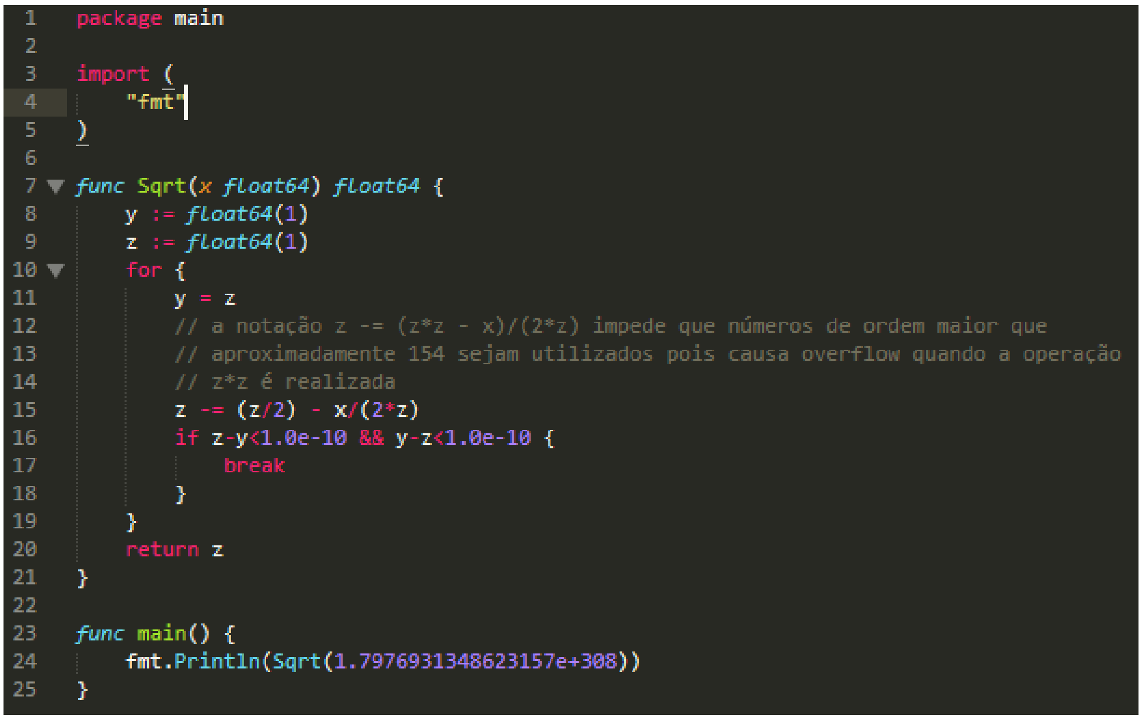
CES-27

TOUR

Professores: Juliana de Melo Bezerra
Vitor Curtis
Aluno: Dennys Leandro Agostini Rocha

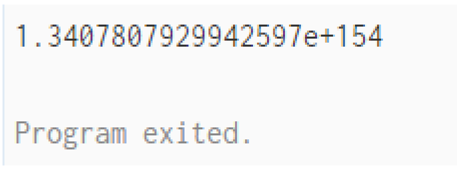
Objetivo
Aprender o básico do golang com alguns exercícios selecionados.

1)



```go
package main

import (
    "fmt"
)

func Sqrt(x float64) float64 {
    y := float64(1)
    z := float64(1)
    for {
        y = z
        // a notação z -= (z*z - x)/(2*z) impede que números de ordem maior que
        // aproximadamente 154 sejam utilizados pois causa overflow quando a operação
        // z*z é realizada
        z -= (z/2) - x/(2*z)
        if z-y<1.0e-10 && y-z<1.0e-10 {
            break
        }
    }
    return z
}

func main() {
    fmt.Println(Sqrt(1.7976931348623157e+308))
}
```

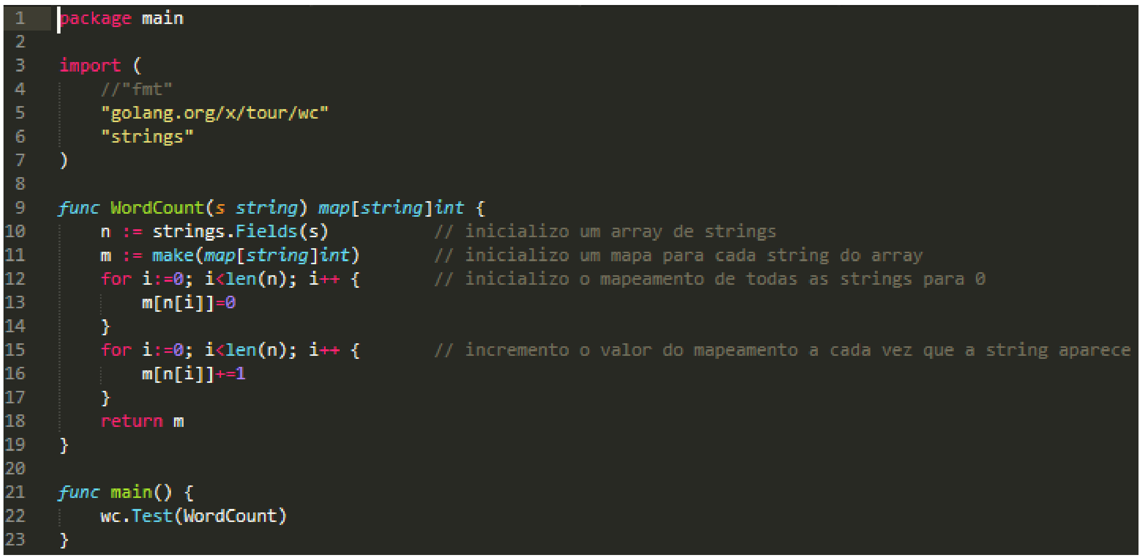
Rodando o código no golang.org:



```
1.3407807929942597e+154

Program exited.
```

