

Re: Fastcode RoundTo

Source: <http://coding.derkeiler.com/Archive/Delphi/borland.public.delphi.language.basm/2004-06/0247.html>

From: John Herbster (*herb-sci1_AT_sbcglobal.net*)

Date: 06/15/04

Date: Tue, 15 Jun 2004 11:32:40 -0500

"Dennis" <mariandkc@home3.gvnet.dk> wrote
> *If it is then I will rename the challenge to RoundToEx. This*
> *means that we will have free hands to modify the functionality.*

For some ideas, I suggest considering something like the following calls:

```
function RoundDbIto (  
    const Value: extended;  
    const Digits: integer;  
    const Ctrl: tDecimalRoundingCtrl = drHalfEven;  
    const SafetyFactor: double = 2)  
    : extended;
```

With variations for RoundExtTo and RoundSglTo.
The different variations are required in order to automatically pull in the appropriate relative error thresholds for rounding the different number types.

and/or

```
procedure DecRound (  
    const Digits: integer;  
    var Value: double; {Note overloaded calls for other types}  
    const Ctrl: tDecimalRoundingCtrl = drHalfEven;  
    const SafetyFactor: double = 2)  
    : extended; overloaded;
```

With overloaded versions with Value's type equal type extended and type single.

The advantage of using a procedure, is that it allows the overloading mechanism to cause the linker to link to the proper call. I think that the use of a procedure while a little more trouble for the programmer would save him trouble in the long run.

The rounding control could be like the following:

```
tDecimalRoundingCtrl = {Defined rounding methods}
  (drNone, {No rounding.}
  drHalfEven, {Round to nearest else to even digit. a.k.a Bankers}
  drHalfOdd, {Round to nearest else to odd digit. }
  drHalfPos, {Round to nearest else toward positive. }
  drHalfNeg, {Round to nearest else toward negative. }
  drHalfDown, {Round to nearest else toward zero. }
  drHalfUp, {Round to nearest else away from zero. }
  drRndNeg, {Round toward negative. a.k.a. Floor}
  drRndPos, {Round toward positive. a.k.a. Ceil }
  drRndDown, {Round toward zero. a.k.a. Trunc}
  drRndUp); {Round away from zero.}
```

In my experience I have only seen one rounding requirement that would not match one of the above types.

Maybe "Up" and "Down" should be replaced with "Out" and "In".

I suggest the further consideration be given to choosing the sign for the Digits parameter. I think that the sign for the RoundTo Digits parameter is the negative of what seems natural. Perhaps it should be named NumDecFractDigits.

By the way, there may be a Java "BigDecimal" package (JSR13 in J2SE 1.5?) that should be considered for function and nomenclature compatibility. For info see

<http://www.ddj.com/articles/2004/0407/>

<http://www.google.com/search?q=BigDecimal%20JSR13%20OR%20Java>

<http://groups.google.com/groups?q=BigDecimal%20JSR13%20OR%20Java>

--

Regards, John Herbster
(Support the movement to add floating and fixed
decimal fraction numbers to the language.)