FP trivia Sprachreferenz

2020-07-31

Namenskonventionen

name	der Name selbst
name	der Typus / die Klasse
(?)	Unsicherheiten
*	Fußnote / Hinweis

Datentypen

Datentyp	Syntax	Typbezeichner
data		// Allgemein
null	( )	_null
*int**	[_123]	_integer
real	_31.415e_123	_real
string	"abc"	_string
ident	abc oder +-*/	_ident
prefix	@...	_prefix
index	[abc]	_index
array	{a b c}	_array
error	(*index* _error ... ...)	_error
table/dict**	(a x b y c z)	// Paarweise
list	(a ; b ; c ;)	;
object	(*ident* :: a x b y c z)	::
turtle	(turtle :: ... ...)	// Objekt
combi	(*term* _combine .. *arg*)	_combine
monad	(*int* _act ... ...)	_act
bool	true oder false	// Idents
etc...		

*man beachte, daß *int*- und *dict*-Literale des Konstanten-Kombinators bedürfen!

Kommentare

codetext **//** *comment*

Definition von Bezeichnern

ident **==** *term*
ident **≡** *term*

Skriptaufbau

term definition1 definition2 definition3 ...

Ddot

prop = *head infix* **..** *tail*

Include-Dateien

coreimport == "*Script1.txt*" ; "*Script2.txt*" ; "*Script3.txt*" ; ... ;

userimport == "*Script1.txt*" ; "*Script2.txt*" ; "*Script3.txt*" ; ... ;

corepath?

userpath?

Listen/Dict Funktionen und Operatoren

dict = (first1 infix1 first2 infix2 ... ... firstm infixm)
list = (element0 ; element1 ; element2 ; ... ;)

[*i* **]** ° *list* -- *elementi*

head ° *dict* -- first
head ° *list* -- first
head ° *object* --

tail ° *dict* -- rest
tail ° *list* -- rest
tail ° *object* --

infix ° *dict* -- infixwert
infix ° *object* --

prop ° *first*,*infixwert*,*rest*, -- dict

top ° *dict* -- first
top ° *list* -- first

pop ° *dict* -- rest
pop ° *list* -- rest

tag ° *data* -- typus
tag ° *dict* -- infixwert

term ° *combi* -- termwert

arg ° *combi* -- argwert

termoarg -- term o arg

first **,** *rest* -- list
Appendleft

length ° *dict* -- *real*
length ° *list* -- *real*
Anzahl der Listenelemente.

reverse ° *dict* -- *dict*
reverse ° *list* -- *list*
reverse ° *object* --
Dreht die Listenelemente um.

data **distl** *list* -- matrix

list **distr** *data* -- matrix

dict **++** *dict* -- *dict*
list **++** *list* -- *list*
Verkettet die Listen.

dict **take** *num* -- *dict*
list **take** *num* -- *list*

Holt die ersten *num* Elemente aus der Liste.

dict **drop** *num* -- *dict*
list **drop** *num* -- *list*
Lässt die ersten *num* Elemente der Liste fallen.

trans ° *matrix* -- matrix
transpose ° *matrix* -- matrix

num **sel** *list* -- element

last ° *list* --

(*num* **r**) ° *list* --

tailr ° *list* -- list
tailr ° *dict* -- dict

rotl ° *list* -- list

rotr ° *list* -- list

iota ° *num* -- *list*
⍳ ° *num* -- *list*
Generiert eine Liste von Zahlen von 1 bis *num*.

iota0 ° *num* -- *list*
Generiert eine Liste von Zahlen von 0 bis *num*-1.

int **to** *int* -- list
real **to** *real* -- list

int **upto** *int* -- list
real **upto** *real* -- list

int **downto** *int* -- list
real **downto** *real* -- list

Mathematische Funktionen und Operatoren

int **+** *int* -- *int*
real **+** *real* -- *real*
Addition von Zahlen.

int **-** *int* -- *int*

real **-** *real* -- *real*
Subtraktion von Zahlen.

int ***** *int* -- *int*
real ***** *real* -- *real*
int **×** *int* -- *int*
real **×** *real* -- *real*
Multiplikation von Zahlen.

num **/** *num* -- *real*
num **÷** *num* -- *real*
Division von Zahlen.

