

TYPES AND VALUES

What does a type do?



PROVIDES A GUARANTEE OF
SAFETY DURING COMPILATION



HELPS THE COMPILER PRODUCE
EFFICIENT CODE



MAKE A PROGRAM MORE
PREDICTABLE

C# Specification wording

- C# is a strongly typed language. Every variable and constant has a type, as does every expression that evaluates to a value. Every method declaration specifies a name, the type and kind (value, reference, or output) for each input parameter and for the return value.
- <https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/types/>

Java Specification wording

- The Java programming language is a statically typed language, which means that every variable and every expression has a type that is known at compile time.
- The Java programming language is also a strongly typed language, because types limit the values that a variable (§4.12) can hold or that an expression can produce, limit the operations supported on those values, and determine the meaning of the operations. Strong static typing helps detect errors at compile time.
- <https://docs.oracle.com/javase/specs/jls/se8/html/jls-4.html>

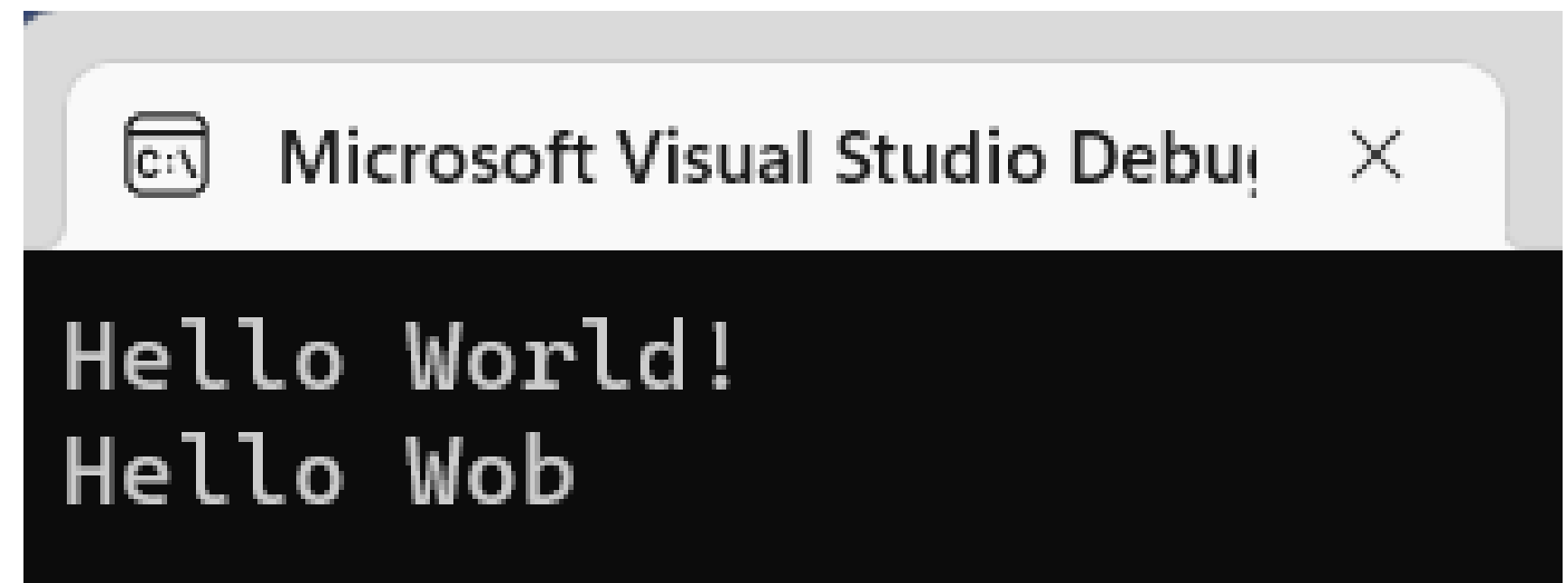
C++ and C are Weakly Typed

- This does not mean bad
- There is a static type checker
- It is possible to convert a value into an incompatible type
- In other words, you can work around the type system

