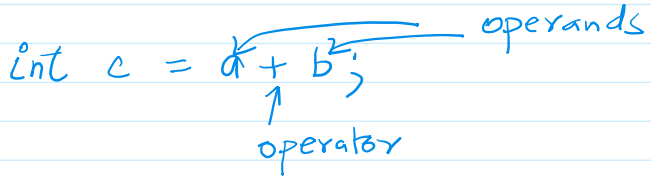
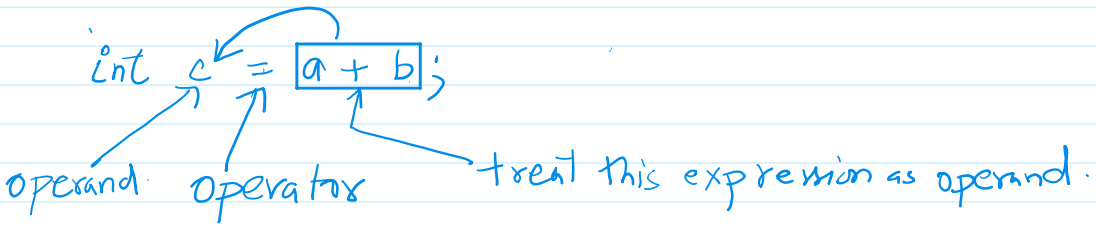
Operators in C language

07 February 2026 20:05

What in an Operator?
An operator in C is a symbol that performs a specific operation on one or more operands (variables, constants, or expressions.)





Classification of operators in C:
C operators are broadly classified into following categories:

1. Arithmetic operators (+, -, *, /(quotient), %(remainder)).
2. Relational operators (>, <, >=, <=, ==(equality comparison), != (not equal to)).
3. Logical operators (&&(AND), ||(Or), !(Not)) It is used to combine multiple conditions.
4. Assignment operators (=, +=, -=, *=, /=, %=) used to assign values to the variable.
   int x = 5;
   x += 4; //9
5. Increment and decrement operators: Used to increase or decrease a value by 1.(++, --)
   a. Pre-increment or Pre-decrement: We need to write the operator first and then operand. Here the value of the variable updated first and then it is used in the expression.
   b. Post-increment or Post-decrement: We need to write the operand first and then operator. Here the value of the variable is used first in the expression and then it is updated.

one.c

```c
#include<stdio.h>
int main(int argc, char const *argv[])
{
    int x = 5, y = 2;
    int z = ++x + --y;
    print("z = %d", z);
    return 0;
}
```

argument count

array which stores arguments of this function.

C-app programs starts with main() function

```
if (argv[0] == __) {
    →
    →
    →
} else if (argv[1] == __ ) {
```

```
gcc one.c
./one arg1 arg2 arg3
./app user_name password
```

} else if (argv[1] == '--') {
→
→
→
}

```c
#include<stdio.h>
int main(int argc, char const *argv[])
{
    int x = 5, y = 2;
    int z = ++x + --y;
    print("z = %d", z);
    return 0;
}
```

x = 5 6
y = 2 1
z = 7

z = ++x + --y
    6   + 1

Output:
z = 7

```c
#include<stdio.h>
int main(int argc, char const *argv[])
{
    int x = 5, y = 2;
    int z = ++x + --y;
    printf("z = %d", z);
→   z = x++ + y++;
    printf("z = %d", z);
    return 0;
}
```

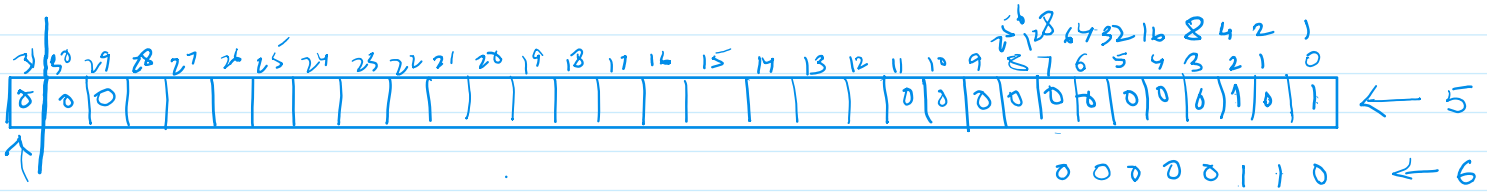
x = 6 7
y = 1 2
z = 7 7
z = x++ + y++;
    6   + 1

