



GO CURSO DE ((COLANG))

AUTOR: LEONARDO LEITÃO

Índice

1. Introdução	2
1.1. Visão Geral do Curso	2
1.2. Assine o Nossa Canal	2
1.3. Repositório do Curso	3
1.4. Autor	3
1.5. Suporte	4
2. Fundamentos	5
2.1. Primeiro Programa em Go	5
2.2. Explorando os Comandos do Terminal	5
2.3. Constantes e Variáveis	6
2.4. Imprimindo Valores no Console	7
2.5. Tipos Básicos	8
2.6. Tipos Básicos: Os Zeros	9
2.7. Conversão entre Tipos Básicos	10
2.8. Funções Básicas	11
2.9. Operadores Aritméticos	12
2.10. Operadores de Atribuição	13
2.11. Operadores Relacionais	14
2.12. Operadores Lógicos	15
2.13. Operadores Unários	16
2.14. Operador Ternário???.	17
2.15. Ponteiros em Go	17
3. Estruturas de Controle	18
3.1. If/Else	18
3.2. If/Else If	18
3.3. If com Init	19
3.4. Laço For	20
3.5. Switch #01	21
3.6. Switch #02	22
3.7. Resposta do Desafio Switch	23
3.8. Switch #03	24
4. Array/Slices/Map	26
4.1. Trabalhando com Arrays	26
4.2. Percorrendo Arrays com For (Range)	26
4.3. Conhecendo o Slice	27
4.4. Construindo Slices com Make	28
4.5. Array Interno	29
4.6. Slice: Usando Append e Copy	29

4.7. Trabalhando com Maps #01	30
4.8. Trabalhando com Maps #02	31
4.9. Maps Aninhados	32
5. Funções	34
5.1. Funções Básicas	34
5.2. Pilha de Funções	34
5.3. Retorno Nomeado	35
5.4. Armazenar Funções em Variáveis	36
5.5. Passar Função como Parâmetro	37
5.6. Funções Variáticas #01	37
5.7. Funções Variáticas #02	38
5.8. Closure	38
5.9. Recursividade	39
5.10. Recursividade (Simples)	40
5.11. Defer	41
5.12. Passando Ponteiro para Função	42
5.13. Função Init	42
6. Sistema de Tipos	44
6.1. Usando Struct	44
6.2. Struct Aninhada	44
6.3. Métodos em Structs	45
6.4. Pseudo-Herança em Structs	46
6.5. Tipo Personalizado	47
6.6. Usando Interfaces #01	48
6.7. Usando Interfaces #02	50
6.8. Composição de Interfaces	50
6.9. Tipo Interface	51
6.10. Convertendo uma Struct em JSON	51
7. Pacotes	53
7.1. Pacotes & Visibilidade	53
7.2. Criando um Pacote Reutilizável	54
7.3. Criando & Instalando um Pacote do Github	54
8. Concorrência	56
8.1. Cuidado com os Deadlocks	56
8.2. Channel com Buffer	56
8.3. Conhecendo a Goroutine	57
8.4. Conhecendo o Channel (Canal)	58
8.5. Usando Goroutine e Channel	59
8.6. Curiosidade: Número de CPUs	60
8.7. Padrão de Concorrência: Generators	61
8.8. Padrão de Concorrência: Multiplexador	62

8.9. Multiplexador com Select	63
8.10. Channel: Usando Range e Close	64
8.11. Estrutura de Controle: Select	65
9. Testes	67
9.1. Tipo de Arquitetura e os Testes	67
9.2. Teste Unitário Básico	67
9.3. Criando Dataset para os Testes	68
10. Banco de Dados	70
10.1. Criando o Schema e a Tabela	70
10.2. Executando Inserts	70
10.3. Executando Inserts em uma Transação	71
10.4. Executando Update e Delete	72
10.5. Executando Select e Mapeando p/ um Struct	73
11. Http	75
11.1. Criando um Servidor Estático	75
11.2. Gerando Conteúdo Dinâmico	75
11.3. Integrando Http e SQL (2 Serviços REST)	76
Appendix A: Tabela de Códigos	79
Glossário	80

Sumário

Apostila do curso de Go (Golang) da Cod3r.

<https://www.cod3r.com.br>

1. Introdução

1.1. Visão Geral do Curso

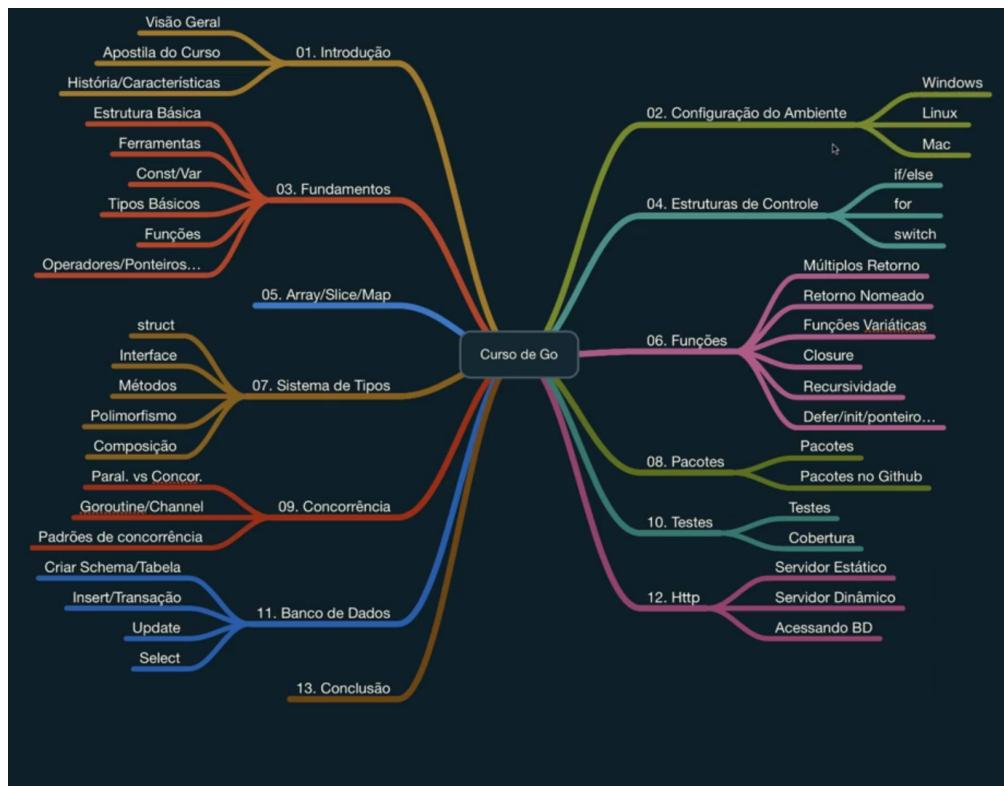


Figura 1. Visão Geral do Curso

1.2. Assine o Nossa Canal

Estamos também no **Youtube!** Temos um canal no youtube com várias séries de vídeos para complementar o seu aprendizado, então gostaria de te convidar para fazer parte da nossa comunidade em <https://www.youtube.com/aulasdeprogramacao>

