Subject: Re: Named Indices
Posted by John-David T. Smith on Wed, 27 Sep 2000 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> J.D. Smith (jdsmith@astro.cornell.edu) writes:
>
>> A slightly more portable solution, in the sense that it requires no modification
>> of startup code but simply inclusion of a program somewhere on the path
>> (presumably in the same directory tree as the program which would use it), is a
>> special purpose function. E.g.
>>
>> plot, findgen(11), LINESTYLE=linestyle(/DASH)
>>
>> You could even make an uber-function that encapsulates all such "named
>> entity"<->"integer value" constant mappings:
>>
>> const(/LINESTYLE,/DASH) ; produces 2
>>
>> const(/AXIS_STYLE,/FORCE,/SUPPRESS) ; produces 4 or 2 = 6
>>
>> const(/SIZE,/INTEGER) ; produces 2
>>
>> const(/WIDGET_DRAW_TYPE,/BUTTON_RELEASE) ; produces 1
>>
>> const(/AXIS_NUMBER,/X) ; produces 0
>>
>> const(/COLOR_TABLE,/PRISM) ; produces 6
>>
>> etc.
>>
>> The problem is you have to maintain it with new versions of IDL, and ensure
>> backwards compatibility too. Other issues crop up, such as private color table
>> lists, etc. It would definitely help readability though. Sounds like a great
>> beginner programming project that would readily educate one on the subtle
>> distinctions among keyword_set(), n_elements() ne 0, and arg_present().
>
> I thought about a function too, although not such
> a generic one. But I got this far:
>
> FUNCTION LS, Solid=Solid, Dot=Dot, Dash=Dash, DashDot=DashDot...
>
> When I realized I was in trouble with keyword names.
> In the end, I decided cryptic numbers were better
> than a long-winded explanation of how keyword names
> work. :-(
>

> But what I'm really surprised you didn't offer was
> a way to write a polymorphic object to do the job. :-)

I said it was a *beginner* function to code up, David. Intermediate users would of course wrap it in a superclass so that new maps could be created at will by self-aware plotting entities which lie hiding silently under your terminal and creep out at night to plot things you'll never see. But then again, you'll need to create that object sometime, perhaps in the startup file and then... o damn we're back to where we started.

Ahh yes, the familiar "Ambiguous keyword abbreviation" a.k.a. a.k.a. The only workaround being carnal manipulation of the _EXTRA structure. So that makes this not so beginner a problem. You could always use strings instead, but that's an extra character to type:

const(/LINESTYLE,'dotdash')

The solution is left as an exercise.

JD

JD

--
J.D. Smith /*\ WORK: (607) 255-6263
Cornell University Dept. of Astronomy */ (607) 255-5842
304 Space Sciences Bldg. /*\ FAX: (607) 255-5875
Ithaca, NY 14853 */

Subject: Re: Named Indices
Posted by davidf on Wed, 27 Sep 2000 07:00:00 GMT
View Forum Message <> Reply to Message

J.D. Smith (jdsmith@astro.cornell.edu) writes:

> A slightly more portable solution, in the sense that it requires no modification
> of startup code but simply inclusion of a program somewhere on the path
> (presumably in the same directory tree as the program which would use it), is a
> special purpose function. E.g.
>
> plot, findgen(11), LINESTYLE=linestyle(/DASH)
>
> You could even make an uber-function that encapsulates all such "named
> entity"<->"integer value" constant mappings:
>
> const(/LINESTYLE,/DASH) ; produces 2
>

> const(/AXIS_STYLE,/FORCE,/SUPPRESS) ; produces 4 or 2 = 6
>
> const(/SIZE,/INTEGER) ; produces 2
>
> const(/WIDGET_DRAW_TYPE,/BUTTON_RELEASE) ; produces 1
>
> const(/AXIS_NUMBER,/X) ; produces 0
>
> const(/COLOR_TABLE,/PRISM) ; produces 6
>
> etc.
>
> The problem is you have to maintain it with new versions of IDL, and ensure
> backwards compatibility too. Other issues crop up, such as private color table
> lists, etc. It would definitely help readability though. Sounds like a great
> beginner programming project that would readily educate one on the subtle
> distinctions among keyword_set(), n_elements() ne 0, and arg_present().

I thought about a function too, although not such
a generic one. But I got this far:

FUNCTION LS, Solid=Solid, Dot=Dot, Dash=Dash, DashDot=DashDot...

When I realized I was in trouble with keyword names.
In the end, I decided cryptic numbers were better
than a long-winded explanation of how keyword names
work. :-(

But what I'm really surprised you didn't offer was
a way to write a polymorphic object to do the job. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Named Indices
Posted by promashkin on Wed, 27 Sep 2000 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:

> And most of us (present company included) don't have any
> idea what that code does five minutes after we write the
> program.

