Subject: Re: Converting Doubles to Strings
Posted by Andrew Cool on Sat, 05 Nov 2005 09:46:14 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Folks,
>
> I've run into an interesting problem. I have a double precision
> number that I wish to convert to a string (so I can put it into
> a text widget, for example). I don't know ahead of time how many
> significant digits will be in this number. The number could
> look like this 45.6, or this 123456789.123456789, or this
> -22.1234567890. If it looks like the latter number, I am
> having a very hard time converting it to a string. For example,
> this doesn't work:
>
>   IDL> v = String(-22.1234567890, Format='(D0)')
>   IDL> Print, v
>   -22.123457
>
> Is this a bug in IDL? Or am I overlooking something?
>

Dropping the '0' from 'D0' seems to work OK. Why though is anyone's guess...

Andrew

Subject: Re: Converting Doubles to Strings
Posted by biophys on Sat, 05 Nov 2005 10:40:49 GMT
View Forum Message <> Reply to Message

I had thought of this before when I was writing a widget. My understanding is that this so-called natural width is only meaningful when you are reading ascii data. In other words, the natural width is always tied to the digital/string representation of a number. However, the problem with string() function is that it converts numerical data in its binary representation into string. For example, when we issue the following command to idl.

IDL>print,String(-22.123, Format='(D0)')
-22.122999

you provide a string representation of a number which is '-22.123', idl will read it and figure out it is a floating number and convert that into its binary representation, which rounds to -22.122999, and pass it to function string(), and then it convert it into the string '

-22.122999'. Similarly if you try the following command,

IDL>print,String(-22.123d0, Format='(D0)')
-22.123000

Now idl knows it is a double precision and converts it into 8 bytes double data and then converts it into string with the natural width of the converted double, which is '-22.123000'.

But if you really provide more than 6 digits after decimal point, you will see what david was showing. Natural width seems to be fixed to 6 digits after decimal point no matter whether it is float or double.

IDL> print,String(-22.1234567890, Format='(D0)')
-22.123457
IDL> print,String(-22.1234567890d0, Format='(D0)')
-22.123457

if you use full width, you will get

IDL> print,String(-22.1234567890, Format='(D)')
-22.1234570
IDL> print,String(-22.1234567890d0, Format='(D)')
-22.1234567889999987

So the bottomline is, numbers in digits are always rounded and stored in binary representation in computer. Which means exact digital representation of float/double numbers only exist in some "string representation". Once the number is taken by the machine and being stored in its closest binary representation you lose all your width information of its original digital representation. And if later you want to have some digital representation, then you have to specify the width. Otherwise, you can either use F0/D0 to get 6 digits after decimal point or use F/D to get the closest full width digital/string representation. I end up using the latter for my widgets.

Please correct me if I'm wrong.

Thx

David Fanning wrote:
> Folks,
>

> I've run into an interesting problem. I have a double precision
> number that I wish to convert to a string (so I can put it into
> a text widget, for example). I don't know ahead of time how many
> significant digits will be in this number. The number could
> look like this 45.6, or this 123456789.123456789, or this
> -22.1234567890. If it looks like the latter number, I am
> having a very hard time converting it to a string. For example,
> this doesn't work:
>
> IDL> v = String(-22.1234567890, Format='(D0)')
> IDL> Print, v
> -22.123457
>
> Is this a bug in IDL? Or am I overlooking something?
>
> Cheers,
>
> David
> --
> David Fanning, Ph.D.,
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Converting Doubles to Strings
Posted by Craig Markwardt on Sat, 05 Nov 2005 15:31:41 GMT
View Forum Message <> Reply to Message

David Fanning <david@dfanning.com> writes:
> Folks,
>
> I've run into an interesting problem. I have a double precision
> number that I wish to convert to a string (so I can put it into
> a text widget, for example). I don't know ahead of time how many
> significant digits will be in this number. The number could
> look like this 45.6, or this 123456789.123456789, or this
> -22.1234567890. If it looks like the latter number, I am
> having a very hard time converting it to a string. For example,
> this doesn't work:
>
> IDL> v = String(-22.1234567890, Format='(D0)')
> IDL> Print, v
> -22.123457
>
> Is this a bug in IDL? Or am I overlooking something?

