



# Phần 1: Máy tính và Lập trình



# Nội dung

I

**Một số khái niệm cơ bản**

II

**Ngôn ngữ lập trình C**


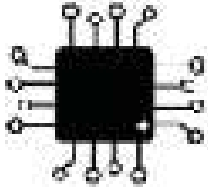

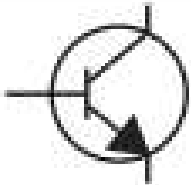
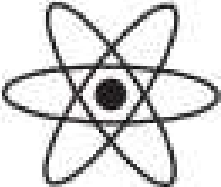


# Nội dung

I

**Một số khái niệm cơ bản**

# 1. Máy tính và hệ đếm nhị phân

Computer	
Integrated Circuit	
Logic Gates	
Transistors	
Electrons	

- ❖ Máy tính được hợp thành từ các mạch tích hợp (Integrated Circuit).
- ❖ Các mạch tích hợp được tạo thành từ các cổng logic (logic gate).
- ❖ Logic gate được tạo thành từ các Transistor.
- ✓ Transistor là tế bào của máy tính.
- ✓ Hoạt động của Transistor: **on/off** (1/0) → Nguồn gốc của việc máy tính sử dụng Hệ nhị phân !



## 2. Một số khái niệm về lập trình máy tính

- **Chương trình** máy tính (computer program)<sup>1</sup> là *chuỗi các chỉ dẫn* (instruction) có thể đưa được vào máy tính để cho nó thực hiện một hoạt động nào đó.
- **Lập trình** (programming)<sup>1,2</sup> là *hoạt động tạo ra một tập các chỉ dẫn* khiến máy tính thực hiện một nhiệm vụ cụ thể.
- **Ngôn ngữ lập trình** (program language)<sup>3</sup> là *bộ từ vựng và tập hợp các quy tắc ngữ pháp* dùng để chỉ dẫn máy tính hoặc thiết bị điện toán thực hiện các nhiệm vụ cụ thể.

