

# Bài 12: TOÁN TỬ SỐ HỌC, TOÁN TỬ TĂNG GIẢM, TOÁN TỬ GÁN SỐ HỌC TRONG C++ (OPERATORS)

Xem bài học trên website để ủng hộ Kteam: [Toán tử số học, toán tử tăng giảm, toán tử gán số học trong C++ \(Operators\)](#)

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage [How Kteam](#) nhé!

## Dẫn nhập

Ở bài học trước, bạn đã nắm được các loại [HẰNG SỐ TRONG C++ \(Constants\)](#), và đã biết được những kinh nghiệm cũng như thắc mắc liên quan đến việc khởi tạo và sử dụng hằng số một cách hiệu quả.

Hôm nay, mình sẽ hướng dẫn chi tiết về **Toán tử số học, toán tử tăng giảm và toán tử gán trong C++ (Arithmetic operators)**, là tiền đề để bạn có thể giải được các bài toán trong lập trình.

---

## Nội dung

Để đọc hiểu bài này tốt nhất các bạn nên có kiến thức cơ bản về các phần:

- [BIẾN TRONG C++ \(Variables\)](#)
- [CÁC KIỂU DỮ LIỆU CƠ BẢN TRONG C++](#) (Integer, Floating point, Character, Boolean)
- [NHẬP XUẤT DỮ LIỆU TRONG C++](#) (Input and Output)

Trong bài ta sẽ cùng tìm hiểu các vấn đề:

- Tổng quan về toán tử
- Toán tử số học trong C++
- Toán tử tăng giảm trong C++
- Toán tử gán số học trong C++

## Tổng quan về toán tử

Trong bài [CẤU TRÚC MỘT CHƯƠNG TRÌNH C++ \(Structure of a program\)](#), bạn đã biết được một thành phần không thể thiếu trong một chương trình máy tính là **Các câu lệnh và biểu thức (Statements and expressions)**.

Biểu thức ở đây chính là một **thực thể toán học**. Nói cách khác, nó là một sự kết hợp giữa 2 thành phần:

- **Toán hạng:** có thể là một hằng số, biến số hoặc một lời gọi hàm.
- **Toán tử:** xác định cách thức làm việc giữa các toán hạng.

### Ví dụ:

```
// Một số toán tử trong biểu thức bên dưới = ( + ) *  
int nSoNguyen = (6 + 9) * (6 - 9);
```

Có 3 loại toán tử trong C++:

- **Toán tử 1 ngôi (Unary):** chỉ có 1 toán hạng trong biểu thức. **Ví dụ:** +, -, ++, --.
- **Toán tử 2 ngôi (Binary):** có 2 toán hạng trong biểu thức. **Ví dụ:** +, -, \*, /, %.
- **Toán tử 3 ngôi (Ternary):** có 3 toán hạng trong biểu thức, **Ví dụ:** Conditional operator ?:

**Chú ý:** Một số toán tử có thể mang nhiều ý nghĩa khác nhau. **Ví dụ:** Unary plus (+) và Binary plus (+).

## Toán tử số học trong C++ (Arithmetic operators)

### Toán tử 1 ngôi số đối (Unary)

**Toán tử 1 ngôi (Unary)** là toán tử chỉ có 1 toán hạng trong biểu thức.

Bảng bên dưới mô tả 2 toán tử 1 ngôi số đối trong C++, giả sử **x = 69**:

Operator	Symbol	Form	Operation	Example
Unary plus	+	+x	Value of x	+a will give 69
Unary minus	-	-x	Negation of x	-a will give -69

#### Chú ý 1:

- **Unary plus (+)** trả về giá trị của chính toán hạng đó, bạn sẽ không cần phải sử dụng toán tử này vì nó **không cần thiết**.
- **Unary minus (-)** trả về số đối của toán hạng đó.

#### Chú ý 2:

- **Nên đặt 2 toán tử một ngôi số đối ngay trước toán hạng** để dễ dàng phân biệt với các toán tử số học khác. **Ví dụ:** -a, không nên - a.

- **Tránh nhầm lẫn** giữa toán tử 1 ngôi (+, -) và toán tử 2 ngôi (+, -). **Ví dụ:**  $x = 6 - -9$ .