int **^** *int* -- int
real **^** *real* -- real

int **idiv** *int* -- *int*
Integerdivision

int **imod** *int* -- *int*
Integermodulo

pred ° *int* -- int
pred ° *real* -- real

succ ° *int* -- int
succ ° *real* -- real

sign ° *int* -- int
sign ° *real* -- real

abs ° *int* -- *int*
abs ° *real* -- *real*
Betrag einer Zahl.

neg ° *int* -- *int*
neg ° *real* -- *real*
_ ° *int* -- *int*
_ ° *real* -- *real*
Negation einer Zahl.

round ° *num* -- *int*
Rundung zur Integerzahl.

trunc ° *num* -- int

int ° *num* -- *real*
Integeranteil der Zahl als Realzahl.

frac ° *num* -- real

float ° *num* -- *real*
Umwandlung zur Realzahl.

exp ° *real* -- *real*
Exponentialfunktion

ln ° *real* -- *real*
Natürlicher Logarithmus.

lg ° *real* -- *real*
Zehnerlogarithmus

ld ° *real* -- real

sq ° *int* -- *int*
sq ° *real* -- *real*
Quadrat einer Zahl.

sqrt ° *num* -- *real*
Quadratwurzel einer Zahl.

pi -- 3.141592653589793
Ludolfsche Zahl.(?)

2pi -- 6.283185307179586
Umfang des Einheitskreises.

sin ° *real* -- *real*
Sinusfunktion

cos ° *real* -- *real*
Cosinusfunktion

tan ° *real* -- *real*
Tangensfunktion

arcsin ° *real* -- real

arccos ° *real* -- real

arctan ° *real* -- *real*
Arcustangensfunktion

sinh ° *real* -- real

cosh ° *real* -- real

tanh ° *real* -- real

deg ° *num* -- *real*
Radiant-To-Degree-Funktion

rad ° *num* -- *real*
Degree-To-Radiant-Funktion

real **mod** *real* -- real

sum ° *list* -- *num*
Summe der Listenelemente.

prod ° *list* -- *num*
Produkt der Listenelemente.

avg ° *list* -- *real*
Durchschnittswert der Listenelemente.

integral

dd

Dictionary Operatoren und Kombinatoren

dict = (*value0 key0 value1 key1 value2 key2* ... ...)

_super

dict **get** *key* -- value

dict **put** *key*,*value*, -- dict

dict **iget** *ident* -- value
dict **iget** *index* -- value
API-Get für identische Keys.

dict **iput** *ident*,*value*, -- dict
dict **iget** *index*,*value*, -- dict

API-Put für identische Keys.

#*ident* ° *dict* -- value
(*ident* **_v**) ° *dict* -- value
Wert der Instanzenvariable.

(*ident* **:=** *value*) ° *dict* -- dict
Substitution (?)

func **<-** *key1* ; *key2* ; ... ;
func **←** *key1* ; *key2* ; ... ;
Assign-Kombinator, allgemein.

func **<-** *key1 isfunc1 key2 isfunc2* ... ...
func **←** *key1 isfunc1 key2 isfunc2* ... ...
Assign-Kombinator, typisiert.

Boole'sche Funtionen und Operatoren

bool = **true** oder **false**

'**true** -- *bool*
Wert für Wahr.

'**false** -- *bool*
Wert für Unwahr.

data **=** *data* -- *bool*
Prüfung auf Gleichheit.

data **<>** *data* -- *bool*
data **!=** *data* -- *bool*
data **≠** *data* -- *bool*
Prüfung auf Ungleichheit.

data **<** *data* -- bool

data **>** *data* -- bool

data **<=** *data* -- bool

data **>=** *data* -- bool

¬ ° *bool* -- *bool*

not ° *bool* -- *bool*
not ° *int* -- *int*
Nicht-Funktion

bool **and** *bool* -- *bool*
int **and** *int* -- *int*
Und-Operator

bool **or** *bool* -- *bool*
int **or** *int* -- *int*
Oder-Operator

bool **xor** *bool* -- *bool*
int **xor** *int* -- *int*
Exklusiv-Oder-Operator

isatom ° *data* -- bool

isprop ° *data* -- bool

islist ° *data* -- bool

isbool ° *data* -- bool

isnum ° *data* -- bool

iszero ° *data* -- bool

ispos ° *data* -- bool

isneg ° *data* -- bool

isnil (?)

ispreg (?)

