



Operators And Expression

Dr. Sheak Rashed Haider Noori
Professor & Associate Head
Department of Computer Science

Operators and Expressions

- Consider the expression $A + B * 5$, where,
 - $+$, $*$ are **operators**,
 - A , B are **variables**,
 - 5 is **constant**,
 - A , B and 5 are called **operand**, and
 - $A + B * 5$ is an **expression**.

$(a+b) * c$	Operator is $*$, operands are $(a+b)$ and c
$(a+b)$	Operator is $()$, operand is $a+b$
$a+b$	Operator is $+$, operands are a and b

Types of C operators



- C language offers many types of operators, such as:
 - Arithmetic operators
 - Assignment operators
 - Increment/decrement operators
 - Relational operators
 - Logical operators
 - Bit wise operators
 - Conditional operators (ternary operators)
 - Special operators

Arithmetic Operators

Arithmetic operators
Assignment operators
Inc/dec operators
Relational operators
Logical operators
Bit wise operators
Conditional operators
Special operators

- C Arithmetic operators are used to perform mathematical calculations like addition, subtraction, multiplication, division and modulus in C programs.

Arithmetic Operators	Operation	Example
+	Addition	A+B
-	Subtraction	A-B
*	multiplication	A*B
/	Division	A/B
%	Modulus	A%B

Arithmetic Operators

Arithmetic operators
Assignment operators
Inc/dec operators
Relational operators
Logical operators
Bit wise operators
Conditional operators
Special operators

- There are three types of arithmetic operations using arithmetic operators:
 1. **Integer arithmetic** : when all operands are integer. If $a=15$, $b=10$,
 - $a + b = 25$
 - $a / b = 1$ (decimal part)
 - $a \% b = 5$ (remainder of division)
 2. **Real arithmetic** : All operands are only real number. If $a=15.0$, $b=10.0$
 - $a / b = 1.5$
 3. **Mixed model arithmetic** : when one operand is real and another is integer. If $a=15$ and $b= 10.0$
 - $a / b = 1.5$ whereas, $15/10=1$
- Note: The modulus operator $\%$ gives you the remainder when two integers are divided: $1 \% 2$ is 1 and $7 \% 4$ is 3.
- The modulus operator can only be applied to integers.

Arithmetic Operators

Arithmetic operators

Assignment operators

Inc/dec operators

Relational operators

Logical operators

Bit wise operators

Conditional operators

Special operators

1. Integer arithmetic :

- When an arithmetic operation is performed on two whole numbers or integers than such an operation is called as integer arithmetic.
- It always gives an integer as the result.
- Let $x = 27$ and $y = 5$ be 2 integer numbers. Then the integer operation leads to the following results.

$$\blacksquare x + y = 32$$

$$\blacksquare x - y = 22$$

$$\blacksquare x * y = 115$$

$$\blacksquare x \% y = 2$$

$$\blacksquare x / y = 5$$

Example program for C arithmetic operators

Arithmetic operators

Assignment operators
Inc/dec operators
Relational operators
Logical operators
Bit wise operators
Conditional operators
Special operators

```
#include <stdio.h>
int main()
{
    int a=40,b=20, add,sub,mul,div,mod;

    add = a+b;
    sub = a-b;
    mul = a*b;
    div = a/b;
    mod = a%b;

    printf("Addition of a, b is : %d\n", add);
    printf("Subtraction of a, b is : %d\n", sub);
    printf("Multiplication of a, b is : %d\n", mul);
    printf("Division of a, b is : %d\n", div);
    printf("Modulus of a, b is : %d\n", mod);
}
```

Output:

Addition of a, b is : 60
Subtraction of a, b is : 20
Multiplication of a, b is : 800
Division of a, b is : 2
Modulus of a, b is : 0

Arithmetic Operators

Arithmetic operators
Assignment operators
Inc/dec operators
Relational operators
Logical operators
Bit wise operators
Conditional operators
Special operators