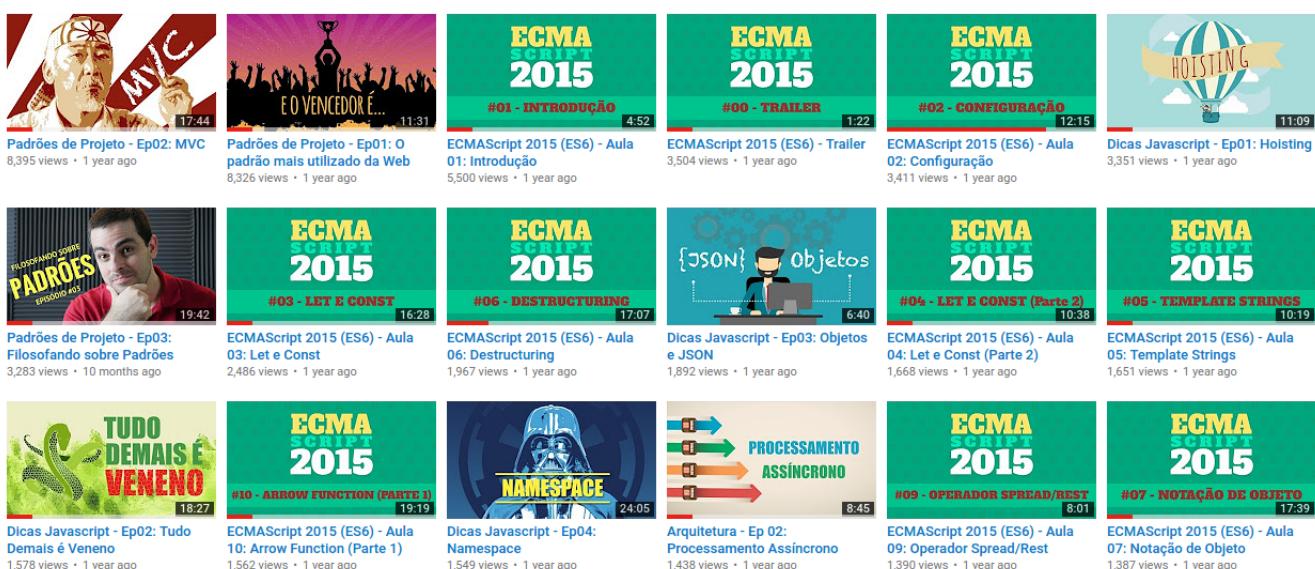


Figura 2. Algumas Aulas do Canal

1.3. Repositório do Curso

Precisa ter acesso ao **código fonte** dos exercícios do curso? Todos os exemplos desenvolvidos durante as aulas estão disponíveis no **Github** da Cod3r: <https://github.com/cod3rcursos/curso-kotlin>



Se você não nunca trabalhou com repositórios Git, uma excelente ferramenta para facilitar a obtenção do código pode ser obtida no seguinte endereço:

<https://desktop.github.com/>

É possível ter acesso ao código fonte do curso sem precisar instalar nenhuma ferramenta, baixando diretamente o arquivo **.ZIP** da página do **Github**: <https://github.com/cod3rcursos/curso-kotlin/archive/master.zip>

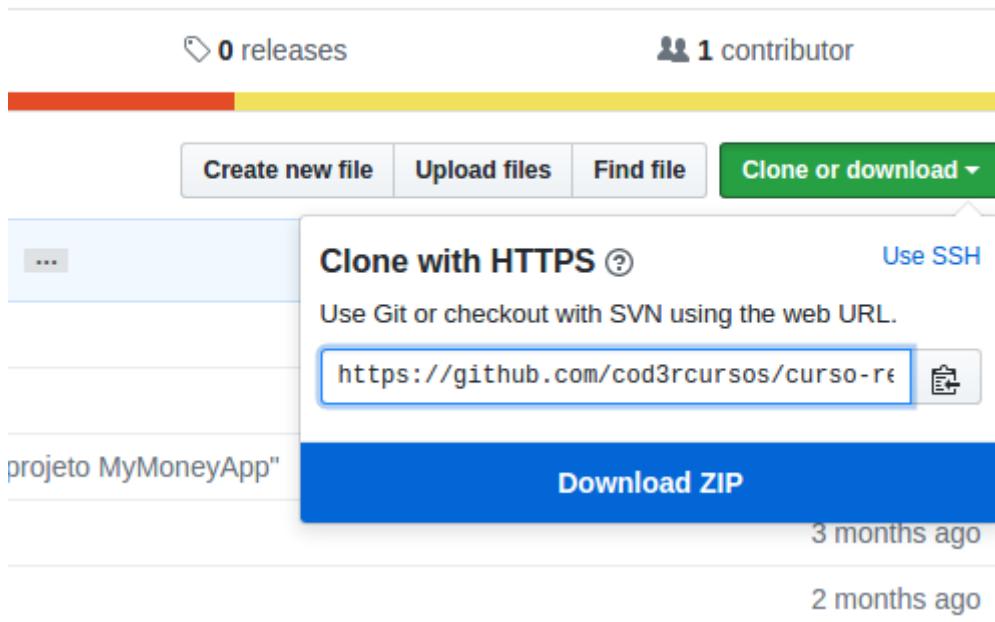


Figura 3. Opção para Baixar Repositório como .ZIP

1.4. Autor



Leonardo Leitão é graduado em Engenharia Elétrica pela Universidade Federal do Ceará e Mestre em Informática Aplicada pela Universidade de Fortaleza, na qual trabalhou com Integração de Redes de Sensores sem Fio e Computação em Nuvem. Há doze anos atua na área como desenvolvedor de softwares e atualmente trabalha na DATAPREV como arquiteto, desenvolvendo sistemas para o INSS. Professor de desenvolvimento de software há quase dez anos.

Em 2016 foi um dos fundadores da **Cod3r**, na qual atua na criação de cursos online voltados para desenvolvimento de software. Com mais de 17.000 alunos, a Cod3r tem se tornado uma referência em ensino de qualidade no Brasil na área de programação.

1.5. Suporte

Não basta um curso ter qualidade, é fundamental um bom suporte para que você tenha uma excelente experiência no aprendizado. E nossa equipe estará à disposição para sanar eventuais dúvidas que você tenha durante as aulas.

E pra que a gente consiga oferecer um melhor suporte e de forma mais rápida, é importante que algumas regras sejam seguidas:

- **Manter perguntas separadas** - Não aproveite perguntas de outros alunos para fazer novas perguntas. Mesmo quando o erro parece ser o mesmo, as causas geralmente são diferentes.
- **Disponibilize seu código no Git** - Quando um problema ocorre, é muito comum eu ter que analisar o seu código para saber de fato o que está gerando o problema reportado, então disponibilize o seu código em algum repositório público. Poder ser o Github, Bitbucket, Gitlab... Você escolhe!
- **Suporte dentro do escopo** - Manter as dúvidas sobre questões relacionadas à execução dos exercícios e as dúvidas conceituais relativas ao escopo do curso. É inviável para nós professores, analisar problemas específicos ou implementar soluções para atender necessidades específicas de um aluno.
- **Curso está aberto** - Se entendermos que existe a demanda de adicionar um tópico que seja necessidade de um conjunto expressivo de alunos e que julgarmos que faz parte do escopo do curso, pode ter certeza que iremos acrescentar o material com o maior prazer.

Conto com vocês e pode ter certeza que estaremos presente na plataforma para te ajudar!

2. Fundamentos

2.1. Primeiro Programa em Go

Listagem 1 - Primeiro Programa em Go

fundamentos/primeiro.go

```
// Programas executáveis iniciam pelo pacote main
package main

/*
Os códigos em Go são organizados em pacotes
e para usá-los é necessário declarar um ou vários imports
*/
import "fmt"

// A porta de entrada de um programa Go é a função main
func main() {
    fmt.Println("Primeiro ")
    fmt.Println("Programa!")

    /*
        Sobre comentários...
        1) Priorize código legível e faça comentários que agrega valor!
        2) Evite comentários óbvios
        3) Durante o curso abuse dos comentários
    */
}
```

2.2. Explorando os Comandos do Terminal

Listagem 2 - Explorando os Comandos do Terminal

fundamentos/comandos.go

```
package main

import "fmt"

func main() {
    fmt.Printf("Outro programa em %s!!!!\n", "Go")
}
```

Listagem 3 - Comandos no Terminal

Executado no terminal

```
$ go
$ go help get
$ go version
$ godoc -http=:6060
$ go env
$ go doc cmd/vet
$ go vet comandos.go
$ go build comandos.go
$ ./comandos
$ go run comandos.go
# Mac/Linux
$ ls ~/go/src/github.com
# Windows
$ dir ~/go/src/github.com
$ go get -u github.com/go-sql-driver/mysql
```

2.3. Constantes e Variáveis

Listagem 4 - Constantes e Variáveis

fundamentos/constvar.go

```
package main

import (
    "fmt"
    m "math"
)

func main() {
    const PI float64 = 3.1415
    var raio = 3.2 // tipo (float64) inferido pelo compilador

    // forma reduzida de criar uma var
    area := PI * m.Pow(raio, 2)
    fmt.Println("A área da circunferência é", area)

    const (
        a = 1
        b = 2
    )

    var (
        c = 3
        d = 4
    )

    fmt.Println(a, b, c, d)

    var e, f bool = true, false
    fmt.Println(e, f)

    g, h, i := 2, false, "epa!"
    fmt.Println(g, h, i)
}
```