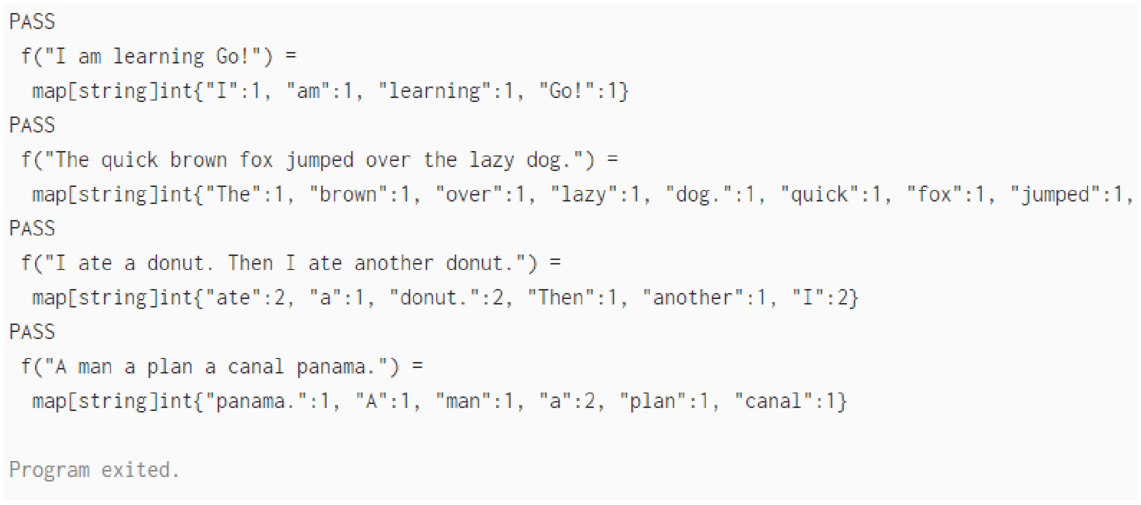
Pode-se testar qualquer valor entre 1 e 1.7976931348623157e+308, que é onde ocorre o overflow do float64.

2)



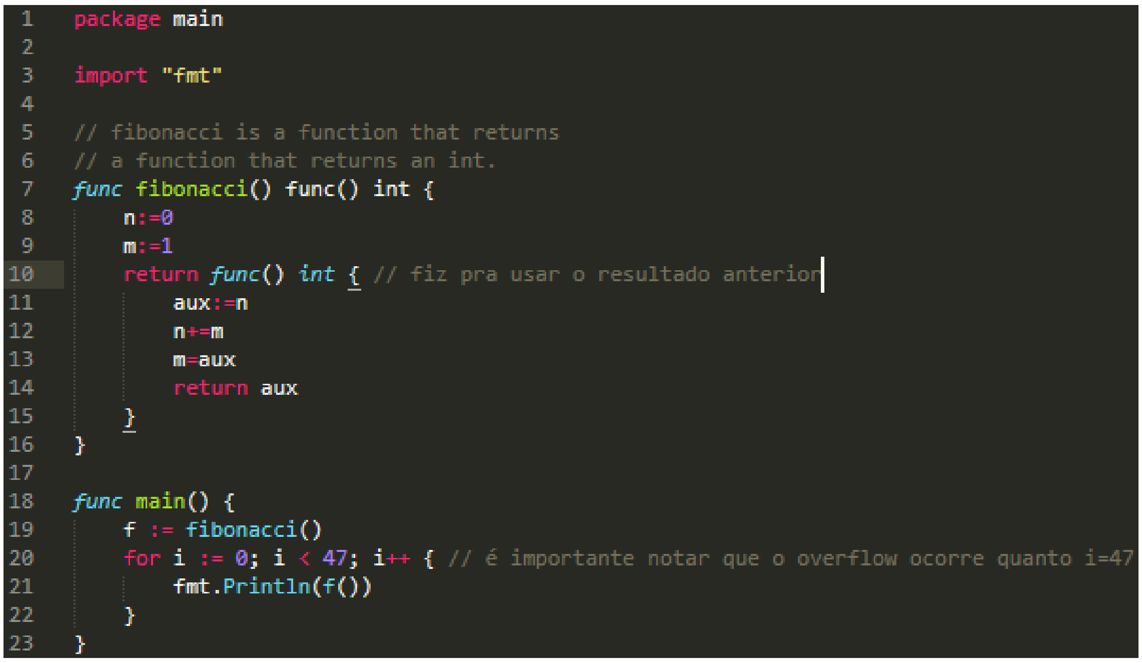
```go
package main

import (
    //"fmt"
    "golang.org/x/tour/wc"
    "strings"
)

func WordCount(s string) map[string]int {
    n := strings.Fields(s)          // inicializo um array de strings
    m := make(map[string]int)       // inicializo um mapa para cada string do array
    for i:=0; i<len(n); i++ {       // inicializo o mapeamento de todas as strings para 0
        m[n[i]]=0
    }
    for i:=0; i<len(n); i++ {       // incremento o valor do mapeamento a cada vez que a string aparece
        m[n[i]]+=1
    }
    return m
}

func main() {
    wc.Test(WordCount)
}
```

