Operators
&
Expressions in C

Types of Operators

- Arithmetic
- Assignment
- Increment/ Decrement
- Relational/ Comparison
- Logical
- Conditional
- Sizeof
- Other Operators

Integer Arithmetic

```
#include<stdio.h>
main( )
{
    int  a=17,b=4;
    printf("Sum  =  %d\n",a+b);
    printf("Difference  =  %d\n",a-b);
    printf("Product  =  %d\n",a*b);
    printf("Quotient  =  %d\n",a/b);
    printf("Remainder  =  %d\n",a%b);
}
```

Output:

```
Sum = 21
Difference = 13
Product = 68
Quotient = 4
Remainder = 1
```

Floating-Point Arithmetic

```
#include<stdio.h>
main  (  )
{
    float  a=12.4,b=3.8;
    printf("Sum  =  %.2f\n",a+b);
    printf("Difference  =  %.2f\n",a-b);
    printf("Product  =  %.2f\n",a*b);
    printf("a/b  =  %.2f\n",a/b);
}
```

Output

```
Sum = 16.20
Difference = 8.60
Product = 47.12
a/b  = 3.26
```

Assignment Operator

x -=5	x = x - 5
y*=5	y = y*5
sum /=5	sum = sum / 5
k%= 5	k = k % 5

Increment/ Decrement Operator

++x	x = x + 1
--x	x = x - 1

-These operators can be used only with variables
-They can't be used with constants or expressions.
-Example: ++5 or ++(x+y+z) are invalid.

Pre-Increment/ Pre-Decrement

- First the value of variable is incremented / decremented then the new value is used in the operation
- Let us take a variable x whose value is 3.
- The statement y = ++x; means first increment the value of x by 1, then assign the value of x to y
- This single statement is equivalent to these two statements

x= x+1;

y = x;

- Now value of x is 4 and value of y is 4

```
#include<stdio.h>
main( )
{
    int  x=8;
    printf("x  =  %d\t",x);
    printf("x  =  %d\t",++x);
    printf("x  =  %d\t",x);
    printf("x  =  %d\t",- -x);
    printf("x  =  %d\n",x);
}
```

Output:

x = 8 x = 9 x = 9 x = 8 x = 8

Post-Increment/ Post-Decrement

```
#include<stdio.h>
main( )
{
    int  x=8;
    printf("x  =  %d\n",x);
    printf("x  =  %d\t",x++);
    printf("x  =  %d\t",x);
    printf("x  =  %d\t",x- -);
    printf("x  =  %d\n",x);
}
```

Output:

x = 8 x = 8 x = 9 x = 9 x = 8

Relational Operator

- Let a = 9 and b = 5

Expression	Relation	Value of Expression
a < b	False	0
a <= b	False	0
a= =b	False	0
a != b	True	1
a > b	True	1
a >= b	True	1
a = = 0	False	0
b!=0	True	1
a>8	True	1
2 > 4	False	0

```c
#include<stdio.h>
main( )
{
    int a,b;
    printf("Enter values for a and b : ");
    scanf("%d%d",&a,&b);
    if(a<b)
        printf("%d is less than %d\n",a,b);
    if(a<=b)
        printf("%d is less than or equal to %d\n",a,b);
    if(a==b)
        printf("%d is equal to %d\n",a,b);
    if(a!=b)
        printf("%d is not equal to %d\n",a,b);
    if(a>b)
        printf("%d is greater than %d\n",a,b);
    if(a>=b)
        printf("%d is greater than or equal to %d\n",a,b);
}
```

Output:

```
Enter values for a and b :  12  7
12 is not equal to 7
12 is greater than 7
12 is greater than or equal to 7
```

Logical Operator

Operator	Meaning
&&	AND
||	OR
!	NOT

Let a = 10, b = 5, c = 0

Expression		Result	Value of expression
(a= =10) && (b>a)	true && false	false	0
(b>=a) && (b= =3)	false && false	false	0
a && b	true && true	true	1
a && c	true && false	false	0

