若要确定字符串是否是指定数值类型的有效表示形式，请使用由所有基元数值类型以及如 [DateTime](https://docs.microsoft.com/zh-cn/dotnet/api/system.datetime) 和 [IPAddress](https://docs.microsoft.com/zh-cn/dotnet/api/system.net.ipaddress) 等类型实现的静态 TryParse 方法。 以下示例演示如何确定“108”是否为有效的 [int](https://docs.microsoft.com/zh-cn/dotnet/csharp/language-reference/keywords/int)。

int i = 0;

string s = "108";

bool result = int.TryParse(s, out i); //i now = 108

如果该字符串包含非数字字符，或者数值对于指定的特定类型而言太大或太小，则 TryParse 将返回 false 并将 out 参数设置为零。否则，它将返回 true 并将 out 参数设置为字符串的数值。

字符串可能仅包含数字字符，但对于你使用的 TryParse 方法的类型仍然无效。 例如，“256”不是 byte 的有效值，但对 int有效。 “98.6”不是 int 的有效值，但它是有效的 decimal。

示例：

string numString = "1287543"; //"1287543.0" will return false for a long

long number1 = 0;

bool canConvert = long.TryParse(numString, out number1);

if (canConvert == true)

Console.WriteLine("number1 now = {0}", number1);

else

Console.WriteLine("numString is not a valid long");

byte number2 = 0;

numString = "255"; // A value of 256 will return false

canConvert = byte.TryParse(numString, out number2);

if (canConvert == true)

Console.WriteLine("number2 now = {0}", number2);

else

Console.WriteLine("numString is not a valid byte");

decimal number3 = 0;

numString = "27.3"; //"27" is also a valid decimal

canConvert = decimal.TryParse(numString, out number3);

if (canConvert == true)

Console.WriteLine("number3 now = {0}", number3);

else

Console.WriteLine("number3 is not a valid decimal");

可靠编程：

基元数值类型还实现 Parse 静态方法，如果字符串不是有效数字，该方法将引发异常。 TryParse 通常更高效，因为如果数值无效，它仅返回 false。