Netaji Subhas University

Subject Name:- Computer science
Topic Name:- Data type in c lang
Faculty Name:- Dr. Ranjan kumar Mishra
Dept Name:- IT

Operator

- An operator is a symbol that tells the compiler to perform specific mathematical or logical functions. C language is rich in built-in operators and provides the following types of operators.
  - Arithmetic Operators
  - Relational Operators
  - Logical Operators
  - Bitwise Operators
  - Assignment Operators
  - Misc. Operators

Types of Operator(According to Operand):

Let us see the mathematical expression : a+b

where a and b are operands and + is an operator.

1. Unary Operator:-> The operator which requires only one operand to produce a new value.
   Ex:- minus(-), increment(++),decrement(--),Not(!), sizeof() and address operator(&)

2. Binary Operator:-> The operator which requires minimum two operands to produce a new value or to perform any operation.
   Ex:- plus(+), minus(-), relational operators , logical operators(except Not(!)) etc.

Arithmetic Operator

- + (Addition) Operator: It is a Binary Operator. It return the sum of two operands without effecting the value of any operand.
  - Consider a = 12 & b = 15.
  - If we write a + b then, it will return 27.
  - If we write 10 + 2 then, it will return 12.

Arithmetic Operator

- - (Subtraction) Operator: It is a Binary Operator. It returns the subtraction of two operands without effecting the value of any operand.
  - Let us consider a = 12 & b = 5.
  - If we write a - b then, it will return 7.
  - If we write 10 - 2 then, it will return 8.

Arithmetic Operator

- * (Multiplication) Operator: It is a Binary Operator. It return the product of two operands without effecting the value of any operand.
  - Let us consider a = 2 & b = 5.
  - If we write a * b then, it will return 10.
  - If we write 10 * 2 then, it will return 20.

Arithmetic Operator

- / (Division) Operator: It is a Binary Operator. It return the quotient by dividing left operand with right operand without effecting the value of any operand.
  - Let us consider a = 2 & b = 5.
  - If we write a / b then, it will return 0.
  - If we write 10 / 2 then, it will return 5.

Arithmetic Operator

- % (Modulus) Operator: It is a Binary Operator. It return the remainder after dividing left operand by right operand without effecting the value of any operand.
  - Let us consider a = 2 & b = 5.
  - If we write a % b then, it will return 2.
  - If we write 10 % 2 then, it will return 0.

Arithmetic Operator

- ++ Increment Operator: It is a Unary Operator. It increase the value of operand used with this operator by 1 and assigned the new updated value in the operand and return the value according to its notation (prefix/postfix).
  - Let us consider a = 4. Then,
    - Using prefix notation:

      b = ++a,

      value of a & b will be respectively 5 & 5.
    - Using postfix notation:

      b = a++,

      value of a & b will be respectively 5 & 4.

Arithmetic Operator

- **-- Decrement Operator**: It is a Unary Operator. It decrease the value of operand used with this operator by 1 and assigned the new updated value in the operand and return the value according to its notation (prefix/postfix).
  - Let us consider a = 4. Then,
    - Using prefix notation:

      b = --a,

      value of a & b will be respectively 3 & 3.
    - Using postfix notation:

      b = a--,

      value of a & b will be respectively 3 & 4.

Relational Operator

- == (is Equals to?) Operator
- != (is Not Equals to?) Operator
- < (is Less Than?) Operator
- > (is Greater Than?) Operator
- <= (is Less Than or Equals to?) Operator
- >= (is Greater Than or Equals to?) Operator

Relational Operator

- == (is Equals to?) Operator: It is also a Binary Operator. It compare the value of two operand and check whether they are equal or not, without effecting the value of any operand. If they are equal then it returns true else, it returns false.
  - Let us consider a = 5, b = 6
  - If we write (a == b) then, it returns false.
  - If we write (5 == 5) then, it returns true.
  - If we write (3 == 2) then, it returns false.

