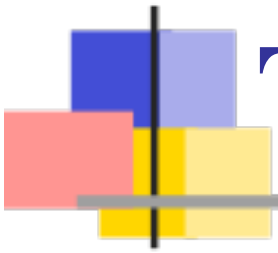
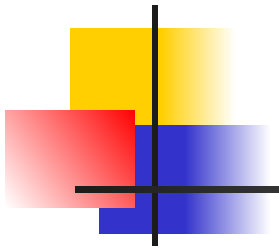


Toán tử và Biểu thức

Chương 3



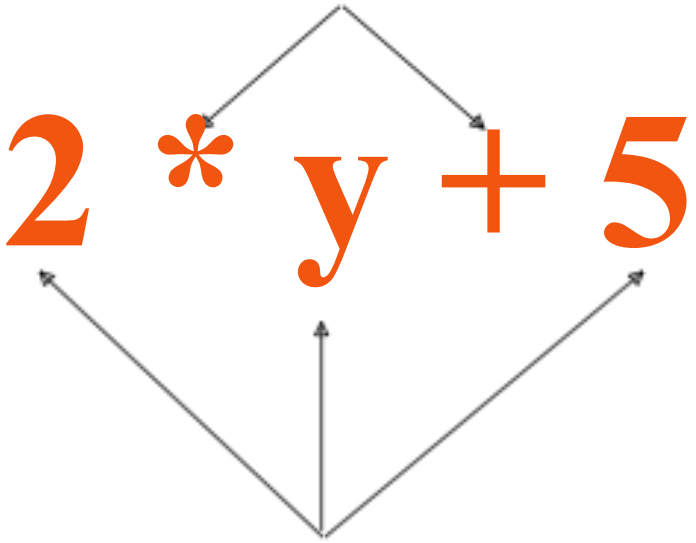
Mục Tiêu

- Hiểu được toán tử gán
- Hiểu được biểu thức số học
- Nắm được toán tử quan hệ và luận lý (Relational and Logical Operators)
- Nắm được toán tử luận lý nhị phân và biểu thức (Bitwise Logical Operators and Expression)
- Hiểu được khái niệm ép kiểu (Cast)
- Hiểu được độ ưu tiên của các toán tử

Biểu thức (Expressions)

