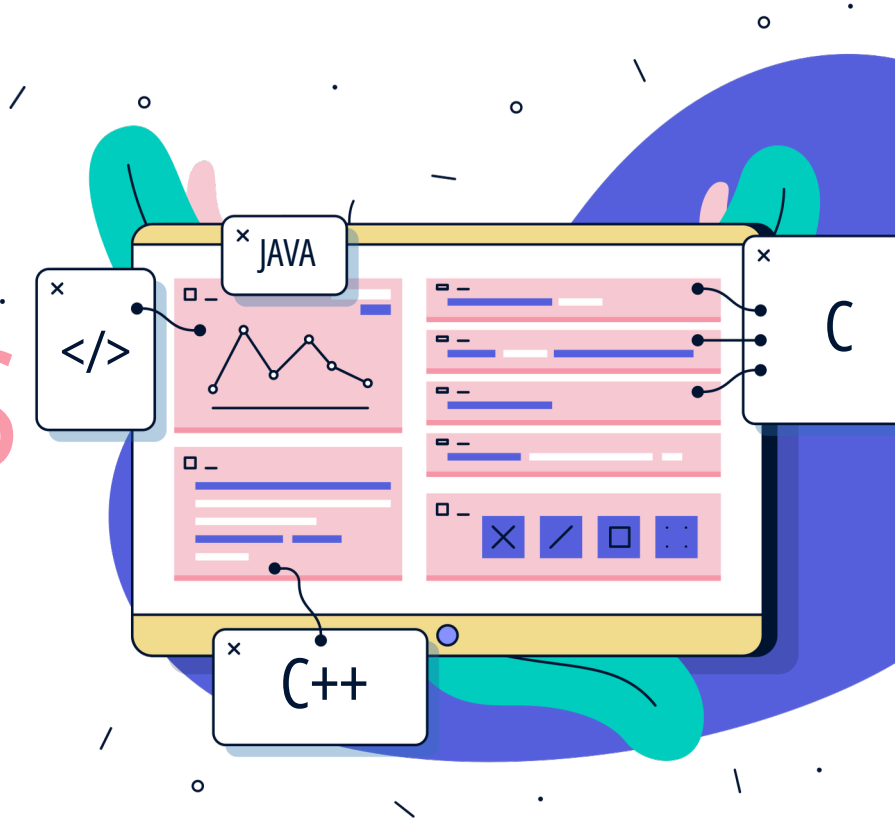
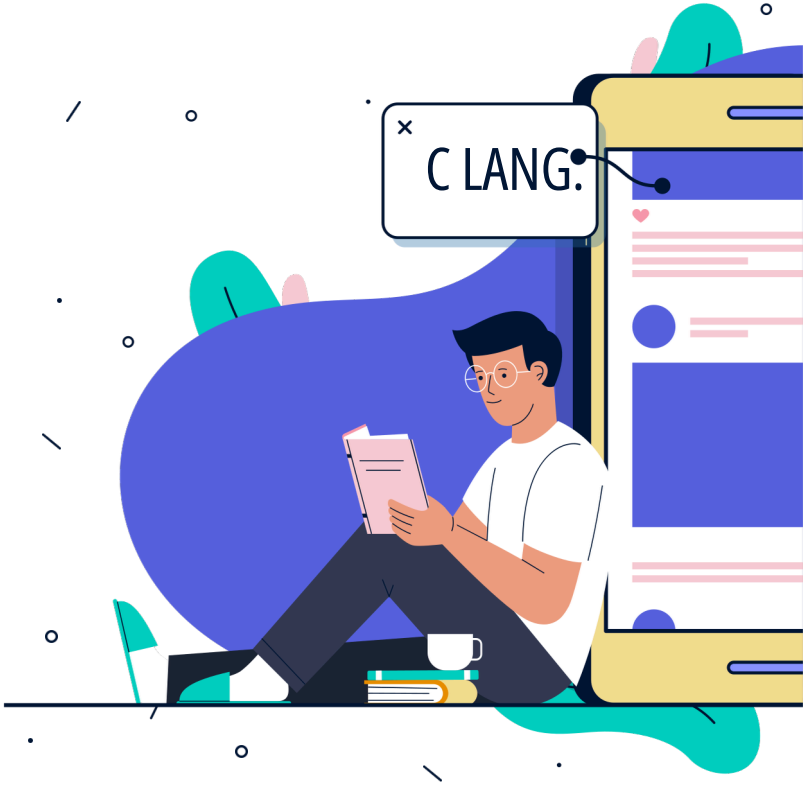
Welcome,

