Basic

The intuition of Data-type-Function Programming comes from math function, look, there's a function below

$$f(x)$$

it has two part $f()$ and x, actually in math $f()$ is function but in code language like C the function should be $f(x)$, because when we define a function we also have to claim the type of x, so the function can't be separated from the data type.

Now considering making a difference. First let review some math .

Starting with the most simple function +, and we will use sum to express it which will make it look like a function in your code. For the real number x, we can simply see sum is just

$$sum(x_1, x_2) = x_1 + x_2$$

this sounds stupid because it's very clearly, but let consider x to be a 2×1 matrix, so

$$sum(x_1, x_2) = \begin{pmatrix} a_1 \\ b_1 \end{pmatrix} + \begin{pmatrix} a_2 \\ b_2 \end{pmatrix} = \begin{pmatrix} a_1 + a_2 \\ b_1 + b_2 \end{pmatrix}$$

That looks like different, but ponder for a while you will find the structure of sum is not changed, that maybe hard to understand, but I just want to show you a abstract function can take different forms for different variables, or data type in programming.

So there's an addition for real number and another for matrix, but we all call it addition because there have the same structure, or in math we say the obey the same rule. And it explicit form is only determined by data type.

So now if we want to design a coding language similar to this, Using compare function for example which is different for real number and complex number, try

```
define compare(x_1,x_2)

struct real{
com
}
```

```
define sum(u1,u2) #here we define a abstract function

struct real
{
addition:
x_new = x1 +x2
}

struct matrix
{
addition:
a_new = a1 + a2
b_new = b1 + b2
```

```
}
#This is not a standard code, I only use it to express the different addition is
defined in different data type
```

Above we define the explicit addition operator for real & matrix in their data type, so when you use $sum(x_1, x_2)$ it will look for the addition operation in the data type, then use it to make a sum.

Now we can see the function is separated from the data type, sum only means sum, how to make a sum, you should refer to the addition in the data type. And the biggest difficulty I find is to explain what is the difference between it and object-oriented programming, and now I will give it a try. To see the difference, we need the function to be more complex.

```
define arrange(x1,x2,x3)
{
if x1>x2 then swith(x1,x2)
if x3>x2 then swith(x2,x3)
}
# This is a function will arrange it x1,x2,x3 according to its size
# Notice there is two
```

%TODO define the basic type for DFS