BÀI 3 – TOÁN TỬ

Các toán tử là nền tảng của bất kỳ ngôn ngữ lập trình nào. Có thể định nghĩa các toán tử là các ký hiệu giúp chúng ta thực hiện các phép tính Toán học và logic cụ thể trên các toán hạng. Ví dụ: '+' là một toán tử được sử dụng để thêm vào: `a = b + c ;`

C ++ có nhiều toán tử cài sẵn và có thể được phân thành 6 loại:

1. Toán tử số học

Các toán tử này được sử dụng để thực hiện các phép toán số học / toán học trên các toán hạng. Ví dụ: (+, -, *, /,%, ++, --). Toán tử số học có hai loại:

a) ***Các toán tử đơn vị:*** Các toán tử vận hành hoặc làm việc với một toán hạng đơn là các toán tử đơn nguyên. Ví dụ: Các toán tử tăng dần (++) và giảm dần (--)

Ví dụ: int a = 5; a++ // biến a được tăng thêm 1 đơn vị thành 6

b) ***Toán tử nhị phân:*** Toán tử hoạt động hoặc làm việc với hai toán hạng là toán tử nhị phân. Ví dụ: Các toán tử cộng (+), trừ (-), nhân (*), Chia (/)

Ví dụ: a + b; x*y; p/q

Trong đó các phép toán %, ++, -- chỉ có thể áp dụng với kiểu dữ liệu số nguyên:

- Phép toán % : là phép toán chia lấy số dư. Vd: 8 % 5 = 3; 17 % 4 = 1; 15%5 = 0
- Phép toán ++ : tăng giá trị biến nguyên lên 1 đơn vị, có hai kiểu tăng: tăng trước khi sử dụng (++a) và tăng sau khi sử dụng giá trị biến (a++).

  Ví dụ: int a = 10 thì : b = (++a) -1 ; // b sẽ có giá trị bằng 10

  b = (a++) – 1; // b sẽ có giá trị bằng 9
- Phép toán -- : giảm giá trị biến nguyên xuống 1 đơn vị, có hai kiểu giảm: giảm trước khi sử dụng (--a) và giảm sau khi sử dụng giá trị biến (a--).

2. Toán tử quan hệ

Các toán tử quan hệ được sử dụng để so sánh giá trị của hai toán hạng. Ví dụ, kiểm tra xem một toán hạng có bằng toán hạng kia hay không, một toán hạng có lớn hơn toán hạng kia hay không, v.v.

a) ***Toán tử bằng:*** Được biểu diễn dưới dạng '**==**' , toán tử bằng để kiểm tra xem hai toán hạng đã cho có bằng nhau hay không. Nếu đúng, nó trả về **true**. Nếu không, nó trả về **false**. Ví dụ, 5 == 5 sẽ trả về true còn 3 == 7 sẽ trả về giá trị false.

b) ***Toán tử không bằng:*** Được biểu diễn là '**! =**' , Toán tử không bằng để kiểm tra xem hai toán hạng đã cho có bằng nhau hay không. Nếu không, nó trả về

true. Nếu không, nó trả về **false**. Nó là phần bù logic của toán tử '==' . Ví dụ: 5! = 5 sẽ trả về false, 3 != 7 sẽ trả về giá trị true

c) ***Toán tử lớn hơn:*** Được biểu diễn là '**>**' , toán tử lớn hơn kiểm tra toán hạng đầu tiên có lớn hơn toán hạng thứ hai hay không. Nếu vậy, nó trả về true. Nếu không, nó trả về false. Ví dụ: 6> 5 sẽ trả về true.

d) ***Toán tử nhỏ hơn:*** Được biểu diễn dưới dạng '**<**' , toán tử nhỏ hơn kiểm tra xem toán hạng đầu tiên có nhỏ hơn toán hạng thứ hai hay không. Nếu vậy, nó trả về true. Nếu không, nó trả về false. Ví dụ, 6 <5 sẽ trả về false.

e) ***Toán tử lớn hơn hoặc bằng:*** Được biểu diễn dưới dạng '**>=**' , toán tử lớn hơn hoặc bằng để kiểm tra xem toán hạng đầu tiên có lớn hơn hoặc bằng toán hạng thứ hai hay không. Nếu vậy, nó trả về true nếu không, nó trả về false. Ví dụ: 5 >= 5 sẽ trả về true.

f) ***Toán tử nhỏ hơn hoặc bằng:*** Được biểu diễn dưới dạng '<=' , toán tử nhỏ hơn hoặc bằng kiểm tra toán hạng đầu tiên nhỏ hơn hoặc bằng toán hạng thứ hai. Nếu vậy, nó trả về true, khác là false. Ví dụ: 5 <= 5 cũng sẽ trả về true.

