C# string 字符串的前面可以加 @（称作"逐字字符串"）将转义字符（\）当作普通字符对待。

string str = @"C:\Windows";

等价于：

string str = "C:\\Windows";

类型转换从根本上说是类型铸造，或者说是把数据从一种类型转换为另一种类型。在 C# 中，类型铸造有两种形式：

* **隐式类型转换** - 这些转换是 C# 默认的以安全方式进行的转换, 不会导致数据丢失。例如，从小的整数类型转换为大的整数类型，从派生类转换为基类。
* **显式类型转换** - 显式类型转换，即强制类型转换。显式转换需要强制转换运算符，而且强制转换会造成数据丢失。

**C#类型转换方法**

ToBoolean,ToByte,Tochar,ToDateTime,ToDecimal,ToDouble,ToInt16,ToInt32,ToInt64,ToSbyte,ToSingle(小浮点数类型),ToString,ToType(转化成指定类型),ToUInt16,ToUInt32,ToUInt64。

int.TryParse(string s,out int i)

该方式也是将数字内容的字符串转换为int类型，但是该方式比int.Parse(string s) 好一些，它不会出现异常，最后一个参数result是输出值，如果转换成功则输出相应的值，转换失败则输出0。

例子：

class test

{

    static void Main(string[] args)

    {

        string s1="abcd";

        string s2="1234";

        int a,b;

        bool bo1=int.TryParse(s1,out a);

        Console.WriteLine(s1+" "+bo1+" "+a);

        bool bo2=int.TryParse(s2,out b);

        Console.WriteLine(s2+" "+bo2+" "+b);

    }

}

C# 中对 double 类型的数据取整，可以使用 **convert.ToInt32()** 方法，也可使用 int 强制转换为整数，使用 int 时并不存在四舍五入的情况，而是直接将后面的小数位数丢掉。

**Convert.ToInt32()和int.Parse()区别**

1. 对null的处理方法

Convert.ToInt32(null)返回0，而int.Parse(null)产生异常。

1. int.Parse()只能转换整数，如int.Parse(“4.3”)则会报错

**隐式转换只能是低精度到高精度，否则会报错。**

**Console.ReadLine()接受来自输入的字符串，如果要转换成其他类型，用convert.**

**类中的成员函数如果没有指定修饰符，则使用默认修饰符private。**

**按引用传递参数：**

**在 C# 中，使用 ref 关键字声明引用参数。如 public void swap(ref int x,ref int y)。**

**按输出传递参数：**

**可以使用 输出参数 来从函数中返回值。输出参数会把方法输出的数据赋给自己，其他方面与引用参数相似。如 public void getValue(out int x)。**

**注意：使用out关键字来声明输出参数时，其本身必须是未赋初值的，否则会报错。**

**二维数组声明：**

**int[,] array=new int[2,3];**

**交错数组声明：**

**int[][] array=new int[2][3];**

**二维数组可以看做是一个矩阵，交错数组和二维数组直观上的区别是交错数组在每一行的长度上可以不一样。**

**C# 结构的特点**

**-结构可以带有方法，字段，索引，属性，运算符方法和事件。**

**-结构可以定义构造函数，但是不能定义析构函数。**

**-结构不能继承**

**-结构可以实现一个或者多个接口**

**-可以不用调用new来对结构进行实例化。**

## 类 vs 结构

类和结构有以下几个基本的不同点：

* 类是引用类型，在堆中分配空间，结构是值类型，在栈中分配空间。
* 结构不支持继承。
* 结构不能声明默认的构造函数。
* 结构中声明的字段无法赋初值，类可以。

## C# 类的静态成员

我们可以使用 **static** 关键字把类成员定义为静态的。当我们声明一个类成员为静态时，意味着无论有多少个类的对象被创建，只会有一个该静态成员的副本。如果没有对类成员进行初始化，默认为0。

当声明一个函数为静态成员函数时，它只能访问类的静态成员。

如果将类的成员函数声明为public static，那么不用实例化该类也能访问该函数。