TOÁN TỬ (OPERATOR)

NỘI DUNG

1. TOÁN TỬ GÁN: (ASSIGNMENT OPERATOR)

Cú pháp: `[Toán hạng 1] = [Toán hạng 2]`

Toán hạng 1 ⟵ Toán hạng 2

Ý nghĩa: Gán giá trị của toán hạng 2 cho toán hạng 1.

Ví dụ:

```
ban_kinh = 100;    // Gán giá trị 100 cho biến ban_kinh
chuvi = 1000;    // Gán giá trị của chuvi là 1000
```

TOÁN TỬ TOÁN HỌC: (ARITHMETIC OPERATOR)

Toán tử	Ý nghĩa	Ví dụ
+	Cộng 2 toán hạng	X = Y + Z X = 100 + 200; // 300
-	Trừ 2 toán hạng	X = Y - Z X = 200 - 100; // 100
*	Nhân 2 toán hạng	X = Y * Z X = 10 * 50; // 500
/	Chia 2 toán hạng	X = Y / Z X = 300 / 200; // 1
%	Chia dư 2 toán hạng	X = Y % Z X = 300 % 200; // 100

TOÁN TỬ TOÁN HỌC: (ARITHMETIC OPERATOR)

Nếu bạn chia 2 số nguyên (int, long long) cho nhau thì phép chia ở trên sẽ là phép chia nguyên, tức là nó chỉ lấy phần nguyên và bỏ phần thập phân ở thương. Nếu muốn kết quả ở số thập phân thì ít nhất 1 trong 2 số phải ở kiểu số thực và biến thương phải ở dạng số thực.

Dùng kiểu int để lưu biến thương

```
#include <stdio.h>

int main(){
    int a = 100, b = 30;
    int thuong = a / b;
    printf("%d", thuong);
    return 0;
}
```

Output: 3

Biến thương có kiểu là float/double

```
#include <stdio.h>

int main(){
    int a = 100, b = 30;
    float thuong = a / b;
    printf("%.2f", thuong);
    return 0;
}
```

Output: 3.00

TOÁN TỬ TOÁN HỌC: (ARITHMETIC OPERATOR)

Dùng float/double để lưu biến thương và ép kiểu

```c
#include <stdio.h>

int main(){
    int a = 100, b = 30;
    float thuong = (float)a / b;
    printf("%.2f", thuong);
    return 0;
}
```

Output: 3.33

Nhân a với số 1.0

```c
#include <stdio.h>

int main(){
    int a = 100, b = 30;
    float thuong = 1.0 * a / b;
    printf("%.2f", thuong);
    return 0;
}
```

Output: 3.33

Bạn thử chạy chương trình khi sử dụng 1 nhân với a thay vì 1.0 để thấy sự khác nhau

TOÁN TỬ TOÁN HỌC: (ARITHMETIC OPERATOR)

Nếu bạn nhân 2 số nguyên int với nhau mà kết quả của tích vượt giới hạn lưu của số int (khoảng 2 tỉ) thì kết quả sẽ bị tràn, ngay cả khi bạn sử dụng biến long long để lưu biến tích. Việc cần xử lý ở đây là can thiệp vào phép nhân.

Ví dụ sai:

```
#include <stdio.h>

int main(){
    int a = 1000000, b = 1000000;
    long long tich = a * b; // sai
    printf("%lld", tich);
    return 0;
}
```

Output: -727379968

Ví dụ đúng: Ép a sang long long

```
#include <stdio.h>

int main(){
    int a = 1000000, b = 1000000;
    long long tich = (long long)a * b;
    printf("%lld", tich);
    return 0;
}
```

Output: 1000000000000

TOÁN TỬ TOÁN HỌC: (ARITHMETIC OPERATOR)

Nếu bạn nhân 2 số nguyên int với nhau mà kết quả của tích vượt giới hạn lưu của số int (khoảng 2 tỉ) thì kết quả sẽ bị tràn, ngay cả khi bạn sử dụng biến long long để lưu biến tích. Việc cần xử lý ở đây là can thiệp vào phép nhân.

Ví dụ đúng: Ép a sang long long

```
#include <stdio.h>

int main(){
    int a = 1000000, b = 1000000;
    long long tich = (long long) a * b;
    printf("%lld", tich);
    return 0;
}
```

Output: 1000000000000

Ví dụ đúng: Nhân 1ll vào tích a * b

```
#include <stdio.h>

int main(){
    int a = 1000000, b = 1000000;
    long long tich = 1ll * a * b;
    printf("%lld", tich);
    return 0;
}
```

Output: 1000000000000

TOÁN TỬ SO SÁNH: (COMPARISION OPERATOR)

Khi bạn sử dụng các toán tử so sánh để so sánh 2 toán hạng thì kết quả của phép so sánh sẽ trả về đúng hoặc sai.