2.4. Imprimindo Valores no Console

Listagem 5 - Imprimindo Valores no Console

fundamentos/prints.go

```
package main

import "fmt"

func main() {
    fmt.Println("Mesma")
    fmt.Println(" linha.")

    fmt.Println(" Nova")
    fmt.Println("linha.")

    x := 3.141516

    // fmt.Println("O valor de x é " + x)
    xs := fmt.Sprint(x)
    fmt.Println("O valor de x é " + xs)
    fmt.Println("O valor de x é", x)

    fmt.Printf("O valor de x é %.2f.", x)

    a := 1
    b := 1.9999
    c := false
    d := "opa"
    fmt.Printf("\n%d %f %.1f %t %s", a, b, b, c, d)
    fmt.Printf("\n%v %v %v %v", a, b, c, d)
}
```

2.5. Tipos Básicos

Listagem 6 - Tipos Básicos

fundamentos/tipos.go

```
package main

import (
    "fmt"
    "math"
    "reflect"
)

func main() {
    // números inteiros
```

```

fmt.Println(1, 2, 1000)
fmt.Println("Literal inteiro é", reflect.TypeOf(32000))

// sem sinal (só positivos)... uint8 uint16 uint32 uint64
var b byte = 255
fmt.Println("O byte é", reflect.TypeOf(b))

// com sinal... int8 int16 int32 int64
i1 := math.MaxInt64
fmt.Println("O valor máximo do int é", i1)
fmt.Println("O tipo de i1 é", reflect.TypeOf(i1))

var i2 rune = 'a' // representa um mapeamento da tabela Unicode (int32)
fmt.Println("O rune é", reflect.TypeOf(i2))
fmt.Println(i2)

// números reais (float32, float64)
var x float32 = 49.99
fmt.Println("O tipo de x é", reflect.TypeOf(x))
fmt.Println("O tipo do literal 49.99 é", reflect.TypeOf(49.99)) // float64

// boolean
bo := true
fmt.Println("O tipo de bo é", reflect.TypeOf(bo))
fmt.Println(!bo)

// string
s1 := "Olá meu nome é Leo"
fmt.Println(s1 + "!")
fmt.Println("O tamanho da string é", len(s1))

// string com multiplas linhas
s2 := `Olá
meu
nome
é
Leo`
fmt.Println("O tamanho da string é", len(s2))

// char???
// var x char = 'b'
char := 'a'
fmt.Println("O tipo de char é", reflect.TypeOf(char))
fmt.Println(char)
}

```

2.6. Tipos Básicos: Os Zeros

Listagem 7 - Tipos Básicos: Os Zeros

[fundamentos/zeros.go](#)

```
package main

import "fmt"

func main() {
    var a int
    var b float64
    var c bool
    var d string
    var e *int

    fmt.Printf("%v %v %v %q %v", a, b, c, d, e)
}
```

2.7. Conversão entre Tipos Básicos

Listagem 8 - Conversão entre Tipos Básicos

fundamentos/conversoes.go

```
package main

import (
    "fmt"
    "strconv"
)

func main() {
    x := 2.4
    y := 2
    fmt.Println(x / float64(y))

    nota := 6.9
    notaFinal := int(nota)
    fmt.Println(notaFinal)

    // cuidado...
    fmt.Println("Teste " + string(97))

    // int para string
    fmt.Println("Teste " + strconv.Itoa(123))

    // string para int
    num, _ := strconv.Atoi("123")
    fmt.Println(num - 122)

    b, _ := strconv.ParseBool("true")
    if b {
        fmt.Println("Verdadeiro")
    }
}
```

2.8. Funções Básicas

Listagem 9 - Funções Básicas

fundamentos/main.go

```
package main

func main() {
    resultado := somar(3, 4)
    imprimir(resultado)
}
```

Listagem 10 - ??

fundamentos/funcoes.go

```
package main

import "fmt"

func somar(a int, b int) int {
    return a + b
}

func imprimir(valor int) {
    fmt.Println(valor)
}
```

2.9. Operadores Aritméticos

Listagem 11 - Operadores Aritméticos

fundamentos/aritmeticos.go

```
package main

import (
    "fmt"
    "math"
)

func main() {
    a := 3
    b := 2

    fmt.Println("Soma =>", a+b)
    fmt.Println("Subtração =>", a-b)
    fmt.Println("Divisão =>", a/b)
    fmt.Println("Multiplicação =>", a*b)
    fmt.Println("Módulo =>", a%b)

    // bitwise
    fmt.Println("E =>", a&b)    // 11 & 10 = 10
    fmt.Println("Ou =>", a|b)   // 11 | 10 = 11
    fmt.Println("Xor =>", a^b)  // 11 ^ 10 = 01

    c := 3.0
    d := 2.0

    // outras operações usando math...
    fmt.Println("Maior =>", math.Max(float64(a), float64(b)))
    fmt.Println("Menor =>", math.Min(c, d))
    fmt.Println("Exponenciação =>", math.Pow(c, d))
}
```

2.10. Operadores de Atribuição

Listagem 12 - Operadores de Atribuição

fundamentos/atribuicao.go

```
package main

import "fmt"

func main() {
    var b byte = 3
    fmt.Println(b)

    i := 3 // inferência de tipo
    i += 3 // i = i + 3
    i -= 3 // i = i - 3
    i /= 2 // i = i / 2
    i *= 2 // i = i * 2
    i %= 2 // i = i % 2

    fmt.Println(i)

    x, y := 1, 2
    fmt.Println(x, y)

    x, y = y, x
    fmt.Println(x, y)
}
```

2.11. Operadores Relacionais

Listagem 13 - Operadores Relacionais

fundamentos/relacionais.go

```
package main

import (
    "fmt"
    "time"
)

func main() {
    fmt.Println("Strings:", "Banana" == "Banana")
    fmt.Println("!=", 3 != 2)
    fmt.Println("<", 3 < 2)
    fmt.Println(">", 3 > 2)
    fmt.Println("<=", 3 <= 2)
    fmt.Println(">=", 3 >= 2)

    d1 := time.Unix(0, 0)
    d2 := time.Unix(0, 0)

    fmt.Println("Datas:", d1 == d2)
    fmt.Println("Datas:", d1.Equal(d2))

    type Pessoa struct {
        Nome string
    }

    p1 := Pessoa{"João"}
    p2 := Pessoa{"João"}
    fmt.Println("Pessoas:", p1 == p2)
}
```

2.12. Operadores Lógicos

Listagem 14 - Operadores Lógicos

fundamentos/logicos.go

```
package main

import "fmt"

func compras(trab1, trab2 bool) (bool, bool, bool) {
    comprarTv50 := trab1 && trab2
    comprarTv32 := trab1 != trab2 // ou exclusivo
    comprarSorvete := trab1 || trab2

    return comprarTv50, comprarTv32, comprarSorvete
}

func main() {
    tv50, tv32, sorvete := compras(true, true)
    fmt.Printf("Tv50: %t, Tv32: %t, Sorvete: %t, Saudável: %t",
        tv50, tv32, sorvete, !sorvete)
}
```

2.13. Operadores Unários

Listagem 15 - Operadores Unários

fundamentos/unario.go

```
package main

import "fmt"

func main() {
    x := 1
    y := 2

    // apenas postfix
    x++ // x += 1 ou x = x + 1
    fmt.Println(x)

    y-- // y -= 1 ou y = y - 1
    fmt.Println(y)

    // fmt.Println(x == y--)
}
```

2.14. Operador Ternário???

Listagem 16 - Operador Ternário???

fundamentos/naoternario.go

```
package main

import "fmt"

// Não tem operador ternário
func obterResultado(nota float64) string {
    // return nota >= 6 ? "Aprovado" : "Reprovado"
    if nota >= 6 {
        return "Aprovado"
    }
    return "Reprovado"
}

func main() {
    fmt.Println(obterResultado(6.2))
}
```

2.15. Ponteiros em Go

Listagem 17 - Ponteiros em Go

fundamentos/ponteiro.go

```
package main

import "fmt"

func main() {
    i := 1

    var p *int = nil
    p = &i // pegando o endereço da variável
    *p++ // desreferenciando (pegando o valor)
    i++

    // Go não tem aritmética de ponteiros
    // p++

    fmt.Println(p, *p, i, &i)
}
```

3. Estruturas de Controle

3.1. If/Else

Listagem 18 - *If/Else*

controles/ifelse.go

```
package main

import "fmt"

func imprimirResultado(nota float64) {
    if nota >= 7 {
        fmt.Println("Aprovado com nota", nota)
    } else {
        fmt.Println("Reprovado com nota", nota)
    }
}

func main() {
    imprimirResultado(7.3)
    imprimirResultado(5.1)
}
```

