CHƯƠNG 3 : CÁC TOÁN TỬ TRONG C++

3.1 Toán tử số học.

3.2 Toán tử quan hệ và logic .

3.3 Toán tử thao tác bit.

3.4 Các toán tử khác.

3.5 Biểu thức.

3.6 Độ ưu tiên của các toán tử.

Bài tập

3.1 Toán tử số học.

3.1.1 Phép chia / :

3.1.2 Phép chia nguyên lấy phần dư % :

3.1.3 Toán tử tăng ++:

3.1.4 Toán tử giảm -- :

3.1.5 Thứ tự ưu tiên các toán tử số học :

C++ có 8 toán tử số học :

Toán tử	Ý nghĩa	Ví dụ	Ghi chú
-	Lấy đối	-a ; -(a+b)	Toán tử 1 ngôi
--	Tự giảm dần	--x	Toán tử 1 ngôi
++	Tự tăng dần	++x	Toán tử 1 ngôi
+	Cộng	a + b	Toán tử 2 ngôi
-	Trừ	a –b	Toán tử 2 ngôi
*	Nhân	a * b	Toán tử 2 ngôi
/	Chia	a /b	Toán tử 2 ngôi
%	Chia lấy phần dư	a%b	Toán tử 2 ngôi

3.1.1 Phép chia / :

- Là toán tử 2 ngôi, toán hạng là các số (thực hay nguyên).

- Kết quả của phép chia :

  a) Nếu cả 2 toán hạng đều là số nguyên thì kết quả là là số nguyên (chặt cụt phần phân).

  b) Nếu có ít nhất 1 toán hạng là số thực thì cho ra một số thực .

  Ví dụ :

  ```
  int x = 3;
  float y = 3.;
  x/2  →   Giá trị 1 (số nguyên )
  y/2  →   Giá trị 1.5 ( số thực )
  ```

3.1.2 Phép chia nguyên lấy phần dư % :

- Là toán tử 2 ngôi, toán hạng là số nguyên , kết quả cho ra số nguyên.

13 % 3 = 1

-13 % 3 = -1

- Ghi chú :

Kết quả luôn cùng dấu với số bị chia.

3.1.3 Toán tử tăng ++:

- Là toán tử 1 ngôi, toán hạng là nguyên hay thực .
- Dạng viết 1 : ++ n ;

  Tác dụng : n tăng thêm 1 trước khi sử dụng giá trị n.
- Dạng viết 2 : n++ ;

  Tác dụng : n tăng thêm 1 sau khi sử dụng giá trị n.
- Minh họa :

3.1.4 Toán tử giảm --:

- Là toán tử 1 ngôi, toán hạng là nguyên hay thực .
- Dạng viết 1 : -- n ;

  Tác dụng : n giảm bớt 1 trước khi sử dụng giá trị n.
- Dạng viết 2 : n-- ;

  Tác dụng : n sẽ giảm bớt 1 sau khi sử dụng giá trị n

  Minh họa :

3.1.5 Thứ tự ưu tiên các toán tử số học :

Ưu tiên của các toán tử số học được cho trong bảng sau đây theo thứ tự từ trên xuống dưới. Các toán tử cùng độ ưu tiên sẽ được thực hiện từ trên trái sang phải.

Ưu tiên	Toán tử
1	++ -- -
2	* / %
3	+ -

Ghi chú :

Các ký hiệu trong các toán tử ++, -- không được viết rời.

3.2 Toán tử quan hệ và logic.

3.2.1 Các toán tử quan hệ.

3.2.2 Các toán tử logic.

3.2.3 Thứ tự ưu tiên của các toán tử quan hệ và logic.

- Các toán tử quan hệ và logic :

  Thường được sử dụng chung với nhau để tạo ra các kết quả đúng, sai.

- Trong C++:
  - Mọi số khác 0 đều được coi là giá trị đúng (true),
  - Giá trị duy nhất sai (false) mang hình thức số 0.

3.2.1 Các toán tử quan hệ .

Các toán tử quan hệ được dùng để so sánh 2 giá trị với nhau.

Sau đây là bảng các toán tử quan hệ và ý nghĩa của chúng :

Toán tử	Ý nghĩa	Ví dụ	
>	Lớn hơn	a > b	3>7 có giá trị 0
>=	Lớn hơn hay bằng	a >= b	3 >= 7 có gía trị 0
<	Nhỏ hơn	a < b	3<7 có giá trị 1
<=	Nhỏ hơn hay bằng	a <= b	3 <= 7 có gía trị 1
==	Bằng nhau	a == b	3 == 7 có giá trị 0
!=	Khác nhau	a != b	3 != 7 có giá trị 1

Ghi chú :

Các ký hiệu trong các toán tử >=,<=, ==, != không được viết rời.

