String.Format Método

Namespace: System

Assemblies: mscorlib.dll, System.Runtime.dll

Converte o valor de objetos em cadeias de caracteres com base nos formatos especificados e os insere em outra cadeia de caracteres.

Se você não estiver familiarizado com o método `String.Format`, consulte a seção Introdução ao método String.Format para obter uma visão geral rápida.

Consulte a seção Comentários para ver a documentação geral do método `String.Format`.

Neste artigo

Sobrecargas

Format(String, Object)	Substitui um ou mais itens de formato em uma cadeia de caracteres pela representação de cadeia de caracteres de um objeto especificado.
Format(String, Object[])	Substitui o item de formato em uma cadeia de caracteres especificada pela representação de cadeia de caracteres de um objeto correspondente em uma matriz especificada.
Format(IFormatProvider,	Substitui o item ou itens de formato em uma cadeia de

String, Object)	caracteres especificada pela representação de cadeia de caracteres do objeto correspondente. Um parâmetro fornece informações de formatação específicas da cultura.
Format(IFormatProvider, String, Object[])	Substitui os itens de formato em uma cadeia de caracteres pelas representações cadeia de caracteres de objetos correspondentes em uma matriz especificada. Um parâmetro fornece informações de formatação específicas da cultura.
Format(String, Object, Object)	Substitui os itens de formato em uma cadeia de caracteres pela representação de cadeia de caracteres de dois objetos especificados.
Format(IFormatProvider, String, Object, Object)	Substitui os itens de formato em uma cadeia de caracteres pela representação de cadeia de caracteres de dois objetos especificados. Um parâmetro fornece informações de formatação específicas da cultura.
Format(String, Object, Object, Object)	Substitui os itens de formato em uma cadeia de caracteres pela representação de cadeia de caracteres de três objetos especificados.
Format(IFormatProvider, String, Object, Object, Object)	Substitui os itens de formato em uma cadeia de caracteres pela representação de cadeia de caracteres de três objetos especificados. Um parâmetro fornece informações de formatação específicas da cultura.

Exemplos

Vários exemplos que chamam o Format método são intercalados na seção de comentários deste artigo.

Você também pode baixar um conjunto completo de `String.Format` exemplos, que são incluídos em um projeto .NET Core para C#.

Veja a seguir alguns dos exemplos incluídos no artigo:

Criar uma cadeia de formato

Inserindo uma cadeia de caracteres
O item de formato
Itens de formato que têm o mesmo índice

Saída formatada de controle

Tornar as cadeias de caracteres de formato sensíveis à cultura

Personalizar a operação de formatação

Comentários

> ⓘ **Importante**
>
> Em vez de chamar o método **String.Format** ou usar **cadeias de caracteres de formato de composição**, é possível usar *cadeias de caracteres interpoladas* quando a linguagem é compatível com eles. Uma cadeia de caracteres interpolada é uma cadeia de caracteres que contém *expressões interpoladas*. Cada expressão interpolada é resolvida com o valor da expressão e incluída na cadeia de caracteres resultante quando a cadeia de caracteres é atribuída. Para saber mais, consulte o tópico **Interpolação de cadeia de caracteres (Referência do C#)** ou **Cadeias de caracteres interpoladas (Referência do Visual Basic)**.

Nesta seção:

Operações de formatação personalizadas
String. Format Q & A

Introdução ao método String. Format

Use String.Format se você precisar inserir o valor de um objeto, uma variável ou uma expressão em outra cadeia de caracteres. Por exemplo, você pode inserir o valor de um Decimal valor em uma cadeia de caracteres para exibi-lo ao usuário como uma única cadeia de caracteres:

C# Copiar Executar

```
Decimal pricePerOunce = 17.36m;
String s = String.Format("The current price is {0} per ounce.",
                         pricePerOunce);
Console.WriteLine(s);
// Result: The current price is 17.36 per ounce.
```

E você pode controlar a formatação desse valor:

C# Copiar

```
Decimal pricePerOunce = 17.36m;
String s = String.Format("The current price is {0:C2} per ounce.",
                         pricePerOunce);
Console.WriteLine(s);
// Result if current culture is en-US:
//      The current price is $17.36 per ounce.
```

Além da formatação, você também pode controlar o alinhamento e o espaçamento.

Inserir uma cadeia de caracteres

String.Format começa com uma cadeia de caracteres de formato, seguida por um ou mais objetos ou expressões que serão convertidas em cadeias de caracteres e inseridas em um local especificado na cadeia de caracteres de formato. Por exemplo:

C# Copiar Executar

```
decimal temp = 20.4m;
string s = String.Format("The temperature is {0}°C.", temp);
Console.WriteLine(s);
// Displays 'The temperature is 20.4°C.'
```

O `{0}` na cadeia de caracteres de formato é um item de formato. `0` é o índice do objeto cujo valor da cadeia de caracteres será inserido nessa posição. (Os índices começam em 0.) Se o objeto a ser inserido não for uma cadeia de caracteres, seu `ToString` método será chamado para convertê-lo em um antes de inseri-lo na cadeia de caracteres de resultado.

Veja outro exemplo que usa dois itens de formato e dois objetos na lista de objetos:

C# Copiar

```
string s = String.Format("At {0}, the temperature is {1}°C.",
                         DateTime.Now, 20.4);
Console.WriteLine(s);
// Output similar to: 'At 4/10/2015 9:29:41 AM, the temperature is 20.4°C.'
```

Você pode ter tantos itens de formato quanto quantos objetos forem desejados na lista de objetos, desde que o índice de cada item de formato tenha um objeto correspondente na lista de objetos. Você também não precisa se preocupar com qual sobrecarga você chama; o compilador selecionará o apropriado para você.

Formatação de controle

Você pode seguir o índice em um item de formato com uma cadeia de caracteres de formato para controlar como um objeto é formatado. Por exemplo, `{0:d}` aplica a cadeia de caracteres de formato "d" ao primeiro objeto na lista de objetos. Aqui está um exemplo com um único objeto e dois itens de formato:

C# Copiar

```
string s = String.Format("It is now {0:d} at {0:t}", DateTime.Now);
Console.WriteLine(s);
// Output similar to: 'It is now 4/10/2015 at 10:04 AM'
```

Vários tipos oferecem suporte A cadeias de caracteres de formato, incluindo todos os tipos numéricos (cadeias de caracteres de formato padrão e personalizado), todas as datas e horas (cadeias de caracteres de formato padrão e personalizadas) e intervalos de tempo (cadeias de caracteres de formato padrão e personalizadas), todos os tipos de enumeração e GUIDs. Você também pode adicionar suporte para cadeias de caracteres de formato aos seus próprios tipos.

Espaçamento de controle

Você pode definir a largura da cadeia de caracteres que é inserida na cadeia de caracteres de resultado usando `{0,12}` uma sintaxe como, que insere uma cadeia de 12 caracteres. Nesse caso, a representação de cadeia de caracteres do primeiro objeto é alinhada à direita no campo de 12 caracteres. (Se a representação de cadeia de caracteres do primeiro objeto tiver mais de 12 caracteres de comprimento, no entanto, a largura do campo preferencial será ignorada e toda a cadeia de caracteres será inserida na cadeia de caracteres de resultado.)

O exemplo a seguir define um campo de 6 caracteres para manter a cadeia de caracteres "Year" e algumas cadeias de ano, bem como um campo de 15 caracteres para conter a cadeia de caracteres "população" e alguns dados populacionais. Observe que os caracteres estão alinhados à direita no campo.

C# Copiar

```
      int[] years = { 2013, 2014, 2015 };
      int[] population = { 1025632, 1105967, 1148203 };
      var sb = new System.Text.StringBuilder();
      sb.Append(String.Format("{0,6} {1,15}\n\n", "Year", "Population"));
      for (int index = 0; index < years.Length; index++)
         sb.Append(String.Format("{0,6} {1,15:N0}\n", years[index],
population[index]));

      Console.WriteLine(sb);

      // Result:
      //      Year      Population
      //
      //      2013       1,025,632
      //      2014       1,105,967
      //      2015       1,148,203
```

Alinhamento de controle

Por padrão, as cadeias de caracteres são alinhadas à direita no campo se você especificar uma largura de campo. Para alinhar as cadeias de caracteres em um campo, você precede a largura do campo com um sinal negativo, como `{0,-12}` para definir um campo alinhado à esquerda de 12 caracteres.

O exemplo a seguir é semelhante ao anterior, exceto que ele alinha os rótulos e os dados à esquerda.

