Định nghĩa toán tử trong [Wikipedia]:

*Trong toán học, một toán tử (tiếng Anh operator, phân biệt với operation - phép toán) là một hàm, thông thường có một vai trò quan trọng trong một lĩnh vực nào đấy. Chẳng hạn trong đại số tuyến tính có “toán tử tuyến tính” (linear operator). Trong giải tích có “toán tử vi phân” (differential operator)… Thông thường, một “toán tử” là một hàm tác động lên các hàm khác; hoặc nó có thể là tổng quát hóa của một hàm, như trong đại số tuyến tính.*

Các kiểu toán tử trong C:

* Toán tử số học
* Toán tử quan hệ
* Toán tử logic
* Toán tử so sánh bit
* Toán tử gán
* Toán tử hỗn hợp

**Toán tử số học**

Giả sử khai báo: int A=5, B=6; float C=1.2;

| **Toán tử** | **Miêu tả** | **Ví dụ** |
| --- | --- | --- |
| + | Cộng 2 toán hạng | A + B sẽ cho kết quả là 11 |
| - | Trừ 2 toán hạng | A - B sẽ cho kết quả là -1 |
| \* | Nhân 2 toán hạng | A \* B sẽ cho kết quả là 30 |
| / | Chia 2 toán hạng | B / A sẽ cho kết quả là 0 (2 toán hạng đều là số nguyên thì kết quả là phần nguyên của thương) A / C sẽ cho kết quả là 4.166667 (1 trong 2 toán hạng là số thực dấu phẩy động thì thương là số thực dấu phẩy động) |
| % | Chia lấy phần dư | B % A sẽ cho kết quả là 1 |

**Toán tử so sánh(quan hệ): Trả về giá trị là true (đúng) hoặc false (sai)**

int A=5, B=6;

* < (nhỏ hơn) A < B -> *true*
* <= (nhỏ hơn hoặc bằng) A <= B -> *true*
* > (lớn hơn) A > B -> false
* >= (lớn hơn hoặc bằng) A >= B -> *false*
* == (bằng) A == B -> *false*
* != (khác) A != B -> *true*

**Toán tử logic: Trả về giá trị là true (đúng) hoặc false (sai)**

Trong ngôn ngữ C, 2 trạng thái true(đúng) và false(sai) được biểu diễn bởi các số nguyên int:

* Số 0 biểu diễn cho trạng thái false (sai).
* Tất cả các số nguyên khác 0 biểu diễn cho trạng thái true (đúng).

Các toán tử logic:

* && (and) trả về true khi cả 2 toán hạng đều đúng. Ngược lại trả về false.
  + (5<6) && (7<8) -> *true*
  + (5>6) && (7<8) -> *false*
  + (5>6) && (7>8) -> *false*
* || (or) trả về true khi ít nhất một trong 2 toán hạng đúng. Ngược lại trả về false.
  + (5<6) && (7<8) -> *true*
  + (5>6) && (7<8) -> *true*
  + (5>6) && (7>8) -> *false*
* ! (not) trả về true khi toán hạng (đằng sau dấu ! sai). Ngược lại trả về false.
  + !(7>8) -> *true*
  + !(7<8) -> *false*

**Toán tử thao tác trên bit**

* & (and bit)
* | (or bit)
* ~ (phủ định)
* >> (dịch bit sang phải)
* << (dịch bit sang trái)

**Toán tử gán**

Sử dụng dấu = cho việc gán giá trị vào biến.

