An Introduction To Programming In Go

Fragoso Pérez, Jonathan

November 15, 2014

Abstract

Describing the basics of the Go programming language. This document is more like a resume, a quick reference to everyone that has programmed for some time and wants to introduce fastly to this language, without reading many links to the docs or a book. Some of the information/examples is directly copied from the book [1].

1. Introduction

Why Go?

2. Basic Types

2.1. Numbers

- Integer
  - uint8 → same as byte
  - uint32
  - uint64
  - int8
  - int16
  - int32 → same as rune
  - int64

  Machine dependent → their size is dependent on the type of architecture of the machine -> uint, int, uintptr

  Go allows to increment/decrement by a unit using the operator ++/−. The language also enables to increment/decrement using the operator +=/-= .
- Float
  - float32 → single precision
  - float64 → double precision
- Complex
  - complex64
  - complex128

2.2. Strings

Some operations:

- Length → len("Hello world")
- Char. accessing -> "Hello World"[1] → returns 101 instead of e as a character is represented as a byte.
- Concatenation → "Hello " + " world!"

2.3. Booleans

1 bit integer representing true or false.

Operations:

- &&
- ||
- !

3. Other Types

3.1. Arrays

```
var integerArray [10]int

x := [5]float64{ 2, 5, 3, 1}
```

3.2. Slices

Are like arrays, but their size is allowed to change.

```
//slice associated with an underlying
//float64 array of length 5
x := make([]float64, 5)

//slice associated with an underlying
//float64 array of length 5,
//where 10 is the capacity of the underlying
//array which the slice points to
x := make([]float64, 5, 10)

arr := []float64{1,2,3,4,5}
x := arr[0:4] // this will assign to x values [1,2,3,4] because the high
              // index is not included
```

Built-in functions:

- Append → creates a new slice by taking an existing one.

```
slice1 := []int{1,2,3}
slice2 := append(slice1, 4, 5)

// RESULT:
// slice 1 value is [1,2,3]
// slice 2 value is [1,2,3,4,5]
```

- Copy

```
slice1 := []int{1,2,3}
slice2 := make([]int, 2)
copy(slice2, slice1)

// RESULT:
// slice 2 now will have values [1,2] because slice2 has room for only two
   elements
```

Bibliography

[1] C. Doxsey. *An Introduction to Programming in Go:*. CreateSpace Independent Publishing Platform, 2012.