Toán tử trong C++

Trong C++, để thao tác với chúng ta sử dụng các toán tử, đó là các từ khoá và các dấu không có trong bảng chữ cái nhưng lại có trên hầu hết các bàn phím trên thế giới. Hiểu biết về chúng là rất quan trọng vì đây là một trong những thành phần cơ bản của ngôn ngữ C++.

1. Toán tử gán (=)

Toán tử gán dùng để gán một giá trị nào đó cho một biến

```
a = 5;
```

gán giá trị nguyên 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 được thực hiện từ trái sang phải và không bao giờ đảo ngược

```
a = b;
```

gán giá trị của biến **a** 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**.

Một thuộc tính của toán tử gán trong 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. Ví dụ:

```
a = 2 + (b = 5);
```

tương đương với:

```
b = 5;
a = 2 + b;
```

Vì vậy biểu thức sau cũng hợp lệ trong 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 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 lấy phần dư, ký hiệu bằng dấu phần trăm (%). Đây chính là phép toán 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.

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;**
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. 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 nhưng trong các thao tác khác khi mà kết quả của việc tăng hay giảm được sử dụng trong một biểu thức thì chúng có thể có một khác biệt quan trọng về ý nghĩa: Trong trường hợp toán tử được sử dụng như là một tiền tố (**++a**) giá trị được tăng trước khi biểu thức được tính và giá trị đã tăng được sử dụng trong biểu thức; trong trường hợp ngược lại (**a++**) giá trị trong biến a được tăng sau khi đã tính toán. Hãy chú ý sự khác biệt:

Ví dụ 1	**Ví dụ 2**
B=3; **A=++B;** *// A is 4, B is 4*	**B=3;** **A=B++;** *// A is 3, B is 4*

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** hoặc **false**, 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++

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

Ví dụ:

(7 == 5)	sẽ trả giá trị **false**

(6 >= 6)	sẽ trả giá trị **true**

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**.
(b+4 < a*c)	sẽ trả giá trị **false**

Cần chú ý rằng = (một dấu bằng) lf hoàn toàn khác với == (hai dấu bằng). Dấu đầu tiên 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.

Trong nhiều trình dịch có trước chuẩn ANSI-C++ cũng như trong ngôn ngữ C, các toán tử quan hệ không trả về giá trị logic **true** hoặc **false** mà trả về giá trị **int** với **0** tương ứng với **false** còn giá trị khác 0 (thường là 1) thì tương ứng với **true**.

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. Ví dụ:

!(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**

Ví dụ:

((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*.

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**.

8. Các toán tử thao tác bit (&, |, ^, ~, <<, >>)

Các toán tử thao tác bit thay đổi các bit biểu diễn một biến, có nghĩa là thay đổi biểu diễn nhị phân của chúng

Toán tử	asm	Mô tả
&	**AND**	Logical AND
|	**OR**	Logical OR
^	**XOR**	Logical exclusive OR
~	**NOT**	Đảo ngược bit
<<	**SHL**	Dịch bit sang trái
>>	**SHR**	Dịch bit sang phải

9. Các toán tử chuyển đổi kiểu

Các toán tử chuyển đổi kiểu cho phép bạn chuyển đổi dữ liệu từ kiểu này sang kiểu khác. Có vài cách để làm việc này trong C++, cách cơ bản nhất được thừa kế từ ngôn ngữ C là đặt trước biểu thức cần chuyển đổi tên kiểu dữ liệu được bọc trong cặp ngoặc đơn **()**, ví dụ:

```
int i;
float f = 3.14;
i = (int) f;
```

Đoạn mã trên chuyển số thập phân 3.14 sang một số nguyên (3). Ở đây, toán tử chuyển đổi kiểu là (int). Một cách khác để làm điều này trong C++ là sử dụng các constructors (ở một số sách thuật ngữ này được dịch là **cấu tử** nhưng tôi thấy nó có vẻ không xuôi tai lắm) thay vì dùng các toán tử : đặt trước biểu thức cần chuyển đổi kiểu tên kiểu mới và bao bọc **biểu thức** giữa một cặp ngoặc đơn.

```
i = int ( f );
```

Cả hai cách chuyển đổi kiểu đều hợp lệ trong C++. Thêm vào đó ANSI-C++ còn có những toán tử chuyển đổi kiểu mới đặc trưng cho lập trình hướng đối tượng.

```
sizeof();
```

Toán tử này có một tham số, đó có thể là một kiểu dữ liệu hay là một biến và trả về kích cỡ bằng byte của kiểu hay đối tượng đó.

```
a = sizeof (char);
```

a sẽ mang giá trị 1 vì kiểu **char** luôn có kích cỡ 1 byte trên mọi hệ thống. Giá trị trả về của **sizeof** là một hằng số vì vậy nó luôn luôn được tính trước khi chương trình thực hiện.

10. Các toán tử khác

Trong C++ còn có một số các toán tử khác, như các toán tử liên quan đến con trỏ hay lập trình hướng đối tượng. Chúng sẽ được nói đến cụ thể trong các phần tương ứng.

11. Thứ tự ưu tiên của các toán tử

Khi viết các biểu thức phức tạp với nhiều toán hạng các bạn có thể tự hỏi toán hạng nào được tính trước, toán hạng nào được tính sau. Ví dụ như trong biểu thức sau:

```
a = 5 + 7 % 2
```

có thể có hai cách hiểu sau:

a = 5 + (7 % 2) với kết quả là **6**, hoặc
a = (5 + 7) % 2 với kết quả là **0**

Câu trả lời đúng là biểu thức đầu tiên. Vì nguyên nhân nói trên, ngôn ngữ C++ đã thiết lập một thứ tự ưu tiên giữa các toán tử, không chỉ riêng các toán tử số học mà tất cả các toán tử có thể xuất hiện trong C++. Thứ tự ưu tiên của chúng được liệt kê trong bảng sau theo thứ tự từ cao xuống thấp.

Thứ tự	**Toán tử**	**Mô tả**	**Associativity**
1	::	scope	Trái
2	() [ ] -> . sizeof		Trái
3	++ --	tăng/giảm	Phải
	~	Đảo ngược bit	
	!	NOT	
	& *	Toán tử con trỏ	
	(*type*)	Chuyển đổi kiểu	
	+ -	Dương hoặc âm	
4	* / %	Toán tử số học	Trái
5	+ -	Toán tử số học	Trái
6	<< >>	Dịch bit	Trái
7	< <= > >=	Toán tử quan hệ	Trái
8	== !=	Toán tử quan hệ	Trái
9	& ^ |	Toán tử thao tác bit	Trái
10	&& ||	Toán tử logic	Trái
11	?:	Toán tử điều kiện	Phải
12	= += -= *= /= %= >>= <<= &= ^= |=	Toán tử gán	Phải

13	,	Dấu phẩy	Trái

Associativity định nghĩa trong trường hợp có một vài toán tử có cùng thứ tự ưu tiên thì cái nào sẽ được tính trước, toán tử ở phía xa nhất bên phải hay là xa nhất bên trái.

Nếu bạn muốn viết một biểu thức phức tạp mà lại không chắc lắm về thứ tự ưu tiên của các toán tử thì nên sử dụng các ngoặc đơn. Các bạn nên thực hiện điều này vì nó sẽ giúp chương trình dễ đọc hơn.

Revision #3
Created 4 October 2019 15:26:52 by Laptrinh.vn
Updated 5 October 2019 03:57:51 by Laptrinh.vn