C++ Type System Violation

```
#include <iostream>

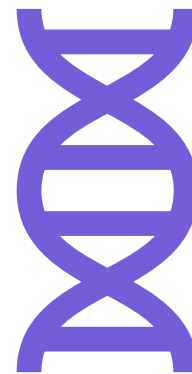
int main()
{
    char* text = new char[13] { "Hello World!" };
    std::cout << text << std::endl;
    int* ptr = (int*)(text);
    ptr[2] = 98;
    std::cout << text << std::endl;
}
```



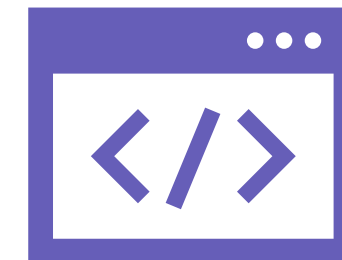
Difference Between C/C++ and C#/Java



In C/C++ you manipulate bits in a memory location



In C/C++ types are properties of variables and heap-allocated polymorphic objects



In C#/Java you are creating or querying values

C# is a Strongly Typed Language

- All variables (including parameters) and expressions have types
- Types are checked at compile-time (static-typing)
- Values are assigned types at run-time
- Those values cannot change types at run-time
- By default it is also statically typed
- Expressions of type “dynamic” defer all checking until run-time

Run-Time Type vs Type System

In C# values have type information



Theorists call this a “type tag”



It is not part of a “type system”.



This is confusing!

C++ RTTI

It is possible to ask the type of certain values allocated on the heap in C++

You use the typeid operator and get a `type_info` struct if you are lucky

Gives you a hash code and an implementation specific name

C# Values all have Fixed Types



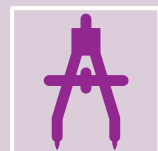
All run-time values have types



You cannot change the type of a value in C#



You can query the exact type using the GetType method



This is possible because all C# values derive from System.Object

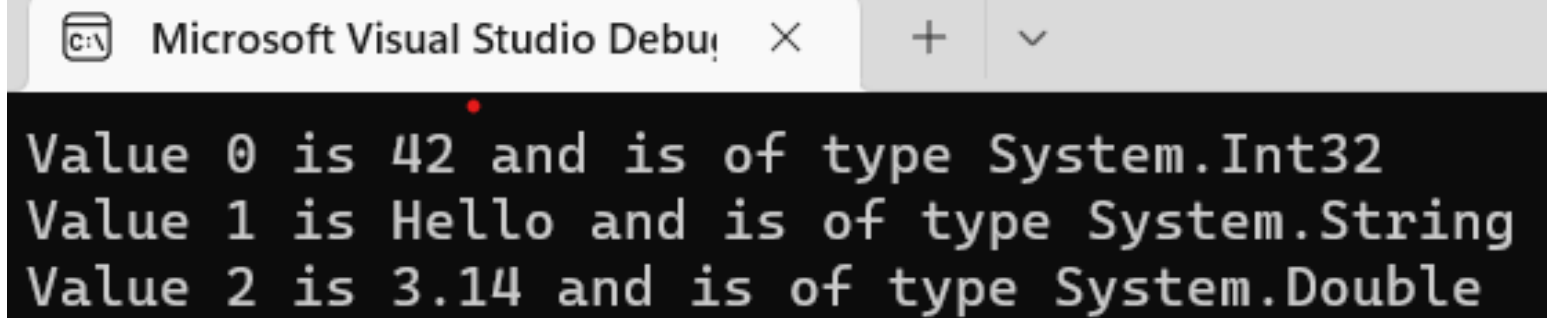
Common System.Object Methods

- [Equals](#) - Compare two objects
- [GetHashCode](#) - Generates a number from the value for use in hash tables
- [ToString](#) - Returns a human-readable string
- [GetType](#) – Returns a System.Type object describing the object

Why is this cool?

```
using System;

namespace SystemObjectDemo
{
    public static class ObjectDemo
    {
        public static void DemoObjects()
        {
            object a = 42;
            object b = "Hello";
            object c = 3.14;
            object[] xs = { a, b, c };
            var i = 0;
            foreach (var x in xs)
            {
                Console.WriteLine($"Value {i++} is {x} and is of type {x.GetType()}");
            }
        }
    }
}
```



Value 0 is 42 and is of type System.Int32
Value 1 is Hello and is of type System.String
Value 2 is 3.14 and is of type System.Double