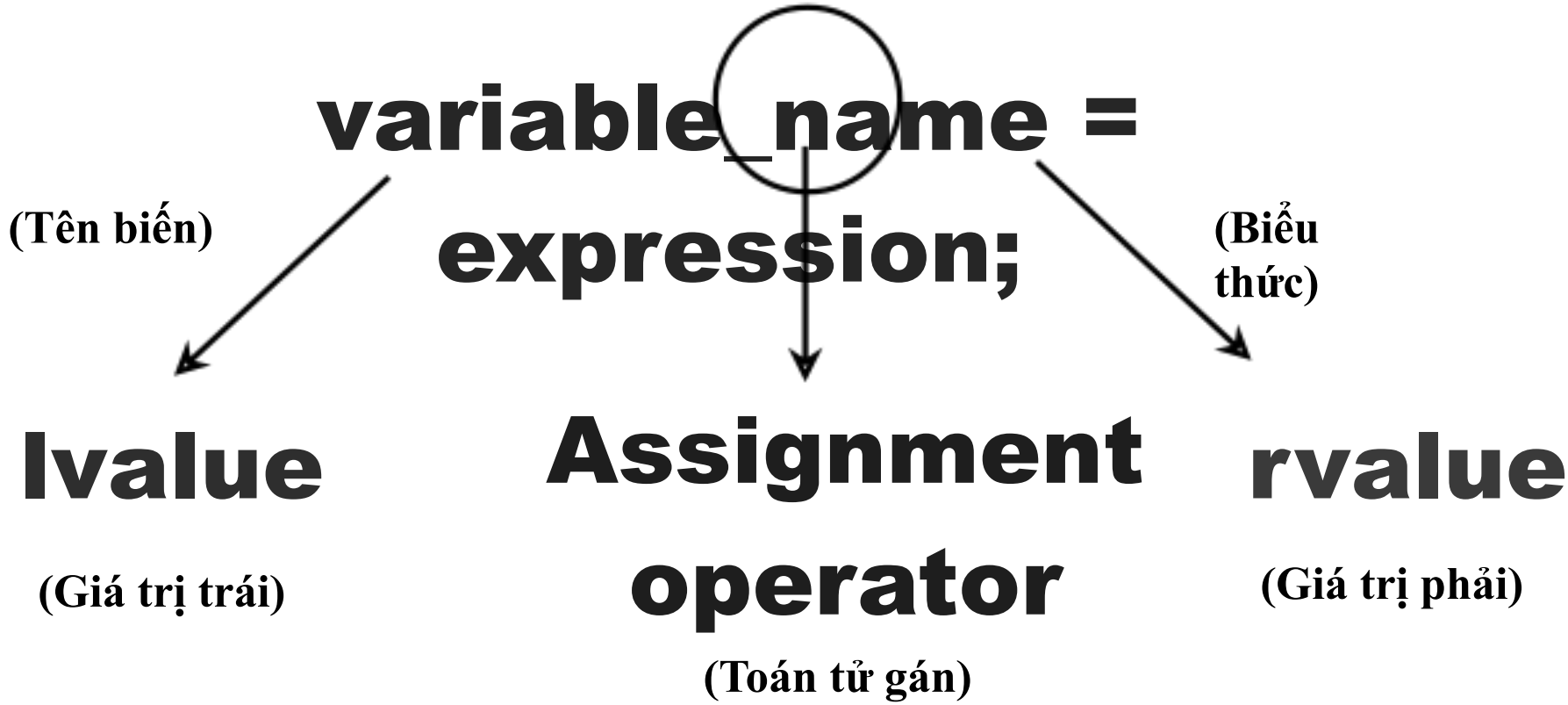
Sự kết hợp các toán tử và các toán hạng

Ví dụ:



Toán tử gán

Toán tử gán (=) có thể được dùng với bất kỳ biểu thức C hợp lệ nào



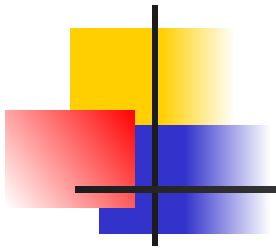
Gán liên tiếp

Nhiều biến có thể được gán với cùng một giá trị trong một câu lệnh đơn

a = b = c = 10; ✓

Tuy nhiên, không thể áp dụng quy tắc trên khi khai báo biến

int a = int b = int b = int c = 10 X



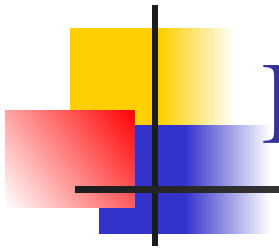
Bốn Kiểu Toán Tử

Số học
(Arithmetic)

Luận Lý
(Logical)

Quan hệ
(Relational)

Nhị phân
(Bitwise)



Biểu thức số học

Biểu thức số học có thể được biểu diễn trong C bằng cách sử dụng các toán tử số học

Ví dụ :

```
++i % 7

5 + (c = 3 + 8)

a * (b + c/d) - 22
```

Toán tử quan hệ và luận lý

Được dùng để :

Kiểm tra mối quan hệ giữa hai biến hay giữa một biến và một hằng

Toán tử quan hệ

Toán tử	Ý nghĩa
>	Lớn hơn
>=	Lớn hơn hoặc bằng
<	Nhỏ hơn
<=	Nhỏ hơn hoặc bằng
==	Bằng
!=	Không bằng

Toán tử quan hệ và luận lý (tt.)

Toán tử luận lý là những ký hiệu dùng để kết hợp hay phủ định biểu thức chứa các toán tử quan hệ

Toán tử	Ý nghĩa
&&	**AND**: Kết quả là True khi cả 2 điều kiện đều đúng
||	**OR** : Kết quả là True khi chỉ một trong hai điều kiện là đúng
!	**NOT**: Tác động trên các giá trị riêng lẻ, chuyển đổi True thành False và ngược lại.

Ví dụ: if (a>10) && (a<20)

Những biểu thức dùng toán tử luận lý trả về 0 thay cho false và 1 thay cho true

Toán tử luận lý nhị phân

Dữ liệu chỉ được xử lý sau khi đã chuyển đổi giá trị SỐ thành giá trị NHỊ PHÂN

Toán tử	**Mô tả**
Bitwise AND (x & y)	Mỗi vị trí của bit trả về kết quả là 1 nếu bit của hai toán hạng là 1.
Bitwise OR (x | y)	Mỗi vị trí của bit trả về kết quả là 1 nếu bit của một trong hai toán hạng là 1.
Bitwise NOT (~ x)	Đảo ngược giá trị của toán hạng (1 thành 0 và ngược lại).
Bitwise XOR (x ^ y)	Mỗi vị trí của bit chỉ trả về kết quả là 1 nếu bit của một trong hai toán hạng là 1 mà không không phải cả hai toán hạng cùng là 1.

