Subject: HASH question
Posted by Gray on Sat, 05 Mar 2011 13:41:53 GMT
View Forum Message <> Reply to Message

Hi all,

I have a bunch of information which I'd like to store in an organized fashion:
~IDs of some stars
~Stellar types
~Magnitudes and fluxes in different images

One way I could store the information would be as an array of structures, with each element being a single star, but I don't a priori know how many stars I have, and to find a particular star I'd have to search on the ID element. So, I could use a HASH of structures so I could index by ID, which would be ideal, but then assigning values to the individual tags of the structures is much more complicated. I could instead have a bunch of hashes, one for each type of information, but that would get pretty unwieldy.

So, IDL gurus, anyone have a suggestion for how to organize this most efficiently and elegantly? Thanks!

--Gray

Subject: Re: HASH question
Posted by Jeremy Bailin on Mon, 07 Mar 2011 19:15:56 GMT
View Forum Message <> Reply to Message

> An additional "but": If the hash keys to access the data is a vector (e.g. representing the "many of them") so would
> bethe new values (you can't assume that the new values would all be the same, right?). Assuming it was possible, do you
> think doing something like,
>
> hashtable[ids_to_change].type = array_of_new_values
>
> is o.k.? I don't (but remember, I'm a physicist who writes code so what the hell would I know? :o) That would make it
> way to easy to assign the wrong value. What if the two vectors, "ids_to_change" and "array_of_new_values", were
> different sizes? Of what if some of the keys in the "ids_to_change" vector didn't yet exist in the hash?

I do think it's okay - in fact, that's exactly what I'd expect to be able to do. And I would expect it to behave more or less in the same way as if hashtable were an array of structures which was

indexed by ids_to_change: if they don't have the same number of elements, then a runtime error is thrown (a la "Array subscript for XXXX must have same size as source expression". And if there are elements of ids_to_change that are not yet in the hash, they should be created, the same as if you assigned an individual value.

Admittedly, my experience with hashes are mainly from perl, which probably colours what I think the behaviour ought to be.

-Jeremy.

Subject: Re: HASH question
Posted by pgrigis on Mon, 07 Mar 2011 20:32:16 GMT
View Forum Message <> Reply to Message

On Mar 5, 8:41 am, Gray <grayliketheco...@gmail.com> wrote:
> Hi all,
>
> I have a bunch of information which I'd like to store in an organized
> fashion:
> ~IDs of some stars
> ~Stellar types
> ~Magnitudes and fluxes in different images
>
> One way I could store the information would be as an array of
> structures, with each element being a single star, but I don't a
> priori know how many stars I have, and to find a particular star I'd
> have to search on the ID element. So, I could use a HASH of
> structures so I could index by ID, which would be ideal, but then
> assigning values to the individual tags of the structures is much more
> complicated. I could instead have a bunch of hashes, one for each type
> of information, but that would get pretty unwieldy.
>
> So, IDL gurus, anyone have a suggestion for how to organize this most
> efficiently and elegantly? Thanks!
>
> --Gray

Well since I don;t have IDL 8.0 yet. I can't really comment
on the hashes method, but what's wrong with an array of structures?

allstars=replicate({starID:'',spectralType:'',magnitude:0.0} ,
100)
allstars[0].starID='Sirius' & allstars[0].spectralType='A1' &
allstars[0].magnitude=-1.46
allstars[1].starID='Delta Pavonis' & allstars[1].spectralType='G8' &
allstars[1].magnitude=3.56

etc.

Single stars can easily be located with where:

```
ind=where(allstars.starID EQ 'Delta Pavonis',count)
if count EQ 1 then print,'The magnitude of '+allstars[ind].starID+' is
'+strtrim(allstars[ind].magnitude,2)
```

Ciao,
Paolo

Subject: Re: HASH question
Posted by chris_torrence@NOSPAM on Mon, 07 Mar 2011 21:00:35 GMT
View Forum Message <> Reply to Message

Hi all,

You could also use a nested hash of hashes. For example:

```
h = HASH('Sirius', HASH('Color': 'blue', 'Size': 'big'), 'Betelgeuse',
HASH(...), ...)
```

In IDL 8.1 you will be able to index into array/list/hash elements
within a Hash (or List) using simple array indexing. So in the above
case, you would be able to do:

```
  print, h['Sirius', 'Color']
```

And it will print out "blue".

This also works for assignment as well.

```
  h['Sirius', 'Color'] = 'white'
```

In IDL 8.0 you could do this, but you would need to use parentheses...

The only problem with using a Hash is that the keys are not predefined
like in a named structure - you could easily miss adding a field or
spell a field incorrectly.

Just a thought.

