



# IT001 - NHẬP MÔN LẬP TRÌNH

## CHƯƠNG 3.2: CÁC PHÉP TOÁN TRONG C++

Toán tử là các ký hiệu đặc biệt được sử dụng để thực hiện các phép toán trên dữ liệu trong C++. Chúng đóng vai trò quan trọng trong việc thao tác với biến, biểu thức và các cấu trúc dữ liệu khác.

Khoa Khoa học Máy tính



# NỘI DUNG

- 3.1 Cấu trúc chương trình C++
- 3.2 Bộ từ vựng (**keywords**) trong C++
- 3.3 Các kiểu dữ liệu cơ sở (**Fundamental data types**)
- 3.4 Biến (**Variable**)
- 3.5 Hằng (**Constant**)
- 3.6 Các phép toán (**Operators**)
- 3.7 Biểu thức và độ ưu tiên toán tử
- 3.8 Nhập xuất dữ liệu
- Bài tập



## 3.6 Các phép toán (Operators)



## 3.6 Các phép toán

3.6.1 Toán tử gán - Assignment Operators

3.6.2 Toán tử toán học - Arithmetic Operators

3.6.3 Toán tử tăng giảm - Increment Operator and Decrement Operator

3.6.4 Toán tử phẩy - Comma Operator

3.6.5 Toán tử kết hợp - Compound operators

3.6.6 Toán tử bit - Bitwise Operators

3.6.7 Toán tử điều kiện - Conditional ternary Operator

3.6.8 Toán tử quan hệ - Relational Operators

3.6.9 Toán tử luận lý - Logical Operators

3.6.10 Toán tử sizeof - sizeof Operators

3.6.11 Toán tử thay thế (alternative operator)



## 3.6.1 Toán tử gán - Assignment operator

- Dùng để gán giá trị cho 1 biến

```
int x = 10;
```

Call Copy constructor

```
int y = 10;  
int x = y;
```

Call Copy constructor

```
int x, y;  
y = 2 + (x = 5);
```

Call Assignment operator

x = 5;

Call Assignment operator

y = 2 + x;

```
int a, b;  
a = 10;  
b = 4;  
a = b;  
b = 7;
```

Call Assignment operator

a = ?, b = ?

a = 10, b = ?

a = 10, b = 4

a = 4, b = 4

a = 4, b = 7

```
int x, y, z;  
x = y = z = 5;
```

Gán giá trị 5 cho 3 biến z, y, x



## 3.6.2 Toán tử toán học - Arithmetic operators

Phép toán	Giải thích	Ví dụ
+	Cộng	$x = 11 + 3$
-	Trừ	$x = 11 - 3$
*	Nhân	$x = 11 * 3$
/	Chia	$x = 11 / 3.$
/	Lấy phần nguyên	$x = 11 / 3$
%	Lấy phần dư	$x = 11 \% 3$

- Ví dụ 1: Xét đoạn lệnh sau:

```
int x = 4;  
float y = 2.5F;  
double z = x * y;
```

→ Hỏi khi chạy chương trình có phát sinh lỗi hay không?

- Ví dụ 2: Cho biết kết quả của các phép tính sau

```
float a = 5 / 2;  
float b = 5 / 2.;  
float c = 5. / 2;  
float d = (float)5/2;  
float d1 = float(5)/2;  
float d2 = float(5/2);
```



## 3.6.2 Toán tử số học - Arithmetic operators

### Vấn đề:

- Phép nhân (tràn kiểu dữ liệu)
- Phép chia (sai logic do sử dụng phép lấy phần nguyên)

### Hướng giải quyết:

Sử dụng kỹ thuật ép kiểu

- Ví dụ:

```
#include <iostream>
int main(){
    int a = 123456;
    int b = 654321;
    std::cout << a+b << "\n";
    std::cout << a-b << "\n";
    std::cout << a*b << "\n";
    std::cout << a/b << "\n";
    return 0;
}
```

Kết quả in ra:

777777  
-530865  
-824525248  
0

Kết quả không như mong muốn

80779853376  
0.188678

```
std::cout << (long long)a*b << "\n";
std::cout << (float)a/b << "\n";
```



