# Bài: Toán tử trong C#

*Xem bài học trên website để ủng hộ Kteam: Toán tử trong C#*

*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 các bài trước, chúng ta đã tìm hiểu về BIẾN và KIỂU DỮ LIỆU. Vậy "làm sao để có thể thực hiện phép toán giữa các biến cũng như xử lý dữ liệu của biến?"

Bài học hôm nay sẽ giúp chúng ta giải quyết vấn đề này. Cùng tìm hiểu chi tiết về **TOÁN TỬ TRONG C#**.

---

## Nội dung

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

- CẤU TRÚC LỆNH CỦA C# VIẾT TRÊN NỀN CONSOLE APPLICATION.
- CẤU TRÚC NHẬP XUẤT CỦA C# TRÊN NỀN CONSOLE APPLICATION.
- BIẾN & KIỂU DỮ LIỆU trong C#.

Trong bài học này, chúng ta sẽ cùng tìm hiểu các vấn đề:

- Toán tử là gì? Có mấy loại toán tử?
- Cú pháp và ý nghĩa của từng toán tử.
- Độ ưu tiên của toán tử.
- Ví dụ chương trình sử dụng một số toán tử.

---

## Toán tử là gì?

Toán tử được định nghĩa như sau:

- Là một công cụ để thao tác với dữ liệu.
- Một toán tử là một ký hiệu dùng để đại diện cho một thao tác cụ thể được thực hiện trên dữ liệu.

Có 6 loại toán tử cơ bản:

- Toán tử toán học.
- Toán tử quan hệ.
- Toán tử logic.
- Toán tử khởi tạo và gán.
- Toán tử so sánh trên bit.
- Toán tử khác.

---

## Cú pháp và ý nghĩa của từng toán tử

### Toán tử toán học

Giả sử biến a có giá trị là 10 biến b có giá trị là 9

| Toán tử | Mô tả | Ví dụ |
|---|---|---|

| | | |
|---|---|---|
| + | Thực hiện **cộng** hai toán hạng | **a + b** kết quả bằng 19 |
| - | Thực hiện **trừ** hai toán hạng | **a - b** kết quả bằng 1 |
| * | Thực hiện **nhân** hai toán hạng | **a * b** kết quả bằng 90 |
| / | Thực hiện **chia lấy phần nguyên** hai toán hạng nếu 2 toán hạng là số nguyên. Ngược lại thì thực hiện chia bình thường | **a / b** kết quả bằng 1 |
| % | Thực hiện **chia lấy dư** | **a%b** kết quả bằng 1 |
| ++ | **Tăng** giá trị lên 1 đơn vị | **a++** kết quả bằng 11 |
| -- | **Giảm** giá trị xuống 1 đơn vị | **a--** kết quả bằng 9 |

**Lưu ý:** đối với toán tử ++ và -- cần phân biệt a++ và ++a (hoặc a-- và --a):

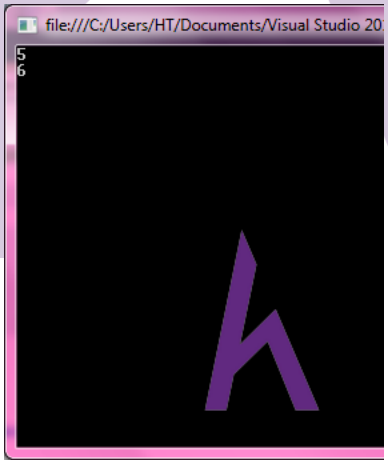
- **a++**: là sử dụng giá trị của biến a để thực hiện biểu thức trước rồi mới thực hiện tăng lên 1 đơn vị. Tương tự cho a--.
- **++a**: là tăng giá trị biến a lên 1 đơn vị rồi mới sử dụng biến a để thực hiện biểu thức. Tương tự cho --a.

**Ví dụ:**

**C#:**

```
static void Main(string[] args)
        {
            int i = 5, j = 5;

            Console.WriteLine(i++); // Sử dụng giá trị i để in ra rồi mới tăng i
            Console.WriteLine(++j); // Tăng j lên rồi mới in giá trị j ra màn hình

            Console.ReadKey();
        }
```

- Kết quả khi chạy chương trình:




- Biến **i** đã được sử dụng để in ra màn hình rồi sau đó mới tăng lên 1 đơn vị cho nên màn hình mới in ra số 5.
- Biến **j** thì ngược lại, tăng lên 1 đơn vị trước rồi mới sử dụng giá trị mới để in ra màn hình nên màn hình in ra số 6.