Rodando o código no golang.org:



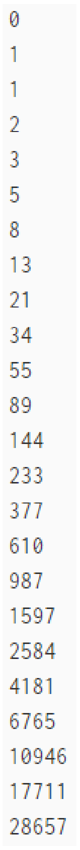
```
PASS
 f("I am learning Go!") =
  map[string]int{"I":1, "am":1, "learning":1, "Go!":1}
PASS
 f("The quick brown fox jumped over the lazy dog.") =
  map[string]int{"The":1, "brown":1, "over":1, "lazy":1, "dog.":1, "quick":1, "fox":1, "jumped":1,
PASS
 f("I ate a donut. Then I ate another donut.") =
  map[string]int{"ate":2, "a":1, "donut.":2, "Then":1, "another":1, "I":2}
PASS
 f("A man a plan a canal panama.") =
  map[string]int{"panama.":1, "A":1, "man":1, "a":2, "plan":1, "canal":1}

Program exited.
```

3)

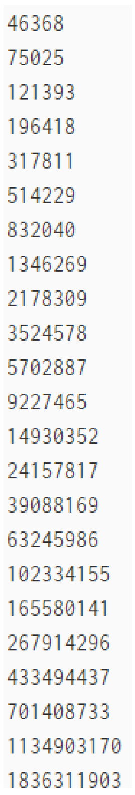


```go
package main

import "fmt"

// fibonacci is a function that returns
// a function that returns an int.
func fibonacci() func() int {
	n:=0
	m:=1
	return func() int { // fiz pra usar o resultado anterior
		aux:=n
		n+=m
		m=aux
		return aux
	}
}

func main() {
	f := fibonacci()
	for i := 0; i < 47; i++ { // é importante notar que o overflow ocorre quanto i=47
		fmt.Println(f())
	}
}
```

Rodando o código no golang.org:

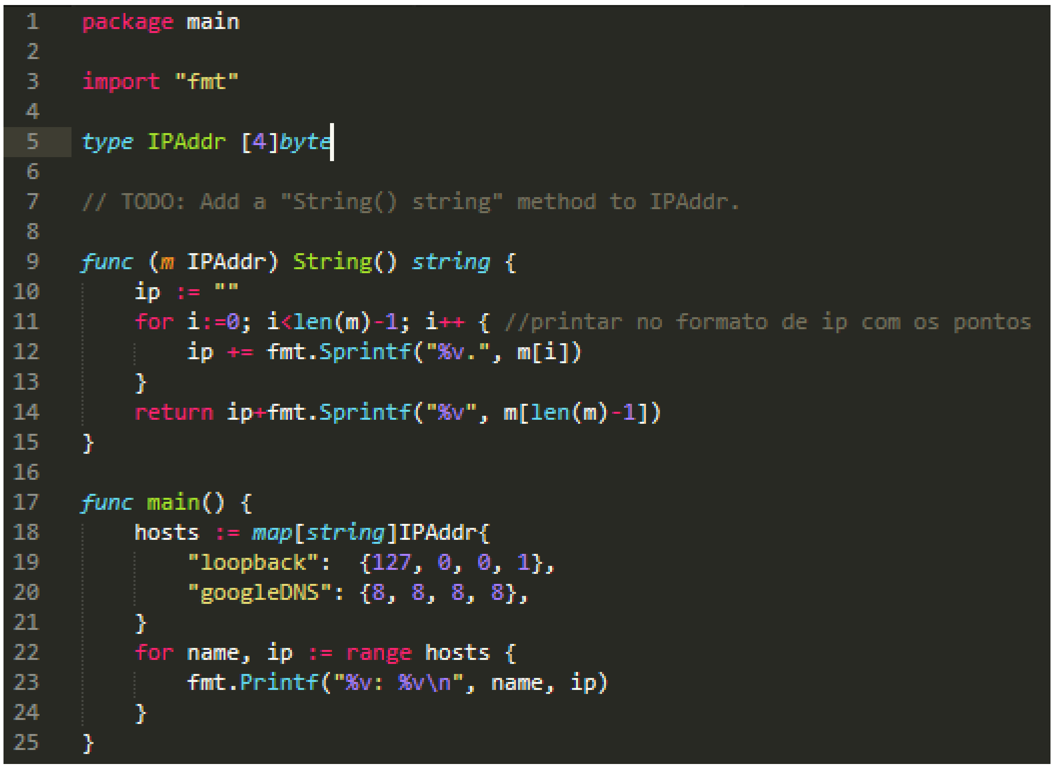


```
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
10946
17711
28657
```



