UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Variáveis

QXD0001 - Fundamentos de Programação

Prof. Bruno Góis Mateus (brunomateus@ufc.br)

Agenda

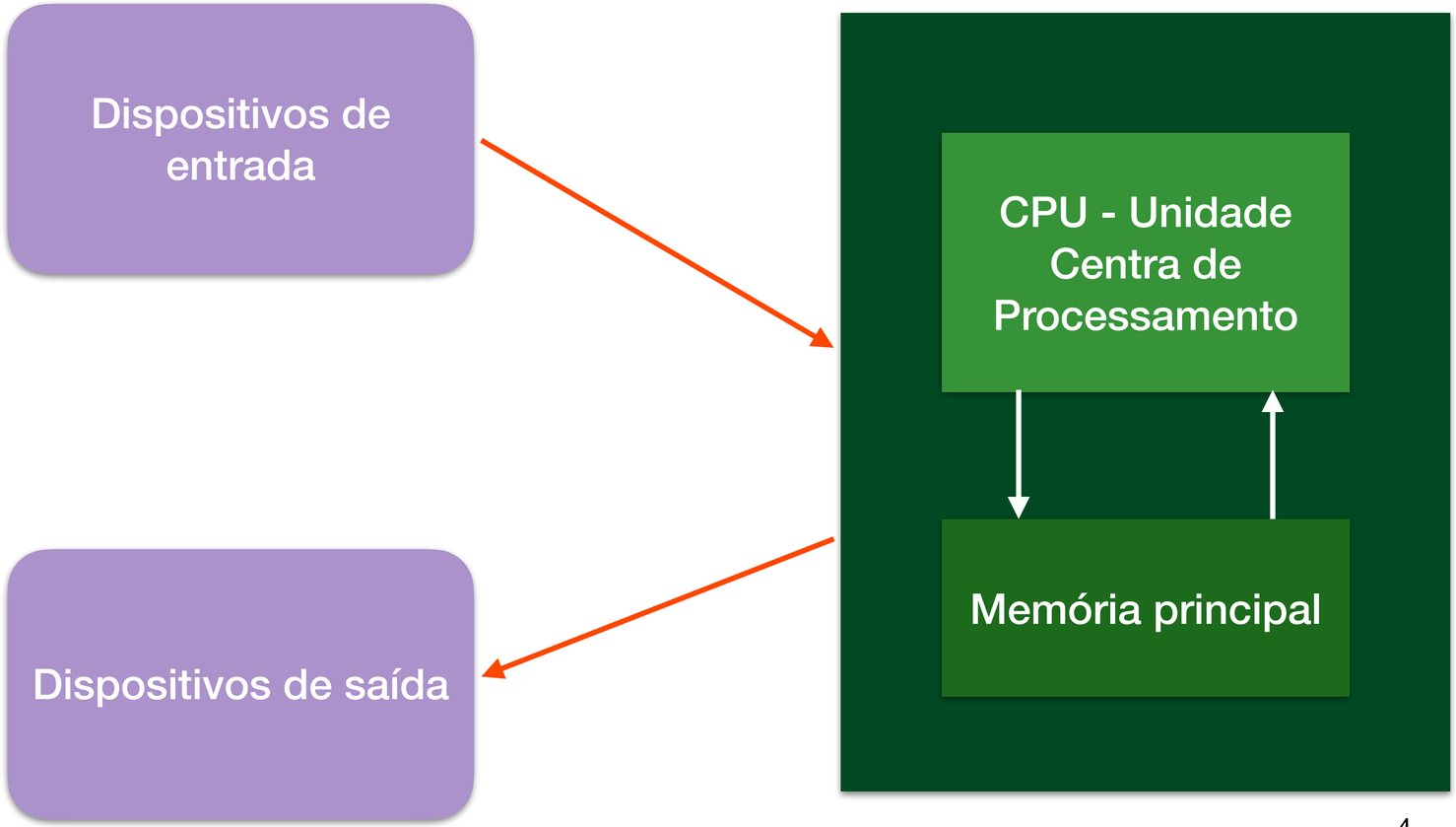
- Introdução
- A memória
- Variáveis
- Entrada e Saída
- 10 erros clássicos com variáveis em Go

Introdução

Introdução

Entrada, Processamento e Saída

- Um programa de computador basicamente faz três coisas:
  - Recebe dados (entrada)
    - Teclado/Arquivo
  - Processa os dados
    - CPU
  - Produz resultados (saída)
    - Monitor/Arquivo