isnull ° *data* -- bool
isint ° *data* -- bool
isreal ° *data* -- bool
isstring ° *data* -- bool
isident ° *data* -- bool
isprefix ° *data* -- bool
isindex ° *data* -- bool
isarray ° *data* -- bool
iscons ° *data* -- bool
iscombi ° *data* -- bool

isalt ° *data* -- bool
isobj ° *data* -- bool
isquote ° *data* -- bool
isivar ° *data* -- bool
isact ° *data* -- bool
Prädikate um den entsprechenden Datentyp zu überprüfen.

isbound ° *ident* -- bool
isbound ° *prefix* -- bool

isundef ° *data* -- bool
Prüfung auf _undef

iscomplex ° *complex* -- bool (?)

object **is** *ident* -- bool (?)

Kombinatoren für den Programmablauf (?)

combi = (*term* **_combine** *arg*)

func **_s**
Single-Funktions-Auswertung

' *literal*
literal **k**
literal **_q**
Konstanten-Kombinator

f **:** *x* // Ursprünglich
Applikation, deprecated // keine Application mehr

func1 **°** *func2*
func1 **o** *func2*
func1 ∘ *func2*
Komposition von Funktionen.

funktional **app** *argument*
Apply-Operator

func1 **,** *func2* **,** *func3* **,** ... **,**
Konstruktion von Listen.

test **->** *then* **|** *else*
test **→** *then* **|** *else*
test **->** *then* **;** *else*
Kondition mit Alternal.

test **->*** *func*
test **→*** *func*
while-Schleife

func **loopif** *test*
do-while-Schleife

list **map** *funktional*
Map-Operator

(*func* **aa**) ° *list*
(*func* **α**) ° *list*
Apply-to-All-Kombinator

list **insl** *funktional*
Insertl-Operator

list **insr** *funktional*
Insertr-Operator

(*func* ****) ° *list*
Insertr-Kombinator

list **filter** *funktional*
Filter-Operator

(*list*,*arg1*,*arg2*,...,) **map0** *funktional*

(*func* **aa0**) ° *list*,*arg1*,*arg2*,...,
Kombination aus **aa** und **distr**, erweitert.

func1 **ee** *func2*
ee ° *data*,*data*,
Eval-Eval-Kombinator für Infixnotation.

func1 **swee** *func2*
swee ° *data*,*data*,
Swap-Eval-Eval-Kombinator

(*func* **dip**) ° *list*

(*func* **dip**) ° *object*
Dipp-Kombinator (von Joy geklaut)

ifnull

ifprop

data1 **??** *data2* -- data

(func Y)

quote ° *data* -- func
Quote-Funktional

func1 **comp** *func2* -- func
Kompose-Funktional

Misc Funktionen und Operatoren

undef -- *error*
Funktion ist definiert als Undefiniert.

id ° *argument* -- *argument*
Identitätsfunktion.

out ° *argument* -- *argument* // *Seiteneffekt
Output für Debugging.

data **min** *data* -- *data*
min ° *data*,*data*, -- *data*
Minimum zweier Werte.

data **max** *data* -- *data*
max ° *data*,*data*, -- *data*
Maximum zweier Werte.

name ° *ident* -- string

body ° *ident* -- value

address ° *data* -- real

identlist -- *list*
Liste aller verwendeten Bezeichner.(?)

indexdict -- dict

_reserve

_undef

gc ° *argument* -- argument

String Funktionen und Operatoren

substring ° *string*,*num*,*num*, -- string

string **concat** *string* -- string
string **&** *string* -- string

length ° *string* -- real

string **mid** *num*,*num*, -- string

string **left** *num* -- string

string **right** *num* -- string

char ° *num* -- string

unicode ° *string* -- real

trim ° *string* -- string

triml ° *string* -- string

trimr ° *string* -- string

upper ° *string* -- string

lower ° *string* -- string

upper1lower ° *string* -- string

parse ° *string* -- list

value ° *string* -- data

string ° *data* -- string

unpack ° *string* -- list

pack ° *list* -- string

OOP

object = (*cap* **::** *inst*)

pair = *objekt* , *parameter* ,

self ° *pair*

para ° *pair*

index **op** *func*

index **swop** *func*

index **fn** *func*

cap ° *list* -- ( )
cap ° *object* -- (*cap* **::**)

ident **obj** *list* -- (*ident* **::** *list*)
ident **obj** *dict* -- (*ident* **::** *dict*)

ident **new** *parameter*

object **as** ident (?) -- object

box ° *primdata* -- object

unbox ° *object* -- primdata

object == .. { ( ) ... ... }
Objekt-Klasse

list == .. { object ... ... }
List-Klasse

dict == .. { object ... ... }
Dict-Klasse

Monaden und Effekte

monad = (*int* **_act** *dict*) // absolute

monad = (*index* **_act** *dict*) // relative

#it (?)