C# Copiar

```
int[] years = { 2013, 2014, 2015 };
int[] population = { 1025632, 1105967, 1148203 };
String s = String.Format("{0,-10} {1,-10}\n\n", "Year", "Population");
```

```
for(int index = 0; index < years.Length; index++)
   s += String.Format("{0,-10} {1,-10:N0}\n",
                      years[index], population[index]);
Console.WriteLine($"\n{s}");
// Result:
//    Year       Population
//
//    2013       1,025,632
//    2014       1,105,967
//    2015       1,148,203
```

String.Format usa o recurso de formatação composta. Para obter mais informações, veja Formatação de composição.

Qual método devo chamar?

Para	Call
Formate um ou mais objetos usando as convenções da cultura atual.	Exceto para as sobrecargas que incluem um `provider` parâmetro, as Format sobrecargas restantes incluem um String parâmetro seguido por um ou mais parâmetros de objeto. Por isso, você não precisa determinar qual Format sobrecarga você pretende chamar. O seu compilador de linguagem seleciona a sobrecarga apropriada entre as sobrecargas que não têm um `provider` parâmetro, com base na sua lista de argumentos. Por exemplo, se a sua lista de argumentos tiver cinco argumentos, o compilador chamará o Format(String, Object[]) método.
Formate um ou mais objetos usando as convenções de uma cultura específica.	Cada Format sobrecarga que começa com um `provider` parâmetro é seguida por um String parâmetro e um ou mais parâmetros de objeto. Por isso, você não precisa determinar qual Format sobrecarga específica você pretende chamar. O seu compilador de linguagem seleciona a sobrecarga apropriada entre as sobrecargas que têm um `provider` parâmetro, com base na sua lista de argumentos. Por exemplo, se a sua lista de argumentos tiver cinco argumentos, o compilador chamará o Format(IFormatProvider, String, Object[]) método.
Execute uma operação de formatação personalizada com uma ICustomFormatter implementação ou uma IFormattable implementação.	Qualquer uma das quatro sobrecargas com um `provider` parâmetro. O compilador seleciona a sobrecarga apropriada entre as sobrecargas que têm um `provider` parâmetro, com base na sua lista de argumentos.

O método Format resumido

Cada sobrecarga do Format método usa o recurso de formatação composta para incluir espaços reservados indexados com base em zero, chamados de *itens de formato*, em uma cadeia de caracteres de formato composto. Em tempo de execução, cada item de formato é substituído pela representação de cadeia de caracteres do argumento correspondente em uma lista de parâmetros. Se o valor do argumento for `null` , o item de formato será substituído por String.Empty . Por exemplo, a chamada a seguir para o Format(String, Object, Object, Object) método inclui uma cadeia de caracteres de formato com três itens de formato,, {0} {1} e {2} , e uma lista de argumentos com três itens.

C# Copiar

```
DateTime dat = new DateTime(2012, 1, 17, 9, 30, 0);
string city = "Chicago";
int temp = -16;
string output = String.Format("At {0} in {1}, the temperature was {2} de-
grees.",
                              dat, city, temp);
Console.WriteLine(output);
// The example displays output like the following:
//    At 1/17/2012 9:30:00 AM in Chicago, the temperature was -16 degrees.
```

O item de formato

Um item de formato tem essa sintaxe:

Copiar

```
{index[,alignment][:formatString]}
```

Os colchetes denotam elementos opcionais. As chaves de abertura e fechamento são necessárias. (Para incluir uma chave literal de abertura ou fechamento na cadeia de caracteres de formato, consulte a seção chaves de escape no artigo formatação composta .)

Por exemplo, um item de formato para formatar um valor de moeda pode ser semelhante a este:

C# Copiar

```
var value = String.Format("{0,-10:C}", 126347.89m);
Console.WriteLine(value);
```

Um item de formato tem os seguintes elementos:

index
O índice de base zero do argumento cuja representação de cadeia de caracteres deve ser incluída nessa posição na cadeia de caracteres. Se esse argumento for `null` , uma cadeia de caracteres vazia será incluída nessa posição na cadeia de caracteres.

sintonia
Opcional. Um inteiro assinado que indica o comprimento total do campo no qual o argumento é inserido e se está alinhado à direita (um inteiro positivo) ou alinhado à esquerda (um inteiro negativo). Se você omitir o *alinhamento*, a representação da cadeia de caracteres do argumento correspondente será inserida em um campo sem espaços à esquerda ou à direita.

Se o valor de *Alignment* for menor que o comprimento do argumento a ser inserido, o *alinhamento* será ignorado e o comprimento da representação da cadeia de caracteres do argumento será usado como a largura do campo.

formatString
Opcional. Uma cadeia de caracteres que especifica o formato da cadeia de caracteres de resultado do argumento correspondente. Se você omitisse *FormatString*, o método sem parâmetros do argumento correspondente `ToString` será chamado para produzir sua representação de cadeia de caracteres. Se você especificar *FormatString*, o argumento referenciado pelo item de formato deverá implementar a IFormattable interface. Os tipos que oferecem suporte a cadeias de caracteres de formato incluem:

- Todos os tipos de ponto flutuante e integral. (Consulte cadeias de caracteres de formato numérico padrão e cadeias de caracteres de formato numérico personalizado.)
- DateTime e DateTimeOffset. (Consulte cadeias de caracteres de formato de data e hora padrão e cadeias de caracteres de formato de data e hora personalizadas.)
- Todos os tipos de enumeração. (Consulte cadeias de caracteres de formato de enumeração.)
- valores TimeSpan. (Consulte cadeias de caracteres de formato TimeSpan padrão e cadeias de caracteres de formato TimeSpan personalizado.)
- GUIDs. (Consulte o Guid.ToString(String) método.)

No entanto, observe que qualquer tipo personalizado pode implementar IFormattable ou estender a implementação de um tipo existente IFormattable .

O exemplo a seguir usa `alignment` os `formatString` argumentos e para produzir a saída formatada.

C# Copiar Executar

```
// Create array of 5-tuples with population data for three U.S. cities,
1940-1950.
Tuple<string, DateTime, int, DateTime, int>[] cities =
    { Tuple.Create("Los Angeles", new DateTime(1940, 1, 1), 1504277,
                   new DateTime(1950, 1, 1), 1970358),
      Tuple.Create("New York", new DateTime(1940, 1, 1), 7454995,
                   new DateTime(1950, 1, 1), 7891957),
      Tuple.Create("Chicago", new DateTime(1940, 1, 1), 3396808,
                   new DateTime(1950, 1, 1), 3620962),
      Tuple.Create("Detroit", new DateTime(1940, 1, 1), 1623452,
                   new DateTime(1950, 1, 1), 1849568) };

// Display header
var header = String.Format("{0,-12}{1,8}{2,12}{1,8}{2,12}{3,14}\n",
                           "City", "Year", "Population", "Change (%)");
Console.WriteLine(header);
foreach (var city in cities) {
   var output = String.Format("{0,-12}{1,8:yyyy}{2,12:N0}{3,8:yyyy}{4,12:N0}
{5,14:P1}",
                           city.Item1, city.Item2, city.Item3, city.Item4,
city.Item5,
                           (city.Item5 - city.Item3)/ (double)city.Item3);
   Console.WriteLine(output);
}
// The example displays the following output:
//    City            Year  Population    Year  Population    Change (%)
//
//    Los Angeles     1940   1,504,277    1950   1,970,358        31.0 %
//    New York        1940   7,454,995    1950   7,891,957         5.9 %
//    Chicago         1940   3,396,808    1950   3,620,962         6.6 %
//    Detroit         1940   1,623,452    1950   1,849,568        13.9 %
```

Como os argumentos são formatados

Os itens de formato são processados sequencialmente a partir do início da cadeia de caracteres. Cada item de formato tem um índice que corresponde a um objeto na lista de argumentos do método. O Format método recupera o argumento e deriva sua representação de cadeia de caracteres da seguinte maneira:

- Se o argumento for `null` , o método será inserido String.Empty na cadeia de caracteres de resultado. Você não precisa se preocupar em manipular um NullReferenceException para argumentos nulos.

- Se você chamar a Format(IFormatProvider, String, Object[]) sobrecarga e a `provider` implementação do objeto IFormatProvider.GetFormat retornar uma

implementação não nula ICustomFormatter , o argumento será passado para seu ICustomFormatter.Format(String, Object, IFormatProvider) método. Se o item de formato incluir um argumento *FormatString* , ele será passado como o primeiro argumento para o método. Se a ICustomFormatter implementação estiver disponível e produzir uma cadeia de caracteres não nula, essa cadeia de caracteres será retornada como a representação de cadeia de caracteres do argumento; caso contrário, a próxima etapa é executada.

- Se o argumento implementar a IFormattable interface, sua IFormattable.ToString implementação será chamada.

- O método sem parâmetros do argumento `ToString` , que substitui ou herda de uma implementação de classe base, é chamado.