1: Cambridge Dictionary

2: <https://www.khanacademy.org>

3: <https://www.webopedia.com/definitions/programming-language/>



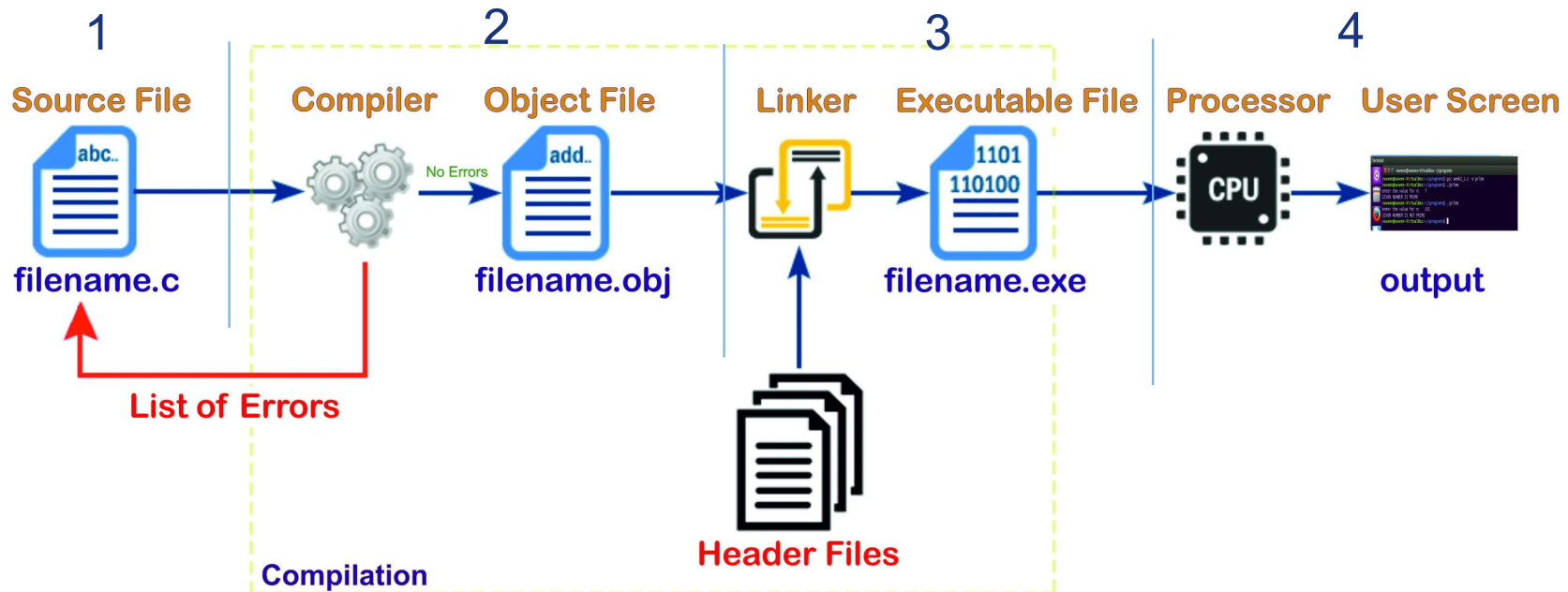
### 3. Mã nguồn, mã máy

- ❖ Máy tính chỉ hiểu ngôn ngữ máy/**mã máy** (machine language/***machine code***). Machine code là nhóm *các chỉ dẫn* cho máy tính thực thi. Mỗi chỉ dẫn là một chuỗi cấu thành từ các số nhị phân (0, 1).
- ❖ **Mã nguồn** (***source code***) là kịch bản để máy tính thực thi, được tạo ra bởi con người bằng ngôn ngữ bậc cao.
- Cần phải chuyển mã nguồn(source code) thành mã máy (machine code). Chương trình chuyển đó gọi là *translator*.
- Translator có 2 loại là: *compiler* hoặc *interpreter*.

*Source code* → *compiler/interpreter* → *machine code*

### 3. Mã nguồn, mã máy (tiếp)

- ❖ Ngôn ngữ C sử dụng compiler
- ❖ Chu trình để tạo và thực thi một chương trình C:  
(1) Soạn thảo/sửa chữa → (2) Dịch → (3) Liên kết → (4) Thực thi





## 4. Phân loại ngôn ngữ lập trình

Ngôn ngữ tự nhiên của máy tính là các con số nhị phân (0 và 1), khó tiếp cận với đa số người dùng. Để có thể tạo ra các chương trình cho máy tính thực thi cần phải có ngôn ngữ lập trình.

### ❖ Hợp ngữ

- Là các từ viết tắt, miêu tả các thao tác cơ bản của máy tính, dễ hiểu hơn so với ngôn ngữ máy.

✓ *Assembly*

### ❖ Ngôn ngữ bậc cao

- Được thiết kế để thân thiện, dễ tiếp cận đối với người và mang lại hiệu quả cao khi lập trình.

✓ *C/C++, Java, Python, Matlab, ...*





## 5. Các phương pháp lập trình

- ❖ Tuần tự
- ❖ Thủ tục
- ❖ Hướng đối tượng



## 6. Thuật toán

- Cho một **bài toán** nghĩa là cho Input và tìm Output của bài toán.
  - Có các dữ kiện gì ?
  - Ta phải làm gì ? làm thế nào ?
- ❖ **Thuật toán** là một tập hợp hữu hạn các *quy tắc/bước* làm việc với dữ kiện đầu vào để tìm kết quả đầu ra như dự đoán.
- ❖ Một chương trình hoạt động tốt phụ thuộc hoàn toàn vào thuật toán. Thiết kế thuật toán tốt không chỉ đảm bảo có đầu ra như mong muốn mà còn giúp sử dụng tài nguyên của máy tính một cách tối ưu, hiệu quả: *thời gian xử lý, bộ nhớ,...*



## 6.1. Ví dụ về thuật toán

❖ Thuật toán giải phương trình bậc 2:

$$f(x) = a * x^2 + b * x + c; \text{ (a,b,c là các số thực)}$$

trong tập số thực gồm các bước sau đây:

1. Nhập a, b, c

2. Nếu  $a == 0$

▪  $f(x)$  suy biến, có nghiệm duy nhất:  $x = -c/b$ ;

3. Nếu  $a \neq 0$

▪  $d = b * b - 4 * a * c$ ;

