Subject: Re: HASH question
Posted by Jeremy Bailin on Mon, 07 Mar 2011 11:37:28 GMT
View Forum Message <> Reply to Message

On Saturday, March 5, 2011 8:41:53 AM UTC-5, Gray 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

I haven't used hashes in idl, but I think that a hash of structures makes the most sense. What makes that too complicated?

-Jeremy.

Subject: Re: HASH question
Posted by Paul Van Delst[1] on Mon, 07 Mar 2011 15:34:32 GMT
View Forum Message <> Reply to Message

Jeremy Bailin wrote:
> On Saturday, March 5, 2011 8:41:53 AM UTC-5, Gray 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
>
> I haven't used hashes in idl, but I think that a hash of structures makes the most sense. What makes that too complicated?

Well, you wouldn't be able to access the individual elements of the structure values in the hash without first pulling
it out. E.g.

```
IDL> z=hash()
IDL> x={id:123,name:'blue',type:5,flux:3.14e+07}
IDL> z[x.id]=x
IDL> x={id:75,name:'red',type:5,flux:2.7e+07}
IDL> z[x.id]=x
IDL> help, z
Z               HASH  <ID=8  NELEMENTS=2>
IDL> print, z.keys()
         123
          75
```

Let's say I want to change the "type" of the added star with id 75 from 5 to 4, i.e. it is in error

```
IDL> help, z[75]
** Structure <95dd194>, 4 tags, length=24, data length=24, refs=4:
   ID              LONG                 75
   NAME            STRING    'red'
   TYPE            LONG                 5
   FLUX            FLOAT       2.70000e+07
```

I can't just do:

```
IDL> z[75].type = 4
% Illegal subscript range: Z.
% Error occurred at: $MAIN$
% Execution halted at: $MAIN$
```

I would have to extract it, change it, and then put it back:

```
IDL> a = z[75]
IDL> help, a
** Structure <95dd194>, 4 tags, length=24, data length=24, refs=3:
   ID              LONG                 75
   NAME            STRING    'red'
   TYPE            LONG                 5
   FLUX            FLOAT       2.70000e+07
IDL> a.type = 4
IDL> z[a.id] = a
IDL> print, z
123: {        123 blue           5  3.14000e+07}
75: {         75 red            4  2.70000e+07}
```

Now, while I don't think that's a particularly onerous thing to do, the OP might.

Not being an OOP expert I may be blowing smoke out of my proverbial, but I think the way IDL does this is The Better Way
- encapsulation and information hiding are the two OOP concepts that I find most frequently influence the way I write
code (OO and regular old procedural) such that it is reusable, extendable, and easily maintained.

cheers,

paulv

Subject: Re: HASH question
Posted by Jeremy Bailin on Mon, 07 Mar 2011 15:57:17 GMT
View Forum Message <> Reply to Message

> Well, you wouldn't be able to access the individual elements of the structure values in the hash without first pulling
> it out. E.g.
>
> IDL> z=hash()
> IDL> x={id:123,name:'blue',type:5,flux:3.14e+07}
> IDL> z[x.id]=x
> IDL> x={id:75,name:'red',type:5,flux:2.7e+07}
> IDL> z[x.id]=x
> IDL> help, z
> Z   HASH <ID=8 NELEMENTS=2>
> IDL> print, z.keys()
>   123
>   75
>
> Let's say I want to change the "type" of the added star with id 75 from 5 to 4, i.e. it is in error
>

> IDL> help, z[75]
> ** Structure <95dd194>, 4 tags, length=24, data length=24, refs=4:
> ID LONG 75
> NAME STRING 'red'
> TYPE LONG 5
> FLUX FLOAT 2.70000e+07
>
> I can't just do:
>
> IDL> z[75].type = 4
> % Illegal subscript range: Z.
> % Error occurred at: $MAIN$
> % Execution halted at: $MAIN$
>
> I would have to extract it, change it, and then put it back:
>
> IDL> a = z[75]
> IDL> help, a
> ** Structure <95dd194>, 4 tags, length=24, data length=24, refs=3:
> ID LONG 75
> NAME STRING 'red'
> TYPE LONG 5
> FLUX FLOAT 2.70000e+07
> IDL> a.type = 4
> IDL> z[a.id] = a
> IDL> print, z
> 123: { 123 blue 5 3.14000e+07}
> 75: { 75 red 4 2.70000e+07}
>
> Now, while I don't think that's a particularly onerous thing to do, the OP might.
>
> Not being an OOP expert I may be blowing smoke out of my proverbial, but I think the way IDL does this is The Better Way
> - encapsulation and information hiding are the two OOP concepts that I find most frequently influence the way I write
> code (OO and regular old procedural) such that it is reusable, extendable, and easily maintained.

