NHẬP MÔN LẬP TRÌNH

CHƯƠNG 3.2: CÁC PHÉP TOÁN OPERATORS

ThS. Nguyễn Thị Ngọc Diễm
diemntn@uit.edu.vn

double vs long double???

#define vs const

double vs long double???

- Phụ thuộc vào trình biên dịch và hệ điều hành.
- Kiểu double là 8 bytes.
- Thông thường hệ điều hành x86 kiểu long double là 8 bytes, một số hệ điều hành x64 là 16 bytes.

#define vs const

- Sử dụng câu lệnh #define thì khi biên dịch tên hằng sẽ được thay thế bằng giá trị. → Ko cần xài bộ nhớ để lưu hằng
- const là một biến hằng → Chiếm dung lượng trên bộ nhớ

Nội dung

5. Các phép toán
6. Biểu thức
7. Câu lệnh
8. Một số hàm hữu ích
9. Một số ví dụ minh họa

5. Các phép toán

1. Toán tử gán – Assignment Operators
2. Toán tử toán học – Arithmetic Operators
3. Toán tử tăng giảm – Increment Operator and Decrement Operator
4. Toán tử phẩy – Comma Operator
5. Toán tử toán học và gán -
6. Toán tử bit – Bitwise Operators
7. Toán tử điều kiện - Conditional ternary Operator
8. Toán tử quan hệ - Relational Operators
9. Toán tử luận lý – Logical Operators
10. Toán tử sizeof – sizeof Operators
11. Độ ưu tiên các toán tử

5.1. Toán tử gán - Assignment operator

Dùng để gán giá trị cho 1 biến

```
int x = 10;
```

Calls **Copy constructor**

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

Calls **Copy constructor**

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

a = ?, b = ?
a = 10, b = ?
a = 10, b = 4
a = 4, b = 4
a = 4, b = 7

Calls **Assignment operator**

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

x = 5; y = 2 + x; Calls **Assignment operator**

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

Gán giá trị 5 cho 3 biến z, y, x Calls **Assignment operator**

5.2. 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

???

Phép /

Khi nào là phép chia?

→ Khi 1 trong các đối số là số thực

Khi nào là phép lấy phần nguyên?

→ Khi các đối số đều là số nguyên

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

a, b, c, d, e = ???

5.2. Toán tử số học - Arithmetic operators

```
#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";
    std::cin.get();
    return 0;
}
```

	Kết quả đúng
777777 -530865	
-824525248 0	80779853376 0.188678

Kết quả không như mong muốn

Vấn đề:

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

Hướng giải quyết:

Sử dụng kỹ thuật ép kiểu

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

5.3. Toán tử tăng ++, giảm – (Increment and decrement)

Dùng để tăng ++ hoặc giảm -- 1 đơn vị:

Ví dụ:
Để tăng giá trị của biến a lên 1 đơn vị ta có thể dùng các câu lệnh sau:

```
int a;
a = a + 1;
a += 1;
a++;
```

Sự khác biệt giữa ++x và x++ ???

```
int x = 5;
int y = ++x;
// x = 6, y = 6
```

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

```
int x = 5;
int y = x++;
//x = 6, y = 5
```

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

5.4. 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:

```
int x = 0;
int y = 2;
int z = (++x, ++y);
int t = (y=3, y+1);
```

```
1. ++x          → x = 1
2. ++y          → y = 3
3. z = y        → z = 3
```

```
1. y=3          → y = 3
2. y+1          → t = 4
```

5.5. Toán tử kết hợp - Compound operators

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

Ví dụ: price *= units + 1; ***equivalent to*** price = price * (units + 1);

5.6. Toán tử bit - Bitwise operators

- Các toán tử trên bit
- Tác động lên các bit của toán hạng (nguyên).
- & (and), | (or), ^ (xor), ~ (not hay lấy số bù 1)
- >> (shift bits right), << (shift bits left)
- Toán tử gộp: &=, |=, ^=, ~=, >>=, <<=

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

Toán tử dịch bit sang trái

3 = 0011
3 << 1 = 0110 = 6
3 << 2 = 1100 = 12
3 << 3 = 1000 = 8

Toán tử dịch bit sang phải

12 = 1100
12 >> 1 = 0110 = 6
12 >> 2 = 0011 = 3
12 >> 3 = 0001 = 1

5.6. Toán tử bit

- Ví dụ toán tử trên bit:

```
int main(){
    int a = 5;// 0000 0000 0000 0101
    int b = 6;// 0000 0000 0000 0110
    int z1, z2, z3, z4, z5, z6;
    z1 = a & b;// 0000 0000 0000 0100
    z2 = a | b;// 0000 0000 0000 0111
    z3 = a ^ b;// 0000 0000 0000 0011
    z4 = ~a;// 1111 1111 1111 1010
    z5 = a >> 2;// 0000 0000 0000 0001
    z6 = a << 2;// 0000 0000 0001 0100
    return 0;
}
```