✓  $d < 0$ , phương trình vô nghiệm

✓  $d == 0$ , phương trình có nghiệm kép:  $x = -b/(2 * a)$

✓  $d > 0$ , phương trình có 2 nghiệm phân biệt:  $x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4 * a * c}}{2 * a}$



## 6.2. Các phương pháp biểu diễn thuật toán

### ❖ Liệt kê (List)

### ❖ Lưu đồ (Flowchart)

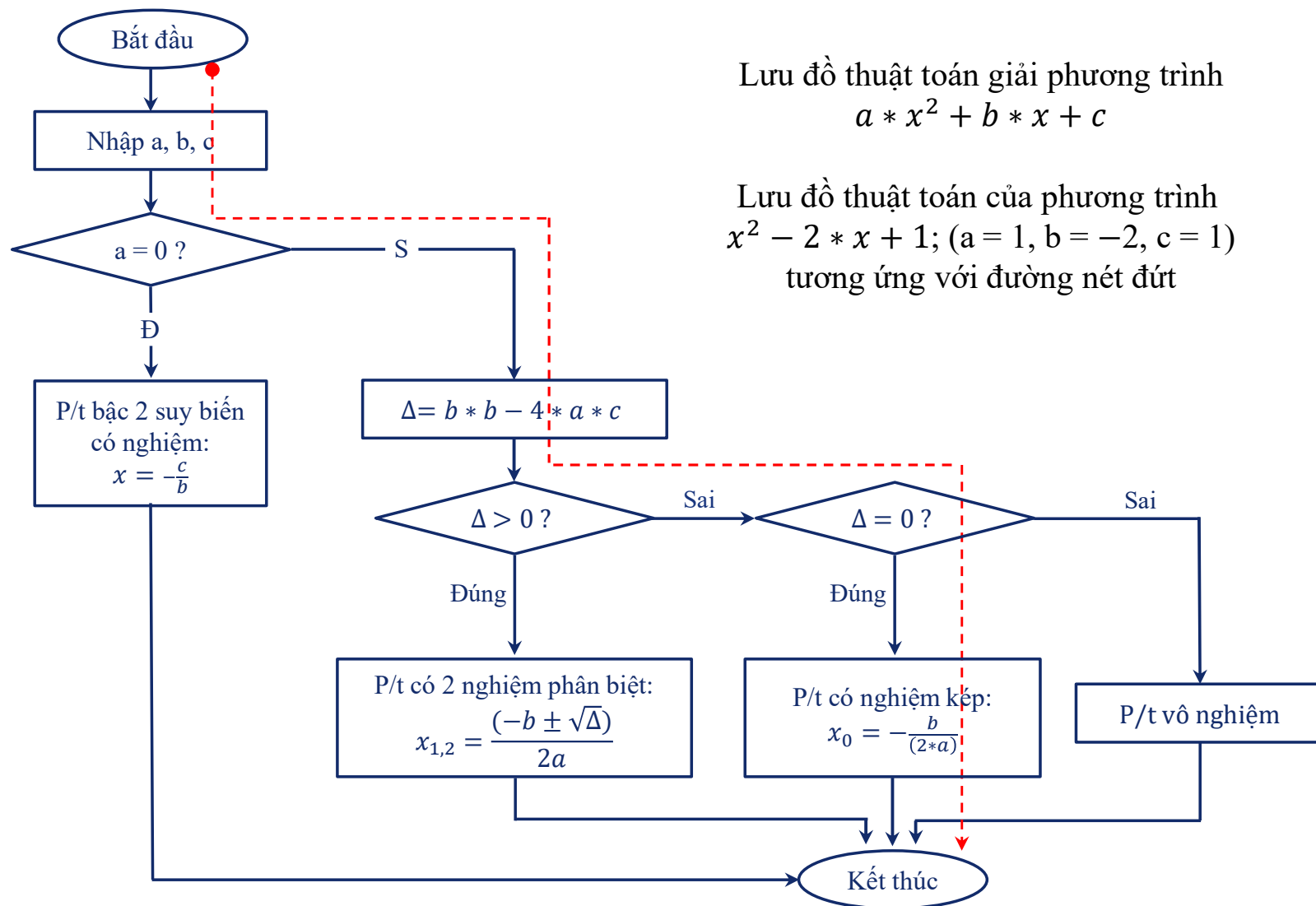
*(Mô tả thuật toán bằng bằng sơ đồ khối)*

- Lựa chọn
- Xử lý
- Đường đi
- Điểm bắt đầu, điểm cuối, điểm nối, điểm sang trang
- ...