2. Real arithmetic :

- When an arithmetic operation is performed on two real numbers or fraction numbers such an operation is called real or floating point arithmetic.
- The modulus (remainder) operator is not applicable for floating point arithmetic operands.
- Let $x = 14.0$ and $y = 4.0$ then
 - $x + y = 18.0$
 - $x - y = 10.0$
 - $x * y = 56.0$
 - $x / y = 3.50$

Arithmetic Operators

Arithmetic operators
Assignment operators
Inc/dec operators
Relational operators
Logical operators
Bit wise operators
Conditional operators
Special operators

3. Mixed mode arithmetic :

- When one of the operand is real and other is an integer and if the arithmetic operation is carried out on these 2 operands then it is called as mixed mode arithmetic.
- If any one operand is of real type then the result will always be real
- Let $x = 15$ and $y = 10.0$ then
 - $x / y = 1.5$
- Note that: $15 / 10 = 1$ (since both of the operands are integer)

Assignment Operators

- In C programs, values for the variables are assigned using assignment operators.
- For example, if the value “10” is to be assigned for the variable “sum”, it can be assigned as **sum = 10;**

Arithmetic operators
Assignment operators
Inc/dec operators
Relational operators
Logical operators
Bit wise operators
Conditional operators
Special operators

Operators		Example	Explanation
Simple assignment operator	=	sum=10	10 is assigned to variable sum
Shorthand or Compound assignment operators	+=	sum+=10	This is same as sum=sum+10
	-=	sum-=10	This is same as sum = sum-10
	=	sum=10	This is same as sum = sum*10
	/=	sum/=10	This is same as sum = sum/10
	%=	sum%=10	This is same as sum = sum%10
	&=	sum&=10	This is same as sum = sum&10
	^=	sum^=10	This is same as sum = sum^10

Increment and Decrement Operators

Arithmetic operators
Assignment operators
Inc/dec operators
Relational operators
Logical operators
Bit wise operators
Conditional operators
Special operators

- There are two more shorthand operators:
 - Increment **++**
 - Decrement **--**
- These two operators are for incrementing and decrementing a variable by 1.
- For example, the following code increments *i* by 1 and decrements *j* by 1.

```
int i = 3, j = 3;  
i++; // i becomes 4  
j--; // j becomes 2
```

Increment and Decrement Operators

Arithmetic operators
Assignment operators
Inc/dec operators
Relational operators
Logical operators
Bit wise operators
Conditional operators
Special operators

- The ++ and - - operators can be used in **prefix** or **suffix** mode, as shown in Table

Operator	Name	Description	Example (assume $i = 1$)
++var	preincrement	Increment var by 1 and use the new var value	int j = ++i; // j is 2, // i is 2
var++	postincrement	Increment var by 1 , but use the original var value	int j = i++; // j is 1, // i is 2
--var	predecrement	Decrement var by 1 and use the new var value	int j = --i; // j is 0, // i is 0
var--	postdecrement	Decrement var by 1 and use the original var value	int j = i--; // j is 1, // i is 0

Increment and Decrement Operators

Arithmetic operators
Assignment operators
Inc/dec operators
Relational operators
Logical operators
Bit wise operators
Conditional operators
Special operators

- If the operator is *before* (prefixed to) the variable, the variable is incremented or decremented by 1, then the *new* value of the variable is returned.
- If the operator is *after* (suffix to) the variable, then the variable is incremented or decremented by 1, but the original *old* value of the variable is returned.
- Therefore, the prefixes ++x and --x are referred to, respectively, as the *preincrement* operator and the *predecrement* operator; and the suffixes x++ and x-- are referred to, respectively, as the *postincrement* operator and the *postdecrement* operator.

