ΔFIX Preprocessor Cheat Sheet

[ΔFIX Cheat Sheet.md]

Cat.	Item	Example

Cont.	Continuation line symbol… or .. (dots / diaereses are ignored)	a←1 + 2 + …
		  3 4 5
		s←'one two three ..
		  four five'
		[See also Continuing DQ Strings]

Cont.	Continuing Parenthetical Expressions Across lines	a←¯1+(2*
		  31 32 33)÷(
		  1+⍳3)

Cont.	Continuing SQ Strings Across Lines	'This is line 1 and line 2.' 'This is line 1 and line 2.'
Cont.	Continuing DQ Strings Across Lines	"This is line1 and line 2." ('This is line1',(⎕UCS 10),'and line 2.')
Cont.	Quotes with continuation line symbol … or ..	This is a cat.. alog. This is a cat ..alog.' 'This is a catalog. This is a cat alog.'

Where	Semicolon at end or beginning of line (outside parens, brackets, braces) represents ⍪ "where".
	
	For semicolons within parens, see Lists. Parens inside brackets follow APL standards.)
	
	Remember: "where" code is executed right to left.
		S← (A×x*2)+(B×x)+C
		  ; A←10
		  ; B←¯5
		  ; C←○1
		  ; x←⍳100
		S←(A×x*2)+(B×x)+C⍪A←10⍪B←¯5⍪C←○1⍪x←⍳10
		  S
		3.141592654 8.141592654 33.14159265 78.14159265
		143.1415927 228.1415927 333.1415927 458.1415927
		603.1415927
		  768.1415927

Where	What about "where" inside parentheses?
	Use ⍪, where you might have used ';'.
		S←myNS.((A×x*2)+(B×x)+C
		  ⍪ A←10
		  ⍪ B←¯5
		  ⍪ C←○1
		  ⍪ x←⍳100
		  )
		S←myNS.((A×x*2)+(B×x)+C⍪A←10⍪B←¯5⍪C←○1⍪x←⍳10)

Unicode	Decimal ⎕Unnn	⎕U123≡⎕UCS 123 ⋄ ⎕U123≡'{'
Unicode	Hexadecimal ⎕Unhhx	⎕U7BX≡⎕U123
Nums	Hexadecimal Integers dhhhx	1122X≡123X+0FFFX
Nums	Long number separator _ (underscore)	123_245_343_122.35 3.14159_26534
Atoms	Atoms consist of APL names, numbers, and APL strings.	:FOR a :IN `fred 'jack 123'… 3.14159 55

ΔFIX Preprocessor Cheat Sheet

[ΔFIX Cheat Sheet.md]

Cat.	Item	Example
Atoms	Atoms as names `atom1 atom2	colors←`red orange yellow reds← `red orange 1≡∧/reds∈colors
Atoms	Atoms as numbers	local←`CA 14850
Atoms	Atoms as strings `name1 'string2'	Last←`Smith 'Van Buren' Jones
Parms (Parameters)	Parameters	atom1 atom2→ arbitrary code [See Lists for examples]
Lists	Lists (code1 ; code2;)	Create mappings from names/numbers/strings to arbitrary code expressions
Lists	Ordinary code (code1; code2;)	test←(⍳3 ; ⍳4)
Lists	Function parameters	graph←(XY type 3→(⍳20)(1○⍳20); legend x→'Voltage'; legend y→'Amplitude')
Lists	With atoms	graph(type→`XY 3; smooth → `true; line color→`green; line height→`2.5 in)
Lists	Omitted parameters (code1;;code3)	address(2525; 'Cozy'; 'Lane'; ; ; 90212; USA) ⍝ city/state opal with zip
Lists	Null list ()	Always true: () ≡ ⍬
Name Suffixes	Is name defined? name..DEF ≡ (0<⎕NC 'name')	:IF print..DEF
Name Suffixes	Is name undefined? name..UNDEF ≡ .. (0≥⎕NC 'name')	:IF print..UNDEF
Name Suffixes	Put name in quotes: name..Q (possibly after macro or other processing). Internal quotes are doubled automatically.	⎕NPARTS fileName..Q ⍝ ⎕NPARTS 'fileName'
Name Suffixes	Get value of environment variable 'name'	PATH←PATH..ENV{×≠⍺: ⍵ ⋄ ⍺}'.:..'
Directive	If clause ::IF code	::IF 0≠≠DEBUG..ENV
Directive	Test that name is defined	::IFDEF DEBUG_FLAG
Directive	Test that name is not defined	::IFNDEF DEBUG_FLAG
Directive	Undefine name	::UNDEF DEBUG_FLAG
Directive	Else-if clause ::ELSEIF/ELIF code	::ELSEIF DEBUG_FLAG≥3

