Sintaxe e Semântica das Linguagens de Programação
Linguagem GO

Nome do Arthur[1], **Nome Uriel**[1], **Vinícius Giovanini**[1]

[1]Instituto de Ciências Exatas e Informatica
Pontifícia Universidade Católica de Minas Gerais (PUCMG)
Belo Horizonte – MG – Brazil

{aosgomes,uandrade,vgiovanini}@sga.pucminas.br

Resumo. *Tal documentação descreve a linguagem de programação GO, com várias informações sobre como ela foi desenvolvida, e suas diferenças para linguagens tradicionais C-Like. Dessa forma aprofundando na história, conceito e criação da linguagem do Gopher, com o objetivo do entendimento de como se deve utiliza-la da melhor maneira, e para descobrir quais são as melhoras aplicações que ela se encaixa.*

1. Links

2. Domínio de Aplicação

A linguagem de programação GO, foi criada pelo Google em 2007, porém somente disponibilizada em código aberto em 2009, desde então ela ganhou muita popularidade, enquanto Java e C continuam dominando a programação ela está dominando a programação moderna. O uso do Go tem aumentado pois, é uma linguagem de código aberto e é utilizada em diversos campos, como em micros serviços, que tem como desenvolvimento o Docker, e umas das principais áreas que está usando GOLANG consiste na criação de sistemas nativo da nuvem, que utiliza go por ter um fácil entendimento, uma grande segurança, e principalmente uma alta performance.

Figure 1. Logo e Mascote (Gopher) da linguagem GO

3. Tipos de dados e Sistemas de Tipos

3.1. Tipagem e Tipos Numéricos

A linguagem go é uma LP de tipagem estática, isso consiste que quando se declara alguma variável, é necessário especificar seu tipo. Sua padronização de declaração inicialmente parece com as linguagens de tipagem dinâmica, visto que declara-se antes do

nome da variável um elemento VAR, ou seja, não tem um tipo primitivo especificado na declaração, porém o que faz de fato o GO ser de tipagem estática, consiste em ser feito uma verificação em tempo de compilador, dessa maneira o compilador verifica os tipos usado em varáveis para garantir que o dado atribuido é um tipo compatível com sua declaração.

Outro ponto observado, consta que como na linguagem Java, além do modelo booleano que se declara através de um bit o valor true e false, é possível declarar variáveis numéricas com diferentes tipos de tamanho de memória, porém no GO está presente um tipo numérico, além de todos os comuns encontrados nas linguagens C-Like, não encontrado nas mesmas, que é o tipo **complexo**, na qual engloba números do conjunto complexo, representados na linguagem de programação go por dois números Float.

DATA TYPE	DESCRIPTION
int8	8-bit signed integer
int16	16-bit signed integer
int32	32-bit signed integer
int64	64-bit signed integer
uint8	8-bit unsigned integer
uint16	16-bit unsigned integer
uint32	32-bit unsigned integer
uint64	64-bit unsigned integer
int	Both in and uint contain same size, either 32 or 64 bit.
uint	Both in and uint contain same size, either 32 or 64 bit.
rune	It is a synonym of int32 and also represent Unicode code points.
byte	It is a synonym of int8 .
uintptr	It is an unsigned integer type. Its width is not defined, but its can hold all the bits of a pointer value.

Figure 2. Tipos de dados numéricos

3.2. Tipo String

Diferente de C, GO tem o tipo String, que tem como objetivo tratar dos caracteres e seus conjuntos, na qual é uma sequência de bytes, porém em GO essa sequência de caracteres são **imutáveis**, dessa forma as String foram feitas somente para leitura, ocorrendo um erro de atribuição caso ocorra a tentativa de alteração das mesmas.

4. Alocação de Memória

Na linguagem GO, é presente diversas ferramentas para facilitar e otimizar a alocação e liberação da memória, entre elas está presente o **Garbage Collector** e o **Escape Analysis**. O primeiro respectivamente tem a função de desalocar automaticamente a memória quando se encerra o uso da mesma, comparando o uso dessa funcionalidade em GO com outras linguagens, na qual detêm certas desvantagens, ocorridas pois é comum detectar alguns consumos excessivos, além de possíveis travamentos durante a execução do programa, por conta disso, em Go o Garbage Collector foi desenvolvido embutido no core, assim ele está otimizado para recursos da própria linguagem, tornando o gerenciamento de memória mais rápido em comparação a outras linguagens que utilizam da mesma funcionalidade. O escape analysis é uma ferramenta que compõe a alocação, ela é utilizada para analisar se a variável deve ser alocada no Heap ou na Stack.