### ❖ Mã giả (Pseudo-code)

- Là một cách mô tả lập trình không chính thức, không yêu cầu phải giống cú pháp của một ngôn ngữ lập trình cụ thể nào. Nó đơn giản chỉ là việc triển khai thuật toán dưới dạng văn bản thông tin bằng tiếng Anh. Nó không thể dịch được sang mã máy để thực thi.
- Có thể vay mượn một ít cú pháp cơ bản của một ngôn ngữ lập trình nào đó để thể hiện

## Ví dụ: Lưu đồ thuật toán giải p/t bậc 2





## Ví dụ: Mã giả thuật toán giải p/t bậc 2

**if** Delta > 0 **then begin**

$x_1 = (-b - \sqrt{\text{delta}}) / (2 * a)$

$x_2 = (-b + \sqrt{\text{delta}}) / (2 * a)$

xuất kết quả : phương trình có hai nghiệm là  $x_1$  và  $x_2$

**end**

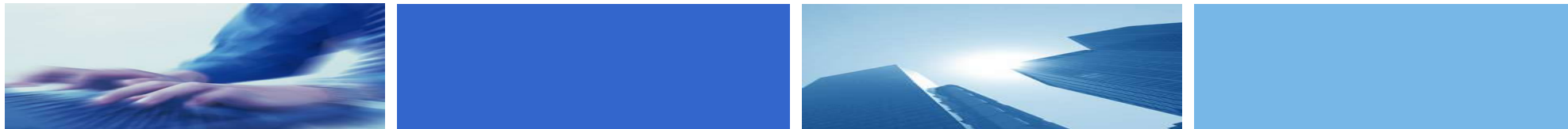
**else**

**if** delta = 0 **then**

xuất kết quả : phương trình có nghiệm kép là  $-b / (2 * a)$

**else** {trường hợp delta < 0 }

xuất kết quả : phương trình vô nghiệm



# Ưu nhược điểm của các phương pháp biểu diễn

## ❖ Liệt kê

- Sử dụng ngôn ngữ tự nhiên.
- Dài dòng, chỉ nên sử dụng cho các bài toán đơn giản.

## ❖ Lưu đồ thuật toán

- Dễ hình dung
- Công kênh (trong trường hợp muốn chi tiết bài toán)

## ❖ Mã giả

- Khả năng hiểu thuật toán và triển khai code của lập trình viên dễ dàng được thực hiện hơn.
- Không có định dạng, tiêu chuẩn cụ thể



II

## Ngôn ngữ lập trình C

1

Lịch sử phát triển

2

Các thành phần cơ bản của C





## 1.1. Lịch sử phát triển của ngôn ngữ C

- ❖ Dennis Ritchie (Bell lab) phát triển dựa trên:
  - ✓ ngôn ngữ BCPL (do Martin Richards đưa ra vào năm 1967)
  - ✓ ngôn ngữ B (do Ken Thompson phát triển từ ngôn ngữ BCPL vào năm 1970 khi viết hệ điều hành UNIX đầu tiên trên máy PDP-7)
- ❖ C được sử dụng đầu tiên trên hệ điều hành UNIX
- ❖ Được sử dụng rộng rãi như là công cụ chính để phát triển các phần mềm lớn và là ngôn ngữ lựa chọn dành cho người mới lập trình.