Ứng dụng của toán tử bit:

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";
```

```
a & 1   → a % 2
a & 3   → a % 4
a & 7   → a % 8
```

2. Tích và thương cho 2^n

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

```
a = 2*2^1       → a = 4
b = 2*2^2       → b = 8
c = 8/2^1       → c = 4
d = 8/2^2       → d = 2
```

5.7. Toán tử điều kiện - Conditional Ternary Operator

Cú pháp: <điều_kiện> ? <biểu_thức_1>:<biểu_thức_2>

Ý nghĩa: Nếu <điều_kiện>
đúng thì thực hiện <biểu_thức_1>
ngược lại thì thực hiện <biểu_thức_2>

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

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

Ứng dụng: Dùng để định nghĩa khi câu lệnh if không thể sử dụng được.
Ví dụ: sử dụng khi định nghĩa **hằng**

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;
```

5.8. 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 → true (1) Ngược lại → false (0)
Less than	Nhỏ hơn	<	x < y	Nếu x nhỏ hơn y → true (1) Ngược lại → false (0)
Greater than or equal to	Lớn hơn hoặc bằng	>=	x >= y	Nếu x lớn hơn hoặc bằng y → true (1) Ngược lại → false (0)
Less than or equal to	Nhỏ hơn hoặc bằng	<=	x <= y	Nếu x nhỏ hơn hoặc bằng y → true (1) Ngược lại → false (0)
Equal to	Bằng	==	x == y	Nếu x bằng y → true (1) Ngược lại → false (0)
Not equal to	Khác	!=	x != y	Nếu x khác y → true (1) Ngược lại → false (0)

5.8. Toán tử quan hệ - Relational and comparison operators

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
```

5.9. Toán tử luận lý - Logical operators

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

Bài tập:

1. (true && true) || false
2. (false && true) || true
3. (false && true) || false || true
4. (5 > 6 || 4 > 3) && (7 > 8)
5. !(7 > 6 || 3 > 4)
6. !true

1	2	||	&&
false	false	false	false
false	true	true	false
true	false	true	false
true	true	true	true

1. true
2. true
3. true
4. false
5. False
6. false

5.9. Toán tử luận lý - Logical operators

Ví dụ:

```
( (5 == 5) && (3 > 6) )  // evaluates to false ( true && false )

((5 == 5) || (3 > 6) )    // evaluates to true ( true || false )
```

- When using the logical operators, C++ only evaluates what is necessary from left to right to come up with the combined relational result, ignoring the rest. Therefore, in the last example ((5==5)||(3>6)), C++ evaluates first whether 5==5 is true, and if so, it never checks whether 3>6 is true or not. This is known as short-circuit evaluation, and works like this for these operators:

operator	short-circuit
&&	if the left-hand side expression is false, the combined result is false (the right-hand side expression is never evaluated).
||	if the left-hand side expression is true, the combined result is true (the right-hand side expression is never evaluated).

5.9. Toán tử luận lý - Logical operators

- This is mostly important when the right-hand expression has side effects, such as altering values:

Ví dụ: `if ( (i<10) && (++i<n) ) { /*...*/ } // note that the condition increments`

- Here, the combined conditional expression would increase i by one, but only if the condition on the left of && is true, because otherwise, the condition on the right-hand side (++i<n) is never evaluated.

5.10 Toán tử sizeof

- This operator accepts one parameter, which can be either a type or a variable, and returns the size in bytes of that type or object:

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

- Here, x is assigned the value 1, because char is a type with a size of one byte.
- The value returned by sizeof is a compile-time constant, so it is always determined before program execution.

5.11 Độ ưu tiên toán tử - Precedence of operators

Mức độ	Toán tử	Nhóm ưu tiên
1	::	Trái sang phải
2	++ --	Trái sang phải
	()	
	[]	
	. ->	
3	++ --	Phải sang trái
	~ !	
	+ -	
	& *	
	new delete	
	sizeof	
	(type)	
4	.* ->*	Trái sang phải
5	* / %	Trái sang phải
6	+ -	Trái sang phải
7	<< >>	Trái sang phải
8	< > <= >=	Trái sang phải
9	== !=	Trái sang phải
10	&	Trái sang phải
11	^	Trái sang phải
12	|	Trái sang phải
13	&&	Trái sang phải
14	||	Trái sang phải
15	= *= /= %= += -=	Phải sang trái
	>>= <<= &= ^= |=	
	?:	
16	,	Trái sang phải

5.11 Độ ưu tiên toán tử - Precedence of operators

- When an expression has two operators with the same precedence level, *grouping* determines which one is evaluated first: either left-to-right or right-to-left.

  Enclosing all sub-statements in parentheses (even those unnecessary because of their precedence) improves code readability.

- Quy tắc thực hiện
  - Thực hiện biểu thức trong ( ) sâu nhất trước.
  - Thực hiện theo thứ tự ưu tiên các toán tử.

  => Tự chủ động thêm ( )

Ví dụ:

Viết biểu thức cho các mệnh đề:

- x lớn hơn hay bằng 3

  x >= 3

- a và b cùng dấu

  ((a>0) && (b>0)) || ((a<0) && (b<0))

  (a>0 && b>0) || (a<0 && b<0)

- p bằng q bằng r

  (p == q) && (q == r) hoặc (p == q && q == r)

- –5 < x < 5

  (x > –5) && (x < 5) hoặc (x > –5 && x < 5)

Bài tập:

1. x = 3 + 4 + 5;
2. x = y = z;
3. z *= ++y + 5;
4. a || b && c || d;

Bài giải:

1. x = ((3 + 4) + 5);
2. x = (y = z);
3. z *= (++y) + 5;
4. (a || (b && c)) || d;

Bài tập:

Bài 1: Tính

1. (5 > 3 && 4 < 8)
2. (4 > 6 && true)
3. (3 >= 3 || false)
4. (true || false) ? 4 : 5

Bài 2: Tính

1. 7 / 4
2. 14 % 5
3. 3 / 0

6. BIỂU THỨC

- Tạo thành từ các toán tử (Operator) và các toán hạng (Operand).
- Toán tử tác động lên các giá trị của toán hạng và cho giá trị có kiểu nhất định.
- Toán hạng: hằng, biến, lời gọi hàm...
- Định nghĩa gộp: Biểu thức được tạo thành được từ các hằng số, biến số và hàm số (toán hạng) liên kết với nhau bằng các phép toán số học (toán tử).
- Ví dụ:

