Subject: Re: AND statements
Posted by steinhh on Mon, 01 Mar 1999 08:00:00 GMT
View Forum Message <> Reply to Message

In article <36da962e.18718769@146.80.9.44> philaldis@geocities.com (Phil Aldis) writes:

> So, to avoid that you have to do some pretty messy code. Say for
> example I've got :
>
> IF Ptr_Valid(ThisPointer) THEN BEGIN
> IF Size(*ThisPointer, /type) EQ 10 THEN BEGIN

Yep, that's right. You do get used to it, though, even if you're grown up with C style logical operators, taking advantage of the non-evaluation of unnecessary parts for every little scrap of efficiency improvement.

> However, I want to execute the same bit of code if it fails the
> Ptr_Valid and the Size(*ThisPointer, type0 EQ 10, so as far as I can
> see, (and I realise that I may be missing something pretty blatent),
> you have to use flags

[..snip..]

> While obviously this is not the end of the world, there could be more
> complex examples, and the code does look messy.

Yes, though you learn to rewrite those statements somewhat, like this:

```
flag = NOT ptr_valid(thispointer)
IF NOT flag then flag = size(*thispointer,/type) EQ 10

IF NOT flag THEN BEGIN
  ;; Pointer is valid and points to type 10
END ELSE BEGIN
  ;; Pointer is not valid or doesn't point to type 10
END
```

There could be some improvement with the ?: construct:

```
flag = NOT (ptr_valid(thispointer) ? size(*thispointer,/type) eq 10 : 0b)
```

should be equivalent to the first two lines in my example.
In fact, you could rewrite your original code like this:

```
if (ptr_valid(ptr) ? size(*ptr,/type) eq 10 : 0b) then print,*ptr $
```

else flag=1b

I'm not sure whether *I* would use the ?: construct in cases like this.... it looks more messy to me, in fact..

Regards,

Stein Vidar

Subject: Re: AND statements
Posted by philaldis on Mon, 01 Mar 1999 08:00:00 GMT
View Forum Message <> Reply to Message

>> If you write some code like this:
>>
>> test=Ptr_New()
>>
>> IF Ptr_Valid(test) AND Size(*test, /type) NE 10 THEN print, *test
>>
>> ....can you always guarantee that it will not try to evaluate the second
>> statement if the first one was false - or is this a dangerous tactic to
>> adopt?
>
> Nope. In fact, I can guarantee that it *will* evaluate the whole
> logical expression. IDL is in this respect totally unlike C.
>
> If the urge is big enough, one could ask RSI nicely to implement
> operators like "AND THEN" and "OR ELSE" used like this:
>
> IF ptr_valid(test) AND THEN size(*test,/type) ne 10 then print,*test
>
> IF error_occurred OR ELSE check_for_error() then print,"Error"
>
> "AND THEN" works like C &&
> "OR ELSE" works like C ||, i.e. check_for_error() isn't called
> if "error_occurred" is already true.
>

> Stein Vidar

So, to avoid that you have to do some pretty messy code. Say for example I've got :

```
IF Ptr_Valid(ThisPointer) THEN BEGIN
 IF Size(*ThisPointer, /type) EQ 10 THEN BEGIN
   .
   .
```

.
.
.
ENDIF

ENDIF

However, I want to execute the same bit of code if it fails the
Ptr_Valid and the Size(*ThisPointer, type0 EQ 10, so as far as I can
see, (and I realise that I may be missing something pretty blatent),
you have to use flags

IF Ptr_Valid(ThisPointer) THEN BEGIN
 IF Size(*ThisPointer, /type) EQ 10 THEN BEGIN
.
.
.
.
.
ENDIF ELSE flag = 1

ENDIF ELSE flag = 1

IF flag .....

While obviously this is not the end of the world, there could be more
complex examples, and the code does look messy.

Cheers,
Phil

Subject: Re: AND statements
Posted by steinhh on Mon, 01 Mar 1999 08:00:00 GMT
View Forum Message <> Reply to Message

In article <36D9C642.99878C2A@geocities.com> Phil Aldis
<philaldis@geocities.com> writes:

> Hi,
> If you write some code like this:
>
> test=Ptr_New()
>
> IF Ptr_Valid(test) AND Size(*test, /type) NE 10 THEN print, *test
>
> ....can you always guarantee that it will not try to evaluate the second
> statement if the first one was false - or is this a dangerous tactic to

> adopt?

Nope. In fact, I can guarantee that it *will* evaluate the whole
logical expression. IDL is in this respect totally unlike C.

If the urge is big enough, one could ask RSI nicely to implement
operators like "AND THEN" and "OR ELSE" used like this:

```
  IF ptr_valid(test) AND THEN size(*test,/type) ne 10 then print,*test

  IF error_occurred OR ELSE check_for_error() then print,"Error"
```

"AND THEN" works like C &&
"OR ELSE" works like C ||, i.e. check_for_error() isn't called
if "error_occurred" is already true.

Operators with such names exist in e.g. Simula, IIRC.

I wouldn't be holding my breath, though.

Regards,

Stein Vidar

Subject: Re: AND statements
Posted by Steve Scheele on Wed, 03 Mar 1999 08:00:00 GMT
View Forum Message <> Reply to Message

>> If you write some code like this:

>> test=Ptr_New()

>> IF Ptr_Valid(test) AND Size(*test, /type) NE 10 THEN print, *test

>> ....can you always guarantee that it will not try to evaluate the second
>> statement if the first one was false - or is this a dangerous tactic to
>> adopt?

I recently took an IDL course from RSI and asked the instructor almost
exactly the same question. I got the same answer you have been getting - no!

It is possible is C and it is not regarded as dangerous. I asked the
instructor to request that this capability be added - I don't hold out much
hope.

Subject: Re: AND statements
Posted by Steve Scheele on Fri, 05 Mar 1999 08:00:00 GMT
View Forum Message <> Reply to Message

I wrote the previous message late at night - it really wasn't what I was thinking. I think that compound AND statements can be concisely written as;

```
If  Not (A EQ 0) Then If B/A GT 5 Then . . .
```

Formed in this manner, the second part will not be evaluated is the first part is false. The problem that I have is with the OR statement. Something like

```
If A EQ 0 OR B/A GT 5
```

will blow up on A=0

I haven't really figured out a good way to implement this type of OR statement.