3.2. If/Else If

Listagem 19 - If/Else If

controles/ifelseif.go

```
package main

import "fmt"

func notaParaConceito(n float64) string {
    if n >= 9 && n <= 10 {
        return "A"
    } else if n >= 8 && n < 9 {
        return "B"
    } else if n >= 5 && n < 8 {
        return "C"
    } else if n >= 3 && n < 5 {
        return "D"
    } else {
        return "E"
    }
}

func main() {
    fmt.Println(notaParaConceito(9.8))
    fmt.Println(notaParaConceito(6.9))
    fmt.Println(notaParaConceito(2.1))
}
```

3.3. If com Init

Listagem 20 - If com Init

controles/ifinit.go

```
package main

import (
    "fmt"
    "math/rand"
    "time"
)

func numeroAleatorio() int {
    s := rand.NewSource(time.Now().UnixNano())
    r := rand.New(s)
    return r.Intn(10)
}

func main() {
    if i := numeroAleatorio(); i > 5 { // tb suportado no switch
        fmt.Println("Ganhou!!!")
    } else {
        fmt.Println("Perdeu!")
    }
}
```

3.4. Laço For

Listagem 21 - Laço For

controles/for.go

```
package main

import (
    "fmt"
    "time"
)

func main() {

    i := 1
    for i <= 10 { // não tem while em Go
        fmt.Println(i)
        i++
    }

    for i := 0; i <= 20; i += 2 {
        fmt.Printf("%d ", i)
    }

    fmt.Println("\nMisturando... ")
    for i := 1; i <= 10; i++ {
        if i%2 == 0 {
            fmt.Print("Par ")
        } else {
            fmt.Print("Impar ")
        }
    }

    for {
        // laço infinito
        fmt.Println("Para sempre...")
        time.Sleep(time.Second)
    }

    // Veremos o foreach no capítulo de array
}
```

3.5. Switch #01

Listagem 22 - Switch #01

controles/switch.go

```
package main

import "fmt"

func notaParaConceito(n float64) string {
    var nota = int(n)
    switch nota {
    case 10:
        fallthrough
    case 9:
        return "A"
    case 8, 7:
        return "B"
    case 6, 5:
        return "C"
    case 4, 3:
        return "D"
    case 2, 1, 0:
        return "E"
    default:
        return "Nota inválida"
    }
}

func main() {
    fmt.Println(notaParaConceito(9.8))
    fmt.Println(notaParaConceito(6.9))
    fmt.Println(notaParaConceito(2.1))
}
```

3.6. Switch #02

Listagem 23 - Switch #02

controles/switch.go

```
package main

import (
    "fmt"
    "time"
)

func main() {
    t := time.Now()
    switch { // switch true
    case t.Hour() < 12:
        fmt.Println("Bom dia!")
    case t.Hour() < 18:
        fmt.Println("Boa tarde.")
    default:
        fmt.Println("Boa noite.")
    }
}
```

3.7. Resposta do Desafio Switch

Listagem 24 - Resposta do Desafio Switch

controles/desafio.go

```
package main

import "fmt"

func notaParaConceito(n float64) string {
    switch {
    case n >= 9 && n <= 10:
        return "A"
    case n >= 8 && n < 9:
        return "B"
    case n >= 5 && n < 8:
        return "C"
    case n >= 3 && n < 5:
        return "D"
    default:
        return "E"
    }
}

func main() {
    fmt.Println(notaParaConceito(9.8))
    fmt.Println(notaParaConceito(6.9))
    fmt.Println(notaParaConceito(2.1))
}
```

3.8. Switch #03

Listagem 25 - Switch #03

controles/switch.go

```
package main

import (
    "fmt"
    "time"
)

func tipo(i interface{}) string {
    switch i.(type) {
    case int:
        return "inteiro"
    case float32, float64:
        return "real"
    case string:
        return "string"
    case func():
        return "função"
    default:
        return "não sei"
    }
}

func main() {
    fmt.Println(tipo(2.3))
    fmt.Println(tipo(1))
    fmt.Println(tipo("Opa"))
    fmt.Println(tipo(func() {}))
    fmt.Println(tipo(time.Now()))
}
```

4. Array/Slices/Map

4.1. Trabalhando com Arrays

Listagem 26 - Trabalhando com Arrays

arrayslicemap/array.go

```
package main

import "fmt"

func main() {
    // homogênea (mesmo tipo) e estática (fixo)
    var notas [3]float64
    fmt.Println(notas)

    notas[0], notas[1], notas[2] = 7.8, 4.3, 9.1
    // notas[3] = 10
    fmt.Println(notas)

    total := 0.0
    for i := 0; i < len(notas); i++ {
        total += notas[i]
    }

    media := total / float64(len(notas))
    fmt.Printf("Média %.2f\n", media)
}
```

4.2. Percorrendo Arrays com For (Range)

Listagem 27 - Percorrendo Arrays com For (Range)

arrayslicemap/forrange.go

```
package main

import "fmt"

func main() {
    numeros := [...]int{1, 2, 3, 4, 5} // compilador conta!

    for i, numero := range numeros {
        fmt.Printf("%d) %d\n", i+1, numero)
    }

    for _, num := range numeros {
        fmt.Println(num)
    }
}
```

4.3. Conhecendo o Slice

Listagem 28 - Conhecendo o Slice

arrayslicemap/slice.go

```
package main

import (
    "fmt"
    "reflect"
)

func main() {
    a1 := [3]int{1, 2, 3} // array
    s1 := []int{1, 2, 3}  // slice
    fmt.Println(a1, s1)
    fmt.Println(reflect.TypeOf(a1), reflect.TypeOf(s1))

    a2 := [5]int{1, 2, 3, 4, 5}

    // Slice não é um array! Slice define um pedaço de um array.
    s2 := a2[1:3]
    fmt.Println(a2, s2)

    s3 := a2[:2] // novo slice, mas aponta para o mesmo array
    fmt.Println(a2, s3)

    // vc pode imaginar um slice como: tamanho e um ponteiro para um elemento de
    // um array
    s4 := s2[:1]
    fmt.Println(s2, s4)
}
```

4.4. Construindo Slices com Make

Listagem 29 - Construindo Slices com Make

arrayslicemap/slicemake.go

```
package main

import "fmt"

func main() {
    s := make([]int, 10)
    s[9] = 12
    fmt.Println(s)

    s = make([]int, 10, 20)
    fmt.Println(s, len(s), cap(s))

    s = append(s, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0)
    fmt.Println(s, len(s), cap(s))

    s = append(s, 1)
    fmt.Println(s, len(s), cap(s))
}
```

4.5. Array Interno

Listagem 30 - Array Interno

arrayslicemap/arrayinterno.go

```
package main

import "fmt"

func main() {
    s1 := make([]int, 10, 20)
    s2 := append(s1, 1, 2, 3)
    fmt.Println(s1, s2)

    s1[0] = 7
    fmt.Println(s1, s2)
}
```

4.6. Slice: Usando Append e Copy

Listagem 31 - Slice: Usando Append e Copy

`arrayslicemap/appendcopy.go`

```
package main

import "fmt"

func main() {
    array1 := [3]int{1, 2, 3}
    var slice1 []int

    // array1 = append(array1, 4, 5, 6)
    slice1 = append(slice1, 4, 5, 6)
    fmt.Println(array1, slice1)

    slice2 := make([]int, 2)
    copy(slice2, slice1)
    fmt.Println(slice2)
}
```

4.7. Trabalhando com Maps #01

Listagem 32 - Trabalhando com Maps #01

map.go

```
package main

import "fmt"

func main() {
    // var aprovados map[int]string
    // mapas devem ser inicializados
    aprovados := make(map[int]string)

    aprovados[12345678978] = "Maria"
    aprovados[98765432100] = "Pedro"
    aprovados[95135745682] = "Ana"
    fmt.Println(aprovados)

    for cpf, nome := range aprovados {
        fmt.Printf("%s (CPF: %d)\n", nome, cpf)
    }

    fmt.Println(aprovados[95135745682])
    delete(aprovados, 95135745682)
    fmt.Println(aprovados[95135745682])
}
```

