



Het probleem met de Val-procedure

In Object Pascal is volgens de Delphi-helfunctie de `Val`-procedure bedoeld om een string te converteren naar een getal en daarbij mogelijke fouten te ondervangen. Het getal kan daarbij zowel een gebroken getal als een geheel getal zijn. De procedure wordt op de volgende manier gebruikt:

```
Val ( stringVar, getalVar, code );
```

Naar wat voor type getal de `stringVar` geconverteerd wordt is afhankelijk van het type van de variabele `getalVar`. Deze procedure kan erg handig zijn bij het 'error-proof' of 'fool-proof' maken van je programma omdat het niet alleen de conversie uitvoert maar ook ondervangt dat er een string ingevoerd wordt die niet geconverteerd kan worden naar een getal. In dat geval zal de variabele `code` niet nul blijven, maar de plaats van het eerste karakter aangeven dat niet geconverteerd kan worden. Als `stringVar` '1234hoedje van' zou zijn, zouden `getalVar` en `code` respectievelijk 1234 en 5 moeten worden. (Merk op dat `getalVar` en `code` 'call by reference'-argumenten zijn.)

Helaas werkt de procedure `Val` niet helemaal foutloos met gebroken getallen buiten Amerika. Het werkt overigens wel foutloos bij gehele getallen. Wat is het probleem?

- Als de `Val`-procedure moet werken met getallen met een komma of een punt erin gaat er iets fout. Neem de onderstaande code:

```
procedure TfrmValForm.btnConvertClick(Sender: TObject);
var
    realValue : real;
    code      : integer;
begin
    Val(edtInput.Text, realValue, code);
    ShowMessage (FloatToStr(realValue)+'
'+IntToStr(code));
end;
```

Deze bijlage is geschreven door Michael Capalbo, ... aan de Universiteit Maastricht.

502 Appendix G Het probleem met de Val-procedure

Als bij `edtInput.Text` bijvoorbeeld '1234,56' invoert zie je na de conversie voor `realValue` de waarde 1234 en code 6. Terwijl als `edtInput.Text` '1234.56' is `realValue` 1234,56 wordt en code 0. Wat is de oorzaak hiervan?

- Omdat Delphi uit de Verenigde Staten komt, gebruikt Delphi intern voor variabelen voor gebroken getallen (variabelen van het type `real`) een punt als decimaal scheider.
- In de regional settings van Windows staat hoe de decimaal scheider eruitziet in het land van gebruik. In Nederland en België is dat een komma.
- Daarom verwacht Delphi in Nederland voor conversie naar `real` getallen een komma maar slaat het de variabele op met een punt. Om dezelfde reden gebruikt Delphi voor conversie van `real` naar `string` een komma in de `string`.
- De `Val`-procedure kijkt echter niet naar de regional settings, maar verwacht gewoon een punt en ziet een komma als een illegaal karakter. Als er een komma staat stopt de `Val`-procedure het getal voor de komma in de variabele. `Floattostr` werkt echter wel zoals het hoort en schrijft naar de `Showmessage` met een komma als decimaalscheider.

Dit gedrag van de `Val`-procedure is de makers van Delphi bekend, maar zij zien dit niet als een fout. (Getuige Quality Central (<http://qc.borland.com/>) case 2639.) Daarom zullen we, als we `Val` toch willen gebruiken voor gebroken getallen, een oplossing moeten vinden. Mogelijke oplossingen zijn:

- Maak je eigen `Val`-procedure. We zullen hieronder een werkend alternatief zetten. De procedure `FVal` hieronder doet wat `Val` eigenlijk zou moeten doen.

```
{The overload procedure for conversion to reals.}
procedure Val (InputString: TCaption ; var OutputReal:
real; var ErrorLocation: integer); overload;
var
  I          : Integer;
  InputLength : Integer;
  WorkString : string;
  RealChar   : Set of Char;
begin
  RealChar := ['0'..'9'];
  ErrorLocation := 0;
  InputString := trim(InputString);
  InputLength:= length(InputString);
  for I := 1 to InputLength do
    begin
      if (InputString[I] = DecimalSeparator) or (InputString[I] = ThousandSeparator) or (InputString[I] in RealChar) or ((Inputstring[I] = '-') and (I = 1)) then
        WorkString := Workstring + Inputstring[I]
      else
        begin
```

```
        ErrorLocation := I;
        Break;
    end;
end;
If WorkString = '' then
begin
    OutputReal := 0;
    ErrorLocation := 1;

end
else
    OutputReal := StrtoFloat(WorkString);
end;

{The overload procedure for conversion to integers.}

procedure Val (InputString: TCaption ; var OutputReal:
integer; var ErrorLocation: integer); overload;
var
    I, InputLength : Integer;
    WorkString: string;
    IntChar : Set of Char;
begin
    IntChar := ['0'..'9'];
    ErrorLocation := 0;
    InputString := trim(InputString);
    InputLength := length(InputString);
    for I := 1 to InputLength do
        begin
            if (InputString[I] in IntChar) or ((Inputstring[I]
= '-' ) and (I = 1)) then
                WorkString := Workstring + Inputstring[I]
            else
                begin
                    ErrorLocation := I;
                    Break;
                end;
            end;
        end;
    If WorkString = '' then
    begin
        OutputReal := 0;
        ErrorLocation := 1;
    end
    else
        OutputReal := StrtoInt(WorkString);
    end;
```

504 Appendix G Het probleem met deVal-procedure

Je zou, om deze procedure niet altijd opnieuw te hoeven typen, deze procedure op een aparte unit bij je project kunnen toevoegen (met `Project | Add to Project...`) en dan de naam van die nieuwe unit bij de uses list van de unit waar je `Val` in wil gebruiken toevoegen. Vergeet niet de nieuwe procedures 'public' te declareren in de extra unit. Het begin van je programma ziet er dan zo uit (als `FixVal` de naam is van de aparte unit):

```
unit Unit1;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Classes, Graphics,  
Controls, Forms, Dialogs, StdCtrls, FixVal;
```

- De snelste oplossing is het downloaden en uitvoeren van een 'fix', die er voor zorgt dat het probleem wordt opgelost binnen Delphi. Download het bestand `FixVal.dcu.exe` van (www.pearsoneducation.nl/kerman). Je kunt de 'fix' meteen uitvoeren door 'Run this program from its current location' te kiezen en je keuze te bevestigen. (Eerst downloaden en dan uitvoeren kan natuurlijk ook.) Je moet nu echter wel, zoals in het voorbeeld hierboven `FixVal` aan het eind van je `Uses` lijst toevoegen, daarna kun je `Val` gewoon gebruiken.
- Deze laatste 'oplossing' is niet echt een oplossing maar het stelt je wel in staat om de functie te gebruiken en consistent te zijn in je programma (in jargon een 'workaround' genoemd). Zet in de `FormCreate` event de volgende code:

```
DecimalSeparator := '.';  
ThousandSeparator := ',';
```

Hiermee geef je aan dat in jou hele programma de decimalen gescheiden worden door een punt en de duizenden van de honderden met een komma. Dit is dan consistent met de `Val`-functie en blijft geldig in je hele programma. In de invoer moet nu echter ook met punten voor de decimalen gewerkt worden (Je kunt dit ook zien in de functies die naar string converteren). Je kunt via de `Windows` settings of via de `Delphi DBE Administrator` deze instellingen voor heel `Windows` veranderen.