Output:
z = 7
z = 7

6. Bitwise operators: Used to perform operations at the bit level.

Operator	Meaning
&	Bitwise AND
~	Bitwise Not
^	Bitwise XOR
<<	Left Shift
>>	Right Shift

int x = 5;
↑
size of int data-type. 4 bytes = 32 bits
int y = 6;



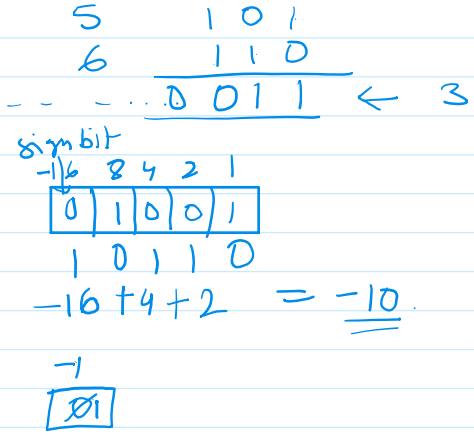
sign bit 0 → +ve
1 → -ve

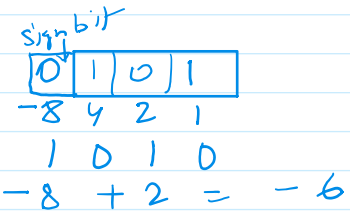
```
z = (x & y); // 4
z = (x | y); // 7
z = ~x;
```

```
#include<stdio.h>
int main(int argc, char const *argv[])
{
    int x = 5, y = 6;
    int z = x & y;
    printf("z = %d", z);
    return 0;
}
```

Output:
z = 4

$2^{32} - 1$





7. Conditional or ternary operators: The only operator that takes 3 operands
Syntax:
Variable = (condition)? Expression1: Expression2;

```
#include<stdio.h>
int main(int argc, char const *argv[])
{
    int x, y, z, max, min;
    printf("Input three int values: ");
    scanf("%d%d%d", &x, &y, &z);
    max = (x > y && x > z)? x : (y > z)? y : z;
    min = (x < y && x < z)? x : (y < z)? y : z;
    printf("\nMaximum value = %d", max);
    printf("\nMinimum value = %d", min);
    return 0;
```

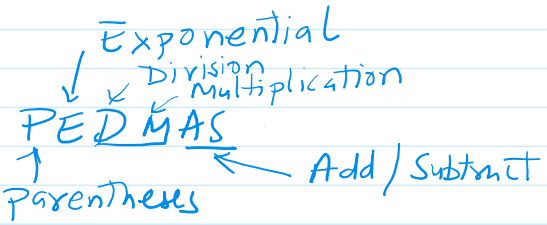
comma operator
address of operator
logical operator

```
}
```

```
Output:
   Input three int values: 15 12 3

   Maximum value = 15
   Minimum value = 3
```

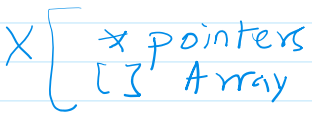
```c
#include<stdio.h>
int main(int argc, char const *argv[])
{
    int x, y, z, max, min;
    printf("Input three int values: ");
    scanf("%d%d%d", &x, &y, &z);
    max = (x > y && x > z)? x : (y > z)? y : z;
    min = (x < y && x < z)? x : (y < z)? y : z;
    printf("\nMaximum value = %d", max);
    printf("\nMinimum value = %d", min);
    printf("\nsize of int: %d bytes", sizeof(x));
    return 0;
}
```

```
Output:

   Input three int values: 21 14 19

   Maximum value = 21
   Minimum value = 14
   size of int: 4
```



Operator Precedence (high to low)

1. **Unary operators (++, --, !)**
2. **Arithmetic operators(PEDMAS rule) (*, /, %)**
3. **Arithmetic operators (+, -)**
4. **Relational operators (<, >, <=, >=)**
5. **Equality (==, !=)**
6. **Logical AND(&&)**
7. **Logical OR (||)**
8. **Assignment (=, +=, -=, *=, /=, %=)**
9. **Bitwise operator(&, |, ^, ~)**



Homework questions

Below are practice questions only (✗ no solutions included), strictly based on Operators in C as per your content.