Ví dụ:

1int a,b,c;

2int main(void)

3{

4 a = 5; *// Gán cho a giá trị là 5*

5 c = b = a; *// Gán cho b và c giá trị bằng giá trị của a*

6 return 0;

7}

*C*

Các toán tử gán mở rộng:

| **Toán tử** | **Ví dụ** | **Tương đương với** |
| --- | --- | --- |
| += | C += A | C = C + A |
| -= | C -= A | C = C - A |
| \*= | C \*= A | C = C \* A |
| /= | C /= A | C = C / A |
| %= | C %= A | C = C % A |
| <<= | C <<= 2 | C = C << 2 |
| >>= | C >>= 2 | C = C >> 2 |
| &= | C &= 2 | C = C & 2 |
| ^= | C ^= 2 | C = C ^ 2 |
| |= | C |= 2 | C = C | 2 |

**Toán tử tăng giảm**

* ++ là toán tử tăng
  + ++i tương đương với i = i + 1
* -- là toán tử giảm
  + --i tương đương với i = i - 1
* Có 2 cách viết ++i và i++ nhưng ý nghĩa của chúng khác nhau:
  + ++i thì i được tăng trước sau đó sẽ lấy kết quả để thực hiện biểu thức
  + i++ thì i được đưa vào thực hiện biểu thức trước sau đó mới tăng i lên.

**Một số toán tử khác**

| **Toán tử** | **Miêu tả** | **Ví dụ** |
| --- | --- | --- |
| sizeof() | Trả lại kích cỡ của một biến | sizeof(a), với a là integer, thì sẽ trả lại kết quả là 4. |
| & | Trả lại địa chỉ của một biến. | &a sẽ cho địa chỉ thực sự của biến a. |
| \* | Trỏ tới một biến. | \*a sẽ trỏ tới biến a. |
| <điều kiện>? X:Y | Biểu thức điều kiện | Nếu điều kiện đúng ? thì trả về giá trị X : Nếu không thì trả về giá trị Y |
| <toán hạng 1>,<toán hạng 2> | Ước lượng giá trị toán hạng 1, ước lượng giá trị toán hạng 2 và trả về giá trị toán hạng 2 là giá trị cuối cùng | t = (x=10, x+5) sẽ gán giá trị x <- 10; thực hiện phép tính x + 5 = 10 + 5 = 15 và gán giá trị 15 cho t |

**Bảng thứ tự ưu tiên thực hiện của toán tử (theo thứ tự giảm dần mức độ ưu tiên):**

| **Loại** | **Toán tử** | **Thứ tự ưu tiên** |
| --- | --- | --- |
| Postfix | () [] -> . ++ - - | Trái sang phải |
| Unary | + - ! ~ ++ - - (type) \* & sizeof | Phải sang trái |
| Tính nhân | \* / % | Trái sang phải |
| Tính cộng | + - | Trái sang phải |
| Dịch chuyển bit | << >> | Trái sang phải |
| So sánh không ngang bằng | < <= > >= | Trái sang phải |
| So sánh ngang bằng | == != | Trái sang phải |
| Phép AND bit | & | Trái sang phải |
| Phép XOR bit | ^ | Trái sang phải |
| Phép OR bit | | | Trái sang phải |
| Phép AND logic | && | Trái sang phải |
| Phép OR logic | || | Trái sang phải |
| Điều kiện | ? : | Phải sang trái |
| Gán | = += -= \*= = %= >>= <<= &= ^= |= | Phải sang trái |
| Dấu phảy | , | Trái sang phải |

**Ép kiểu**

Đôi khi chúng ta cần chuyển đổi giá trị một biểu thức sang kiểu dữ liệu khác. Ví dụ trong trường hợp ta muốn thực hiện phép toán chia lấy phần dư của 2 số nguyên, nhưng lại được lưu trong 2 biến kiểu float, ta không thể áp dụng trực tiếp toán tử % cho 2 biến đó. Bạn chạy chương trình thế này sẽ báo lỗi:

1#include <stdio.h>

2int main(void)

3{

4 int a = 5, c;

5 float b = 6;

6 c = a % (int)b;

7 printf("%d", c);

8 return 0;

9}

*C*

Vì thế cần ép kiểu theo cú pháp: (<kiểu dữ liệu>) <biểu thức> để lấy giá trị từ biến b, đổi sang số nguyên để thực hiện phép %. Code đúng như sau:

1#include <stdio.h>

2int main(void)

3{

4 int a = 5, c;

5 float b = 6;

6 c = a % (int)b;

7 printf("%d", c);

8 return 0;

9}