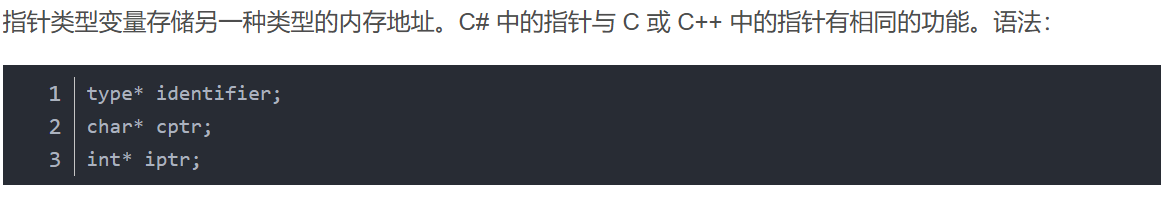
* 1. 数据类型
* 值类型



* 引用类型

C# 中内置的引用类型包括 Object（对象）、Dynamic（动态）和 string（字符串）。

* 指针类型



* 装箱与拆箱

装箱就是值类型转换为引用类型；拆箱就是引用类型转换为值类型

值类型,包括原类型（Sbyte、Byte、Short、Ushort、Int、Uint、Long、Ulong、Char、Float、Double、Bool、Decimal）、枚举 (enum) 、结构 (struct)。

引用类型,包括类、数组、接口、委托、字符串等

装箱:值类型到引用类型或到此值类型所实现的任何接口类型的隐式转换

例如: int temp = 3;

System.Object obj = temp；

其中，temp为值类型，在栈中分配；当分配obj这个引用类型时，我们需要在堆中分配一个obj对象，然后把temp值赋给它，这么一系列的过程就是装箱的过程。

拆箱:从引用类型到任意值类型的显式转换。与装箱不同，拆箱式显示转换。

例如：int temp = 3；

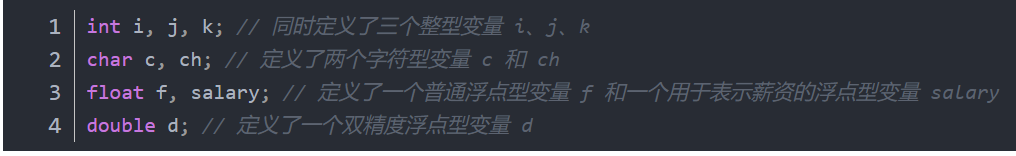
System.Object obj = temp；

int i = (int) obj;