Outro ponto importante na alocação de memória, equivale ao estabelecimento de um modelo de segurança para evitar que seja lido uma memória utilizada por outro processo, dessa forma o Go utiliza uma memória virtual que cria o Page Table, para alocar nessa ”página” somente parte da memória que está disponível, evitando usar assim a memória física.

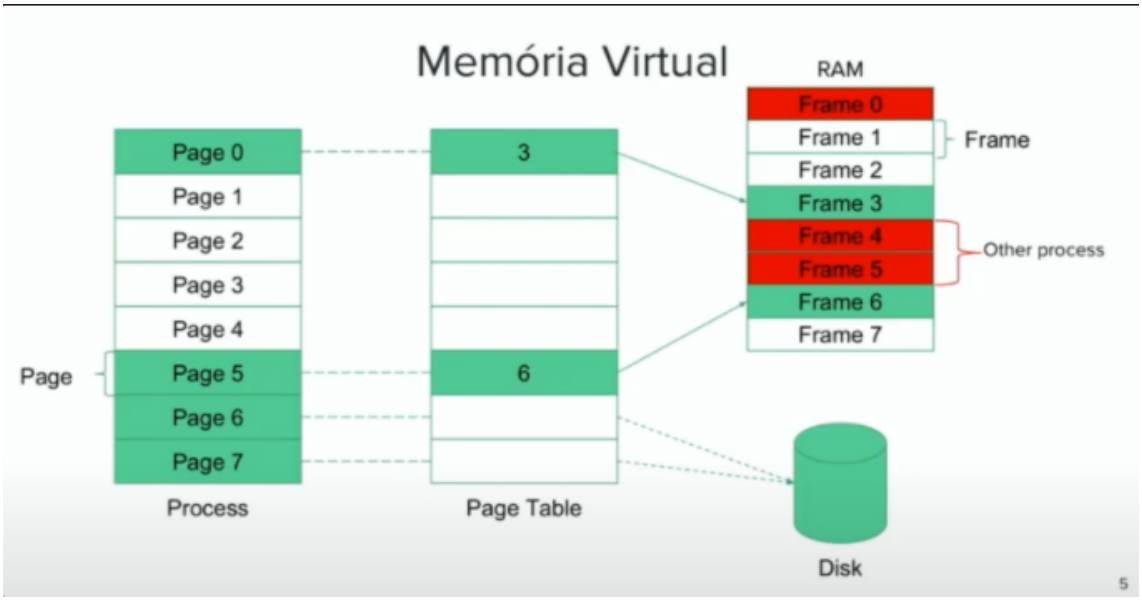

Figure 3. Representação da alocação com Memória Virtual. [InfoQBrasil 2022]

5. Instruções de Controle

Em relação as instruções de controle, o Go é bem parecido com as estruturas presente nas linguagens C-Like, porém com a perdas de algumas formas de controle. Para realizar uma repetição em Go é utilizado somente o **For**, dessa maneira o while não existe nesse contexto, assim utilizando for para qualquer tipo de repetição que for implementada. Comparando as estruturas de comparação se mantém iguais ao C-Like, contendo o IF e o Switch Case.

A linguagem Go tem o escopo estático, isso ocorre pois uma variável declarada dentro de uma estrutura ou um bloco em Go, só pode ser acessada por elementos dentro desse escopo, jamais será permitido acessar uma varável inicializada dentro de um bloco fora do mesmo.

6. Sintaxe e Representatividade

A sintaxe da linguagem Go é muito parecida com a C, suas declarações são baseadas em Pascal, porém existe algumas diferenças entre o C e o Go, como a declaração de tipos, ausência de parênteses em estruturas como For e IF, Go possui coletor de lixo, na qual C não contém. Algumas funcionalidades ausentes são o tratamento de exceção, herança, sobrecarga entre outras.

Quando o Golang estava sendo planejado, seus criadores procuraram criar uma linguagem estaticamente tipada, que fosse fácil de usar quanto comparado com linguagens dinamicamente tipadas, e conter um nível de expressividade similar. O Google queria combinar características de três linguagens diferentes, todas juntas para formar o Golang, eram elas: o C++, Java e o Phyton, pegando seus melhores recursos para implementar na sua própria LP, dessa forma Go tem elementos de tipagem estática e dinâmica.