Cheers,
Chris
ITTVIS

Subject: Re: HASH question

Posted by penteado on Mon, 07 Mar 2011 21:02:34 GMT
View Forum Message <> Reply to Message

On Mar 7, 5:32 pm, Paolo <pgri...@gmail.com> wrote:
> Single stars can easily be located with where:
>
> ind=where(allstars.starID EQ 'Delta Pavonis',count)
> if count EQ 1 then print,'The magnitude of '+allstars[ind].starID+' is
> '+strtrim(allstars[ind].magnitude,2)

I would not call that "easily". The two problems are mentioned in the first question: "but I don't a priori know how many stars I have, and to find a particular star I'd have to search on the ID element."

The search is awkward and not efficient, the arrays cannot have elements dynamically added or removed, and they would not avoid repetitions (elements with the same key).

Subject: Re: HASH question
Posted by penteado on Mon, 07 Mar 2011 21:05:15 GMT
View Forum Message <> Reply to Message

On Mar 7, 6:00 pm, Chris Torrence <gorth...@gmail.com> wrote:
> Hi all,
>
> You could also use a nested hash of hashes. For example:
>
> h = HASH('Sirius', HASH('Color': 'blue', 'Size': 'big'), 'Betelgeuse',
> HASH(...), ...)
>
> In IDL 8.1 you will be able to index into array/list/hash elements
> within a Hash (or List) using simple array indexing. So in the above
> case, you would be able to do:
> print, h['Sirius', 'Color']
> And it will print out "blue".
>
> This also works for assignment as well.
> h['Sirius', 'Color'] = 'white'

Nice. Just what I was doing with an inherited class. Any predictions on when 8.1 will be out? Maybe I will not have to finish the half-written classes I have.

Subject: Re: HASH question
Posted by chris_torrence@NOSPAM on Mon, 07 Mar 2011 21:38:22 GMT

View Forum Message <> Reply to Message

On Mar 7, 2:05 pm, Paulo Penteado <pp.pente...@gmail.com> wrote:
> On Mar 7, 6:00 pm, Chris Torrence <gorth...@gmail.com> wrote:
>
>> Hi all,
>
>> You could also use a nested hash of hashes. For example:
>
>> h = HASH('Sirius', HASH('Color': 'blue', 'Size': 'big'), 'Betelgeuse',
>> HASH(...), ...)
>
>> In IDL 8.1 you will be able to index into array/list/hash elements
>> within a Hash (or List) using simple array indexing. So in the above
>> case, you would be able to do:
>> print, h['Sirius', 'Color']
>> And it will print out "blue".
>
>> This also works for assignment as well.
>> h['Sirius', 'Color'] = 'white'
>
> Nice. Just what I was doing with an inherited class. Any predictions
> on when 8.1 will be out? Maybe I will not have to finish the half-
> written classes I have.

Soon enough that you should not have to finish your classes... :-)

-Chris

Subject: Re: HASH question
Posted by Gray on Tue, 08 Mar 2011 12:23:26 GMT
View Forum Message <> Reply to Message

On Mar 7, 4:38 pm, Chris Torrence <gorth...@gmail.com> wrote:
> On Mar 7, 2:05 pm, Paulo Penteado <pp.pente...@gmail.com> wrote:
>
>
>
>
>
>
>
>
>
>> On Mar 7, 6:00 pm, Chris Torrence <gorth...@gmail.com> wrote:
>
>>> Hi all,

>
>>> You could also use a nested hash of hashes. For example:
>
>>> h = HASH('Sirius', HASH('Color': 'blue', 'Size': 'big'), 'Betelgeuse',
>>> HASH(...), ...)
>
>>> In IDL 8.1 you will be able to index into array/list/hash elements
>>> within a Hash (or List) using simple array indexing. So in the above
>>> case, you would be able to do:
>>> print, h['Sirius', 'Color']
>>> And it will print out "blue".
>
>>> This also works for assignment as well.
>>> h['Sirius', 'Color'] = 'white'
>
>> Nice. Just what I was doing with an inherited class. Any predictions
>> on when 8.1 will be out? Maybe I will not have to finish the half-
>> written classes I have.
>
> Soon enough that you should not have to finish your classes... :-)
>
> -Chris

So, here's an update. I did end up going with a HASH of structures, and just dealt with the difficulty. Note that a HASH indexed by an array of keys returns a LIST, not an array (since the values which have been indexed are not necessarily homogeneous). However, I was able to make good use of the HasKey() method, so my code snippet looks something like this:

```
collect_my_info, id, type, mag1, mag2
tmp = replicate({mystruct},n_elements(id))
old = where(myHash.haskey(id),nold)
if (nold gt 1) then tmp[old] = (myHash[id[old]]).toArray() $
  else if (nold eq 1) then tmp[old] = myHash[id[old]]
tmp.id = id & tmp.type = type & tmp.mag1 = mag1 & tmp.mag2 = mag2
myHash[id] = temporary(tmp)
```

A little unwieldy, but the HasKey() method is very useful so I don't have to search on my keys individually.

