Subject: Re: negation operator
Posted by Foldy Lajos on Thu, 13 Jul 2006 14:15:21 GMT
View Forum Message <> Reply to Message

Hi,

On Thu, 13 Jul 2006, greg michael wrote:

> This is odd...
>
> IDL> print,-100 > -90 < 90
>
> print,-100 > -90 < 90
>   ^
> Syntax error.
> IDL> print,-100 > (-90) < 90
>   -90
>
> Is there any reason why a negation operator should ever come lower than
> anything else?
>
> Greg
>
>

yes, operator precedence. <, > and unary - has the same precedence in IDL,
so they can not follow each other immediately.

regards,
lajos

ps: in this respect I have followed the C precedence rules in FL, where
unary + and - have higher precedence than * (multiplication). So in FL
the above statements are correct and give the same result.

Subject: Re: negation operator
Posted by JD Smith on Thu, 13 Jul 2006 16:17:20 GMT
View Forum Message <> Reply to Message

On Thu, 13 Jul 2006 06:31:18 -0700, greg michael wrote:

> This is odd...
>
> IDL> print,-100 > -90 < 90
>
> print,-100 > -90 < 90
>   ^

> Syntax error.
> IDL> print,-100 > (-90) < 90
>   -90
>
> Is there any reason why a negation operator should ever come lower than
> anything else?

You get the same problem with exponentiation, which is 2 steps up in
the operator precedence chain (see
http://www.dfanning.com/misc_tips/precedence.html).

IDL> print, 2.^-2

print, 2.^-2
          ^
% Syntax error.
IDL> print, 2.^(-2)
    0.250000

So it's not that fact that `-' shares precedence with '>', just that
both are left-right associative, so you can't put two adjacent. This
doesn't always hold though... for example, how about `++' (2nd highest
precedence) and '~' (2nd to last precedence):

IDL> a=1
IDL> print,~++a
 0

but how about:

IDL> print,~-a

print,~-a
        ^
% Syntax error.

This *should* work. Negation is priority 5, logical negation is
priority 8, so it should evaluate -a first, and then ~(-a). Not sure
if this should be considered a bug or just a weakness of the
interpreter.

JD

Hi,

On Thu, 13 Jul 2006, JD Smith wrote:

> IDL> a=1
> IDL> print,~++a
> 0
>
> but how about:
>
> IDL> print,~-a
>
> print,~-a
>   ^
> % Syntax error.
>
> This *should* work. Negation is priority 5, logical negation is
> priority 8, so it should evaluate -a first, and then ~(-a). Not sure
> if this should be considered a bug or just a weakness of the
> interpreter.
>

well, this should work according to the documentation, but does not.
I think this is a bug in the documentation, logical negation has the
same precedence as bitwise negation and unary -. IDL/GDL/FL agree on this.

regards,
lajos

Subject: Re: negation operator
Posted by greg michael on Fri, 14 Jul 2006 15:02:02 GMT
View Forum Message <> Reply to Message

Yes, you're right - I'd got used to the quirk with exponents. I've only
just realised, though, that it appears to mean that there's no such
thing as a negative constant in IDL. They're always calculated values.
Maybe that's normal - I don't know.

greg

JD Smith wrote:
> On Thu, 13 Jul 2006 06:31:18 -0700, greg michael wrote:
>
>> This is odd...
>>
>> IDL> print,-100 > -90 < 90
>>

>> print,-100 > -90 < 90
>>   ^
>> Syntax error.
>> IDL> print,-100 > (-90) < 90
>>   -90
>>
>> Is there any reason why a negation operator should ever come lower than
>> anything else?
>
> You get the same problem with exponentiation, which is 2 steps up in
> the operator precedence chain (see
> http://www.dfanning.com/misc_tips/precedence.html).
>
> IDL> print, 2.^-2
>
> print, 2.^-2
>   ^
> % Syntax error.
> IDL> print, 2.^(-2)
>   0.250000
>
> So it's not that fact that `-' shares precedence with '>', just that
> both are left-right associative, so you can't put two adjacent. This
> doesn't always hold though... for example, how about `++' (2nd highest
> precedence) and '~' (2nd to last precedence):
>
> IDL> a=1
> IDL> print,~++a
> 0
>
> but how about:
>
> IDL> print,~-a
>
> print,~-a
>   ^
> % Syntax error.
>
> This *should* work. Negation is priority 5, logical negation is
> priority 8, so it should evaluate -a first, and then ~(-a). Not sure
> if this should be considered a bug or just a weakness of the
> interpreter.
>
> JD