Toán tử luận lý nhị phân (tt.)

Ví dụ

- 10 & 15→ **1010 & 1111→1010** → 10
- 10 | 15→ **1010** | **1111→1111** → 15
- 10 ^ 15→ **1010** ^ **1111→0101** → 5
- ~ **10** → ~**1010** →1…11110**101** → **-11**

Chuyển đổi kiểu

Qui tắc chuyển đổi kiểu tự động trình bày dưới đây nhằm xác định giá trị biểu thức:

a. char và short được chuyển thành int và float được chuyển thành double.

b. Nếu có một toán hạng là double, toán hạng còn lại sẽ được chuyển thành double, và kết quả là double.

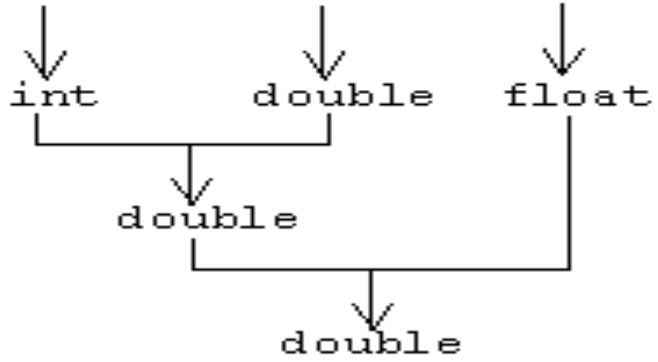
c. Nếu có một toán hạng là long, toán hạng còn lại sẽ được chuyển thành long, và kết quả là long.

d. Nếu có một toán hạng là unsigned, toán hạng còn lại sẽ được chuyển thành unsigned và kết quả cũng là unsigned.

e. Nếu tất cả toán hạng kiểu int, kết quả là int.

Ví dụ

```
char ch;
int i;
float f;
double d;
result = (ch/i) + (f*d) - (f+i);
```



Ép kiểu

Một biểu thức được ép thành một kiểu nhất định bằng cách dùng kỹ thuật ép kiểu (**cast).**

Cú pháp :

(kiểu dữ liệu) cast

Kiểu → Bất cứ kiểu dữ liệu hợp lệ trong C

Ví dụ:

```
float x,f;
f = 3.14159;
x = (int) f;
```

Giá trị của x sẽ là 3 (số nguyên)

Giá trị số nguyên trả về bởi (int) f được chuyển thành số thực khi nó được toán tử GÁN xử lý. Song, giá trị của f vẫn không đổi.

Độ ưu tiên của toán tử

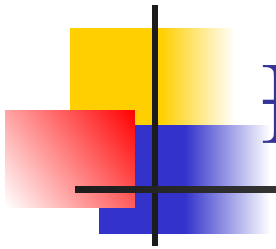
- Độ ưu tiên tạo nên cấu trúc phân cấp của loại toán tử này so với loại toán tử khác khi tính giá trị một biểu thức số học
- Nó đề cập đến thứ tự thực thi các toán tử trong C
- Độ ưu tiên của các toán tử này được thay đổi bởi các dấu ngoặc đơn trong biểu thức

Loại toán tử	**Toán tử**	**Tính kết hợp**
Một ngôi	- ++ --	Phải đến trái
Hai ngôi	^	Trái đến phải
Hai ngôi	* / %	Trái đến phải
Hai ngôi	+ -	Trái đến phải
Hai ngôi	=	Phải đến trái

Độ ưu tiên của toán tử (tt.)

Ví dụ **-8 * 4 % 2 - 3**

Trình tự	Thao tác	Kết quả
1.	- 8 (phép trừ một ngôi)	số âm của 8
2.	- 8 * 4	- 32
3.	- 32 % 2	16
4.	16-3	13



Độ ưu tiên của toán tử so sánh

Độ ưu tiên của toán tử so sánh (quan hệ) luôn được tính từ trái sang phải

Độ ưu tiên của toán tử luận lý

Thứ tự ưu tiên	Toán tử
1	NOT
2	AND
3	OR

Khi có nhiều toán tử luận lý trong một điều kiện, ta áp dụng quy tắc tính từ phải sang trái

Độ ưu tiên của toán tử luận lý (tt.)

