**Standard Fx**

**Fx is a language which design to use in some small occasion:**

**For example, you can print "Hello World" like this:**

**print("Hello World")**

**A function you write could insert in any right places and means the same function in the same application's source code, like this function:**

**?(x){x < 0:f(x),x = 0:g(x),x > 0:t(x)}**

**You can create an object like this:**

**pair{fst:0,snd:1}**

**Here is the definition of Fx:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sign** | | **Form** | **Description** | **Note** | | | |
| **#** | | **One of + or – or \* or / or ^ or < or > or = or /= or <= or >= or & or |** | **An operator** | **There is no priority in operators** | | | |
| **N** | | **Consist of any chars** | **A note** | **() in N must be paired** | | | |
| **Ti** | | **Consist of Aa-Zz or \_** | **A type** | **If Ti beginning with \_ then Ti is the standard type else Ti is the user type** | | | |
| **Ei** | **Vi** | **Consist of Aa-Zz or \_ but no beginning with \_** | **An identifier** | **Vi is an variable in default**  **The variable Vi refers the global variable Vi's value in default** | | | |
| **C** | **?(V1,V2,...,Vm){E-1:E1, E-2:E2,..., E-n:En}** | **A function** | **Note** | | | **Type** |
| **The variable V0 in E-j or Ej must refers the global variable V0's value(It is Law A)**  **The variable Vi in E-j or Ej must refers the NO.i value ?(V1,V2,...,Vm){E-1:E1, E-2:E2,..., E-n:En} received(Except disobey Law A)**  **As soon as E-j is \_true, it returns Ej 's value**  **If n=1 and E-1 always is \_true then E-1:E1 could write as E1**  **i=1,2,...,m,j=1,2,...,n,m>0,n>0** | | | **\_func** |
| **Consist of 0-9 and at most one . and e or e- in it** | **A number** | **Float number** | | | **\_num** |
| **\_nan** | **\_nan refers nan** | | |
| **\_inf** | **\_inf refers inf** | | |
| **Consist of chars in ""** | **A string** | **"" means "** | | | **\_str** |
| **Consist of chars in ''** | **An error message** | **'' means '** | | | **\_err** |
| **\_true** | **A bool** | **If E0's value is \_true then what statement expressed by E0 is true** | | | **\_bool** |
| **\_false** | **If E0's value is \_false then what statement expressed by E0 is false** | | |
| **\_illegal** | **If E0's value is \_illegal then what statement expressed by E0 is illegal** | | |
| **\_possible** | **If E0's value is \_possible then what statement expressed by E0 is possible** | | |
| **{}** | **A list** | **An empty list** | | | **\_list** |
| **{E1,E2,...,En}** | **A list that has n(n>0) members** | | |
| **V0{V1:E1,V2:E2,...,Vn:En}** | **An object or an error message** | **V0 is a type name**  **Vi is a member variable**  **T is a value which type is V0**  **The member variable V-1 of T refers 'undefined'**  **The member variable Vi of T refers Ei's value**  **is\_V0 is a global variable**  **If is\_V0(T)'s value is \_true then V0{V1:E1,V2:E2,...,Vn:En}'s value is T else V0{V1:E1,V2:E2,...,Vn:En}'s value is 'Create object error'**  **i=1,2,...,n,n>0** | | | |
| **E0(E1,E2,...,En)** | **A function call** | **E0 receives E1,E2,...,En in order and return a value as E0(E1,E2,...,En)'s value**  **i=1,2,...,n,n>0** | | | |
| **E1.V1** | **A member variable** | **Get the member variable V1's value of E1's value** | | | |
| **(-E1)** | **A calculation** | **If the char before (-E1) is ( or { or , or : and the char after (-E1) is not . or ( then (-E1) could write as -E1** | | | |
| **(E1#E2)** | **If the char before (E1#E2) is ( or { or , or : and the char after (E1#E2) is not . or ( then (E1#E2) could write as E1#E2** | | | |
| **Pi** | **Di** | **V0(N):E0** | **Define a global variable** | **The global variable V0's value is E0's value** | | **(N) could be omitted** | **Blank chars will be ignored except in "" or ''** |
| **V0(N):** | **The global variable V0's value is inexpressible by Fx** | |
| **-T1(N):E-1** | **Define a calculation** | **If E-1 exists then (-E1)'s value is E-1(E1)'s value**  **else if E-2 exists then (-E1)’s value is E-2(E1)’s value**  **else (-E1)’s value is '-T1 is undefined. '** | **E1's type is T1**  **E2's type is T2** |
| **-?(N):E-2** |
| **T1#T2(N):E-3** | **If E-3 exists then (E1#E2)'s value is E-3(E1,E2)'s value**  **else if E-4 exists and E-5 does not exist then (E1#E2)'s value is E-4(E1,E2)'s value**  **else if E-4 does not exist and E-5 exists then (E1#E2)'s value is E-5(E1,E2)'s value**  **else if E-4 exists and E-5 exists then (E1#E2)'s value is 'T1#T2 is undefined.'**  **else if E-6 exists then (E1#E2)'s value is E-6(E1,E2)'s value**  **else (E1#E2)'s value is 'T1#T2 is undefined.'** |
| **T1#?(N):E-4** |
| **?#T2(N):E-5** |
| **?#?(N):E-6** |
| **$V0** | **Import the file V0.fx** | **The file V0.fx is in the standard library** | **V0 is a file name** | |
| **@V0** | **The file V0.fx is under the same path as the file containing this @V0** |
| **M** | **D1;D2** | **Multiple definitions or import files** | **D1;D2 is the same as D2;D1** | | |