## 1.1. Lịch sử phát triển của ngôn ngữ C

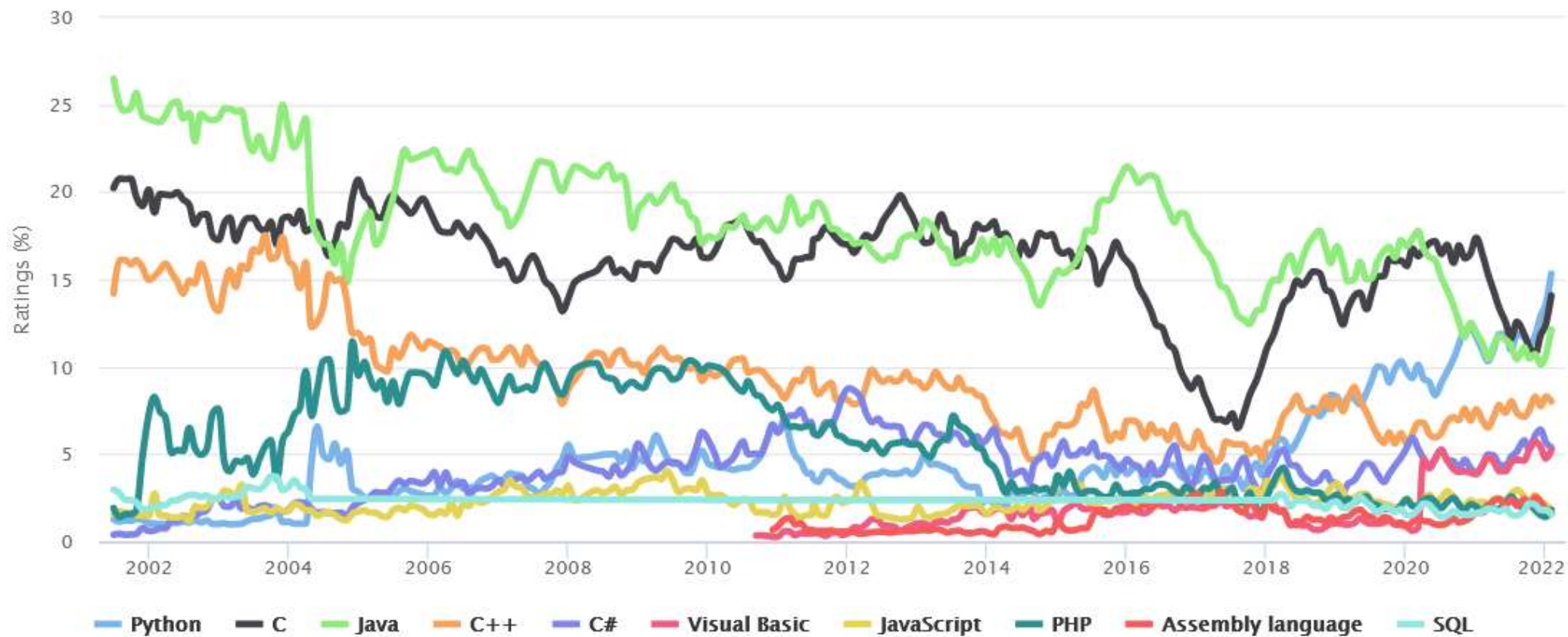
<b>1960</b>	<b>Algol</b>	• International Group
<b>1967</b>	<b>BCPL</b>	• Martin Richards
<b>1970</b>	<b>B</b>	• Ken Thomson
<b>1972</b>	<b>Traditional C</b>	• Dennis Ritchie
<b>1978</b>	<b>K&amp;R C</b>	• Kernighan & Ritchie
<b>1989</b>	<b>ANSI C</b>	• ANSI Commitee
<b>1990</b>	<b>ANSI/ISO C</b>	• ISO Commitee
<b>1999</b>	<b>C99</b>	• Standard Commitee

## 1.2. Các ngôn ngữ lập trình

02/2022

TIOBE Programming Community Index

Source: [www.tiobe.com](http://www.tiobe.com)



# 1.2. Các ngôn ngữ lập trình 02/2022

## ❖ IEEE spectrum

Rank	Language	Type	Score
1	Python	🌐 🖥️ ⚙️	100.0
2	Java	🌐 📱 🖥️	95.4
3	C	📱 🖥️ ⚙️	94.7
4	C++	📱 🖥️ ⚙️	92.4
5	JavaScript	🌐	88.1
6	C#	🌐 📱 🖥️ ⚙️	82.4
7	R	🖥️	81.7
8	Go	🌐 🖥️	77.7