3.2.2 Các toán tử logic.

- Các toán tử logic được dùng để nối kết hai giá trị, hoặc trong trường hợp phủ định sẽ tạo một giá trị đảo ngược. Các giá trị có thể nguyên hay thực .
- Trong C++ có 3 toán tử logic :
  - Phép phủ định 1 ngôi : !
  - Phép và (AND) : &&
  - Phép hoặc (OR) : || // ký hiệu đường ống trên bàn phím
- Ý nghĩa của các toán tử được cho trong bảng sau :

a	b	!a	a&&b	a||b
Khác không (1)	Khác không (1)	0	1	1
Khác không (1)	Bằng không	0	0	1
Bằng không	Khác không (1)	1	0	1
Bằng không	Bằng không	1	0	0

3.2.3 Thứ tự ưu tiên của các toán tử quan hệ và logic :

Ưu tiên	Toán tử
1	!
2	> >= < <=
3	== !=
4	&& ||

Ghi chú :

Các toán tử quan hệ và logic được dùng để thiết lập điều kiện rẽ nhánh trong câu lệnh if và điều kiện kết thúc vòng lặp trong các câu lệnh for, while và do.

3.3 Các Toán Tử Thao Tác Bit.

3.3.1 Bảng giá trị của các toán tử ~, &, |, ^

3.3.2 Các toán tử dịch chuyển

Khác với các ngôn ngữ lập trình cấp cao khác, C++ cung cấp nhiều toán tử có thể tác động tới những bit thực sự ở bên trong 1 biến.

Các toán tử thao tác bit chỉ dùng cho số nguyên hoặc ký tự.

Các toán tử thao tác bit bao gồm :

Toán tử	Ý nghĩa	Ví dụ
&	Phép và (AND) theo bit	a&b
|	Phép hoặc (OR) theo bit	a|b
^	Phép hoặc loại trừ (XOR) theo bit	a^b
<<	Dịch trái (Shift left)	a << 4
>>	Dịch phải (Shift right)	a>>4
~	Lấy phần bù theo bit (Not)	~a

Ghi chú :

Cách viết các toán tử : &&, &, ||, | nếu lẫn lộn sẽ bị lỗi cú pháp, hoặc trong một số trường hợp chương trình chạy được nhưng cho kết quả sai.

3.3.1 Bảng giá trị của các toán tử ~, &, |, ^ :

a	b	~a	a&b	a|b	a^b
1	1	0	1	1	0
1	0	0	0	1	1
0	1	1	0	1	1
0	0	1	0	0	0

3.3.2 Các toán tử dịch chuyển :

1. Toán tử dịch chuyển trái
2. Toán tử dịch chuyển phải
3. Ghi chú

3.3.2 Các toán tử dịch chuyển :

- Các toán tử dịch chuyển trái << , hoặc phải >> đều có thể dịch chuyển tất cả các bit nằm trong 1 byte hay 1 từ - theo một số lượng ấn định ở bên phải .

  Chẳng hạn :

n<<2	Dịch chuyển các bit của biến n sang trái 2 bit
n>>2	Dịch chuyển các bit của biến n sang phải 2 bit

1. Toán tử dịch chuyển trái :

- Khi thực hiện toán tử dịch chuyển trái các bit ở bên phải của biến sẽ được điền vào các giá trị 0.

Ví dụ 1:

```
short int  n = 201;
n << 2;
```

2. Toán tử dịch chuyển phải :

C++ sẽ thực hiện tùy theo kiểu dữ liệu của toán hạng bên trái :

a. Toán hạng bên trái có kiểu unsigned (int, long, char).

b. Toán hạng bên trái có kiểu signed (int, long, char).

a. Nếu toán hạng bên trái có kiểu unsigned (int, long, char).

- Phép dịch chuyển phải sẽ điền giá trị 0 vào các bit bên trái.

  Ví dụ 2:

  unsigned short int n = 39470; //hệ 10;

  n >> 2;

b. Nếu toán hạng bên trái có kiểu signed (int, long, char).

- Phép dịch chuyển phải sẽ điền bit dấu (số 1) vào các bit bên trái.