# Ví dụ:

- Ví dụ: **Hoàn chỉnh đoạn chương trình sau.** Biết rằng chương trình đầu vào nhận 2 số nguyên ( $\leq$  2triệu), đầu ra là kết quả của phép +, -, \* và / (lấy kết quả chính xác).

```
#include <iostream>
int main(){
    int a , b;
    cin >> a >> b;
    std::cout << a+b << "\n";
    std::cout << a-b << "\n";
    std::cout << a*b << "\n";
    std::cout << a/b << "\n";
    return 0;
}
```



### 3.6.3 Toán tử tăng ++, giảm -- (increment and decrement)

- Toán tử tăng (++) **tăng 1 đơn vị** cho toán hạng của nó, toán tử giảm (--) **trừ 1 đơn vị** cho toán hạng của nó.
- Các toán tử tăng, giảm **chỉ áp dụng cho toán hạng là các biến dạng số hoặc con trỏ**.
- Ví dụ:
  - $++x$ ; tương đương  $x += 1$ ; tương đương  $x = x + 1$ ;
  - Biểu thức  $(i+j)++ \rightarrow$  **không hợp lệ**
- Giả sử gọi x là biến (toán hạng) sử dụng toán tử ++, --, ta có các dạng sau:
  - $++x$  : Đây được gọi là Prefix increment. Trong biểu thức chứa  $++x$ : x sẽ **tăng lên 1 đơn vị trước tiên**, sau đó giá trị cập nhật sẽ được sử dụng.
  - $x++$  : Đây được gọi là Postfix increment. Trong biểu thức chứa  $x++$ : biến x sử dụng giá trị hiện tại của x (chưa tăng lên 1 đơn vị), tính xong biểu thức rồi **cuối cùng mới tăng x lên 1 đơn vị**.



### 3.6.3 Toán tử tăng ++, giảm - (increment and decrement)

- $--x$  : Đây được gọi là Prefix decrement. Trong biểu thức chứa  $--x$ : biến  $x$  sẽ **giảm xuống 1 đơn vị trước**, sau đó giá trị cập nhật sẽ được sử dụng.
- $x--$  : Đây được gọi là Postfix decrement. Trong biểu thức chứa  $x--$ : biến  $x$  sử dụng giá trị hiện tại của  $x$  (chưa giảm 1 đơn vị), tính xong biểu thức rồi **cuối cùng mới giảm  $x$  1 đơn vị**.
- Ví dụ:

```
int x = 5;  
int y = ++x;
```

```
1. ++x      → x = 6  
2. y = x    → y = 6
```

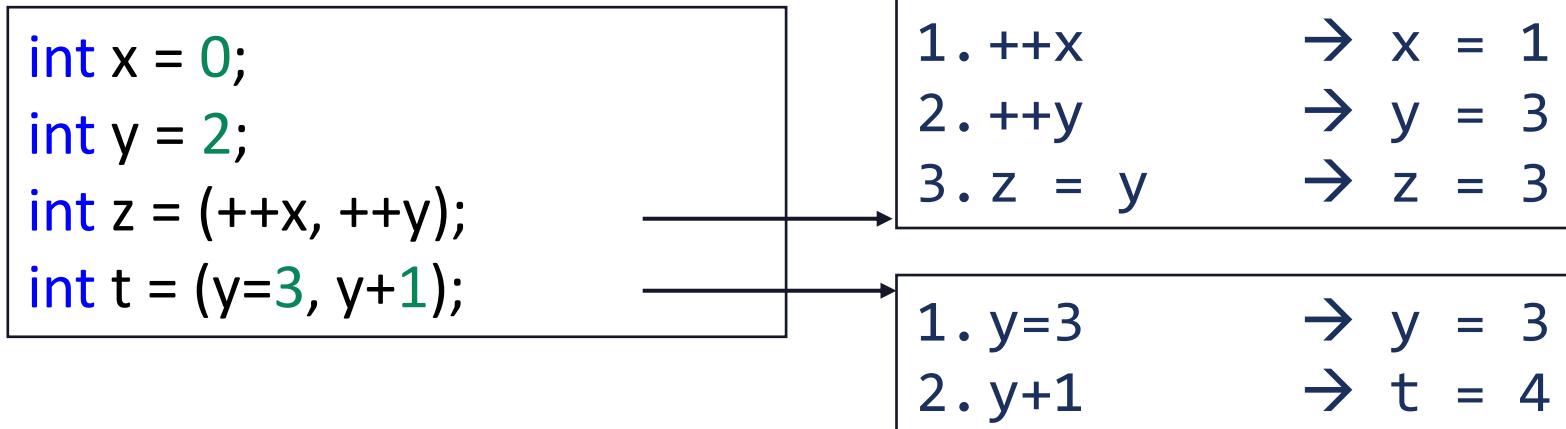
```
int x = 5;  
int y = x++;
```

```
1. y = x    → x = 5, y = 5  
2. x++      → y = 5, x = 6
```