```
int a = 2 + 3;
int b = a / 5;
int c = (a + b) * 5;
int d = (x >= 3);
(x >= 0) ^ (y < 0) (Biểu thức này kiểm tra gì?)
```

```
int year = 2000;
int month = 29;
if((year%4!=0)||(year%400!=0))
        return 28;
```

7. Câu lệnh - Statement

- Khái niệm
  - Là một chỉ thị trực tiếp, hoàn chỉnh nhằm ra lệnh cho máy tính thực hiện một số tác vụ nhất định nào đó.
  - Trình biên dịch bỏ qua các khoảng trắng (hay tab hoặc xuống dòng) chen giữa lệnh.
- Ví dụ

```
a = 2912;
```

```
a=2912;
```

```
a
=
2912;
```

Viết chương trình hoàn chỉnh gồm:

1. Khai báo các biến lưu trữ bán kính, chu vi và diện tích của hình tròn.

2. Khai báo hằng ký hiệu số pi

3. Cho bán kính có giá trị bằng 10

4. Tính chu vi và diện tích hình tròn

7. Câu lệnh - Statement

- Phân loại
  - Câu lệnh đơn: chỉ gồm một câu lệnh.
  - Câu lệnh phức (khối lệnh): gồm nhiều câu lệnh đơn được bao bởi { và }
- Ví dụ:

```
  a = 5;
  // Khối lệnh
  {
     b = 2;
     c = 10;
   }
}
```

Ví dụ: Tìm lỗi sai trong đoạn code sau:

```
#include<iostream>
using namespace std;


int main() {
  int a, b;
  a = 10;
  b = 5;
  {
    int c(1);
    a = b + c;
  }
  a = b;
  b = c;
}
```

8. Một số hàm hữu ích khác

Các hàm trong thư viện toán học

Thư viện: #include <math.h>

- 1 đối số đầu vào: double, trả kết quả: double
  - acos, asin, atan, cos, sin, …
  - exp, log, log10
  - sqrt
  - ceil, floor
  - abs, fabs
- 2 đối số đầu vào: double, trả kết quả: double
  - double pow(double x, double y)

8. Chương trình ví dụ các hàm hữu ích

```
#include <iostream>
#include <math.h>
int main() {
    float x = 2;
    std::cout<<cos(x)<<"\n"; // Hàm cos
    std::cout<<sin(x)<<"\n"; // Hàm sin
    std::cout<<tan(x)<<"\n"; // Hàm tan
    std::cout<<acos(x)<<"\n"; // Hàm arc cos
    std::cout<<asin(x)<<"\n"; // Hàm arc sin
    std::cout<<atan(x)<<"\n"; // Hàm arc tan
    std::cout<<log(x)<<"\n"; // Hàm log thường
    std::cout<<log10(x)<<"\n"; // Hàm log 10
    std::cout<<sqrt(x)<<"\n"; // Hàm căn bậc 2
    std::cout<<fabs(-x)<<"\n"; // lấy giá trị tuyệt đối
    std::cout<<pow(x,2); //Hàm mũ
    std::cin.get();
    return 0;
}
```

```
-0.416147
0.909297
-2.18504
nan
nan
1.10715
0.693147
0.30103
1.41421
2
4
```

Chúc các em học tốt!