Standard Fx

Sign		Form	Description		Note				
E[0]	V[0]	Consist of Aa-Zz and _ but no beginning with _	An identifier		Define V[0]’s value is the global variable V[0]’s value				X[t] is E[t]’s value i=1,2,...,n, j=1,2,...,m Blank chars in E[0] except in "" or '' will be ignore
	A[0]	E[0].V[0]	The member variable V[0]’s value of X[0]		X[0]’s type is not beginning with _				
		E[0](E[1],E[2],...,E[m])	A value what X[0] returns after received X[1],X[2],...,X[m] in one time		X[0] received X[i] as NO.i value received	X[0]’s type is _func			
		Consist of 0-9 and at most one . and e or e- in it	A value of type _num		A number				
		_nan							
		_inf							
		Consist of chars in ""	A value of type _str		"" in "" means "				
		Consist of chars in ''	A value of type _err		'' in '' means '				
		{E[1],E[2],...,E[m]}	A value of type _list		A list that has m elements in it				
		{}			An empty list				
		_true	A value of type _bool		If X[0] is _true then what statement expressed by E[0] is true				
		_false			If X[0] is _false then what statement expressed by E[0] is false				
		Consist of Aa-Zz and begin with __	A value of type _func which defined inline		It will be used in the standard library’s code				
		(V[1],V[2],...,V[n])=>{E[1],E[-1]|E[2],E[-2]|...|E[m],E[-m]}	A value of type _func		A function received n values in one time then return a value Once X[-j] is _true,return X[j] If X[-j] is always _true then suggest write E[j] instead of E[j],E[-j]			Redefine V[i]’s value in E[j] or E[-j] is NO.i value received X[-j]’s type is _bool	
	(C[0])	(-E[0])	The opposite of X[0]				A value that its type is X[0]’s type	When before is (or { or , or | or : and after is not . then suggest write C[0] instead of (C[0])	
		(E[0]$E[1])	Apply X[1] to each element of X[0] in orders		X[1]’s type is _func				
		(E[0]<-E[1])	Fold X[0] by apply X[1] fold two elements in orders						
		(E[0]\E[1])	Filter all elements of X[0] by apply X[1] in orders						
		(E[0]^E[1])	X[0] to the power of X[1]			X[1]’s type is X[0]’s type			
		(E[0]*E[1])	Multiply X[0] by X[1]						
		(E[0]/E[1])	X[0] divided by X[1]						
		(E[0]+E[1])	X[0] plus X[1]						
		(E[0]-E[1])	X[0] subtract X[1]						
		(E[0]->E[1])	X[0] has sub sequence X[1]				A value that its type is _bool		
		(E[0]/->E[1])	X[0] hasn’t sub sequence X[1]						
		(E[0]<E[1])	X[0] less than X[1]						
		(E[0]>E[1])	X[0] greater than X[1]						
		(E[0]=E[1])	X[0] equal to X[1]						
		(E[0]<=E[1])	X[0] less than or equals to X[1]						
		(E[0]>=E[1])	X[0] greater than or equals to X[1]						
		(E[0]/=E[1])	X[0] not equal to X[1]						
		(E[0]/\E[1])	X[0] and X[1]		X[0]’s type is _bool				
		(E[0]\/E[1])	X[0] or X[1]						
L[0]		V[0]:E[0]	Define the global variable V[0]’s value is X[0]					Defined it once at most	
		-V[0]:E[1]	Define (-E[0])’s value is ?(V[0])=>{E[1]}(X[0])		X[0]’s type is V[0]				
		V[0]$V[1]:E[2]	Define (E[0]$E[1])’s value is	?(V[0],V[1])=>{E[2]}(X[0],X[1])					
		V[0]<-V[1]:E[2]	Define (E[0]<-E[1])’s value is						
		V[0]\V[1]:E[2]	Define (E[0]\E[1])’s value is						
		V[0]^V[1]:E[2]	Define (E[0]^E[1])’s value is						
		V[0]*V[1]:E[2]	Define (E[0]*E[1])’s value is						
		V[0]/V[1]:E[2]	Define (E[0]/E[1])’s value is						
		V[0]+V[1]:E[2]	Define (E[0]+E[1])’s value is						
		V[0]-V[1]:E[2]	Define (E[0]-E[1])’s value is						
		V[0]->V[1]:E[2]	Define (E[0]->E[1])’s value is						
		V[0]/->V[1]:E[2]	Define (E[0]/->E[1])’s value is						
		V[0]<V[1]:E[2]	Define (E[0]<E[1])’s value is						
		V[0]>V[1]:E[2]	Define (E[0]>E[1])’s value is						
		V[0]=V[1]:E[2]	Define (E[0]=E[1])’s value is						
		V[0]<=V[1]:E[2]	Define (E[0]<=E[1])’s value is						
		V[0]>=V[1]:E[2]	Define (E[0]>=E[1])’s value is						
		V[0]/=V[1]:E[2]	Define (E[0]/=E[1])’s value is						
		V[0](V[1],V[2],...,V[n])|E[0]	Define the global variable V[0] inline		V[0] received n values in one time Redefine V[i]’s value in E[0] is NO.i value received V[-1].V[i]’s value is NO.i value received V[-1].V[-2]’s value is 'Undefined the Member variable V[-2] of type V[0]' If X[0] is _true,return V[-1] else return 'Create type V[0]’s value error'	V[0]’s type is _func X[0]’s type is _bool V[-1]’s type is V[0]			
		$V[0]	Expand to file V[0]’s code at the first time and ignore after expand begun		File V[0] could be found only in one dir in the standard library dir or the project dir				
		Consist of chars in ##	A description of code		## in ## means #				
G[0]		L[1];L[2];...;L[k]	A code in file		k>1				
		L[1]							

P.S. I’m not good at English,so some mistake will include.