## 3.6.4 Toán tử phẩy - Comma operator

- Các biểu thức đặt cách nhau bằng dấu ` , `
- Các biểu thức con lần lượt được tính từ trái sang phải
- Biểu thức mới nhận được là giá trị của biểu thức bên phải cùng
- Ví dụ:





## 3.6.5 Toán tử kết hợp - Compound operators

Toán tử	Ví dụ	Giải thích	Phép toán
<code>+=</code>	<code>x += 5</code>	$x = x + 5$	Cộng
<code>-=</code>	<code>x -= 5</code>	$x = x - 5$	Trừ
<code>*=</code>	<code>x *= 5</code>	$x = x * 5$	Nhân
<code>/=</code>	<code>x /= 5</code>	$x = x / 5$	Chia hay lấy phần nguyên
<code>%=</code>	<code>x %= 5</code>	$x = x \% 5$	Lấy phần dư
<code>&lt;=&gt;</code>	<code>x &lt;=&gt; 5</code>	$x = x <> 5$	Dịch trái
<code>&gt;=&gt;</code>	<code>x &gt;=&gt; 5</code>	$x = x >> 5$	Dịch phải
<code>&amp;=</code>	<code>x &amp;= 5</code>	$x = x \& 5$	AND
<code>^=</code>	<code>x ^= 5</code>	$x = x ^ 5$	XOR
<code> =</code>	<code>x  = 5</code>	$x = x   5$	OR

- Ví dụ: `price *= units + 1`; **tương đương** `price = price * (units + 1)`;



## 3.6.6 Toán tử bit - Bitwise operators

- Toán tử bit (bitwise operators) trong C++ là các toán tử thao tác trực tiếp trên các bit của toán hạng (nguyên).
- Gồm các toán tử: & (and), | (or), ^ (xor), ~ (not hay lấy số bù 1), >> (shift bits right: dịch phải), << (shift bits left: dịch trái)
- Toán tử gộp: &=, |=, ^=, ~=, >>=, <<=
- Bảng chân trị:

p	q	p & q (AND)	p ^ q (XOR)	p   q (OR)	~p (NOT)
0	0	0	0	0	1
0	1	0	1	1	1
1	1	1	0	1	0
1	0	0	1	1	0



## 3.6.6 Toán tử bit - Bitwise operators

- Ví dụ:
  - `5 & 3` // Kết quả: 1 (`0101 & 0011 = 0001`)
  - `5 | 3` // Kết quả: 7 (`0101 | 0011 = 0111`)
  - `5 ^ 3` // Kết quả: 6 (`0101 ^ 0011 = 0110`)
  - `~5` // Kết quả: -6 (bit đảo ngược của `0101` là `1010`, tương đương với `-6`)
  - `12 >> 3 = 1100 >> 3 = 0001 = 1` (toán tử dịch phải)
  - `3 << 2 = 0011 << 2 = 1100 = 12` (toán tử dịch trái)



## 3.6.6 Toán tử bit - Bitwise operators

- Một số ứng dụng của toán tử bit trong lập trình:

### 1. Kiểm tra chia hết cho 2

```
int number = 5;  
if(number & 1 == true)  
    std::cout << "So le\n";  
else  
    std::cout << "So chan\n";
```

$$\begin{aligned} a \& 1 &\rightarrow a \% 2 \\ a \& 3 &\rightarrow a \% 4 \\ a \& 7 &\rightarrow a \% 8 \end{aligned}$$

### 2. Tích và thương cho $2^n$

```
int a = 2 << 1;  
int b = 2 << 2;  
int c = 8 >> 1;  
int d = 8 >> 2;
```

$$\begin{aligned} a &= 2 * 2^1 &\rightarrow a &= 4 \\ b &= 2 * 2^2 &\rightarrow b &= 8 \\ c &= 8 / 2^1 &\rightarrow c &= 4 \\ d &= 8 / 2^2 &\rightarrow d &= 2 \end{aligned}$$