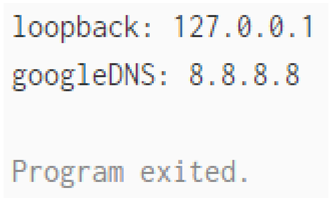
```
46368
75025
121393
196418
317811
514229
832040
1346269
2178309
3524578
5702887
9227465
14930352
24157817
39088169
63245986
102334155
165580141
267914296
433494437
701408733
1134903170
1836311903
```

4)



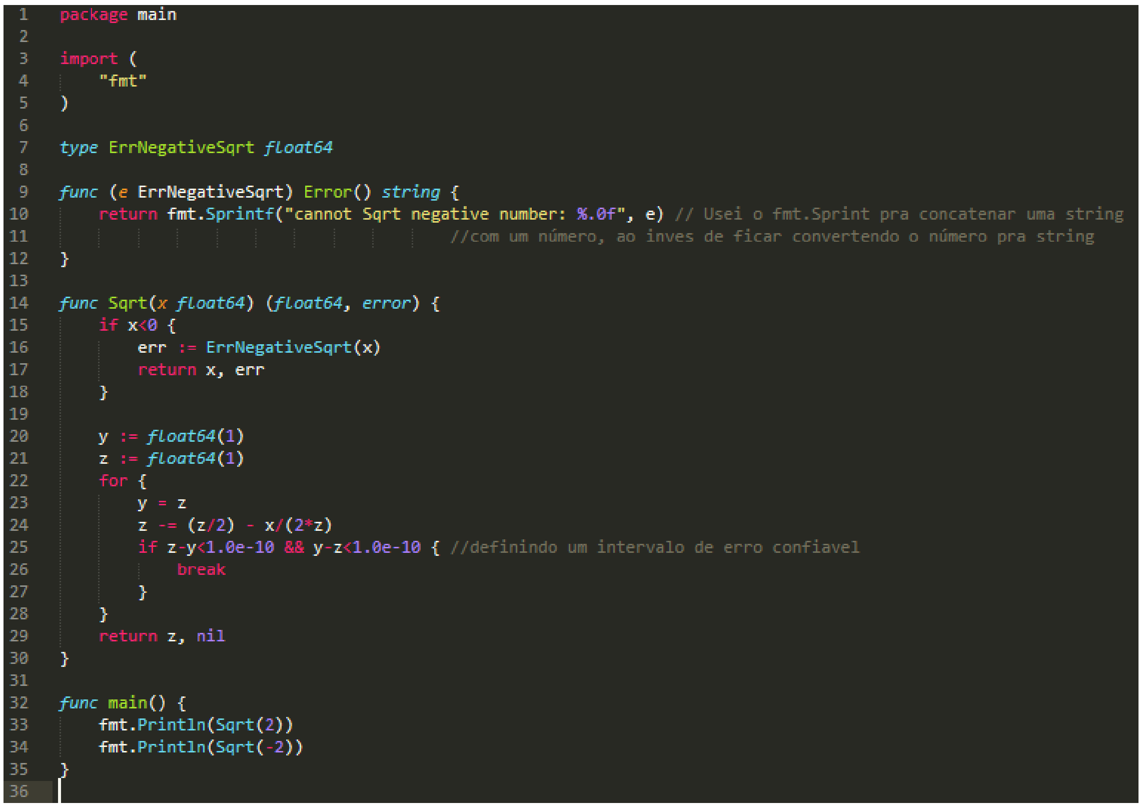
```go
package main

import "fmt"

type IPAddr [4]byte

// TODO: Add a "String() string" method to IPAddr.

func (m IPAddr) String() string {
	ip := ""
	for i:=0; i<len(m)-1; i++ { //printar no formato de ip com os pontos
		ip += fmt.Sprintf("%v.", m[i])
	}
	return ip+fmt.Sprintf("%v", m[len(m)-1])
}

func main() {
	hosts := map[string]IPAddr{
		"loopback":  {127, 0, 0, 1},
		"googleDNS": {8, 8, 8, 8},
	}
	for name, ip := range hosts {
		fmt.Printf("%v: %v\n", name, ip)
	}
}
```

Rodando o código no golang.org:



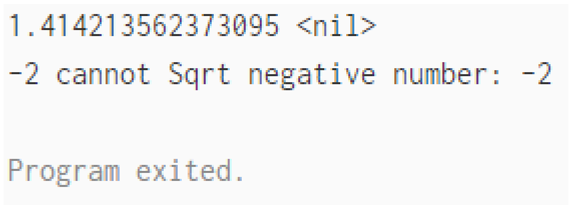
```
loopback: 127.0.0.1
googleDNS: 8.8.8.8

Program exited.
```