4.8. Trabalhando com Maps #02

Listagem 33 - Trabalhando com Maps #02

arrayslicemap/map.go

```
package main

import "fmt"

func main() {
    funcsESalarios := map[string]float64{
        "José João":      11325.45,
        "Gabriela Silva": 15456.78,
        "Pedro Junior":   1200.0,
    }

    funcsESalarios["Rafael Filho"] = 1350.0
    delete(funcsESalarios, "Inexistente")

    for nome, salario := range funcsESalarios {
        fmt.Println(nome, salario)
    }
}
```

4.9. Maps Aninhados

Listagem 34 - Maps Aninhados

arrayslicemap/mapaninhado.go

```
package main

import "fmt"

func main() {
    funcsPorLetra := map[string]map[string]float64{
        "G": {
            "Gabriela Silva": 15456.78,
            "Guga Pereira": 8456.78,
        },
        "J": {
            "José João": 11325.45,
        },
        "P": {
            "Pedro Junior": 1200.0,
        },
    }

    delete(funcsPorLetra, "P")

    for letra, funcs := range funcsPorLetra {
        fmt.Println(letra, funcs)
    }
}
```

5. Funções

5.1. Funções Básicas

Listagem 35 - Funções Básicas

funcoes/basicas.go

```
package main

import "fmt"

func f1() {
    fmt.Println("F1")
}

func f2(p1 string, p2 string) {
    fmt.Printf("F2: %s %s\n", p1, p2)
}

func f3() string {
    return "F3"
}

func f4(p1, p2 string) string {
    return fmt.Sprintf("F4: %s %s", p1, p2)
}

func f5() (string, string) {
    return "Retorno 1", "Retorno 2"
}

func main() {
    f1()
    f2("Param1", "Param2")

    r3, r4 := f3(), f4("Param1", "Param2")
    fmt.Println(r3)
    fmt.Println(r4)

    r51, r52 := f5()
    fmt.Println("F5:", r51, r52)
}
```

5.2. Pilha de Funções

Listagem 36 - Pilha de Funções

funcoes/pilha.go

```
package main

import "runtime/debug"

func f3() {
    debug.PrintStack()
}

func f2() {
    f3()
}

func f1() {
    f2()
}

func main() {
    f1()
}
```

5.3. Retorno Nomeado

Listagem 37 - Retorno Nomeado

funcoes/retornonomeado.go

```
package main

import "fmt"

func trocar(p1, p2 int) (segundo, primeiro int) {
    segundo = p2
    primeiro = p1
    return // retorno limpo
}

func main() {
    x, y := trocar(2, 3)
    fmt.Println(x, y)

    segundo, primeiro := trocar(7, 1)
    fmt.Println(segundo, primeiro)
}
```

5.4. Armazenar Funções em Variáveis

Listagem 38 - Armazenar Funções em Variáveis

funcoes/primeiraclasse.go

```
package main

import "fmt"

var soma = func(a, b int) int {
    return a + b
}

func main() {
    fmt.Println(soma(2, 3))

    sub := func(a, b int) int {
        return a - b
    }

    fmt.Println(sub(2, 3))
}
```

5.5. Passar Função como Parâmetro

Listagem 39 - Passar Função como Parâmetro

funcoes/comoparametro.go

```
package main

import "fmt"

func multiplicacao(a, b int) int {
    return a * b
}

func exec(funcao func(int, int) int, p1, p2 int) int {
    return funcao(p1, p2)
}

func main() {
    resultado := exec(multiplicacao, 3, 4)
    fmt.Println(resultado)
}
```

5.6. Funções Variáticas #01

Listagem 40 - Funções Variáticas #01

funcoes/variatica.go

```
package main

import "fmt"

func media(numeros ...float64) float64 {
    total := 0.0
    for _, num := range numeros {
        total += num
    }
    return total / float64(len(numeros))
}

func main() {
    fmt.Printf("Média: %.2f", media(7.7, 8.1, 5.9, 9.9))
}
```

5.7. Funções Variáticas #02

Listagem 41 - Funções Variáticas #02

funcoes/variaticaslice.go

```
package main

import "fmt"

func imprimirAprovados(aprovados ...string) {
    fmt.Println("Lista de Aprovados")
    for i, aprovado := range aprovados {
        fmt.Printf("%d) %s\n", i+1, aprovado)
    }
}

func main() {
    aprovados := []string{"Maria", "Pedro", "Rafael", "Guilherme"}
    imprimirAprovados(aprovados...)
}
```

5.8. Closure

Listagem 42 - Closure

funcoes/closure.go

```
package main

import "fmt"

func closure() func() {
    x := 10
    var funcao = func() {
        fmt.Println(x)
    }
    return funcao
}

func main() {
    x := 20
    fmt.Println(x)

    imprimeX := closure()
    imprimeX()
}
```

5.9. Recursividade

Listagem 43 - Recursividade

funcoes/recursividade.go

```
package main

import "fmt"

func factorial(n int) (int, error) {
    switch {
    case n < 0:
        return -1, fmt.Errorf("número inválido: %d", n)
    case n == 0:
        return 1, nil
    default:
        fatorialAnterior, _ := factorial(n - 1)
        return n * fatorialAnterior, nil
    }
}

func main() {
    resultado, _ := factorial(5)
    fmt.Println(resultado)

    _, err := factorial(-4)
    if err != nil {
        fmt.Println(err)
    }

    // Uma solução melhor seria... uint!
}
```

5.10. Recursividade (Simples)

Listagem 44 - Recursividade (Simples)

funcoes/recursividade.go

```
package main

import "fmt"

func fatorial(n uint) uint {
    switch {
    case n == 0:
        return 1
    default:
        return n * fatorial(n-1)
    }
}

func main() {
    resultado := fatorial(5)
    fmt.Println(resultado)
}
```

5.11. Defer

Listagem 45 - Defer

funcoes/defer.go

```
package main

import "fmt"

func obterValorAprovado(numero int) int {
    defer fmt.Println("fim!")
    if numero > 5000 {
        fmt.Println("Valor alto...")
        return 5000
    }
    fmt.Println("Valor baixo...")
    return numero
}

func main() {
    fmt.Println(obterValorAprovado(6000))
    fmt.Println(obterValorAprovado(3000))
}
```

5.12. Passando Ponteiro para Função

Listagem 46 - Passando Ponteiro para Função

funcoes/ponteiro.go

```
package main

import "fmt"

func inc1(n int) {
    n++ // n = n + 1
}

// revisão: um ponteiro é representado por um *
func inc2(n *int) {
    // revisão: * é usado para desreferenciar, ou seja,
    // ter acesso ao valor no qual o ponteiro aponta
    *n++
}

func main() {
    n := 1

    inc1(n) // por valor
    fmt.Println(n)

    // revisão: & usado para obter o endereço da variável
    inc2(&n) // por referência
    fmt.Println(n)
}
```

5.13. Função Init

Listagem 47 - Função Init

funcoes/init.go

```
package main

import "fmt"

func init() {
    fmt.Println("Inicializando...")
}

func main() {
    fmt.Println("Main...")
}
```

Listagem 48 - ??

funcoes/init2.go

```
package main

import "fmt"

func init() {
    fmt.Println("Inicializando2...")
}
```

6. Sistema de Tipos

6.1. Usando Struct

Listagem 49 - Usando Struct

tipos/struct.go

```
package main

import "fmt"

type produto struct {
    nome      string
    preco     float64
    desconto float64
}

// Método: função com receiver (receptor)
func (p produto) precoComDesconto() float64 {
    return p.preco * (1 - p.desconto)
}

func main() {
    var produto1 produto
    produto1 = produto{
        nome:      "Lapis",
        preco:    1.79,
        desconto: 0.05,
    }
    fmt.Println(produto1, produto1.precoComDesconto())

    produto2 := produto{"Notebook", 1989.90, 0.10}
    fmt.Println(produto2.precoComDesconto())
}
```

6.2. Struct Aninhada

Listagem 50 - Struct Aninhada

tipos/structaninhada.go

```
package main

import "fmt"

type item struct {
    produtoID int
    qtde      int
    preco     float64
}

type pedido struct {
    userID int
    itens  []item
}

func (p pedido) valorTotal() float64 {
    total := 0.0
    for _, item := range p.itens {
        total += item.preco * float64(item.qtde)
    }
    return total
}

func main() {
    pedido := pedido{
        userID: 1,
        itens: []item{
            item{produtoID: 1, qtde: 2, preco: 12.10},
            item{2, 1, 23.49},
            item{11, 100, 3.13},
        },
    }
    fmt.Printf("Valor total do pedido é R$ %.2f", pedido.valorTotal())
}
```