  Note: Relational operators always returns a Boolean value (0 or 1)

Relational Operator

- != (is Not Equals to?) Operator: It is Binary Operator. It compare the value of two operand and check whether they are equal or not, without effecting the value of any operand. If they are equal then it returns false else, it returns false.
  - Let us consider a = 5, b = 6
  - If we write (a != b) then, it returns true.
  - If we write (5 != 5) then, it returns false.
  - If we write (3 != 2) then, it returns true.

Relational Operator

- < (is Less Than?) Operator: It is Binary Operator. It compare the value or two operand and check whether left operand is smaller than right operand or not, without effecting the value of any operand. If left operand is smaller then, it returns true else, it returns false.
  - Let us consider a = 5, b = 6
  - If we write (a < b) then, it returns true.
  - If we write (5 < 5) then, it returns false.
  - If we write (3 < 2) then, it returns false.
  - If we write (2 < 3) then, it returns true.

Relational Operator

- > (is Greater Than?) Operator: It is Binary Operator. It compares the value or two operand and check whether left operand is greater than right operand or not, without effecting the value of any operand. If left operand is greater then, it returns true else, it returns false.
  - Let us consider a = 5, b = 6
  - If we write (a > b) then, it returns false.
  - If we write (5 > 5) then, it returns false.
  - If we write (3 > 2) then, it returns true.
  - If we write (2 > 3) then, it returns false.

Relational Operator

- <= (is Less Than or Equals to?) Operator: It is Binary Operator. It compare the value or two operand and check whether left operand is smaller than or equals to right operand or not, without effecting the value of any operand. If left operand is smaller or equal then, it returns true else, it returns false.
  - Let us consider a = 5, b = 6
  - If we write (a < b) then, it returns true.
  - If we write (5 < 5) then, it returns true.
  - If we write (3 < 2) then, it returns false.
  - If we write (2 < 3) then, it returns true.

Relational Operator

- >= (is Greater Than or Equals to?) Operator: It is Binary Operator. It compare the value or two operand and check whether left operand is greater than or equals to right operand or not, without effecting the value of any operand. If left operand is greater or equal then, it returns true else, it returns false.
  - Let us consider a = 5, b = 6
  - If we write (a < b) then, it returns false.
  - If we write (5 < 5) then, it returns true.
  - If we write (3 < 2) then, it returns true.
  - If we write (2 < 3) then, it returns false.

Bitwise Operator

- Bitwise & (AND) Operator
- Bitwise | (OR) Operator
- Bitwise ^ (XOR) Operator
- Bitwise ~ (One's Compliment) Operator
- Bitwise << (Left Shift) Operator
- Bitwise >> (Right Shift) Operator

Bitwise Operator

- Bitwise operators work on bits and perform bit by bit operation. All Bitwise Operators are Binary Operator except ~(One's Compliment) operator. The Truth table for bitwise & (AND), | (OR), ^ (XOR) operator

p	q	p & q	p | q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

- There are two more bitwise operator i.e. >> (Right Shift), << (Left Shift) Operator.

Bitwise Operator

- **Binary & (AND) Operator**:
  - Bitwise & (AND) operator requires two operands to perform AND operation bit by bit on them.
  - Let us consider a = $(5)_{10} = (0101)_2$, b = $(3)_{10} = (0011)_2$
  - Then c = a & b will be 1 that can be calculated as AND operation perform on a and b as shown below:

```
a = 0101
b= 0011
---------
c = 0001
```