Para obter um exemplo que intercepta chamadas para o ICustomFormatter.Format método e permite que você veja quais informações o Format método passa para um método de formatação para cada item de formato em uma cadeia de caracteres de formato composto, consulte exemplo: um provedor de interceptação e um formatadorromano.

Para obter mais informações, consulte a seção ordem de processamento no artigo formatação composta .

Itens de formato que têm o mesmo índice

O Format método gera uma FormatException exceção se o índice de um item de índice for maior ou igual ao número de argumentos na lista de argumentos. No entanto, `format` o pode incluir mais itens de formato do que os argumentos, desde que vários itens de formato tenham o mesmo índice. Na chamada ao Format(String, Object) método no exemplo a seguir, a lista de argumentos tem um único argumento, mas a cadeia de caracteres de formato inclui dois itens de formato: um exibe o valor decimal de um número e o outro exibe seu valor hexadecimal.

C# Copiar Executar

```
short[] values= { Int16.MinValue, -27, 0, 1042, Int16.MaxValue };
Console.WriteLine("{0,10}  {1,10}\n", "Decimal", "Hex");
foreach (short value in values)
{
   string formatString = String.Format("{0,10:G}: {0,10:X}", value);
   Console.WriteLine(formatString);
}
// The example displays the following output:
//       Decimal         Hex
//
//        -32768:       8000
```

```
//           -27:       FFE5
//             0:          0
//          1042:        412
//         32767:       7FFF
```

Formato e cultura

Em geral, os objetos na lista de argumentos são convertidos em suas representações de cadeia de caracteres usando as convenções da cultura atual, que é retornada pela CultureInfo.CurrentCulture propriedade. Você pode controlar esse comportamento chamando uma das sobrecargas do Format que inclui um `provider` parâmetro. O `provider` parâmetro é uma IFormatProvider implementação que fornece informações de formatação personalizadas e específicas de cultura que são usadas para moderar o processo de formatação.

A IFormatProvider interface tem um único membro, GetFormat que é responsável por retornar o objeto que fornece informações de formatação. O .NET tem três IFormatProvider implementações que fornecem formatação específica de cultura:

- CultureInfo. Seu GetFormat método retorna um objeto específico de cultura NumberFormatInfo para formatar valores numéricos e um objeto específico de cultura DateTimeFormatInfo para formatar valores de data e hora.
- DateTimeFormatInfo, que é usado para a formatação específica de cultura de valores de data e hora. Seu GetFormat método retorna a si mesmo.
- NumberFormatInfo, que é usado para a formatação específica de cultura de valores numéricos. Sua GetFormat propriedade retorna a si mesma.

Operações de formatação personalizadas

Você também pode chamar qualquer uma das sobrecargas do Format método que têm um `provider` parâmetro do tipo IFormatProvider para executar operações de formatação personalizadas. Por exemplo, você pode formatar um inteiro como um número de identificação ou como um número de telefone. Para executar a formatação personalizada, o `provider` argumento deve implementar as IFormatProvider ICustomFormatter interfaces e. Quando o Format método passa uma ICustomFormatter implementação como o `provider` argumento, o Format método chama sua IFormatProvider.GetFormat implementação e solicita um objeto do tipo ICustomFormatter . Em seguida, ele chama o ICustomFormatter método do objeto retornado Format para formatar cada item de formato na cadeia de caracteres composta passada para ele.

Para obter mais informações sobre como fornecer soluções de formatação personalizadas, consulte como: definir e usar provedores de formato numérico personalizado e ICustomFormatter . Para obter um exemplo que converte inteiros em números personalizados formatados, consulte exemplo: uma operação de formatação personalizada. Para obter um exemplo que converte bytes não assinados em algarismos romanos, consulte exemplo: um provedor de interceptação e um formatadorromano.

Exemplo: uma operação de formatação personalizada

Este exemplo define um provedor de formato que formata um valor inteiro como um número de conta de cliente no formato x-xxxxx-XX.

C# Copiar Executar

```
using System;

public class TestFormatter
{
   public static void Main()
   {
      int acctNumber = 79203159;
      Console.WriteLine(String.Format(new CustomerFormatter(), "{0}", acct-
Number));
      Console.WriteLine(String.Format(new CustomerFormatter(), "{0:G}", ac-
ctNumber));
      Console.WriteLine(String.Format(new CustomerFormatter(), "{0:S}", ac-
ctNumber));
      Console.WriteLine(String.Format(new CustomerFormatter(), "{0:P}", ac-
ctNumber));
      try {
         Console.WriteLine(String.Format(new CustomerFormatter(), "{0:X}",
acctNumber));
      }
      catch (FormatException e) {
         Console.WriteLine(e.Message);
      }
   }
}

public class CustomerFormatter : IFormatProvider, ICustomFormatter
{
   public object GetFormat(Type formatType)
   {
      if (formatType == typeof(ICustomFormatter))
         return this;
      else
         return null;
   }

   public string Format(string format,
                        object arg,
                        IFormatProvider formatProvider)
```

```
    {
       if (! this.Equals(formatProvider))
       {
          return null;
       }
       else
       {
          if (String.IsNullOrEmpty(format))
             format = "G";

          string customerString = arg.ToString();
          if (customerString.Length < 8)
             customerString = customerString.PadLeft(8, '0');

          format = format.ToUpper();
          switch (format)
          {
             case "G":
                return customerString.Substring(0, 1) + "-" +
                                      customerString.Substring(1, 5) + "-" +
                                      customerString.Substring(6);
             case "S":
                return customerString.Substring(0, 1) + "/" +
                                      customerString.Substring(1, 5) + "/" +
                                      customerString.Substring(6);
             case "P":
                return customerString.Substring(0, 1) + "." +
                                      customerString.Substring(1, 5) + "." +
                                      customerString.Substring(6);
             default:
                throw new FormatException(
                          String.Format("The '{0}' format specifier is not
supported.", format));
          }
       }
    }
}
// The example displays the following output:
//       7-92031-59
//       7-92031-59
//       7/92031/59
//       7.92031.59
//       The 'X' format specifier is not supported.
```

Exemplo: um provedor de interceptação e um formatador Romano

Este exemplo define um provedor de formato personalizado que implementa ICustomFormatter as IFormatProvider interfaces e para fazer duas coisas:

- Ele exibe os parâmetros passados para sua ICustomFormatter.Format implementação. Isso nos permite ver quais parâmetros o Format(IFormatProvider, String, Object[]) método está passando para a implementação de formatação

personalizada para cada objeto que ele tenta Formatar. Isso pode ser útil quando você estiver Depurando seu aplicativo.

- Se o objeto a ser formatado for um valor de byte não assinado que deve ser formatado usando a cadeia de caracteres de formato padrão "R", o formatador personalizado formatará o valor numérico como um numeral romano.

