CÁC TOÁN TỬ CƠ BẢN
TRONG NGÔN NGỮ LẬP TRÌNH C/C++

Một số lưu ý:
// <=> comment ghi chú, giải thích
; <=> kết thúc câu lệnh, luôn luôn có nếu không sẽ bị lỗi.

1. Toán tử gán (=).

- Toán tử gán - dùng để gán một ***giá trị*** bất kì cho một ***biến.***

VD_1:

```
int a  =  5;  // khởi tạo biến a có kiểu dữ liệu là số nguyên, sau đó gán giá trị 5 cho biến a.
```

- ***Vế trái*** bắt buộc phải là một ***biến*** còn ***vế phải*** có thể là bất kì ***hằng, biến*** hay kết quả của ***một biểu thức***.
- Cần phải nhấn mạnh rằng toán tử gán ***luôn luôn*** được thực hiện ***từ phải sang trái*** và không bao giờ đảo ngược.

Lưu ý:

- Kiểu dữ liệu của 2 vế trái và phải là đồng nhất với nhau(có thể 1 số trường hợp ngoại lệ).

Trường hợp ngoại lệ:

VD_2:

int a = ‘c’; // hợp lệ
==> Bởi vì vế trái là 1 biến thuộc kiểu dữ liệu là số nguyên – biến a, còn vế phải là 1 kí tự - mà bản chất của kí tự là được lưu trữ dưới dạng mã **ASCII – mã số nguyên**(mỗi kí tự hay kí hiệu có một mã số từ 0 cho đến 255 – kí tự trên bàn phím), vì vậy 2 vế được xem là đồng nhất kiểu dữ liệu với nhau.

VD_3:

```
int a;
int b = 6;
a = b; // hợp lệ, vì đồng nhất kiểu dữ liệu số nguyên - int
```

VD_4:

```
bool  KT = true;
int x = KT; // hợp lệ
```

==> Bởi vì giá trị *true* và *false* thuộc kiểu dữ liệu **bool**, thì thực chất *true* tương đương với 1 giá trị khác 0(thường là 1), còn *false* tương đương với giá trị 0. Vì vậy 2 vế trái và phải đồng nhất kiểu dữ liệu với nhau.

VD_5:

```
int a, b = 5;
a  =  b;  // giá trị của biến a được gán bằng giá trị đang chứa trong biến b.
```

*Chú ý rằng chúng ta chỉ gán **giá trị** của **b** cho **a** và sự thay đổi của **b** sau đó sẽ không ảnh hưởng đến giá trị của **a**.*

VD_6:
- Một thuộc tính của toán tử gán trong C/C++ góp phần giúp nó vượt lên các ngôn ngữ lập trình khác là việc cho phép *vế phải* có thể chứa các phép gán khác.

```
a = 2 + (b = 5);
tương đương với:
b = 5;
a = 2 + b;
```

- Vì vậy biểu thức sau cũng hợp lệ trong C/C++

```
a = b = c = 5;  // gán giá trị 5 cho cả ba biến a, b và c.
```

2. Các toán tử số học (+, -, *, /, %)

- Năm toán tử số học được hỗ trợ bởi ngôn ngữ là:

+ *cộng*
- *trừ*
* *nhân*
/ *chia lấy phần nguyên*
% *chia lấy phần dư (trong phép chia)*

- Thứ tự thực hiện các toán tử này cũng giống như chúng được thực hiện trong toán học.
- Điều duy nhất có vẻ hơi lạ đối với bạn là phép chia lấy phần dư, ký hiệu bằng dấu phần trăm (%).

Chú ý:

- Đây chính là phép toán chia lấy phần dư trong phép chia hai *số nguyên* với nhau.
  + Ví dụ, nếu **a = 11 % 3;**

biến **a** sẽ mang giá trị 2 vì 11 = 3*3 + 2.

Lưu ý

- / : phép chia lấy phần nguyên
  + Số thực chia số nguyên , kết quả là 1 số thực.
  + Số thực chia số thực, kết quả là 1 số thực.
  + Số nguyên chia số nguyên, kết quả là 1 số nguyên.
  + Số nguyên chia số thực, kết quả là 1 số thực

VD:

7 / 3 = 2

3 / 4 = 0

3. Các toán tử gán phức hợp (+=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=)

- Một đặc tính của ngôn ngữ C++ làm cho nó nổi tiếng là một ngôn ngữ súc tích chính là các toán tử gán phức hợp - cho phép chỉnh sửa giá trị của một biến với một trong những toán tử cơ bản sau:

value += increase; tương đương với **value = value + increase;**

VD_7:

a -= 5**;** tương đương với a = a - 5;

a /= b; tương đương với a = a / b;

price *= units + 1;

tương đương với **price = price * (units + 1);**

và tương tự cho tất cả các toán tử khác.

4. Tăng và giảm.

- Một ví dụ khác của việc tiết kiệm khi viết mã lệnh là toán tử *tăng* (++) và *giảm* (--). Chúng tăng hoặc giảm giá trị chứa trong một biến đi 1 đơn vị.
- Chúng tương đương với ++ <=> **+=1** hoặc -- <=> **-=1**. Vì vậy, các dòng sau là tương đương:

a++;

a += 1;

a = a+1;

- Một tính chất của toán tử này là nó có thể là *tiền tố* hoặc *hậu tố*, có nghĩa là có thể viết trước tên biến (**++a**) hoặc sau (**a++**), và mặc dù trong hai biểu thức rất đơn giản đó nó có cùng ý nghĩa là tăng lên 1 đơn vị, nhưng trong các thao tác khác khi mà kết quả của việc *tăng* hay *giảm* được sử dụng trong một biểu thức hoặc là câu lệnh thì chúng có thể có một khác biệt quan trọng về ý nghĩa:

++a – tức là tăng giá trị của biến a lên 1 đơn vị trước, sau đó mới lấy giá trị(giá trị sau khi tăng) của biến a đem đi thao tác trong câu lệnh hoặc là biểu thức.

