# Math.Round Method

名前空間: WS.Theia.ExtremelyPrecise

アセンブリ: ExtremelyPrecise.dll

最も近い整数または指定した小数点以下の桁数に値を丸めます。

# オーバーロード

|  |  |
| --- | --- |
| Round(Rational,int,　MidpointRounding) | Rational の値は指定した小数部の桁数に丸められ、中間値には指定した丸め処理が使用されます。 |
| Round(Rational,　MidpointRounding) | Rational の値は最も近い整数に丸められ、中間値には指定した丸め処理が使用されます。 |
| Round(Rational,int) | Rational の値は指定した小数部の桁数に丸められ、中間値は最も近い偶数値に丸められます。 |
| Round(Rational) | Rational の値は最も近い整数値に丸められ、中間値は最も近い偶数値に丸められます。 |

# Round(Rational,int,MidpointRounding)

Rational の値は指定した小数部の桁数に丸められ、中間値には指定した丸め処理が使用されます。

public static WS.Theia.ExtremelyPrecise.Rational Round(WS.Theia.ExtremelyPrecise.Rational value,int digits,MidpointRounding mode);

## パラメーター

value　Rational

丸め対象のRational値。

digits　Rational

戻り値の小数部の桁数。

mode　Rational

value が 2つの整数の中間にある場合に丸める方法を指定します。

## 戻り値

Rational  
digits に等しい小数部の桁数を格納する value に最も近い数値。 value の小数部の桁数が digits よりも少ない場合、value がそのまま返されます。

## 例外

ArgumentException

modeがMidpointRoundingの正しい値ではありません。

## 例

次の例では、Round(Rational,int,MidpointRounding)メソッドをMidpointRounding値列挙体で丸めモードを指定しています。

// This example demonstrates the Math.Round() method in conjunction   
// with the MidpointRounding enumeration.  
using System;  
using WS.Theia.ExtremelyPrecise;  
  
class Sample   
{  
 public static void Main()   
 {  
 Rational result = 0.0m;  
 Rational posValue = 3.45m;  
 Rational negValue = -3.45m;  
  
// By default, round a positive and a negative value to the nearest even number.   
// The precision of the result is 1 decimal place.  
  
 result = Math.Round(posValue, 1);  
 Console.WriteLine("{0,4} = Math.Round({1,5}, 1)", result, posValue);  
 result = Math.Round(negValue, 1);  
 Console.WriteLine("{0,4} = Math.Round({1,5}, 1)", result, negValue);  
 Console.WriteLine();  
  
// Round a positive value to the nearest even number, then to the nearest number away from zero.   
// The precision of the result is 1 decimal place.  
  
 result = Math.Round(posValue, 1, MidpointRounding.ToEven);  
 Console.WriteLine("{0,4} = Math.Round({1,5}, 1, MidpointRounding.ToEven)", result, posValue);  
 result = Math.Round(posValue, 1, MidpointRounding.AwayFromZero);  
 Console.WriteLine("{0,4} = Math.Round({1,5}, 1, MidpointRounding.AwayFromZero)", result, posValue);  
 Console.WriteLine();  
  
// Round a negative value to the nearest even number, then to the nearest number away from zero.   
// The precision of the result is 1 decimal place.  
  
 result = Math.Round(negValue, 1, MidpointRounding.ToEven);  
 Console.WriteLine("{0,4} = Math.Round({1,5}, 1, MidpointRounding.ToEven)", result, negValue);  
 result = Math.Round(negValue, 1, MidpointRounding.AwayFromZero);  
 Console.WriteLine("{0,4} = Math.Round({1,5}, 1, MidpointRounding.AwayFromZero)", result, negValue);  
 Console.WriteLine();  
 }  
}  
/\*  
This code example produces the following results:  
  
 3.4 = Math.Round( 3.45, 1)  
-3.4 = Math.Round(-3.45, 1)  
  
 3.4 = Math.Round( 3.45, 1, MidpointRounding.ToEven)  
 3.5 = Math.Round( 3.45, 1, MidpointRounding.AwayFromZero)  
  
-3.4 = Math.Round(-3.45, 1, MidpointRounding.ToEven)  
-3.5 = Math.Round(-3.45, 1, MidpointRounding.AwayFromZero)  
  
\*/

## 注釈

MidpointRounding値列挙体で丸めモードを使用した場合次の表のとおりとなります。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 小数部＜0.5 | 小数部＝0.5 | 小数部＞0.5 |
| ToEven | 切り捨て | 整数部が偶数の場合、切り捨て 整数部が奇数の場合、切り上げ | 切り上げ |
| AwayFromZero | 切り捨て | 切り上げ | 切り上げ |