Subject: Re: HASH question
Posted by pgrigis on Tue, 08 Mar 2011 15:05:06 GMT
View Forum Message <> Reply to Message

On Mar 7, 4:02 pm, Paulo Penteado <pp.pente...@gmail.com> wrote:

> On Mar 7, 5:32 pm, Paolo <pgri...@gmail.com> wrote:
>
>> Single stars can easily be located with where:
>
>> ind=where(allstars.starID EQ 'Delta Pavonis',count)
>> if count EQ 1 then print,'The magnitude of '+allstars[ind].starID+' is
>> '+strtrim(allstars[ind].magnitude,2)
>
> I would not call that "easily". The two problems are mentioned in the
> first question: "but I don't a priori know how many stars I have, and
> to find a particular star I'd have to search on the ID element."
>
> The search is awkward and not efficient,
well i never thought of "where" as being particularly awkward
inefficient, yes if used a lot
alternatively the stars could be sorted alphabetically by ID
to allow for fast search

> the arrays cannot have elements dynamically added or removed,

you can do concatenatation to add, say, another bunch of 100 stars
stars can be removed by setting the ID to the empty string or
using array concatenation yet again
not extremly efficient but works

> and they would not avoid
> repetitions (elements with the same key).
stars are unique - so why would he have duplicates?

Ciao,
Paolo

Subject: Re: HASH question
Posted by David Fanning on Tue, 08 Mar 2011 15:11:27 GMT
View Forum Message <> Reply to Message

Paolo writes:

>> and they would not avoid
>> repetitions (elements with the same key).
> stars are unique - so why would he have duplicates?

Probably for the same reason that whenever a web site
asks for my e-mail address I know I can expect about
five of the same messages from them every time they
send one out! :-(

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: HASH question
Posted by penteado on Tue, 08 Mar 2011 15:56:10 GMT
View Forum Message <> Reply to Message

On Mar 8, 12:05 pm, Paolo <pgri...@gmail.com> wrote:
> well i never thought of "where" as being particularly awkward
> inefficient, yes if used a lot
> alternatively the stars could be sorted alphabetically by ID
> to allow for fast search

Sorted arrays would not be any faster with where(). It would take a
custom search function to do it.

>> the arrays cannot have elements dynamically added or removed,
>
> you can do concatenatation to add, say, another bunch of 100 stars
> stars can be removed by setting the ID to the empty string or
> using array concatenation yet again
> not extremly efficient but works

Yes, but it is awkward. One big reason for using dynamic containers is
to avoid all this juggling, allowing one to just do the operation one
is interested in doing. Like just adding an element. The same goes for
retrieving or setting an element from a hash: one can just do the
access, without having to do searches. That way code is not
complicated to do a conceptually simple task.

Subject: Re: HASH question
Posted by pgrigis on Tue, 08 Mar 2011 18:12:05 GMT
View Forum Message <> Reply to Message

On Mar 8, 10:56 am, Paulo Penteado <pp.pente...@gmail.com> wrote:
> On Mar 8, 12:05 pm, Paolo <pgri...@gmail.com> wrote:

>
>> well i never thought of "where" as being particularly awkward
>> inefficient, yes if used a lot
>> alternatively the stars could be sorted alphabetically by ID
>> to allow for fast search
>
> Sorted arrays would not be any faster with where(). It would take a
> custom search function to do it.
yes, value_locate

>
>>> the arrays cannot have elements dynamically added or removed,
>
>> you can do concatenatation to add, say, another bunch of 100 stars
>> stars can be removed by setting the ID to the empty string or
>> using array concatenation yet again
>> not extremly efficient but works
>
> Yes, but it is awkward. One big reason for using dynamic containers is
> to avoid all this juggling, allowing one to just do the operation one
> is interested in doing. Like just adding an element. The same goes for
> retrieving or setting an element from a hash: one can just do the
> access, without having to do searches. That way code is not
> complicated to do a conceptually simple task.

Agreed that it is more complicated to do that with structures.

Ciao,
Paolo

Subject: Re: HASH question
Posted by penteado on Tue, 08 Mar 2011 18:15:11 GMT
View Forum Message <> Reply to Message

On Mar 8, 3:12 pm, Paolo <pgri...@gmail.com> wrote:
>> Sorted arrays would not be any faster with where(). It would take a
>> custom search function to do it.
>
> yes, value_locate

With the addition of tests to find out if there was a match.