David, the "D" format (without zeroes) is probably what you want. I
have a utility routine in INPUTFORM which converts a floating point

number to a string. It tries both the G and E formats and takes the
shortest version that is still correct. See below.

Craig

```
;; Convert a floating style value to a string.  Note the conversion
;; happens twice, once as a E and once as a G.  The shortest correct
;; version of the two is used.
;;  X - number to convert, scalar or array, float or double
;;  FORMAT - optional format to use (set to '(E)' or '(D)' for max precision)
;;  DCONVERT - set this if the output should be double precision
function inputform_float, x, format, dconvert=dcon
  n = n_elements(x)
  str = string(x(*), format=format)
  sz = size(x) & tp = sz(sz(0)+1)

  ;; Sorry, there appears to be no other way to make nice looking
  ;; floating point numbers.
  str1 = string(x(*), format='(G0)')
  if tp EQ 4 then x1 = float(str1)
  if tp EQ 5 then x1 = double(str1)
  wh = where(x-x1 EQ 0, ct)
  if ct GT 0 then str(wh) = str1(wh)
  str1 = 0
  str = strtrim(str,2)

  p = strpos(str(0), 'E')  ;; Make sure at least one element is float-type
  ;; Note, the space is needed in case the string is placed inside
  ;; another expression down the line.
  if p LT 0 then begin
     if keyword_set(dcon) then str(0) = str(0) + 'D' $
     else str(0) = str(0) + 'E'
  endif
  if keyword_set(dcon) then begin
     ;; Convert from floating to double
     p = strpos(str, 'E')
     wh = where(p GE 0, ct)
     for i = 0L, ct-1 do begin
        str1 = str(wh(i))
        strput, str1, 'D', p(wh(i))
        str(wh(i)) = str1
     endfor
  endif
  ;; Construct format like (N(A,:,","))
  fmt = '('+strtrim(n,2)+'(A,:,","))'
  return, string(str, format=fmt)
end
```

--
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@REMOVEcow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response

Subject: Re: Converting Doubles to Strings
Posted by biophys on Sun, 06 Nov 2005 01:01:53 GMT
View Forum Message <> Reply to Message

Hi, Craig

I tried your inputform_float and It seems that it won't do max
precision with '(d)',

IDL> print,inputform_float(-22.1234567890, '(d0)', /dconvert)
-22.123457D

IDL> print,inputform_float(-22.1234567890, '(d)', /dconvert)
-22.1234570D

Craig Markwardt wrote:

> David, the "D" format (without zeroes) is probably what you want. I
> have a utility routine in INPUTFORM which converts a floating point
> number to a string. It tries both the G and E formats and takes the
> shortest version that is still correct. See below.
>
> Craig
>
> ;; Convert a floating style value to a string. Note the conversion
> ;; happens twice, once as a E and once as a G. The shortest correct
> ;; version of the two is used.
> ;; X - number to convert, scalar or array, float or double
> ;; FORMAT - optional format to use (set to '(E)' or '(D)' for max precision)
> ;; DCONVERT - set this if the output should be double precision
> function inputform_float, x, format, dconvert=dcon
> n = n_elements(x)
> str = string(x(*), format=format)
> sz = size(x) & tp = sz(sz(0)+1)
>
> ;; Sorry, there appears to be no other way to make nice looking
> ;; floating point numbers.

```
>    str1 = string(x(*), format='(G0)')
>    if tp EQ 4 then x1 = float(str1)
>    if tp EQ 5 then x1 = double(str1)
>    wh = where(x-x1 EQ 0, ct)
>    if ct GT 0 then str(wh) = str1(wh)
>    str1 = 0
>    str = strtrim(str,2)
>
>    p = strpos(str(0), 'E')  ;; Make sure at least one element is float-type
>    ;; Note, the space is needed in case the string is placed inside
>    ;; another expression down the line.
>    if p LT 0 then begin
>       if keyword_set(dcon) then str(0) = str(0) + 'D' $
>       else str(0) = str(0) + 'E'
>    endif
>    if keyword_set(dcon) then begin
>       ;; Convert from floating to double
>       p = strpos(str, 'E')
>       wh = where(p GE 0, ct)
>       for i = 0L, ct-1 do begin
>          str1 = str(wh(i))
>          strput, str1, 'D', p(wh(i))
>          str(wh(i)) = str1
>       endfor
>    endif
>    ;; Construct format like (N(A,:,","))
>    fmt = '('+strtrim(n,2)+'(A,:,","))'
>    return, string(str, format=fmt)
> end
>
>
>
> --
>  ------------------------------------------------------------ --------------
> Craig B. Markwardt, Ph.D.      EMAIL: craigmnet@REMOVEcow.physics.wisc.edu
> Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
>  ------------------------------------------------------------ --------------
```

