**将字符串转换为数字（C# 编程指南）**

可以使用 [Convert](https://msdn.microsoft.com/zh-cn/library/system.convert.aspx) 类中的方法或使用各种数值类型（int、long、float 等）中的 TryParse 方法将[字符串](https://msdn.microsoft.com/zh-cn/library/362314fe.aspx)转换为数字。

如果你具有字符串，则调用 TryParse 方法（例如 int.TryParse(“11”)）会稍微更加高效且简单。使用 Convert 方法对于实现 [IConvertible](https://msdn.microsoft.com/zh-cn/library/system.iconvertible.aspx) 的常规对象更有用。

可以对预期字符串会包含的数值类型（如 [System.Int32](https://msdn.microsoft.com/zh-cn/library/system.int32.aspx) 类型）使用 Parse 或 TryParse 方法。 [Convert.ToUInt32](https://msdn.microsoft.com/zh-cn/library/hs06dadw.aspx) 方法在内部使用 [Parse](https://msdn.microsoft.com/zh-cn/library/b3h1hf19.aspx)。如果字符串的格式无效，则 Parse 会引发异常，而 TryParse 会返回 [false](https://msdn.microsoft.com/zh-cn/library/67bxt5ee.aspx)。

Parse 和 TryParse 方法会忽略字符串开头和末尾的空格，但所有其他字符必须是组成合适数值类型（int、long、ulong、float、decimal 等）的字符。组成数字的字符中的任何空格都会导致错误。例如，可以使用 decimal.TryParse 分析“10”、“10.3”、“ 10 ”，但不能使用此方法分析从“10X”、“1 0”（注意空格）、“10 .3”（注意空格）、“10e1”（float.TryParse 在此处适用）等中分析出 10。

下面的示例演示了对 Parse 和 TryParse 的成功调用和不成功的调用。

using System;

using System.Linq;

using System.Collections;

using System.Collections.Generic;

int numVal = Int32.Parse("-105");

Console.WriteLine(numVal);

// Output: -105

// TryParse returns true if the conversion succeeded

// and stores the result in j.

int j;

if (Int32.TryParse("-105", out j))

Console.WriteLine(j);

else

Console.WriteLine("String could not be parsed.");

// Output: -105

try

{

int m = Int32.Parse("abc");

}

catch (FormatException e)

{

Console.WriteLine(e.Message);

}

// Output: Input string was not in a correct format.

string inputString = "abc";

int numValue;

bool parsed = Int32.TryParse(inputString, out numValue);

if (!parsed)

Console.WriteLine("Int32.TryParse could not parse '{0}' to an int.\n", inputString);

// Output: Int32.TryParse could not parse 'abc' to an int.

// This snippet shows a couple of examples that extract number characters from the

// beginning of the string to avoid TryParse errors.

StringBuilder sb = new StringBuilder();

var str = " 10FFxxx";

foreach (char c in str) {

// Check for numeric characters (hex in this case). Add "." and "e" if float,

// and remove letters. Include initial space because it is harmless.

if ((c >= '0' && c <= '9') || (c >= 'A' && c <= 'F') || (c >= 'a' && c <= 'f') || c == ' ') {

sb.Append(c);

}

else

break;

}

if (int.TryParse(sb.ToString(), System.Globalization.NumberStyles.HexNumber, null, out i))

Console.WriteLine(sb.ToString());

str = " -10FFXXX";

sb.Clear();

foreach (char c in str) {

// Check for numeric characters (allow negative in this case but no hex digits).

// Though we use int.TryParse in the previous example and this one, int.TryParse does NOT

// allow a sign character (-) AND hex digits at the same time.

// Include initial space because it is harmless.

if ((c >= '0' && c <= '9') || c == ' ' || c == '-') {

sb.Append(c);

} else

break;

}

if (int.TryParse(sb.ToString(), out i))

Console.WriteLine(sb.ToString());

下表列出了 [Convert](https://msdn.microsoft.com/zh-cn/library/system.convert.aspx) 类中可使用的一些方法。

| **数值类型** | **方法** |
| --- | --- |
| **decimal** | [ToDecimal(String)](https://msdn.microsoft.com/zh-cn/library/hf9z3s65.aspx) |
| **float** | [ToSingle(String)](https://msdn.microsoft.com/zh-cn/library/faab9yks.aspx) |
| **double** | [ToDouble(String)](https://msdn.microsoft.com/zh-cn/library/zh1hkw6k.aspx) |
| **short** | [ToInt16(String)](https://msdn.microsoft.com/zh-cn/library/basyfs27.aspx) |
| **int** | [ToInt32(String)](https://msdn.microsoft.com/zh-cn/library/sf1aw27b.aspx) |
| **long** | [ToInt64(String)](https://msdn.microsoft.com/zh-cn/library/0zahhahw.aspx) |
| **ushort** | [ToUInt16(String)](https://msdn.microsoft.com/zh-cn/library/5cat4fzy.aspx) |
| **uint** | [ToUInt32(String)](https://msdn.microsoft.com/zh-cn/library/hs06dadw.aspx) |
| **ulong** | [ToUInt64(String)](https://msdn.microsoft.com/zh-cn/library/f65a8ex6.aspx) |

此示例调用 [Convert.ToInt32(String)](https://msdn.microsoft.com/zh-cn/library/sf1aw27b.aspx) 方法将输入的 [string](https://msdn.microsoft.com/zh-cn/library/362314fe.aspx) 转换为 [int](https://msdn.microsoft.com/zh-cn/library/5kzh1b5w.aspx)。代码将捕获此方法可能引发的最常见的两个异常：[FormatException](https://msdn.microsoft.com/zh-cn/library/system.formatexception.aspx) 和 [OverflowException](https://msdn.microsoft.com/zh-cn/library/system.overflowexception.aspx)。如果该数字可以递增而不溢出整数存储位置，则程序使结果加上 1 并打印输出。

using System;

using System.Linq;

using System.Collections;

using System.Collections.Generic;

static void Main(string[] args)

{

int numVal = -1;

bool repeat = true;

while (repeat)

{

Console.WriteLine("Enter a number between −2,147,483,648 and +2,147,483,647 (inclusive).");

string input = Console.ReadLine();

// ToInt32 can throw FormatException or OverflowException.

try

{

numVal = Convert.ToInt32(input);

}

catch (FormatException e)

{

Console.WriteLine("Input string is not a sequence of digits.");

}

catch (OverflowException e)

{

Console.WriteLine("The number cannot fit in an Int32.");

}

finally

{

if (numVal < Int32.MaxValue)

{

Console.WriteLine("The new value is {0}", numVal + 1);

}

else

{

Console.WriteLine("numVal cannot be incremented beyond its current value");

}

}

Console.WriteLine("Go again? Y/N");

string go = Console.ReadLine();

if (go == "Y" || go == "y")

{

repeat = true;

}

else

{

repeat = false;

}

}

// Keep the console open in debug mode.

Console.WriteLine("Press any key to exit.");

Console.ReadKey();

}

// Sample Output:

// Enter a number between -2,147,483,648 and +2,147,483,647 (inclusive).

// 473

// The new value is 474

// Go again? Y/N

// y

// Enter a number between -2,147,483,648 and +2,147,483,647 (inclusive).

// 2147483647

// numVal cannot be incremented beyond its current value

// Go again? Y/N

// Y

// Enter a number between -2,147,483,648 and +2,147,483,647 (inclusive).

// -1000

// The new value is -999

// Go again? Y/N

// n

// Press any key to exit.