PROGRAMMERS



01.

What is Operator?



Operator

An **operator** is **a symbol that represents a specific operation** on one or more operands.
Operators are fundamental to programming languages as they enable you to perform various computations and manipulate data.

Types of Operators

Operators in C

	Operator	Type
Unary operator	+ +, - -	Unary operator
Binary operator	+, -, *, /, %	Arithmetic operator
	<, <=, >, >=, ==, !=	Relational operator
	&&, ||, !	Logical operator
	&, |, <<, >>, ~, ^	Bitwise operator
	=, +=, -=, *=, /=, %=	Assignment operator
Ternary operator	?:	Ternary or conditional operator

Image Reference from GeeksforGeeks

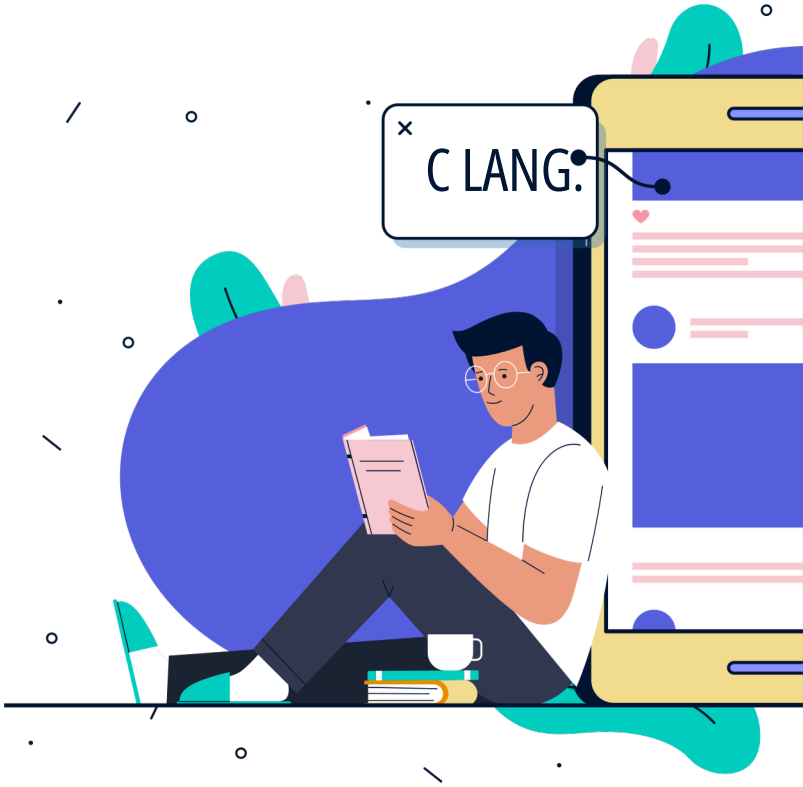
Types of Operators

Miscellaneous Operators:

1. **sizeof:** Returns the size in bytes of a data type or variable.
   & (Address-of): Returns the address of a variable.
2. *** (Pointer dereference):** Accesses the value stored at the address pointed to by a pointer.
3.

02.

What is Operator Precedence?



Operator Precedence

Operator precedence in C determines **the order** in which operators are evaluated when an expression contains multiple operators.
Operators with higher precedence are evaluated before

operators with lower precedence.