Go	Python
• Go paradigms are Procedural, functional and concurrent language.	• Python paradigms are object-oriented, imperative, functional, and procedural language.
• Statically typed language	• Dynamically typed language
• More focused on being a system language.	• More focused on writing web applications.
• Go syntax is based on the opening and closing braces.	• Python syntax uses indentation to indicate code blocks.
• Go doesn't have classes and objects.	• Python has classes and objects.
• Offers limited support for Object Orientation and functional concepts.	• Object-Oriented programming, which supports functional concepts.
• Go does not provide exceptions.	• Python supports exceptions.
• Go does not support inheritance.	• Python supports inheritance.
• Go needs more code to perform the same number of actions.	• Python needs fewer code compares to Go.
• 28.5 K Github stars.	• 67.5 K Github stars.

Figure 4. Comparação entre Golang e Phyton

7. Recursividade

A linguagem Go suporta funções recursivas, isso consiste que o Go consegue declarar uma função que retorna uma chamada para si mesma, também podendo retornar uma expressão de função contendo sua própria chamada, mudando valores das variáveis passadas como parâmetro, fazendo assim um loop infinito, que só irá acabar através de uma condição de parada, na qual será executada a cada chamada, e caso essa condição de chamada teste **True**, ele entra no bloco e retorna um valor, acabando assim com a repetição, um exemplo comum de função recursiva é o fatorial.

8. Concorrência

A concorrência consiste na capacidade de um programa executar várias partes simultaneamente, dessa forma evitando a execução sequencial, na qual iria ser um método menos otimizado. A concorrência é muito importante nos softwares atuais, pois a cada dia é preciso mais rapidez nas operações de um programa computacional.

Na linguagem Go, essa concorrência é aplicada através de uma função **Goroutine**, que é uma função capaz de ser executada ao mesmo tempo de outras, quando uma função é criada, ela é executada em um processador lógico que está disponível, e a Golang tem o chamado *"Runtime Scheduler"*, com o objetivo de gerenciar todas as funções Goroutine que são criadas e precisam de tempo de processador, após isso o chamado *"Scheduler"*, ligando as threads do sistema operacional aos processadores lógicos para executar as funções, assim controlando todo o processo de execução em conjunto.

Linguagens de programação como Java e Phyton, impementam a concorrência usando threads, já Golang tem construções baseadas em goroutines e canais. Canais consiste em um canal pipeline para enviar e receber dados, é através dos canais que os goroutines recebem dados de outras funções.

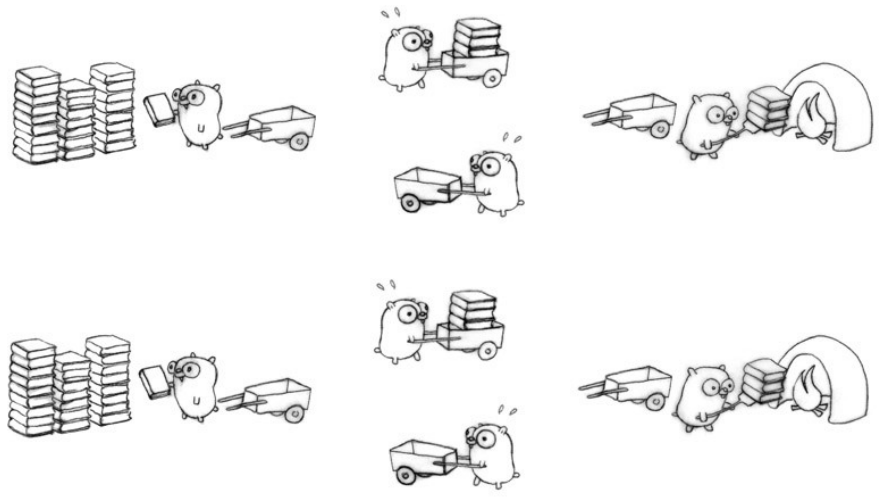

Figure 5. Representação da Concorrência, com os Gophers realizando as tarefas simultaneamente.

9. Paradigmas de Programação da Linguagem Go

Os autores da linguagem Go, descreveram-na como ”uma linguagem de programação multi-paradigma que é orientada para as necessidades de programação segura, qualidade de produção e aplicações baseadas em agentes, fortemente tipado e de ordem superior, tem relação, funções e definições de procedimentos de ação”. Os criadores também propõem que a linguagem é de alta qualidade em programação lógica, funcional e imperativa.

Dessa maneira, considerando que os criadores utilizaram as melhores características de outras linguagens para a sua criação, essa LP ficou com construções de distintos paradigmas computacionais, sendo descrita com **multi-paradigma**, porém mesmo não sendo uma linguagem funcional, Golang tem muitos princípios desse paradigma, permitindo assim auxiliar no desenvolvimento do código. O paradigma procedural também está muito presente, logo que a linguagem conta com variáveis locais, chamadas recursivas entre outros fatores característicos desse paradigma.