Increment and Decrement Operators

Arithmetic operators
Assignment operators
Inc/dec operators
Relational operators
Logical operators
Bit wise operators
Conditional operators
Special operators

The prefix form of ++ (or --) and the suffix form of ++ (or --) are the same if they are used in isolation, but they cause different effects when used in an expression. The following code illustrates this:

```
int i = 10;  
int newNum = 10 * i++;
```

Same effect as

```
int newNum = 10 * i;  
i = i + 1;
```

In this case, *i* is incremented by 1, then the *old* value of *i* is returned and used in the multiplication. So *newNum* becomes **100**. If *i++* is replaced by *++i* as follows,

```
int i = 10;  
int newNum = 10 * (++i);
```

Same effect as

```
i = i + 1;  
int newNum = 10 * i;
```

i is incremented by 1, and the new value of *i* is returned and used in the multiplication. Thus *newNum* becomes **110**.

Exercise on ++ and - -

Arithmetic operators
Assignment operators
Inc/dec operators
Relational operators
Logical operators
Bit wise operators
Conditional operators
Special operators

■ `int x=2 , y = 5;`

1. `x++ ; y++ ;`
2. `x=y++ + x++;`
3. `y=++y + ++x;`
4. `y=++y + x++;`
5. `y += ++y;`
6. `y += 1 + (++x);`
7. `y += 2 + x++;`

Relational Operators

- Relational operators are used to find the relation between two variables. i.e. to compare the values of two variables.

S.no	Operators	Example	Description
1	>	$x > y$	x is greater than y
2	<	$x < y$	x is less than y
3	>=	$x >= y$	x is greater than or equal to y
4	<=	$x <= y$	x is less than or equal to y
5	==	$x == y$	x is equal to y
6	!=	$x != y$	x is not equal to y

Exercise

■ `int i=10, j=20, k=30;`

■ `float f=5.5;`

■ `char ch='A';`

1) `i < j`

2) `(j+k)>=(i+k)`

3) `i+f <=10`

4) `i+(f <=10)`

5) `ch==65`

6) `ch >= 10*(i+f)`

Logical Operators

Arithmetic operators
Assignment operators
Inc/dec operators
Relational operators
Logical operators
Bit wise operators
Conditional operators
Special operators

- These operators are used to perform logical operations on the given expressions.
- There are 3 logical operators in C language. They are, logical AND (&&), logical OR (||) and logical NOT (!).

S.no	Operators	Name	Example	Description
1	&&	logical AND	<code>(x>5)&&(y<5)</code>	It returns true when both conditions are true
2		logical OR	<code>(x>=10) (y>=10)</code>	It returns true when at-least one of the condition is true
3	!	logical NOT	<code>!((x>5)&&(y<5))</code>	It reverses the state of the operand " <code>((x>5) && (y<5))</code> " If " <code>((x>5) && (y<5))</code> " is true, logical NOT operator makes it false

Logical Operators



Operator Operands	&&	
Zero and Zero	0	0
Zero and Nonzero	0	1
Nonzero and zero	0	1
Nonzero and Nonzero	1	1

Logical AND & Logical OR

Input (A)	Output (!A)
Zero	1
Nonzero	0


Logical NOT

Exercise

- Given that: `int a = 5, b = 2, c = 4, d = 6, e = 3 ;`
What is the result of each of the following relational expressions?

1. `a > b`
2. `a != b`
3. `d % b == c % b`
4. `a * c != d * b`
5. `d * b == c * e`
6. `a * b < a % b * c`
7. `c % b * a == b % c * a`
8. `b % c * a != a * b`
9. `d % b * c > 5 || c % b * d < 7`
10. `d % b * c > 5 && c % b * d < 7`

Exercise

- 
- For each of the following statements, assign variable names for the unknowns and rewrite the statements as relational expressions.
 1. A customer's age is 65 or more.
 2. The temperature is less than 0 degrees and greater than -15 degrees.
 3. A person's height is in between 5.8 to 6 feet.
 4. The current month is 12 (December).
 5. The person's age is 65 or more but less than 100.
 6. A number is evenly divided by 4 or 400 but not with 100
 7. A person is older than 55 or has been at the company for more than 25 years.
 8. A width of a wall is less than 4 meters but more than 3 meters.
 9. An employee's department number is less than 500 but greater than 1, and they've been at the company more than 25 years.