## Toán tử quan hệ

Giả sử biến a có giá trị bằng 10 và biến b có giá trị bằng 9:

| Toán tử | Mô tả | Ví dụ |
|---|---|---|
| == | So sánh 2 toán hạng có bằng nhau hay không. Nếu **bằng** thì trả về true nếu **không bằng** thì trả về false | a == b sẽ trả về false |
| != | So sánh 2 toán hạng có bằng nhau hay không. Nếu **không bằng** thì trả về true nếu **bằng** thì trả về false | a != b sẽ trả về true |
| > | So sánh 2 toán hạng bên trái có **lớn hơn** toán hạng bên phải hay không. Nếu **lớn hơn** thì trả về true nếu **không lớn hơn** thì trả về false | a > b sẽ trả về true |
| < | So sánh 2 toán hạng bên trái có **nhỏ hơn** toán hạng bên phải hay không. Nếu **nhỏ hơn** thì trả về true nếu **không nhỏ hơn** thì trả về false | a < b sẽ trả về false |
| >= | So sánh 2 toán hạng bên trái **có lớn hơn hoặc bằng** toán hạng bên phải hay không. Nếu **lớn hơn hoặc bằng** thì trả về true nếu **nhỏ hơn** thì trả về false | a >= b sẽ trả về true |
| <= | So sánh 2 toán hạng có **nhỏ hơn hoặc bằng** hay không. Nếu **nhỏ hơn hoặc bằng** thì trả về true nếu **lớn hơn** thì trả về false | a <= b sẽ trả về false |

**Lưu ý:**

1. Các toán tử quan hệ này chỉ áp dụng cho số hoặc ký tự.
2. Hai toán hạng hai bên phải cùng loại (cùng là số hoặc cùng là ký tự).
3. Bản chất của việc so sánh 2 ký tự với nhau là so sánh mã ASCII của các ký tự đó.

- **Ví dụ:** so sánh '**A**' và '**B**' bản chất là so sánh số **65** với **66**.

4. Không nên sử dụng các toán tử trên để so sánh các chuỗi với nhau vì bản chất việc so sánh chuỗi là so sánh từng ký tự tương ứng với nhau mà so sánh ký tự là so sánh mã ASCII của ký tự đó như vậy ký tự '**K**' sẽ khác ký tự '**k**'. Để so sánh hai chuỗi người ta thường dùng hàm so sánh chuỗi đã được hỗ trợ sẵn (sẽ tìm hiểu ở những bài tiếp theo).

## Toán tử logic

Giả sử mệnh đề A là đúng và mệnh đề B là sai:

| Toán tử | Mô tả | Ví dụ |
|---|---|---|
| && | Hay còn gọi là toán tử logic AND (và). Trả về true nếu **tất cả** toán hạng đều mang giá trị true. Và trả về false nếu có **ít nhất 1** toán hạng mang giá trị false. | A && B kết quả là false |

| | | |
|---|---|---|
| \|\| | Hay còn gọi là toán tử logic OR (hoặc). Trả về true nếu có **ít nhất 1** toán hạng mang giá trị true. Và trả về false nếu **tất cả** toán hạng đều mang giá trị false. | A \|\| B kết quả là true. |
| ! | Hay còn gọi là toán tử logic NOT (phủ định). Có chức năng **đảo ngược** trạng thái logic của toán hạng. Nếu toán hạng đang mang giá trị true thì kết quả sẽ là false và ngược lại. | !A kết quả là false |

**Lưu ý:**

- Các toán tử **&&** và **||** có thể áp dụng đồng thời nhiều toán hạng, ví dụ như: **A && B && C || D || K** (Thứ tự thực hiện sẽ được trình bày ở phần sau).
- Các toán hạng trong biểu thức chứa toán tử logic phải trả về true hoặc false.

## Toán tử khởi tạo và gán

Toán tử khởi tạo và gán thường được sử dụng nhằm mục đích lưu lại giá trị cho một biến nào đó. Một số toán tử khởi tạo và gán hay được sử dụng:

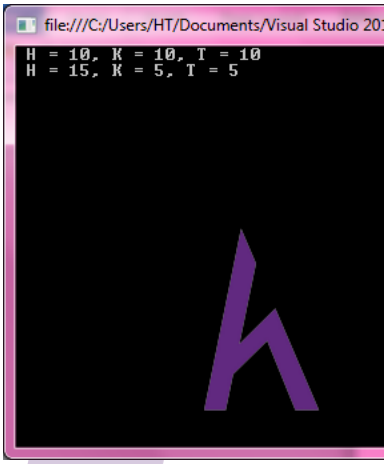
| Toán tử | Mô tả | Ví dụ |
|---|---|---|
| = | Gán giá trị của toán hạng **bên phải** cho toán hạng **bên trái**. | **K = 10** sẽ gán 10 cho biến **K** |
| += | Lấy toán hạng **bên trái cộng** toán hạng **bên phải** sau đó gán kết quả lại cho toán hạng **bên trái**. | **K += 1** tương đương với **K = K + 1** |
| -= | Lấy toán hạng **bên trái trừ** toán hạng **bên phải** sau đó gán kết quả lại cho toán hạng **bên trái**. | **K -= 1** tương đương với **K = K – 1** |
| *= | Lấy toán hạng **bên trái nhân** toán hạng **bên phải** sau đó gán kết quả lại cho toán hạng **bên trái**. | **K *= 1** tương đương với **K = K * 1** |
| /= | Lấy toán hạng **bên trái chia lấy phần nguyên** với toán hạng **bên phải** sau đó gán kết quả lại cho toán hạng **bên trái**. | **K /= 1** tương đương với **K = K / 1** |
| %= | Lấy toán hạng **bên trái chia lấy dư** với toán hạng **bên phải** sau đó gán kết quả lại cho toán hạng **bên trái**. | **K %= 1** tương đương với **K = K % 1** |

Một số lưu ý khi sử dụng các toán tử trên:

1. Toán tử bên trái thường là một biến, còn toán tử bên phải có thể là biến có thể là biểu thức đều được.
2. Một phép toán gán hoặc khởi tạo có thể được sử dụng như là toán hạng bên phải cho một phép gán hoặc khởi tạo khác. Ví dụ:

**C#:**

```
int H, K, T;
H = K = T = 10;
Console.WriteLine(" H = {0}, K = {1}, T = {2}", H, K, T);

H += K = T = 5;
Console.WriteLine(" H = {0}, K = {1}, T = {2}", H, K, T);
```




- Phép toán **H =K =T = 10** sẽ thực hiện gán 10 cho biến T sau đó gán giá trị biến T cho biến K sau đó gán giá trị biến K cho biến H. Như vậy ta được cả 3 biến H K T đều có giá trị bằng 10.
- Phép toán **H += K = T = 5** sẽ thực hiện gán 5 cho biến T sau đó gán giá trị biến T cho biến K sau đó lấy H + K gán kết quả cho biến H. Cuối cùng ta được H = 15, K = 5, T = 5.

## Toán tử so sánh trên bit

Các toán tử so sánh trên bit cũng ít gặp nên mình chỉ giới thiệu qua cho các bạn tham khảo thôi chứ chúng ta không giới thiệu rõ phần này.

Giả sử a có giá trị bằng 10 và b có giá trị bằng 9. Giá trị biến a đổi ra nhị phân là 1010 và giá trị biến b đổi ra nhị phân là 1001

| Toán tử | Mô tả | Ví dụ |
|---|---|---|
| & | Sao chép bit 1 tới kết quả nếu nó **tồn tại trong cả hai** toán hạng tại vị trí tương ứng, ngược lại thì bit kết quả bằng 0 | **a&b** sẽ cho kết quả là 1000 tương đương với số 8 trong hệ thập phân |
| \| | Sao chép bit 1 tới kết quả nếu nó **tồn tại ở một trong hai** toán hạng tại vị trí tương ứng, ngược lại thì bit kết quả bằng 0 | **a\|b** sẽ cho kết quả 1011 tương đương với số 11 trong hệ thập phân |
| ^ | Sao chép bit 1 tới kết quả nếu **nó chỉ tồn tại ở một** toán hạng tại vị trí tương ứng, ngược lại thì bit kết quả bằng 0 | **a^b** sẽ cho kết quả 0011 tương đương với số 3 trong hệ thập phân |
| ~ | Dùng để **đảo bit** 0 thành 1 và ngược lại 1 thành 0 | **~a** sẽ cho kết quả 0101 |

| | | |
|---|---|---|
| << | Dịch trái n bit. Giá trị toán hạng bên trái sẽ được **dịch trái** n bit với n được xác định bởi toán hạng bên phải | **a<<2** sẽ cho kết quả 101000 |
| >> | Dịch phải n bit. Giá trị toán hạng bên trái sẽ được **dịch phải** n bit với n được xác định bởi toán hạng bên phải | **a>>2** sẽ cho kết quả 0010 |