VD_8:

int a = 5;

int b;

b = ++a;

==> a = 6, b = 6

a++ - tức là đem biến a đi thực hiện câu lệnh hoặc biểu thức trước, sau đó biến a mới tăng lên 1 đơn vị.

VD_9:
int a = 8;
int b;
b = a++;
==> a = 9, b = 8

Chú ý:
- Có thể kết hợp nhiều lần ++ hay -- với 1 biến. Vì ++ và -- có cùng độ ưu tiên nên chúng sẽ được kết hợp từ *phải sang trái.*
- Cho nên ++++a là hợp lệ. Còn a++++ là không hợp lệ.

Nhận xét:
- *Việc kết hợp linh hoạt và tài tình các toán tử tiền tố(++a), hậu tố(a++), tăng(++), giảm(--) sẽ giúp cho chương trình của chúng ta trở nên gọn và chuyên nghiệp hơn, nhưng làm cho người chưa biết về nó có thể gặp chút bối rối và thao thức.*

5. Các toán tử quan hệ (==, !=, >, <, >=, <=)

- Để có thể so sánh hai biểu thức với nhau chúng ta có thể sử dụng các toán tử quan hệ. Theo chuẩn ANSI-C++ thì giá trị của thao tác quan hệ chỉ có thể là giá trị logic - chúng chỉ có thể có giá trị **true** <=> **1** hoặc **false** <=> **0**, tuỳ theo biểu thức kết quả là đúng hay sai.
- Sau đây là các toán tử quan hệ bạn có thể sử dụng trong C/C++:

== **Bằng**
!= **Khác**
> **Lớn hơn**
< **Nhỏ hơn**
> = **Lớn hơn hoặc bằng**
< = **Nhỏ hơn hoặc bằng**

VD_10:

(7 == 5) sẽ trả giá trị **false** <=> **0**
(6 >= 6) sẽ trả giá trị **true** <=> **1**

tất nhiên thay vì sử dụng các số, chúng ta có thể sử dụng bất cứ biểu thức nào.

Cho **a=2**, **b=3** và **c=6**
(a*b >= c) sẽ trả giá trị **true** <=> **1**
(b+4 < a*c) sẽ trả giá trị **false** <=> **0**

- Cần chú ý rằng = (một dấu bằng) hoàn toàn khác với == (hai dấu bằng). Dấu = là một toán tử gán (gán giá trị của biểu thức bên phải cho biến ở bên trái) và dấu còn lại == là một toán tử quan hệ nhằm so sánh xem hai biểu thức có bằng nhau hay không.

Chú ý:
- Trong C/C++ thì phép toán quan hệ đôi khi không trả về *true* hay *false* mà trả về giá trị nguyên. Giá trị nguyên mà là 0 thì hiểu là *false*, còn tất cả các giá trị nguyên khác 0 thì hiểu là *true* (69), thường là 1.

6. Các toán tử logic (!, &&, ||).

- Toán tử **!** tương đương với toán tử logic NOT, nó chỉ có một đối số ở phía bên phải và việc duy nhất mà nó làm là đổi ngược giá trị của đối số từ **true** sang **false** hoặc ngược lại.
VD_11:

!(5 == 5) trả về **false** vì biểu thức bên phải (5 == 5) có giá trị **true**.
!(6 <= 4) trả về **true** vì (6 <= 4) có giá trị **false**.
!true trả về **false**.
!false trả về **true**.

- Toán tử logic **&&** và **||** được sử dụng khi tính toán hai biểu thức để lấy ra một kết quả duy nhất. Chúng tương ứng với các toán tử logic *AND* và *OR*. Kết quả của chúng phụ thuộc vào mối quan hệ của hai đối số:

Đối số thứ nhất A	**Đối số thứ hai** **b**	**Kết quả** **a && b**	**Kết quả** **a || b**
true	true	**true**	**true**
true	false	**false**	**true**
false	true	**false**	**true**
false	false	**false**	**false**

VD_12:
((5 == 5) && (3 > 6)) trả về **false** (*true* && *false*) .
((5 == 5) || (3 > 6)) trả về **true** (*true* || *false*) .

7. Toán tử điều kiện (?).

- Toán tử điều kiện tính toán một biểu thức và trả về một giá trị khác tuỳ thuộc vào biểu thức đó là đúng hay sai.
Cấu trúc của nó như sau:

condition **?** *result1* **:** *result2*

- Nếu *condition* là **true** thì giá trị trả về sẽ là *result1*, nếu không giá trị trả về là *result2*.

VD_13:
7==5 ? 4 : 3 trả về **3** vì **7** không bằng **5**.
7==5+2 ? 4 : 3 trả về **4** vì **7** bằng **5+2**.
5>3 ? a : b trả về **a**, vì **5** lớn hơn **3**.
a>b ? a : b trả về giá trị lớn hơn, **a** hoặc **b**.

---------- ***END*** ----------

- Đây là tài liệu dùng để học và tham khảo.

- Trong quá trình biên soạn bộ tài liệu có:

+ Tham khảo 1 số nguồn tài liệu(google search, ebook, sách chuyên ngành…).

+ Kiến thức – kinh nghiệm cá nhân của tôi.

- Có thể còn 1 số lỗi sai sót – sẽ luôn cập nhật phiên bản mới.

- Mong nhận được sự đóng góp và phản hồi tích cực từ các bạn đọc. Chân thành cảm ơn.