C# Copiar

```
using System;
using System.Globalization;

public class InterceptProvider : IFormatProvider, ICustomFormatter
{
   public object GetFormat(Type formatType)
   {
      if (formatType == typeof(ICustomFormatter))
         return this;
      else
         return null;
   }

   public string Format(String format, Object obj, IFormatProvider provider)
   {
      // Display information about method call.
      string formatString = format ?? "<null>";
      Console.WriteLine("Provider: {0}, Object: {1}, Format String: {2}",
                        provider.GetType().Name, obj ?? "<null>", formatS-
tring);

      if (obj == null) return String.Empty;

      // If this is a byte and the "R" format string, format it with Roman
numerals.
      if (obj is Byte && formatString.ToUpper().Equals("R")) {
         Byte value = (Byte) obj;
         int remainder;
         int result;
         String returnString = String.Empty;

         // Get the hundreds digit(s)
         result = Math.DivRem(value, 100, out remainder);
         if (result > 0)
            returnString = new String('C', result);
         value = (Byte) remainder;
         // Get the 50s digit
         result = Math.DivRem(value, 50, out remainder);
         if (result == 1)
            returnString += "L";
         value = (Byte) remainder;
         // Get the tens digit.
         result = Math.DivRem(value, 10, out remainder);
         if (result > 0)
            returnString += new String('X', result);
         value = (Byte) remainder;
```

```
         // Get the fives digit.
         result = Math.DivRem(value, 5, out remainder);
         if (result > 0)
            returnString += "V";
         value = (Byte) remainder;
         // Add the ones digit.
         if (remainder > 0)
            returnString += new String('I', remainder);

         // Check whether we have too many X characters.
         int pos = returnString.IndexOf("XXXX");
         if (pos >= 0) {
            int xPos = returnString.IndexOf("L");
            if (xPos >= 0 & xPos == pos - 1)
               returnString = returnString.Replace("LXXXX", "XC");
            else
               returnString = returnString.Replace("XXXX", "XL");
         }
         // Check whether we have too many I characters
         pos = returnString.IndexOf("IIII");
         if (pos >= 0)
            if (returnString.IndexOf("V") >= 0)
               returnString = returnString.Replace("VIIII", "IX");
            else
               returnString = returnString.Replace("IIII", "IV");

         return returnString;
      }

      // Use default for all other formatting.
      if (obj is IFormattable)
         return ((IFormattable) obj).ToString(format,
CultureInfo.CurrentCulture);
      else
         return obj.ToString();
   }
}

public class Example
{
   public static void Main()
   {
      int n = 10;
      double value = 16.935;
      DateTime day = DateTime.Now;
      InterceptProvider provider = new InterceptProvider();
      Console.WriteLine(String.Format(provider, "{0:N0}: {1:C2} on {2:d}\n",
n, value, day));
      Console.WriteLine(String.Format(provider, "{0}: {1:F}\n", "Today: ",
                                      (DayOfWeek) DateTime.Now.DayOfWeek));
      Console.WriteLine(String.Format(provider, "{0:X}, {1}, {2}\n",
                                      (Byte) 2, (Byte) 12, (Byte) 199));
      Console.WriteLine(String.Format(provider, "{0:R}, {1:R}, {2:R}\n",
                                      (Byte) 2, (Byte) 12, (Byte) 199));
   }
}
// The example displays the following output:
//    Provider: InterceptProvider, Object: 10, Format String: N0
```

```
//    Provider: InterceptProvider, Object: 16.935, Format String: C2
//    Provider: InterceptProvider, Object: 1/31/2013 6:10:28 PM, Format
String: d
//    10: $16.94 on 1/31/2013
//
//    Provider: InterceptProvider, Object: Today: , Format String: <null>
//    Provider: InterceptProvider, Object: Thursday, Format String: F
//    Today: : Thursday
//
//    Provider: InterceptProvider, Object: 2, Format String: X
//    Provider: InterceptProvider, Object: 12, Format String: <null>
//    Provider: InterceptProvider, Object: 199, Format String: <null>
//    2, 12, 199
//
//    Provider: InterceptProvider, Object: 2, Format String: R
//    Provider: InterceptProvider, Object: 12, Format String: R
//    Provider: InterceptProvider, Object: 199, Format String: R
//    II, XII, CXCIX
```

String. Format Q & A

Por que você recomenda a interpolação de cadeia de caracteres sobre chamadas para o `String.Format` método?

A interpolação da cadeia de caracteres é:

- Mais flexível. Ele pode ser usado em qualquer cadeia de caracteres sem a necessidade de uma chamada para um método que ofereça suporte à formatação composta. Caso contrário, você precisa chamar o Format método ou outro método que dá suporte à formatação composta, como Console.WriteLine ou StringBuilder.AppendFormat .

- Mais legível. Como a expressão a ser inserida em uma cadeia de caracteres aparece na expressão interpolada em vez de em uma lista de argumentos, as cadeias de caracteres interpoladas são muito mais fáceis de codificar e de ler. Devido à sua maior legibilidade, cadeias de caracteres interpoladas podem substituir não apenas chamadas para métodos de formato composto, mas também podem ser usadas em operações de concatenação de cadeia de caracteres para produzir um código mais conciso e mais preciso.

Uma comparação dos dois exemplos de código a seguir ilustra a superioridade de cadeias de caracteres interpoladas sobre a concatenação de cadeias e chamadas para métodos de formatação compostos. O uso de várias operações de concatenação de cadeia de caracteres no exemplo a seguir produz código Detalhado e difícil de ler.

C# Copiar

```
  string[] names = { "Balto", "Vanya", "Dakota", "Samuel", "Koani", "Yiska",
"Yuma" };
  string output = names[0] + ", " + names[1] + ", " + names[2] + ", " +
                  names[3] + ", " + names[4] + ", " + names[5] + ", " +
                  names[6];

  output += "\n";
  var date = DateTime.Now;
  output += String.Format("It is {0:t} on {0:d}. The day of the week is
{1}.",
                          date, date.DayOfWeek);
  Console.WriteLine(output);
  // The example displays the following output:
  //     Balto, Vanya, Dakota, Samuel, Koani, Yiska, Yuma
  //     It is 10:29 AM on 1/8/2018. The day of the week is Monday.
```

Por outro lado, o uso de cadeias de caracteres interpoladas no exemplo a seguir produz um código muito mais claro e mais conciso do que a instrução de concatenação de cadeia de caracteres e a chamada para o Format método no exemplo anterior.

C# Copiar

```
  string[] names = { "Balto", "Vanya", "Dakota", "Samuel", "Koani", "Yiska",
"Yuma" };
  string output = $"{names[0]}, {names[1]}, {names[2]}, {names[3]},
{names[4]}, " +
                  $"{names[5]}, {names[6]}";

  var date = DateTime.Now;
  output += $"\nIt is {date:t} on {date:d}. The day of the week is
{date.DayOfWeek}.";
  Console.WriteLine(output);
  // The example displays the following output:
  //     Balto, Vanya, Dakota, Samuel, Koani, Yiska, Yuma
  //     It is 10:29 AM on 1/8/2018. The day of the week is Monday.
```

Onde posso encontrar uma lista das cadeias de caracteres de formato predefinidas que podem ser usadas com itens de formato?

- Para todos os tipos de ponto flutuante e integral, consulte cadeias de caracteres de formato numérico padrão e cadeias de caracteres de formato numérico personalizado.
- Para valores de data e hora, consulte cadeias de caracteres de formato padrão de data e hora e cadeias de caracteres de formato de data e hora personalizadas.
- Para valores de enumeração, consulte cadeias de formato de enumeração.

- Para TimeSpan valores, consulte cadeias de caracteres de formato standard TimeSpan e cadeias de caracteres de formato TimeSpan personalizado.
- Para Guid valores, consulte a seção comentários da Guid.ToString(String) página de referência.

Como fazer controlar o alinhamento das cadeias de caracteres de resultado que substituem os itens de formato?

A sintaxe geral de um item de formato é:

Copiar

```
{index[,alignment][: formatString]}
```

em que *Alignment* é um inteiro assinado que define a largura do campo. Se esse valor for negativo, o texto no campo será alinhado à esquerda. Se for positivo, o texto será alinhado à direita.

Como fazer controlar o número de dígitos após o separador decimal?

Todas as cadeias de caracteres de formato numérico padrão , exceto "D" (que são usadas somente com inteiros), "G", "R" e "X" permitem um especificador de precisão que define o número de dígitos decimais na cadeia de caracteres de resultado. O exemplo a seguir usa cadeias de caracteres de formato numérico padrão para controlar o número de dígitos decimais na cadeia de caracteres de resultado.

C# Copiar

```
object[] values = { 1603, 1794.68235, 15436.14 };
string result;
foreach (var value in values) {
   result = String.Format("{0,12:C2}   {0,12:E3}   {0,12:F4}   {0,12:N3}
{1,12:P2}\n",
                          Convert.ToDouble(value), Convert.ToDouble(value) /
10000);
   Console.WriteLine(result);
}
// The example displays output like the following:
//       $1,603.00     1.603E+003      1603.0000      1,603.000       16.03
%
//
//       $1,794.68     1.795E+003      1794.6824      1,794.682       17.95
%
//
```

```
//       $15,436.14      1.544E+004      15436.1400      15,436.140       154.36
%
```

Se você estiver usando uma cadeia de caracteres de formato numérico personalizado, use o especificador de formato "0" para controlar o número de dígitos decimais na cadeia de caracteres de resultado, como mostra o exemplo a seguir.

C# Copiar Executar

```
decimal value = 16309.5436m;
string result = String.Format("{0,12:#.00000} {0,12:0,000.00}
{0,12:000.00#}",
                              value);
Console.WriteLine(result);
// The example displays the following output:
//        16309.54360    16,309.54    16309.544
```

Como fazer controlar o número de dígitos integrais?

Por padrão, as operações de formatação exibem apenas dígitos integrais diferentes de zero. Se você estiver Formatando inteiros, poderá usar um especificador de precisão com as cadeias de caracteres de formato "D" e "X" para controlar o número de dígitos.

C# Copiar Executar

```
int value = 1326;
string result = String.Format("{0,10:D6} {0,10:X8}", value);
Console.WriteLine(result);
// The example displays the following output:
//     001326   0000052E
```

Você pode preencher um número inteiro ou de ponto flutuante com zeros à esquerda para produzir uma cadeia de caracteres de resultado com um número especificado de dígitos integrais usando o especificador de formato numérico personalizado"0", como mostra o exemplo a seguir.