Example program for logical operators in C

Arithmetic operators
Assignment operators
Inc/dec operators
Relational operators
Logical operators
Bit wise operators
Conditional operators
Special operators

```
#include <stdio.h>
int main()
{
    int m=40,n=20;
    int o=20,p=30;

    if (m>n && m !=0)
    {
        printf("&& Operator : Both conditions are true\n");
    }
    if (o>p || p!=20)
    {
        printf("|| Operator : Only one condition is true\n");
    }
    if (!(m>n && m !=0))
    {
        printf("! Operator : Both conditions are true\n");
    }
    else
    {
        printf("! Operator : Both conditions are true. " \
            "But, status is inverted as false\n");
    }
}
```

Output:

&& Operator : Both conditions are true

|| Operator : Only one condition is true

! Operator : Both conditions are true. But, status is inverted as false

Try this example program
and explain the results



```
#include<stdio.h>
int main()
{
    int a=5, b=-7, c=0, d;
    d = ++a && ++b || ++c;
    printf("\n %d %d %d %d",a,b,c,d);
}
```

Bit wise Operators

Arithmetic operators
Assignment operators
Inc/dec operators
Relational operators
Logical operators
Bit wise operators
Conditional operators
Special operators

- One of C's powerful features is a set of bit manipulation operators. These permit the programmer to access and manipulate individual bits within a piece of data to perform bit operations. The various Bitwise Operators available in C are shown in Figure
- Decimal values are converted into binary values which are the sequence of bits and bit wise operators work on these bits.
- These operators can operate upon **ints** and **chars** but not on **floats** and **doubles**.

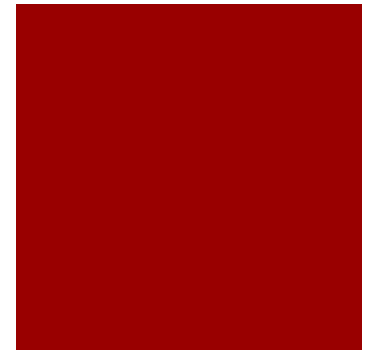
Operator	Meaning
~	One's complement
>>	Right shift
<<	Left shift
&	Bitwise AND
	Bitwise OR
^	Bitwise XOR(Exclusive OR)

Special Operators

Arithmetic operators
Assignment operators
Inc/dec operators
Relational operators
Logical operators
Bit wise operators
Conditional operators
Special operators

S.no	Operators	Description
1	&	This is used to get the address of the variable. Example : &a will give address of a.
2	*	This is used as pointer to a variable. Example : * a where, * is pointer to the variable a.
3	Sizeof ()	This gives the size of the variable. Example : size of (char) will give us 1.

Example program for Special operators in C



```
#include <stdio.h>

int main()
{
    int *ptr, q;
    q = 50;
    /* address of q is assigned to ptr */
    ptr = &q;
    /* display q's value using ptr variable */
    printf("%d", *ptr);
    return 0;
}
```

Example program for sizeof() operator in C



```
#include <stdio.h>
#include <limits.h>
```

```
int main()
{
    int a;
    char b;
    float c;
    double d;
    printf("Storage size for int data type:%d \n", sizeof(a));
    printf("Storage size for char data type:%d \n", sizeof(b));
    printf("Storage size for float data type:%d \n", sizeof(c));
    printf("Storage size for double data type:%d\n", sizeof(d));
    return 0;
}
```

Output:

Storage size for int data type:4
Storage size for char data type:1
Storage size for float data type:4
Storage size for double data type:8