6.3. Métodos em Structs

Listagem 51 - Métodos em Structs

tipos/metodos.go

```
package main

import (
    "fmt"
    "strings"
)

type pessoa struct {
    nome      string
    sobrenome string
}

func (p pessoa) getNomeCompleto() string {
    return p.nome + " " + p.sobrenome
}

func (p *pessoa) setNomeCompleto(nomeCompleto string) {
    partes := strings.Split(nomeCompleto, " ")
    p.nome = partes[0]
    p.sobrenome = partes[1]
}

func main() {
    p1 := pessoa{"Pedro", "Silva"}
    fmt.Println(p1.getNomeCompleto())

    p1.setNomeCompleto("Maria Pereira")
    fmt.Println(p1.getNomeCompleto())
}
```

6.4. Pseudo-Herança em Structs

Listagem 52 - Pseudo-Herança em Structs

tipos/pseudoheranca.go

```
package main

import "fmt"

type carro struct {
    nome          string
    velocidadeAtual int
}

type ferrari struct {
    carro        // campos anonimos
    turboLigado bool
}

func main() {
    f := ferrari{}
    f.nome = "F40"
    f.velocidadeAtual = 0
    f.turboLigado = true

    fmt.Printf("A ferrari %s está com turbo ligado? %v\n", f.nome, f.turboLigado)
    fmt.Println(f)
}
```

6.5. Tipo Personalizado

Listagem 53 - *Tipo Personalizado*

tipos/meutipo.go

```
package main

import "fmt"

type nota float64

func (n nota) entre(inicio, fim float64) bool {
    return float64(n) >= inicio && float64(n) <= fim
}

func notaParaConceito(n nota) string {
    if n.entre(9.0, 10.0) {
        return "A"
    } else if n.entre(7.0, 8.99) {
        return "B"
    } else if n.entre(5.0, 7.99) {
        return "C"
    } else if n.entre(3.0, 4.99) {
        return "D"
    } else {
        return "E"
    }
}

func main() {
    fmt.Println(notaParaConceito(9.8))
    fmt.Println(notaParaConceito(6.9))
    fmt.Println(notaParaConceito(2.1))
}
```

6.6. Usando Interfaces #01

Listagem 54 - Usando Interfaces #01

tipos/interface.go

```
package main

import "fmt"

type imprimivel interface {
    toString() string
}

type pessoa struct {
    nome      string
    sobrenome string
}

type produto struct {
    nome  string
    preco float64
}

// interfaces são implementadas implicitamente
func (p pessoa) toString() string {
    return p.nome + " " + p.sobrenome
}

func (p produto) toString() string {
    return fmt.Sprintf("%s - R$ %.2f", p.nome, p.preco)
}

func imprimir(x imprimivel) {
    fmt.Println(x.toString())
}

func main() {
    var coisa imprimivel = pessoa{"Roberto", "Silva"}
    fmt.Println(coisa.toString())
    imprimir(coisa)

    coisa = produto{"Calça Jeans", 79.90}
    fmt.Println(coisa.toString())
    imprimir(coisa)

    p2 := produto{"Calça Jeans", 179.90}
    imprimir(p2)
}
```

6.7. Usando Interfaces #02

Listagem 55 - Usando Interfaces #02

tipos/interface.go

```
type interface { }
```

6.8. Composição de Interfaces

Listagem 56 - Composição de Interfaces

tipos/composicao.go

```
package main

import "fmt"

type esportivo interface {
    ligarTurbo()
}

type luxuoso interface {
    fazerBaliza()
}

type esportivoLuxuoso interface {
    esportivo
    luxuoso
    // pode adicionar mais métodos
}

type bwm7 struct{ }

func (b bwm7) ligarTurbo() {
    fmt.Println("Turbo...")
}

func (b bwm7) fazerBaliza() {
    fmt.Println("Baliza...")
}

func main() {
    var b esportivoLuxuoso = bwm7{}
    b.ligarTurbo()
    b.fazerBaliza()
}
```

6.9. Tipo Interface

Listagem 57 - Tipo Interface

tipos/tipointerface.go

```
package main

import "fmt"

type curso struct {
    nome string
}

func main() {
    var coisa interface{}
    fmt.Println(coisa)

    coisa = 3
    fmt.Println(coisa)

    type dinamico interface{}

    var coisa2 dinamico = "Opa"
    fmt.Println(coisa2)

    coisa2 = curso{"Golang: Explorando a Linguagem do Google"}
    fmt.Println(coisa2)
}
```

6.10. Convertendo uma Struct em JSON

Listagem 58 - Convertendo uma Struct em JSON

tipos/json.go

```
package main

import (
    "encoding/json"
    "fmt"
)

type produto struct {
    ID      int      `json:"id"`
    Nome   string   `json:"nome"`
    Preco  float64  `json:"preco"`
    Tags   []string `json:"tags"`
}

func main() {
    // struct para json
    p1 := produto{1, "Notebook", 1899.90, []string{"Promoção", "Eletrônico"}}
    p1Json, _ := json.Marshal(p1)
    fmt.Println(string(p1Json))

    // json para struct
    var p2 produto
    jsonString :=
        `{"id":2,"nome":"Caneta","preco":8.90,"tags":["Papelaria","Importado"]}`
    json.Unmarshal([]byte(jsonString), &p2)
    fmt.Println(p2.Tags[1])
}
```

7. Pacotes

7.1. Pacotes & Visibilidade

Listagem 59 - Funções relacionadas com Reta

pacote/reta.go

```
package main

import "math"

// Iniciando com letra maiúscula é PÚBLICO (visível dentro e fora do pacote)!
// Iniciando com letra minúscula é PRIVADO (visível apenas dentro do pacote)!

// Por exemplo...
// Ponto - gerará algo público
// ponto ou _Ponto - gerará algo privado

// Ponto representa uma coordenada no plano cartesiano
type Ponto struct {
    x float64
    y float64
}

func catetos(a, b Ponto) (cx, cy float64) {
    cx = math.Abs(b.x - a.x)
    cy = math.Abs(b.y - a.y)
    return
}

// Distancia é responsável por calcular a distância linear entre dois pontos
func Distancia(a, b Ponto) float64 {
    cx, cy := catetos(a, b)
    return math.Sqrt(math.Pow(cx, 2) + math.Pow(cy, 2))
}
```

Listagem 60 - Arquivo Principal

pacote/main.go

```
package main

import "fmt"

func main() {
    p1 := Ponto{2.0, 2.0}
    p2 := Ponto{2.0, 4.0}

    fmt.Println(catetos(p1, p2))
    fmt.Println(Distancia(p1, p2))
}
```

7.2. Criando um Pacote Reutilizável

Listagem 61 - Criando um Pacote Reutilizável

pacote/main.go

```
package main

import (
    "fmt"
    "github.com/cod3rcursos/area"
)

func main() {
    fmt.Println(area.Circ(6.0))
    fmt.Println(area.Rect(5.0, 2.0))
    // fmt.Println(area._TrianguloEq(5.0, 2.0))
}
```

7.3. Criando & Instalando um Pacote do Github

Listagem 62 - Criando & Instalando um Pacote do Github

[pacote/usandolib.go](#)

```
package main

import "github.com/cod3rcursos/goarea"
import "fmt"

func main() {
    fmt.Println(goarea.Circ(4.0))
}
```

8. Concorrência

8.1. Cuidado com os Deadlocks

Listagem 63 - Cuidado com os Deadlocks

concorrencia/bloqueio.go

```
package main

import (
    "fmt"
    "time"
)

func rotina(c chan int) {
    time.Sleep(time.Second)
    c <- 1 // operação bloqueante
    fmt.Println("Só depois que foi lido")
}

func main() {
    c := make(chan int) // canal sem buffer

    go rotina(c)

    fmt.Println(<-c) // operação bloqueante
    fmt.Println("Foi lido")
    fmt.Println(<-c) // deadlock
    fmt.Println("Fim")
}
```

8.2. Channel com Buffer

Listagem 64 - Channel com Buffer

concorrencia/buffer.go

```
package main

import (
    "fmt"
    "time"
)

func rotina(ch chan int) {
    ch <- 1
    ch <- 2
    ch <- 3
    ch <- 4
    ch <- 5
    fmt.Println("Executou!")
    ch <- 6
}

func main() {
    ch := make(chan int, 3)
    go rotina(ch)

    time.Sleep(time.Second)
    fmt.Println(<-ch)
}
```