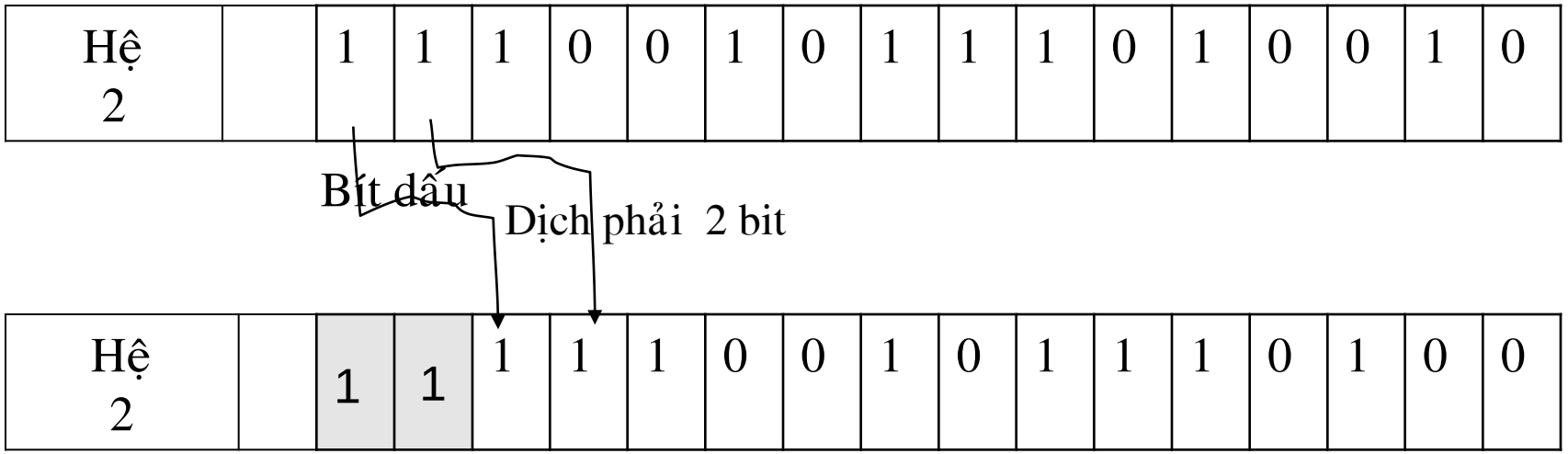
  Ví dụ 3:

  short int n = 26066; //hệ 10;

  Đổi n sang hệ 2 : 0110 0101 1101 0010

  n >> 2;

  Dịch chuyển phải n >> 2 sẽ là :



Kết quả : 0001 1001 0111 0100

Đổi n sang hệ 10 : 6516.

Ghi chú :

Thực chất phép dịch trái là hình thức nhân 2 liên tiếp , còn phép dịch phải phải là hình thức chia 2 liên tiếp.

Tức là : $x = a << n \equiv x = a*(2^n)$

$x = a >> n \equiv x = a/(2^n)$

3.4 Các toán tử khác.

3.4.1 Toán tử sizeof

3.4.2 Toán tử ( ) :

3.4.3 Toán tử dấu phảy ‘ , ‘ (Comma operator) :

3.4.1 Toán tử sizeof

- Toán tử sizeof cho ta kích thước (tính theo byte) của 1 kiểu dữ liệu cũng như một đối tượng dữ liệu. Cách viết toán tử như sau :

  sizeof (kiểu dữ liệu)

  sizeof (đối tượng dữ liệu)

  - Kiểu dữ liệu có thể là các kiểu chuẩn như int, float hoặc kiểu dữ liệu được định nghĩa bằng từ khóa typedef.
  - Đối tượng dữ liệu có thể là biến, mảng, cấu trúc ...(tên của vùng nhớ dữ liệu).

  Chẳng hạn :

```
int m;
sizeof(m)    = 4;
sizeof(int)  = 4;
sizeof(double) = 8;
```

3.4.2 Toán tử ( ) :

- Dùng để xác định trình tự ưu tiên các thành phần trong biểu thức.
- Nếu có nhiều toán tử () lồng nhau thì thực hiện ưu tiên từ trong ra ngoài.
- Nếu có nhiều toán tử () rời nhau thì thực hiện từ trái sang phải.

3.4.3 Toán tử dấu phảy ‘ , ‘ (Comma operator) :

- Được dùng để tạo sự thi hành tuần tự cho các thao tác, thường dùng trong câu lệnh for hay biểu thức vế phải của câu lệnh gán.
- Trong vế phải câu lệnh gán, thì giá trị toàn thể biểu thức là giá trị của biểu thức cuối cùng trong danh sách các biểu thức được tách biệt bởi dấu phảy.