Subject: Re: Converting Doubles to Strings
Posted by biophys on Sun, 06 Nov 2005 03:20:22 GMT

View Forum Message <> Reply to Message

oops, i'm sorry i forgot the D. it does work fine!

```
IDL>  print,inputform_float(-22.1234567890D, '(d)', /dconvert)
-22.1234567889999987D
```

biophys wrote:

```
>  Hi, Craig
```

>
> I tried your inputform_float and It seems that it won't do max
> precision with '(d)',
>
> IDL> print,inputform_float(-22.1234567890, '(d0)', /dconvert)
> -22.123457D
>
> IDL> print,inputform_float(-22.1234567890, '(d)', /dconvert)
> -22.1234570D
>
>
>
> Craig Markwardt wrote:
>
>> David, the "D" format (without zeroes) is probably what you want. I
>> have a utility routine in INPUTFORM which converts a floating point
>> number to a string. It tries both the G and E formats and takes the
>> shortest version that is still correct. See below.
>>
>> Craig
>>
>> ;; Convert a floating style value to a string. Note the conversion
>> ;; happens twice, once as a E and once as a G. The shortest correct
>> ;; version of the two is used.
>> ;; X - number to convert, scalar or array, float or double
>> ;; FORMAT - optional format to use (set to '(E)' or '(D)' for max precision)
>> ;; DCONVERT - set this if the output should be double precision
>> function inputform_float, x, format, dconvert=dcon
>> n = n_elements(x)
>> str = string(x(*), format=format)
>> sz = size(x) & tp = sz(sz(0)+1)
>>
>> ;; Sorry, there appears to be no other way to make nice looking
>> ;; floating point numbers.
>> str1 = string(x(*), format='(G0)')
>> if tp EQ 4 then x1 = float(str1)
>> if tp EQ 5 then x1 = double(str1)
>> wh = where(x-x1 EQ 0, ct)
>> if ct GT 0 then str(wh) = str1(wh)
>> str1 = 0
>> str = strtrim(str,2)
>>
>> p = strpos(str(0), 'E') ;; Make sure at least one element is float-type
>> ;; Note, the space is needed in case the string is placed inside
>> ;; another expression down the line.
>> if p LT 0 then begin
>> if keyword_set(dcon) then str(0) = str(0) + 'D' $
>> else str(0) = str(0) + 'E'

```
>>    endif
>>    if keyword_set(dcon) then begin
>>       ;; Convert from floating to double
>>       p = strpos(str, 'E')
>>       wh = where(p GE 0, ct)
>>       for i = 0L, ct-1 do begin
>>          str1 = str(wh(i))
>>          strput, str1, 'D', p(wh(i))
>>          str(wh(i)) = str1
>>       endfor
>>    endif
>>    ;; Construct format like (N(A,:,","))
>>    fmt = '('+strtrim(n,2)+'(A,:,","))'
>>    return, string(str, format=fmt)
>>  end
>>
>>
>>
>>  --
>>   ------------------------------------------------------------- --------------
>>  Craig B. Markwardt, Ph.D.      EMAIL: craigmnet@REMOVEcow.physics.wisc.edu
>>  Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
>>   ------------------------------------------------------------- --------------
```

Subject: Re: Converting Doubles to Strings
Posted by R.Bauer on Mon, 07 Nov 2005 19:53:50 GMT

View Forum Message <> Reply to Message

David Fanning wrote:

```
>  Folks,
>
>  I've run into an interesting problem. I have a double precision
>  number that I wish to convert to a string (so I can put it into
>  a text widget, for example). I don't know ahead of time how many
>  significant digits will be in this number. The number could
>  look like this 45.6, or this 123456789.123456789, or this
>  -22.1234567890. If it looks like the latter number, I am
>  having a very hard time converting it to a string. For example,
>  this doesn't work:
>
>     IDL> v = String(-22.1234567890, Format='(D0)')
>     IDL> Print, v
>           -22.123457
>
```

David

the sky is falling down

Why do you think thats 22.1234567890 is of type double. This is float!