## Toán tử số học 2 ngôi (Binary)

**Toán tử 2 ngôi (Binary)** là toán tử có 2 toán hạng trong biểu thức. Có 5 toán tử số học 2 ngôi trong C++.

Bảng bên dưới mô tả các toán tử số học trong C++, giả sử  $x = 6$ ,  $y = 9$ :

Operator	Symbol	Form	Operation	Example
<b>Addition</b>	+	$x + y$	x plus y	a + b will give 15
<b>Subtraction</b>	-	$x - y$	x minus y	a - b will give -3
<b>Multiplication</b>	*	$x * y$	x multiplied by y	a * b will give 54
<b>Division</b>	/	$x / y$	x divided by y	b / a will give 1
<b>Modulus (Remainder)</b>	%	$x \% y$	The remainder of x divided by y	b % a will give 3

Toán tử cộng (+), trừ (-), nhân (\*) hoạt động một cách bình thường như trong toán học.

Về toán tử chia sẽ được phân thành 2 dạng:

- **Chia lấy phần nguyên (/):** trả về phần nguyên của phép chia. **Ví dụ:**  $9 / 6 = 1$  dư 3
- **Chia lấy phần dư (%):** trả về phần dư của phép chia. **Ví dụ:**  $9 \% 6 = 3$

**Ví dụ:**

```
#include <iostream>
using namespace std;

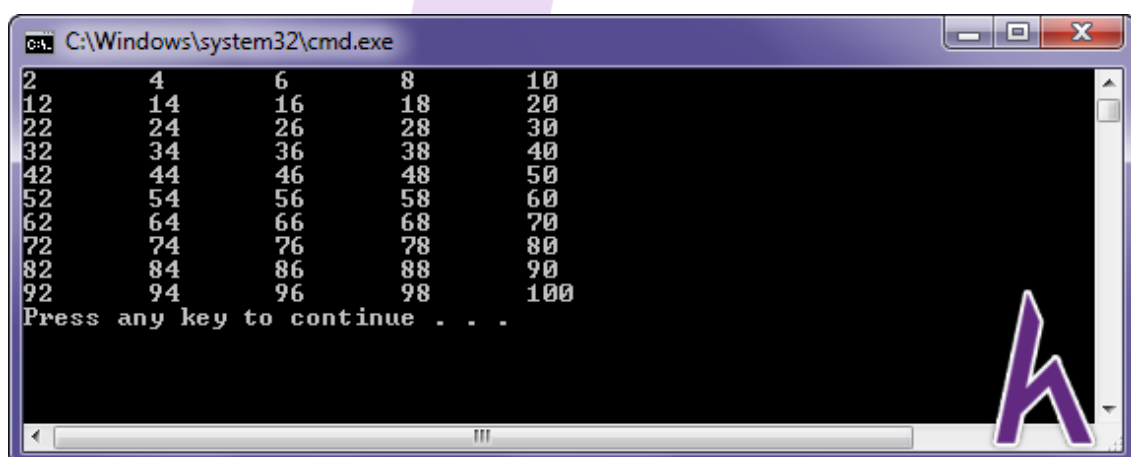
int main()
{
    // count holds the current number to print
    int count = 1; // start at 1
```

```
// Loop continually until we pass number 100
while (count <= 100)
{
    // if count is evenly divisible by 10, print a new line
    if (count % 10 == 0)
        cout << count << "\n"; // print the current number

    // if count is evenly divisible by 20, print a new line
    if (count % 20 == 0)
        cout << "\n";

    count = count + 1; // go to next number
} // end of while
return 0;
}
```

### Outputs:



```
C:\Windows\system32\cmd.exe
2      4      6      8      10
12     14     16     18     20
22     24     26     28     30
32     34     36     38     40
42     44     46     48     50
52     54     56     58     60
62     64     66     68     70
72     74     76     78     80
82     84     86     88     90
92     94     96     98     100
Press any key to continue . . .
```

### Chú ý khi thực hiện phép chia lấy phần nguyên (/):

- Nếu 2 toán hạng là số nguyên sẽ cho kết quả là số nguyên. **Ví dụ:**  $9 / 6 = 1$
- Nếu 1 trong 2, hoặc cả 2 toán hạng là số chấm động thì sẽ cho kết quả là số chấm động (**Ví dụ:**  $9.0 / 6 = 9 / 6.0 = 9.0 / 6.0 = 1.5$ ). Bạn có thể **ép kiểu** hoặc **nhân** một trong 2 toán hạng với 1.0 để có kết quả là một số chấm động.

