【Unity基础教程】重点知识汇总

（十九）

Unity C#值类型与引用类型的区别

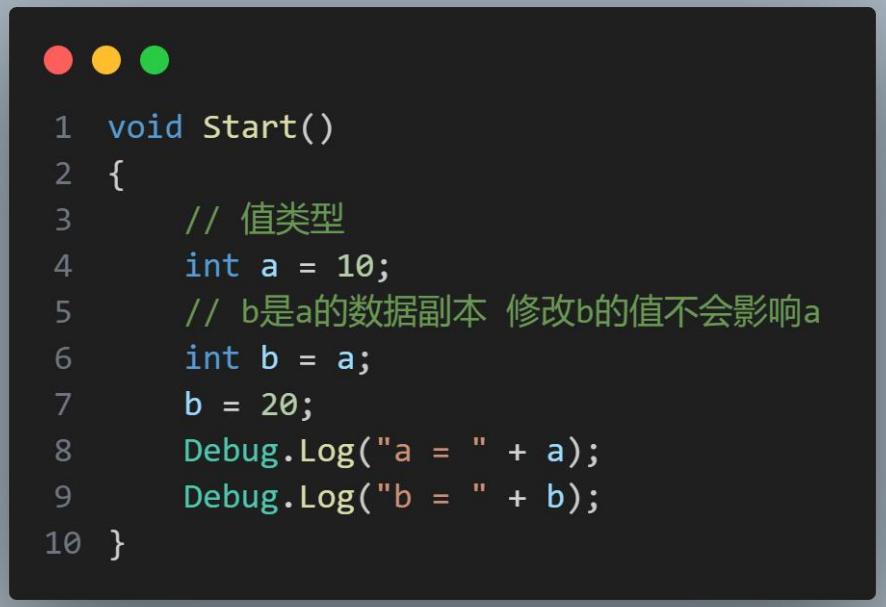
值类型（概念）

概念：值类型存储的是**数据本身**，也就是**实际的值**。当你将一个值类型的变量赋值给另一个变量时，实际上是将**数据的副本复制给另一个变量**。常见的值类型有**int（整型）、float（单精度浮点型）、double（双精度浮点型）、char（字符类）、bool（布尔型）、struct（结构体复合数据类型）**等。值类型通常存储在**栈**上。当它超过作用域时会立即被销毁。

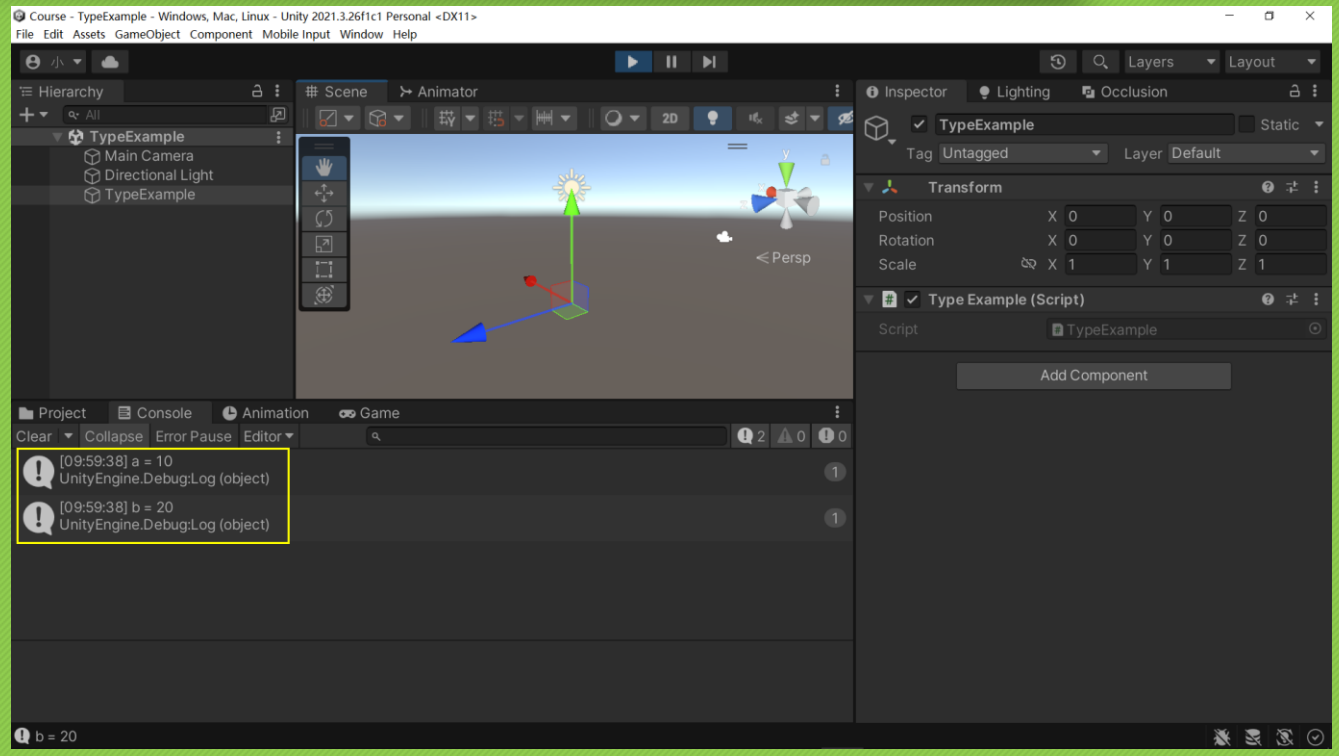
*在计算机科学中，栈（Stack）是一种特殊的**数据结构**，**后进先出**（LIFO, Last In First Out），简单来说，栈就像是一叠叠放的盘子，**最后放进去的盘子会是最先被取出来的**。在内存中，栈是一块专门分配出来的区域，用来存储函数调用相关的数据，比如局部变量、函数参数、返回地址等。