C# Copiar Executar

```
int value = 16342;
string result = String.Format("{0,18:00000000} {0,18:00000000.000}
{0,18:000,0000,000.0}",
                              value);
Console.WriteLine(result);
// The example displays the following output:
//           00016342       00016342.000    0,000,016,342.0
```

Quantos itens posso incluir na lista de formatos?

Não há nenhum limite prático. O segundo parâmetro do Format(IFormatProvider, String, Object[]) método é marcado com o ParamArrayAttribute atributo, que permite que você inclua uma lista delimitada ou uma matriz de objetos como sua lista de formatos.

Como fazer incluir chaves literais ("{" e "}") na cadeia de caracteres de resultado?

Por exemplo, como impedir que a chamada de método a seguir emita uma FormatException exceção?

C# Copiar

```
result = String.Format("The text has {0} '{' characters and {1} '}' charac-
ters.",
                       nOpen, nClose);
```

Uma única chave de abertura ou de fechamento sempre é interpretada como o início ou o fim de um item de formato. Para ser interpretado literalmente, ele deve ser ignorado. Você sai de uma chave adicionando outra chave ("{{" e "}}" em vez de "{" e "}"), como na seguinte chamada de método:

C# Copiar Executar

```
string result;
int nOpen = 1;
int nClose = 2;
result = String.Format("The text has {0} '{{' characters and {1} '}}' cha-
racters.",
                       nOpen, nClose);
Console.WriteLine(result);
```

No entanto, até mesmo as chaves com escape são facilmente interpretadas. É recomendável que você inclua chaves na lista de formatos e use itens de formato para inseri-las na cadeia de caracteres de resultado, como mostra o exemplo a seguir.

C# Copiar Executar

```
string result;
int nOpen = 1;
int nClose = 2;
result = String.Format("The text has {0} '{1}' characters and {2} '{3}' cha-
racters.",
                       nOpen, "{", nClose, "}");
Console.WriteLine(result);
```

Por que minha chamada para o método String. Format lança um FormatException?

A causa mais comum da exceção é que o índice de um item de formato não corresponde a um objeto na lista de formatos. Geralmente, isso indica que você digitou informadamente os índices de itens de formato ou esqueceu de incluir um objeto na lista de formatos. A tentativa de incluir um caractere de chave esquerda ou direita sem escape também gera um FormatException . Ocasionalmente, a exceção é o resultado de um erros de digitação; por exemplo, um erro típico é digitar incorretamente "[" (o colchete esquerdo) em vez de "{" (a chave esquerda).

Se o método de formato (System. IFormatProvider, System. String, System. Object []) oferecer suporte a matrizes de parâmetros, por que meu código lança uma exceção quando uso uma matriz?

Por exemplo, o código a seguir gera uma FormatException exceção:

C# Copiar

```
Random rnd = new Random();
int[]  numbers = new int[4];
int total = 0;
for (int ctr = 0; ctr <= 2; ctr++) {
   int number = rnd.Next(1001);
   numbers[ctr] = number;
   total += number;
}
numbers[3] = total;
Console.WriteLine("{0} + {1} + {2} = {3}", numbers);
```

Esse é um problema da resolução de sobrecarga do compilador. Como o compilador não pode converter uma matriz de inteiros em uma matriz de objeto, ele trata a matriz de inteiros como um único argumento, de modo que chama o Format(String, Object) método. A exceção é gerada porque há quatro itens de formato, mas apenas um único item na lista de formatos.

Como nem Visual Basic nem C# podem converter uma matriz de inteiros em uma matriz de objeto, você mesmo precisa executar a conversão antes de chamar o Format(String, Object[]) método. O exemplo a seguir fornece uma implementação.

C# Copiar Executar

```
Random rnd = new Random();
int[]  numbers = new int[4];
int total = 0;
for (int ctr = 0; ctr <= 2; ctr++) {
   int number = rnd.Next(1001);
   numbers[ctr] = number;
   total += number;
}
numbers[3] = total;
object[] values = new object[numbers.Length];
numbers.CopyTo(values, 0);
Console.WriteLine("{0} + {1} + {2} = {3}", values);
```

Format(String, Object)

Substitui um ou mais itens de formato em uma cadeia de caracteres pela representação de cadeia de caracteres de um objeto especificado.

C# Copiar

```
public static string Format (string format, object? arg0);
```

Parâmetros

`format` String

Uma cadeia de caracteres de formato composto.

`arg0` Object

O objeto a ser formatado.

Retornos

String

Uma cópia do `format` na qual os itens de formato são substituídos pela representação de cadeia de caracteres de `arg0`.

Exceções

ArgumentNullException

`format` é `null`.

FormatException

O item de formato em `format` é inválido.

- ou -

O índice de um item de formato não é zero.

Comentários

> **Importante**
>
> Em vez de chamar o método **String.Format** ou usar **cadeias de caracteres de formato de composição**, é possível usar *cadeias de caracteres interpoladas* quando a linguagem é compatível com eles. Uma cadeia de caracteres interpolada é uma cadeia de caracteres que contém *expressões interpoladas*. Cada expressão interpolada é resolvida com o valor da expressão e incluída na cadeia de caracteres resultante quando a cadeia de caracteres é atribuída. Para saber mais, consulte o tópico **Interpolação de cadeia de caracteres (Referência do C#)** ou **Cadeias de caracteres interpoladas (Referência do Visual Basic)**.

Esse método usa o recurso de formatação composta para converter o valor de uma expressão em sua representação de cadeia de caracteres e inserir essa representação em uma cadeia de caracteres.

Entretanto, ao chamar o método **String.Format**, não é necessário se concentrar na sobrecarga específica que você deseja chamar. Em vez disso, é possível chamar o método com uma cadeia de caracteres de formato composto que inclui um ou mais itens de formato. Você atribui a cada item de formato um índice numérico. O primeiro índice começa em 0. Além da cadeia de caracteres inicial, sua chamada de método deve ter tantos argumentos adicionais quantos valores de índice. Por exemplo, uma cadeia de caracteres cujos itens de formato têm índices 0 e 1 deve ter 2 argumentos; uma com índices de 0 a 5 deve ter 6 argumentos. O compilador de linguagem, então, resolverá a chamada de método para uma sobrecarga específica do método **String.Format**.

Para obter uma documentação mais detalhada sobre como usar o método **String.Format**, consulte o Guia de Introdução ao método String.Format e Qual método chamar?.

Exemplo: Formatando um único argumento

O exemplo a seguir usa o Format(String, Object) método para inserir a idade de um indivíduo no meio de uma cadeia de caracteres.

C# Copiar Executar

```
DateTime birthdate = new DateTime(1993, 7, 28);
DateTime[] dates = { new DateTime(1993, 8, 16),
                     new DateTime(1994, 7, 28),
                     new DateTime(2000, 10, 16),
                     new DateTime(2003, 7, 27),
                     new DateTime(2007, 5, 27) };

foreach (DateTime dateValue in dates)
{
   TimeSpan interval = dateValue - birthdate;
   // Get the approximate number of years, without accounting for leap
years.
   int years = ((int) interval.TotalDays) / 365;
   // See if adding the number of years exceeds dateValue.
   string output;
   if (birthdate.AddYears(years) <= dateValue) {
      output = String.Format("You are now {0} years old.", years);
      Console.WriteLine(output);
   }
   else {
      output = String.Format("You are now {0} years old.", years - 1);
      Console.WriteLine(output);
   }
}
// The example displays the following output:
//       You are now 0 years old.
//       You are now 1 years old.
//       You are now 7 years old.
//       You are now 9 years old.
//       You are now 13 years old.
```

Aplica-se a

▶ .NET 6.0 preview 3 e outras versões

Format(String, Object[])

Substitui o item de formato em uma cadeia de caracteres especificada pela representação de cadeia de caracteres de um objeto correspondente em uma matriz especificada.

C# Copiar

```
public static string Format (string format, params object?[] args);
```

Parâmetros

format String

Uma cadeia de caracteres de formato composto.

args Object[]

Uma matriz de objetos que contém zero ou mais objetos a serem formatados.

Retornos

String

Uma cópia do `format` na qual os itens de formato foram substituídos pela representação de cadeia de caracteres dos objetos correspondentes no `args`.

Exceções

ArgumentNullException

`format` ou `args` é `null`.

FormatException

`format` é inválido.

- ou -

O índice de um item de formato é menor que zero, ou maior ou igual ao tamanho da matriz `args`.

Comentários

> **Importante**
>
> Em vez de chamar o método **String.Format** ou usar **cadeias de caracteres de formato de composição**, é possível usar *cadeias de caracteres interpoladas* quando a linguagem é compatível com eles. Uma cadeia de caracteres interpolada é uma cadeia de caracteres que contém *expressões interpoladas*. Cada expressão interpolada é resolvida com o valor da expressão e incluída na cadeia de caracteres resultante quando a cadeia de caracteres é atribuída. Para

saber mais, consulte o tópico **Interpolação de cadeia de caracteres (Referência do C#)** ou **Cadeias de caracteres interpoladas (Referência do Visual Basic)**.