# Ví dụ: Chương trình in số nhị phân

```
//: C03:Bitwise.cpp
//{L} printBinary
//Demonstration of bit manipulation

#include <iostream>
using namespace std;
// A macro to save typing:
#define PR(STR, EXPR) \
cout << STR; printBinary(EXPR); cout << endl;

void printBinary(const unsigned char val) {
    for(int i = 7; i >= 0; i--)
        if(val & (1 << i))
            std::cout << "1";
        else
            std::cout << "0";
} //:{~
```

```
int main() {
    unsigned int getval;
    unsigned char a, b;
    cout << "Enter a number between 0 and 255: ";
    cin >> getval;
    a = getval;
    PR("a in binary: ", a);
    cout << "Enter a number between 0 and 255: ";
    cin >> getval;
    b = getval;
    PR("b in binary: ", b);
    PR("a | b = ", a | b);
    PR("a & b = ", a & b);
```

```
PR("a ^ b = ", a ^ b);
PR("~a = ", ~a);
PR("~b = ", ~b);
// An interesting bit pattern:
unsigned char c = 0x5A;
PR("c in binary: ", c);
a |= c;
PR("a |= c; a = ", a);
b &= c;
PR("b &= c; b = ", b);
b ^= a;
PR("b ^= a; b = ", b);
} //:{~
```



## 3.6.7 Toán tử điều kiện - Conditional Ternary Operator

- Cú pháp:

<condition> ? <Statement 1> : < Statement 2>



- Ví dụ 1: Tìm số lớn nhất giữa 2 số a và b.

```
int a = 1;  
int b = 2;  
int c = (a>b) ? a : b;
```

- Ví dụ 2: Số sinh viên lớn hơn 50 sinh viên thì số lớp bằng 2, ngược lại là 1.

```
int so_sinh_vien = 55;  
const int so_lop = (so_sinh_vien>50)? 2:1;
```



## 3.6.8 Toán tử quan hệ - Relational operators

Operator	Toán tử	Ký hiệu	Ví dụ	Giải thích
Greater than	Lớn hơn	>	$x > y$	Nếu $x$ lớn hơn $y \rightarrow \text{true} (1)$ Ngược lại $\rightarrow \text{false} (0)$
Less than	Nhỏ hơn	<	$x < y$	Nếu $x$ nhỏ hơn $y \rightarrow \text{true} (1)$ Ngược lại $\rightarrow \text{false} (0)$
Greater than Or: Equal to	Lớn hơn Hoặc: Bằng	$\geq$	$x \geq y$	Nếu $x$ lớn hơn hoặc bằng $y \rightarrow \text{true} (1)$ Ngược lại $\rightarrow \text{false} (0)$
Less than Or: Equal to	Nhỏ hơn Hoặc: Bằng	$\leq$	$x \leq y$	Nếu $x$ nhỏ hơn hoặc bằng $y \rightarrow \text{true} (1)$ Ngược lại $\rightarrow \text{false} (0)$
Equal to	Bằng	$\equiv$	$x \equiv y$	Nếu $x$ bằng $y \rightarrow \text{true} (1)$ Ngược lại $\rightarrow \text{false} (0)$
Not equal to	Khác	$\neq$	$x \neq y$	Nếu $x$ khác $y \rightarrow \text{true} (1)$ Ngược lại $\rightarrow \text{false} (0)$



## 3.6.8 Toán tử quan hệ

- **Ví dụ 1:**

`(7 == 5) // evaluates to false`

`(5 > 4) // evaluates to true`

`(3 != 2) // evaluates to true`

`(6 >= 6) // evaluates to true`

`(5 < 5) // evaluates to false`

- **Ví dụ 2:**

`a=2; b=3; c=6;`

`(a == 5) // evaluates to false, since a is not equal to 5`

`(a*b >= c) // evaluates to true, since (2*3 >= 6) is true`

`(b+4 > a*c) // evaluates to false, since (3+4 > 2*6) is false`

`((b=2) == a) // evaluates to true`



## 3.6.9 Toán tử luận lý - Logical operators

Toán tử	Ký hiệu	Ví dụ
NOT	!	$!x$
AND	$\&\&$	$x \&\& y$
OR	$\ $	$x \  y$

Bảng chân trị			
1	2	$\ $	$\&\&$
false	false	false	false
false	true	true	false
true	false	true	false
true	true	true	true

