

# CHƯƠNG 1. MỘT SỐ KHÁI NIỆM CƠ BẢN VỀ LẬP TRÌNH

## 1. Thuật toán (Algorithm)

### 1.1. Khái niệm

- Thuật toán là khái niệm cơ sở của toán học và tin học.
- Thuật toán là phương pháp thể hiện lời giải của vấn đề – bài toán.
- Thuật toán là dãy các thao tác, các hướng dẫn rõ ràng, được sắp xếp theo một trình tự xác định, sao cho 2 bộ xử lý (người/máy) khác nhau, với cùng điều kiện đầu vào như nhau thì sau một số bước hữu hạn thực hiện, sẽ cho kết quả giống

- Thuật toán là dãy các thao tác, các hướng dẫn rõ ràng, đã sắp xếp theo một trình tự xác định, sao cho 2 bộ xử lý (người/máy) khác nhau, với cùng điều kiện đầu vào như nhau thì sau một số bước hữu hạn thực hiện, sẽ cho kết quả giống nhau mà không cần biết ý nghĩa của các thao tác này. Cần lưu ý là không phải mọi dãy thao tác, chỉ dẫn nào cũng đều tạo ra thuật toán. Phương pháp nấu ăn, cách dùng thuốc,.. .. đều không phải là thuật toán do các thao tác, các chỉ dẫn không xác định, không rõ ràng.

## 1.2. Các đặc trưng của thuật toán

**Tính xác định:** Các thao tác của thuật toán phải xác định, không được nhập nhằng, mơ hồ để có thể dễ dàng cài đặt trên một hệ tự động hóa.

**Tính dừng:** Thuật toán phải dừng sau một số hữu hạn bước thực hiện.

**Tính đúng đắn:** Thuật toán phải cho kết quả đúng theo yêu cầu của bài toán đặt ra.

***Tính phổ dụng:*** Thuật toán có thể được sử dụng lại để giải một lớp bài toán tương tự.

***Tính hiệu quả:*** Thuật toán cần tối ưu về sử dụng bộ nhớ và đáp ứng yêu cầu của bài toán trong thời gian ngắn nhất có thể được. Thực tế rất khó đạt được cả 2 yêu cầu này trong một thuật toán.

### **1.3. Các công cụ biểu diễn thuật toán**

**Ngôn ngữ tự nhiên:** là ngôn ngữ liệt kê các bước, mô tả thuật toán theo ngôn ngữ tự nhiên của con người.

Ví dụ: Thuật toán xác định trị lớn nhất trong 5 số nguyên.






B1. Gọi a, b, c, d, e là 5 biến lưu trữ các trị nguyên cho trước (nhập từ bàn phím).

B2. Gọi max là biến lưu trữ trị lớn nhất trong 5 số nguyên trên, và giả sử a có trị lớn nhất.

B3. Lần lượt so sánh trị của max với các biến b, c, d, e còn lại. Nếu trị của max nhỏ hơn bất kỳ biến nào thì gán trị của biến đó cho max.

B4. Xuất kết quả trị biến max ra màn hình

**Lưu đồ thuật toán hay sơ đồ khối (Flow chart):** là công cụ cho phép biểu diễn thuật toán một cách trực quan. Thường chỉ có thể dùng công cụ lưu đồ đối với các thuật toán tương đối ngắn, có thể được biểu diễn trong một trang giấy. Các hình cơ bản sử dụng trong lưu đồ:

Hình oval mô tả điểm xuất phát / kết thúc.	
Hình chữ nhật mô tả một hay nhiều chỉ thị máy cần thực hiện.	
Hình bình hành mô tả thao tác nhập/xuất dữ liệu.	
Hình thoi mô tả sự rẽ nhánh, lựa chọn, phép kiểm tra điều kiện	
<i>Mũi tên chỉ hướng lưu chuyển của các thao tác.</i>	



**Mã giả (Pseudo code)** gần giống như ngôn ngữ tự nhiên, nhưng có sử dụng các cấu trúc chuẩn hóa (khai báo biến, chú thích, cấu trúc điều khiển, . . .) do người thiết kế quy định.