Esse método usa o recurso de formatação composta para converter o valor de quatro ou mais expressões em suas representações de cadeia de caracteres e inserir essas representações em uma cadeia de caracteres. Como o `args` parâmetro é marcado com o System.ParamArrayAttribute atributo, você pode passar os objetos para o método como argumentos individuais ou como uma Object matriz.

Entretanto, ao chamar o método **String.Format**, não é necessário se concentrar na sobrecarga específica que você deseja chamar. Em vez disso, é possível chamar o método com uma cadeia de caracteres de formato composto que inclui um ou mais itens de formato. Você atribui a cada item de formato um índice numérico. O primeiro índice começa em 0. Além da cadeia de caracteres inicial, sua chamada de método deve ter tantos argumentos adicionais quantos valores de índice. Por exemplo, uma cadeia de caracteres cujos itens de formato têm índices 0 e 1 deve ter 2 argumentos; uma com índices de 0 a 5 deve ter 6 argumentos. O compilador de linguagem, então, resolverá a chamada de método para uma sobrecarga específica do método **String.Format**.

Para obter uma documentação mais detalhada sobre como usar o método **String.Format**, consulte o Guia de Introdução ao método String.Format e Qual método chamar?.

Exemplo: Formatar mais de três argumentos

Este exemplo cria uma cadeia de caracteres que contém dados sobre a temperatura alta e baixa em uma data específica. A cadeia de caracteres de formato composto tem cinco itens de formato no exemplo em C# e seis no exemplo de Visual Basic. Dois dos itens de formato definem a largura da representação de cadeia de caracteres do seu valor correspondente e o primeiro item de formato também inclui uma cadeia de caracteres de formato de data e hora padrão.

C# Copiar

```
DateTime date1 = new DateTime(2009, 7, 1);
TimeSpan hiTime = new TimeSpan(14, 17, 32);
decimal hiTemp = 62.1m;
TimeSpan loTime = new TimeSpan(3, 16, 10);
decimal loTemp = 54.8m;

string result1 = String.Format("Temperature on {0:d}:\n{1,11}: {2} de-
grees (hi)\n{3,11}: {4} degrees (lo)",
                               date1, hiTime, hiTemp, loTime, loTemp);
Console.WriteLine(result1);
```

```
Console.WriteLine();

string result2 = String.Format("Temperature on {0:d}:\n{1,11}: {2} de-
grees (hi)\n{3,11}: {4} degrees (lo)",
                               new object[] { date1, hiTime, hiTemp, lo-
Time, loTemp });
Console.WriteLine(result2);
// The example displays output like the following:
//       Temperature on 7/1/2009:
//          14:17:32: 62.1 degrees (hi)
//          03:16:10: 54.8 degrees (lo)
//       Temperature on 7/1/2009:
//          14:17:32: 62.1 degrees (hi)
//          03:16:10: 54.8 degrees (lo)
```

Você também pode passar os objetos a serem formatados como uma matriz em vez de como uma lista de argumentos.

C# Copiar Executar

```
using System;

public class CityInfo
{
   public CityInfo(String name, int population, Decimal area, int year)
   {

      this.Name = name;
      this.Population = population;
      this.Area = area;
      this.Year = year;
   }

   public readonly String Name;
   public readonly int Population;
   public readonly Decimal Area;
   public readonly int Year;
}

public class Example
{
   public static void Main()
   {
      CityInfo nyc2010 = new CityInfo("New York", 8175133, 302.64m,
2010);
      ShowPopulationData(nyc2010);
      CityInfo sea2010 = new CityInfo("Seattle", 608660, 83.94m, 2010);
      ShowPopulationData(sea2010);
   }

   private static void ShowPopulationData(CityInfo city)
   {
      object[] args = { city.Name, city.Year, city.Population, city.Area
};
      String result = String.Format("{0} in {1}: Population {2:N0}, Area
{3:N1} sq. feet",
```

```
                                        args);
      Console.WriteLine(result);
   }
}
// The example displays the following output:
//       New York in 2010: Population 8,175,133, Area 302.6 sq. feet
//       Seattle in 2010: Population 608,660, Area 83.9 sq. feet
```

Aplica-se a

▶ .NET 6.0 preview 3 e outras versões

Format(IFormatProvider, String, Object)

Substitui o item ou itens de formato em uma cadeia de caracteres especificada pela representação de cadeia de caracteres do objeto correspondente. Um parâmetro fornece informações de formatação específicas da cultura.

C# Copiar

```
public static string Format (IFormatProvider? provider, string format,
object? arg0);
```

Parâmetros

`provider` IFormatProvider

Um objeto que fornece informações de formatação específicas da cultura.

`format` String

Uma cadeia de caracteres de formato composto.

`arg0` Object

O objeto a ser formatado.

Retornos

String

Uma cópia do `format` na qual o item ou itens de formato foram substituídos pela representação de cadeia de caracteres do `arg0`.

Exceções

ArgumentNullException

`format` é `null`.

FormatException

`format` é inválido.

- ou -

O índice de um item de formato não é zero.

Comentários

> **Importante**
>
> Em vez de chamar o método **String.Format** ou usar **cadeias de caracteres de formato de composição**, é possível usar *cadeias de caracteres interpoladas* quando a linguagem é compatível com eles. Uma cadeia de caracteres interpolada é uma cadeia de caracteres que contém *expressões interpoladas*. Cada expressão interpolada é resolvida com o valor da expressão e incluída na cadeia de caracteres resultante quando a cadeia de caracteres é atribuída. Para saber mais, consulte o tópico **Interpolação de cadeia de caracteres (Referência do C#)** ou **Cadeias de caracteres interpoladas (Referência do Visual Basic)**.

Esse método usa o recurso de formatação composta para converter o valor de uma expressão em sua representação de cadeia de caracteres e inserir essa representação em uma cadeia de caracteres. Ao executar a conversão, o método usa formatação sensível à cultura ou um formatador personalizado. O método converte `arg0` para sua representação de cadeia de caracteres chamando seu método **ToString (IFormatProvider)** ou, se o item de formato correspondente do objeto incluir uma cadeia de caracteres de formato, chamando seu método **ToString (String, IFormatProvider)** . Se esses métodos não existirem, ele chamará o método **ToString** sem parâmetros do objeto.

Entretanto, ao chamar o método **String.Format**, não é necessário se concentrar na sobrecarga específica que você deseja chamar. Em vez disso, é possível chamar o método com um objeto que fornece formatação personalizada ou sensível à cultura e uma cadeia de caracteres de formato composto que inclui um ou mais itens de formato. Você atribui a cada item de formato um índice numérico. O primeiro índice começa em 0. Além da cadeia de caracteres inicial, sua chamada de método deve ter tantos argumentos adicionais quantos valores de índice. Por exemplo, uma cadeia

de caracteres cujos itens de formato têm índices 0 e 1 deve ter 2 argumentos; uma com índices de 0 a 5 deve ter 6 argumentos. O compilador de linguagem, então, resolverá a chamada de método para uma sobrecarga específica do método **String.Format**.

Para obter uma documentação mais detalhada sobre como usar o método **String.Format**, consulte o Guia de Introdução ao método String.Format e Qual método chamar?.

Aplica-se a

▶ .NET 6.0 preview 3 e outras versões

Format(IFormatProvider, String, Object[])

Substitui os itens de formato em uma cadeia de caracteres pelas representações cadeia de caracteres de objetos correspondentes em uma matriz especificada. Um parâmetro fornece informações de formatação específicas da cultura.

C# Copiar

```
public static string Format (IFormatProvider? provider, string format,
params object?[] args);
```

Parâmetros

provider IFormatProvider

Um objeto que fornece informações de formatação específicas da cultura.

format String

Uma cadeia de caracteres de formato composto.

args Object[]

Uma matriz de objetos que contém zero ou mais objetos a serem formatados.

Retornos

String

Uma cópia do `format` na qual os itens de formato foram substituídos pela representação de cadeia de caracteres dos objetos correspondentes no `args`.

Exceções

ArgumentNullException

`format` ou `args` é `null`.

FormatException

`format` é inválido.

- ou -

O índice de um item de formato é menor que zero, ou maior ou igual ao tamanho da matriz `args`.