为什么将值类型存储在栈上？栈上的内存分配要求大小是已知的（值类型的大小也是固定的），同时栈的内存分配和释放非常快速，适合处理值类型的小数据。

值类型（具体实现）



```
void Start()
{
    // 值类型
    int a = 10;
    // b是a的数据副本 修改b的值不会影响a
    int b = a;
    b = 20;
    Debug.Log("a = " + a);
    Debug.Log("b = " + b);
}
```



引用类型（概念）

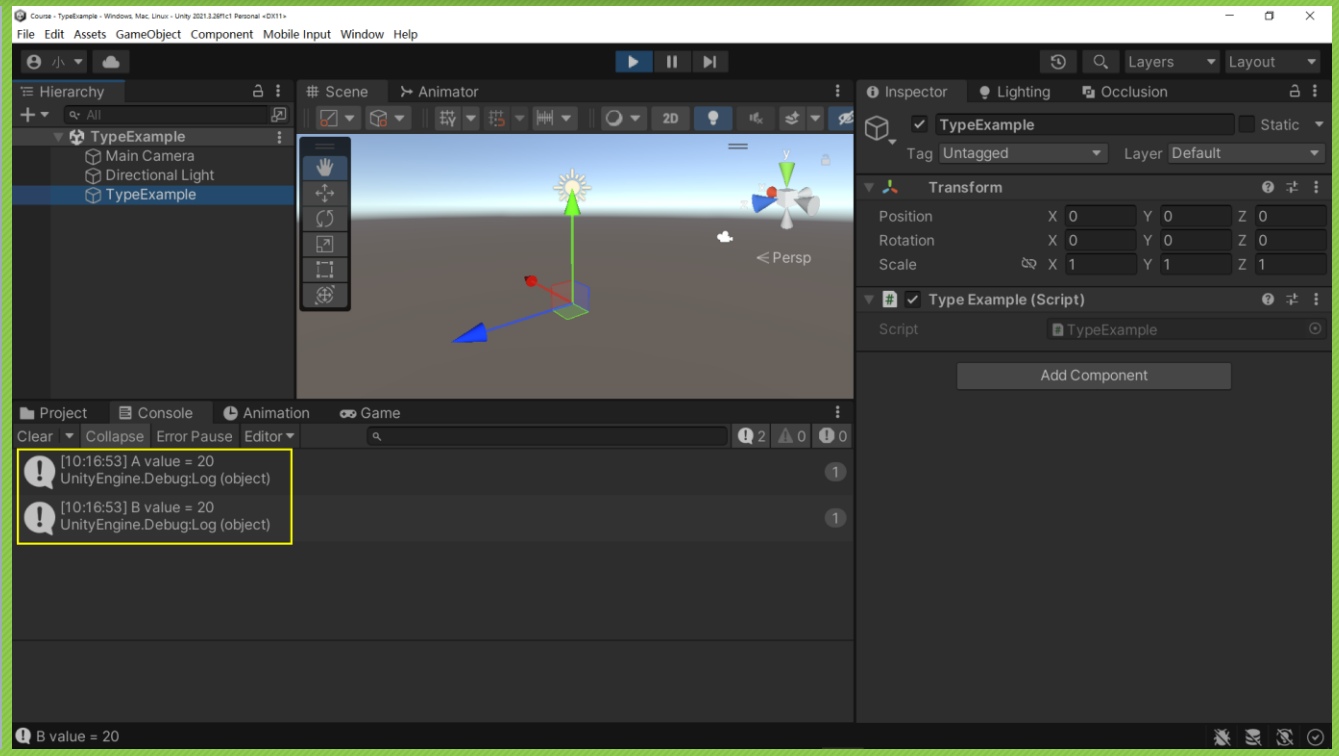
概念：引用类型存储的是**对象的引用**，也就是**内存地址**，而**不是数据本身**。当你将一个引用类型的变量赋值给另一个变量时，**两个变量指向同一个对象**。常见的引用类型有**class（类）、interface（接口）、delegate（委托）、array（数组）**等。引用类型通常存储在**堆**上，引用类型的对象生命周期由**垃圾回收器（GC）**管理，当没有任何引用指向该对象时，垃圾回收器会自动回收该对象。

