**Operators in C#**

Operators in C# are symbols that perform operations on variables and values. C# provides several types of operators, including arithmetic, assignment, comparison, logical, bitwise, and more.

**1️⃣ Arithmetic Operators**

These operators perform mathematical operations.

| **Operator** | **Description** | **Example (a = 10, b = 5)** |
| --- | --- | --- |
| + | Addition | a + b → 15 |
| - | Subtraction | a - b → 5 |
| \* | Multiplication | a \* b → 50 |
| / | Division | a / b → 2 |
| % | Modulus | a % b → 0 |

**2️⃣ Assignment Operators**

Used to assign values to variables.

| **Operator** | **Example** | **Equivalent To** |
| --- | --- | --- |
| = | a = b | a = b |
| += | a += b | a = a + b |
| -= | a -= b | a = a - b |
| \*= | a \*= b | a = a \* b |
| /= | a /= b | a = a / b |
| %= | a %= b | a = a % b |

**3️⃣ Comparison (Relational) Operators**

Used to compare values.

| **Operator** | **Description** | **Example (a = 10, b = 5)** | **Result** |
| --- | --- | --- | --- |
| == | Equal to | a == b | false |
| != | Not equal to | a != b | true |
| > | Greater than | a > b | true |
| < | Less than | a < b | false |
| >= | Greater or equal | a >= b | true |
| <= | Less or equal | a <= b | false |

**4️⃣ Logical Operators**

Used for logical operations.

| **Operator** | **Description** | **Example (a = true, b = false)** | **Result** |
| --- | --- | --- | --- |
| && | AND | a && b | false |
| ` |  | ` | OR |
| ! | NOT | !a | false |

**5️⃣ Bitwise Operators**

Used to perform bit-level operations.

| **Operator** | **Description** | **Example (a = 5 (0101), b = 3 (0011))** | **Result** |
| --- | --- | --- | --- |
| & | AND | a & b → 0001 (1) |  |
| ` | ` | OR | `a |
| ^ | XOR | a ^ b → 0110 (6) |  |
| ~ | NOT | ~a (bitwise complement) |  |
| << | Left Shift | a << 1 → 1010 (10) |  |
| >> | Right Shift | a >> 1 → 0010 (2) |  |

**6️⃣ Increment & Decrement Operators**

| **Operator** | **Description** | **Example (a = 5)** | **Result** |
| --- | --- | --- | --- |
| ++ | Increment | a++ (post) or ++a (pre) | 6 |
| -- | Decrement | a-- (post) or --a (pre) | 4 |

**7️⃣ Ternary Operator (?:)**

Short-hand for if...else.

csharp

CopyEdit

int a = 10, b = 5;

string result = (a > b) ? "A is greater" : "B is greater";

Console.WriteLine(result); // Output: A is greater

**8️⃣ Null-Coalescing Operators (??, ??=)**

Used to deal with null values.

csharp

CopyEdit

string name = null;

string result = name ?? "Default Name"; // If `name` is null, use "Default Name"

Console.WriteLine(result); // Output: Default Name

**Conclusion**

C# provides various operators for arithmetic, assignment, comparison, logic, and bitwise operations. The most commonly used ones are arithmetic (+, -, \*, /), logical (&&, ||, !), and comparison (==, !=, <, >). Understanding these is essential for programming in C#. 🚀

### ****Type Conversion in C#****

Type conversion in C# allows you to convert one data type into another. There are **two types** of type conversion:

1. **Implicit Conversion (Automatically done by the compiler)**
2. **Explicit Conversion (Manually done using methods or casting)**

## ****1️⃣ Implicit Type Conversion (Widening)****

✅ **Done automatically by the compiler** when there is no risk of data loss.  
✅ **Smaller type → Larger type** (e.g., int → long, float → double).

csharp

CopyEdit

int num = 10;

double d = num; // Implicit conversion from int to double

Console.WriteLine(d); // Output: 10

**Examples of implicit conversion:**

* byte → short → int → long → float → double
* char → int
* int → float
* int → double
* long → double

## ****2️⃣ Explicit Type Conversion (Narrowing)****

❌ **Not done automatically** because it may cause data loss.  
✅ **Larger type → Smaller type** (e.g., double → int).  
✅ Done using **casting ((type))**.

csharp

CopyEdit

double d = 10.5;

int num = (int)d; // Explicit conversion using casting

Console.WriteLine(num); // Output: 10 (fractional part lost)

**Example of explicit conversion:**

* double → int
* float → int
* long → int
* decimal → float

## ****3️⃣ Conversion Using Built-in Methods****

C# provides built-in methods to convert data types.