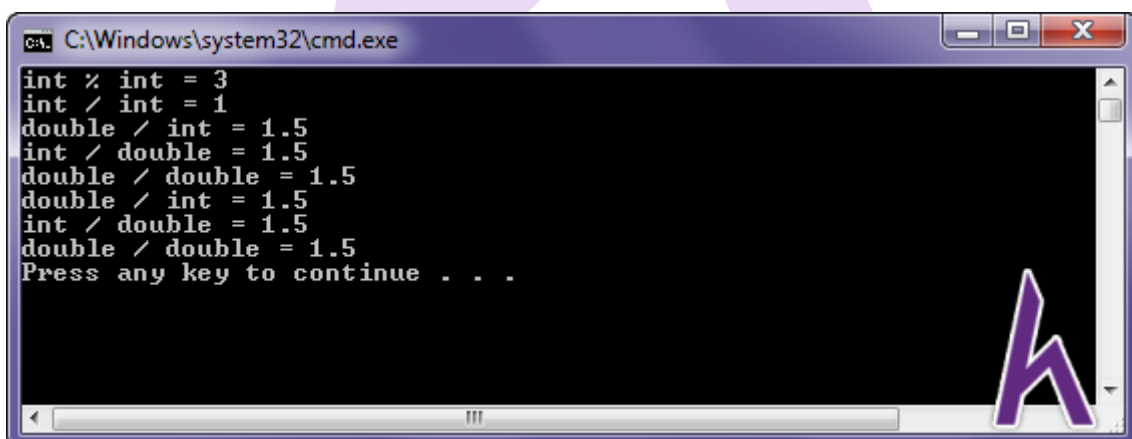
**Ví dụ:**

```
#include <iostream>
using namespace std;

int main()
{
    int x = 9;
    int y = 6;

    cout << "int % int = " << x % y << "\n";
    cout << "int / int = " << x / y << "\n";
    cout << "double / int = " << (1.0 * x) / y << "\n";
    cout << "int / double = " << x / (1.0 * y) << "\n";
    cout << "double / double = " << (1.0 * x) / (1.0 * y) << "\n";
    cout << "double / int = " << static_cast<double>(x) / y << "\n";
    cout << "int / double = " << x / static_cast<double>(y) << "\n";
    cout << "double / double = " << static_cast<double>(x) /
static_cast<double>(y) << "\n";

    return 0;
}
```

**Outputs:**

```
C:\Windows\system32\cmd.exe
int % int = 3
int / int = 1
double / int = 1.5
int / double = 1.5
double / double = 1.5
double / int = 1.5
int / double = 1.5
double / double = 1.5
Press any key to continue . . .
```

## Toán tử 1 ngôi tăng, giảm (Increment/decrement operators)

Toán tử 1 ngôi tăng, giảm khá phổ biến trong C/C++. Bảng bên dưới mô tả toán tử 1 ngôi tăng, giảm:

Operator	Symbol	Form	Operation
<b>Prefix increment (pre-increment)</b>	++	++x	Increment x, then evaluate x
<b>Prefix decrement (pre-decrement)</b>	--	--x	Decrement x, then evaluate x
<b>Postfix increment (post-increment)</b>	++	x++	Evaluate x, then increment x
<b>Postfix decrement (post-decrement)</b>	--	x--	Evaluate x, then decrement x

Toán tử 1 ngôi tăng, giảm được phân thành 2 loại:

- **Tiền tố (Prefix):** tăng hoặc giảm giá trị của biến x, sau đó x được sử dụng để tính toán.

**Ví dụ:**

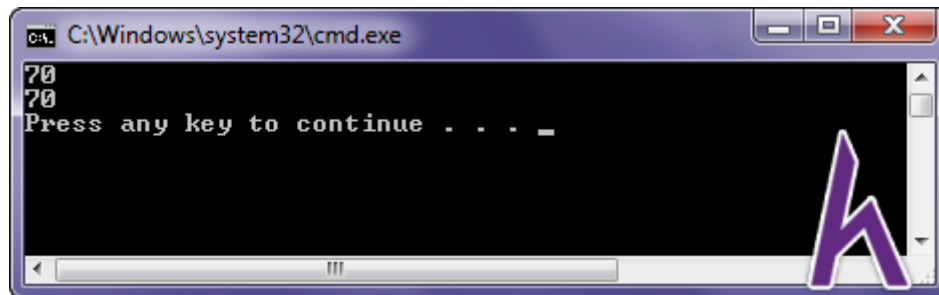
```
#include <iostream>
using namespace std;

int main()
{
    int x = 69;
    int y = ++x;    // x is now equal to 70, and 70 is assigned to y

    cout << x << endl;    // x = 70
    cout << y << endl;    // y = 70
}
```

```
    return 0;  
}
```