5)



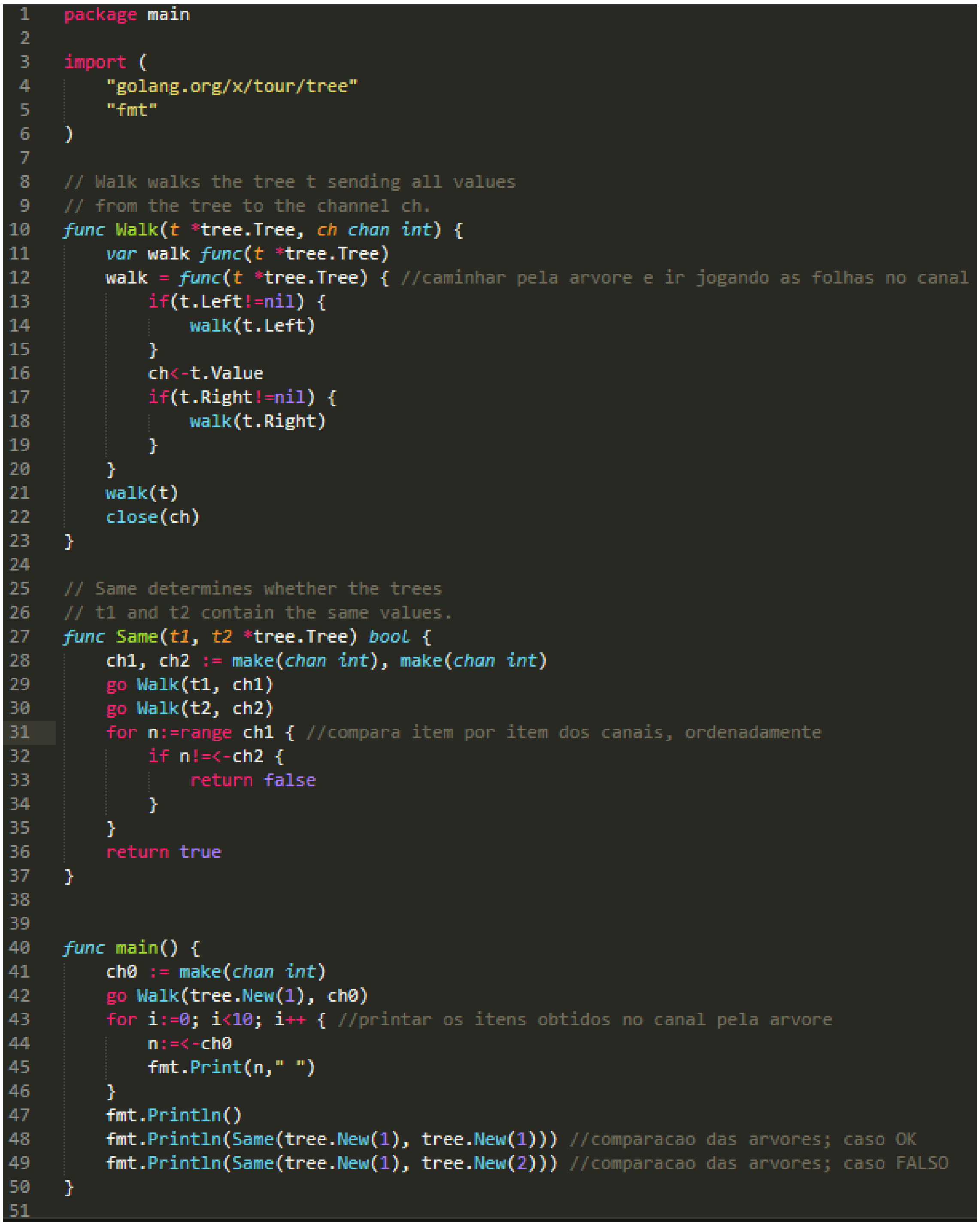
```go
package main

import (
	"fmt"
)

type ErrNegativeSqrt float64

func (e ErrNegativeSqrt) Error() string {
	return fmt.Sprintf("cannot Sqrt negative number: %.0f", e) // Usei o fmt.Sprint pra concatenar uma string
	                                                  //com um número, ao inves de ficar convertendo o número pra string
}

func Sqrt(x float64) (float64, error) {
	if x<0 {
		err := ErrNegativeSqrt(x)
		return x, err
	}

	y := float64(1)
	z := float64(1)
	for {
		y = z
		z -= (z/2) - x/(2*z)
		if z-y<1.0e-10 && y-z<1.0e-10 { //definindo um intervalo de erro confiavel
			break
		}
	}
	return z, nil
}

func main() {
	fmt.Println(Sqrt(2))
	fmt.Println(Sqrt(-2))
}
```

Rodando o código no golang.org:



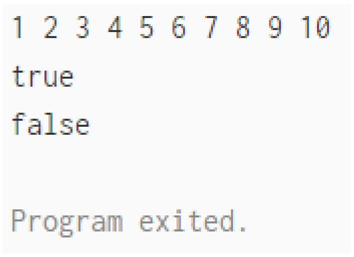
```
1.4142135623730951 <nil>
-2 cannot Sqrt negative number: -2


Program exited.
```