Comentários

> ⓘ **Importante**
>
> Em vez de chamar o método **String.Format** ou usar **cadeias de caracteres de formato de composição**, é possível usar *cadeias de caracteres interpoladas* quando a linguagem é compatível com eles. Uma cadeia de caracteres interpolada é uma cadeia de caracteres que contém *expressões interpoladas*. Cada expressão interpolada é resolvida com o valor da expressão e incluída na cadeia de caracteres resultante quando a cadeia de caracteres é atribuída. Para saber mais, consulte o tópico **Interpolação de cadeia de caracteres (Referência do C#)** ou **Cadeias de caracteres interpoladas (Referência do Visual Basic)**.

Esse método usa o recurso de formatação composta para converter quatro ou mais expressões em suas representações de cadeia de caracteres e inserir essas representações em uma cadeia de caracteres. Ao executar a conversão, o método usa formatação sensível à cultura ou um formatador personalizado. O método converte cada Object argumento em sua representação de cadeia de caracteres chamando seu método **ToString (IFormatProvider)** ou, se o item de formato correspondente do objeto incluir uma cadeia de caracteres de formato, chamando seu método **ToString (String, IFormatProvider)** . Se esses métodos não existirem, ele chamará o método **ToString** sem parâmetros do objeto.

Entretanto, ao chamar o método **String.Format**, não é necessário se concentrar na sobrecarga específica que você deseja chamar. Em vez disso, é possível chamar o

método com um objeto que fornece formatação personalizada ou sensível à cultura e uma cadeia de caracteres de formato composto que inclui um ou mais itens de formato. Você atribui a cada item de formato um índice numérico. O primeiro índice começa em 0. Além da cadeia de caracteres inicial, sua chamada de método deve ter tantos argumentos adicionais quantos valores de índice. Por exemplo, uma cadeia de caracteres cujos itens de formato têm índices 0 e 1 deve ter 2 argumentos; uma com índices de 0 a 5 deve ter 6 argumentos. O compilador de linguagem, então, resolverá a chamada de método para uma sobrecarga específica do método **String.Format**.

Para obter uma documentação mais detalhada sobre como usar o método **String.Format**, consulte o Guia de Introdução ao método String.Format e Qual método chamar?.

Exemplo: formatação sensível à cultura

Este exemplo usa o Format(IFormatProvider, String, Object[]) método para exibir a representação de cadeia de caracteres de alguns valores de data e hora e valores numéricos usando várias culturas diferentes.

C# Copiar Executar

```
string[] cultureNames = { "en-US", "fr-FR", "de-DE", "es-ES" };

DateTime dateToDisplay = new DateTime(2009, 9, 1, 18, 32, 0);
double value = 9164.32;

Console.WriteLine("Culture     Date
Value\n");
foreach (string cultureName in cultureNames)
{
   System.Globalization.CultureInfo culture = new
System.Globalization.CultureInfo(cultureName);
   string output = String.Format(culture, "{0,-11} {1,-35:D} {2:N}",
                                 culture.Name, dateToDisplay, value);
   Console.WriteLine(output);
}
// The example displays the following output:
//    Culture     Date                                 Value
//
//    en-US       Tuesday, September 01, 2009          9,164.32
//    fr-FR       mardi 1 septembre 2009               9 164,32
//    de-DE       Dienstag, 1. September 2009          9.164,32
//    es-ES       martes, 01 de septiembre de 2009     9.164,32
```

Confira também

- DateTimeFormatInfo
- ICustomFormatter
- IFormatProvider
- NumberFormatInfo
- Tipos de formatação no .NET
- Formatação composta
- Cadeias de caracteres de formato de data e hora padrão
- Cadeias de caracteres de formato de data e hora personalizado
- Cadeias de caracteres de formato numérico padrão
- Cadeias de caracteres de formato numérico personalizado
- Cadeias de caracteres de formato TimeSpan padrão
- Cadeias de caracteres de formato TimeSpan personalizado
- Cadeias de caracteres de formato de enumeração

Aplica-se a

▶ .NET 6.0 preview 3 e outras versões

Format(String, Object, Object)

Substitui os itens de formato em uma cadeia de caracteres pela representação de cadeia de caracteres de dois objetos especificados.

C# Copiar

```
public static string Format (string format, object? arg0, object? arg1);
```

Parâmetros

`format` String

Uma cadeia de caracteres de formato composto.

`arg0` Object

O primeiro objeto a ser formatado.

`arg1` Object

O segundo objeto a ser formatado.

Retornos

String

Uma cópia do `format` na qual os itens de formato são substituídos pelas representações da cadeia de caracteres de `arg0` e `arg1`.

Exceções

ArgumentNullException

`format` é `null`.

FormatException

`format` é inválido.

- ou -

O índice de um item de formato não é zero nem um.

Comentários

> **ⓘ Importante**
>
> Em vez de chamar o método **String.Format** ou usar **cadeias de caracteres de formato de composição**, é possível usar *cadeias de caracteres interpoladas* quando a linguagem é compatível com eles. Uma cadeia de caracteres interpolada é uma cadeia de caracteres que contém *expressões interpoladas*. Cada expressão interpolada é resolvida com o valor da expressão e incluída na cadeia de caracteres resultante quando a cadeia de caracteres é atribuída. Para saber mais, consulte o tópico **Interpolação de cadeia de caracteres (Referência do C#)** ou **Cadeias de caracteres interpoladas (Referência do Visual Basic)**.

Esse método usa o recurso de formatação composta para converter o valor de duas expressões em suas representações de cadeia de caracteres e inserir essas representações em uma cadeia de caracteres.

Entretanto, ao chamar o método **String.Format**, não é necessário se concentrar na sobrecarga específica que você deseja chamar. Em vez disso, é possível chamar o método com uma cadeia de caracteres de formato composto que inclui um ou mais itens de formato. Você atribui a cada item de formato um índice numérico. O primeiro índice começa em 0. Além da cadeia de caracteres inicial, sua chamada de método deve ter tantos argumentos adicionais quantos valores de índice. Por

exemplo, uma cadeia de caracteres cujos itens de formato têm índices 0 e 1 deve ter 2 argumentos; uma com índices de 0 a 5 deve ter 6 argumentos. O compilador de linguagem, então, resolverá a chamada de método para uma sobrecarga específica do método **String.Format**.

Para obter uma documentação mais detalhada sobre como usar o método **String.Format**, consulte o Guia de Introdução ao método String.Format e Qual método chamar?.

Exemplo: Formatar dois argumentos

Este exemplo usa o Format(String, Object, Object) método para exibir dados de tempo e de temperatura armazenados em um Dictionary<TKey,TValue> objeto genérico. Observe que a cadeia de caracteres de formato tem três itens de formato, embora haja apenas dois objetos a serem formatados. Isso ocorre porque o primeiro objeto na lista (um valor de data e hora) é usado por dois itens de formato: o primeiro item de formato exibe a hora e o segundo exibe a data.

C# Copiar

```
Dictionary<DateTime, Double> temperatureInfo = new Dictionary<DateTime,
Double>();
temperatureInfo.Add(new DateTime(2010, 6, 1, 14, 0, 0), 87.46);
temperatureInfo.Add(new DateTime(2010, 12, 1, 10, 0, 0), 36.81);

Console.WriteLine("Temperature Information:\n");
string output;
foreach (var item in temperatureInfo)
{
   output = String.Format("Temperature at {0,8:t} on {0,9:d}:
{1,5:N1}°F",
                          item.Key, item.Value);
   Console.WriteLine(output);
}
// The example displays output like the following:
//       Temperature Information:
//
//       Temperature at  2:00 PM on  6/1/2010:  87.5°F
//       Temperature at 10:00 AM on 12/1/2010:  36.8°F
```

Aplica-se a

▶ .NET 6.0 preview 3 e outras versões

Format(IFormatProvider, String, Object,

Object)

Substitui os itens de formato em uma cadeia de caracteres pela representação de cadeia de caracteres de dois objetos especificados. Um parâmetro fornece informações de formatação específicas da cultura.

C# Copiar

```
public static string Format (IFormatProvider? provider, string format,
object? arg0, object? arg1);
```

Parâmetros

`provider` IFormatProvider

Um objeto que fornece informações de formatação específicas da cultura.

`format` String

Uma cadeia de caracteres de formato composto.

`arg0` Object

O primeiro objeto a ser formatado.

`arg1` Object

O segundo objeto a ser formatado.

Retornos

String

Uma cópia do `format` na qual os itens de formato são substituídos pelas representações da cadeia de caracteres de `arg0` e `arg1`.

Exceções

ArgumentNullException

`format` é `null`.

FormatException

`format` é inválido.

- ou -

O índice de um item de formato não é zero nem um.

