Estruturas de Linguagem

Linguagem de programação Go

Alunos:
Eduardo Arthur
Leonardo Andrade

Professor:
Francisco Sant'Anna

Origens e Influências

- **Go** é uma linguagem de programação criada pela Google e lançada em 2009 com programa de código aberto
- Inspirada em trabalhos do sistema operacional inferno e baseada em C com várias otimizações
- Foi criada em 2007 para resolver problemas de performance que apareciam quando se construía algoritmos complexos em linguagens já existentes como C++, python ou java

Origens e influências

- Foi criada inicialmente com o objetivo de ser uma linguagem contendo as funcionalidades consideradas mais importantes nativamente
- Algumas características:
  - Programação concorrente e paralela nativa
  - Coletor de Lixo nativo
  - Gestão própria de memória e threads de forma transparente (evitando invasão de memória)
  - Simples e otimizada

Classificação

- Compilada
- Multiplataforma
- Paradigmas
- Concorrente
- Imperativa
- Estruturada
- Open Source
- Tipagem estática e inferência de tipos

Avaliação Comparativa

Go é uma linguagem que tenta combinar simplicidade e performance, assim juntando a facilidade de uma linguagem interpretada e dinamicamente tipada, um exemplo disso é o caso da inferência:

Go

```
var (
    name     string
    age      int
    location string
)
```

Sem inferência

```
var (
    name     = "Bruce Wayne"
    age      = 32
    location = "Gotham"
)
```

Com inferência

Figura 2 — Inferência de tipos

C++

```
string name;
int age;
string location;
```

```
string name = "Bruce Wayne";
int age = 32;
string location = "Gotham";
```

Avaliação Comparativa

- Goroutines
  - É uma thread mais “leve” administrada pelo runtime do Go
  - São executadas no mesmo endereço, dessa forma precisando estar sincronizadas para acessar a memória compartilhada
- Canais
  - É um “condutor” de informações que manda e recebe dados
  - Por padrão ele manda e recebe dados até que o outro lado esteja pronto, assim sincronizando as goroutines

Avaliação Comparativa

Goroutine

```go
package main

import (
        "fmt"
        "time"
)

func say(s string, done chan string) {
        for i := 0; i < 5; i++ {
                time.Sleep(100 * time.Millisecond)
                fmt.Println(s)
        }
        done <- "Terminei"

}

func main() {
        done := make(chan string)
        go say("world", done)
        fmt.Println(<-done)
}
```

```
world
world
world
world
world
Terminei

Program exited.
```

Thread

```python
from threading import Thread

class Th(Thread):

    def __init__ (self, num):
            Thread.__init__(self)
            self.num = num

    def run(self):

            print "Hello "
            print self.num


a = Th(1)
a.start()
```

```
>>> Hello
1
```

Defer

```go
func ex1(){
    i:=0
    defer fmt.Println(i)
    i++
    return
}
// Imprime 0 na tela
func ex2() {
    defer fmt.Print("\n")
    for i:=0;i<=10;i++ {
        defer fmt.Print(" ")
        defer fmt.Print(i)
    }
}
//Imprime 10 9 8 7 6 5 4 3 2 1
func ex3()(i int){
    defer func() { i++ }()
    return 1
}
// Retorna 2
```

Avaliação Comparativa

Go

```go
package main
import (
    "fmt"
    "os"
    "strconv"
)
func fib(n int ) int {
    if n==0 { return 0
    }else if n==1 {return 1
    }else  {return fib(n-1)+ fib(n-2)
    }
}
func main() {
    f, err := os.Create("text.txt")
    if err != nil {
        panic("cannot create file")
    }
    defer f.Close()
    for i:=0; i <= 40 ; i++ {
        fmt.Fprintln(f,strconv.Itoa(fib(i)))
    }
}
```

Lisp

```lisp

(defun fibonacci(n)
  (cond
    ((eq n 0) 0)
    ((eq n 1) 1)
    ( (+ (fibonacci (- n 1)) (fibonacci (- n 2))))))

(with-open-file (str "text0.txt"
                     :direction :output
                     :if-exists :supersede
                     :if-does-not-exist :create)
  (loop for a from 0 to 40
    do (format str (write-to-string(fibonacci a)))
    (format str "~%" )) )

```