Ví dụ 4:

```
int x = 10;
x = (x = x-5, 25/x);          // Khi đó x sẽ có giá trị là 5
```

Giá trị đầu của x là 10. biểu thức đầu bên vế phải gán x-5 cho x, vậy giá trị biểu thức này là 5. thực hiện biểu thức cuối, giá trị này được 25 chia cho, kết quả cuối cùng đem gán cho vế trái là x. Nên x = 5.

3.5 Biểu Thức.

3.5.1 Biểu thức gán :

3.5.2 Toán tử tam phân ?: (thể hiện biểu thức điều kiện)

3.5.3 Chuyển đổi kiểu dữ liệu :

- Biểu thức là một giá trị tạo ra từ sự kết hợp giữa các toán tử và các toán hạng để diễn đạt một công thức toán học nào đó. Toán hạng có thể xem là các đại lượng có một giá trị nào đó, và theo nghĩa đó, toán hạng có thể là biến, hằng, hàm . . .
- Mỗi biểu thức có một giá trị, và nói chung cái gì có giá trị đều được xem là biểu thức. Như vậy, các biến, hằng, hàm . . . đều có thể xem là biểu thức.
- Khi viết biểu thức có thể dùng các dấu ngoặc tròn (,) để thể hiện đúng trình tự tính toán trong biểu thức.

- Biểu thức được phân loại theo kiểu giá trị : nguyên, thực. Trong các mệnh đề logic, biểu thức được phân thành đúng (giá trị khác không), sai (giá trị bằng không).
- Biểu thức thường được dùng trong :
  - Vế phải của lệnh gán.
  - Làm tham số thực sự của hàm.
  - Làm chỉ số.

  . . .

  Ta giới thiệu các biểu thức : biểu thức gán, biểu thức điều kiện.

3.5.1 Biểu thức gán :

a) Một dạng viết khác trong câu câu lệnh gán : op=

Trong đó "op" là toán tử hai ngôi (+, -, *, /, %, &, |, <<, >>, ^)

Nếu e1,e2 là biểu thức thì e1 op= e2 tương đương với e1 = e1 op e2 nhưng cách viết trước hiệu quả hơn.

n = n + 1;	Có thể viết	n+ = 1;
x = x*(y+3);	Có thể viết	x* = y+3;
a = a%3	Có thể viết	a %= 3;
a = a/3	Có thể viết	a /= 3;

b) Biểu thức gán là biểu thức có dạng : v = e

- Trong đó v là biến, e là một biểu thức. Giá trị của biểu thức gán là giá trị của e, kiểu của nó là kiểu của v.

  (Nếu viết v = e; ta có câu lệnh gán)

- Biểu thức gán có thể sử dụng trong các toán tử và các câu lệnh như các biểu thức khác.

  Chẳng hạn : Nếu viết z = (y = 2) * (x = 6);

  có nghĩa là : y lấy giá trị 2, x có giá trị 6 và z có giá trị 12.

- Trường hợp riêng ta đã biết ý nghĩa của nó là dạng :

  a = b = c = 3;

3.5.2 Toán tử tam phân ?:

(thể hiện biểu thức điều kiện)

Biểu thức điều kiện thể hiện bởi toán tử tam phân có dạng :

e1 ? e2 : e3

Trong đó, e1, e2, e3 là các biểu thức nào đó.

Giá trị của biểu thức điều kiện bằng :
- *Giá trị của e2 nếu e1 khác 0 (Đúng),*
- *Giá trị của e3 nếu e1 bằng 0 (Sai).*

Ví dụ 5:

Kiểu của biểu thức điều kiện phải là kiểu cao nhất trong các kiểu của e2 và e3. Chẳng hạn nếu kiểu của e2 là int, kiểu của e3 là float thì kiểu của biểu thức điều kiện là float.

Biểu thức điều kiện có thể sử dụng như một biểu thức bất kỳ khác.

3.5.3. Chuyển đổi kiểu dữ liệu :

a) Trong biểu thức

b) Trong câu lệnh gán

c) Trong việc truyền tham số cho hàm

d) Sử dụng phép ép kiểu

a) Trong biểu thức :

Khi các toán hạng có kiểu khác nhau xuất hiện trong biểu thức, chúng được chuyển một cách tự động về một kiểu chung theo một số qui tắc.