*堆（Heap）是一种内存管理区域，用于**动态分配内存**。它是与栈（Stack）相对的内存区域，堆允许在运行时分配和释放内存，通常用于存储对象和数据结构。

为什么将引用类型存储在堆上？引用类型的大小在编译时是**不确定**的，因此它们需要动态分配内存。堆是一个适合存储这种动态数据的地方。栈内存一般用于存储大小已知并且生命周期较短的数据（如局部变量），而堆则适合存储生命周期较长的数据（如对象实例）。

引用类型（具体实现）

```
1  class MyClass
2  {
3      public int value;
4  }
5  void Start()
6  {
7      // 引用类型
8      MyClass objA = new MyClass();
9      objA.value = 10;
10     // objB引用的是和objA相同的对象 修改B的Value值会影响到A的Value值
11     MyClass objB = objA;
12     objB.value = 20;
13     Debug.Log("A value = " + objA.value);
14     Debug.Log("B value = " + objB.value);
15 }
```



值类型与引用类型的区别（总结）

特性	值类型	引用类型
存储位置	存储在 **栈（Stack）** 中。	存储在 **堆（Heap）** 中，变量本身存储的是堆对象的引用。
数据传递方式	传递的是 **数据的副本**（复制整个值）。	传递的是 **对象的引用地址**（指针）。
内存分配效率	**高效**，因栈是连续分配的，分配和释放内存速度快。	**稍慢**，堆分配需要更多时间，且可能导致内存碎片化。
修改影响范围	修改副本不会影响原始变量。	修改对象会影响所有引用它的变量。
默认值	默认值是其类型的“零值”（如 `0`、`false` 等）。	默认值是 `null`。
继承	值类型不能从其他类型继承，也不能被继承。	引用类型支持继承和多态。
垃圾回收	不会被垃圾回收（由编译器自动管理生命周期）。	会被垃圾回收器（GC）管理。
示例	`int`、`float`、`bool`、`struct`、`enum`	`class`、`string`、`object`、`array`

*栈与堆的区别（总结）

特性	堆（Heap）	栈（Stack）
内存分配	动态分配内存，通过 `new` 等手动分配	自动分配内存，由操作系统或编译器管理
分配速度	较慢	较快
内存管理	由操作系统管理，但需要垃圾回收（C#中）	由编译器自动管理，方法调用结束后自动释放
内存大小	较大，可以动态扩展	较小，受限于线程栈大小
存储类型	存储引用类型数据（对象、数组等）	存储值类型数据（基本类型、局部变量等）
生命周期	生命周期由开发者或垃圾回收器管理	生命周期由方法调用决定，方法退出后自动销毁
访问方式	随机访问，可以通过指针访问	后进先出（LIFO），仅能在栈顶操作
内存碎片	容易产生内存碎片	通常没有内存碎片问题
使用场景	用于存储需要动态分配且生命周期不确定的对象	用于存储方法调用中的局部变量和参数

【Unity基础教程】重点知识汇总

（十九）

Unity C#值类型与引用类型的区别