Xét biểu thức sau:

False OR True AND NOT False AND True

Điều kiện này được tính như sau:

False OR True AND [NOT False] AND True

NOT có độ ưu tiên cao nhất.

False OR True AND [True AND True]

Ở đây, AND có độ ưu tiên cao nhất, những toán tử có cùng ưu tiên được tính từ phải sang trái.

False OR [True AND True]

[False OR True]

True

Độ ưu tiên giữa các toán tử

Khi một biểu thức có nhiều loại toán tử thì độ ưu tiên giữa chúng phải được thiết lập.

Thứ tự ưu tiên	Kiểu toán tử
1	Số học (Arithmetic)
2	So sánh (Comparison)
3	Luận lý (Logical)

Độ ưu tiên giữa các toán tử (tt.)

Ví dụ :

2*3+4/2 > 3 AND 3<5 OR 10<9

Việc tính toán như sau :

[2*3+4/2] > 3 AND 3<5 OR 10<9

Toán tử số học sẽ được tính trước

[[2*3]+[4/2]] > 3 AND 3<5 OR 10<9

[6+2] >3 AND 3<5 OR 10<9

[8 >3] AND [3<5] OR [10<9]

Độ ưu tiên giữa các toán tử (tt.)

Kế đến là toán tử so sánh có cùng độ ưu tiên. Ta áp dụng quy tắc tính từ trái sang phải.

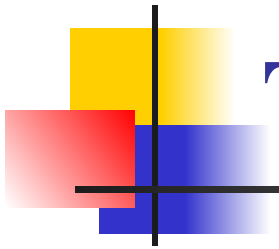
True AND True OR False

Cuối cùng là toán tử kiểu luận lý. AND sẽ có độ ưu tiên cao hơn OR

[True AND True] OR False

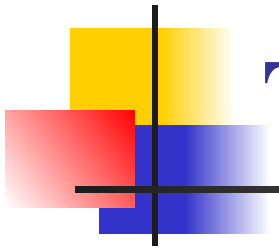
True OR False

True



Thay đổi độ ưu tiên

- Dấu ngoặc đơn ( ) có độ ưu tiên cao nhất
- Độ ưu tiên của các toán tử có thể được thay đổi bởi dấu ngoặc đơn
- Toán tử có độ ưu tiên thấp hơn nếu đặt trong dấu ngoặc đơn sẽ được thực thi trước
- Khi các cặp ngoặc đơn lồng nhau ((( ))), cặp ngoặc đơn trong cùng nhất sẽ được thực thi trước
- Nếu trong biểu thức có nhiều cặp ngoặc đơn thì việc thực thi sẽ theo thứ tự từ trái sang phải



Thay đổi độ ưu tiên (tt.)

Ví dụ :

5+9*3^2-4 > 10 AND (2+2^4-8/4 > 6 OR (2<6 AND 10>11))

Cách tính :

1) **5+9*3^2-4 > 10 AND (2+2^4-8/4 > 6 OR (True AND False))**

Dấu ngoặc đơn bên trong sẽ được tính trước

2) **5+9*3^2-4 > 10 AND (2+2^4-8/4 > 6 OR False)**

Thay đổi độ ưu tiên (tt.)

3) 5+9*3^2-4 >10 AND (2+16-8/4 > 6 OR False)

Kế đến dấu ngoặc đơn ở ngoài được tính đến

1) **5+9*3^2-4 > 10 AND (2+16-2 > 6 OR False)**
2) **5+9*3^2-4 > 10 AND (18-2 > 6 OR False)**
6) 5+9*3^2-4 > 10 AND (16 > 6 OR False)

7) 5+9*3^2-4 > 10 AND (True OR False)

8) 5+9*3^2-4 > 10 AND True

Thay đổi độ ưu tiên (tt.)

1) **5+9*9-4>10 AND True**

Biểu thức bên trái được tính trước

2) **5+81-4>10 AND True**

11) 86-4>10 AND True

12) 82>10 AND True

13) True AND True

14) True