**Ngôn ngữ lập trình (Programming language)** là hệ thống các ký hiệu cho phép mô tả các quy trình tính toán dưới dạng văn bản.



## 2. Ngôn ngữ lập trình (NNLT)

Các thành phần cơ bản của NNLT bao gồm:

- Bộ kí tự (character set) hay bảng chữ cái dùng để viết chương trình.
- Cú pháp (syntax) là bộ quy tắc để viết chương trình.
- Ngữ nghĩa (semantic) xác định ý nghĩa các thao tác, hành động cần phải thực hiện, ngữ cảnh (context) của các câu lệnh trong chương trình.

Hiện đã có hàng nghìn NNLT được thiết kế, và hàng năm lại có thêm nhiều NNLT mới xuất hiện. Sự phát triển của NNLT gắn liền

---

với sự phát triển của ngành tin học. Mỗi loại NNLT phù hợp với một số lớp bài toán nhất định.

## Phân loại NNLT:

- ***Ngôn ngữ máy*** (machine language) hay còn gọi là NNLT cấp thấp có tập lệnh phụ thuộc vào một hệ máy cụ thể. Chương trình viết bằng ngôn ngữ máy sử dụng bảng chữ cái chỉ gồm 2 kí tự 0, 1. Chương trình ngôn ngữ máy được nạp trực tiếp vào bộ nhớ và thực hiện ngay.
- ***Ngôn ngữ lập trình cấp cao*** nói chung không phụ thuộc vào loại máy tính cụ thể. Chương trình viết bằng NNLT cấp cao sử dụng bộ kí tự phong phú hơn, và phải được chuyển đổi sang dạng mã máy để máy tính có thể hiểu được bằng chương trình dịch. Một số NNLT cấp cao thông dụng hiện nay: Pascal, C, C++, Java, Smalltalk, Basic, Ruby, Fortran, Algol, Lisp, Prolog, Cobol, ...

### 3. Chương trình (máy tính)

Là tập hợp hữu hạn các chỉ thị máy được bố trí, sắp xếp theo một trật tự xác định, nhằm giải quyết yêu cầu của bài toán đặt ra. Chương trình được viết bằng một NNLT cụ thể nào đó.

Các chương trình C/C++ (trong môi trường DOS) được tạo ra bằng 1 trình soạn thảo văn bản (EDITOR) như: SK, NC Editor, VRES  
... Hiện nay, các chương trình dịch đều tích hợp sẵn editor riêng cho phép USER soạn thảo, biên dịch, kiểm lỗi, liên kết và thực hiện chương trình một cách dễ dàng.

Các chương trình này (mã nguồn – source code), thực chất là ở dạng ngôn ngữ tự nhiên, do đó phải được biên dịch lại dưới dạng mã máy (object code) mà máy tính có thể hiểu được. Việc này được thực hiện bởi chương trình dịch.



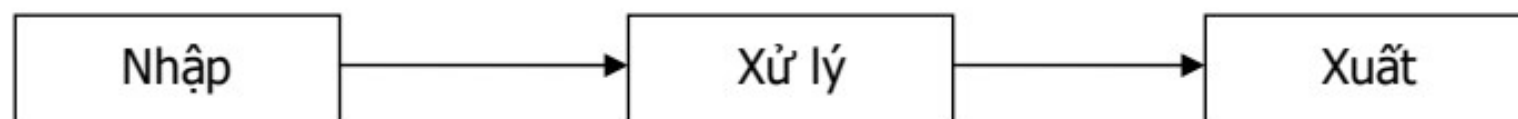
Có 2 loại chương trình dịch:

**Trình thông dịch (interpreter):** mỗi lệnh được dịch sang mã máy và cho thực hiện ngay.

**Trình biên dịch (compiler):** toàn bộ chương trình nguồn được dịch sang mã máy (tập tin.obj), sau đó trình liên kết (linker) sẽ kết nối module chương trình để tạo thành tập tin EXE.



