức tạp của thuật toán

- Thời gian thực hiện chương trình là một hàm phụ thuộc **kích thước dữ liệu** vào:
  - ký hiệu  **$T(n)$** :
  - $n$**  : kích thước (độ lớn) của dữ liệu vào.
  - $T(n) \geq 0 \forall n \geq 0$ .
  - Ví dụ: Thời gian thực hiện CT là  $T(n) = cn$ ,  $c$  : hằng số, nghĩa là CT cần  $cn$  chỉ thị thực thi

### Đánh giá độ phức tạp của thuật toán

- Thời gian thực hiện CT không chỉ phụ thuộc vào **kích thước** mà còn phụ thuộc vào **tính chất của dữ liệu** vào.  
 Ví dụ: - Sắp xếp dãy số nguyên tăng dần  $\rightarrow$  *phụ thuộc vào thứ tự dãy*  
           - Tìm phần tử  $x$  trong mảng  $a[]$
- Thường xem  $T(n)$  như là thời gian thực hiện trong **trường hợp xấu nhất** (lớn nhất) trên dữ liệu vào có kích thước  $n$ .

### Quy tắc đánh giá độ phức tạp thuật toán

- Đặt vấn đề:
  - $P1, T1(n) = 100n^2$
  - $P2, T2(n) = 5n^3$
 Giải thuật nào nhanh hơn?  
 Trả lời: nếu  $n < 20$  thì P2 nhanh hơn,  $n > 20$  thì ngược lại
- Ta xét tỷ suất tăng của  $T(n)$  thay vì xét chính  $T(n)$**   
 $\rightarrow$  Giải thuật có độ phức tạp là  $f(n)$ , Ký hiệu:  $T(n) = O(f(n))$

### Quy tắc đánh giá độ phức tạp thuật toán

- Tỷ suất tăng :  
 Ta nói  $T(n)$  có tỷ suất tăng  **$f(n)$**  nếu tồn tại các hằng số  $c$  và  $n_0$  sao cho  $T(n) \leq c \cdot f(n)$  với mọi  $n \geq n_0$ .
- Ví dụ:  
 $T(n) = 3n^3 + 2n^2$   
 $\rightarrow$  tỷ suất tăng  $f(n) = n^3$   
 (vì với mọi  $n \geq 0$  thì  $3n^3 + 2n^2 \leq 5n^3$ ,  $c = 5, n_0 = 0$ )

### Đánh giá độ phức tạp của thuật toán

$\left. \begin{array}{l} O(\log_2 n) \\ O(n) \\ O(n \log_2 n) \\ O(n^2) \\ O(n^k) \end{array} \right\}$	độ phức tạp đa thức $\Rightarrow$ chấp nhận được
$\left. \begin{array}{l} O(2^n) \\ n! \\ n^n \end{array} \right\}$	độ phức tạp cao $\Rightarrow$ khó chấp nhận

### Đánh giá độ phức tạp của thuật toán

	$n$	$n \log_2 n$	$n^2$	$n^3$	$1.5^n$	$2^n$	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	$10^{25}$ years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	$10^{17}$ years	very long
$n = 1,000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10,000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100,000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1,000,000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

## Quy tắc xác định độ phức tạp của thuật toán

### Quy tắc bỏ hằng số

Nếu đoạn chương trình P có thời gian thực hiện :

$$T(n) = O(c1.f(n))$$

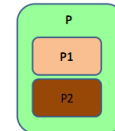
với c1 là một hằng số dương

=> thì có thể coi đoạn chương trình đó có độ phức tạp tính toán là  $O(f(n))$ .

## Quy tắc xác định độ phức tạp của thuật toán

### Nguyên tắc cộng:

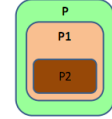
- Nếu: chương trình P gồm 2 đoạn chương trình P1 ( $O_1(n)$ ) và P2 ( $O_2(n)$ ) chạy nối tiếp nhau



- Thì: độ phức tạp của CT P:  
 $O(n) = \max(O_1(n), O_2(n))$

### Nguyên tắc nhân:

- Nếu: chương trình P gồm 2 đoạn chương trình P1 ( $O_1(n)$ ) và P2 ( $O_2(n)$ ) lồng nhau



- Thì: độ phức tạp của CT P:  
 $O(n) = O_1(n).O_2(n)$

## Quy tắc tổng quát

- Mỗi lệnh gán, cout, cin :  $O(1)$
- Một chuỗi tuần tự các lệnh : qui tắc cộng → thời gian thi hành một lệnh nào đó lâu nhất.
- Cấu trúc IF: thời gian lớn nhất thực hiện lệnh sau IF hoặc sau ELSE và thời gian kiểm tra điều kiện (thường là  $O(1)$ ).
- Vòng lặp: tổng (trên tất cả các lần lặp) thời gian thực hiện thân vòng lặp.
- Nếu thời gian thực hiện thân vòng lặp không đổi thì thời gian thực hiện vòng lặp là tích của số lần lặp với thời gian thực hiện thân vòng lặp.