## Toán tử khác

Ngoài những toán tử đã giới thiệu ở trên chúng ta vẫn còn nhiều toán tử khác cũng hay sử dụng:

| Toán tử | Mô tả | Ví dụ |
|---|---|---|
| **sizeof()** | Trả về kích cỡ của một kiểu dữ liệu | sizeof(int) sẽ trả về 4 |
| **typeof()** | Trả về kiểu của một lớp (khái niệm về lớp sẽ được trình bày trong bài CLASS TRONG C#) | typeof(string) sẽ trả về **System.String** |
| **new** | Cấp phát vùng nhớ mới, áp dụng cho kiểu dữ liệu tham chiếu | DateTime dt = new DateTime() (sẽ được trình bày chi tiết trong STRUCT TRONG C#) |
| **is** | Xác định đối tượng có phải là một kiểu cụ thể nào đó hay không. Nếu đúng sẽ trả về true ngược lại trả về false | Sẽ được trình bày trong những bài sau |
| **as** | Ép kiểu mà không gây ra lỗi. Nếu ép kiểu không thành công sẽ trả về null | Sẽ được trình bày chi tiết trong ÉP KIỂU TRONG C# |
| **? :** | Được gọi là **toán tử 3 ngôi**. Tương đương với cấu trúc điều kiện (sẽ được trình bày ở bài CẤU TRÚC RẼ NHÁNH TRONG C#)<br>**Cú pháp:**<br>**(toán hạng 1) ? (toán hạng 2) : (toán hạng 3)**<br>**Ý nghĩa:** trả về toán hạng 2 nếu toán hạng 1 là true và ngược lại trả về toán hạng 3 | **(1 < 2) ? 1 : 0**<br>kết quả là 1 vì **toán hạng 1** là (1 < 2) là đúng nên trả về **toán hạng 2** là 1 |
| **,** | Sử dụng toán tử "**,**" để kết nối nhiều biểu thức lại với nhau.<br>**Cú pháp:**<br>**(biểu thức 1, biểu thức 2)**<br>**Ý nghĩa:** Duyệt qua **biểu thức 1** sau đó duyệt qua **biểu thức 2** và trả về giá trị của biểu thức 2 | **(t = 5, 2)** sẽ duyệt qua biểu thức 1 là **t = 5**, thực hiện gán 5 cho t sau đó duyệt qua biểu thức 2 là 2, cuối cùng trả về giá trị là 2 |

# Độ ưu tiên của toán tử

Độ ưu tiên của các toán tử biểu thị cho việc toán tử nào được ưu tiên thực hiện trước trong câu lệnh. Độ ưu tiên được tóm tắt ở bảng sau:

| Mức | Toán tử | Thứ tự |
|---|---|---|
| Cao nhất | () [] . | Trái sang phải |
| | + - ++ -- ! ~ new sizeof() typeof() | Phải sang trái |
| | * / % | Trái sang phải |
| | << >> | Trái sang phải |
| | < <= > >= | Trái sang phải |
| | == != | Trái sang phải |
| | & ^ \| | Trái sang phải |
| | && \|\| | Trái sang phải |
| | ? : | |
| | = += -= *= /= %= | Phải sang trái |
| Thấp nhất | , | |

Bảng thống kê trên chỉ thể hiện những toán tử chúng ta đã học, ngoài ra vẫn còn nhiều toán tử khác những ít khi sử dụng nên không đề cập đến.

Từ bảng trên ta có thể rút ra được 1 kinh nghiệm đó là nếu muốn biểu thức nào được thực hiện trước ta chỉ cần nhóm chúng vào cặp ngoặc tròn **( )** là được!

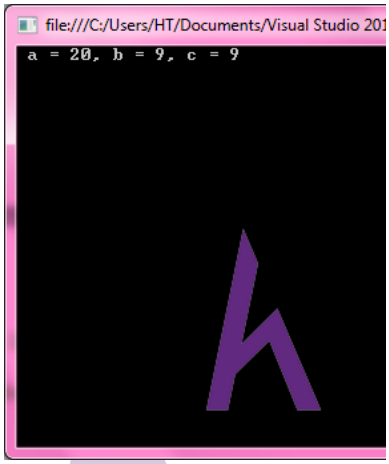
# Ví dụ chương trình sử dụng một số toán tử

## Ví dụ 1: Các phép toán cơ bản

**C#:**

```
static void Main(string[] args)
        {
            int a, b, c;

            a = b = (c = 9) + 1; // khởi tạo giá trị: a = 10, b = 10, c = 9
            a += b; // tương đương a = a + b
            b = c++; // thực hiện gán giá trị c cho biến b sau đó thực hiện c = c + 1
            --c; // thực hiện c = c - 1

            Console.WriteLine(" a = {0}, b = {1}, c = {2}", a, b, c);

            Console.ReadKey();

        }
```

- Các bạn chú ý vào **phần chú thích** (chữ màu xanh lá) để hiểu ý nghĩa từng câu lệnh. Kết quả khi chạy chương trình:




> **Lưu ý:** đối với phép toán --c hoặc c-- khi đứng độc lập thì không có khác biệt gì.