## ❖ Github PYPL

Worldwide, Feb 2022 compared to a year ago:				
Rank	Change	Language	Share	Trend
1		Python	28.52 %	-1.7 %
2		Java	18.12 %	+1.2 %
3		JavaScript	8.9 %	+0.4 %
4	↑	C/C++	7.62 %	+1.1 %
5	↓	C#	7.39 %	+0.6 %
6		PHP	5.81 %	-0.3 %
7		R	4.04 %	+0.2 %
8		Objective-C	2.46 %	-1.1 %
9		Swift	2.03 %	+0.0 %
10		TypeScript	1.94 %	+0.2 %
11		Matlab	1.76 %	+0.0 %



## 2.1. Bộ ký tự

### ❖ Tập ký tự hợp lệ dùng trong ngôn ngữ C

- Các chữ cái thường:  $a, b, c, \dots, z$  (26 chữ cái)
- Các chữ in hoa:  $A, B, C, \dots, Z$
- Các chữ số :  $0, 1, \dots, 9$
- Các kí hiệu toán học:  $+, -, *, /, =, <, >, ()$
- Các kí hiệu khác:  $[ ], \{ \}, !, @, \#, \$, \%, \wedge, :, \%, ;, \dots$
- Khoảng trống (dấu cách):



## 2.2. Định danh (Identifier)

- ❖ Là một dãy kí tự bắt đầu bằng chữ cái hoặc dấu gạch dưới, theo sau là chữ cái, chữ số hoặc ký tự gạch nối. Được dùng để đặt **tên** cho *biến, hằng, hàm, từ khóa,...*
  - Từ khóa: tên bị chiếm dụng bởi ngôn ngữ lập trình
  - Tên chuẩn: chuẩn hóa, ai cũng biết(độ phổ biến cao)
  - Tên do người lập trình tự đặt
- ❖ Quy tắc đặt tên:
  - Bắt đầu bằng một chữ cái hoặc bằng dấu gạch dưới
  - Không có khoảng trống ở giữa các từ
  - Không được trùng với từ khóa
  - Chữ hoa và chữ thường được xem là khác nhau
  - Nên đặt chữ hoa cho các hằng
  - Không nên đặt trùng với tên chuẩn



## Từ khóa(keyword)

- ❖ Là các từ dành riêng cho ngôn ngữ lập trình
- ❖ Các từ khoá trong C: *break, char, continue, case, do, double, default, else, float, for, goto, int, if, long, return, struct, switch, unsigned, while, typedef, union, void, volatile, ...*





## 2.3. Các kiểu dữ liệu sơ cấp

- ❖ *Kiểu dữ liệu* là một dạng phân loại thông tin. Nó chỉ ra dạng thức của thông tin gồm: miền giá trị và kích thước bộ nhớ mà nó chiếm dụng. Ngôn ngữ C cung cấp một số kiểu dữ liệu sơ cấp sau.

Tên kiểu	Dung lượng bộ nhớ	Mô tả
char	1 byte	Ký tự
int	4 byte	Số nguyên
float	4 byte	Số thực
double	8 byte	Số thực lớn
void		Rỗng

- ❖ Các kiểu dữ liệu trên có thể dùng kết hợp với các *modifier* để thay đổi miền giá trị.
  - short (co hẹp miền giá trị)
  - long (mở rộng miền giá trị)
  - signed và unsigned (dịch chuyển/tĩnh tiến miền giá trị)
- ❖ Để kiểm tra kích thước tính bằng byte của một kiểu dữ liệu dùng hàm: **sizeof**





## Một số kiểu dữ liệu và phạm vi mô tả

Type	Bytes	Bits	Range
short int	2	16	-32,768 -> +32,767
unsigned short int	2	16	0 -> +65,535
unsigned int	4	32	0 -> +4,294,967,295
int	4	32	-2,147,483,648 -> +2,147,483,647
long int	4	32	-2,147,483,648 -> +2,147,483,647
signed char	1	8	-128 -> +127
unsigned char	1	8	0 -> +255
float	4	32	
double	8	64	
long double	12	96	