Nói chung, một chương trình máy tính được bố cục thành 3 phần:



#### 4. Các bước xây dựng chương trình

- B1. *Xác định đúng yêu cầu của bài toán*: Cần xác định phạm vi, các giới hạn, ràng buộc, các giả thiết của bài toán. Đặc biệt cần khắc phục sức ì về mặt tâm lý trong quá trình tìm hiểu bài toán.
- B2. *Xây dựng thuật giải*: Cần có kiến thức liên quan đến vấn đề đang giải quyết. Cần xác định rõ thuật toán sẽ tác động trên những đối tượng (thông tin) nào ? Có bao nhiêu đối tượng (biến) cần xử lý? Mỗi biến có thể được lưu trữ dưới dạng nào, kiểu gì ? Giá trị ban đầu của các biến ? Hình dung trước kết xuất DL sau khi xử lý sẽ như thế nào?

B3. Thể hiện thuật giải bằng *lưu đồ thuật toán* (nếu được).

B4. *Cài đặt thuật toán bằng một NNLT cụ thể*: Dùng một trình soạn thảo VB để tạo chương trình nguồn (source code) theo một NNLT nào đó.

B5. *Thử nghiệm thuật toán*, nếu sai quay lại B2. Cần xác định lỗi của thuật toán thuộc loại nào: lỗi về mặt cú pháp (syntax error), lỗi lúc thực hiện chương trình (run-time error), và lỗi logic. Lỗi cú pháp xảy ra lúc biên dịch chương trình, do vi phạm các quy định về mặt cú pháp của NNLT đang sử dụng. Lỗi này tương đối dễ khắc phục. Lỗi run-time error như: divide by zero, stack overflow, không đủ bộ nhớ, ... Lỗi logic (logic error) khó phát hiện hơn nhiều.

B6. Kết thúc.

## 5. Câu hỏi và bài tập

- Thuật toán (Algorithm)
  - Thuật toán là gì?
  - Nêu các đặc trưng cần có của một thuật toán.
  - Các cách biểu diễn một thuật toán?
- Ngôn ngữ lập trình (Programming language) là gì?
- Nêu các bước xây dựng thuật toán.
- Danh hiệu (Identifier)
  - Danh hiệu được dùng để làm gì?
  - Như thế nào là một danh hiệu hợp lệ?
  - Nguyên tắc sử dụng danh hiệu?

- Từ khóa là gì?
  - Đặc điểm của các từ khóa trong NNLT “C/C++”?
- Kiểu dữ liệu (Data type)
  - Trình bày các kiểu dữ liệu đơn giản mà Anh (Chị) đã biết.

- Hằng (Constant)

- Hằng là gì?
- Hằng được sử dụng khi nào?
- Cho biết cách thức khai báo một hằng?
- Cho ví dụ về cách biểu diễn hằng nguyên, thực, ký tự, chuỗi ký tự.



- Biến (Variable)

- Biến là gì?
- Biến được sử dụng để làm gì?
- Cho biết cách thức khai báo một biến?
- Hãy cho biết cách thức làm thay đổi nội dung (giá trị) của một biến?

- Biểu thức (Expression)

- Biểu thức là gì?
- Kiểu của biểu thức do.....quyết định?

- Khi nào xảy ra việc ép kiểu tự động?

Trình bày nguyên tắc NNLT “C/C++” tính trị các biểu thức?

- Toán tử
  - Hãy trình bày các toán tử mà Anh (Chị) biết.
  - Cho ví dụ về toán tử điều kiện (.. .? ..... ).

