General conventions

The comment symbol is #; comments extend to the end of the line.

**Numeric constants** can be written in exponential notation (e.g., \(1.7 \times 10^{-4}\)) or in the radix notation \(<\text{radix}>r<constant>\), e.g., \(16r0D0A\) for hexadecimal \(0D0A\).

**String constants** are written between double quotes ("..."), Escapes include: \(\textbackslash b\) backspace, \(\textbackslash e\) escape, \(\textbackslash f\) formfeed, \(\textbackslash l\) linefeed, \(\textbackslash n\) newline, \(\textbackslash r\) return, \(\textbackslash t\) horizontal tab, \(\textbackslash '\) single quote, \(\textbackslash "\) double quote, \(\textbackslash ddd\) for octal code \(ddd\), and \(\textbackslash xdd\) for hexadecimal code \(dd\).

**Variable names** must start with a letter, followed by letters, digits, or underbar (_). They can be any length.

Built-in types

**Fundamental types**

- **integer**: Signed integers. They may have many digits (thousands, even) but extended-precision arithmetic can be slow.
- **real**: Real numbers.
- **string**: Strings of characters. Any character can be stored in a string, including nulls.
- **cset**: An improper subset of the character set.

**Structural types**

- **file**: An open file handle.
- **list**: A list of zero or more values of any type or mixture of types. Written as: \([e_1, e_2, \ldots, e_n]\)
- **null**: The type of variables to which no value is assigned.
- **procedure**: An Icon procedure.
- **table**: An associative array, containing a set of ordered pairs \((k, v)\) where each \(k\) is a unique key value and \(v\) is any value associated with that key.
- **co-expression**: See the manual.

Reserved words

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

global <name>, ...  
record <name>{<field>, ...}# Create with: var := <name>() 
local <name>, ...  
static <name>, ...

Control structures

break e          Exit the enclosing loop and produce e
case n of{      Produces the e_i whose x_i matches n
  x_1 : e_1
  x_2 : e_2
...          
  default: e_d
}
create x        Create a co-expression for x
every e_1 do e_2 Iterate over outcomes of e_1
fail            Fail out of the current procedure
if e_1 then e_2 [else e_3] Produces e_2 if e_1 succeeds, else e_3
next            Go to the top of the loop
not e           Fails if e succeeds
repeat e        Repeat forever (or until break)
return e        Return e to the caller
suspend e       Suspend procedure, return e
until e_1 do e_2 Loop while e_1 is false
while e_1 do e_2 Loop while e_1 is true
  e_1 | e_2   Outcome of e_1 followed by outcome of e_2
  | e        Produce e forever
  e \ n     Produce no more than n results from e
s ? e          Establish s as &subject, evaluate e

"Preprocessor" features

$define n text   Define a textual substitution
$else            Else construct for $ifdef and $ifndef
$endif           End of conditional compilation
$error           Signal a fatal compilation error
$ifdef n         Conditional compilation when n is defined
$ifndef n        Conditional compilation when n is undefined
#include "f"      Include file f
$undef n         Undefine a name
Operators

Precedence

\((e)\{e;\ldots\} [e,\ldots]\)  Highest precedence
\(e[e]\)
\(e(e)\)

All unary operators
\(\backslash @ ! ~ \)
\(* / \% **\)
\(+ - ++ --\)

All relationals
\(e \mid e \mid e \mid e\) [by \(e\)]

:= ○ :=
where ○ is any binary operator
?
&
;
,

Lowest precedence

Unary operators

\(+n\) Numeric value of \(n\)
\(-n\) Negative of \(n\)
\(?n\) Pseudorandom number \(\in [1, n]\) for integers, \(\in [0, 1]\)

for \(n \equiv 0\)

\(?x\) Randomly selected element of \(x\)
\(~c\) Set complement of \(c\) set
\(=s\) \(\text{tab(match}(s))\)
\(!x\) Sequence of elements from \(x\)
\(@e\) Outcome of activating co-expression \(e\)
\(^e\) Refresh co-expression \(e\)
\(*x\) Size of \(x\)
\(x\) Value of \(x\)
\(/x\) Is \(x\) undefined?
\(\backslash x\) Is \(x\) defined?

Binary operators

\(+ - * /\) Ambition, distraction, uglification, derision
\(\%\) Remainder
\(a^b\) \(a^b\)
\(< <= >= > =\) Numerical comparison
\(++\) Set union
\(--\) Set difference
\(**\) Set intersection
\(||\) String concatenation
\(<< <<< == >>>> ==\) Lexical comparison
\(||||\) Concatenate two lists
\(x \@ e\) Activate co-expression \(e\), transmit \(x\) to it

:= Assignment
### Built-in functions

#### Numeric functions

- `abs(x)`: Absolute value of integer or real
- `acos(x)`: Arccos, returns radians
- `asin(x)`: Arcsin, returns radians
- `atan(y, x)`: Arctan, default 2nd arg is 1
- `dtr(x)`: Degrees to radians
- `exp(x)`: Exponential
- `cos(x)`: Argument in radians
- `iand(a, b)`: Bitwise and
- `icom(a)`: Bitwise not (one’s complement)
- `ior(x, y)`: Bitwise or
- `ishift(x, n)`: Shift `x` left `n`; negative `n` for right shift
- `ixor(x, y)`: Bitwise exclusive or
- `log(x, b)`: `log_b(x)`, default `b` is `e`
- `rtod(x)`: Radians to degrees
- `sin(x)`: Argument in radians
- `sqrt(x)`: Square root
- `tan(x)`: Argument in radians

