Solace Running Documentation

Noé Garcia

March, 2023

1 Introduction

This document is a running documentation for the Solace language. This documentation contains information regarding value types, and general syntax along with simple examples.

2 Types

Solace is a statically typed language. All values are also immutable. The following table contains all available types in Solace:

Solace Type	Description
int	general integer
float	general floating point number
char	general character
string	general string value
bool	general boolean value
:sym	symbol value
func	function type

2.1 Integers and Floats

Built in number values are represented within Solace as integers or floats. Much like other languages, integers represent whole values, and floats represent floating point real values. integer values can be defined in the following manner:

```
module:Main

func main int()
{
        // Declare a to be an integer and b to be a float
        a int = 1;
        b float = 0.1;

        // values of the same type can be defined with the use of commas:
        c, d int = 2, 3;

        0;
}
```

There are a number of arithmetic and comparison operations that are available for both integers and floats. Like other languages, number values are able to be added, subtracted, multiplied, and divided. Numbers are also able to be compared between one another. The following table contains operators for number types in Solace:

Operators	Description
+	addition
−	subtraction
*	multiplication
/	division
=	value assignment
==	equal to
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
!=	not equal to

Arithmetic and comparison operators are performed between two given values. The following is an example of each operator usage:

module : Main

```
fun main int ()
{
        // assume a and b are integers:
        a, b int = 5, 10;

        // arithmetic can be performed
        addition int = a + b;
        subtraction int = b - a;
        division int = a / b;
        multiplication int = a * b;

        // numbers can also be compared:
        a > b;  // false
        a < b;  // true
        a >= b; // false
        a <= b; // true
        a == b; // false
        a != b; // true

        // The above can also be done with float type variables, or between integers
        // and float variables.

        0;
}
```

2.2 Strings and Characters

Solace has two different types for handling raw text values: characters and strings. Characters represent a single character value, while strings represent a longer collection of character values in succession. Character variables and values can be compared between one another with the equality (==) operator. Strings can also be compared to one another in a similar way. Character values can be concatenated together to result in a new string value containing both character values. Strings can be concatenated together to result in a new string value containing both original strings.

module : Main

```
func main int ()
{
        // define character and string variables
        a char = 'a';
        b char = 'b';
        c string = "hello,";
        d string = " world!";

        // comparisons between character values
        a == a; // true
        a == b; // false
        // comparisons between string values
        c == c; // true
        c == d; //false
```

```
        // concatenate values together
        ab string = a | b; // "ab"
        cd string = c | d; // "hello, world!"

        0;
}
```

2.3 Boleans and Symbols

Boolean values can represent two different states: true and false. boolean values are able to be compared through the use of comparison operations. Symbols are a unique type as they represent defined names rather than raw values. There are two predefined symbols: $:ok$ and $:err$.

module:Main

```
func main int ()
{
        // function call will return two values: a symbol and the original integer.
        // the following will return (:ok, 4)

        evenResult :sym, val int = isEven(4);

        // function call will return a single boolean value: true or false.
        // the following will return false

        oddResult = isOdd(6);

        0;
}

func isEven (:sym, int) (n int)
{
        { (n%2 != 0) => (:err, n);}
        (:ok, n);
}

func isOdd bool (n int)
{
        { (n%2==0) -> false;}
        true;
}
```

Symbols are special values within Solace. Symbols aside from :ok and :err can be defined for furhter use. Symbols hold special value within programs as their values are directly tied to their name, and no other value. variables can contain symbol values. The following is a simple example of symbol definition and use.

module:Main

```
// define two new symbols
:sym :apple
:sym :orange

func main int ()
{
```

```
        // define a symbol variable
        apple :sym = :apple;
        orange :sym = :orange;

        // symbols can be compared.
        :orange == :apple; // false
        :apple == :apple;  // true

        // symbol variables can also be compared
        apple == orange;   // false
        orange != apple;   // true

        0;
}
```

2.4 functions

things

3 Complex Data Types

things

4 Controll Flow

things