Rocket software

Rocket Uniface Library 10.4

Comparing Structs

You can compare `struct` variables with one another to check whether they have the same value, reference the same object, or have the same structure and annotations.

Structs can be compared using the standard ProcScript comparison operators: ==, !=, <, <=, >, and >=

The following rules apply when using comparison operators:

1. When a Struct is compared with a scalar value (either an ordinary non-Struct value, or with s scalar Struct), the comparison is done by value. Thus, the comparison operator compares the scalar value with the Struct's value.
2. If both sides of the comparison are Structs, the complete trees of both Structs are compared recursively. Two members are equal if they have the same number of members, with the same names and values.

To determine whether two `struct` variables refer to ths same physical Struct, the `$equalStructRefs` ProcScript function can be used.

Comparing Struct Values

When a Struct is compared with a scalar value (either an ordinary non-Struct value, or with s scalar Struct), the comparison is done by value. Thus, the comparison operator compares the scalar value with the Struct's value.

The value of a Struct is defined as:

- An empty value, if it has no scalar members.
- Its typed scalar member, if it has exactly one scalar member.
- The string concatenation of all its scalar members, if it has more than one scalar member.

The data type of scalar structs may differ, since Uniface implicitly converts the type, so `"1" == 1` is true.

If a scalar Struct has a data type, it is treated exactly as a variable of that data type. If the other side of the comparison is a Struct, it is interpreted as a string, which means its value is taken and used for the comparison.

Comparing Structs

When a Struct is compared with another Struct, and neither Struct is a scalar Struct, a deep Struct comparison is performed. This is a recursive comparison of the complete Struct trees, which stops when an inequality is encountered, or the Structs are found to be the same.

Any Struct, or any Struct expression, can be regarded as a list of zero or more references to single Structs. Thus, when two Structs are compared using any of the comparison operators, both Structs are treated as lists of single Structs.

A Struct comparison starts by comparing the number of items (`$collsize`) in each Struct list. If they differ, the comparison stops there. If they have the same number of items, each pair of single Structs is compared, in the order in which they occur. The comparison is recursive, so the first pair of Structs is compared down to the deepest level (assuming no inequality is found) before proceeding to the next pair of Structs in the Struct lists.

When comparing pairs of single Structs, the `$membersize` is compared first. If they differ, comparison stops. If they are the same, each corresponding pair of members is compared, in the order in which they occur.

Note: The names of the Structs are not significant when comparing Struct lists and single Structs. They

only play a role when comparing Struct members.

Thus, given the following Structs:

StructA	StructB

`[]`
` [a] = "apple"`
` [$tags]`
` [climate] = "moderate"`
` [b] = "banana"`
	`[]`
	` [x1] = "apple"`
	` [$tags]`
	` [climate] = "moderate"`
	` [x2] = "banana"`

The following comparison evaluates to true:

```
vStructA->* == vStructB->*
```

For each pair of members, the names are compared first, and if they are same, the values are compared. (The values can be scalar values or single Struct values.) If these are also the same, the annotations (`$tags`) of the Structs are compared.

One value is considered greater than another if it is a higher number or if it is higher alphabetically. Thus `b` is greater than `a`.

Consider the Struct in the first column, which is referenced by `vStruct`. The comparisons in the column to the right

Table: Struct Comparisons

vStruct	Condition is True

`[Example]`
` [p]`
` [a] = "apple"`
` [$tags]`
` [climate] = "moderate"`
` [b] = "banana"`
	`$equalstructrefs(vStruct->p->$parent, vStruct->q->$parent)`
	The parents of pand qare the same physical Struct.

`[q]`
` [a] = "apple"`
` [$tags]`
` [climate] = "moderate"`
` [b] = "banana"`
	`vStruct->q == vStruct->p`
	The members have the same names, tags and values.

`[r]`	`vStruct->r > vStruct->p`

vStruct	Condition is True
[a] = "apple" [$tags] [climate] = "moderate" [b] = "banana" [c] = "coconut"	Struct r has an additional member.
[s] [a] = "apple" [$tags] [climate] = "moderate"	vStruct->s < vStruct->p Struct s has fewer members than p.
[t] [a] = "apple" [$tags] [climate] = "moderate" [ba] = "banana"	vStruct->t > vStruct->p The names of the second members differ; ba comes after b in the alphabet, so t is greater than p.
[u] [a] = "apple" [$tags] [climate] = "moderate" [b] = "blueberry"	vStruct->u > vStruct->p The values of the second members differ; blueberry comes after banana in the alphabet, so u is greater than p.
[v] [a] = "apple" [$tags] [climate] = "cold" [b] = "banana"	vStruct->v < vStruct->p The tags of the first members differ; cold comes before moderate in the alphabet, so v is less than p.
[w] [a] = "apple" [$tags] [habitat] = "moderate" [b] = "banana"	vStruct->w > vStruct->p The tags of the first members differ; habitat comes after climate in the alphabet, w is greater than p.

vStruct	Condition is True
[x] [x1] = "apple" [$tags] [climate] = "moderate" [x2] = "banana"	vStruct->x->* == vStruct->p->* All Structs in the two lists are equal because names of Structs are not significant in comparing Struct lists and single Structs.

Related concepts

[$equalStructRefs]

Related reference

[Struct Leaves]