### ****🔹 Convert Class****

The Convert class contains methods like ToInt32(), ToDouble(), etc.

csharp

CopyEdit

string str = "100";

int num = Convert.ToInt32(str); // Convert string to int

Console.WriteLine(num); // Output: 100

**Common Methods in Convert Class:**

| **Method** | **Description** |
| --- | --- |
| Convert.ToInt32(value) | Converts to int |
| Convert.ToDouble(value) | Converts to double |
| Convert.ToString(value) | Converts to string |
| Convert.ToBoolean(value) | Converts to bool |
| Convert.ToDecimal(value) | Converts to decimal |

## ****4️⃣ Parsing Methods (****int.Parse()****,**** double.Parse()****)****

Used to convert **strings to numeric types**.

csharp

CopyEdit

string str = "123";

int num = int.Parse(str);

Console.WriteLine(num); // Output: 123

❌ **Throws an exception if conversion fails.**

csharp

CopyEdit

string str = "123abc";

int num = int.Parse(str); // Error: FormatException

## ****5️⃣**** TryParse() ****(Safer Than**** Parse()****)****

✅ **Returns true if conversion succeeds, false if it fails.**  
✅ **Does not throw an exception.**

csharp

CopyEdit

string str = "123";

bool success = int.TryParse(str, out int num);

Console.WriteLine(success); // Output: True

Console.WriteLine(num); // Output: 123

If the conversion fails:

csharp

CopyEdit

string str = "abc";

bool success = int.TryParse(str, out int num);

Console.WriteLine(success); // Output: False

Console.WriteLine(num); // Output: 0 (default int value)

## ****6️⃣**** ToString() ****Method (Any Type to String)****

Used to convert **any data type to string**.

csharp

CopyEdit

int num = 100;

string str = num.ToString();

Console.WriteLine(str); // Output: "100"

## ****7️⃣ Boxing & Unboxing (Value Type ⬌ Object)****

Boxing and unboxing help convert **value types to object types** and vice versa.

🔹 **Boxing (Value Type → Object)**

csharp

CopyEdit

int num = 10;

object obj = num; // Boxing

Console.WriteLine(obj); // Output: 10

🔹 **Unboxing (Object → Value Type)**

csharp

CopyEdit

object obj = 10;

int num = (int)obj; // Unboxing

Console.WriteLine(num); // Output: 10

### ****Summary of Type Conversion Methods****

| **Method** | **Purpose** | **Example** |
| --- | --- | --- |
| **Implicit Conversion** | Auto conversion (small → large) | int → double |
| **Explicit Casting** | Manual conversion (large → small) | (int)doubleValue |
| **Convert Class** | Converts between types | Convert.ToInt32(str) |
| **Parse()** | Converts string to numeric (throws error on failure) | int.Parse("123") |
| **TryParse()** | Safer than Parse(), returns true/false | int.TryParse("123", out num) |
| **ToString()** | Converts any type to string | num.ToString() |
| **Boxing/Unboxing** | Convert value type to object and back | int → object → int |

### ****💡 Best Practices****

✅ **Use implicit conversion** when possible (e.g., int → double).  
✅ **Use explicit casting** when converting from a larger to a smaller type.  
✅ **Use TryParse() instead of Parse()** to avoid exceptions.  
✅ **Use Convert methods** for flexible conversions.  
✅ **Use ToString()** for converting any type to string.

Let me know if you need more details! 🚀

4o

Top of Form



O

Bottom of Form