```
in lundi lang we have next abstractions:
 + constant (1, 2.0, "string", true, nothing)
 + datastruct (list - [] (list generate? ))
 + entity ({} - function, data struct)
 + recurse
 + not variable, only label and only one
 + monads. create binds for c plus plus (cpp,c++)
 + create modules

 closed:
 - threads (only for lang code). very cool control threads and
operation serial with structure like a neural network-

based code:

a is 6

label a with data 6

comment is only block:
/\ this is comments \/

b is 2.4
/\lundi lang haven't float/int/double -> all is number\/

c is false
/\ true or false <- it logic \/

d is "r"
/\ <- this is string of length 1, we haven't char -> all is text, but
you can get info from char\/
/\char is not abstraction\/

c is [1,2,"2",true]
/\list is abstract\/
c1 is c.1
/\
    get c[1] is 1
    first elem have index 1
```

```
\/

base operation with base type:
nothing:
    no operation
    but it call error
    //{}  =  nothing
Int:
    +_*/ = /= > < >= <= 5.s (string)
Char:
    "4".i - get code(Int)
Bool:
    = /= > < >= <= | & ^ true.s (string)
List:
    + ++l --l l++ l--

    [555,55] + [444] = [555,55,444]

    base operation
    list++a (add to end)
    list--  (del of start)
    a++list (add to start)
    --list (del of start)

    generate list

    list1 is n^N \x, l\{}

    /\(a_n)n^N is 2+5n\/

String:
    str.1 (char)
    .i (int) .b (bool)
    all list operation





Entity:
    add is /x,y/{x+y}
    Entity in create as curry
    add1 is add 1
    a is add1 7
    b is add 1 (add 3 (add 3 5))
    b1 is add1-add1-add1 1
        /x/{add1 (add1 (add1 x))}
        add1 <- add1 <- add1 <- x

    Obj is /color, w, h,_/{
        c is color
        ww is w
        hh is h
    }
    cop is Obj "#006787" 5 5
    /\cop == /_/{}
    but...
    \/
    a is (cop.w+cop.h).s+" "+cop.c
    /\
        function save arguments while its not all
    \/

Recurse:
    f_ is /p,n/{
        if n==0 {p}
        else{f_' (p*n) (n-1)}

        /\set '
        all var flush!!!\/
    }
    f is /n/{
        f_ 1 n
    }

monads. create binds for c plus plus
    new type -> Cpp_t
    it have void* and function for you
    strio is #load "libstdio.o"#
    strio.function_name "one string arguments"

    this function auto monad

    monads create from entity


create modules:
    #import "filename"#
    #name mystdio# using name for stop recurse and second include in
first row
```