$c = (0001)_2 = (1)_{10}$

Bitwise Operator

- **Binary| (OR) Operator**:
  - Bitwise | (OR) operator requires two operands to perform OR operation bit by bit on them.
  - Let us consider a = $(5)_{10}$ = $(0101)_2$, b = $(3)_{10}$ = $(0011)_2$
  - Then c = a | b will be 7 that can be calculated as OR operation perform on a and b as shown below:

a = 0101
b= 0011
---
c = 0111

c = $(0111)_2$ = $(7)_{10}$

Bitwise Operator

- **Binary ^ (XOR) Operator**:
  - Bitwise ^ (XOR) operator requires two operands to perform XOR operation bit by bit on them.
  - Let us consider a = $(5)_{10} = (0101)_2$, b = $(3)_{10} = (0011)_2$
  - Then c = a ^ b will be 6 that can be calculated as OR operation perform on a and b as shown below:

a = 0101
b= 0011
---
c = 0110

$c = (0111)_2 = (6)_{10}$

Bitwise Operator

- **Binary ~ (One's Complement) Operator**:
  - Bitwise ~ (One's Complement) operator require only a single operands to perform it's operation bit by bit or it performs one's compliment on the operand.
  - Let us consider a = $(5)_{10}$ = $(0101)_2$
  - Then b = ~a will be 10 that can be calculated as inverting each bit as shown below:

a = 0101
---
b = 1010

b = $(0111)_2$ = $(10)_{10}$

Bitwise Operator

- **Binary << (Left Shift) Operator**:
  - This operator shift the each bit of the operand to left.
  - It requires two operand
    - Left operand on which operation is to be performed
    - Right operand to specify how many times bits are shifted by 1.
  - Let us consider a = $(5)_{10} = (0101)_2$
  - Then b = a << 2 will be 4 that can be calculated as shown below

```
a = 0101
----------
b = 1010
----------
b = 0100
```

$b = (0100)_2 = (4)_{10}$

Bitwise Operator

- **Binary >> (Right Shift) Operator**:
  - This operator shift the each bit of the operand to right.
  - It requires two operand
    - Left operand on which operation is to be performed
    - Right operand to specify how many times bits are shifted by 1.
  - Let us consider a = $(5)_{10} = (0101)_2$
  - Then b = a >> 2 will be 1 that can be calculated as shown below

```
a = 0101
----------
b = 0010
----------
b = 0001
```

$b = (0001)_2 = (1)_{10}$

Assignment Operator

- = (Simple Assignment) Operator
- += (Add AND Assignment) Operator
- -= (Subtract AND Assignment) Operator
- *= (Multiply AND Assignment) operator
- /= (Divide AND Assignment) operator
- %= (Modulus AND Assignment) operator
- <<= (Left Shift AND Assignment) operator
- >>= (Right Shift AND Assignment) operator
- &= (Bitwise AND and Assignment) operator
- |= (Bitwise OR and Assignment) operator
- ^= (Bitwise XOR and Assignment) operator

Assignment Operator

- **= (Simple Assignment) operator**: This operator assign the value of the operand or value returned after executing an expression on it's right to the operand to it's left.
  - Let us consider a = 5, b = 3, c = 0
  - If we write c = a + b, then c will be 8
  - And if we write c = a, then c will be 5

Assignment Operator

- **+= (Add AND Assignment) operator**: This operator add the operand on it's both sides and assign the result to the operand on it's left.
  - Let us consider a = 5, b = 3, c = 2
  - If we write c += a + b, then c will be 10,
  - If we write c+= a, then c will be 7,
  - And if we write c += 2, then c will be 4.

Assignment Operator

- **-= (Subtract AND Assignment) operator**: This operator subtract the operand on it's right from the left operand and assign the result to the operand on it's left.
  - Let us consider a = 5, b = 3, c = 9
  - If we write c -= a + b, then c will be 1,
  - If we write c-= a, then c will be 4,
  - And if we write c -= 2, then c will be 7.

Assignment Operator

- ***= (Multiply AND Assignment) operator**: This operator Multiply the operand on it's both sides and assign the result to the operand on it's left.
  - Let us consider a = 5, b = 3, c = 2
  - If we write c *= a + b, then c will be 16,
  - If we write c*= a, then c will be 10,
  - And if we write c *= 3, then c will be 6.

