**Value Types**

Value types store data directly. When you assign a value type to a variable, it holds the actual data. Common value types include:

Primitive types: int, float, double, char, bool

Structs: Custom structures defined using the struct keyword

Enumerations: Defined using the enum keyword

When you pass a value type to a method, a copy of the data is passed. This means changes to the parameter within the method do not affect the original data.

Example:

int a = 10;

int b = a; // b is a copy of a

b = 20; // Changing b does not affect a

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

**Reference Types**

Reference types store references to the actual data. When you assign a reference type to a variable, it holds a reference (or address) to the data. Common reference types include:

Classes: Defined using the class keyword

Arrays

Delegates

Strings: Although they behave like value types in some contexts

When you pass a reference type to a method, a reference to the data is passed. This means changes to the parameter within the method can affect the original data.

Example:

class MyClass

{

public int Value;

}

MyClass obj1 = new MyClass();

obj1.Value = 10;

MyClass obj2 = obj1; // obj2 is a reference to obj1

obj2.Value = 20; // Changing obj2 affects obj1

Console.WriteLine(obj1.Value); // Output: 20

**Key Differences**

Memory Allocation: Value types are usually allocated on the stack, while reference types are allocated on the heap.

Copy Behavior: Value types are copied by value, whereas reference types are copied by reference.

Default Values: Value types have default values (e.g., 0 for int), while reference types default to null.