#### String functions

- `any(c, s, i, j)`: Fails if `s[i]` not `c`, else produces `i`
- `bal(c, c_l, c_r, s, i, j)`: Balances `c_l`, `c_r` up to `c`
- `center(s, n, p)`: Centers `s` in field of `n`, padded with `p`
- `detab(s, i_1, i_2, ..., i_n)`: Untabify; default `(9)`
- `entab(s, i_1, i_2, ..., i_n)`: Tabify
- `find(t, s, i, j)`: Find target `t` in source `s`
- `left(s, n, p)`: Flush-left `s` in field of `n`, padded with `p`
- `many(c, s, i, j)`: Scan `s` while `c` in `s`
- `map(s, f, t)`: Map characters from `f` to `t` in `s`. E.g., to upshift: `map(s, &lcase, &ucase)`
- `match(t, s, i, j)`: Is target `t` a prefix of `s`?
- `move(n)`: Advance `&pos` by `n`, return part skipped
- `pos(n)`: Is `&pos` at `n`?
- `repl(s, n)`: Replicate `n` copies of `s`
reverse(s)  Reverse string
right(s, n, p)  Flush right s in field of width n, pad p
tab(n)  Set &pos to n, return string from old &pos to there
trim(s, c)  Trim trailing characters ∈ c
upto(c, s, i, j)  Scan s until a character ∈ c

Structural functions

char(o)  Character whose ordinal is o
copy(x)  Structure copy
cset(x)  Cast to cset
delete(s, x)  Delete element x from set s
get(L)  Pop first element of list L
image(x)  String image of x
insert(s, x)  Insert x into set s
integer(x)  Cast to integer
list(n, x)  List of n elements equal to x
member(s, x)  Set membership: is x ∈ s?
numeric(x)  Cast to integer or real
ord(s)  Ordinal of 1-character strings
pop(L)  Pop first element of list L
pull(L)  Pop last element of list L
push(L, x)  Push x as first element of L
put(L, x)  Push x as last element of L
real(x)  Cast to real
seq(i, j)  Infinite sequence: i, i + j, i + 2j, . . .
set(L)  Set of distinct elements in list L
sort(a)  Sort a list
sort(t, k)  Sort table t. Produces a list of 2-elt lists (i, v) of indices and values; sort by i if k = 1, by v if k = 2.
sortf(x, k)  Sort lists and records, using the kth field as key
string(x)  Cast to string
table(x)  New table with initial value x
type(x)  String describing the type of x

System functions

chdir(s)  Change directory to s
close(f)  Close file
delay(i)  Delay i milliseconds
display(i, f)  Stack traceback with locals, i levels
exit(i)  Terminate with status i
flush(f)  Flush output to file f
function()  Generates names of all builtin functions
getch()  Get next character
getch()  Get next character, no echo
getenv(s)  Value of environmental variable s
kbhit()  Is there a character available for getch()?
open(name, options)  Open a file: options taken from r (read), w (write), 
a (append), b (read/write), c (create), t (enable line 
terminator translation), and u (inhibit line terminator 
translation).

name(v)  Name of variable v
read(f)  Return next line from f
reads(f, n)  Read next n characters from f, may be short, fails 
only if no characters remain
remove(s)  Delete file whose name is s
rename(s_o, s_n)  Rename file s_o as s_n
seek(f, p)  Set file position, origin 1; negative = before end
stop(e_1, ..., e_n)  Write the expressions and stop with exit status 1
system(s)  Execute a command line
variable(s)  Returns the variable named s
where(f)  Returns a file position
write(e_1, e_2, ...)  Write expressions, then newline
writes(e_1, e_2, ...)  Write expressions, no newline

Other operations

i to j [ by k ]  Arithmetic progression generator
    e_0(e_1, e_2, ...)  The outcome of e_n
    p(e_1, e_2, ...)  Procedure or function call
    s[i]  ith element of s
    s[i:j]  Elements i through j
    s[i+:j]  Elements i through i+j
    s[i:-j]  Elements i-j through i
    [e_1, e_2, ...]  Form a list from expressions

Keywords (built-in variables)

&ascii  a cset of 7-bit ASCII
&clock  hh:mm:ss
&cset  Full character set, universe for csets
&date  yyyy/mm/dd
&dateline  Sunday, December 7, 1941 7:01 am
&digits  012...9
&e  2.71828...
&errout  Standard error stream
&fail  Fails
&host  Host system identification
&input  Standard input stream
&letters  a...zA...Z
&lcase  a...z
&level  Current procedure call level
&main  Co-expression for the initial call of main()
&null  The null value
&output  Standard output stream
&phi  : Golden ratio φ, 1.61803...
&pi  : π, 3.14159...
&pos : Current scan position
&program : File name of executing program
&random : Pseudorandom seed
&source : Co-expression for the activator of the current co-expression
&subject : Current scan string
&time : Milliseconds of cpu time since program start
&trace : Controls procedure trace: 0 for none, -1 for ∞
&ucase : A...Z
&version : Current Icon version