- **Ví dụ:**
  1.  $(\text{true} \&\& \text{true}) \| \text{false}$
  2.  $(\text{false} \&\& \text{true}) \| \text{true}$
  3.  $(\text{false} \&\& \text{true}) \| \text{false} \| \text{true}$
  4.  $(5 > 6 \| 4 > 3) \&\& (7 > 8)$
  5.  $!(7 > 6 \| 3 > 4)$
  6.  $!\text{true}$

- **Kết quả:**
  1. true
  2. true
  3. true
  4. false
  5. false
  6. false



## 3.6.9 Toán tử luận lý - Logical operators

- **Short-circuit evaluation** (đánh giá ngắn mạch) là một kỹ thuật tối ưu hóa trong lập trình mà ở đó các biểu thức logic được đánh giá từ trái sang phải và sẽ ngừng đánh giá khi kết quả của biểu thức đã được xác định. Trong C++, kỹ thuật này được áp dụng cho các toán tử logic **&&** (AND) và **||** (OR).

Operator	Short-circuit
<b>&amp;&amp;</b>	Đánh giá từ trái sang phải. Nếu vé trái là FALSE, toàn bộ biểu thức là FALSE và vé phải không được đánh giá.
<b>  </b>	Đánh giá từ trái sang phải. Nếu vé trái là TRUE, toàn bộ biểu thức là TRUE và vé phải không được đánh giá.

- **Ví dụ:** Giải thích cách hoạt động của các biểu thức bên dưới:
  - `A = ( (5 == 5) && (3 > 6) ) // evaluates to false ( true && false )`
  - `B = ( (5 == 5) || (3 > 6) ) // evaluates to true ( true || false )`
  - `y = 9; x = 10 || --y;`



### 3.6.9 Toán tử luận lý - Logical operators

- Điều này chủ yếu quan trọng khi biểu thức bên phải có tác dụng phụ, chẳng hạn như thay đổi giá trị. Ví dụ:

```
if ( (i<10) && (++i<n) ) {
```

```
/*...*/
```

```
}
```

- Ở đây, biểu thức điều kiện kết hợp sẽ tăng i lên một, nhưng chỉ khi điều kiện ở bên trái && là đúng, vì nếu không thì điều kiện ở về phải (++i<n) sẽ không bao giờ được đánh giá.



## 3.6.10 Toán tử sizeof

- Toán tử **sizeof** chấp nhận một tham số, có thể là một kiểu hoặc một biến và trả về kích thước tính bằng byte của kiểu hoặc đối tượng đó:

```
x = sizeof(char);
```

- Ở đây, x được gán giá trị 1, vì char là kiểu có kích thước một byte.
- Giá trị được trả về bởi sizeof là hằng số thời gian biên dịch, do đó nó luôn được xác định trước khi thực hiện chương trình.



## 3.6.10 Toán tử sizeof

```
#include <iostream>

int main() {
    char a; int b;
    short c; double d;

    std::cout << sizeof(a) << " " << sizeof(char) << " bytes\n";
    std::cout << sizeof(b) << " " << sizeof(int) << " bytes\n";
    std::cout << sizeof(c) << " " << sizeof(short) << " bytes\n";
    std::cout << sizeof(d) << " " << sizeof(double) << " bytes\n";

    return 0;
}
```



## 3.6.11 Toán tử thay thế (alternative operator)

Primary	Alternative
&&	and
&=	and_eq
&	bitand
	bitor
~	compl
!	not
!=	not_eq
	or
=	or_eq
^	xor
^=	xor_eq

Primary	Digraph
{	<%
}	%>
[	<:
]	::>
#	%:
##	%::%

- Ví dụ:  
 $(a \text{ and not } b)$   
tương đương:  $(a \&& !b)$
- $(a \text{ not\_eq } b)$   
tương đương:  $(a \neq b)$



## 3.6.11 Toán tử thay thế (alternative operator)

- Ví dụ:

```
#include<iostream>

#include<bitset>

int main(){

    std::bitset<4> mask("1100");

    std::bitset<4> val("0111");

    val &= mask;

    std::cout << val << '\n';

}
```

- Tương đương:

```
%:include<iostream>

%:include<bitset>

int main()%>

    std::bitset<4> mask("1100");

    std::bitset<4> val("0111");

    val and_eq mask;

    std::cout << val << '\n';

%
```