- Hãy viết biểu thức tương đương không có toán tử ‘!’:

- $!(x \leq 5)$
- $!(x > 5)$
- $!(x > 2 \ \&\& \ y \neq 3)$
- $!(x > 2 \ \parallel \ y == 3)$
- $!(x == 1 \ \&\& \ !(x \neq 3) \parallel x > 10)$
- $!(x > 100 \parallel x < 0 \ \&\& \ !(x == 0))$

- Câu lệnh (Statement, Instruction)
  - Cho biết các loại câu lệnh trong “C/C++”? Cho ví dụ.

## CHƯƠNG 2. CÁC YẾU TỐ CƠ BẢN CỦA NNLT C/C++

### 1. Bộ ký tự (character set)

NNLT C/C++ chỉ chấp nhận các ký tự sau:

- Các ký tự chữ hoa: A, B, C, ..., Z
- Các ký tự chữ thường: a, b, c, ..., z
- Các chữ số: 0, 1, ..., 9
- Các ký tự dấu: , . ! ? : . . .
- Các ký tự trắng: ENTER, BACKSPACE, khoảng trắng.
- Các ký tự đặc biệt khác: + - \* / ^ | # \$ & % ( ) [ ] \_ = ~ ' " . .



## **2. Danh hiệu (identifier)**

Dùng để đặt tên cho các đối tượng như hằng, biến, hàm, ...

Độ dài tối đa của 1 danh hiệu (tùy theo chương trình dịch) thường là 31-32 kí tự.

Danh hiệu hợp lệ được bắt đầu bằng một kí tự chữ cái hoặc dấu gạch nối (underscore), tiếp theo sau là dãy các kí tự chữ hoặc số hoặc dấu gạch nối, và không phép có khoảng trắng ở giữa.

Nên đặt danh hiệu theo các gợi ý sau:

- Đặt các đối tượng một cách gợi nhớ (Mnemonic).
- Tên hãng được đặt bằng chữ hoa.

Tên biến, tên hàm được đặt như sau: từ đầu tiên bằng chữ thường, các từ còn lại bắt đầu bằng chữ hoa.

Tên kiểu dữ liệu do USER định nghĩa được bắt đầu bằng chữ hoa.

Chú ý:

- Mọi danh hiệu phải được khai báo trước khi sử dụng. TRÌNH BIÊN DỊCH sẽ báo lỗi undefined symbol trong quá trình biên dịch chương trình nếu vi phạm nguyên tắc này.
- “C/C++” phân biệt chữ hoa và chữ thường (Case sensitive).
- Trong cùng một phạm vi (scope), không được đặt trùng danh hiệu..

### 3. Từ khóa (keyword)

Là từ dành riêng và có ngữ nghĩa xác định do NNLT quy định. Mọi NNLT đều có một bộ từ khóa riêng. Ngôn ngữ lập trình “C/C++” (version 3.1) thể hiện các từ khóa dưới dạng các kí tự màu trắng, các danh hiệu dưới dạng các kí tự màu vàng, số dưới dạng màu xanh lơ, . .

.

#### **4. Chú thích (comment)**

Được dùng để làm cho chương trình dễ đọc, dễ hiểu, dễ bảo trì hơn.

Chương trình dịch sẽ bỏ qua những nội dung nằm trong phần chú thích.

Có 2 loại chú thích:

- **Chú thích trên một dòng:**

---

`// . . . phần chú thích (cho đến cuối dòng)`

- **Chú thích trên nhiều dòng:**

`/*... phần chú thích (có thể trải dài trên nhiều dòng)... */`



## 5. Các kiểu dữ liệu cơ bản (base type)

### 5.1. Số nguyên

Tên kiểu: **int**

Kích thước: 2 bytes và có phạm vi biểu diễn giá trị  $-32768 \dots 32767$

Các phép toán áp dụng được trên kiểu **int**:

- Các phép toán số học:  $+$ ,  $-$ ,  $*$ ,  $/$ ,  $\%$
- Các phép toán so sánh:  $<$ ,  $<=$ ,  $>$ ,  $>=$ ,  $==$ ,  $!=$
- Các phép toán dịch chuyển số học:  $>>$ , và  $<<$
- Các phép toán trên bit:  $\sim$  (not bit),  $\&$  (and bit),  $|$  (or bit),  $\wedge$  (xor bit)