Probably it's only a type mismatch in this example here.

I prefer the 'G' format

```
IDL> print, String(-22.1234567890D, Format='(G)')
      -22.12345678900000
```

instead of

```
IDL> print, String(-22.1234567890D, Format='(D)')
    -22.1234567889999987
```

cheers
Reimar

> Is this a bug in IDL? Or am I overlooking something?
>
> Cheers,
>
> David

Subject: Re: Converting Doubles to Strings
Posted by biophys on Mon, 07 Nov 2005 22:17:20 GMT
View Forum Message <> Reply to Message

Hi, David

I think I've got a more decent solution by assuming that double precision numbers have exactly 14 significant digits and throw away zeros at the end. In IDL help, it says that double type has appoximately 14 digits of significance. So I am sure my solution will fail somewhere. However, so far it works pretty nice. Please let me know if any bugs should be fixed or any improvement should be made. A few examples here,

```
IDL> print,dbl2str(-22.12)
-22.12E
IDL> print,dbl2str(-22.12d)
-22.12D
```

```
IDL> print,dbl2str(-22.1234567890)
-22.12346E
IDL> print,dbl2str(-22.1234567890d)
-22.123456789D
IDL> print,dbl2str(!dpi*1e20)
3.1415927165501E+20
IDL> print,dbl2str(!dpi*1d20)
3.1415926535897D+20
```

here's the code,

```
;converting float/double type numer into its original inputform
function dbl2str, a

tp=size(a,/type)
if tp ne 4 and tp ne 5 then begin
   a=double(a)
   tp=5
endif
tps=tp eq 4?'E':'D'

rawstr=strtrim(string(a,format='(g)'),2);full width G format
sign=strmid(rawstr,0,1) eq '-'?'-':'';extract sign
rawstr=sign eq ''?rawstr:strmid(rawstr,1);throw away sign
epos=strpos(rawstr,'e')
indx=epos gt -1?strmid(rawstr,epos+1):'';extract exponent index
rawstr=indx eq ''?rawstr:strmid(rawstr,0,epos);throw away exponent part
dpos=strpos(rawstr,'.');extract decimal point position


;rounding process(assume 14 significant digits)
outstr=strarr(15)
for i=0,14 do outstr[i]=strmid(rawstr,i,1)
aux=fix(strmid(rawstr,16,1)) ge 5?1:0
for i=14, 0, -1 do begin
   if i ne dpos then begin
 sumstr=strtrim(string(aux+fix(outstr[i])),2)
 sumlen=strlen(sumstr)
 outstr[i]=sumstr[sumlen-1]
 aux=sumlen eq 1?0:1
   endif
endfor

;throw away '0's at the end
ii=14
while outstr[ii] eq '0' do begin
ii=ii-1
endwhile
```

```
;reconstruct the inputform
saux=aux ne 0?'1':''
outstr=sign+saux+strjoin(outstr[0:ii])+tps+indx

return,outstr

end
```

cheers,
bp

David Fanning wrote:
> Folks,
>
> I've run into an interesting problem. I have a double precision
> number that I wish to convert to a string (so I can put it into
> a text widget, for example). I don't know ahead of time how many
> significant digits will be in this number. The number could
> look like this 45.6, or this 123456789.123456789, or this
> -22.1234567890. If it looks like the latter number, I am
> having a very hard time converting it to a string. For example,
> this doesn't work:
>
> IDL> v = String(-22.1234567890, Format='(D0)')
> IDL> Print, v
> -22.123457
>
> Is this a bug in IDL? Or am I overlooking something?
>
> Cheers,
>
> David
> --
> David Fanning, Ph.D.,
> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Converting Doubles to Strings
Posted by biophys on Mon, 07 Nov 2005 22:39:06 GMT
View Forum Message <> Reply to Message

```
outstr[i]=sumstr[sumlen-1]
       ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

sorry, this line should be