3. Toán tử lôgic

Chúng được sử dụng để kết hợp hai hoặc nhiều điều kiện / ràng buộc hoặc để bổ sung cho việc đánh giá điều kiện ban đầu đang được xem xét:

a) ***Toán tử logic and:*** Có thể viết là **and** hoặc kí hiệu ' **&&**' trả về true khi thỏa mãn cả hai điều kiện đang xem xét. Nếu không, nó trả về false. Ví dụ, a && b trả về true khi cả a và b đều đúng (nghĩa là khác 0).

a	b	a && b
1	1	1
1	0	0
0	1	0
0	0	0

Trong toán tử logic AND , toán hạng thứ hai không được đánh giá nếu toán hạng đầu tiên là sai. Ví dụ sau sẽ in ra **false** ngay sau khi biểu thức (a==b) được đánh giá xong mà không cần xét đến biểu thức (a ==c).

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int a = 10, b = 4, c = 10;
7      cout <<((a == b) && (a == c));
8      return 0;
9  }
```

Vì thế trong khi sử dụng toán tử &&, chúng ta phải đặt điều kiện đầu tiên có xác suất sai cao để trình biên dịch không cần kiểm tra điều kiện thứ hai nếu điều kiện đầu tiên là sai

b) ***Toán tử logic or :*** Có thể viết là **or** hoặc kí hiệu '||' , toán tử or trả về true khi một (hoặc cả hai) điều kiện đang xem xét được thỏa mãn. Nếu không, nó trả về false. Ví dụ, a || b trả về true nếu một trong a hoặc b hoặc cả hai đều đúng (nghĩa là khác 0). Tất nhiên, nó trả về true khi cả a và b đều đúng.

a	b	a || b
1	1	1
1	0	1
0	1	1
0	0	0

Trong toán tử logic OR , toán hạng thứ hai trở đi sẽ không được đánh giá nếu toán hạng đầu tiên là true. Trong khi sử dụng || *(logic or),* chúng ta phải đặt điều kiện đầu tiên mà xác suất nhận đúng là cao để trình biên dịch không cần kiểm tra điều kiện thứ hai nếu điều kiện đầu tiên là đúng.

c) ***Toán tử logic not(toán tử phủ định):*** Có thể viết là **not** hoặc kí hiệu '!' , toán tử not trả về true nếu điều kiện đang xem xét không được thỏa mãn. Ngược lại, nó trả về false. Ví dụ: bool a = true , b = false thì not(a) sẽ là false, not(b) là true.

4. Toán tử Bitwise *(Kiến thức nâng cao)*

Các toán tử Bitwise được sử dụng để thực hiện các phép toán mức **bit** trên các toán hạng. Các phép toán như cộng, trừ, nhân, v.v. có thể được thực hiện ở mức bit để xử lý nhanh hơn. Trong C, 6 toán tử sau là toán tử theo chiều bit (hoạt động ở cấp độ bit):

a) **&** (Bitwise AND) trong C hoặc C ++ nhận hai số làm toán hạng và thực hiện AND trên mỗi bit của hai số. Kết quả của **&** chỉ là 1 nếu cả hai bit đều là 1. Toán tử & có thể được sử dụng để nhanh chóng kiểm tra xem một số là số lẻ hay số chẵn. Giá trị của biểu thức (x & 1) chỉ khác 0 nếu x là số lẻ, nếu không giá trị sẽ là 0.

b) | (bitwise OR) trong C hoặc C ++ nhận hai số làm toán hạng và thực hiện OR trên mỗi bit của hai số. Kết quả của OR là 1 nếu bất kỳ bit nào trong hai bit là 1.

c) **^** (Bitwise XOR) trong C hoặc C ++ nhận hai số làm toán hạng và thực hiện XOR trên mỗi bit của hai số. Kết quả của XOR là 1 nếu hai bit khác nhau.

d) **<<** (left shift) trong C hoặc C ++ nhận hai số, dịch chuyển trái các bit của toán hạng đầu tiên, toán hạng thứ hai quyết định số vị trí cần dịch chuyển. Toán tử dịch trái tương đương với phép nhân với 2 . Nó chỉ hoạt động nếu các số là số dương.

e) **>>** (right shift) trong C hoặc C ++ nhận hai số, dịch phải các bit của toán hạng đầu tiên, toán hạng thứ hai quyết định số vị trí cần dịch chuyển. Toán tử dịch phải tương đương với phép chia cho 2 . Nó chỉ hoạt động nếu các số là số dương.

f) **~** (Bitwise NOT) trong C hoặc C ++ nhận một số và đảo ngược tất cả các bit của nó

Ví dụ:
```
int a = 5, b = 9; // a = 5 (00000101), b = 9 (00001001)