8.3. Conhecendo a Goroutine

Listagem 65 - Conhecendo a Goroutine

concorrencia/goroutine.go

```
package main

import (
    "fmt"
    "time"
)

func fale(pessoa, texto string, qtde int) {
    for i := 0; i < qtde; i++ {
        time.Sleep(time.Second)
        fmt.Printf("%s: %s (iteração %d)\n", pessoa, texto, i+1)
    }
}

func main() {
    // fale("Maria", "Pq vc não fala comigo?", 3)
    // fale("João", "Só posso falar depois de vc!", 1)

    // go fale("Maria", "Ei...", 500)
    // go fale("João", "Opa...", 500)

    go fale("Maria", "Entendi!!!", 10)
    fale("João", "Parabéns!", 5)
}
```

8.4. Conhecendo o Channel (Canal)

Listagem 66 - Conhecendo o Channel (Canal)

concorrencia/channel.go

```
package main

import "fmt"

func main() {
    ch := make(chan int, 1)

    ch <- 1 // enviando dados para o canal (escrita)
    <-ch    // recebendo dados do canal (leitura)

    ch <- 2
    fmt.Println(<-ch)
}
```

8.5. Usando Goroutine e Channel

Listagem 67 - Usando Goroutine e Channel

concorrencia/channel.go

```
package main

import (
    "fmt"
    "time"
)

// Channel (canal) - é a forma de comunicação entre goroutines
// channel é um tipo

func doisTresQuatroVezes(base int, c chan int) {
    time.Sleep(time.Second)
    c <- 2 * base // enviando dados para o canal

    time.Sleep(time.Second)
    c <- 3 * base

    time.Sleep(3 * time.Second)
    c <- 4 * base
}

func main() {
    c := make(chan int)
    go doisTresQuatroVezes(2, c)

    a, b := <-c, <-c // recebendo dados do canal
    fmt.Println(a, b)

    fmt.Println(<-c)
}
```

8.6. Curiosidade: Número de CPUs

Listagem 68 - Curiosidade: Número de CPUs

concorrencia/cpus.go

```
package main

import (
    "fmt"
    "runtime"
)

func main() {
    fmt.Println(runtime.NumCPU())
}
```

8.7. Padrão de Concorrência: Generators

Listagem 69 - Padrão de Concorrência: Generators

concorrencia/generator.go

```
package main

import (
    "fmt"
    "io/ioutil"
    "net/http"
    "regexp"
)

// Google I/O 2012 - Go Concurrency Patterns

// <-chan - canal somente-leitura
func titulo(urls ...string) <-chan string {
    c := make(chan string)
    for _, url := range urls {
        go func(url string) {
            resp, _ := http.Get(url)
            html, _ := ioutil.ReadAll(resp.Body)

            r, _ := regexp.Compile("<title>(.*)<\\\/title>")
            c <- r.FindStringSubmatch(string(html))[1]
        }(url)
    }
    return c
}

func main() {
    t1 := titulo("https://www.cod3r.com.br", "https://www.google.com")
    t2 := titulo("https://www.amazon.com", "https://www.youtube.com")
    fmt.Println("Primeiros:", <-t1, "|", <-t2)
    fmt.Println("Segundos:", <-t1, "|", <-t2)
}
```

8.8. Padrão de Concorrência: Multiplexador

Listagem 70 - ??

concorrencia/multiplexar.go

```
package main

import (
    "fmt"
    "github.com/cod3rcursos/html"
)

func encaminhar(origem <-chan string, destino chan string) {
    for {
        destino <- <-origem
    }
}

// multiplexar - misturar (mensagens) num canal
func juntar(entrada1, entrada2 <-chan string) <-chan string {
    c := make(chan string)
    go encaminhar(entrada1, c)
    go encaminhar(entrada2, c)
    return c
}

func main() {
    c := juntar(
        html.Titulo("https://www.cod3r.com.br", "https://www.google.com"),
        html.Titulo("https://www.amazon.com", "https://www.youtube.com"),
    )
    fmt.Println(<-c, "|", <-c)
    fmt.Println(<-c, "|", <-c)
}
```

8.9. Multiplexador com Select

Listagem 71 - ??

concorrencia/NOME_DO_EXERCICIO.go

```
package main

import (
    "fmt"
    "time"
)

func falar(pessoa string) <-chan string {
    c := make(chan string)
    go func() {
        for i := 0; i < 3; i++ {
            time.Sleep(time.Second)
            c <- fmt.Sprintf("%s falando: %d", pessoa, i)
        }
    }()
    return c
}

func juntar(entrada1, entrada2 <-chan string) <-chan string {
    c := make(chan string)
    go func() {
        for {
            select {
            case s := <-entrada1:
                c <- s
            case s := <-entrada2:
                c <- s
            }
        }()
    }()
    return c
}

func main() {
    c := juntar(falar("João"), falar("Maria"))
    fmt.Println(<-c, <-c)
    fmt.Println(<-c, <-c)
    fmt.Println(<-c, <-c)
}
```

8.10. Channel: Usando Range e Close

Listagem 72 - Channel: Usando Range e Close

concorrencia/primos.go

```
package main

import (
    "fmt"
    "time"
)

func isPrimo(num int) bool {
    for i := 2; i < num; i++ {
        if num%i == 0 {
            return false
        }
    }
    return true
}

func primos(n int, c chan int) {
    inicio := 2
    for i := 0; i < n; i++ {
        for primo := inicio; ; primo++ {
            if isPrimo(primo) {
                c <- primo
                inicio = primo + 1
                time.Sleep(time.Millisecond * 180)
                break
            }
        }
    }
    close(c)
}

func main() {
    c := make(chan int, 30)
    go primos(60, c)
    for primo := range c {
        fmt.Printf("%d ", primo)
    }
    fmt.Println("Fim!")
}
```

8.11. Estrutura de Controle: Select

Listagem 73 - Estrutura de Controle: Select

concorrencia/select.go

```
package main

import (
    "fmt"
    "time"

    "github.com/cod3rcursos/html"
)

func oMaisRapido(url1, url2, url3 string) string {
    c1 := html.Titulo(url1)
    c2 := html.Titulo(url2)
    c3 := html.Titulo(url3)

    // estrutura de controle específica para concorrência
    select {
        case t1 := <-c1:
            return t1
        case t2 := <-c2:
            return t2
        case t3 := <-c3:
            return t3
        case <-time.After(1000 * time.Millisecond):
            return "Todos perderam!"
            // default:
            //    return "Sem resposta ainda!"
    }
}

func main() {
    campeao := oMaisRapido(
        "https://www.youtube.com",
        "https://www.amazon.com",
        "https://www.google.com",
    )
    fmt.Println(campeao)
}
```

9. Testes

9.1. Tipo de Arquitetura e os Testes

Listagem 74 - ??

testes/arquitetura_test.go

```
package arquitetura

import (
    "runtime"
    "testing"
)

func TestDependente(t *testing.T) {
    t.Parallel()
    if runtime.GOARCH == "amd64" {
        t.Skip("Não funciona em arquitetura amd64")
    }
    // ...
    t.Fail()
}
```

9.2. Teste Unitário Básico

Listagem 75 - Teste Unitário Básico

testes/matematica.go

```
package matematica

import (
    "fmt"
    "strconv"
)

// Media é responsável por calcular o que você já sabe... :)
func Media(numeros ...float64) float64 {
    total := 0.0
    for _, num := range numeros {
        total += num
    }
    media := total / float64(len(numeros))
    mediaArredondada, _ := strconv.ParseFloat(fmt.Sprintf("%.2f", media), 64)
    return mediaArredondada
}
```

Listagem 76 - ??

testes/matematica_test.go

```
package matematica

import "testing"

const erroPadrao = "Valor esperado %v, mas o resultado encontrado foi %v."

func TestMedia(t *testing.T) {
    t.Parallel()
    valorEsperado := 7.28
    valor := Media(7.2, 9.9, 6.1, 5.9)

    if valor != valorEsperado {
        t.Errorf(erroPadrao, valorEsperado, valor)
    }
}
```