## 5.2. Số thực (độ chính xác đơn – 6 chữ số lẻ)

Tên kiểu: **float**

Kích thước: 4 bytes và có phạm vi biểu diễn giá trị  $-3.4E-38 \dots 3.4E+38$

Các phép toán áp dụng được trên kiểu float:

- Các phép toán số học: + , - , \* , / (không có phép toán %)
- Các phép toán so sánh: < , <= , > , >= , == , !=

### 5.3. Số thực (độ chính xác kép – 15 chữ số lẻ):

Tên kiểu: **double** (Mặc định trong Visual C++)

Kích thước: 8 bytes và có phạm vi biểu diễn giá trị  $-1.7E-308 \dots 1.7E+308$

Các phép toán áp dụng được trên kiểu double: như kiểu float

## 5.4. K í t ự

Tên kiểu: **char**

Kích thước: 1 byte và có phạm vi biểu diễn giá trị  $-128 \dots 127$

Các phép toán áp dụng được trên kiểu char:

- Các phép toán số học:  $+$ ,  $-$ ,  $*$ ,  $/$ ,  $\%$
- Các phép toán so sánh:  $<$ ,  $<=$ ,  $>$ ,  $>=$ ,  $==$ ,  $!=$

Để mở rộng các kiểu dữ liệu cơ sở, “C/C++” đưa thêm các tiền tố (prefix): *short*, *long*, *unsigned*, *signed* vào trước tên các kiểu cơ sở như sau:

Tên kiểu	Kích thước
unsigned char	1 byte (0 .. 256)
Char	1 byte (−128 .. 127)
unsigned int	2 bytes (0 .. 65535)
(unsigned)	
short int	2 bytes (−32768 .. 32767)
Int	2 bytes (−32768 .. 32767)

unsigned long int (unsigned long)	4 bytes (0 .. 4294967295)
long int (long)	4 bytes (−2147483648 .. 2147483647)
float	4 bytes (−3.4E −38 .. 3.4E +38)
double	8 bytes (−1.7E −308 .. 1.7E +308)
long double	10 bytes (−3.4 E −4932 .. 1.1E+4932)

## 6. Hằng (constant)

Là đại lượng có giá trị không thay đổi trong suốt thời gian tồn tại của nó.

Tên hằng phải là một danh hiệu hợp lệ và phải “được khai báo trước khi sử dụng”.

Hằng được dùng để thay thế cho các con số tối nghĩa trong chương trình. Việc sử dụng hằng những lúc cần thiết là một phong cách lập trình tốt và cần được khuyến khích.



**Hàng số nguyên:** có thể được biểu diễn dưới dạng thập phân, nhị phân, bát phân, và thập lục phân.

**Hàng số thực:** có thể được biểu diễn dưới dạng kí pháp thông thường hoặc dạng kí pháp khoa học.

- Kí pháp thông thường (còn gọi là số thực dấu phẩy tĩnh) gồm 2 phần, được phân cách bởi dấu chấm thập phân. Ví dụ: 1234.5
- Kí pháp khoa học (còn gọi là số thực dấu phẩy động) gồm phần định trị (là một số thực) và phần mũ (là một số

nguyên). Hai phần này được phân cách bởi chữ e hoặc E. Ví dụ: 1.2345 E+03.

**Hằng kí tự:** được đặt trong cặp nháy đơn và có thể được biểu diễn bằng:

- Kí hiệu trong bảng mã ASSCI. Ví dụ: 'A'.
- Escape character (kí tự thoát) bao gồm cặp kí tự \n, với n là số thứ tự của kí tự trong bảng mã ASSCII. Ví dụ: '\65', '\7', '\n'

## Một số kí tự đặc biệt:

Kí hiệu	Ý nghĩa
\n	Kí tự xuống dòng
\t	K í tự TAB
\0	Kí tự NULL
\'	Dấu nháy đơn
\"	Dấu nháy kép
\\	Dấu sổ chéo ngược (Backslash)
....	