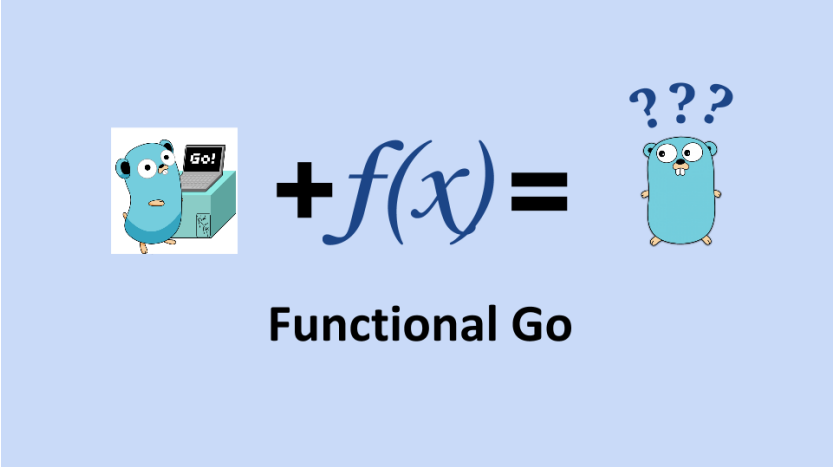

Figure 6. Representação dos multi-paradigmas. [GoLangPrograms 2022c]

10. Referências

Para a pesquisa de domínio de aplicação fora utilizados [GeekH 2022a] e [GeekH 2022b], para tipos de dados e sistemas de tipos as fontes foram [Acervolima 2022] e [GeekForGeeks 2022], para alocação de memória foi utilizado como fonte um vídeo do youtube [InfoQBrasil 2022], instruções de controle foi usado [Alura 2022] e [Magnum 2022], sintaxe e representatividade foi utilizado [CodLime 2022] e [Wikipedia 2022], a parte de recursividade foi baseada em [GoLangPrograms 2022b], concorrência foi baseado em [GoLangPrograms 2022a], e paradigmas de programação teve como fonte [GoLangPrograms 2022c], além de também ser utilizado no trabalho [GoLang 2022].

References

Acervolima (2022). Acervolima:tipos primitivos da linguagem go. `https://acervolima.com/strings-in-golang/`. Accessed: 2022-03-29.

Alura (2022). Alura:instruções de controle da linguagem go. https://www.alura.com.br/artigos/variaves-com-go-lang. Accessed: 2022-03-29.

CodLime (2022). Sintaxe e expressividade da linguagem go. https://codilime.com/blog/why-golang-may-be-a-good-choice-for-your-project/#:~:text=Go%20(or%20Golang)%20is%20an,high%20performance%2C%20expressiveness%20and%20readability.. Accessed: 2022-04-04.

GeekForGeeks (2022). Geeksforgeeks:tipos primitivos da linguagem go. https://www.geeksforgeeks.org/data-types-in-go/. Accessed: 2022-03-29.

GeekH (2022a). Webs site de dominio de aplicação da linguagem go. https://blog.geekhunter.com.br/golang/.

GeekH (2022b). Webs site de dominio de aplicação da linguagem go. {https://coodesh.com/blog/candidates/carreiras/go-lang-developers/}. Accessed: 2022-03-29.

GoLang (2022). Documentação da linguagem go. https://go.dev/doc/. Accessed: 2022-03-29.

GoLangPrograms (2022a). Concorrência da linguagem go. https://www.golangprograms.com/go-language/concurrency.html. Accessed: 2022-04-04.

GoLangPrograms (2022b). Funções recursivas da linguagem go. http://goporexemplo.golangbr.org/recursion.html. Accessed: 2022-03-29.

GoLangPrograms (2022c). Paradigmas da linguagem go. https://medium.com/@geisonfgfg/functional-go-bc116f4c96a4. Accessed: 2022-04-04.

InfoQBrasil (2022). Alocação golang. https://www.youtube.com/watch?v=Qc-tgDkU7VI&ab_channel=InfoQBrasil. Accessed: 2022-03-29.

Magnum, L. (2022). Medium:instruções de controle. https://lucasmagnum.medium.com/iniciando-em-go-estruturas-de-controle-265256ec73b4. Accessed: 2022-03-29.

Wikipedia (2022). Sintaxe e expressividade da linguagem go. https://en.wikipedia.org/wiki/Go!_(programming_language). Accessed: 2022-04-04.