Comentários

> ⓘ **Importante**
>
> Em vez de chamar o método **String.Format** ou usar **cadeias de caracteres de formato de composição**, é possível usar *cadeias de caracteres interpoladas* quando a linguagem é compatível com eles. Uma cadeia de caracteres interpolada é uma cadeia de caracteres que contém *expressões interpoladas*. Cada expressão interpolada é resolvida com o valor da expressão e incluída na cadeia de caracteres resultante quando a cadeia de caracteres é atribuída. Para saber mais, consulte o tópico **Interpolação de cadeia de caracteres (Referência do C#)** ou **Cadeias de caracteres interpoladas (Referência do Visual Basic)**.

Esse método usa o recurso de formatação composta para converter duas expressões em suas representações de cadeia de caracteres e inserir essas representações em uma cadeia de caracteres. Ao executar a conversão, o método usa formatação sensível à cultura ou um formatador personalizado. O método converte cada Object argumento em sua representação de cadeia de caracteres chamando seu método **ToString (IFormatProvider)** ou, se o item de formato correspondente do objeto incluir uma cadeia de caracteres de formato, chamando seu método **ToString (String, IFormatProvider)** . Se esses métodos não existirem, ele chamará o método **ToString** sem parâmetros do objeto.

Entretanto, ao chamar o método **String.Format**, não é necessário se concentrar na sobrecarga específica que você deseja chamar. Em vez disso, é possível chamar o método com um objeto que fornece formatação personalizada ou sensível à cultura e uma cadeia de caracteres de formato composto que inclui um ou mais itens de formato. Você atribui a cada item de formato um índice numérico. O primeiro índice começa em 0. Além da cadeia de caracteres inicial, sua chamada de método deve ter tantos argumentos adicionais quantos valores de índice. Por exemplo, uma cadeia de caracteres cujos itens de formato têm índices 0 e 1 deve ter 2 argumentos; uma com índices de 0 a 5 deve ter 6 argumentos. O compilador de linguagem, então, resolverá a chamada de método para uma sobrecarga específica do método **String.Format**.

Para obter uma documentação mais detalhada sobre como usar o método **String.Format**, consulte o Guia de Introdução ao método String.Format e Qual método chamar?.

Aplica-se a

▶ .NET 6.0 preview 3 e outras versões

Format(String, Object, Object, Object)

Substitui os itens de formato em uma cadeia de caracteres pela representação de cadeia de caracteres de três objetos especificados.

C# Copiar

```
public static string Format (string format, object? arg0, object? arg1,
object? arg2);
```

Parâmetros

`format` String

Uma cadeia de caracteres de formato composto.

`arg0` Object

O primeiro objeto a ser formatado.

`arg1` Object

O segundo objeto a ser formatado.

`arg2` Object

O terceiro objeto a ser formatado.

Retornos

String

Uma cópia do `format` na qual ou itens de formato foram substituídos pela representação de cadeia de caracteres de `arg0`, `arg1` e `arg2`.

Exceções

ArgumentNullException

`format` é `null`.

FormatException

`format` é inválido.

- ou -

O índice de um item de formato é menor que zero ou maior que dois.

Comentários

> **ⓘ Importante**
>
> Em vez de chamar o método **String.Format** ou usar **cadeias de caracteres de formato de composição**, é possível usar *cadeias de caracteres interpoladas* quando a linguagem é compatível com eles. Uma cadeia de caracteres interpolada é uma cadeia de caracteres que contém *expressões interpoladas*. Cada expressão interpolada é resolvida com o valor da expressão e incluída na cadeia de caracteres resultante quando a cadeia de caracteres é atribuída. Para saber mais, consulte o tópico **Interpolação de cadeia de caracteres (Referência do C#)** ou **Cadeias de caracteres interpoladas (Referência do Visual Basic)**.

Esse método usa o recurso de formatação composta para converter o valor de três expressões em suas representações de cadeia de caracteres e inserir essas representações em uma cadeia de caracteres.

Entretanto, ao chamar o método **String.Format**, não é necessário se concentrar na sobrecarga específica que você deseja chamar. Em vez disso, é possível chamar o método com uma cadeia de caracteres de formato composto que inclui um ou mais itens de formato. Você atribui a cada item de formato um índice numérico. O primeiro índice começa em 0. Além da cadeia de caracteres inicial, sua chamada de método deve ter tantos argumentos adicionais quantos valores de índice. Por exemplo, uma cadeia de caracteres cujos itens de formato têm índices 0 e 1 deve ter 2 argumentos; uma com índices de 0 a 5 deve ter 6 argumentos. O compilador de linguagem, então, resolverá a chamada de método para uma sobrecarga específica do método **String.Format**.

Para obter uma documentação mais detalhada sobre como usar o método **String.Format**, consulte o Guia de Introdução ao método String.Format e Qual método chamar?.

Exemplo: Formatar três argumentos

Este exemplo usa o Format(String, Object, Object, Object) método para criar uma cadeia de caracteres que ilustra o resultado de uma operação booliana `And` com dois valores inteiros. Observe que a cadeia de caracteres de formato inclui seis itens de formato, mas o método tem apenas três itens em sua lista de parâmetros, pois cada item é formatado de duas maneiras diferentes.

C# Copiar Executar

```
string formatString = "    {0,10} ({0,8:X8})\n" +
                      "And {1,10} ({1,8:X8})\n" +
                      "  = {2,10} ({2,8:X8})";
int value1 = 16932;
int value2 = 15421;
string result = String.Format(formatString,
                              value1, value2, value1 & value2);
Console.WriteLine(result);
// The example displays the following output:
//                16932 (00004224)
//       And      15421 (00003C3D)
//         =         36 (00000024)
```

Aplica-se a

▶ .NET 6.0 preview 3 e outras versões

Format(IFormatProvider, String, Object, Object, Object)

Substitui os itens de formato em uma cadeia de caracteres pela representação de cadeia de caracteres de três objetos especificados. Um parâmetro fornece informações de formatação específicas da cultura.

C# Copiar

```
public static string Format (IFormatProvider? provider, string format,
object? arg0, object? arg1, object? arg2);
```

Parâmetros

`provider` IFormatProvider

Um objeto que fornece informações de formatação específicas da cultura.

format String

Uma cadeia de caracteres de formato composto.

arg0 Object

O primeiro objeto a ser formatado.

arg1 Object

O segundo objeto a ser formatado.

arg2 Object

O terceiro objeto a ser formatado.

Retornos

String

Uma cópia do `format` na qual ou itens de formato foram substituídos pela representação de cadeia de caracteres de `arg0`, `arg1` e `arg2`.

Exceções

ArgumentNullException

`format` é `null`.

FormatException

`format` é inválido.

- ou -

O índice de um item de formato é menor que zero ou maior que dois.

Comentários

> ⓘ **Importante**
>
> Em vez de chamar o método **String.Format** ou usar **cadeias de caracteres de formato de composição**, é possível usar *cadeias de caracteres interpoladas* quando a linguagem é compatível com eles. Uma cadeia de caracteres interpolada é uma cadeia de caracteres que contém *expressões interpoladas*. Cada expressão interpolada é resolvida com o valor da expressão e incluída na cadeia de caracteres resultante quando a cadeia de caracteres é atribuída. Para

saber mais, consulte o tópico **Interpolação de cadeia de caracteres (Referência do C#)** ou **Cadeias de caracteres interpoladas (Referência do Visual Basic)**.

Esse método usa o recurso de formatação composta para converter três expressões em suas representações de cadeia de caracteres e inserir essas representações em uma cadeia de caracteres. Ao executar a conversão, o método usa formatação sensível à cultura ou um formatador personalizado. O método converte cada Object argumento em sua representação de cadeia de caracteres chamando seu método **ToString (IFormatProvider)** ou, se o item de formato correspondente do objeto incluir uma cadeia de caracteres de formato, chamando seu método **ToString (String, IFormatProvider)** . Se esses métodos não existirem, ele chamará o método **ToString** sem parâmetros do objeto.

Entretanto, ao chamar o método **String.Format**, não é necessário se concentrar na sobrecarga específica que você deseja chamar. Em vez disso, é possível chamar o método com um objeto que fornece formatação personalizada ou sensível à cultura e uma cadeia de caracteres de formato composto que inclui um ou mais itens de formato. Você atribui a cada item de formato um índice numérico. O primeiro índice começa em 0. Além da cadeia de caracteres inicial, sua chamada de método deve ter tantos argumentos adicionais quantos valores de índice. Por exemplo, uma cadeia de caracteres cujos itens de formato têm índices 0 e 1 deve ter 2 argumentos; uma com índices de 0 a 5 deve ter 6 argumentos. O compilador de linguagem, então, resolverá a chamada de método para uma sobrecarga específica do método **String.Format**.

Para obter uma documentação mais detalhada sobre como usar o método **String.Format**, consulte o Guia de Introdução ao método String.Format e Qual método chamar?.

Aplica-se a

▶ .NET 6.0 preview 3 e outras versões

Esta página é útil?

Yes No