Really? That seems like a horrible design to me. It's fine if you're dealing with one element, but what if you want to change the type of many of them at once? Then there's no way to do it without a for loop?

-Jeremy.

Subject: Re: HASH question
Posted by Michael Galloy on Mon, 07 Mar 2011 17:43:41 GMT

On 3/7/11 8:34 AM, Paul van Delst wrote:
> Jeremy Bailin wrote:
>> On Saturday, March 5, 2011 8:41:53 AM UTC-5, Gray 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
>>
>> I haven't used hashes in idl, but I think that a hash of structures makes the most sense. What makes that too complicated?
>
> Well, you wouldn't be able to access the individual elements of the structure values in the hash without first pulling
> it out. E.g.
>
> IDL> z=hash()
> IDL> x={id:123,name:'blue',type:5,flux:3.14e+07}
> IDL> z[x.id]=x
> IDL> x={id:75,name:'red',type:5,flux:2.7e+07}
> IDL> z[x.id]=x
> IDL> help, z
> Z   HASH<ID=8 NELEMENTS=2>
> IDL> print, z.keys()
>   123
>   75
>
> Let's say I want to change the "type" of the added star with id 75 from 5 to 4, i.e. it is in error
>
> IDL> help, z[75]
> ** Structure<95dd194>, 4 tags, length=24, data length=24, refs=4:

> ID LONG 75
> NAME STRING 'red'
> TYPE LONG 5
> FLUX FLOAT 2.70000e+07
>
> I can't just do:
>
> IDL> z[75].type = 4
> % Illegal subscript range: Z.
> % Error occurred at: $MAIN$
> % Execution halted at: $MAIN$

You would need parentheses to get to the correct place, but that's still
not OK:

IDL> (z[123]).type = 6
% Attempt to store into an expression: Structure reference.
% Error occurred at: $MAIN$
% Execution halted at: $MAIN$

> I would have to extract it, change it, and then put it back:
>
> IDL> a = z[75]
> IDL> help, a
> ** Structure<95dd194>, 4 tags, length=24, data length=24, refs=3:
> ID LONG 75
> NAME STRING 'red'
> TYPE LONG 5
> FLUX FLOAT 2.70000e+07
> IDL> a.type = 4
> IDL> z[a.id] = a
> IDL> print, z
> 123: { 123 blue 5 3.14000e+07}
> 75: { 75 red 4 2.70000e+07}
>
> Now, while I don't think that's a particularly onerous thing to do, the OP might.
>
> Not being an OOP expert I may be blowing smoke out of my proverbial, but I think the way IDL
does this is The Better Way
> - encapsulation and information hiding are the two OOP concepts that I find most frequently
influence the way I write
> code (OO and regular old procedural) such that it is reusable, extendable, and easily
maintained.

You could do a hash of hashes:

[501]> star_table = hash()
[502]> star_table[123] = hash('id', 123, 'name', 'blue', 'type', 5,

'flux', 3.14e7)
[504]> print, star_table[123]
name: blue
id: 123
type: 5
flux: 3.14000e+07
[506]> print, (star_table[123])['type']
       5
[507]> (star_table[123])['type'] = 6
[508]> print, (star_table[123])['type']
       6

Mike
--
www.michaelgalloy.com
Research Mathematician
Tech-X Corporation

Subject: Re: HASH question
Posted by Paul Van Delst[1] on Mon, 07 Mar 2011 17:55:21 GMT
View Forum Message <> Reply to Message

Jeremy Bailin wrote:
>
> Really? That seems like a horrible design to me. It's fine if you're dealing with one element, but what if you want
> to change the type of many of them at once? Then there's no way to do it without a for loop?

Yes. But, I would argue that if you wanted to do that regularly, the hash of structures like I described is probably not
the best choice.

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?

cheers,

paulv