**Go**

Sempre começar o código com package main

**Convenções/ Regras**

* A Chave sempre deve ficar na frente da função
* Não necessita de ;
* Você não pode criar uma variável e não utilizar
* Gerar um arquivo para guardar todas as dependências do seu projeto usando no terminal o comando *go mod init NomeDoModulo*
* Se você não utilizar letra maiúscula no início do nome da função ela não irá se tornar “publica” e só poderá ser chamada dentro do próprio arquivo
* Para importar um pacote que eu mesmo criei utilizar o nome do modulo que foi gerado a partir do comando “***go mod init”*** e importar da seguinte forma ***NomeDoModulo / NomeDoPacote***
* Para importar um pacote externo utilizar o comando ***go get LinkParaOndeEstaOArquivo***

Para iniciar um programa em GO você utiliza o comando

*go run nomeDoArquivo.go*

Tipo das variáveis

***var*** *nomeDaVariavel tipoDaVariavel =*

*ou*

*nomeDaVariavel :=*

*ou*

***var*** *(*

*nomeDaVariavel tipoDaVariavel =*

*nomeDaVariavel2 tipoDaVariavel2 =*

*)*

*Ou*

*variavel1* ***,*** *variavel2* ***:=*** *valorDaVariavel1* ***,*** *valorDaVariavel2*

*Texto

Descrição gerada automaticamente*

***Tipos de dados***

**Inteiros**

*int8, int16, int32, int64*

Além disso também podemos ter suas variações

*uint8, uint16* ...

Em que a diferença é somente que o int aceita números negativos e o uint não

E temos algumas equivalências como por exemplo

*rune = int32*

*byte = uint8*

OBS: Os números na frente da variável representam a quantidade de dígitos

que cada tipo vai conseguir aguardar

***Reais***

*float32 e float64 =*

A variável tipo float não podem ser declaradas sem a quantidade de bits estando escrita

**Escrita**

string = “ ”

Não existe variável char em Golang

**Bool**

Só pode ser declara como false ou true

**Error**

***var*** *nomeDaVariavel TipoDaVariavel = errors.New(“”)*

**Uma imagem contendo Texto

Descrição gerada automaticamente**

*errors* é o nome do pacote para usar as variáveis do tipo error

**Funções**

**Texto

Descrição gerada automaticamente**

Para criar funções com mais de um tipo

Texto

Descrição gerada automaticamente

Tela de celular com aplicativo aberto

Descrição gerada automaticamente com confiança média

Se eu não quiser utilizar os dois retornos da função eu posso substituir o tipo que eu não desejo por um \_

Como por exemplo:



**Retorno nomeado**

Se eu quiser nomear o nome dos retornos eu posso utilizar

Texto

O conteúdo gerado por IA pode estar incorreto.

Variaticas

Caso eu queira criar uma função que receba n parâmetros eu utilizoTexto

O conteúdo gerado por IA pode estar incorreto.

Você só pode ter um termo variatico por função, porém pode ter outros termos na mesma função desde que só recebam x termos

**Operadores**

**Aritméticos:**

**+ soma**

**- Subtração**

**\* Multiplicação**

**/ Divisão**

**% Resto de divisão**

**Atribuição**

*var variavel1 string = “alguma coisa”*

*ou*

*variavel1 := “alguma coisa”*

**Relacionais**

> (maior)

*>=* (maior igual)

< (menor)

<= (menor igual)

== (igual)

!= (diferente)

* *Todos os operadores relacionais retornam um true ou false*

**Lógicos**

&& E

|| OU

! Negação

**Unários**

++ (Incrementa em 1 o valor da variável)

+= x (Incrementa em x o valor da variável)

-= x (Decrementa em x o valor da variável)

**STRUCTS**

* Structs é o jeito que o Golang tem para fazer Orientação a objetos

Para criar um tipo struct você utiliza a seguinte sintaxe

***type*** *nomeDoTipo* ***struct {***

*}*

Exemplo: Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

**Herança**

* Em Go não temos herança e sim uma pseudo herança que podemos fazer

Texto

Descrição gerada automaticamente

Como todo **estudante** é uma **pessoa** você pode colocar como uma propriedade do **estudante**  o struct da **pessoa** sem escrever mais nada na frente e assim ele irá herdar todos os tipos de um tipo **pessoa**

