48. Constants

We can use constants for every type of value.

There are two types contant. Type constant and untype constant.

This is untyped constant

""""const (

a = 12

b = 13.56

c = "hello"

)""""

This is typed constant

""""const (

a int = 12

b float = 13.56

c string = "hello"

)""""

49. Iota

Iota is like number stage.

"""const(

a = iota

b= iota

c = iota

)"""

Ended

52. Hand exercise

Operations: ==, !=, <=, >=, <, >;

58. Loop - init, condition, post.

There is no while in Go.

for init; condition; post {

do something

}

59. Loop - nesting loops

Nesting loop is divided into two types. Outer loop and inner loop. i is outer loop, j is inner loop.

for i := 0; i <= 10; i++ {

for j := 0; j <= 3; j++ {

fmt.Printf("The inner loop : %d\t The outer loop : %d\n", j, i)

}

}

60. Loop - for statement

Using for statemen is might be as like this "for x > 10 {}"

for x < 10 {

var x = 1

fmt.Println("Me again")

x++

}

61. Loop - break, continue

Break and continue are keywords for using during the rendering loop.

Break is for breaking a loop.

Continue is for continuing

x := 1

for {

x++

if x > 100 {

break

}

if x%2 != 0 {

continue

}

fmt.Println(x)

}

62. Loop - printing ascii

for i := 33; i < 122; i++ {

fmt.Printf("%v\t%x\t%#U\n", i, i, i)

}

63. Conditional - if statement

if x := 3; x == 2 {

fmt.Println("Right")

}

64. Conditional - if, else, statement.

x := 3

if x == 2 {

fmt.Println("Right")

} else {

fmt.Println("False")

}

65. Switch

switch {

case (2 == 4):

fmt.Println("False")

case (3 == 3):

fmt.Println("True")

}

66. Conditional Operator

&& is and

|| is OR

79. Array

This is a Array example and we have to use this array as like this.

var x [5]int

fmt.Println(x)

x[3] = 45

fmt.Println(x)

80. Slice - composite literal

x := []int{1, 2, 3, 4, 5} this is Composite Literal

fmt.Println(x)

for finding lenght of array. we have to write code as like this len(a)

fmt.Println(len(x))

81. Slice - for range

Slice index

x := []int{1,2,3,4,5,6}

fmt.Println(x[0])

fmt.Println(x[1])

For ranging array we can use some code as like this

for i, v := range x {

fmt.Println(i, v)

}

82. Slice - sliceing a slice

x := []int{1, 2, 3, 4, 5, 6}

fmt.Println(x) this is for all array

fmt.Println(x[1]) this is for first index of array

fmt.Println(x[1:]) this is for index of array between 1 to last

fmt.Println(x[1:3]) this is for index of array between 1 to 3

fmt.Println(x[:4]) this is for index of array between 0 to 4

83. Slice - append to a slice

x := []int{1,2,3,4,5,6}

x = append(x , 12, 34, 23, 23)

append is worked for add new items and variables to array.

If we want to append two array we wil do that

x := []int{1,2,3}

y := []int{4,5,6}

y = append(x, y...)

84. Slice - deleting data in array

For deleting data from array we have to use append and slicing data.

x = append(x[:3], x[6:]...)

85. Slice - multi dimensional slice

xs := []string{"John", "Doe", "James"}

fmt.Println(xs)

ys := []string{"Jack", "Moth"}

fmt.Println(ys)

xy := [][]string{xs, ys}

fmt.Println(xy)