ΔFIX Preprocessor Cheat Sheet

[ΔFIX Cheat Sheet.md]

Cat.	Item	Example
Directive	Terminate ::IF, ::IFDEF or ::IFNDEF sequence	::END, ::ENDIF, ::ENDIFDEF, ::ENDIFNDEF
Directive	Conditional with single variable name	::COND DEBUG ⍞←CUR_RESULT
Directive	Conditional with arbitrary parenthetical expression	::COND (DEBUG≥3) ⍞←CUR_RESULT
Directive	Preprocessor messages ::MESSAGE/MSG text	::MSG DEBUGGER CODE ACTIVATED!
Directive	Preprocessor error msgs ::ERROR [num] string	::IF CONFLICTING_OPTIONS ::ERROR 911 Conflicting Options Detected!
Directive	Include a file unconditionally	::INCLUDE MyLocalData.dat
Directive	Include a file if not already included earlier	::CINCLUDE printServices.dyalog
Directive	Specify one of more sets of code lines to execute at compile time. Each line of code must refer only to global objects or objects previously defined in order. See also ⍞MY, ⍞FIRST.	::FIRST [any string] compile-time code ::ENDFIRST [matching any string]

Pre-defined macros	Return valid first letters of Dyalog APL variable names.
	⍞LET.ALPH
	⍞LET.UC
	⍞LET.LC
		⍞LET.ALPH: a string of all valid first letters of Dyalog APL variables.
		  Def: (⍞LET.UC,⍞LET.LC,'_ΔΔ')
		⍞LET.UC: only upper-case.
		⍞LET.LC: only lower-case.
		  Synonyms: ⍞LET.alph, ⍞LET.uc, ⍞LET.lc.

Pre-defined macros	Define STATIC objects, viz. those that persist between function calls. ⍞MY, ⍞MY.ΔFIRST	⍞MY specifies a (relative) private namespace specific to the function(s) named in the header, specifically those returned by 2∘⍞FIX, i.e. defined at the “top” level. [See ⍞FIRST for examples.]

ΔFIX Preprocessor Cheat Sheet

[ΔFIX Cheat Sheet.md]

Cat.	Item	Example
Pre-defined macros	Set function flag ⎕FIRST to allow for code to initialize STATIC objects. Define ⎕MY.ΔFIRST or ⎕FIRST, which returns 1 on its first invocation. See also ::FIRST, which can be used with ⎕MY to initialize objects at ΔFIX (compile) time.	∇TEST1 :IF ⎕FIRST ⋄ ⎕MY.count←0 ⋄ :ENDIF 'This fn has been called',… ⎕MY.count,'times.' ∇ To use with :WITH, ⎕MY.ΔFIRST should be used to ensure the right relative namespace: ∇TEST1 :With ⎕MY :IF ΔFIRST ⋄ count←0 ⋄:ENDIF 'This fn has been called',… count,'times.' :EndWith ∇ ⎕MY is a macro that points to the relative namespace ⍙⍙.ΔMY.myfn for function myfn; ⎕FIRST similarly points relatively, to ⍙⍙.ΔMY.myfn.ΔFIRST.
Pre-defined macros	Allow a function to be "reset," so that STATIC objects may be re-initialized. ⎕RESET or ⎕MY.ΔRESET will ensure the next call of ⎕FIRST will return 1 for a specific function.	∇{reset} TEST2 args :IF reset..DEF ⋄ :ANDIF reset ⎕RESET :ENDIF :IF ⎕FIRST ⎕←'et cetera' :ENDIF 'more' ∇