**Arrays**

* Arrays nada mais é do que uma lista de algo

Para declarar um array você pode utilizar



Todo array tem um tamanho fixo desde a sua criação

**Slices**

* Slice é um array com tamanho variável

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Como um slice tem um tamanho variável para adicionar mais termos em um slice você pode utilizar o comando ***append()*** que irá adicionar o valor no slice e retornar um novo slice com esse valor adicionado

Interface gráfica do usuário, Texto, chat ou mensagem de texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

**Ponteiros**

Ponteiros servem para você ao invés de passar um valor para uma variável você passa diretamente o endereço na memória em que a variável é armazenada fazendo assim que em qualquer alteração da variável original seja passada para esse ponteiro

Para gerar um ponteiro você deve colocar um \* antes do valor da variável

EX:

Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente

No exemplo podemos ver que antes do int temos um **\*** para dizer ao compilador que deve se passar o endereço da memória da **variavel3**. Já depois do = podemos ver que temos um & para dizer ao compilador passar o endereço daquela variável especifica

Após isso qualquer alteração que ocorrer na **variavel3** a variável ponteiro também vai ser alterada

**Maps**

**Texto

O conteúdo gerado por IA pode estar incorreto.**

**Maps aninhados**

**Texto

O conteúdo gerado por IA pode estar incorreto.**

**Estruturas de controle**

**Texto

O conteúdo gerado por IA pode estar incorreto.**

Não tem diferença de outras linguagens tirando a parte de você pode criar uma variável dentro do próprio if conforme mostra a figura

**Switch case**

**Texto

O conteúdo gerado por IA pode estar incorreto.Texto

O conteúdo gerado por IA pode estar incorreto.**

Igual outras linguagens

**Texto

O conteúdo gerado por IA pode estar incorreto.Texto

O conteúdo gerado por IA pode estar incorreto.Texto

O conteúdo gerado por IA pode estar incorreto.**

Loops

Loops em Golang só existe for para fazer função de tudo

Texto

O conteúdo gerado por IA pode estar incorreto. = While

Tela preta com letras brancas

O conteúdo gerado por IA pode estar incorreto. = For convencional

Texto

O conteúdo gerado por IA pode estar incorreto.=

For each

Tela de celular com aplicativo aberto

O conteúdo gerado por IA pode estar incorreto. = For Range em um map

**Métodos**

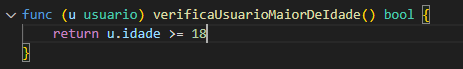
Um método seria uma função dentro de um tipo struct e que irá funcionar para todos os structs que você criar

EX:

Texto

O conteúdo gerado por IA pode estar incorreto.

No exemplo a func(u usuário) salvar(){} irá fazer parte e poderá ser chamada a partir de todos os structs criados com o tipo “usuário”



Um exemplo de um método com um retorno

Texto

O conteúdo gerado por IA pode estar incorreto.

Exemplo de método que utiliza ponteiro

**Interfaces**

Interface é uma forma facilitada e para você não ter que repetir tanto código de utilizar um mesmo método para determinados “Objetos” diferentes

Texto

O conteúdo gerado por IA pode estar incorreto.

Por exemplo aqui podemos ver uma interface chamada “forma” que irá funcionar para todos os “structs” que tiverem um método chamado area() com exatamente a mesma nomenclatura que está na interface

Abaixo podemos ver uma função que vai receber essa forma e irá passar o resultado printado na tela independente da forma que ela receber

O programa só irá permitir você chamar a função *escreverArea()* se o tipo struct tiver um método que atenda aos requisitos da interface que nesse caso seria um método com nome *area()* que não recebe parâmetros e retorna um float64

Texto

O conteúdo gerado por IA pode estar incorreto.

Nesse exemplo tanto o struct retângulo e o struct círculo possuem as funções *area()* que não recebem parâmetros e retornam um float64

Existe também as interfaces genéricas que servem basicamente para uma função ou um tipo receber qualquer tipo de variável

Ex: Texto

O conteúdo gerado por IA pode estar incorreto.