Understanding operator precedence is crucial to correctly interpret and predict the outcome of expressions.

Types of Operators

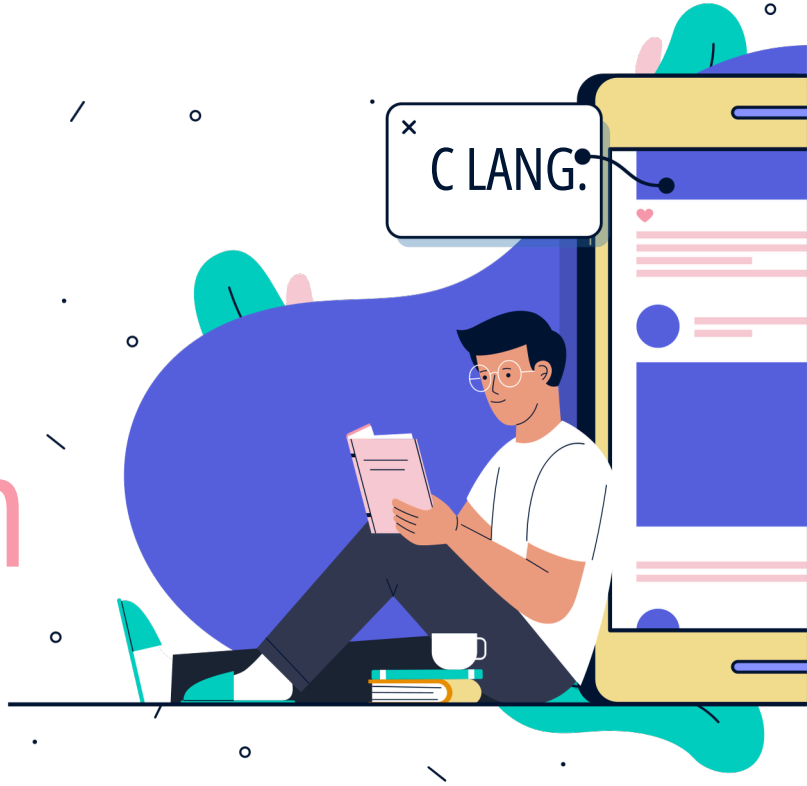
Type of Operator	Associativity	Category
() ++ – –	Left to right	Postfix
(type)* & sizeof	Right to left	The Unary Operator
/ * %	Left to right	The Multiplicative Operator
– +	Left to right	The Additive Operator
< > >= <=	Left to right	The Relational Operator
!= ==	Left to right	The Equality Operator

Types of Operators

Type of Operator	Associativity	Category
&&	Left to right	Logical AND
||	Left to right	Logical OR
?:	Right to left	Conditional
= -= += /= *= %=	Right to left	Assignment
,	Left to right	Comma

03.

What is Type Conversation



Type Conversation

Type conversion in C language, also known as **type casting**, refers to **the process of converting a value from one data type to another**.

Types of Type Conversation

Implicit Type Conversion (**Type Coercion**)

1

Explicit Type Conversion (**Type Casting**)

2

01

Implicit Type Conversion

Implicit type conversion is **performed by the compiler automatically during compilation.**

```
int num1 = 10; float num2 = 5.5;
float ans = num1 + num2;
// num1 is implicitly converted to float before addition
```

02

Explicit Type Conversion

Explicit Type Conversion

Explicit type conversion, or **type casting**, is **done by the programmer explicitly using casting operators**.

It allows the programmer to **override the default behavior of the compiler** and **specify the desired data type for a particular value**.
Type casting is performed using casting operators like **(type)**

LaNguage

Let’s start now...

Explicit Type Conversion

```
int num1 = 10;
float num2 = 5.5;

int ans = num1 + (int)num2;

// num2 is explicitly cast to int before addition
```