User Guide

Anna Bukowska, Marek Kaput

December 20, 2018

Contents

Part I

Introduction

Chapter 1

Hello world

In Intentio each program starts with main function.

Example 1.0.1: The simplest program

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

Chapter 2

Result

Programs consist of assignments and expressions. Each evaluated expression returns a *result*.

The idea of result:

```
1   data Value = ...
2   data Type = None | Int | Float | String | Regex
3
4   newtype Term = (Value, Type)
5
6   data Result = Succ Term
7               | Fail Term
```

The difference between *result* in Intentio and most of other languages is containing a *succ/fail* prefix.

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

The building of *result* allows for using *goal evaluation*.

Example 3.0.1: Some expressions results

```
2 < 3                  # succ 3
3 < 2                  # fail 2
find("a", foo_string) # succ _nr_ | fail none
```

The main idea of *goal evaluation* is to use *succ/fail* to control flow in program.

Example 3.0.2: Simple usage of goal evaluation

```
if (2 < 3) {
  # this code will be evaluated
}

if (3 < 2) {
  # this code won't be evaluated
}

while(find("a", foo_string)){
  pos = find("a", foo_string;
  a = cut(0,pos, foo_string);
  # this code will be evaluated
  # as long as foo_string contains an "a"
}
```

The type and the value are not important. It allows to use complicated conditions, combined with basic expresssions.

Example 3.0.3: The combined condition

```
1    if (
2           -30 < a < 30
3           and if(a < 0){
4                is_prime_num(- a)
5             }else{
6                is_prime_num(a)
7             }
8         ) {
9       println(a);
10   }
```

Chapter 4

Control structures

Intentio provides two control structures:

- *while* loop
- *if* condition (whith optional *else* part)

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: Infinity while loop

```
1   while(){
2     println("infinity");
3   }
4
5   # is the same as
6
7   while(succ none){
8     println("infinity");
9   }
```

Example 4.0.3: If condition

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

Example 4.0.4: If-else condition

```
1   if(condition) {
2     # this code will be evaluated
3     # if condition evaluate with succes
4   } else {
5     # this code will be evaluated
6     # if condition evaluate with fail
7   }
```

Chapter 5

Operations

Intentio provides three types of oparators:

- simple math operators
- relational operators
- logical operators

Example 5.0.1: Simple 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
```

Example 5.0.2: Usage of math operators

```
1    (2 + 7) * 3 - 8 / 4 # succ 25
```

Example 5.0.3: 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    # (below) authomatic conversion from Int to Float
7    1 == 1.0       # succ 1.0
8    1 =! 2         # succ 2
9    1 === 1        # succ 1
10   1 === 1.0      # fail 1.0
11   1 ==! 1        # fail 1
```

Example 5.0.4: Usage of relational operators

```
1    a = 3;
2    b = 7;
3    1 < a < b < 20 # succ 20
```

Example 5.0.5: 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

To call a function it is needed to use sequence:
function id + list of arguments in parens (separated by comas).

Example 6.0.1: Calling a function

```
1   fun main() {
2     a = int(scanln());
3     println(a);
4   }
```

To define a function it is needed to use sequence:
keyword fun + name + list of arguments in parens (separated by comas) + body in brakets.

Example 6.0.2: Defining a function

```
1   fun Identity(s) {
2     s == "a" or s == "b" or s == "c"
3   }
```

Example 6.0.3: Some syntactic sugar

```
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

Intentio provides five types:

- None
- Integer
- Float
- String
- Regex

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

Example 7.0.4: String

```
"Quick brown fox jumps over the lazy dog"

"Quick brown fox
jumps over
the lazy dog"

"C:\\"

"\""
```

Example 7.0.5: Raw String

```
"foo"       == r"foo"             # output: foo
"\"foo\""   == r#""foo""#         # output: "foo"
"x #\"# y"  == r##"x #"# y"##     # output: x #"# y
```

Example 7.0.6: Regex

```
  # dependent on implementation
  "[abc]"      == "a" or "b"  or "c"

```

Chapter 8

Operations on string

Intentio is text processing oriented language. It means there are mechanisms and functions which help to deal with String data. Some of them are presented below.

Example 8.0.1: Trim and unindent

```
1  t"   trim   " == "trim"
2
3  u"  un
4    in
5    dent"
6  ==
7  "un
8  in
9  dent"
```

Example 8.0.2: Functions from standard library

```
1  find("oo", "fooooo")  # succ 1
2  find("ofo", "fooooo") # fail none
3  last("foo")           # succ "o"
4  tail("foo")           # succ "oo"
5  head("foo")           # succ "f"
6  single("foo")         # fail 3
7  empty("")             # succ 0
8  cut(1, 3, "fooooo")   # succ "00"
9  len("foo")            # succ 3
```

Chapter 9

Expressions

Intentio provides eight types of expressions:

- id

```
1    a;
2    foo;
```

- literal

```
1    "foo"
2    12_345_678
```

- block

```
1    { a = 123; b == a}
```

- operator

```
1    5 + b;
2    9 >= a;
```

- call

```
1    add(3, 4);
2    foo();
```

- loop

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

- conditional

```
1    if (a = 3; c = b; a < b) {
2      c = a;
3    } else {
4      println("c is equal to b which value is " + str(b))
       ;
5    };
```

- return

```
1    return a;
```

Example 9.0.1: Combined

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

Chapter 10

Assignments

The most important things about assignmenets are:

- the right site can be all types of expressions
- the result of assignment is the result of last evaluated expression

```
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

Intentio provides five types of import:

- qualified import

```
1    import a
```

- renamed qualified import

```
1    import a as b
```

- item import

```
1    import a:x
```

- renamed item import

```
1    import a:x as b
```

- import all

```
1    import a:*
```

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  }
```

Example 11.0.2: Re-exporting

```
1  # File: myprelude.ieo
2
3  export (open, writeln, sin, pi)
4
5  import io:open
6  import io:writeln
7  import math:sin
8  import math:pi
9
```