    cout << (a & b); // 1(00000001)
    cout << (a | b); // 13(00001101)
    cout << (a ^ b); // 12(00001100)
    cout << (~ a); // 250(11111010)
    cout << (b << 1) // 18 (9x2 = 18)
    cout << (b >> 1) // 4 (9/2 = 4)
```

Lưu ý khi sử dụng các toán tử bitwise:

- Các toán tử dịch chuyển trái *(left shift)* và phải *(right shift)* không được sử dụng cho các số âm. Nếu toán hạng thứ hai (quyết định số lượng dịch chuyển) là một số âm, nó dẫn đến hành vi không xác định trong C. Ví dụ: kết quả của cả 1 << - 1 và 1 >> -1 là không xác định. Ngoài ra, nếu số được dịch chuyển nhiều hơn kích thước của số nguyên, hành vi là không xác định. Ví dụ, 1 << 33 là không xác định nếu các số nguyên được lưu trữ bằng 32 bit.
- Toán tử XOR bitwise **^ :** là toán tử hữu ích nhất được sử dụng trong nhiều vấn đề. Một ví dụ đơn giản có thể là *"Cho một tập hợp các số trong đó tất cả các phần tử đều xuất hiện một số lần ngoại trừ một số, hãy tìm số xuất hiện lẻ"* Vấn đề này có thể được giải quyết một cách hiệu quả bằng cách thực hiện XOR tất cả các số.
- Các toán tử bitwise không nên được sử dụng thay cho các toán tử logic. Kết quả của các toán tử logic (&&, || và!) Là 0 hoặc 1, nhưng các toán tử bitwise trả về một giá trị nguyên. Ngoài ra, các toán tử logic coi mọi toán hạng khác 0 là 1.
- Toán tử ~ nên được sử dụng cẩn thận. Kết quả của toán tử ~ trên một số nhỏ có thể là một số lớn nếu kết quả được lưu trữ trong một biến không dấu. Và kết quả có thể là một số âm nếu kết quả được lưu trữ trong một biến có dấu.

Tham khảo thêm ứng dụng của toán tử bitwise tại địa chỉ:

https://www.geeksforgeeks.org/bit-tricks-competitive-programming/

5. Toán tử gán giá trị

Toán tử gán giá trị: kí hiệu "=", được sử dụng để gán giá trị cho một biến. Toán hạng bên trái của toán tử gán phải là một biến và toán hạng bên phải của toán tử gán là một giá trị (giá trị đơn lẻ, giá trị của một biểu thức). Giá trị ở phía bên phải phải cùng kiểu dữ liệu với biến ở phía bên trái, nếu không trình biên dịch sẽ xuất hiện lỗi.

Các dạng toán tử gán:

- "**=**": Đây là toán tử gán đơn giản nhất. Toán tử này được sử dụng để gán giá trị ở bên phải cho biến ở bên trái. (ví dụ: a = 10; b = 20; ch = 'y';)
- "**+=**" : Toán tử này là sự kết hợp của các toán tử '+' và '='. Toán tử này đầu tiên thêm giá trị hiện tại của biến ở bên trái vào giá trị ở bên phải và sau đó gán kết quả cho biến ở bên trái.
  Ví dụ: *(a += b) cũng có thể được viết thành (a = a + b), nếu giá trị ban đầu được lưu trong a là 5 thì (a += 6) = 11.*
- *Tương tự ta có các phép gán: "**-=**"; "***=**"; "**/=**";*

6. Toán tử sizeof

sizeof là một toán tử được sử dụng nhiều trong C hoặc C ++ . Nó là một toán tử đơn phân thời gian biên dịch có thể được sử dụng để tính toán kích thước của toán hạng của nó. Kết quả của sizeof thuộc loại số nguyên không dấu thường được ký hiệu là size_t. sizeof có thể được áp dụng cho bất kỳ kiểu dữ liệu nào, bao gồm các kiểu nguyên thủy như kiểu số nguyên và dấu phẩy động, kiểu con trỏ hoặc kiểu dữ liệu phức hợp như structure, union, v.v.

Toán tử sizeof () được sử dụng theo nhiều cách khác nhau tùy theo kiểu toán hạng:

a) ***Khi toán hạng là Kiểu dữ liệu:*** Khi sizeof () được sử dụng với các kiểu dữ liệu như int, float, char… vv, nó chỉ trả về lượng bộ nhớ được cấp cho các kiểu dữ liệu đó. Ví dụ: *cout << sizeof(int); // in ra 4, là số byte kiểu int*

b) ***Khi toán hạng là một biểu thức:*** toán tử sizeof () sẽ trả về kích thước của biểu thức. Ví dụ:

```
int a = 10;
double x = 3.5;
cout << sizeof(a+x);   | sẽ in ra giá trị là 8 (8 byte)
```

7. Toán tử điều kiện

Toán tử điều kiện có dạng : *Biểu thức 1* **?** *Biểu thức 2* **:** *Biểu thức 3*

Ở đây, Biểu thức1 là điều kiện để được đánh giá. Nếu điều kiện *Biểu thức 1* là **true** thì trả về kết quả của *Biểu thức 2*, ngược lại nếu điều kiện *Biểu thức 1* là **false** thì sẽ trả về kết quả của *Biểu thức 3*

```
int a = 0;
cout << (a?4:8);
```

Đoạn mã trên in ra giá trị 8 vì a = 0 tương đương với false (những giá trị khác 0 đều được máy tính hiểu là true), lưu ý phải có dấu ngoặc đơn để máy tính hiểu a?4:8 là một biểu thức, nếu không có thì máy tính chỉ in ra giá trị biến a.

BÀI TẬP VẬN DỤNG

Bài tập 300: Cho biết kết quả của những đoạn mã sau

1	`int a = 10, b = 8;`
	`a++;`
	`b--;`
	`cout << (a+b);`	

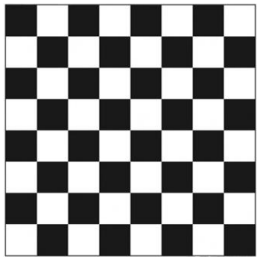
2	`int a = 10, b = 4;`
	`a /= b ;`
	`b *= a ;`
	`cout << "a = " << a << "\nb = " << b;`	

3	`int a = 10, b = 4;` `cout << (a>b?(a-b):(b-a));`	
4	`int a = 15, b = 11;` `cout << (a^b);`	
5	`int a = 15, b = 11;` `cout << (a&b);`	
6	`int a = 12;` `cout << (a<<3);`	
7	`int a = 70;` `cout << (a >> 2 );`	
8	`int a = 75;` `(a%2==0)?cout <<a<<" chan":cout<<a<<" le" ;`	

9	`int a = 30;`
	`long long c = 10000;`
	`cout << sizeof(a+b);`	

Bài tập 301: Cho 3 số thực a,b,c. Hãy viết biểu thức logic để cho giá trị true khi bộ a,b,c lập thành 3 cạnh của một tam giác.

Bài tập 302: Cho bàn cờ vua như hình sau, các hàng và cột được đánh số từ 1 tới 8. Giả sử trên bàn cờ này có hai ô (a,b) : *hàng a, cột b* và (c,d) : *hàng c, cột d* . Hãy viết biểu thức logic cho kết quả true khi:



a) Hai ô *(a,b)* và *(c,d)* cùng màu

b) Quân Xe ở ô *(a,b)* khống chế được ô *(c,d)*

c) Quân Hậu ở ô *(a,b)* khống chế được ô *(c,d)*

d) Quân Mã ở ô *(a,b)* khống chế được ô *(c,d)*

Bài tập 303: Viết chương trình đọc từ bàn phím 2 số nguyên dương a,b. Hoán đổi giá trị của hai số này rồi in ra màn hình kết quả của việc hoán chuyển nói trên.

Bài tập 304: Viết chương trình đọc 3 số a,b,c từ tệp BT302.inp, ghi vào tệp BT302.out giá trị lớn nhất trong 3 số này.

Bài tập 305: Viết chương trình tính giá trị của tổng S = 1 + 2 + 3 + ... + N với giá trị N được nhập trực tiếp từ bàn phím, tổng S được in ra màn hình.

Bài tập 306: Có (m) chiếc kẹo và (n) em bé. Hãy viết chương trình đọc hai số m,n từ tệp BT306.INP và kiểm tra xem có thể chia đều m chiếc kẹo này cho n em bé hay không. Kết quả ghi vào tệp BT306.OUT: số 1 nếu chia được hết và ngược lại ghi số 0.

Bài tập 307: Để chào mừng ngày thành lập Đoàn 26.03, đội văn nghệ của trường THPT Ngô Quyền được cử đi biểu diễn giao lưu ở các phường trong thành phố Hạ Long. Đội văn nghệ có (m) bạn nam và (n) bạn nữ được chia thành các tổ, mỗi tổ sẽ đi giao lưu ở một phường khác nhau. Biết rằng số lượng nam và nữ phải được chia đều giữa các tổ và sau khi chia thì mỗi học sinh thuộc về một tổ.