Assignment Operator

- **/= (Divide AND Assignment) operator**: This operator divide the operand on the left with the right operand and assign the quotient to the operand on it's left.
  - Let us consider a = 2, b = 3, c = 10
  - If we write c /= a + b, then c will be 2,
  - If we write c/= a, then c will be 5,
  - And if we write c /= 5, then c will be 2.

Assignment Operator

- **%= (Modulus AND Assignment) operator**: This operator divide the operand on the left with the right operand and assign the remainder to the operand on it's left.
  - Let us consider a = 2, b = 3, c = 10
  - If we write c %= a + b, then c will be 0,
  - If we write c%= a, then c will be 0,
  - And if we write c %= 3, then c will be 1.

Assignment Operator

- **<<= (Left Shift AND Assignment) operator**: This operator shift bits of the left operand to left by the value of right operand and assign the result to the operand on it's left.
  - Let us consider a = 2, b = 1
  - If we write a<<=b, then a will be 4,
  - If we write b<<=3, then b will be 8.

Assignment Operator

- **>>= (Right Shift AND Assignment) operator**: This operator shift bits of the left operand to right by the value of right operand and assign the result to the operand on it's left.
  - Let us consider a = 2, b = 1
  - If we write a>>=b, then a will be 1,
  - If we write a>>=2, then b will be 0.

Assignment Operator

- **&= (Bitwise AND and Assignment) operator**: This operator performs the AND operation on bits of the left operand with right operand and assign the result to the operand on it's left.
  - Let us consider a = 5, b = 3
  - If we write a&=b, then a will be 1,
  - If we write a&=2, then b will be 0.

Assignment Operator

- **|= (Bitwise OR and Assignment) operator**: This operator performs the OR operation on bits of the left operand with right operand and assign the result to the operand on it's left.
  - Let us consider a = 5, b = 3
  - If we write a|=b, then a will be 7,
  - If we write a|=2, then b will be 7.

Assignment Operator

- **^= (Bitwise XOR and Assignment) operator**: This operator performs the XOR operation on bits of the left operand with right operand and assign the result to the operand on it's left.
  - Let us consider a = 5, b = 3
  - If we write a^=b, then a will be 6,
  - If we write a^=2, then b will be 7.

Misc. Operator

- sizeof Operator
- & (referencing\Address) Operator
- * (dereferencing) Operator
- ? : (Ternary Operator)

Misc. Operator

- sizeof Operator: This operator takes operand(variable name/data type keyword) as parameter and return the memory size consume by the operand in bytes.
  - Let us assume a as integer variable
  - Then sizeof(a) will return 4 (for 32bit C compilers) or 2(for 16bit C compilers).
  - Similarly sizeof(float) will return 4, sizeof(char) will return 1 etc.

Misc. Operator

- **& (referencing/address) Operator**: It uses only one operand on it's right and it returns the memory address of the operand as unsigned integer data type.
  - printf("%u", &a); will display 6422316(may be different on different machine & every execution) on the console.
  - printf("%x", &a); will display 0x61FF2C(may be different on different machine & every execution) on the console.
- *** (dereferencing) Operator**: It is pointer to the variable. This operator used to declare the pointer variable & extracting the value stored at the address stored in pointer variable.
  - Let us assume a = 10, *b = &a,
  - if we write c = *b then c will be 10.

Misc. Operator

- **? : (Conditional) Operator**: It requires three expressions, therefore
  it is also called as Ternary operator.
  - 1. Conditional expression
  - 2. Expression to execute if condition is true
  - 3. Expression to execute if condition is false
- Syntax: (condition) ? (Expression for true) : (expression for false);
- Let us assume a = 15, b = 10
- If c = (a > b) ? a : b; then value of c will be 15
- If c = (a < b) ? a : b; then value of c will be 10

Thanks and hope you enjoy the session