Toán tử	Ý nghĩa	Ví dụ
>	Lớn hơn	10 > 50 : false
>=	Lớn hơn hoặc bằng	20>=10 : true
<	Nhỏ hơn	10 < 50 : true
<=	Nhỏ hơn hoặc bằng	20 <= 20 : true
!=	So sánh khác	10 != 20 : true
==	So sánh bằng	10 == 10 : true

TOÁN TỬ SO SÁNH: (COMPARISION OPERATOR)

Ví dụ	Kết quả
10 == 20	Sai
10 <= 20	Đúng
10 == 10	Đúng
50 != 50	Sai
100 <= 100	Đúng

TOÁN TỬ LOGIC: (LOGICAL OPERATOR)

Trong trường hợp bạn muốn kết hợp nhiều phép so sánh lại với nhau, ta sử dụng 3 cổng logic cơ bản của máy tính là AND, OR, NOT

Toán tử	Ký hiệu	Ví dụ
AND	&&	(x >= 10) && (x <= 50)
OR	||	(a >= 10) || (a % 2 == 0)
NOT	!	!(a <= 10)

Để tính toán giá trị cuối cùng của biểu thức các bạn xác định giá trị của từng mệnh đề thành phần, sau đó áp dụng bảng chân lý của các cổng logic để tìm giá trị của biểu thức ban đầu.

TOÁN TỬ LOGIC: (LOGICAL OPERATOR)

Bảng chân lý của cổng AND

A	B	A && B
0	0	0
0	1	0
1	0	0
1	1	1

Chú ý: Cổng AND chỉ cho kết quả đúng khi mọi mệnh đề thành phần đều có giá trị đúng, sai trong các trường hợp còn lại.

TOÁN TỬ LOGIC: (LOGICAL OPERATOR)

Bảng chân lý của cổng OR

A	B	A || B
0	0	0
0	1	1
1	0	1
1	1	1

Chú ý: Cổng OR chỉ cho kết quả sai khi mọi mệnh đề thành phần đều có giá trị sai, đúng khi chỉ cần ít nhất 1 mệnh đề thành phần có giá trị đúng

TOÁN TỬ LOGIC: (LOGICAL OPERATOR)

Bảng chân lý của cổng NOT

A	NOT A
0	1
1	0

TOÁN TỬ TĂNG GIẢM: (INCREMENT, DECREMENT OPERATOR)

Để tăng hoặc giảm giá trị của một biến đi 1 đơn vị ta có thể sử dụng toán tử tăng, giảm này sẽ thuật tiện hơn.

Toán tử	Ý nghĩa	Ví dụ
++	Tăng trước 1 đơn vị	++a
++	Tăng sau 1 đơn vị	a++
--	Giảm trước 1 đơn vị	--a
--	Giảm sau 1 đơn vị	a--

TOÁN TỬ TĂNG GIẢM: (INCREMENT, DECREMENT OPERATOR)

Ví dụ	Kết quả	Giải thích

int a = 100;
int b = a++;
printf("%d %d", a, b);
	101 100	Đây là tăng sau, tức ban đầu câu lệnh sẽ gán giá trị của a là 100 cho b, sau đó mới tăng giá trị của a lên 101

int a = 100;
int b = ++a;
printf("%d %d", a, b);
	101 101	Đây là tăng trước, giá trị của a sẽ được tăng ngay lập tức lên 101, sau đó mới lấy giá trị đó và gán cho b

TOÁN TỬ 3 NGÔI: (CONDITIONAL OPERATOR)

Cú pháp: `[Biểu thức so sánh] ? [Giá trị trả về khi biểu thức đúng] : [Giá trị trả về khi biểu thức sai];`

Ví dụ	Kết quả	Giải thích
int x = 10 < 20 ? 10 : 20;	x = 10	Nếu 10 < 20 thì vế phải sẽ trả về 10, ngược lại sẽ trả về 20. Sau đó giá trị này được gán cho x
int y = (10 < 20) && (20 > 20) ? 5 : 10;	y = 10	Nếu 10 < 20 và 20 > 20 thì sẽ gán 5 cho y, ngược lại gán 10 cho y

Không nên lạm dụng toán tử 3 ngôi

CHÚ Ý Ở PHẦN TOÁN TỬ

Các toán tử sẽ có thứ tự ưu tiên nhất định, ví dụ như nhân chia trước, cộng trừ sau hoặc cùng mức độ ưu tiên thì thực thi từ trái qua phải

Nhưng dấu đóng mở ngoặc tròn luôn có độ ưu tiên cao nhất, vì thế khi viết biểu thức thì các bạn nên sử dụng dấu ngoặc để biểu thức được thực thi theo đúng mong muốn của mình.