Nesse exemplo quando você chamar essa função ela vai imprimir qualquer coisa na tela independente do tipo da variável. Burlando a forte tipagem do Golang

A Funçõao *fmt.Println()* recebe uma interface variatica, ou seja, spode receber quantos tipos forem precisos e independente do valor da variavel.

Concorrencia

Concorrência permite que você execute dois ou mais comandos ao mesmo tempo e é um dos motivos de tantas empresas estarem começando a usar o Go

Goroutines

Faz com que o comando seja executado, mas não precise terminar a sua execução para executar o restante do código

EX: Tela preta com letras brancas

O conteúdo gerado por IA pode estar incorreto.

A função escrever(texto string){} iria ser executada infinitamente o que não permitiria o código avançar nunca e acabar naquela linha, porém quando adicionamos o *go* antes do nome da função nós dizemos para ele executar o restante do código mesmo que essa primeira chamada da função não tenha terminado

Wait Group

É uma forma de sincronizar as goroutines para conseguir executar todas ao mesmo tempo e aguardar todas acabarem para o programa se encerrar

* Então primeiro criamos uma variável do tipo *sync.WaitGroup*
* Utilizamos o comando *nomeDaVariável.Add(*quantidade de Goroutines)
* *Criamos uma função anônima com o comando go func(){}() e dentro dela chamamos a função que queremos executar*
* *Após a chamada da função que queremos executar adicionar o comando nomeDaVariável.Done() para ele diminuir o contador das goroutines*
* *Repetir o passo 3 e 4 para as próximas goroutines que você deseja executar*
* *No final do código utilizar o comando nomeDaVariável.Wait() que irá fazer o programa continuar até que o contador chegue a 0*

Exemplo: Tela de computador com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.

Canais

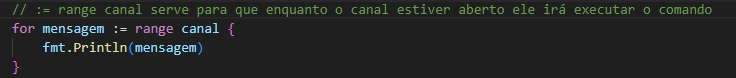
Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

Para substituir esse for infinito nós também podemos utilizar



Canal com buffer

Um canal com buffer é um canal com capacidade de receber mais de um dado



E assim podendo receber n tipo de valores diferentes

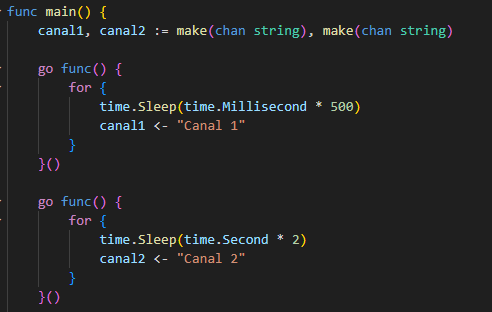
Texto

O conteúdo gerado por IA pode estar incorreto.

Select

Select é como um switch case mas que só é usado em canais

Ele é bem utilizado quando uma função acaba em um tempo diferente da outra, permitindo que a função execute assim que o canal estiver pronto para receber um dado

EX: 

Duas funções em que uma demora 0,5 segundos e a outra 2 segundos para terminarem sua execução ocorrendo infinitamente

Texto

O conteúdo gerado por IA pode estar incorreto.

E aqui como podemos utilizar o Select para executar as duas assim que elas estiverem prontas

Padrão Worker Pools

Um modo de executar funções recursivas mais rapido

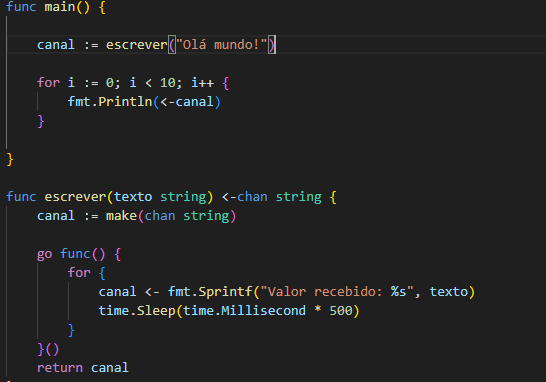
Texto

O conteúdo gerado por IA pode estar incorreto.Texto

O conteúdo gerado por IA pode estar incorreto.

Padrão Generator

Uma forma de esconder toda a complexidade dessas goloutines



Multiplexador

Serve para juntar dois canais em 1 só