# Quy tắc chuyển kiểu dữ liệu

## ❖ Xét ví dụ:

```
int a = 5; float b = 3.2; int c = 7;
```

- $a+b \rightarrow$  kiểu gì ?
- $a/(\text{int})b \rightarrow ?$
- $(\text{int})b \rightarrow ?$
- $a/c \rightarrow ?$

## ❖ Một số quy tắc chuyển kiểu dữ liệu:

- Chuyển kiểu của biểu thức theo kiểu của biến có phạm vi biểu diễn lớn hơn.
- Chuyển kiểu qua phép gán: theo kiểu của thành phần bên trái phép gán
- Ép kiểu: chuyển tức thời, giá trị gốc của biến giữ nguyên kiểu

## 2.4. Các phép toán

Phép toán	Mô tả phép toán	Kết quả của phép toán	Ví dụ	
Số học	$+$ $-$ $*$ $/$ Chia lấy dư: $\%$	Số	$x = 5 + 6;$ $y = 7 \% 5;$	$x = 11;$ $y = 2;$
Gán	$=$	Giá trị bên phải của phép toán được gán cho vế trái	$a = 6; b = 5;$ $z = a + b;$	$a = 6; b = 5$ $z = 11;$
Quan hệ	So sánh hơn, kém: $>$ , $>=$ , $<$ , $<=$ So sánh bằng: $==$ Khác: $!=$	Biểu thức có giá trị: TRUE hoặc FALSE	$a = 6; b = 5;$ $a > b;$ $a == b;$ $a != b$	$a = 6; b = 5;$ TRUE FALSE TRUE
Tăng, giảm	$++$ , $--$ $+=$ , $-=$ , $*=$ , $/=$ , $\%=$	Số	$c = 2; d = 2; b = 2;$ $a = 3; e = 4; f = 6;$ $k = 3;$  $++a; \Leftrightarrow a = a + 1;$ $f++; \Leftrightarrow f = f + 1;$ <i>tăng rồi gán:</i> $x = ++c;$ <i>gán rồi tăng:</i> $y = d++;$ $z -= e; \Leftrightarrow z = e - 1;$ $b += 2; \Leftrightarrow b = b + 2;$ $x *= 2; \Leftrightarrow x = x * 2;$ $k \% = 5; \Leftrightarrow k = k \% 5;$	$a = 3 + 1 = 4;$ $f = 6 + 1 = 7;$ $c = 3; x = 3;$ $y = 2; d = 3;$ $e = 3; z = 3;$ $b = 2 + 2 = 4;$ $x = 3 * 2 = 6;$ $k = 3 \% 5 = 3;$

## 2.4. Các phép toán (tiếp)

Phép toán	Mô tả phép toán	Kết quả của phép toán	Ví dụ	
Logic (dành cho biểu thức)	!(NOT),   (OR), &&(AND)	TRUE(đúng), FALSE(sai)	$(x > 1) \&\& (x < 4)$ $(x > 5)    (x < 10)$	
Bit (dành cho số nhị phân)	&(AND)  (OR) ^(XOR) >> (dịch phải) << (dịch trái)	Số	001 & 111 001   101 001 ^ 100 00011101 >> 1 00011101 << 2	001 101 101 00001110 01110100
Vùng nhớ	& *	Lấy địa chỉ của biến Lấy thông tin lưu tại vùng nhớ mà con trỏ trỏ tới		
Điều kiện	A ? B : C	Nếu biểu thức A đúng thì thực hiện biểu thức B. Ngược lại, nếu biểu thức A sai thì thực hiện biểu thức C	$(x > 0) ? (x * x) : (x + 1)$	

❖ Thứ tự ưu tiên của phép toán: **xem Phụ lục.1.1 đính kèm bài giảng**



## Ví dụ về các phép toán

- $Z = 2;$
- $Z = N = 2;$
- $Z = Z + 3;$                        $//Z += 3;$
- $Y *= Z;$                        $//Y = Y * Z;$
- $B = Y++;$                        $B = ++Y;$
- $A = (5 \% 2 == 1) \&\& (6 != 3)$
- $X = 0101; Y = 0011; Z = X | Y$



