

# Методы Parse; TryParse

## Метод Parse

```
1 int a = int.Parse("15");
2 double b = double.Parse("23,56");
3 decimal c = decimal.Parse("12,45");
4 byte d = byte.Parse("4");
5 char e = char.Parse("H");
6 Console.WriteLine($"int={a} double={b} decimal={c} byte={d} char={e}");
7
```

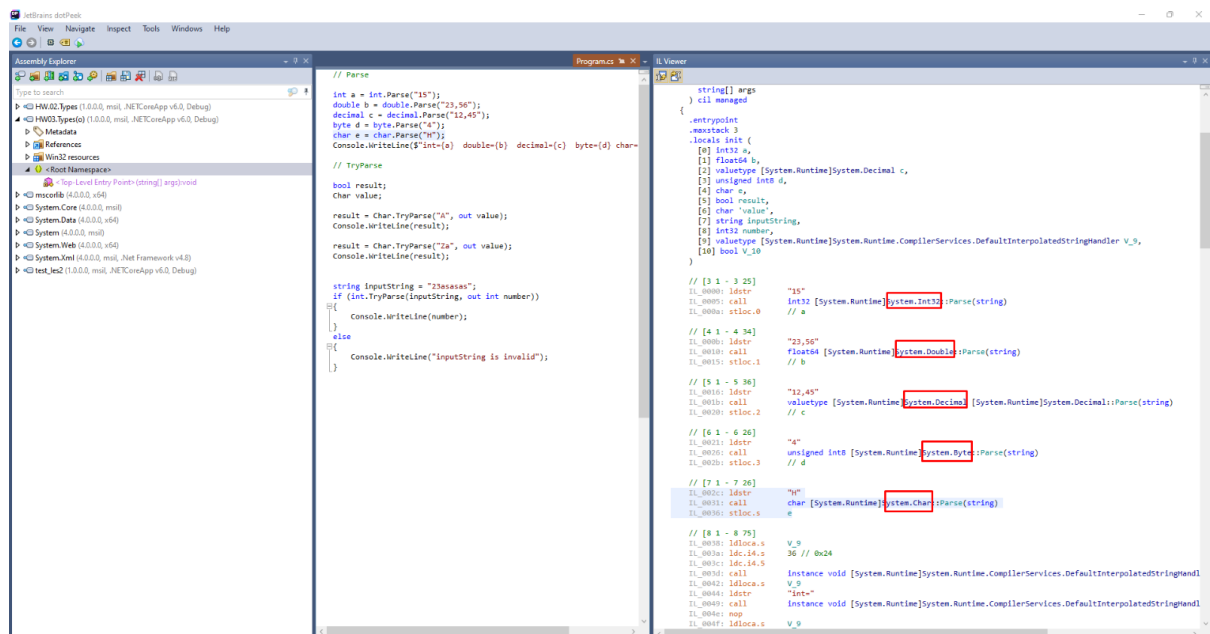
**Parse()** позволяет преобразовать строку к данному типу.

Parse создает исключение, если строка не содержит допустимого числа.

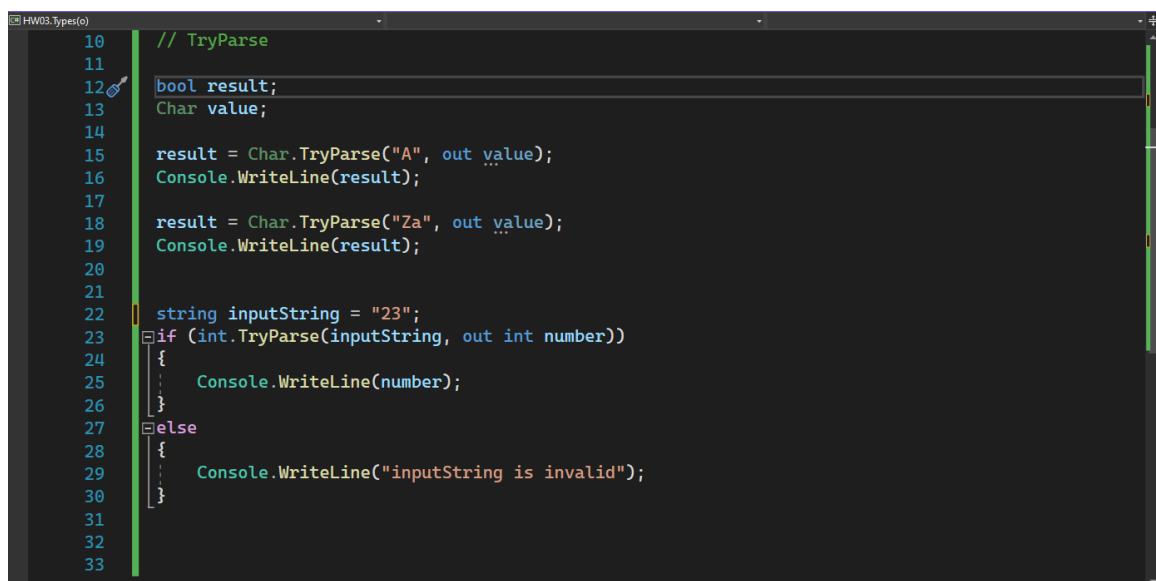
```
1 int a = int.Parse("15");
2 double b = double.Parse("23,56");
3 decimal c = decimal.Parse("12,45");
4 byte d = byte.Parse("4");
5 char e = char.Parse("Hell");
6 Console.WriteLine($"int={a} double={b} decimal={c} byte={d} char={e}");
7
```