# Round(Rational,MidpointRounding)

Rational の値は最も近い整数に丸められ、中間値には指定した丸め処理が使用されます。

public static WS.Theia.ExtremelyPrecise.Rational Round(WS.Theia.ExtremelyPrecise.Rational value,MidpointRounding mode);

## パラメーター

value　Rational

丸め対象のRational値。

mode　 MidpointRounding

value が 2つの整数の中間にある場合に丸める方法を指定します。

## 戻り値

Rational  
value に最も近い整数。 value が 2つの整数 (一方が偶数でもう一方が奇数) の中間にある場合、mode によって 2つの数値のどちらが返されるかが決まります。 このメソッドは、整数型ではなく Rational を返します。

## 例外

ArgumentException

modeがMidpointRoundingの正しい値ではありません。

## 例

次の例では、Round(Rational,MidpointRounding)メソッドをMidpointRounding値列挙体で丸めモードを指定しています。

using System;  
using WS.Theia.ExtremelyPrecise;  
public class Example  
{  
 public static void Main()  
 {  
 Rational[] values = { 12.0, 12.1, 12.2, 12.3, 12.4, 12.5, 12.6,   
 12.7, 12.8, 12.9, 13.0 };  
 Console.WriteLine("{0,-10} {1,-10} {2,-10} {3,-15}", "Value", "Default",   
 "ToEven", "AwayFromZero");  
 foreach (var value in values)  
 Console.WriteLine("{0,-10:R} {1,-10} {2,-10} {3,-15}",  
 value, Math.Round(value),   
 Math.Round(value, MidpointRounding.ToEven),  
 Math.Round(value, MidpointRounding.AwayFromZero));  
 }  
}  
// The example displays the following output:  
// Value Default ToEven AwayFromZero  
// 12 12 12 12  
// 12.1 12 12 12  
// 12.2 12 12 12  
// 12.3 12 12 12  
// 12.4 12 12 12  
// 12.5 12 12 13  
// 12.6 13 13 13  
// 12.7 13 13 13  
// 12.8 13 13 13  
// 12.9 13 13 13  
// 13.0 13 13 13

## 注釈

MidpointRounding値列挙体で丸めモードを使用した場合次の表のとおりとなります。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 小数部＜0.5 | 小数部＝0.5 | 小数部＞0.5 |
| ToEven | 切り捨て | 整数部が偶数の場合、切り捨て 整数部が奇数の場合、切り上げ | 切り上げ |
| AwayFromZero | 切り捨て | 切り上げ | 切り上げ |

# Round(Rational,int)

Rational の値は指定した小数部の桁数に丸められ、中間値は最も近い偶数値に丸められます。

public static WS.Theia.ExtremelyPrecise.Rational Round(WS.Theia.ExtremelyPrecise.Rational value,int digits);

## パラメーター

value　Rational

丸め対象のRational値。

digits　Rational

戻り値の小数部の桁数。

## 戻り値

Rational  
digits に等しい小数部の桁数を格納する value に最も近い数値。 value の小数部の桁数が digits よりも少ない場合、value がそのまま返されます。

## 例

次の例では、Round(Rational,int)メソッドで丸めを行っています。

Math.Round(3.44, 1); //Returns 3.4.  
Math.Round(3.45, 1); //Returns 3.4.  
Math.Round(3.46, 1); //Returns 3.5.  
  
Math.Round(4.34, 1); // Returns 4.3  
Math.Round(4.35, 1); // Returns 4.4  
Math.Round(4.36, 1); // Returns 4.4

# Round(Rational)

Rational の値は最も近い整数値に丸められ、中間値は最も近い偶数値に丸められます。

public static WS.Theia.ExtremelyPrecise.Rational Round(WS.Theia.ExtremelyPrecise.Rational value);

## パラメーター

value　Rational

丸め対象のRational値。

## 戻り値

Rational  
value パラメーターに最も近い整数。 value の小数部が 2つの整数 (一方が偶数で、もう一方が奇数) の中間にある場合は、偶数が返されます。 このメソッドは、整数型ではなく Rational を返します。

## 例

次の例では、Round(Rational)メソッドで丸めを行っています。

using System;  
using WS.Theia.ExtremelyPrecise;  
  
class Program  
{  
 static void Main()  
 {  
 Console.WriteLine("Classic Math.Round in CSharp");  
 Console.WriteLine(Math.Round(4.4m)); // 4  
 Console.WriteLine(Math.Round(4.5 m)); // 4  
 Console.WriteLine(Math.Round(4.6 m)); // 5  
 Console.WriteLine(Math.Round(5.5 m)); // 6  
 }  
}

# 適用対象

### .NET Core

2.0

### .NET Framework

4.6.1

### .NET Standard

2.0

### UWP

10.0.16299

### Xamarin.Android

8.0

### Xamarin.iOS

10.14

### Xamarin.Mac

3.8