## 2.5. Biến, Hằng, Biểu thức

- **Biến** là một *định danh* dùng để biểu đạt giá trị của thông tin. Trong quá trình thực thi chương trình, *giá trị của biến có thể thay đổi*.
- Về bản chất, mỗi biến gắn liền với một phân vùng bộ nhớ được cấp phát và thuộc về một kiểu dữ liệu cụ thể.
- Mỗi biến có một địa chỉ xác định trong bộ nhớ
  - ✓ Phép toán lấy địa chỉ của một biến là: **&**



# Biến

- Cú pháp khai báo biến: **<Kiểu\_dữ\_liệu>** **<danh sách biến>**

Ví dụ:

<b>int</b>	<b>a, b;</b>	//khai báo hai biến a, b kiểu int
<b>float</b>	<b>m;</b>	// khai báo biến m kiểu float
<b>int</b>	<b>a[7];</b>	//khai báo mảng <b>a</b> có 7 phần tử, kiểu <b>int</b>

- Ngay trên dòng khai báo ta có thể gán cho biến một giá trị.

**int** **d = 5, a = 3;**

- Phân chia biến:

- Theo *kiểu dữ liệu* mà biến lưu trữ: *char, int, float, ...*
- Theo *phạm vi sử dụng* trong chương trình: *cục bộ, toàn cục*
- Theo độ phức tạp dữ liệu mà biến lưu trữ: *sơ cấp, dẫn xuất, cao cấp*



# Hằng

- **Hằng**: là định danh bất biến về giá trị trong chương trình. Các loại hằng được sử dụng trong C tương ứng với các kiểu dữ liệu nhất định.
- Cú pháp khai báo hằng(có hai cách):
  - **const** <kiểu\_dữ\_liệu> <tên\_hằng> = <giá trị>
  - **#define** <tên\_hằng> <giá trị>
- Trong C có ba loại hằng:
  - **Hằng số**: Hằng nguyên, Hằng thực
  - **Hằng ký tự**:
    - Hằng ký tự 'A' thực sự đồng nghĩa với giá trị nguyên 65, là giá trị trong bảng mã ASCII
    - Đối với một vài hằng ký tự đặc biệt, ta cần sử dụng cách viết thêm dấu \, như '\t' tương ứng với phím tab
    - Hằng ký tự có thể tham gia vào phép toán như mọi số nguyên khác
$$'8' - '1' = 56 - 49 = 7$$
  - **Hằng chuỗi**:





## Hằng| Một số hằng ký tự đặc biệt

Cách viết	Ý nghĩa
'\n'	Xuống hàng
'\t'	Tab
'\0'	“nul” tương ứng với giá trị nguyên 0 trong bảng mã ASCII
'\b'	Backspace
'\r'	Về đầu dòng
'\f'	Sang trái
'\\'	\
'\"'	”
'\''	,



## Biểu thức

- ❖ Một **biểu thức** là tập hợp của các toán hạng (biến, hằng) và toán tử (phép toán )
  - Biểu thức toán học
  - Biểu thức logic
  - Biểu thức hỗn hợp



## Ví dụ: Biến, Hằng, Biểu Thức

### ❖ Khai báo biến:

- `int`     `x`;
- `float`   `z = 3.4`;

### ❖ Khai báo hằng:

- `const float` `PI = 3.1415926`;
- `#define`     `PI 3.1415926`

### ❖ Biểu thức:

- `z = a * b + (a/(b - c)) ;`



# Tóm tắt bài học

## ❖ Các khái niệm cơ bản

- Máy tính và hệ đếm nhị phân
- Một số khái niệm về lập trình máy tính; Mã nguồn, mã máy
- Phân loại ngôn ngữ LT; Phương pháp LT
- Thuật toán & Các phương pháp biểu diễn thuật toán

## ❖ Ngôn ngữ lập trình C

- Lịch sử phát triển
- Các thành phần cơ bản của C
  - Bộ chữ viết
  - Định danh (tên gọi)
  - Các kiểu dữ liệu
  - Các phép toán
  - Biến, Hằng, Biểu thức



# Tham khảo

1. <https://icarus.cs.weber.edu/~dab/cs1410/textbook/1.Basics/machine.html>
2. <https://web.stanford.edu/class/cs101/lecture02.html#/26>
3. <https://www.tutorialspoint.com/what-are-the-4-steps-to-convert-c-program-to-machine-code>