6)



```go
package main

import (
    "golang.org/x/tour/tree"
    "fmt"
)

// Walk walks the tree t sending all values
// from the tree to the channel ch.
func Walk(t *tree.Tree, ch chan int) {
    var walk func(t *tree.Tree)
    walk = func(t *tree.Tree) { //caminhar pela arvore e ir jogando as folhas no canal
        if(t.Left!=nil) {
            walk(t.Left)
        }
        ch<-t.Value
        if(t.Right!=nil) {
            walk(t.Right)
        }
    }
    walk(t)
    close(ch)
}

// Same determines whether the trees
// t1 and t2 contain the same values.
func Same(t1, t2 *tree.Tree) bool {
    ch1, ch2 := make(chan int), make(chan int)
    go Walk(t1, ch1)
    go Walk(t2, ch2)
    for n:=range ch1 { //compara item por item dos canais, ordenadamente
        if n!=<-ch2 {
            return false
        }
    }
    return true
}


func main() {
    ch0 := make(chan int)
    go Walk(tree.New(1), ch0)
    for i:=0; i<10; i++ { //printar os itens obtidos no canal pela arvore
        n:=<-ch0
        fmt.Print(n," ")
    }
    fmt.Println()
    fmt.Println(Same(tree.New(1), tree.New(1))) //comparacao das arvores; caso OK
    fmt.Println(Same(tree.New(1), tree.New(2))) //comparacao das arvores; caso FALSO
}

```

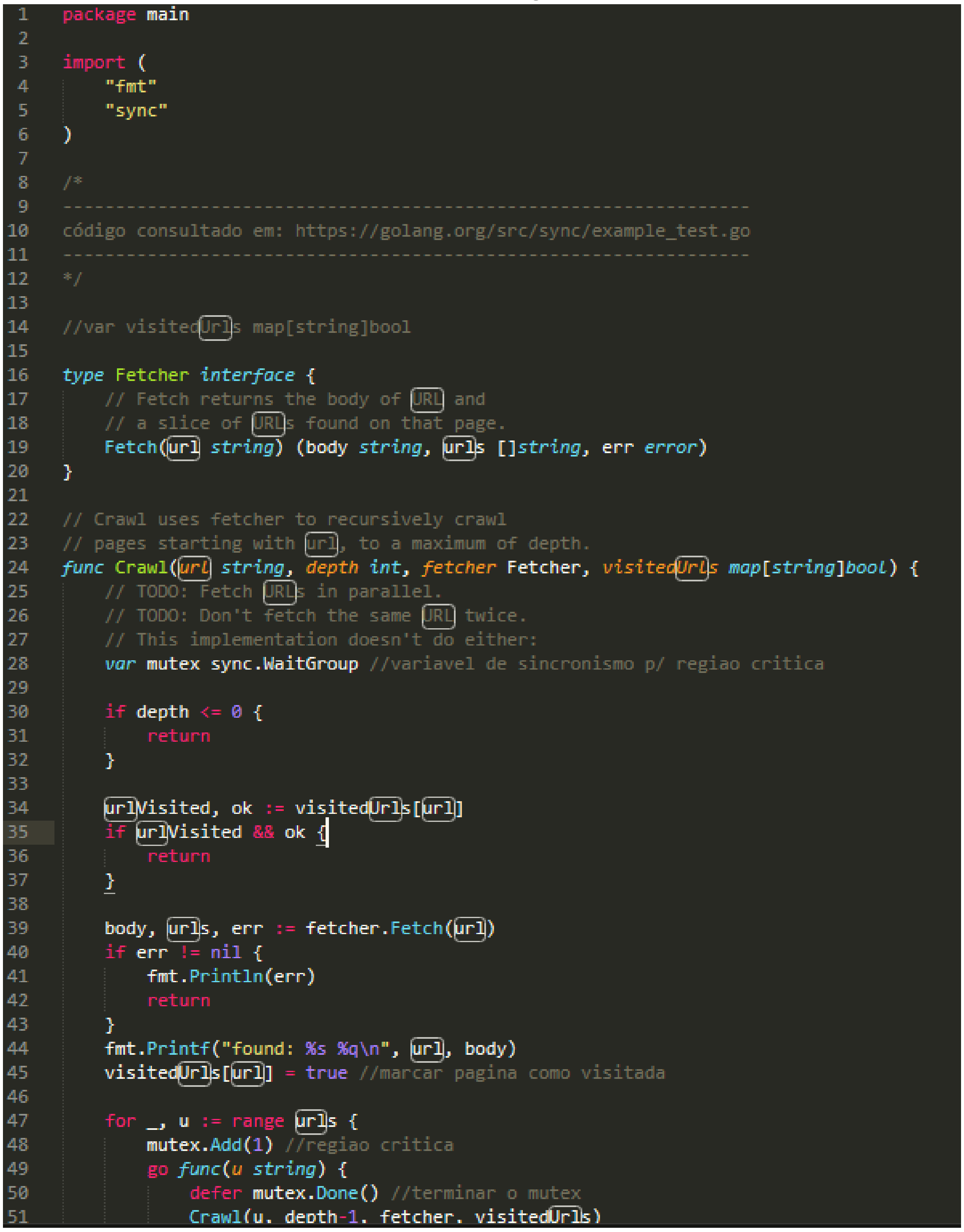
Rodando o código no golang.org:



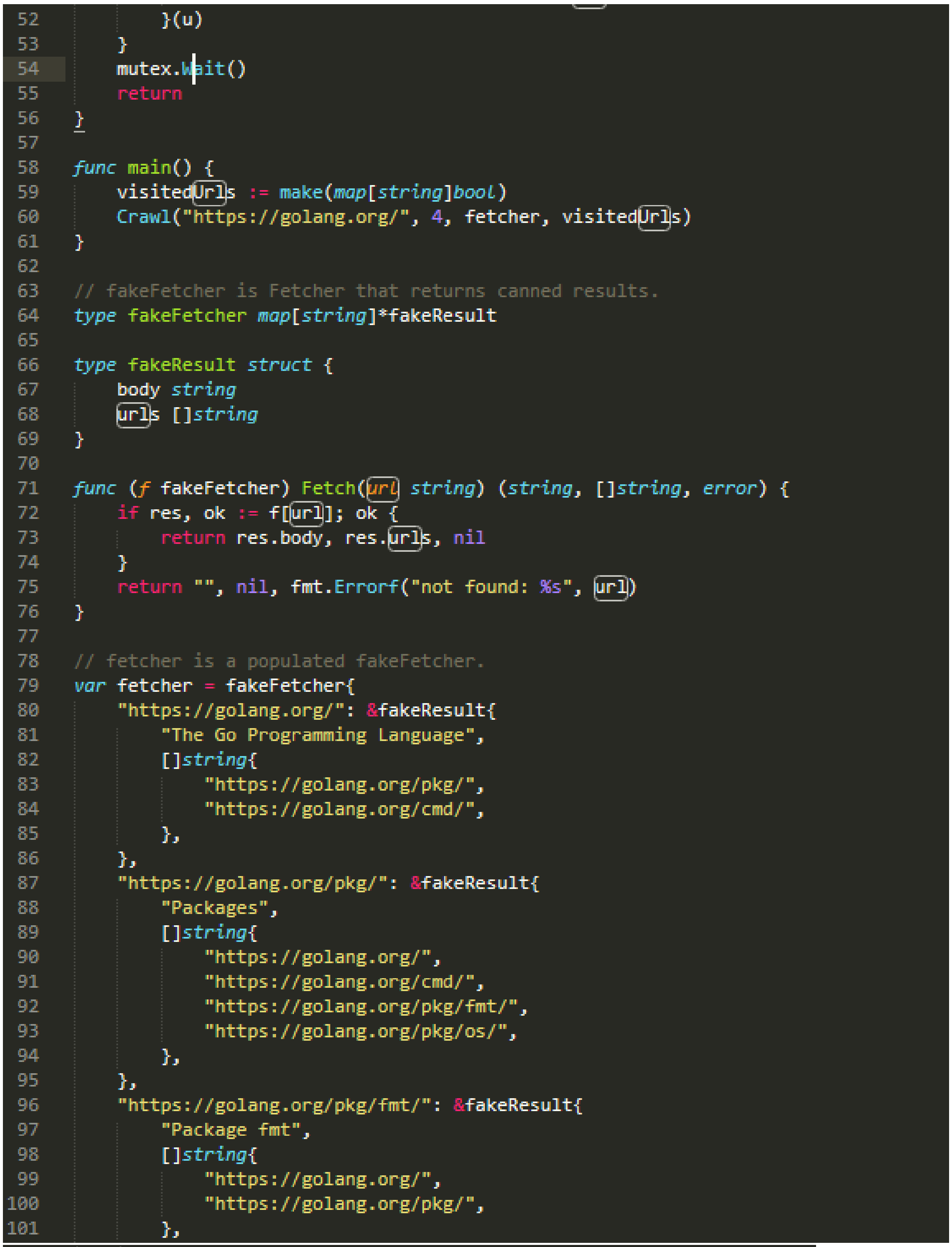
```
1 2 3 4 5 6 7 8 9 10
true
false

Program exited.
```

7)

