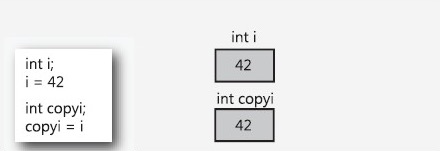
**C#值类型和引用类型**

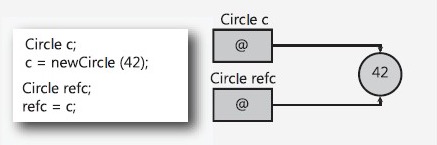
**1、简单比较**

　　值类型的变量直接存储数据，而引用类型的变量持有的是数据的引用，数据存储在数据堆中。

　　值类型（value type）：byte，short，int，long，float，double，decimal，char，bool 和 struct 统称为值类型。值类型变量声明后，不管是否已经赋值，编译器为其分配内存。



　　引用类型（reference type）：string 和 class统称为引用类型。当声明一个类时，只在栈中分配一小片内存用于容纳一个地址，而此时**并没有为其分配堆上的内存空间**。当使用 new 创建一个类的实例时，分配堆上的空间，并把堆上空间的地址保存到栈上分配的小片空间中。



　　值类型的实例通常是在线程栈上分配的（静态分配），但是在某些情形下可以存储在堆中。引用类型的对象总是在进程堆中分配（动态分配）。

**2、相同点**

* 引用类型可以实现接口，值类型当中的结构体也可以实现接口；
* 引用类型和值类型都继承自System.Object类。

**3、不同点**

**1）范围方面**

* C#的值类型包括：结构体（数值类型、bool型、用户定义的结构体），枚举，可空类型。
* C#的引用类型包括：数组，用户定义的类、接口、委托，object，字符串。

**2）内存分配方面**

* 数组的元素不管是引用类型还是值类型，都存储在托管堆上。
* 引用类型在栈中存储一个引用，其实际的存储位置位于托管堆。简称引用类型部署在托管推上。而值类型总是分配在它声明的地方：作为字段时，跟随其所属的变量（实 例）存储；作为局部变量时，存储在栈上。（栈的内存是自动释放的，堆内存是.NET中会由GC来自动释放）

**3）适用场合**

　　值类型在内存管理方面具有更好的效率，并且不支持多态，适合用做存储数据的载体；引用类型支持多态，适合用于定义应用程序的行为。

* 引用类型可以派生出新的类型，而值类型不能，因为所有的值类型都是密封（seal）的；
* 引用类型可以包含null值，值类型不能（可空类型功能允许将 null 赋给值类型，如   int? a = null;  ）；
* 引用类型变量的赋值只复制对对象的引用，而不复制对象本身。而将一个值类型变量赋给另一个值类型变量时，将复制包含的值。