Subject: Re: Modifying Arrays and Structures in HASH's (hint: you can't)
Posted by Lajos Foldy on Mon, 29 Jul 2013 18:32:54 GMT
View Forum Message <> Reply to Message

On Monday, July 29, 2013 1:24:01 PM UTC+2, mschellens wrote:

> I would like to emphasize, that I revided this thread for the suggestion about _OVERLOADBRACKETSLEFTSIDE. This is not limited to a particular container type.
>
> What do you think about it?

```
FL> h=hash('s', {i:0})
FL> h['s'].i=123
FL> print, h
s: {     123}
FL>
FL> l=list({i:0})
FL> l[0].i=123
FL> print, l
{     123}
```

I added these features when HASH and LIST were implemented, so I agree with you, this should be the normal behavior for HASH, LIST and any future container.

> The strength of a LIST is the deletion and insertion of elements.
> Particular at the beginning or at the end (O(1)).

LIST::add, remove and access (subscripting) both have an index parameter, so the interface is that of a random access container, while the implementation is a sequential one. If the index parameter is there, users will use it :-)

> Not the traversal, what you measured.
>
> I am sure, one can build an example, where a list implementation based on an array will loose against a real linked list. What if you fill the complete LIST from the left (like: list.ADD,element[i],0)?

Yes, this will be slow with an array implementation, but can be easily cured with a deque.

>
> For an array based LIST, even as you demonstrate that it is for some cases more efficient, one could say: Why not using a PTR array then directly?

The array management is hidden in LIST.

> Ok, you got some comfort functions. Maybe there is even room (or need) for an array based container with ADD, REMOVE, ...

> But I think if the user uses a LIST he possibly really want one.

Are you sure? :-)

regards,
Lajos