9.3. Criando Dataset para os Testes

Listagem 77 - Criando Dataset para os Testes

testes/strings_test.go

```
package strings

import (
    "strings"
    "testing"
)

const msgIndex = "%s (parte: %s) - índices: esperado (%d) <> encontrado (%d)."

func TestIndex(t *testing.T) {
    t.Parallel()
    testes := []struct {
        texto    string
        parte    string
        esperado int
    }{
        {"Cod3r é show", "Cod3r", 0},
        {"", "", 0},
        {"Opa", "opa", -1},
        {"leonardo", "o", 2},
    }

    for _, teste := range testes {
        t.Logf("Massa: %v", teste)
        atual := strings.Index(teste.texto, teste.parte)

        if atual != teste.esperado {
            t.Errorf(msgIndex, teste.texto, teste.parte, teste.esperado, atual)
        }
    }
}
```

10. Banco de Dados

10.1. Criando o Schema e a Tabela

Listagem 78 - Criando o Schema e a Tabela

sql/estrutura.go

```
package main

import (
    "database/sql"
    _ "github.com/go-sql-driver/mysql"
)

func exec(db *sql.DB, sql string) sql.Result {
    result, err := db.Exec(sql)
    if err != nil {
        panic(err)
    }
    return result
}

func main() {
    db, err := sql.Open("mysql", "root:123456@/")
    if err != nil {
        panic(err)
    }
    defer db.Close()

    exec(db, "create database if not exists cursogo")
    exec(db, "use cursogo")
    exec(db, "drop table if exists usuarios")
    exec(db, `create table usuarios (
        id integer auto_increment,
        nome varchar(80),
        PRIMARY KEY (id)
    )`)
}
```

10.2. Executando Inserts

Listagem 79 - Executando Inserts

sql/insert.go

```
package main

import (
    "database/sql"
    "fmt"

    _ "github.com/go-sql-driver/mysql"
)

func main() {
    db, err := sql.Open("mysql", "root:123456@/cursogo")
    if err != nil {
        panic(err)
    }
    defer db.Close()

    stmt, _ := db.Prepare("insert into usuarios(nome) values(?)")
    stmt.Exec("Maria")
    stmt.Exec("João")

    res, _ := stmt.Exec("Pedro")

    id, _ := res.LastInsertId()
    fmt.Println(id)

    linhas, _ := res.RowsAffected()
    fmt.Println(linhas)
}
```

10.3. Executando Inserts em uma Transação

Listagem 80 - Executando Inserts em uma Transação

sql/transacao.go

```
package main

import (
    "database/sql"
    "log"

    _ "github.com/go-sql-driver/mysql"
)

func main() {
    db, err := sql.Open("mysql", "root:123456@/cursogo")
    if err != nil {
        log.Fatal(err)
    }
    defer db.Close()

    tx, _ := db.Begin()
    stmt, _ := tx.Prepare("insert into usuarios(id, nome) values(?,?)")

    stmt.Exec(4000, "Bia")
    stmt.Exec(4001, "Carlos")
    _, err = stmt.Exec(1, "Tiago") // chave duplicada

    if err != nil {
        tx.Rollback()
        log.Fatal(err)
    }

    tx.Commit()
}
```

10.4. Executando Update e Delete

Listagem 81 - Executando Update e Delete

sql/update.go

```
package main

import (
    "database/sql"
    "log"

    _ "github.com/go-sql-driver/mysql"
)

func main() {
    db, err := sql.Open("mysql", "root:123456@/cursogo")
    if err != nil {
        log.Fatal(err)
    }
    defer db.Close()

    // update
    stmt, _ := db.Prepare("update usuarios set nome = ? where id = ?")
    stmt.Exec("Uóxiton Iptive", 1)
    stmt.Exec("Sheristone Uasleska", 2)

    // delete
    stmt2, _ := db.Prepare("delete from usuarios where id = ?")
    stmt2.Exec(3)
}
```

10.5. Executando Select e Mapeando p/ um Struct

Listagem 82 - Executando Select e Mapeando p/ um Struct

sql/select.go

```
package main

import (
    "database/sql"
    "fmt"
    "log"
    _ "github.com/go-sql-driver/mysql"
)

type usuario struct {
    id    int
    nome string
}

func main() {
    db, err := sql.Open("mysql", "root:123456@/cursogo")
    if err != nil {
        log.Fatal(err)
    }
    defer db.Close()

    rows, _ := db.Query("select id, nome from usuarios where id > ?", 3)
    defer rows.Close()

    for rows.Next() {
        var u usuario
        rows.Scan(&u.id, &u.nome)
        fmt.Println(u)
    }
}
```

11. Http

11.1. Criando um Servidor Estático

Listagem 83 - Criando um Servidor Estático

http/static.go

```
package main

import (
    "log"
    "net/http"
)

func main() {
    fs := http.FileServer(http.Dir("public"))
    http.Handle("/", fs)

    log.Println("Executando...")
    log.Fatal(http.ListenAndServe(":3000", nil))
}
```

Listagem 84 - Página HTML

http/static/public/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Página usando Go</title>
</head>
<body>
    <h1>Go!!!!</h1>
</body>
</html>
```

11.2. Gerando Conteúdo Dinâmico

Listagem 85 - Gerando Conteúdo Dinâmico

<http/dinamico.go>

```
package main

import (
    "fmt"
    "log"
    "net/http"
    "time"
)

func horaCerta(w http.ResponseWriter, r *http.Request) {
    s := time.Now().Format("02/01/2006 03:04:05")
    fmt.Fprintf(w, "<h1>Hora certa: %s</h1>", s)
}

func main() {
    http.HandleFunc("/horaCerta", horaCerta)
    log.Println("Executando...")
    log.Fatal(http.ListenAndServe(":3000", nil))
}
```

11.3. Integrando Http e SQL (2 Serviços REST)

Listagem 86 - Integrando Http e SQL

<http/cliente.go>

```
package main

import (
    "database/sql"
    "encoding/json"
    "fmt"
    "log"
    "net/http"
    "strconv"
    "strings"
    _ "github.com/go-sql-driver/mysql"
)

// Usuario :)
type Usuario struct {
    ID int `json:"id"`
    Nome string `json:"nome"`

    // Adicionando o campo de senha
    Senha string `json:"senha"`
}
```

```

}

// UsuarioHandler analisa o request e delega para função adequada
func UsuarioHandler(w http.ResponseWriter, r *http.Request) {
    sid := strings.TrimPrefix(r.URL.Path, "/usuarios/")
    id, _ := strconv.Atoi(sid)

    switch {
    case r.Method == "GET" && id > 0:
        usuarioPorID(w, r, id)
    case r.Method == "GET":
        usuarioTodos(w, r)
    default:
        w.WriteHeader(http.StatusNotFound)
        fmt.Fprintf(w, "Desculpa... :(")
    }
}

func usuarioPorID(w http.ResponseWriter, r *http.Request, id int) {
    db, err := sql.Open("mysql", "root:123456@/cursogo")
    if err != nil {
        log.Fatal(err)
    }
    defer db.Close()

    var u Usuario
    db.QueryRow("select id, nome from usuarios where id = ?", id).Scan(&u.ID,
    &u.Nome)

    json, _ := json.Marshal(u)

    w.Header().Set("Content-Type", "application/json")
    fmt.Fprint(w, string(json))
}

func usuarioTodos(w http.ResponseWriter, r *http.Request) {
    db, err := sql.Open("mysql", "root:123456@/cursogo")
    if err != nil {
        log.Fatal(err)
    }
    defer db.Close()

    rows, _ := db.Query("select id, nome from usuarios")
    defer rows.Close()

    var usuarios []Usuario
    for rows.Next() {
        var usuario Usuario
        rows.Scan(&usuario.ID, &usuario.Nome)
        usuarios = append(usuarios, usuario)
    }
}

```

```
    json, _ := json.Marshal(usuarios)

    w.Header().Set("Content-Type", "application/json")
    fmt.Fprint(w, string(json))
}
```

Listagem 87 - Servidor em Go

[http/server.go](#)

```
package main

import (
    "log"
    "net/http"
)

func main() {
    http.HandleFunc("/usuarios/", UsuarioHandler)
    log.Println("Executando...")
    log.Fatal(http.ListenAndServe(":3000", nil))
}
```

Appendix A: Tabela de Códigos

- [\[ex-basic-hello-world\]](#)
- [\[ex-basic-var-const-1\]](#)
- [\[ex-basic-var-const-2\]](#)
- [\[ex-basic-var-const-3\]](#)
- [\[ex-basic-var-const-4\]](#)
- [\[ex-basic-var-const-5\]](#)

Glossário

JWT

...

Middleware

...