Languages-beta: OC-L-Funcons-Index *

The PLanCompS Project

`OC-L-Funcons-Index.cbs` | PLAIN | PRETTY

OUTLINE

---

*Suggestions for improvement: `plancomps@gmail.com`.
Reports of issues: `https://github.com/plancomps/CBS-beta/issues`.

Computations

Normal computation

Flowing

[*Funcon* sequential
Alias seq
Funcon effect
Funcon if-true-else
Alias if-else
Funcon while-true
Alias while]

Giving

[*Funcon* initialise-giving
Funcon give
Funcon given
Funcon left-to-right-map
Funcon interleave-map]

Binding

[*Type* environments
Alias envs
Datatype identifiers
Alias ids
Funcon initialise-binding
Funcon bound-value
Alias bound
Funcon scope
Funcon accumulate
Funcon collateral
Funcon recursive]

Storing

[*Funcon* initialise-storing
Datatype variables
Alias vars
Funcon allocate-initialised-variable
Alias alloc-init
Funcon assign
Funcon assigned]

Interacting

Input

[*Funcon* read]

Output

[*Funcon* print]

Abnormal computation

Failing

[*Funcon* finalise-failing
Funcon fail
Funcon else
Funcon checked
Funcon check-true]

Throwing

[*Funcon* finalise-throwing
Funcon throw
Funcon handle-thrown]

Values

Value Types

[*Type* values
Alias vals
Funcon is-in-type
Alias is
Funcon when-true
Alias when
Type ground-values
Alias ground-vals
Funcon is-equal
Alias is-eq]

Primitive values

Booleans

[*Datatype* booleans
Alias bools
Funcon true
Funcon false
Funcon not
Funcon and
Funcon or]

Integers

[*Type* integers
Alias ints
Type bounded-integers
Alias bounded-ints
Type natural-numbers
Alias nats
Funcon natural-successor
Alias nat-succ
Funcon integer-add
Alias int-add
Funcon integer-subtract
Alias int-sub
Funcon integer-multiply
Alias int-mul
Funcon integer-divide
Alias int-div
Funcon integer-modulo
Alias int-mod
Funcon integer-absolute-value
Alias int-abs
Funcon integer-negate
Alias int-neg
Funcon integer-is-less
Alias is-less
Funcon integer-is-less-or-equal
Alias is-less-or-equal
Funcon integer-is-greater
Alias is-greater
Funcon integer-is-greater-or-equal
Alias is-greater-or-equal
Funcon decimal-natural
Alias decimal
Funcon integer-sequence]

Floats

[*Datatype* float-formats
Funcon binary64
Type floats
Funcon float-negate
Funcon float-absolute-value
Funcon float-add
Funcon float-subtract
Funcon float-multiply
Funcon float-divide
Funcon float-remainder
Funcon float-sqrt
Funcon float-float-power
Funcon float-floor
Funcon float-ceiling
Funcon float-truncate
Funcon float-log
Funcon float-log10
Funcon float-exp
Funcon float-sin
Funcon float-cos
Funcon float-tan
Funcon float-asin
Funcon float-acos
Funcon float-atan
Funcon float-sinh
Funcon float-cosh
Funcon float-tanh
Funcon float-atan2]

Characters

[*Type* characters
Alias chars
Funcon unicode-character
Alias unicode-char
Funcon ascii-character
Alias ascii-char
Funcon backspace
Funcon horizontal-tab
Funcon line-feed
Funcon carriage-return
Funcon backslash]

The null value

[*Datatype* null-type
Funcon null-value
Alias null]

Composite values

Sequences of values

[*Funcon* length
Funcon index
Funcon reverse
Funcon n-of
Funcon intersperse]

Tuples

[*Datatype* tuples
Funcon tuple-elements]

Lists

[*Datatype* lists
Funcon list
Funcon list-elements
Funcon list-nil
Alias nil
Funcon list-cons
Alias cons
Funcon list-head
Alias head
Funcon list-tail
Alias tail
Funcon list-length
Funcon list-append]

Strings

[*Type* strings
Funcon string-append
Funcon to-string]

Vectors

[*Datatype* vectors
Funcon vector
Funcon vector-elements]

Bits and bit vectors

[*Datatype* bit-vectors
Funcon bit-vector-not
Funcon bit-vector-and
Funcon bit-vector-or
Funcon bit-vector-xor
Funcon bit-vector-shift-left
Funcon bit-vector-logical-shift-right
Funcon bit-vector-arithmetic-shift-right
Funcon integer-to-bit-vector
Funcon bit-vector-to-integer
Funcon signed-bit-vector-maximum
Funcon signed-bit-vector-minimum]

Sets

[*Funcon* set
Funcon set-elements
Funcon is-in-set]

Maps

[*Type* maps
Funcon map
Funcon map-elements
Funcon map-lookup
Alias lookup
Funcon map-domain
Alias dom
Funcon map-override
Funcon map-unite]

Records

[*Datatype* records
Funcon record
Funcon record-map
Funcon record-select]

Variants

[*Datatype* variants
Funcon variant]

Abstraction values

Generic abstractions

[*Funcon* abstraction
Funcon closure]

Functions

[*Datatype* functions
Funcon function
Funcon apply
Funcon curry]

Patterns

[*Datatype* patterns
Funcon pattern
Funcon pattern-any
Funcon pattern-bind
Funcon pattern-else
Funcon pattern-unite
Funcon match
Funcon match-loosely
Funcon case-match]