**The following table show us the Standard Library:**

|  |  |
| --- | --- |
| **File** | **Code** |
| **stdlib.fx** | **left:?(x,y){x};**  **right:?(x,y){y};**  **id:?(x){x};** |
| **stdcmp.fx** | **$stdbool.fx;**  **?/=?:?(x,y){-(x=y)};**  **?=?:?(x,y){x<=y&(y<=x)};**  **?<=?:?(x,y){-(x>y)};**  **?>?:?(x,y){y<x};**  **?>=?:?(x,y){-(x<y)};** |
| **stdio.fx** | **fgets:;**  **fputs:;**  **gets:fgets(“stdin”);**  **puts:fputs(“stdout”);** |
| **stdbool.fx** | **-\_bool:?(x){ x=\_true:\_false,**  **x=\_false:\_false,**  **x=\_illegal:\_possible,**  **x=\_possible:\_illegal};**  **\_bool&\_bool:?(x,y){ x=\_illegal:\_illegal,**  **x=\_false:?(y){y=\_illegal:\_illegal,y/=\_illegal:\_false}(y),**  **x=\_possible:?(y){y=\_true:\_posible,y/=\_true:y}(y),**  **x=\_true:y};**  **\_bool|\_bool:?(x,y){ x=\_illegal:\_illegal,**  **x=\_true:?(y){y=\_illegal:\_illegal,y/=\_illegal:\_true}(y),**  **x=\_possible:?(y){y=\_false:\_possible,y/=\_false:y}(y);**  **x=\_false:y};**  **eq\_bool:;**  **not\_eq\_bool:;**  **\_bool=\_bool:eq\_bool;**  **\_bool/=\_bool:not\_eq\_bool;** |
| **stdnum.fx** |  |
| **stderr.fx** | **$stdlib.fx**  **bind\_err(Bind two err msg into one):?;**  **\_err+\_err:bind\_err;**  **\_err+?:left;**  **?+\_err:right;**  **\_err-\_err:bind\_err;**  **\_err-?:left;**  **?-\_err:right;**  **\_err\*\_err:bind\_err;**  **\_err\*?:left;**  **?\*\_err:right;**  **\_err/\_err:bind\_err;**  **\_err/?:left;**  **?/\_err:right;**  **\_err^\_err:bind\_err;**  **\_err^?:left;**  **?^\_err:right;**  **\_err<\_err:bind\_err;**  **\_err<?:left;**  **?<\_err:right;**  **\_err>\_err:bind\_err;**  **\_err>?:left;**  **?>\_err:right;**  **\_err=\_err:bind\_err;**  **\_err=?:left;**  **?=\_err:right;**  **\_err/=\_err:bind\_err;**  **\_err/=?:left;**  **?/=\_err:right;**  **\_err<=\_err:bind\_err;**  **\_err<=?:left;**  **?<=\_err:right;**  **\_err>=\_err:bind\_err;**  **\_err>=?:left;**  **?>=\_err:right;**  **\_err&\_err:bind\_err;**  **\_err&?:left;**  **?&\_err:right;**  **\_err|\_err:bind\_err;**  **\_err|?:left;**  **?|\_err:right;**  **-err:id;** |