Yêu cầu: Em hãy lập trình để giúp đội văn nghệ nhà trường biết được có thể chia được tối đa bao nhiêu tổ ? Mỗi tổ có bao nhiêu bạn nam và bao nhiêu bạn nữ ?

Dữ liệu vào: Cho từ tệp BT307.INP chứa hai số nguyên m,n ($1 \le n,m \le 10^{15}$)

Kết quả: ghi vào tệp BT307.OUT gồm:

- Dòng thứ nhất ghi số tổ tối đa có thể chia được
- Dòng thứ 2 ghi ra số nam, nữ của mỗi tổ.

BT307.INP	BT307.OUT
48 72	24 2 3

Bài tập 308: Trong siêu thị, khách hàng thanh toán tiền tại quầy thu ngân. Sau khi máy đọc mã vạch giá tiền của từng sản phẩm sẽ thông báo ra tổng số tiền mà khách phải trả là S đồng, khách hàng đưa cho nhân viên thu ngân số tiền là K đồng ($K \geq S$), K,S là bội của 1000. Nhân viên thu ngân nhận tiền và trả lại tiền thừa (nếu có), em hãy lập trình giúp nhân viên thu ngân trả lại tiền thừa cho khách hàng sao cho tổng số tờ tiền phải trả là ít nhất. Biết rằng tại quầy thu ngân chỉ có các loại tiền 5000, 2000, 1000 với số lượng không hạn chế. Yêu cầu:

- Đọc từ tệp BT308.INP hai số nguyên dương S và K *($0 < S, K \leq 10^9$)*
- Tính số tiền thừa phải trả lại cho khách hàng (nếu có)
- Tính số lượng tờ tiền 1000, 2000, 5000 phải trả lại sao cho số tờ ít nhất

Kết quả tính được ghi vào tệp BT308.OUT

BT308.INP	BT308.OUT
13000 20000	7000 0 1 1

Trong test ví dụ trên khách phải trả 13000, đưa cho thu ngân 20000 nên số tiền phải trả lại là 7000 và trả 1 tờ 5000, 1 tờ 2000, 0 tờ 1000

Bài 1: (6,0 điểm) **QUẦY THU NGÂN**

Gồm 06 test, mỗi test 1,0 điểm. Một test có 2 yêu cầu, yêu cầu câu a 0,25 điểm, câu b 0,75 điểm.

Test	Nhập từ bàn phím	Xuất ra màn hình	Điểm
1	18000 30000	-So tien tra lai cho khach hang: 12000 dong. -So to 5000 dong: 2 to. -So to 2000 dong: 1 to. -So to 1000 dong: 0 to.	0,25 0,25 0,25 0,25
2	1000 20000	-So tien tra lai cho khach hang: 19000 dong. -So to 5000 dong: 3 to. -So to 2000 dong: 2 to. -So to 1000 dong: 0 to.	0,25 0,25 0,25 0,25
3	2000 30000	-So tien tra lai cho khach hang: 28000 dong. -So to 5000 dong: 5 to. -So to 2000 dong: 1 to. -So to 1000 dong: 1 to.	0,25 0,25 0,25 0,25
4	69000 100000	-So tien tra lai cho khach hang: 31000 dong. -So to 5000 dong: 6 to. -So to 2000 dong: 0 to. -So to 1000 dong: 1 to.	0,25 0,25 0,25 0,25
5	155000 200000	-So tien tra lai cho khach hang: 45000 dong. -So to 5000 dong: 9 to. -So to 2000 dong: 0 to. -So to 1000 dong: 0 to.	0,25 0,25 0,25 0,25
6	497000 500000	-So tien tra lai cho khach hang: 3000 dong. -So to 5000 dong: 0 to. -So to 2000 dong: 1 to. -So to 1000 dong: 1 to.	0,25 0,25 0,25 0,25

TEST	VANNGHE.INP	VANNGHE.OUT
1	5 17	1 5 17
2	50 110	10 5 11
3	250 250	250 1 1
4	3250 500	250 13 2
5	23250 21500	250 93 86
6	1000000 1	1 1000000 1
7	6 1000000	2 3 500000
8	8 100000000002	2 4 50000000001
9	1000000000000 8	8 125000000000 1
10	100000000000002 9	3 33333333333334 3