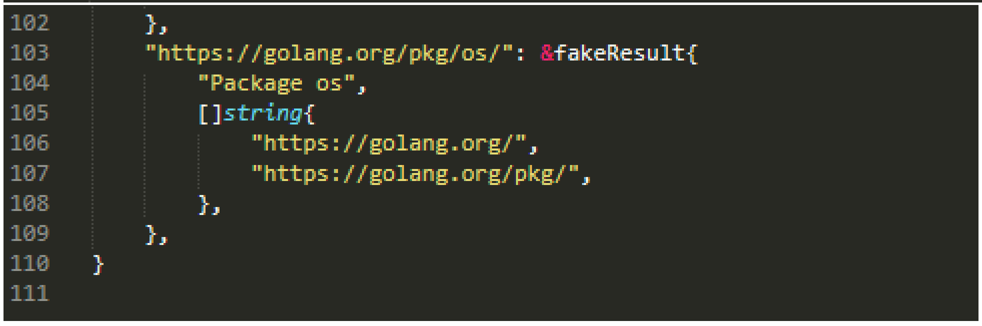
Está indicado a fonte de consulta na linha 10 do código.



```
1   package main
2
3   import (
4       "fmt"
5       "sync"
6   )
7
8   /*
9   -----------------------------------------------------------------
10  código consultado em: https://golang.org/src/sync/example_test.go
11  -----------------------------------------------------------------
12  */
13
14  //var visitedUrls map[string]bool
15
16  type Fetcher interface {
17      // Fetch returns the body of URL and
18      // a slice of URLs found on that page.
19      Fetch(url string) (body string, urls []string, err error)
20  }
21
22  // Crawl uses fetcher to recursively crawl
23  // pages starting with url, to a maximum of depth.
24  func Crawl(url string, depth int, fetcher Fetcher, visitedUrls map[string]bool) {
25      // TODO: Fetch URLs in parallel.
26      // TODO: Don't fetch the same URL twice.
27      // This implementation doesn't do either:
28      var mutex sync.WaitGroup //variavel de sincronismo p/ regiao critica
29
30      if depth <= 0 {
31          return
32      }
33
34      urlVisited, ok := visitedUrls[url]
35      if urlVisited && ok {
36          return
37      }
38
39      body, urls, err := fetcher.Fetch(url)
40      if err != nil {
41          fmt.Println(err)
42          return
43      }
44      fmt.Printf("found: %s %q\n", url, body)
45      visitedUrls[url] = true //marcar pagina como visitada
46
47      for _, u := range urls {
48          mutex.Add(1) //regiao critica
49          go func(u string) {
50              defer mutex.Done() //terminar o mutex
51              Crawl(u, depth-1, fetcher, visitedUrls)
```

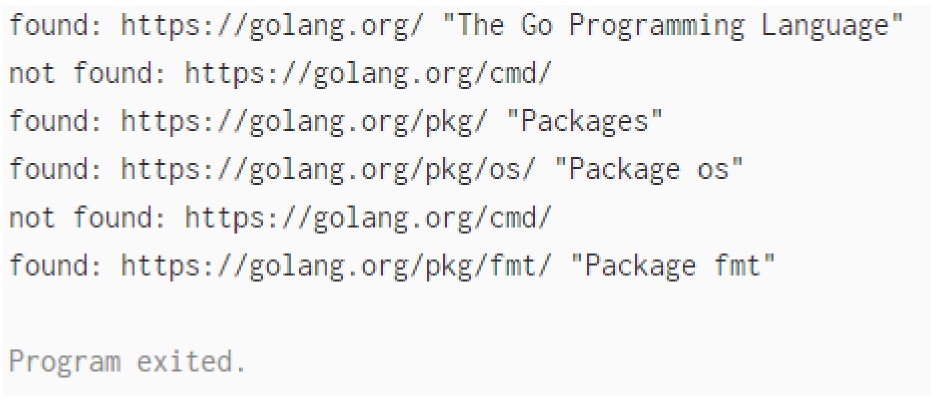


```go
        }(u)
    }
    mutex.Wait()
    return
}

func main() {
    visitedUrls := make(map[string]bool)
    Crawl("https://golang.org/", 4, fetcher, visitedUrls)
}

// fakeFetcher is Fetcher that returns canned results.
type fakeFetcher map[string]*fakeResult

type fakeResult struct {
    body string
    urls []string
}

func (f fakeFetcher) Fetch(url string) (string, []string, error) {
    if res, ok := f[url]; ok {
        return res.body, res.urls, nil
    }
    return "", nil, fmt.Errorf("not found: %s", url)
}

// fetcher is a populated fakeFetcher.
var fetcher = fakeFetcher{
    "https://golang.org/": &fakeResult{
        "The Go Programming Language",
        []string{
            "https://golang.org/pkg/",
            "https://golang.org/cmd/",
        },
    },
    "https://golang.org/pkg/": &fakeResult{
        "Packages",
        []string{
            "https://golang.org/",
            "https://golang.org/cmd/",
            "https://golang.org/pkg/fmt/",
            "https://golang.org/pkg/os/",
        },
    },
    "https://golang.org/pkg/fmt/": &fakeResult{
        "Package fmt",
        []string{
            "https://golang.org/",
            "https://golang.org/pkg/",
        },
```



```go
    },
    "https://golang.org/pkg/os/": &fakeResult{
        "Package os",
        []string{
            "https://golang.org/",
            "https://golang.org/pkg/",
        },
    },
}

```

Rodando o código no golang.org:



```
found: https://golang.org/ "The Go Programming Language"
not found: https://golang.org/cmd/
found: https://golang.org/pkg/ "Packages"
found: https://golang.org/pkg/os/ "Package os"
not found: https://golang.org/cmd/
found: https://golang.org/pkg/fmt/ "Package fmt"

Program exited.
```