#self

#para

_bind
Continuation

_effects
Effekte

monad **>>** *term* -- monad

int **act** *dict* -- monad
index **act** *dict* -- monad
monad(?) **act** *dict* -- monad

monad **eff** *array* -- monad

monad **var** *data* -- monad
monad **var** *dict* -- monad

(*ident* **define** *dict*) ° *dict* -- monad
//(prefix define dict) ° dict

showgraph ° *dict* -- monad

(*data* **showinfo**) ° *dict* -- monad // *+ (*x* eff 'io)

(*data* **print**) ° *dict* -- monad // *+ (*x* eff 'io)

(*string* **input**) ° *dict* -- monad // *+ (*x* eff 'io)

(*fname* **loadtext**) ° *dict* -- monad // *+ (*x* eff 'io)

(*fname* **savetext** *string*) ° *dict* -- monad // *+ (*x* eff 'io)

(*string* **run**) ° *dict* -- monad // *+ (*x* eff 'io)

quit -- monad

io == .. { ... ... }
System-Effekte-Klasse

Laufzeitfehler(?)

error = (*index* **_error** *string* ; ... ...)

index **error** *string*, -- error

fail ° *argument* -- error
Gebrauch für Selektor-Signaturen(?)

stop ° *argument* -- error
Allgemein, z.B. Programmabbruch, etc

raise ° *string* -- exception
Eine Exception wird geworfen.

_error == .. { ... ... }
Klasse für Weiterleitungen...

Komplexe Zahlen

complex = (**complex ::** *real* **re** *real* **im**)

i -- (complex :: 0 re 1 im)
Quadratwurzel aus _1

real **cval** *real* -- *complex*
Zur Bildung einer komplexen Zahl aus Realzahlen.

re ° *complex* -- *real*
Realteil der komplexen Zahl.

im ° *complex* -- *real*
Imaginärteil der komplexen Zahl.

complex **+** *complex* -- *complex*
Addition von komplexen Zahlen.

complex **-** *complex* -- *complex*
Subtraktion von komplexen Zahlen.

complex ***** *complex* -- *complex*
complex **×** *complex* -- *complex*
Multiplikation von komplexen Zahlen.

complex **/** *complex* -- *complex*

complex ÷ *complex* -- *complex*
Division von komplexen Zahlen.

etc

complex == .. { dict ... ... }
Komplex-Klasse mit den komplexen Methoden.

Matrizen Funktionen und Operatoren

matrix = (*list* ; *list* ; ... ;)

IP ° *list*,*list*, // Backus Turing Lecture
list **IP** *list*

MM ° *matrix*,*matrix*, // Backus Turing Lecture
matrix **MM** *matrix*

det ° *matrix* -- real

inv ° *matrix* -- matrix

transpose ° *matrix* -- matrix

Turtlegraphics

turtle = (**turtle ::** *list* **stack** *real* **x** *real* **y** *real* **angle**
bool **pen** *num* **color** *num* **size** *num* **brush**)

pair = (*x* , *y* ,)

// 2pi

initturtle
'*turtle* **new** // empfohlen

pair **moveto** *turtle*

pair **moverel** *turtle*

real **move** *turtle*

real **turnto** *turtle*

real **turn** *turtle*

penup ° *turtle*

pendown ° *turtle*

num **pencolor** *turtle*

num **pensize** *turtle*

num **brushcolor** *turtle*

real **circle** *turtle*

rectangle ° *turtle* // rect

(**draw** eff io) ° *turtle* -- monad
Für die Zeichnung der Turtlespur.

#x ° *turtle* -- real
#y ° *turtle* -- real
#angle ° *turtle* -- real
etc
Attribute des Turtleobjektes.

colors == '(... ...)
#red ° colors für den Farbwert Rot.

turtle == .. { dict ... ... }
Turtle-Klasse,
durch Vererbung können auch eigene Turtle-Klassen erschaffen werden.

xlist (**plot0** eff io) 0-y -- monad