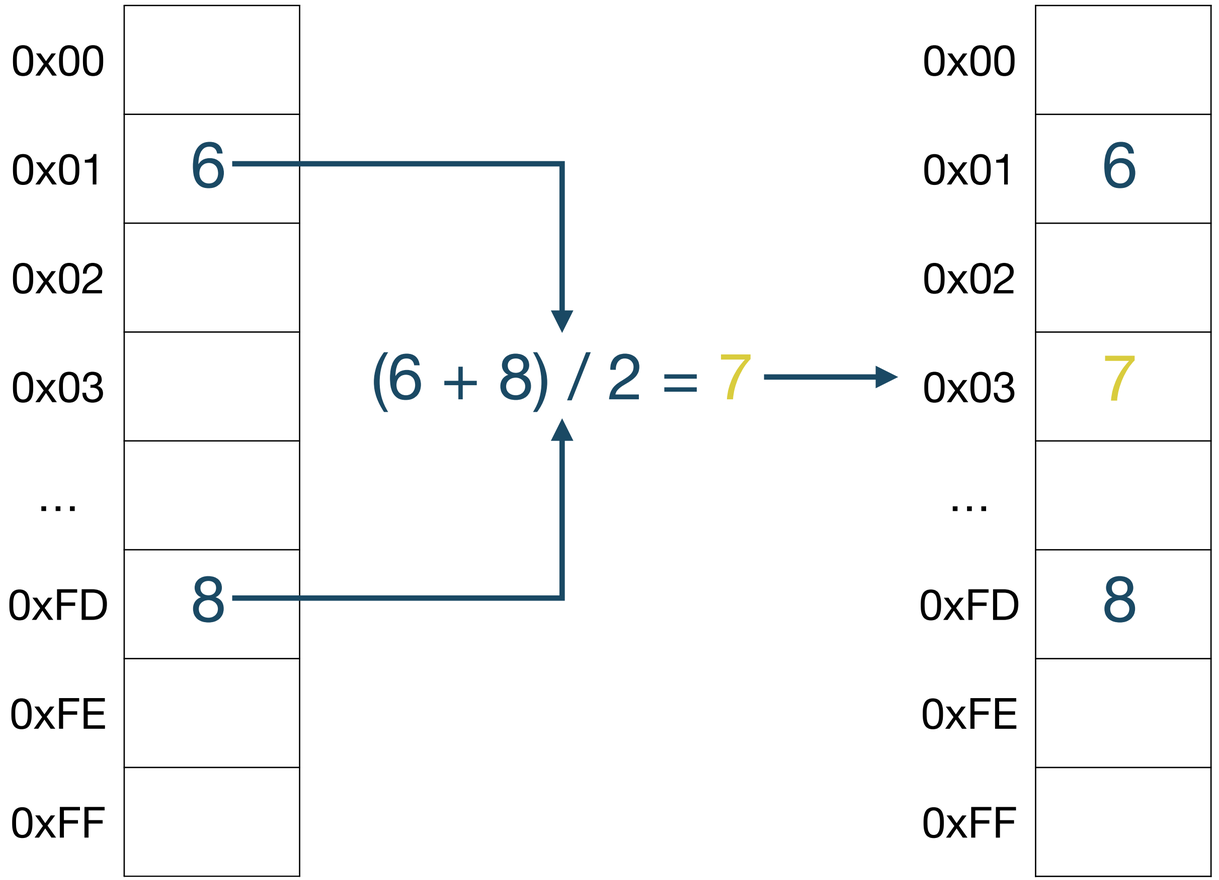
Onde os dados ficam "parados" durante o processamento ?

A memória

A memória

Operações na memória

- Consultar células de memória
- Realizar operações
- Armazenar novo valor em uma célula



A memória

O problema

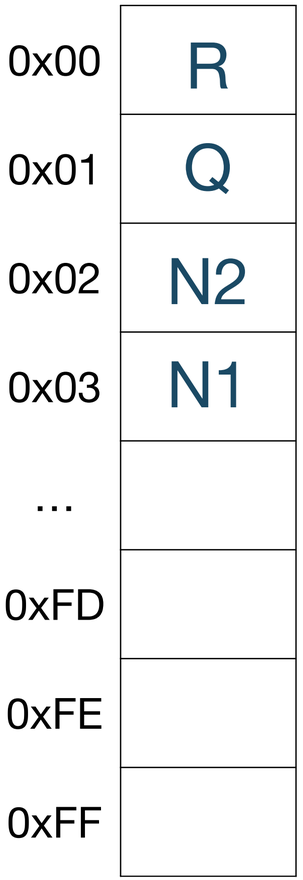
- Seria muito difícil programar se precisássemos escrever programas assim:

```
pegar valor do endereço 0x01
somar com valor do endereço 0xFD
guardar resultado no endereço 0x03
```