- Trong biểu thức có chứa các biến kiểu char và int thì tất cả các biến kiểu char được chuyển thành int .
- Các toán tử số học có hai toán hạng (+,-,*,/,%) và nếu chúng khác nhau thì kiểu "thấp hơn" sẽ nâng thành kiểu "cao hơn" .

Cụ thể : char và short chuyển thành int

float chuyển thành double

int chuyển thành foat, double.

b) Trong câu lệnh gán :

- Giá trị của vế phải trong câu lệnh gán được chuyển đổi thành kiểu của vế trái (Kiểu kết quả là kiểu của vế trái) :

  - int có thể chuyển thành float.

  - float chuyển thành int bằng cách cắt bỏ phần phân .

  - double chuyển thành float bằng cách làm tròn.

  - long int chuyển thành short int bằng cách bỏ các bit cao vượt quá .

Ví dụ 6 :

c) Trong việc truyền tham số cho hàm :

Vấn đề chuyển đổi kiểu cũng xảy ra khi truyền tham số cho hàm.

d) Sư dụng phép ép kiểu :

Cuối cùng, việc chuyển kiểu tường minh (hay gọi là cast) được thực hiện theo cú pháp :

(Kdl) biểu_thức ;

Chẳng hạn :

```
int  p = 3, q = 2;
float x, y;
x = p/q;
y = (float)p/q;
```

Khi đó x bằng 1 và y = 1.5.

Cast (sắc thái) tạo cho giá trị của biến theo kiểu đúng nhưng nội dung thực sự của biến là không thay đổi.

3.6 Độ ưu tiên của các toán tử .

Được trình bày trong bảng sau với một số lưu ý :

- Các toán tử cùng mức ưu tiên (nằm trêm 1 dòng) thì trình tự tính toán được chỉ ra trong cột “trình tự kết hợp”.
- Một số các toán tử chưa được giới thiệu, chẳng hạn :
- Trong ưu tiên 1 :

  [ ] : dùng đẻ biểu diễn phần tử mảng.

  -> : Biểu diễn thành phần của cấu trúc thông qua con trỏ.
- Trong ưu tiên 2 :

  * : khai báo con trỏ .

Ưu tiên	Toán tử	Trình tư kết hợp	Ghi chú
1	() [ ] ->	Trái qua phải	
2	! ~ & * - ++ -- (type) sizeof	Phải qua trái	* : con trỏ ; & lấy địa chỉ.
3	* / %	Trái qua phải	* : Phép nhân
4	+ -	Trái qua phải	- : PT 2ngôi
5	<< >>	Trái qua phải	
6	< <= > >=	Trái qua phải	
7	== !=	Trái qua phải	
8	&	Trái qua phải	& : AND theo bit
9	^	Trái qua phải	
10	|	Trái qua phải	
11	&&	Trái qua phải	
12	||	Trái qua phải	
13	? :	Phải qua trái	
14	Op=	Phải qua trái	
15	,	Trái qua phải	

BÀI TẬP.

Bài 1:

Hãy cho biết giá trị của j sau đoạn chương trình:

```
int j;
char c='1';
j=(c<='9') &&(c>='0');
```

Bài 2:

Cho *b* bằng 5 và *c* bằng 8. Hãy cho biết giá trị của *a,b,c* sau khi thi hành riêng biệt từng dòng lệnh sau:

1. *a=b++ + c++;*
2. *a=b++ + ++c;*
3. *a=++b + c++;*
4. *a=++b + ++c;*
5. *b=a++ + ++a;*
6. *a += a+= a*

   Kiểm tra lại bằng chương trình.

Bài 3:

Các câu lệnh sau làm gì?

```
a^ =b;
b^=a;
a^=b;
a ^= b^= a^= b;
```

Bài 4:

1. Hãy cho biết giá trị của b và a sau đoạn chương trình:

```
int a,b=2;
b=(a=3,(5*b)+(a*=b));
```

2. Hãy cho biết giá trị của n và x sau đoạn chương trình:

```
int n,x=2;
x=x-1;
n=(n=5,n*=10+x++);
```

3. Hãy cho biết giá trị của a, b sau tùng câu lệnh :

```
int a=1,b= a ? 1:2;
b+=1;
a=(b==2)?1:2;
a=(b=2)?1:3;
a=(b=2)?1:2;
```

Bài 5:

Viết chương trình tính 2^n .

Bài 6:

Viết chương trình tính Max(a,b), Min(a,b)