A. Multiple Choice Questions (MCQs)

(Choose the correct option)

1. Which operator is used to find the remainder of a division in C?

a) /
b) %
c) *

d) //
2. Which of the following is a logical operator in C?
a) &
b) |
c) &&
d) ^
3. What is the output type of relational operators in C?
a) float
b) int
c) char
d) double
4. Which operator has the highest precedence?
a) +
b) *
c) ++
d) ==
5. Which operator is known as the ternary operator?
a) :
b) ?
c) ?:
d) &&
6. Which operator performs bitwise NOT operation?
a) &
b) ^
c) ~
d) !
7. Which of the following is an assignment operator?
a) ==
b) <=
c) +=
d) &&
8. What does the && operator do?
a) Bitwise AND
b) Logical AND
c) Logical OR
d) Bitwise OR
9. Which operator is used to shift bits to the left?
a) >>
b) <<
c) <>
d) ^^
10. Which operator compares two values for equality?
a) =
b) !=
c) ==
d) <=

B. Fill in the Blanks
1. An operator that takes three operands is called the _____ operator.
2. The operator used to increment a value by 1 is _____.
3. The symbol != represents the _____ operator.
4. Bitwise operators work at the _____ level.
5. The operator sizeof is an example of a _____ operator.
6. The logical OR operator in C is written as _____.
7. The expression x += 5 is equivalent to _____.

8. The operator used to perform bitwise XOR is _____.
9. Pre-increment operator updates the value _____ it is used in an expression.
10. The operator && returns _____ if all conditions are true.

C. True / False
1. The = operator is used for comparison in C.
2. The % operator cannot be used with floating-point numbers.
3. Relational operators always return true or false.
4. ++x and x++ always produce the same result in expressions.
5. Bitwise operators are used to manipulate individual bits.
6. The ternary operator requires exactly three operands.
7. Logical operators can be used to combine multiple conditions.
8. The precedence of * is higher than +.
9. The ~ operator is a logical NOT operator.
10. Assignment operators have the highest precedence.

D. Output-Based Questions
(Predict the output)

1.
```
int x = 5;
printf("%d", x++);
```
2.
```
int x = 5;
printf("%d", ++x);
```

3.
```
int a = 10, b = 3;
printf("%d", a % b);
```
4.
```
int x = 4, y = 6;
printf("%d", x & y);
```

5.
```
int x = 1;
printf("%d", !x);
```

6.
```
int x = 5, y = 2;
printf("%d", x > y && y < 1);
```

7.
```
int x = 10;
x += 5;
printf("%d", x);
```

8.
```
int x = 8;
printf("%d", x >> 1);
```

9.
```
int a = 5, b = 10;
int c = (a > b) ? a : b;
printf("%d", c);
```

10.
```
int x = 3;
printf("%d", sizeof(x));
```

Prefix and Postfix Operators – Practice Questions

1. Predict the output:
```
int x = 5;
int y = ++x + x++;
printf("%d %d", x, y);
```
2. Predict the output:

```
int a = 10;
int b = --a + a-- + a;
printf("%d %d", a, b);
3. What will be printed?
int x = 3;
printf("%d %d %d", x++, ++x, x);
4. Find the final values of x and y:
int x = 4;
int y = x++ + ++x;
5. Predict the output:
int a = 5;
int b = a++ + a++ + ++a;
printf("%d %d", a, b);
6. What is the output?
int x = 7;
printf("%d", x-- - --x);
7. Find the output:
int x = 2;
int y = ++x * x++;
printf("%d %d", x, y);
8. Predict the output:
int a = 6;
int b = a-- + --a + a++;
printf("%d %d", a, b);
9. What will be the output?
int x = 5;
int y = x++ + ++x + x++;
printf("%d %d", x, y);
10. Determine the output:
int x = 1;
printf("%d %d", ++x, x++);
```