1. Introduction to Operators

An **operator** in C language is a **symbol that performs a specific operation** on one or more operands and produces a result.

Operand

An operand is a variable, constant, or expression on which an operator works.

Example

```
int a = 10, b = 5;
c = a + b;
```

Here:

- `+` is an **operator**
- `a` and `b` are **operands**
- `c` stores the result

---

2. Classification of Operators in C

C language provides a rich set of operators. They are classified as:

1. Arithmetic Operators
2. Relational Operators
3. Logical Operators
4. Assignment Operators
5. Increment and Decrement Operators
6. Bitwise Operators
7. Conditional (Ternary) Operator
8. Special Operators

---

3. Arithmetic Operators

Arithmetic operators are used to perform **mathematical calculations**.

Operator	**Meaning**
+	Addition

Operator	**Meaning**
-	Subtraction
*	Multiplication
/	Division
%	Modulus

Example

```
int a = 10, b = 3;
printf("%d", a + b);   // 13
printf("%d", a % b);   // 1
```

Important Points

- % works only with integers
- Division of integers gives integer result

---

4. Relational Operators

Relational operators are used to **compare two values.**
The result is either **true (1)** or **false (0)**.

Operator	**Meaning**
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Example

```
int a = 10, b = 20;
printf("%d", a < b);   // 1
```

Relational operators are mostly used in **decision making and loops.**

---

5. Logical Operators

Logical operators are used to **combine multiple conditions**.

Operator	**Meaning**
&&	Logical AND
`	
!	Logical NOT

Truth Table

A	B	A && B	A || B
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Example

```
if(a > 0 && b > 0)
{
   printf("Both positive");
}
```

---

6. Assignment Operators

Assignment operators are used to **assign values** to variables.

Operator	**Meaning**
=	Simple assignment
+=	Add and assign
-=	Subtract and assign
*=	Multiply and assign
/=	Divide and assign
%=	Modulus and assign

Example

```
int a = 10;
```

```
a += 5;  // a = a + 5
```

7. Increment and Decrement Operators

These operators increase or decrease a value by **1**.

Operator	**Meaning**
++	Increment
--	Decrement

Types

1. **Pre-increment** (++a)
2. **Post-increment** (a++)

Example

```
int a = 5;
printf("%d", a++);  // 5
printf("%d", a);    // 6
```

8. Bitwise Operators

Bitwise operators work on **binary representation** of data.

Operator	**Meaning**
&	Bitwise AND
`	`
^	Bitwise XOR
~	Bitwise NOT
<<	Left shift
>>	Right shift

Example

```
int a = 5, b = 3;
printf("%d", a & b);   // 1
```

Bitwise operators are used in:

- Embedded systems
- Device drivers
- Low-level programming

---

9. Conditional (Ternary) Operator

This operator is a **short form of if–else**.

Syntax

```
condition ? expression1 : expression2;
```

Example

```
int max = (a > b) ? a : b;
```

---

10. Special Operators in C

10.1 sizeof Operator

Used to find the **size of data type or variable**.

```
printf("%d", sizeof(int));
```

10.2 Comma Operator

Allows multiple expressions.

```
int a = (b = 3, b + 2);
```

10.3 Pointer Operators

- & → Address of operator
- * → Value at address operator

```
int a = 10;
int *p = &a;
```

10.4 Structure Operator

Used to access structure members.

```
s.age;
```

---

11. Operator Precedence and Associativity

Operator precedence determines **which operator is evaluated first**.

Example

```
int x = 10 + 5 * 2;   // Result = 20
```

Associativity

Determines the **direction of evaluation** (left to right or right to left).

---

12. Operators Used in Expressions

Expressions can contain:

- Multiple operators
- Variables
- Constants

Example

```
result = (a + b) * c / d;
```

---

13. Advantages of Operators in C

- Make programs short and efficient
- Increase readability
- Improve performance
- Support low-level operations

---

14. Common Errors with Operators

- Using = instead of ==
- Integer division errors
- Misuse of increment operators
- Ignoring operator precedence

---

15. Conclusion

Operators are the **building blocks of C programs**. Understanding different types of operators, their precedence, and correct usage is essential for writing efficient and error-free programs.