**Operators**

**Operators** are used to perform operations. Operators are the symbols which perform the operation on the some values. These values are known as operands. There are following types of operators to perform different types of operations in C language:

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Assignment Operators
- Bitwise Operators
- Misc Operators

**1. Arithmetic Operators**

| **Operator** | **Operator Name** | **Description** | **Example** |
|---|---|---|---|
| + | Addition | Adds two operands | I = 40, J= 20I + J = 60 |
| – | Subtraction | Subtracts second operand from the first | I = 40, J= 20I – J = 20 |
| * | Multiplication | Multiplies both operands | I = 40, J= 20I * J = 800 |
| / | Divide | Perform division operation | I = 40, J= 20I / J = 2 |
| % | Modulus | Return the remainder after Division | I = 40, J= 20I % J = 0 |
| ++ | Increment | Increase the operand value by 1 | I=40,I++ = 41, ++I = 40 (print 40 but next time its value is 41) |
| — | Decrement | Decrease the operand value by 1 | I=40I– = 39, –I = 40<br>(print 40 but next time its value is 39) |

**2. Relational Operators**

It is also known as comparison operator because it compares the values. After comparison it returns the Boolean value i.e. either true or false.

| **Operator** | **Operator Name** | **Description** | **Example** |
|---|---|---|---|
| == | Equal to | If the values of two operands are equal then it returns true. | I = 20, J =20(I == J) is true |
| != | Not Equal to | If the values of two operands are not equal then it returns true. | I = 20, J =20(I == J) is False |
| < | Less than | If the value of left operand is less than the value of right operand then it returns true | I = 40, J =20(I < J) is False |
| > | Greater than | If the value of left operand is greater than the value of right operand then it returns true | I = 40, J =20(I > J) is True |

| | | | |
|---|---|---|---|
| <= | Less than or equal to | If the value of left operand is less than or equal to the value of right operand then it returns true. | I = 40, J =20(I <= J) is False |
| >= | Greater than or equal to | If the value of left operand is greater than or equal to the value of right operand then it returns true. | I = 40, J =20(I >= J) is True |

**3. Logical Operators**

| Operator | Operator Name | Description | Example |
|---|---|---|---|
| and | Logical AND | When Both side condition is true the result is true otherwise false | 2<1 and 2<3False |
| or | Logical OR | When at least one condition is true then result is true otherwise false | 2<1 or 2<3True |
| not | Logical NOT | Reverse the condition | Not(5>4)False |

**4. Bitwise Operators**

It performs bit by bit operation. Suppose there are two variable I = 10 and J = 20 and their binary values are

I = 10 = 0000 1010

J = 20 = 0001 0100

| Operator | Operator Name | Description | Example |
|---|---|---|---|
| & | Binary AND | If both bits are 1 then 1 otherwise 0 | I & J0000 0000 |
| \| | Binary OR | If one of the bit is 1 then 1 otherwise 0 | I \| J0001 1110 |
| ^ | Binary XOR | If both bit are same then 0 otherwise 1 | I ^ J0001 1110 |
| ~ | Binary Complement | If bit is 1 the make it 0 and if bit is 0 the make it 1 | ~I1111 0101 |
| << | Binary Left Shift | The left operand is moved left by the number of bits specified by the right operand. | I << 2 will give 240 i.e. 1111 0000 |
| >> | Binary Right Shift | The left operand is moved right by the number of bits specified by the right operand. | I >> 2 will give 15 i.e. 1111 |

**5. Assignment Operators**

| Operator | Operator Name | Description | Example |
|---|---|---|---|
| = | Assignment | It assigns value from right side operand to left side operand | I = 40It assigns 40 to I |
| += | Add then assign | It performs addition and then result is assigned to left hand operand | I+=Jthat means I = I + J |
| -= | Subtract then assign | It performs subtraction and then result is assigned to left hand operand | I-=Jthat means I = I – J |
| *= | Multiply the assign | It performs multiplication and then result is assigned to left hand operand. | I*=Jthat means I = I * J |
| /= | Divide then assign | It performs division and then result is assigned to left hand operand | I/=Jthat means I = I / J |

| | | | |
|---|---|---|---|
| %= | Modulus then assign | It performs modulus and then result is assigned to left hand operand | I%=Jthat means I = I % J |
| <<= | Left shift AND assignment operator | It performs Binary left shift and then result is assigned to left hand operand | I<<=5that means I = I << 5 |
| >>= | Right shift AND assignment operator | It performs Binary right shift and then result is assigned to left hand operand | I>>=5that means I = I >>=5 |
| &= | Bitwise AND assignment operator | It performs bitwise AND and then result is assigned to left hand operand | I &= 5that means I = I & 5 |
| ^= | bitwise exclusive OR and assignment operator | It performs bitwise exclusive OR and then result is assigned to left hand operand | I ^= 5that means I = I ^ 5 |
| \|= | bitwise inclusive OR and assignment operator | It performs bitwise inclusive OR and then result is assigned to left hand operand | I \|= 5that means I = I \| 5 |

**6. Misc Operators**

There are few other important operators including sizeof and **? :** supported by C Language.

| **Operator** | **Description** |
|---|---|
| sizeof() | Returns the size of an variable. |
| & | Returns the address of an variable. |
| * | Pointer to a variable. |
| ? : | Conditional Expression |

**Operators Precedence in C**

| **Category** | **Operator** | **Associativity** |
|---|---|---|
| Postfix | () [] -> . ++ – – | Left to right |
| Unary | + – ! ~ ++ – – (type)* & sizeof | Right to left |
| Multiplicative | * / % | Left to right |
| Additive | + – | Left to right |
| Shift | << >> | Left to right |
| Relational | < <= > >= | Left to right |
| Equality | == != | Left to right |
| Bitwise AND | & | Left to right |
| Bitwise XOR | ^ | Left to right |
| Bitwise OR | \| | Left to right |
| Logical AND | && | Left to right |
| Logical OR | \|\| | Left to right |
| Conditional | ?: | Right to left |
| Assignment | = += -= *= /= %=>>= <<= &= ^= \|= | Right to left |
| Comma | , | Left to right |