**Hàng chuỗi kí tự:** được đặt trong cặp nháy đôi. Thực chất đó là mảng các kí tự có kí tự kết thúc chuỗi là kí tự NULL, kí hiệu ‘\0’

Khai báo hằng: thường được đặt trong phần khai báo toàn cục ở đầu chương trình, ngay sau các khai báo chỉ thị tiền xử lý. Có 2 cách khai báo hằng:

- Dùng chỉ thị tiền xử lý: `#define <tên hằng> <chuỗi thay thế>`
- Dùng từ khóa `const`: `const <tên kiểu> <tên hằng>=<giá trị>;`

Ví dụ:

```
const int MAX = 10;
```

hay

```
#define MAX 10
```

## 7. Biến (variable)

Là đại lượng có giá trị có thể thay đổi bằng toán tử gán (kí hiệu =). Biến được dùng để lưu trữ thông tin về các đối tượng và có giá trị cụ thể, xác định tại mỗi thời điểm trong chương trình.

Tên biến phải là một danh hiệu hợp lệ và không được đặt trùng với từ khóa. Nên đặt tên biến sao cho có tính gợi nhớ, không dài quá và cũng không nên quá ngắn. Nên tránh đặt tên biến trùng với tên các hàm thư viện (sin, cos, ...).

“C/C++” cho phép khai báo biến ở khắp mọi nơi trong chương trình, miễn sao đảm bảo nguyên tắc “Mọi danh hiệu trước khi sử dụng phải được khai trước”.



Cách khai báo biến:

```
<type> <variables list>;
```

Trong đó:

---

<type> là một kiểu dữ liệu hợp lệ bất kỳ có trong “C/C++”.

<variables list> là một hay nhiều biến, được phân cách bằng dấu ‘,’.



Chú ý: lệnh khai báo biến luôn được kết thúc bằng dấu ‘;’

Ví dụ:

Để khai báo biến kí tự

```
char ch; // khai báo biến kí tự
```

Để khai báo 2 biến nguyên:

```
int x, y; // khai báo 2 biến kiểu nguyên
```

hoặc

```
int x;
```

```
int y;
```

Có thể khai báo và đồng thời khởi tạo giá trị cho biến như sau:

```
<tên kiểu> <tên biến> = <giá trị>;
```

Ví dụ:

```
int tong = 0, x = 5, y = 7;
```

## 8. Biểu thức (expression)

Là công thức tính toán bao gồm các toán hạng và các toán tử tương ứng. Các toán hạng có thể là một biến, hằng, lời gọi hàm. Bản thân các toán hạng cũng có thể là một biểu thức con khác đặt trong cặp ngoặc đơn, hình thành nên một biểu thức phức hợp.

Có các loại biểu thức thông dụng sau: biểu thức gán, biểu thức số học, biểu thức logic. Đặc biệt, biểu thức logic trong “C/C++” được xem là có giá trị nguyên. Biểu thức có trị khác 0 (kể cả số âm) tương ứng với mệnh đề logic TRUE, và biểu thức có trị = 0 tương ứng với mệnh đề logic FALSE.

~

Trong “C/C++”, biểu thức luôn trả về một giá trị.

Kiểu của biểu thức phụ thuộc vào kiểu của giá trị trả về.

