Date 29/09/2023
Page

operators in C++

BODMAS
↳ Bracket open Divide Multiplication Addition Subtraction.

$2 \times 3 - 4$
$6 - 4 = \boxed{2}$

$2 \times 3 - 4/2$
$6 - 2 = \boxed{4}$

Arithmetic operators

Unary

Binary
→ +
−
×
% (Rem) modulus.
/

precedence

int < float < double

$\frac{float}{int} = float$

$\frac{int}{int} = int$

Operator
↓
$2 + 3 = 5$
↑ ↖
Operand operand

$16/4 = 4$

$18/4 = 4$

$18/5 = 3$

$\frac{12.4}{4} = 3.1$

$63 \% 4 = 3$

$12.6 \times 3 = 43.8$

{* / ÷} > {+, −} ⇔ Left to Right

Precedence.

Associativity.

e.g (i) 3+4-6-2

7-6-2

1-2 = (-1).

(ii) 2×3-4/2

6-2

(4)

unary operator

- ++
- --

(i) ++

(a) post increment : a++ → a=a+1

(b) pre increment : ++a → a=a+1

e.g (i) a=10

b = a++

b=10

a=11

first assignment,
then increment.

(ii) a=10

b = ++a

a=11

b=11

first increment,
then assignment

(ii) --

(a) post decrement : a-- → a=a-1

(b) pre decrement : --a → a=a-1

e.g (i) a=10

b=a--

b=10

b=9

first assignment
then decroment

(ii) a=10

b=--a

a=9

b=9

first ~~assignment~~ derement
then assigment.

Comparison Operator

{ ==, >, <, >=, <=, != }

Answer of comparison operator is yes or NO i.e 1 or 0 (boolean).

(i) ==

e.g (i) 3==4 = 0 (ii) 4==4 = 1

↑ False ↑ ~~yes~~ True.

(ii) >

e.g (i) 10>5 = 1 (ii) 10>20 = 0

(iii) <

e.g (i) 10<5 = 1

(ii) 5>4>3 left to right.

1>3

0 As.

(iv) >=

e.g (i) 4>=4 = 1 (ii) 8>=4 = 1

(iii) 10<=8 = 0

(V) !=

eg (i) 4!=5 = 1 (ii) 5!=5 = 0

(vi) <=

e.g. (i) 4<=5 = 1

(ii) 8<=7 = 0

Precedence

$\{ >, <, >=, <= \} > \{ ==, != \}$

left to Right ← Associativity.

e.g

① $\underline{5>4}<3==2$

$\underline{1<3}==2$

$\underline{1==2}$

$\boxed{0}$ A.

② $\underline{3>4}>5 != 1$

$\underline{0>5}$

$\underline{0 != 1}$

$\boxed{1}$ A

Logical operator

(&&, ||, !)

AND, OR, Not

A	B	And(&&)
0	0	0
0	1	0
1	0	0
1	1	1

e.g:-

2 && 3 ⇓ $\boxed{1}$, 0 && 4 ⇓ 0

A	B	OR
0	0	0
0	1	1
1	0	1
1	1	1

4||5=1

0||5=1

Non-zero is considered as true.

!5 = 0

!0 = 1

A	NOT
0	1
1	0

e.g :-

a , b, c   a=10
                                  b=15
if a>b, a>c   c=2
yes.

```
if(a>b)
{ if(a>c)
    cout<<"yes";
  else
    cout<<"NO";
}
else
cout<<"NO";
```

⇒

```
       0 ← 0 → 1
     10>15       10>2
if((a>b) && (a>c))
   cout<<"yes";
else
cout<<"NO";
   // NO
```

e.g

vowel or consonant.

```
char name='a';

if(name=='a' || name=="e" || name=='i'
      || name=='o' ; name=='u')
   {
      cout<<"vowel";
   }
   else
      cout<<"consonant";
```

SHORT Circuit AND(&&)

The expression is evaluated until we get one false result, because if we do AND(&&) operation with '0' result will be '0' always, No further expression will be evaluated.

SHORT circuit (OR)

The expression is evaluated until we get ~~first~~ one 'True'(1) result because we do OR(||) with true('1') result will will true always, so no further expression will be evaluated.

Bitwise Operator

{ &, |, ^, ~, <<, >> }

& ↑ Bitwise And

| ↑ Bitwise OR

^ ↑ XOR

~ ↑ Compliment

<< ↑ Left shift

>> ↑ Right shift

e.g (i) 2 & 3 = 2

```
   ↓↓
   10
 & 11
 ----
   10 ← 2
```

(ii) 2 | 3 = 3

```
   10
 | 11
 ----
   11 ← 3
```

(iii) ^ (XOR)

A	B	^(XOR)
0	0	0
0	1	1
1	0	1
1	1	0

2 ^ 3 = 1

```
   10
   11
 ----
   01 ← 1
```

(iv) << (left shift

$6 << 1 = 12$ → $6 \times 2^1 = 12$.

↓ left shift, one time.

0.....000110

↙↙↙↙↙↙↙↙

1100 ← 12

$6 << 2 = 24$

if we do left shift, on every shift value get double. (2^n).

...000110 → 6

↙↙↙

0 -...000..1 100 → 12

↙↙↙↙

...0001 1000 → 24

$(num \times 2^n)$

$6 << 2$

↓

$6 \times 2^2 = 6 \times 4 = 24$.

$-6 << 2$ X

<<, only for +ve.

(v) >>, right shift

$6 >> 1$

↓

110

↓↓↓

011 ← 3

$\frac{num}{2^n}$

if we do right shift, on every shift value get half.

VI ~ (compliment.

~5

00...0000101

11 1111010 Ans.

↗ -ve

~ (0 → 1
1 → 0)

1's 0 0 0 0 0 1 0 1

+ 1

0.0 0 0 0 1 1 0 $\boxed{-6}$

~5 → -6

~8 = -9

~13 → -14

precedence

{ <<, >> } > { 4 , 1 , ^ }

(Assignment operator)

{ +=, -=, *=, /=, %= }

a *= 3 ← a = a * 3

a /= 2 ← a = a/2

Refer : Precedence table from gfg.