Exception Unhandled  
**System.FormatException:** 'String must be exactly one character  
View Details | Copy Details | Start Live Share session...  
Exception Settings

В dotpeek видно что происходит преобразование строки к указанному типу.



## Метод TryParse



Метод пытается преобразовать строку к типу и, если преобразование прошло успешно, то возвращает **true**. Если преобразование пройдет неудачно, то исключения никакого не будет выброшено, просто метод TryParse возвратит **false**

```
True
False
23
C:\Users\andre\source\repos\AndreiSpurhiash\CSharp.Homeworks\HW03.Types(o)\bin\Debug\net6.0\HW03.Types(o).exe (process 21148) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

```
int a = int.Parse("15");
double b = double.Parse("23.56");
decimal c = decimal.Parse("12.45");
byte d = byte.Parse("4");
char e = char.Parse("w");
Console.WriteLine($"{int}(a) double(b) decimal(c) byte(d) char(e)");

// TryParse
bool result;
Char value;

result = Char.TryParse("A", out value);
Console.WriteLine(result);

result = Char.TryParse("2a", out value);
Console.WriteLine(result);

string inputString = "23";
if (int.TryParse(inputString, out int number))
{
    Console.WriteLine(number);
}
else
{
    Console.WriteLine("inputString is invalid");
}
```

```
// [15 1 - 15 40]
IL_0000: nop
IL_0001: ldloc.s V_9
IL_0002: ldstr "char="
IL_0003: call instance void [System.Runtime]System.Runtime.CompilerServices.DefaultInterpolatedStringHandler.AppendFormatted<char>(byref, char)
IL_0004: ldloc.s V_9
IL_0005: ldloc.s e
IL_0006: call instance void [System.Runtime]System.Runtime.CompilerServices.DefaultInterpolatedStringHandler.AppendFormatted<char>(byref, char)
IL_0007: call instance string [System.Runtime]System.Runtime.CompilerServices.DefaultInterpolatedStringHandler.ToStringAndRelease()
IL_0008: call void [System.Console]System.Console.WriteLine(string)
IL_0009: nop

// [15 1 - 15 40]
IL_000a: ldstr "A"
IL_000b: ldloc.s V_9
IL_000c: call [mscorlib]System.Char.TryParse(string, char&)
IL_000d: stloc.s result

// [16 1 - 16 27]
IL_000e: ldloc.s result
IL_000f: call void [System.Console]System.Console.WriteLine(bool)
IL_0010: nop

// [18 1 - 18 41]
IL_0011: ldstr "2a"
IL_0012: ldloc.s V_9
IL_0013: call [mscorlib]System.Char.TryParse(string, char&)
IL_0014: stloc.s result

// [19 1 - 19 27]
IL_0015: ldloc.s result
IL_0016: call void [System.Console]System.Console.WriteLine(bool)
IL_0017: nop

// [22 1 - 22 27]
IL_0018: ldstr "23"
IL_0019: stloc.s inputString

// [23 1 - 23 47]
IL_001a: ldloc.s inputString
IL_001b: ldloc.s number
IL_001c: call [mscorlib]System.Int32.TryParse(string, int32&)
IL_001d: stloc.s V_10
IL_001e: brfalse.s IL_001f

// [24 1 - 24 2]
IL_001f: nop

// [25 5 - 25 31]
IL_0020: ldloc.s number
IL_0021: call void [System.Console]System.Console.WriteLine(int)
```

## В чем разница между Parse и TryParse?

Основное различие между Parse и TryParse заключается в том, что Parse создает исключение, если строка не содержит допустимого числа. Метод TryParse возвращает значение true, если строка содержит допустимое число

Parse использовать в случаях, где ожидается допустимый тип строки, который преобразуется в указанный примитивный тип.

TryParse в случаях, где ожидаются недопустимые входные значения, которые могут не успешно синтаксически анализироваться до соответствующего типа примитива.