# **Part 02**

# what's the difference between compiled and interpreted languages and in this way what about Csharp?

**Compiled Languages :**

1. **Process**:
   * Source code is translated into machine code (binary instructions) by a **compiler**.
   * This machine code is directly executed by the CPU.
   * Examples: C, C++,C#.
2. **Advantages**:
   * Faster execution because the code is already in machine language.
   * Can be optimized more heavily by the compiler.
3. **Disadvantages**:
   * Requires a compilation step before execution, making the development cycle slower.
   * Less portable; binaries are typically platform-dependent

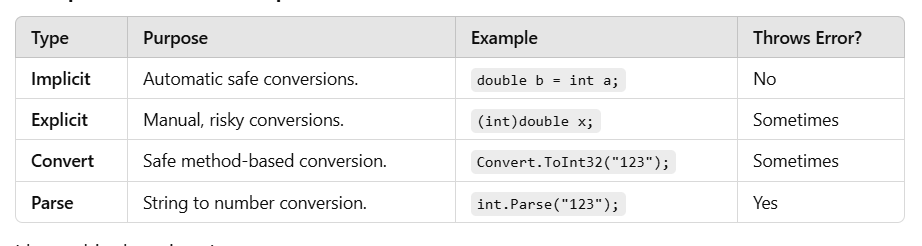
**Interpreted Languages :**

1. **Process**:
   * Source code is executed line-by-line by an **interpreter** without being converted to machine code first.
   * Examples: Python, JavaScript.
2. **Advantages**:
   * Easier to debug since errors are caught at runtime.
   * Faster development cycle as changes can be tested immediately without recompilation.
3. **Disadvantages**:
   * Slower execution because the interpreter processes code dynamically.
   * Performance is generally less optimized.

 C# code is compiled into an intermediate language (IL) by the C# compiler then IL convert to native code (machine code)

The resulting IL is platform-independent and runs on the .NET runtime.

# Compare between implicit, explicit, Convert and parse casting



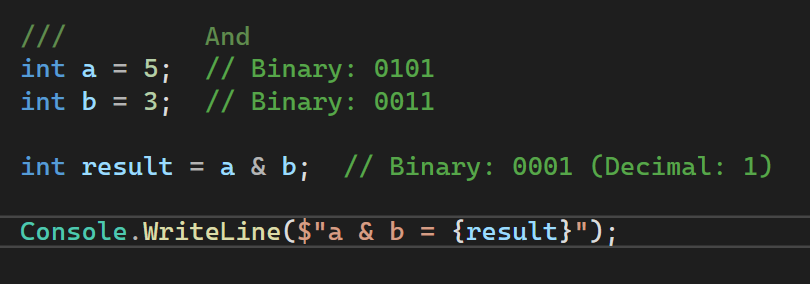
**Part 03**

# Bitwise Operators :

used in programming to perform operations directly on the binary representations of integers. They operate at the bit level, manipulating individual bits in variables. These operators are commonly used in low-level programming, embedded systems, and performance-critical applications.

Example :

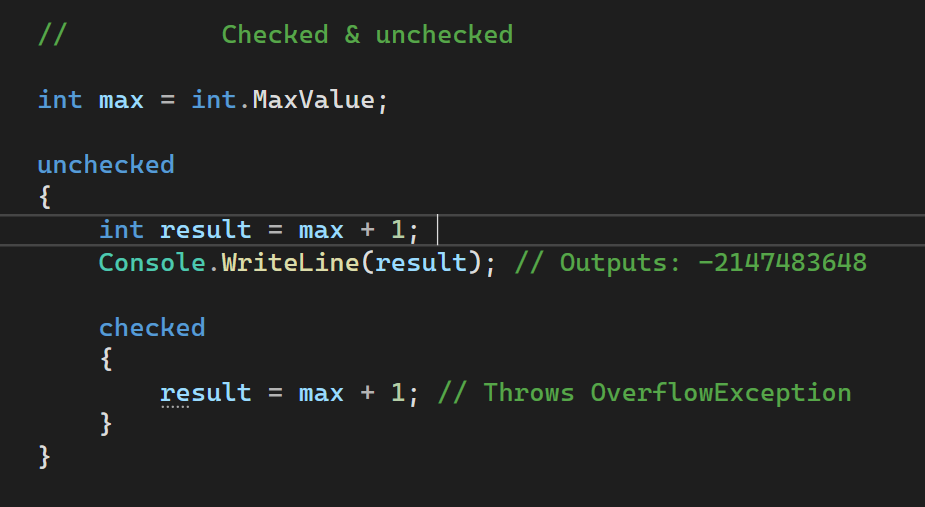
**AND (&) :** Compares each bit of two numbers and returns 1 if both bits are 1, otherwise 0.



**OR(|):**Compares each bit of two numbers and returns 1 if at least one of the bits is 1

# 

# Checked & unchecked : are used to control how the program handles arithmetic overflow .



# what meant by Csharp is managed code?

 Managed code is code that executes in the **managed environment** of the CLR.

 The CLR takes care of tasks such as:

1. **Memory Management**: Automatically allocates and deallocates memory using a Garbage Collector (GC).
2. **Type Safety**: Ensures that type mismatches or unsafe type casts are caught at runtime.
3. **Exception Handling**: Provides a consistent and structured way to handle runtime errors.
4. **Security**: Implements code access security (CAS) and verifies code for potential vulnerabilities.
5. **Cross-Language Interoperability**: Managed code can work seamlessly with other .NET languages like VB.NET or F#.

# what meant by struct is considered like class before ?

| **Feature** | **Struct** | **Class** |
| --- | --- | --- |
| **Type** | Value type | Reference type |
| **Memory Allocation** | Stored on the stack | Stored on the heap |
| **Default Constructor** | No explicit default constructor (compiler provides one) | Can have a default constructor (user-defined) |
| **Inheritance** | Cannot inherit from other structs or classes (except interfaces) | Can inherit from other classes (single inheritance) |
| **Nullability** | Cannot be null (except nullable types like int?) | Can be null |
| **Copying Behavior** | Copies the value (by value) | Copies the reference (by reference) |
| **Garbage Collection** | No garbage collection (managed by stack) | Garbage collected (managed by heap) |
| **Methods** | Can have methods, properties, and events | Can have methods, properties, and events |
| **Properties** | Can have properties | Can have properties |
| **Equality Comparison** | Can override Equals and GetHashCode | Can override Equals and GetHashCode |
| **Immutability** | Often used for immutable types (e.g., structs with readonly fields) | Can be immutable or mutable (depending on the design) |
| **Records (C# 9+)** | Can define as a record struct for immutability and value equality | Can define as a record class for immutability and value equality |
| **Default Values** | Automatically initialized to default values (e.g., 0 for int) | Automatically initialized to null for reference types |