## 3.7 Biểu thức và Độ ưu tiên toán tử



## 3.7 Biểu thức và Độ ưu tiên toán tử

### 3.7.1 Biểu thức

### 3.7.2 Độ ưu tiên toán tử



## 3.7.1 Biểu thức

- **Định nghĩa:** Biểu thức được tạo thành được từ các **toán hạng** (Operand) (hằng số, biến số và hàm số) liên kết với nhau bằng các toán tử (Operator).
- Ví dụ:

```
int a = 2 + 3;
```

```
int year = 2000;
```

```
int b = a / 5;
```

```
int month = 29;
```

```
int c = (a + b) * 5;
```

```
if((year%4!=0) || (year%400!=0))
```

```
int d = (x >= 3);
```

```
return 28;
```



# Ví dụ: Biểu thức

1. `x = 3 + 4 * 5 - 9;`
2. `x = y = z = 9;`
3. `int a=5, x = ++a + 10;`
4. `z = 2; y=2; z *= ++y + 5;`
5. `int i = 5 > 4 > 2 < 3;`
6. `x = 10 + 5 > 2;`
7. `x = 5 & 3 == 3;`
8. `x = (3 & 5 << 2) || (8 >> 2 ^ 3);`

9. `x = (5 > 3 && 4 < 8)`
10. `y = 9; x = 10 || --y;`
11. `int a = 0, b = 0, c = ++a || b++;`
12. `bool a=true, b=false; cout << a && b;`
13. `result = 1 || 0 && 1;`
14. `true || true && false || false;`
15. `x = (true || false) ? 4 : 5`
16. `x = 5 > 8 ? 10 : 1 != 2 < 5 ? 20 : 30;`



## 3.7.1 Biểu thức

- Minh họa triển khai viết biểu thức cho các mệnh đề:

- x lớn hơn hay bằng 3:

$x \geq 3$

- a và b cùng dấu:

$((a > 0) \text{ && } (b > 0)) \text{ || } ((a < 0) \text{ && } (b < 0))$

$(a > 0 \text{ && } b > 0) \text{ || } (a < 0 \text{ && } b < 0)$

Hỏi:  $a > 0 \text{ && } b > 0 \text{ || } a < 0 \text{ && } b < 0 ?$

- p bằng q bằng r:

$(p == q) \text{ && } (q == r) \text{ hoặc } (p == q \text{ && } q == r)$

- $-5 < x < 5$ :

$((x > -5) \text{ && } (x < 5)) \text{ hoặc } (x > -5 \text{ && } x < 5)$



## 3.7.2 Độ ưu tiên toán tử

Precedence	Operator	Description	Associativity
1	::	Scope resolution	Left-to-right
2	++ --	Suffix/postfix increment and decrement	
	type() type{ }	Function-style type cast	
	()	Function call	
	[]	Array subscripting	
	.	Element selection by reference	
	->	Element selection through pointer	
3	++ --	Prefix increment and decrement	Right-to-left
	+ -	Unary plus and minus	
	! ~	Logical NOT and bitwise NOT	
	(type)	C-style type cast	
	*	Indirection (dereference)	
	&	Address-of	
	sizeof	Size-of ( <b>note</b> )	
	new, new[]	Dynamic memory allocation	
	delete, delete[]	Dynamic memory deallocation	
4	.* ->*	Pointer to member	Left-to-right
5	* / %	Multiplication, division, and remainder	

**(note):** The operand of sizeof can't be a C-style type cast: the expression **sizeof (int) \* p** is unambiguously interpreted as **(sizeof(int)) \* p**, but not **sizeof((int)\*p)**.



## 3.7.2 Độ ưu tiên toán tử

Precedence	Operator	Description	Associativity
6	+ -	Addition and subtraction	Left-to-right
7	<< >>	Bitwise left shift and right shift	
8	< <=	For relational operators < and $\leq$ respectively	
	> >=	For relational operators > and $\geq$ respectively	
9	== !=	For relational = and $\neq$ respectively	
10	&	Bitwise AND	
11	^	Bitwise XOR (exclusive or)	
12		Bitwise OR (inclusive or)	
13	&&	Logical AND	
14		Logical OR	
15	?:	Ternary conditional ( <b>note</b> )	Right-to-left
	=	Direct assignment (provided by default for C++ classes)	
	+= -=	Assignment by sum and difference	
	*= /= %=	Assignment by product, quotient, and remainder	
	<<= >>=	Assignment by bitwise left shift and right shift	
	&= ^=  =	Assignment by bitwise AND, XOR, and OR	
16	throw	Throw operator (for exceptions)	Left-to-right
17	,	Comma	

**(note):** The expression in the middle of the conditional operator (between ? and :) is parsed as if parenthesized: its precedence relative to ?: is ignored.



