Rocket software

Rocket Uniface Library 10.4

Example: Struct Collections

This example demonstrates that a `struct` variable is always a reference to a *collection* of Structs, and shows how you can work with collections.

A collection of one looks and works like a reference to a single Struct, but it remains a collection for which `$collSize` returns 1.

```
function STRUCT_COLLECTIONS
variables
  struct vStruct1, vStruct2, vSomeStructs, vOriginalCollection
  numeric I
  string vDescr
endvariables

  call printHeader("STRUCT_COLLECTIONS") ; display entry header in the message frame

  ; Prepare two structs
  vStruct1->$name = "Struct1"
  vStruct2->$name = "Struct2"

  ; Assign a member to the collection 'a',
  ; overwriting any existing members named 'a'
  vStruct1->a    = "A1"
  vStruct1->a{2} = "A2"  ; Add a member at position 2
  vStruct1->a{-1} = "A3" ; Append new member at end
  vStruct1->b    = "B"
  putmess vStruct1->$dbgstring  [1]

  ; The collection size of vStruct1->a (=3) and vStruct1->* (=4 all children)
  putmess "The collection size of vStruct1->a is %%(vStruct1->a->$collsize)%%%"
  ; Use the collection operator ->* to get all references to all children of a Struct
  putmess "The collection size of vStruct1->* is %%(vStruct1->*->$collsize)%%%" [2]
  ; Result:  The collection size of vStruct1->a is 3
  ;          The collection size of vStruct1->* is 4

  ; Looping over the collection vStruct1->a:
  putmess "%%^Loop over all members of vStruct1->a:"
  I = 1
  while (I <= vStruct1->a->$collSize) [3]
    putmess "  vStruct1->a{%%i%%} has value %%(vStruct1->a{I})%%%"
    I = I + 1
  endwhile

  ; Before reassigning Structs, save a reference to the original
  vOriginalCollection = vStruct1->a   [4]

  ; Update one specific member:
  vStruct1->a{1} = "A1 - updated"     [5]
  putmess "%%^Updated vStruct1->a{%%1%%}: %%(vStruct1->a{1})%%%"

  ; A collection of multiple Structs has no value: [6]
  putmess "%%^A collection of multiple Structs always returns an empty value"
  putmess "  vStruct1->a = %%(vStruct1->a)%%%"
```

```
  ; A collection with only one Struct behaves like a single Struct:
  vStruct1->a = "A1 - collection reassigned"
  putmess "However, if the collection contains a single Struct,"
  putmess " it is treated as a single Struct:"
  putmess "   vStruct1->a    = %%(vStruct1->a)%%%"
  putmess "   vStruct1->a{1} = %%(vStruct1->a{1})%%%"

  ; Restore the original collection
  vStruct1->a = vOriginalCollection   [4]

  ; ========================================================
  ;                  == Struct Functions ==   [7]

  putmess "%%^Using Struct functions on collections:"
  putmess "  All Structs vStruct1->a share the same name: [%%(vStruct1->a->$name)%%%]"
  reset $procerror
  putmess "    but not all vStruct1->*                 : [%%(vStruct1->*->$name)%%%]"
  vDescr = $item("DESCRIPTION", $procerrorcontext)
  putmess "    ProcScript error: %%$procerror%%%: %%vDescr%%%"
  putmess "  All Structs vStruct1->* share their parent : %\
   [%%(vStruct1->*->$parent->$name)]"
  vSomeStructs{1} = vStruct1->*
  vSomeStructs{-1} = vStruct1
  reset $procerror
  putmess "    but not all Structs in any collection    : [%%(vSomeStructs->$parent)]"
  vDescr = $item("DESCRIPTION", $procerrorcontext)
  putmess "    ProcScript error: %%$procerror%%%: %%vDescr%%%"

  ;  Manipulating collections using Struct functions:
  putmess "%%^Manipulating structs in collections: Assign $parent:"
  vStruct1->a->$parent = vStruct2           [8]
  putmess "  The members 'a' have moved to vStruct2:"
  putmess vStruct1->$dbgstring
  putmess vStruct2->$dbgstring

  putmess "%%^Manipulating Structs in collections: Assign $name:"
  putmess "  Members 'a' renamed to 'AAA':"
  vStruct2->a->$name = "AAA"                [9]
  putmess vStruct2->$dbgstring


  ; ========================================================
  ;                == Struct Assignment ==

  ; Assign a subnode to each 'a' member:
  vStruct2->*->x = "xyz"                    [10]
  putmess "%%^Member 'x' assigned to all Structs in a collection:"
  putmess vStruct2->$dbgstring
end ;- function STRUCT_COLLECTIONS
```

1. Struct1 has the following structure:

   ```
   [Struct1]
    [a] = "A1"
    [a] = "A2"
   ```

```
 [a] = "A3"
 [b] = "B"
```

2. A `struct` variable is a reference to any number of Structs. Thus this statement: `structVar = aStruct->*`

   assigns the references to all children of `aStruct`to `structVar`.

   The collection operator `->*`is used to access all references to all children of a Struct.

```
The collection size of vStruct1->a is 3
The collection size of vStruct1->* is 4
```

3. Use the `$collSize` Struct function to control the number of iterations in a `while` block that loops over the Struct.

```
Loop over all members of vStruct1->a:
 vStruct1->a{1} has value A1
 vStruct1->a{2} has value A2
 vStruct1->a{3} has value A3
```

4. For more information, see Example: Using Members after Member Reassignment.
5. You can update a specific member of a collection (even if it has no name) using the index operator `->{n}`.

```
Updated vStruct1->a{1}: A1 - updated
```

6. A collection of more than one Struct returns an empty value, but if the collection has only one Struct, it is treated as a single Struct.

```
A collection of multiple Structs always returns an empty value
 vStruct1->a =
However, if the collection contains a single Struct,
 it is treated as a single Struct:
 vStruct1->a = A1 - collection reassigned
 vStruct1->a{1} = A1 - collection reassigned
```

7. Struct functions and assignments on `vStruct1->a` apply to all Structs in a collection. `$dbgstring` is a good example: `vStruct1->*->$dbgstring` prints the collection of all children. Some other Struct functions are meaningful only under certain conditions. For example:

```
Using Struct functions on collections:
 All Structs vStruct1->a share the same name: [a]
 but not all vStruct1->* : []
 ProcScript error: -1151: Structs do not have a common name or parent
 All Structs vStruct1->* share their parent : [Struct1]
 but not all Structs in any collection : []
 ProcScript error: -1151: Structs do not have a common name or parent
```

8. Only one line of code is needed to move all members a from `Struct1` to `Struct2`: `vStruct1->a->$parent = vStruct2`.

```
Manipulating structs in collections: Assign $parent:
 The members 'a' have moved to vStruct2:
[Struct1]
 [b] = "B"
[Struct2]
 [a] = "A1"
 [a] = "A2"
 [a] = "A3"
```

9. Only one line of code is needed to rename multiple Structs. `vStruct2->a->$name = "AAA"`.

```
Manipulating Structs in collections: Assign $name:
 Members 'a' renamed to 'AAA':
[Struct2]
 [AAA] = "A1"
 [AAA] = "A2"
 [AAA] = "A3"
```

10. Only one line of code is needed to add a new member to multiple Structs: `vStruct2->*->x = "xyz"`.

```
Member 'x' assigned to all Structs in a collection:
[Struct2]
 [AAA]
 "A1"
 [x] = "xyz"
 [AAA]
 "A2"
 [x] = "xyz"
 [AAA]
 "A3"
 [x] = "xyz"
```