

Table of Contents

- 1 cmp
- 1.1 Generic Comparison Functions
- 1.2 Comparison by Key

cmp

This package provides utility functions for comparing values.

Generic Comparison Functions

The library provides generic comparison functions that work with any type implementing the Compare trait:

```
1
2  test "generic comparison" {
3
4      inspect(@cmp.maximum(3, 4), content="4")
5      inspect(@cmp.minimum(3, 4), content="3")
6  }
```

Comparison by Key

With @cmp.maximum_by_key() and @cmp.minimum_by_key(), it is possible to compare values based on arbitrary keys derived from them. This is particularly useful when you need to compare complex objects based on some specific aspect or field.

```
1
2  test "cmp_by_key" {
3      struct Person {
4          name : String
5          age  : Int
6      } derive(Show)
7
8
9      let s1 = "hello"
10     let s2 = "hi"
11     let longer = @cmp.maximum_by_key(s1, s2, String::length)
12     inspect(longer, content="hello")
13
14
15     let alice = { name: "Alice", age: 25 }
16     let bob   = { name: "Bob",   age: 30 }
17     let younger = @cmp.minimum_by_key(alice, bob, p => p.age)
18     inspect(younger, content="{name: \"Alice\", age: 25}")
19
20
21     let p1 = ("first", 1)
22     let p2 = ("second", 1)
23     let snd = (p : (_, _)) => p.1
24     assert_eq(@cmp.minimum_by_key(p1, p2, snd), p1)
25     assert_eq(@cmp.maximum_by_key(p1, p2, snd), p2)
26 }
```