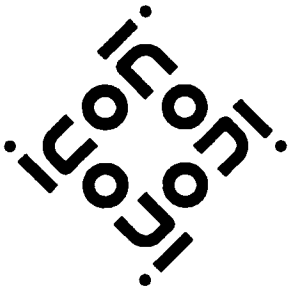


Icon Programming Language Version 8 Reference Sheet

Functions

abs(N)
acos(r)
any(c,s,i1,i2)
args(p)
asin(r)
atan(r1,r2)
bal(c1,c2,c3,s,i1,i2)
callout(x,x1,x2,...,xn)
center(s1,i,s2)
char(i)
close(f)
collect(i1,i2)
copy(x)
cos(r)
cset(x)
delete(X,x)
detab(s1,i1,i2,...,in)
display(i,f)
dtor(r)
entab(s1,i1,i2,...,in)
errorclear(i)
exit(i)
exp(r)
find(s1,s2, i1,i2)
get(L)
†getch()
†getche()
†getenv()
iand(i1,i2)
icom(i)
image(x)
insert(X,x1,x2)
integer(x)
ior(i1,i2)
ishift(i1,i2)
ixor(i1,i2)
†kbhit()
key(T)
left(s1,i,s2)
list(i,x)
log(r)
many(c,s,i1,i2)
map(s1,s2,s3)
match(s1,s2,i1,i2)
member(X,x)
mmout(s)
mmpause(s)
mmshow(x,s)
name(x)
move(i)
numeric(x)
open(s1,s2)
ord(s)
pop(L)
pos(i)
proc(x,i)
pull(L)
push(L,x)
put(L,x)
read(f)
reads(f,i)
real(x)
remove(s)
rename(s1,s2)
repl(s,i)
reverse(s)
right(s1,i,s2)
rtod(r)
runerr(i,x)
†save(s)
seek(f,i)
seq(i1,i2)
set(L)
sin(r)
sort(X,i)
sqrt(r)
stop(x1,x2,...,xn)
string(x)
†system(s)
tab(i)
table(x)
tan(r)
trim(s1,c)
type(x)
upto(c,s,i1,i2)
variable(s)
where(f)
write(x1,x2,...,xn)
writes(x1,x2,...,xn)

†These functions are not available on all implementations.

Syntactic Equivalents

$>	]
$<	[
$)	}
$(	{

Reserved Words

break	local
by	next
case	not
create	of
default	procedure
do	record
else	repeat
end	return
every	static
fail	suspend
global	then
if	to
initial	until
link	while

Keywords

&ascii	&lcase
&clock	&letters
&collections	&level
&cset	&line
&current	&main
&date	&null
&dateline	&output
&digits	&pos
&error	&random
&errorno	&regions
&errortext	&source
&errorvalue	&storage
&errout	&subject
&fail	&time
&features	&trace
&file	&ucase
&host	&version
&input	

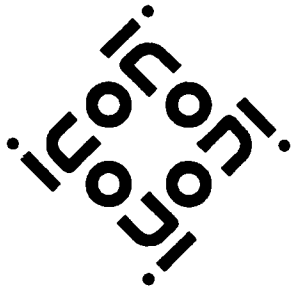
Data Types

co-expression	procedure
cset	real
file	set
integer	string
list	table
null	record types

Escape Sequences

\b	backspace
\d	delete (rubout)
\e	escape
\f	formfeed
\l	linefeed
\n	newline
\r	return
\t	horizontal tab
\v	vertical tab
\'	single quote
\"	double quote
\\	backslash
ddd	octal code
\x*dd*	hexadecimal code
\^*c*	control code

With thanks to Bob Alexander ...



Icon Programming Language Version 8 Reference Sheet

High-Precedence Expressions

(*expr*)	grouping
{*expr1*; *expr2*; ... }	compound
expr (*expr1*, *expr2*, ...)	invocation
expr {*expr1*, *expr2*, ... }	invocation
[*expr1*, *expr2*, ...]	list
expr.f	field reference
expr1 [*expr2*]	subscript
expr1 [*expr2* : *expr3*]	section
expr1 [*expr2* +: *expr3*]	section
expr1 [*expr2* −: *expr3*]	section

Prefix Expressions

not *expr*	success/failure reversal
| *expr*	repeated alternation
! *expr*	element generation
* *expr*	size
+ *expr*	numeric value
− *expr*	negative
. *expr*	value (dereference)
/ *expr*	null
\ *expr*	non-null
= *expr*	match and tab
? *expr*	random value
~ *expr*	cset complement
@ *expr*	activation
^ *expr*	refresh

Low-Precedence Expressions

break *expr*
case *expr0* of {... }
create *exp*
every *expr1* do *expr2*
fail
if *expr1* then *expr2* else *expr3*
next
repeat *expr*
return *expr*
suspend *expr1* do *expr2*
until *expr1* do *expr2*
while *expr1* do *expr2*

Expressions by Precedence

all high-precedence expressions

all prefix expressions

expr1 \ *expr2*	limitation
expr1 @ *expr2*	transmission
expr1 ! *expr2*	invocation
expr1 ^ *expr2* (right assoc.)	power
expr1 * *expr2*	product
expr1 / *expr2*	quotient
expr1 % *expr2*	remainder
expr1 ** *expr2*	intersection
expr1 + *expr2*	sum
expr1 − *expr2*	numeric difference
expr1 ++ *expr2*	union
expr1 −− *expr2*	cset or set difference
expr1 || *expr2*	string concatenation
expr1 ||| *expr2*	list concatenation
expr1 < *expr2*	...
expr1 <= *expr2*	...
expr1 = *expr2*	numeric comparison
expr1 >= *expr2*	...
expr1 > *expr2*	...
expr1 ~= *expr2*	...
expr1 << *expr2*	...
expr1 <<= *expr2*	...
expr1 == *expr2*	string comparison
expr1 >>= *expr2*	...
expr1 >> *expr2*	...
expr1 ~== *expr2*	...
expr1 === *expr2*	value comparison
expr1 ~=== *expr2*	...
expr1 | *expr2*	alternation
expr1 to *expr2* by *expr3*	integer generation
expr1 := *expr2* (all right assoc.)	assignment
expr1 <− *expr2*	reversible assignment
expr1 :=: *expr2*	exchange
expr1 <−> *expr2*	reversible exchange
expr1 op:= *expr2*	(aug. assignments)
expr1 ? *expr2*	string scanning
expr1 & *expr2*	conjunction

all low-precedence expressions