### Outputs:



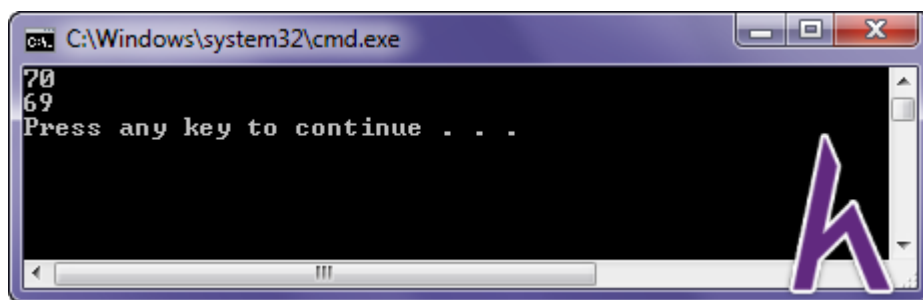
- **Hậu tố (Postfix):** sử dụng x để tính toán, sau đó tăng hoặc giảm giá trị của biến x. Đối với trường hợp này, performance sẽ giảm vì compiler phải thực hiện nhiều hơn. Đầu tiên, compiler sẽ tạo một bản sao của x, sau đó biến x được tăng hoặc giảm, mọi tính toán trong biểu thức sẽ sử dụng giá trị của bản sao và bản sao sẽ được xóa sau khi sử dụng.

### Ví dụ:

```
#include <iostream>  
using namespace std;  
  
int main()  
{  
    int x = 69;  
    int y = x++;    // x is now equal to 70, and 69 is assigned to y  
  
    cout << x << endl;    // x = 70  
    cout << y << endl;    // y = 69  
    return 0;  
}
```

### Outputs:





## Toán tử gán số học trong C++ (Arithmetic assignment operators)

**Toán tử gán** có mục đích **đưa giá trị của một hằng số, biến số, một biểu thức hoặc kết quả của một hàm vào biến được gán.**

Bảng bên dưới mô tả các toán tử gán số học trong C++:

Operator	Symbol	Form	Operation
<b>Assignment</b>	=	x = y	Assign value y to x
<b>Addition assignment</b>	+=	x += y	Add y to x
<b>Subtraction assignment</b>	-=	x -= y	Subtract y from x
<b>Multiplication assignment</b>	*=	x *= y	Multiply x by y
<b>Division assignment</b>	/=	x /= y	Divide x by y
<b>Modulus assignment</b>	%=	x %= y	Put the remainder of x / y in x

Ngoài toán tử gán bằng (=), C++ cung cấp thêm **5 toán tử gán khác** tương ứng với 5 toán tử số học cộng (+), trừ (-), nhân (\*), chia nguyên (/), chia dư (%). Nhằm giúp biểu thức trở nên ngắn gọn hơn.

**Ví dụ:**

```
#include <iostream>
using namespace std;

int sum(int a, int b)
```

```
{
    return a + b;
}

int main()
{
    int x = 9;
    int y = 9;

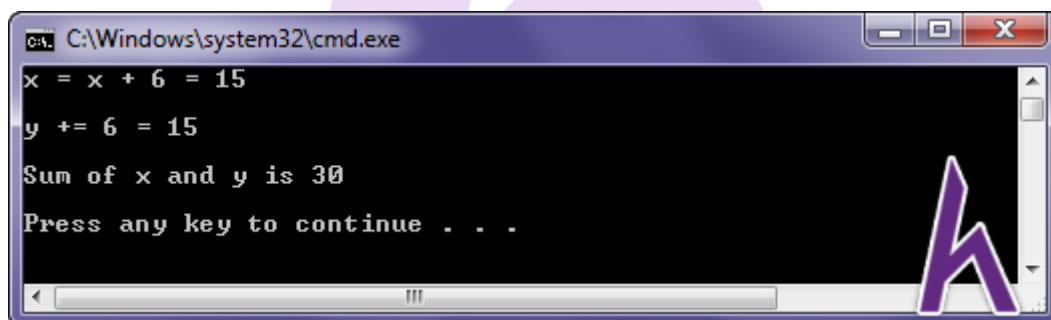
    x = x + 6; // x += 6
    cout << "x = x + 6 = " << x << endl;

    y += 6; // y = y + 6
    cout << "y += 6 = " << x << endl;

    int z = sum(x, y);
    cout << "Sum of x and y is " << z << endl;

    return 0;
}
```

### Outputs:



## Kết luận

Qua bài học này, bạn đã nắm được [Toán tử số học](#), [toán tử tăng giảm](#) và [toán tử gán trong C++](#) (Arithmetic operators).

Ở bài tiếp theo, bạn sẽ được học về [TOÁN TỬ QUAN HỆ, LOGIC, BITWISE, MISC & ĐỘ ƯU TIÊN TOÁN TỬ TRONG C++](#) (Operators).

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên **"Luyện tập – Thử thách – Không ngại khó"**.

