

The Reference Manual for \mathfrak{E}

Version 3.141

The \mathfrak{E} Development Team

March 10, 2016

Contents

1	Language Guide	1
1.1	types	1
1.2	constants	1
1.3	list	1
1.4	tuple	1
1.5	function	1
1.6	control flow	2
2	Usage	3
2.1	declare constants	3
2.2	declare function	3
2.3	pattern match	3
2.4	Input and Output	3
2.5	sample code	3

Chapter 1

Language Guide

§1.1 types

\mathbb{L} provides fundamental types, including `Int` for integers, `Double` for floating-point values, `String` for textual data, and `Bool` for Boolean value. \mathbb{L} also provides `Tuple` and `List` as described in §1.3, §1.4.

§1.2 constants

You cannot use variable. Alternatively, \mathbb{L} provides constant. The value of a constant cannot be changed once it is set in the future. This constraint ensures referential transparency.

§1.3 list

In \mathbb{L} , lists are a homogenous data structure. It stores several elements of the same type. List can be written in three different ways (Enumeration, Range, List comprehension). It should be noted that you don't have to declare the list with its type.

§1.4 tuple

\mathbb{L} provide tuple type. You can bound some values with tuple and handle as an unit value.

§1.5 function

In \mathbb{L} , you must define function to return just a value. Besides, you must declare types of each arguments and type of the return value. Using constants locally, You could declare them in `where` block below the function.

§1.6 control flow

Ⓔ provide only pattern-mach for control flow. Any conditional balancing you need is constructed with pattern-mach and several recursive call.

Chapter 2

Usage

§2.1 declare constants

2.1.1 declare constant

2.1.2 declare list

§2.2 declare function

Listing 2.1 function *fibonacci*

```
1 fibo(n) :: Int -> Int
2   = 0 [n == 0]
3   = 1 [n == 1]
4   = fibo(n-1) + fibo(n-2)
```

§2.3 pattern match

§2.4 Input and Output

§2.5 sample code

Here is the sample code, which returns a fibonacci number.

Listing 2.2 sample code

```
1 fibo(n) :: Int -> Int
2   = 0 [n == 0]
3   = 1 [n == 1]
4   = fibo(n-1) + fibo(n-2)
5 where
6   who :: Int
7   = 5
```

```
8  hoge :: Int
9  =6
10
11 m :: Double
12 =2
```