## 9. Chuyển đổi kiểu (type conversion)

**Chuyển đổi kiểu ngầm định:** Trong cùng 1 biểu thức, nếu các toán hạng không cùng kiểu với nhau thì trước khi tính toán giá trị của biểu thức, chương trình dịch sẽ thực hiện việc chuyển đổi kiểu ngầm định (nếu được) theo nguyên tắc “Kiểu có phạm vi giá trị biểu diễn nhỏ hơn sẽ được chuyển sang kiểu có phạm vi giá trị biểu diễn lớn hơn”. Sơ đồ chuyển đổi kiểu ngầm định:

*char → int → long → float → double → long double*

**Ép kiểu (type casting):** Trong một số trường hợp, ta bắt buộc phải sử dụng đến toán tử ép kiểu để tạo ra một biểu thức hợp lệ như sau:  
<tên kiểu> (<biểu thức>) hoặc (<tên kiểu>) <biểu thức>

Ví dụ, để tạo biểu thức số học hợp lệ sau  $8.0 \% 3$ , ta cần thực hiện ép kiểu như sau:

$(\text{int})\ 8.0 \% 3$

Hay

$\text{int}\ (8.0 \% 3)$



## **10. Các toán tử (operator)**

“C/C++” rất giàu về toán tử. Sau đây là một số toán tử thông dụng:

### 10.1. Toán tử số học

K í hiệu	Ý nghĩa	Số ngôi	Toán hạng
+	Cộng	2	int, float, double, char
-	Trừ	2	- nt -
*	Nhân	2	- nt -
/	Chia	2	- nt -
%	Modulo	2	int, char (không có kiểu số thực)

## 10.2. Toán tử quan hệ (so sánh)

K í hiệu	Ý nghĩa	Số ng ôi	Toán hạng
<	Nhỏ hơn	2	int, float, double, char
<=	Nhỏ hơn hoặc bằng	2	- nt -

>	Lớn hơn	2	- nt -
>=	Lớn hơn hoặc bằng	2	- nt -
==	So sánh bằng	2	- nt -
!=	So sánh khác nhau	2	- nt -

### 10.3. Toán tử logic

K í hiệu	Ý nghĩa	Số ng ôi	Toán hạng
!	NOT logic	1	int, float, double, char
&&	AND logic	2	- nt -
	OR logic	2	- nt -

## 10.4. Toán tử gán

Toán tử gán dùng để thay đổi trị của một biến bằng trị của một biểu thức.

**Kí hiệu:**  $=$

**Biểu thức gán có dạng:**  $\langle \text{biến} \rangle = \langle \text{biểu thức} \rangle$ , trong đó  $\langle \text{biến} \rangle$  là một danh hiệu hợp lệ (nằm ở vế trái), và  $\langle \text{biểu thức} \rangle$  (nằm ở vế phải) là biểu thức có cùng kiểu với kiểu của  $\langle \text{biến} \rangle$ .

**Cách tính trị của biểu thức gán:** TRÌNH BIÊN DỊCH tính trị của <biểu thức>, sau đó gán trị này cho <biến>. Toàn bộ biểu thức gán này cũng trả về một giá trị là trị của <biểu thức> vừa tính được.

**Ví dụ:**

`x = 1; // gán trị 1 cho biến nguyên x`

`x = 2*y; // tính trị 2*y, sau đó gán kết quả tính được cho biến x`



$z = x + 2*y;$  // tính trị  $x+2*y$ , sau đó gán kết quả  
tính được cho biến  $z$

$a = \sin(2*b);$  // gọi hàm thư viện tính  $\sin(2*b)$ ,  
sau đó gán kết quả cho  $a$

C/C++ cho phép viết gọn các biểu thức gán  
bằng các toán tử gán sau:

Dạng viết thông thường	Dạng viết thu gọn	Ý nghĩa
$i = i + \langle bt \rangle$	$i += \langle bt \rangle$	Tự cộng
$i = i - \langle bt \rangle$	$i -= \langle bt \rangle$	Tự trừ
$i = i * \langle bt \rangle$	$i *= \langle bt \rangle$	Tự nhân
$i = i / \langle bt \rangle$	$i /= \langle bt \rangle$	Tự chia
$i = i \% \langle bt \rangle$	$i \% = \langle bt \rangle$	Tự modulo
...		