75

## Ví dụ

### Vòng lặp

```

1 for (i = 0; i < n; i++)
2 {
3     x = a[i]/2;
4     a[i] = x + 1;
5 }
  
```

- Có 4 thao tác cơ bản ở các dòng 3 và 4 gồm 2 phép gán, 1 phép chia và 1 phép cộng
- Cả 4 thao tác đó được lặp lại n lần
- Thời gian chạy:  $t(n) = 4n = O(n)$

*Chú ý: Ở đây, ta bỏ qua 3 thao tác cơ bản điều khiển quá trình lặp ở dòng 1. Kết quả phân tích thuật toán sẽ không thay đổi nếu tính thêm cả 3 thao tác đó.*

## Ví dụ

### Vòng lặp có câu lệnh break

```

1 for (i = 0; i < n; i++)
2 {
3     x = a[i]/2;
4     a[i] = x + 1;
5     if (a[i] > 10) break;
6 }
  
```

- Có 5 thao tác cơ bản ở các dòng 3, 4, 5 gồm 2 phép gán, 1 phép chia, 1 phép cộng và 1 phép so sánh
- Không thể đếm chính xác số lần thực hiện 5 thao tác đó vì ta không biết khi nào điều kiện  $a[i] > 10$  xảy ra
- Trong trường hợp tồi nhất, tức là điều kiện  $a[i] > 10$  xảy ra ở bước lặp cuối cùng hoặc không xảy ra, cả 5 thao tác cơ bản được lặp lại n lần
- Thời gian chạy (trong trường hợp tồi nhất):  $t(n) = 5n = O(n)$

## Ví dụ

### Các vòng lặp tuần tự

```

for (i = 0; i < n; i++)
{
    ... // 3 thao tác cơ bản ở đây
}
for (i = 0; i < n; i++)
{
    ... // 5 thao tác cơ bản ở đây
}
  
```

### Ví dụ

#### Các vòng lặp lồng nhau

```
for (i = 0; i < n; i++)
{
    ... // 2 thao tác cơ bản ở đây
    for (j = 0; j < n; j++)
        ... // 3 thao tác cơ bản ở đây
}
```

- Phân tích các vòng lặp từ trong ra ngoài:
  - Vòng lặp bên trong thực hiện  $3n$  thao tác cơ bản
  - Mỗi bước lặp của vòng lặp bên ngoài thực hiện  $2 + 3n$  thao tác cơ bản
- Thời gian chạy tổng thể:  $t(n) = (2 + 3n)n = 3n^2 + 2n = O(n^2)$

### Ví dụ

#### Câu lệnh if-else

```
1 if (x > 0)
2     i = 0;
3 else
4     for (j = 0; j < n; j++)
5         a[j] = j;
```

- Có 3 thao tác cơ bản:
  - Phép so sánh ở dòng 1 được thực hiện 1 lần
  - Phép gán ở dòng 2 được thực hiện 0 hoặc 1 lần
  - Phép gán ở dòng 5 được thực hiện 0 hoặc  $n$  lần
- Trong trường hợp tồi nhất (điều kiện  $x > 0$  sai), phép gán ở dòng 5 chạy  $n$  lần
- Thời gian chạy:  $t(n) = 1 + n = O(n)$

### Ví dụ

#### Hàm đệ quy

```
1 long factorial(int n)
2 {
3     if (n <= 1)
4         return 1;
5     else
6         return n * factorial(n - 1);
7 }
```

- Nếu  $n = 1$ , chỉ mất 1 phép so sánh ở dòng 3
- Nếu  $n > 1$ :
  - Dòng 3 có 1 phép so sánh
  - Dòng 6 có 1 phép nhân, 1 phép trừ và 1 lời gọi hàm đệ quy tốn thời gian  $t(n-1)$

### Ví dụ

- Suy ra thời gian chạy của thuật toán:

$$t(1) = 1 \quad (\text{với } n = 1)$$

$$t(n) = 3 + t(n-1) \quad (\text{với } n > 1)$$

$$= 3 + 3 + t(n-2)$$

$$= 3 + 3 + 3 + t(n-3)$$

...

$$= 3k + t(n-k)$$

- Chọn  $k = n - 1$ , khi đó:

$$t(n) = 3(n-1) + t(1) = 3n - 2 = O(n)$$

### Bài tập

1. Khái niệm và vai trò quan trọng của CTDL và thuật toán trong việc xây dựng chương trình máy tính? Các tiêu chuẩn của CTDL?
2. Thuật toán là gì? Các tính chất của thuật toán?
3. Vẽ flow chart của bài toán tìm số lớn trong hai số theo ba cách.
4. Khái niệm và ý nghĩa của "độ phức tạp" khi đề cập đến thuật toán.
5. Cách tính độ phức tạp của 1 thuật toán.

**Nộp trên Course – Xem thông báo trên moddle.**