Let a = 10, b = 5, c = 0

Expression		Result	Value of expression
a || b	true || true	true	1
a || c	true || false	true	1
(a<9) || (b>10)	false|| false	false	0
(b!=7) || c	true || false	true	1

Expression		Result	Value of expression
!a	!true	false	0
!c	!false	true	1
!(b>c)	!true	false	0
!(a && c)	!false	true	1

Conditional Operator (? and:)

- It is a ternary operator which requires three expressions as operands.
- This is written as- TestExpression ? expression1 : expression2
- First the TestExpression is evaluated.
- If TestExpression is true(nonzero), then expression1 is evaluated and it becomes the value of the overall conditional expression.
- If TestExpression is false(zero), then expression2 is evaluated and it becomes the vaJue of overall conditional expression.

Example1: Let a = 5 and b = 8,

```
max = a > b ? a : b;
```

So, the value of max becomes 8

Print the larger of two numbers using conditional operator

```
#include<stdio.h>
main( )
{
    int  a,b,max;
    printf("Enter  values  for  a  and  b  :  ");
    scanf("%d  %d",&a,&b);
    max  =  a>b  ?  a:b;          /*ternary  operator*/
    printf("Larger  of  %d  and  %d  is  %d\n",a,b,max);
}
```

Output:

```
Enter values for a and b : 12  7
Larger of 12 and 7 is 12
```

sizeof Operator

- It is an unary operator
- Gives the size of its operand in terms of bytes.
- Operand can be a variable, constant or any datatype(int, float, char etc).

```
#include<stdio.h>
main( )
{
    int  var;
    printf ("Size of int = %d",sizeof(int));
    printf("Size of float = %d",sizeof(float));
    printf("Size of var = %d",sizeof(var));
    printf("Size of an integer constant = %d",sizeof(45));
}
```

Type conversion (implicit)

```
#include<stdio.h>
Void main()
{
    char c1,c2;
    int i1,i2;
    float f1, f2;
    c1='H';
    i1=80. 56;          /*float converted to int, only 80 assigned to i1 * /
    f1=12.6;
    c2=i1; ;        /*int converted to char*/
    i2=f1;          /*float converted to int */
/*Now c2 has character with ASCII value 80, i2 is assig
printf("c2 = %c, i2 = %d\n", c2,i2);
```

Output:

c2 = P, i2 = 12
f2 = 80.00, i2 = 72

Type conversion (explicit)

float z;

int x = 20, y = 3;

z = x/y;

- The value of z will be 6.0 instead of 6.66
- z= (float)x/y;

```
#include<stdio.h>
main( )
{
    int  x=5,y=2;
    float  p,q;
    p=x/y;
    printf("p  =  %f\n",p);
    q=(float)x/y;
    printf("q  =  %f\n",q);
}
```

Output:

```
p = 2.000000
q = 2.500000
```

Precedence & Associativity

Operator	Description	Precedence level	Associativity
( )	Function call	1	Left to Right
[ ]	Array subscript		
→	Arrow operator		
.	Dot operator		
+	Unary plus	2	Right to Left
-	Unary minus		
++	Increment		
- -	Decrement		
!	Logical NOT		
~	One's complement		
*	Indirection		
&	Address		
(datatype)	Type cast		
sizeof	Size in bytes		

*	Multiplication	3	Left to Right
/	Division		
%	Modulus		
+	Addition	4	Left to Right
-	Subtraction		
<<	Left shift	5	Left to Right
>>	Right shift		
<	Less than	6	Left to Right
<=	Less than or equal to		
>	Greater than		
>=	Greater than or equal to		
= =	Equal to	7	Left to Right
!=	Not equal to		

&	Bitwise AND	8	Left to Right
^	Bitwise XOR	9	Left to Right
|	Bitwise OR	10	Left to Right
&&	Logical AND	11	Left to Right
||	Logical OR	12	Left to Right
? :	Conditional operator	13	Right to Left
= *= /= %= += -= &= ^= |= <<= >>=	Assignment operators	14	Right to Left
,	Comma operator	15	Left to Right