User Guide

Anna Bukowska, Marek Kaput

December 18, 2018

Contents

Part I

Introduction

Chapter 1

Hello world

Example 1.0.1: The simplest program

```
1  fun main() {
2    println("Hello World!");
3  }
```

Chapter 2

Result

```
1  data Value = ...
2  data Type = ...
3
4  newtype Term = (Value, Type)
5
6  data Result = Succ Term
7              | Fail Term
```

Example 2.0.1: Some results

```
1  succ "foo"  # succ String foo
2  succ 1.25   # succ Float 1.25
3  fail none   # fail None none
```

Chapter 3

Goal evaluation

Example 3.0.1: Comparation

```
1    2 < 3   # succ 3
2    3 < 2   # fail 2
```

Example 3.0.2: Simple usage of goal evaluation

```
1    if (2 < 3) {
2      # this code will be evaluated
3    }
4
5    if (3 < 2) {
6      # this code won't be evaluated
7    }
```

Type and value are not important, it is succ/fail which decide about the control flow.

Chapter 4

Control structures

Example 4.0.1: While loop

```
1    i = 0;
2    while(i < 10){
3      println(i);
4      i = i + 1;
5    }
```

Example 4.0.2: While infinity loop

```
1    while(){
2      # some code
3    }
4
5    # is the same as
6
7    while(succ none){
8      # some code
9    }
```

Example 4.0.3: If-condition

```
1      if(condition) {
2        # this code will be evaluated, if condition evaluate
       with succes
3      }
```

Example 4.0.4: If-else-condition

```
1  if(condition) {
2    # this code will be evaluated, if condition evaluate
     with succes
3  } else {
4    # this code will be evaluated, if condition evaluate
     with fail
5  }
```

Chapter 5

Math operations

Example 5.0.1: Math operators

```
1    1 + 2        # succ 3
2    1 - 2        # succ -1
3    1 * 2        # succ 2
4    1 / 2        # succ 0
5    1 / 2.0      # succ 0.5
6    1 / (2 - 3) # succ -1
```

Example 5.0.2: Relational operators

```
1    1 < 2        # succ 2
2    1 > 2        # fail 2
3    1 <= 2       # succ 2
4    1 >= 2       # fail 2
5    1 == 2       # fail 2
6    1 == 1.0     # succ 1.0 - authomatic conversion from Int
       to Float
7    1 =! 2       # succ 2
8    1 === 1      # succ 1
9    1 === 1.0    # fail 1.0
10   1 ==! 1      # fail 1
```

Example 5.0.3: Logical operators

```
1    a = succ "Ala"
2    b = fail 8
3
```

```
4    a or b              # succ 8
5    a or succ none      # succ none
6    a and b             # fail 8
7    succ none and a     # succ "Ala"
8    a xor b             # succ 8
9    a xor fail none     # succ none
10   not a               # fail "Ala"
```

Chapter 6

Functions

Example 6.0.1: Calling a function

```
1  fun main() {
2    a = int(scanln());
3    # function id + list of args in parens
4    println(a);
5  }
```

Example 6.0.2: Defining a function

```
1  # keyword fun + name + list of arguments in parens + body
     in brakets
2  fun Identity(s) {
3    s == "a" or s == "b" or s == "c"
4  }
```

Example 6.0.3: Calling arbitrary functions

```
1  fun f(x) { x * 2; }
2  fun g() { f; }
3
4  y = g()(5); # works same as y = f(5)
5  y == 10;
```

Chapter 7

Types

Example 7.0.1: None

```
a = succ none
b = fail none
```

Example 7.0.2: Integer

```
0
1
10
10_0
10_
10___
0b111000
0B111000
0b111_000
0b111_000_
0o01234567
0O01234567
0o0123_4567__
0x0123456789abcdefABCDEF
0X0123456789abcdefABCDEF
0x0123456789_abcdef_ABCDEF_
```

Example 7.0.3: Float

```
1.0
1.0e92
```

```
3    10e_6
4    1_________.0_______
5    1_.0
6    10_e5
7    35e+5
8    7e-5
9    8E5
```

Example 7.0.4: String

```
1    "Quick brown fox jumps over the lazy dog"
2
3    "Quick brown fox
4    jumps over
5    the lazy dog"
6
7    "C:\\"
8
9    "\""
```

Example 7.0.5: Raw String

```
1    "foo"       == r"foo"             # output: foo
2    "\"foo\""   == r#""foo""#         # output: "foo"
3    "x #\"# y" == r##"x #"# y"##     # output: x #"# y
```

Chapter 8

Operations on string

Example 8.0.1: Trim and unindent

```
t"   trim   " == "trim"

u"  un
  in
  dent"
==
"un
in
dent"
```

Example 8.0.2: Functions frrom std library

```
find("oo", "fooooo")  # succ 1
find("ofo", "fooooo") # fail none
last("foo")           # succ "o"
tail("foo")           # succ "oo"
head("foo")           # succ "f"
single("foo")         # fail 3
empty("")             # succ 0
cut(1, 3, "fooooo")   # succ "00"
len("foo")            # succ 3
```

Chapter 9

Expressions

Example 9.0.1: Types of expression

```
# id
a;
foo;

# literal
"foo"
12_345_678

# block
{ a = 123; b == a}

# operator
5 + b;
9 >= a;

# call
add(3, 4);
foo();

# loop
while(i < 4){
  println(i);
  i = i + 1;
}

# conditional
if (a = 3; c = b; a < b) {
  c = a;
} else {
```

```
30      println("c is equal to b which value is " + str(b));
31    };
32
33    # return
34    return a;
```

Example 9.0.2: Combine them

```
1    a and {3 > 5; 6 < 7} or "true" and not false()
2    b = if(a){ while(a){a = not a;};}
```

Chapter 10

Assignments

```
1  # the right site of assignmenet can be all types of
     expressions
2  # type and value are from the last used result
3  a = {94, 4, 7, 8}                              # succ 8
4  b = if(succ none){ b = 42; "foo"}              # succ "foo"
5  c = succ 1 and fail 2 and succ 3               # fail 2
6  d = fail 1 and fail 2 and succ 3 and fail 4 # succ 3
```

Chapter 11

Module

Example 11.0.1: Import declarations

```
1  import io
2  import math:sin
3  import math as m
4
5  fun main() {
6    f = io:open("result.txt", "w");
7    io:writeln(f, sin(m:pi));
8  }
```