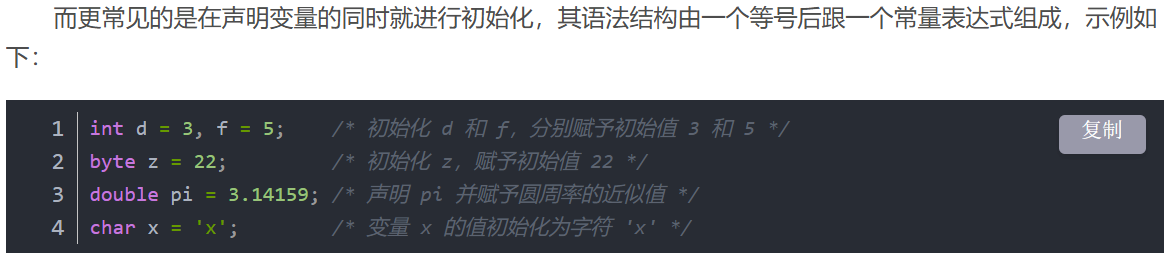
拆箱过程中，首先来确定对象obj为一个值类型的装箱值，然后把值赋给值类型。

* 1. 变量

定义



初始化



* 1. 常量



* 1. 运算符
* 算术运算符



* 关系运算符



* 逻辑运算符



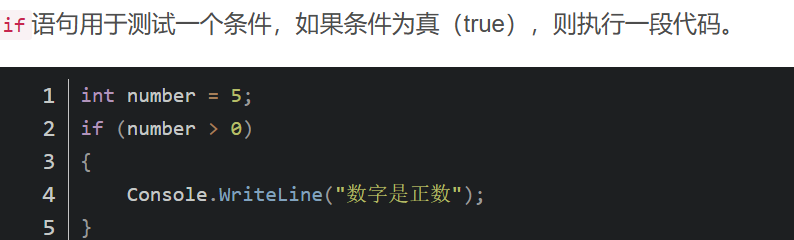
* 位运算符



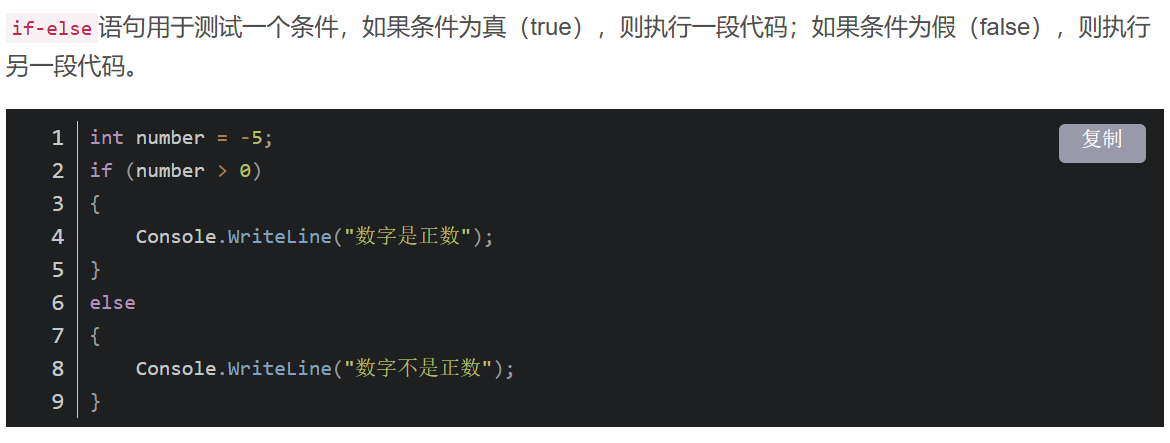
* 赋值运算符



* 1. 判断语句
* if 语句



* if … else语句



* 嵌套 if 语句



* switch 语句



* ?: 条件运算符



**Array与ArrayList的区别**

1. 类型安全：

Array是类型安全的，意味着它只能存储一种特定类型的元素。例如，一个int[]数组只能存储int类型的元素。

ArrayList不是类型安全的，可以存储任何类型的对象，即它可以存储object类型的元素。因此你可以将任何类型的对象添加到ArrayList中，这在编译时不会报错，但可能在运行时引发类型错误。

1. 性能：

Array由于是类型安全且固定大小的，通常在性能上比ArrayList更优，尤其是在类型转换和装箱操作方面。

ArrayList在添加元素时可能需要动态扩展其大小，这个过程涉及到内存分配和数组复制，可能会导致性能开销。

1. 大小：

Array的大小在创建时被确定，之后不能改变。

ArrayList的大小是动态的，可以随着元素的添加和移除而增长或缩小。

1. 功能和方法：

Array提供了基础的数组操作功能，例如长度查询、索引访问和迭代等。

ArrayList提供了更多的集合操作方法，如添加、删除、查找、插入和排序等。

1. 泛型支持：

Array支持泛型，但通常以非泛型的形式使用，例如int[]、string[]等。

ArrayList不支持泛型。如果你需要一个支持泛型的动态数组，应该使用List<T>，它允许你创建一个类型安全的列表，例如List<int>或List<string>。

1. .NET版本：

Array一直是.NET Framework的一部分。

ArrayList是早期.NET版本中的非泛型集合。自从.NET 2.0引入了泛型之后，List<T>通常被推荐使用，因为它提供了类似ArrayList的功能，同时增加了类型安全和更好的性能。