Sure we do. Five minutes, five days and five weeks later, it still sits
on our FTP server, and we are the only ones who ever downloads it when
we happen to be at some other location :-)

Cheers,
Pavel

P.S. I swear I downloaded your FSC_PSConfig! This, actually, does mean
that you have no idea what is happening to it :-) but neither do I :-(

Subject: Re: Named Indices
Posted by John-David T. Smith on Wed, 27 Sep 2000 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> Jason P. Meyers (jpm7934@cis.rit.edu) writes:
>
>> I am brand new to IDL programming. I am using version 5.3.1 on a PC
>> and started going through some examples from my Digital Image Processing
>> (DIP) class as well as some from the online manuals and Dave Fanning's
>> book. I got to wondering if there exists (either built-in or developed
>> by someone else) a list of named indices. Such a beast would be useful
>> for specifying things line styles using names instead of numbers.
>>
>> For example,
>>
>> PLOT, time, curve, LineStyle=LongDash
>>
>> makes more sense to me than
>>
>> PLOT, time, curve, LineStyle=5
>>
>> One idea I had was to create a structured system variable named !I (for
>> indices) which then has the various indices as fields. For example:
>>
>> !I.Solid = 0
>> !I.Dot = 1
>> !I.Dash = 2
>> !I.DashDot = 3
>> !I.Dash3Dot = 4
>> !I.LongDash = 5
>>

>> I couldn't find a reference to such a system variable. Furthermore, I
>> don't even know if it is possible to create new system variables like
>> this. Finally, if something like this can be created, I don't want to
>> reinvent the wheel if it already exists.
>
> Well, it *is* possible to do something like this, since
> you can make new system variables. In fact, to do what
> you want to do requires an IDL statement like this:
>
> DefSysV, '!LS', {solid:0, dot:1, dash:2, dashdot:3, dashdotdot:4, longdash:5}, 1
>
> I used !LS for "line style", but what name you choose is up to you.
> You can put this kind of statement in, for example, an IDL
> start-up file.
>
> The principle reason this is not done more often
> is that programs that rely on it:
>
> Plot, Findgen(11), Linestyle=!LS.Dash
>
> don't work when they are passed along to someone else.
> You always forget to give them your start-up file, or
> they got it, but they forget to run it, etc. So most of
> us stay with the lowest common denominator code:
>
> Plot, Findgen(11), Linestyle=2
>
> And most of us (present company included) don't have any
> idea what that code does five minutes after we write the
> program.
>

A slightly more portable solution, in the sense that it requires no modification of startup code but simply inclusion of a program somewhere on the path (presumably in the same directory tree as the program which would use it), is a special purpose function. E.g.

plot, findgen(11), LINESTYLE=linestyle(/DASH)

You could even make an uber-function that encapsulates all such "named entity"<->"integer value" constant mappings:

const(/LINESTYLE,/DASH) ; produces 2

const(/AXIS_STYLE,/FORCE,/SUPPRESS) ; produces 4 or 2 = 6

const(/SIZE,/INTEGER) ; produces 2

const(/WIDGET_DRAW_TYPE,/BUTTON_RELEASE) ; produces 1

const(/AXIS_NUMBER,/X) ; produces 0

const(/COLOR_TABLE,/PRISM) ; produces 6

etc.

The problem is you have to maintain it with new versions of IDL, and ensure backwards compatibility too. Other issues crop up, such as private color table lists, etc. It would definitely help readability though. Sounds like a great beginner programming project that would readily educate one on the subtle distinctions among keyword_set(), n_elements() ne 0, and arg_present().

JD

--
J.D. Smith /*\ WORK: (607) 255-6263
Cornell University Dept. of Astronomy */ (607) 255-5842
304 Space Sciences Bldg. /*\ FAX: (607) 255-5875
Ithaca, NY 14853 */

Subject: Re: Named Indices
Posted by davidf on Wed, 27 Sep 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Jason P. Meyers (jpm7934@cis.rit.edu) writes:

> I am brand new to IDL programming. I am using version 5.3.1 on a PC
> and started going through some examples from my Digital Image Processing
> (DIP) class as well as some from the online manuals and Dave Fanning's
> book. I got to wondering if there exists (either built-in or developed
> by someone else) a list of named indices. Such a beast would be useful
> for specifying things line styles using names instead of numbers.
>
> For example,
>
> PLOT, time, curve, LineStyle=LongDash
>
> makes more sense to me than
>
> PLOT, time, curve, LineStyle=5
>
> One idea I had was to create a structured system variable named !I (for
> indices) which then has the various indices as fields. For example:
>
> !I.Solid = 0

>   !I.Dot = 1
>   !I.Dash = 2
>   !I.DashDot = 3
>   !I.Dash3Dot = 4
>   !I.LongDash = 5
>
> I couldn't find a reference to such a system variable. Furthermore, I
> don't even know if it is possible to create new system variables like
> this. Finally, if something like this can be created, I don't want to
> reinvent the wheel if it already exists.

Well, it *is* possible to do something like this, since
you can make new system variables. In fact, to do what
you want to do requires an IDL statement like this:

DefSysV, '!LS', {solid:0, dot:1, dash:2, dashdot:3, dashdotdot:4, longdash:5}, 1

I used !LS for "line style", but what name you choose is up to you.
You can put this kind of statement in, for example, an IDL
start-up file.

The principle reason this is not done more often
is that programs that rely on it:

   Plot, Findgen(11), Linestyle=!LS.Dash

don't work when they are passed along to someone else.
You always forget to give them your start-up file, or
they got it, but they forget to run it, etc. So most of
us stay with the lowest common denominator code:

   Plot, Findgen(11), Linestyle=2

And most of us (present company included) don't have any
idea what that code does five minutes after we write the
program.

And, anyway, making code "readable" is not really the
true programmer's solution anyway. If you want to earn
the big bucks, "cryptic" is better. :-)

If everyone in your workgroup would agree to use the same
IDL startup file, however, this would work great.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155