```
outstr[i]=strmid(sumstr,sumlen-1,1)
```

Subject: Re: Converting Doubles to Strings
Posted by R.Bauer on Tue, 08 Nov 2005 06:42:51 GMT
View Forum Message <> Reply to Message

biophys wrote:

Dont't do that conversion of an input float to double please use an internal
variable and look at this nice article

http://www.dfanning.com/math_tips/double.html

cheers
Reimar

> Hi, David
>
> I think I've got a more decent solution by assuming that double
> precision numbers have exactly 14 significant digits and throw away
> zeros at the end. In IDL help, it says that double type has
> appoximately 14 digits of significance. So I am sure my solution will
> fail somewhere. However, so far it works pretty nice. Please let me
> know if any bugs should be fixed or any improvement should be made. A
> few examples here,
>
> IDL> print,dbl2str(-22.12)
> -22.12E
> IDL> print,dbl2str(-22.12d)
> -22.12D
> IDL> print,dbl2str(-22.1234567890)
> -22.12346E
> IDL> print,dbl2str(-22.1234567890d)
> -22.123456789D
> IDL> print,dbl2str(!dpi*1e20)
> 3.1415927165501E+20
> IDL> print,dbl2str(!dpi*1d20)
> 3.1415926535897D+20
>
> here's the code,
>
> ;converting float/double type numer into its original inputform
> function dbl2str, a
>
> tp=size(a,/type)
> if tp ne 4 and tp ne 5 then begin
> a=double(a)
> tp=5
> endif
> tps=tp eq 4?'E':'D'

>
> rawstr=strtrim(string(a,format='(g)'),2);full width G format
> sign=strmid(rawstr,0,1) eq '-'?'-':'';extract sign
> rawstr=sign eq ''?rawstr:strmid(rawstr,1);throw away sign
> epos=strpos(rawstr,'e')
> indx=epos gt -1?strmid(rawstr,epos+1):'';extract exponent index
> rawstr=indx eq ''?rawstr:strmid(rawstr,0,epos);throw away exponent part
> dpos=strpos(rawstr,'.');extract decimal point position
>
>
> ;rounding process(assume 14 significant digits)
> outstr=strarr(15)
> for i=0,14 do outstr[i]=strmid(rawstr,i,1)
> aux=fix(strmid(rawstr,16,1)) ge 5?1:0
> for i=14, 0, -1 do begin
> if i ne dpos then begin
> sumstr=strtrim(string(aux+fix(outstr[i])),2)
> sumlen=strlen(sumstr)
> outstr[i]=sumstr[sumlen-1]
> aux=sumlen eq 1?0:1
> endif
> endfor
>
> ;throw away '0's at the end
> ii=14
> while outstr[ii] eq '0' do begin
> ii=ii-1
> endwhile
>
> ;reconstruct the inputform
> saux=aux ne 0?'1':''
> outstr=sign+saux+strjoin(outstr[0:ii])+tps+indx
>
> return,outstr
>
> end
>
> cheers,
> bp
>
> David Fanning wrote:
>> Folks,
>>
>> I've run into an interesting problem. I have a double precision
>> number that I wish to convert to a string (so I can put it into
>> a text widget, for example). I don't know ahead of time how many
>> significant digits will be in this number. The number could
>> look like this 45.6, or this 123456789.123456789, or this

>> -22.1234567890. If it looks like the latter number, I am
>> having a very hard time converting it to a string. For example,
>> this doesn't work:
>>
>> IDL> v = String(-22.1234567890, Format='(D0)')
>> IDL> Print, v
>> -22.123457
>>
>> Is this a bug in IDL? Or am I overlooking something?
>>
>> Cheers,
>>
>> David
>> --
>> David Fanning, Ph.D.,
>> Fanning Software Consulting, Inc.
>> Coyote's Guide to IDL Programming: http://www.dfanning.com/