## Ví dụ 2: Kết hợp các phép toán để viết chương trình kiểm tra số nhập vào là số chẵn số lẻ:

**C#:**

```
static void Main(string[] args)
        {
            string strSoNguyen; // Biến chứa dữ liệu nhập vào từ bàn phím
            int SoNguyen; // Biến chứa số nhập vào từ bàn phím
            string KetQua; // Biến chứa kết quả kiểm tra số vừa nhập là chẵn hay lẻ

            strSoNguyen = Console.ReadLine(); // Đọc dữ liệu nhập vào từ bàn phím (dữ liệu này ở dạng chuỗi) sau đó gán giá trị vào
biến strSoNguyen
            SoNguyen = Int32.Parse(strSoNguyen); // Ép kiểu dữ liệu vừa nhập vào (dạng chuỗi) sang dạng số rồi gán giá trị vào biến
SoNguyen
            KetQua = (SoNguyen % 2 == 0) ? "so chan" : "so le"; // Sử dụng toán tử 3 ngôi để kiểm tra số chẵn lẻ

            Console.WriteLine("{0} la {1}", SoNguyen, KetQua); // In kết quả ra màn hình

            Console.ReadKey();
        }
```

Đầu tiên ta có 3 biến:

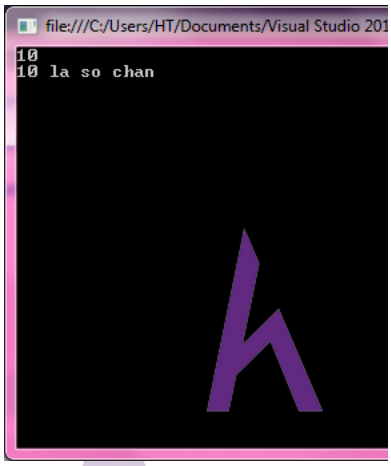
- **strSoNguyen**: Chứa dữ liệu nhập vào từ bàn phím. Vì dữ liệu nhập vào từ bàn phím mặc định là dạng chuỗi nên cần biến kiểu chuỗi để chứa giá trị.
- **SoNguyen**: Chứa dữ liệu nhập vào từ bàn phím ở dạng số. Từ dữ liệu dạng chuỗi của biến **strSoNguyen** ta ép kiểu sang kiểu số để dễ xử lý (chi tiết về ép kiểu sẽ được trình bày ở bài ÉP KIỂU TRONG C#)
- **KetQua**: Chứa kết quả kiểm tra số vừa nhập là chẵn hay lẻ. Kết quả này ở dạng chuỗi để có thể in ra màn hình luôn.

Tiếp theo ta nhận kết quả nhập từ bàn phím bằng lệnh **Console.ReadLine()** rồi gán giá trị cho biến **strSoNguyen**.

Ép kiểu kết quả vừa nhập sang dạng số rồi gán giá trị vào biến **SoNguyen**.

Sử dụng toán tử 3 ngôi kiểm tra xem số vừa nhập có chia hết cho 2 hay không (nếu chia hết cho 2 thì phép chia lấy dư với 2 sẽ cho kết quả là 0 và biểu thức **SoNguyen % 2 == 0** sẽ trả về true ngược lại sẽ trả về false). Nếu chia hết thì trả về chuỗi **"so chan"** ngược lại trả về chuỗi **"so le"**.

Cuối cùng là in kết quả ra màn hình.




Để có thể sử dụng thành thạo các toán tử trên các bạn nên luyện tập thường xuyên, vận dụng kết hợp nhiều toán tử để giải quyết vấn đề!

## Kết luận

Nội dung bài này giúp các bạn nắm được:

- Khái niệm về toán tử và các loại toán tử.
- Cú pháp và ý nghĩa của từng toán tử.
- Độ ưu tiên của các toán tử.
- Ví dụ chương trình sử dụng một số toán tử.

Bài học sau chúng ta sẽ cùng tìm hiểu một khái niệm tiếp theo đó là **HẰNG TRONG C#**

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**".