## 3.7.2 Độ ưu tiên toán tử

- Một biểu thức có nhiều phép toán, thì quy tắc thực hiện như sau:
  - Thực hiện biểu thức trong ( ) sâu nhất trước.
  - Thực hiện theo **thứ tự ưu tiên các toán tử**.
  - Khi một biểu thức có hai toán tử có cùng mức độ ưu tiên: thực hiện phép toán từ **trái sang phải** hoặc từ **phải sang trái** sẽ theo quy ước chung của nhóm phép toán đó.
  - Việc đặt tất cả các câu lệnh phụ trong dấu ngoặc đơn (ngay cả những câu lệnh không cần thiết vì mức độ ưu tiên của chúng) sẽ cải thiện khả năng đọc mã.



# Ví dụ 1:

- **Tính biểu thức:**

```
int a = 5, b = 10, c = 15;
```

```
bool result = (a + b * c > b) && (c / a == b % c) || (a - b < c);
```

- **Ta có:**

- $(a + b * c > b) = ((a + (b * c)) > b) = ((5+(10*15)) > 10) = \text{true}$
- $(c / a == b \% c) = ((c / a) == (b \% c)) = (15 / 5 == 10 \% 15) = \text{false}$
- $(a - b < c) = ((a - b) < c) = (5 - 10 < 15) = \text{true}$

→  $\text{result} = \text{true} \&\& \text{false} \mid\mid \text{true}$  ( $\&\&$  ưu tiên hơn  $\mid\mid$ )

→  $\text{result} = (\text{true} \&\& \text{false}) \mid\mid \text{true}$

→  $\text{result} = \text{true}$



## Ví dụ 2:

- **Tính biểu thức:**

$X = 5 > 8 ? 10 : 1 != 2 < 5 ? 20 : 30;$

- **Ta có:**

$X = (5 > 8) ? 10 : (1 != 2 < 5 ? 20 : 30)$

→  $X = \text{false} ? 10 : (1 != 2 < 5 ? 20 : 30)$

→  $X = (1 != (2 < 5)) ? 20 : 30$     (< ưu tiên hơn !=)

→  $X = (1 != \text{true}) ? 20 : 30$

→  $X = \text{false} ? 20 : 30$

→  $X = 30$



## Ví dụ 3:

- **Tính biểu thức:**

`X = true || true && false || false`

- **Ta có:**

`&&` ưu tiên hơn `||`

- ➔ `X = true || (true && false) || false`
- ➔ `X = true || false || false` ➔ Theo thứ tự trái sang phải
- ➔ `X = (true || false) || false`
- ➔ `X = true || false`
- ➔ `X = true`



# Ví dụ: Biểu thức

- Cho biết giá trị của các biểu thức sau:

- $x = 3 + 4 * 5 - 9;$
- $x = y = z = 9;$
- `int a=5, x = ++a + 10;`
- $z = 2; y=2; z *= ++y + 5;$
- `int i = 5 > 4 > 2 < 3;`
- $x = 10 + 5 > 2;$
- $x = 5 \& 3 == 3;$

- Gợi ý:

- $*/$  trước,  $+-$  sau, (trái sang phải)
- $(x = (y = (z = 9)))$
- Thứ tự ưu tiên:  $++$ ;  $+$
- Thứ tự ưu tiên:  $++$ ;  $+$ ;  $*=$
- $>$  và  $<$  cùng độ ưu tiên => Trái sang phải
- Thứ tự ưu tiên:  $+$ ;  $>$
- Thứ tự ưu tiên:  $==$ ;  $\&$



## 3.7.2 Độ ưu tiên toán tử

- Cho biết giá trị của các biểu thức sau:

9. `x = (5 > 3 && 4 < 8)`

10. `int a = 0, b = 0, c = ++a || b++;`

11. `bool a=true, b=false; cout << a && b;`

12. `result = 1 || 0 && 1;`

13. `true || true && false || false;`

14. `x = (3 & 5 << 2) || (8 >> 2 ^ 3);`

15. `a = b++, c++, d++, e++;`



# Chúc các em học tốt !