A memória

Ex: Máximo divisor comum - Algoritmo de Euclides

1. Leia um número e escreva na célula 0x03
2. Leia um número e escreva na célula 0x02
3. Divida o valor da célula 0x03 pelo valor da célula 0x02.
4. Guarde o quociente na célula 0x01 e o resto na célula 0x00.
5. Se o valor da célula 0x00 for 0 , então mostre o valor da célula 0x02 e PARE.
6. Escreva na célula 0x03 o valor da célula 0x02.
7. Escreva na célula 0x02 o valor da 0x00.
8. Retorne ao passo 3.



A memória

O problema

- Problemas:
  - Complexidade desnecessária no algoritmo
  - Difícil manutenção do código
  - Impossível prever células livres
  - Interferência por execução simultânea

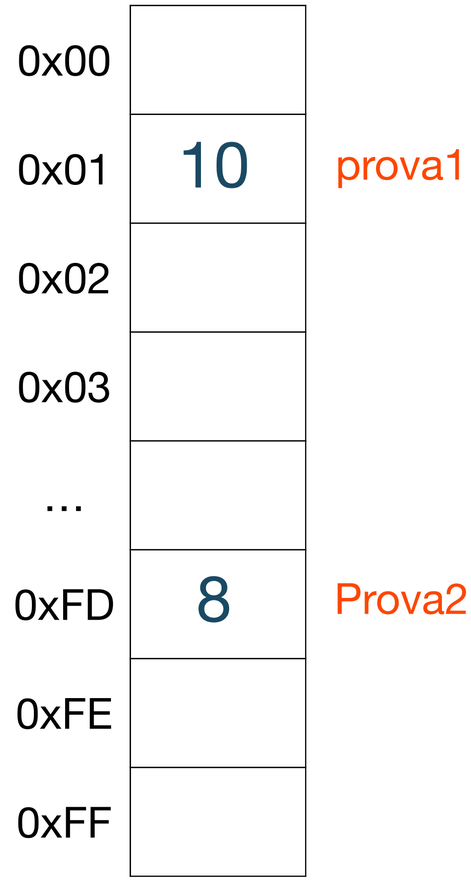
Para resolver esse problema usamos variáveis 👌🏻

Variável

Variável

O que é?

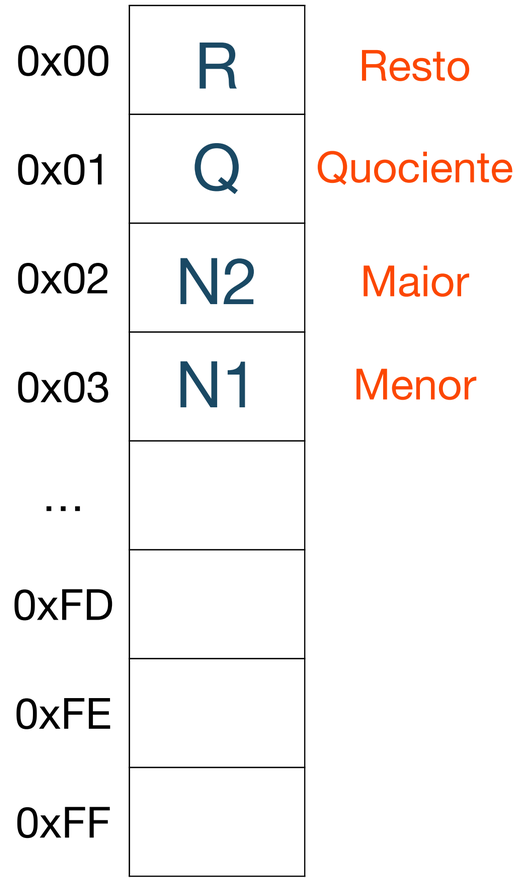
- É uma abstração dos endereços de memória
- É um apelido (label) para um endereço de memória
- Em vez de dizer "Guarde 10 no endereço 0x001", dizemos prova1 = 10.



Variável

Ex: Máximo divisor comum - Algoritmo de Euclides

1. Leia um número e escreva na menor
2. Leia um número e escreva na maior
3. Divida o valor da variável maior pelo valor da variável menor
4. Guarde o quociente na variável quociente e o resto na variável resto
5. Se o valor da variável resto for 0 , então mostre o valor da variável menor e PARE.
6. Copie o valor da variável menor para a variável maior
7. Copie o valor da variável resto para a variável menor
8. Retorne ao passo 3.



Variável

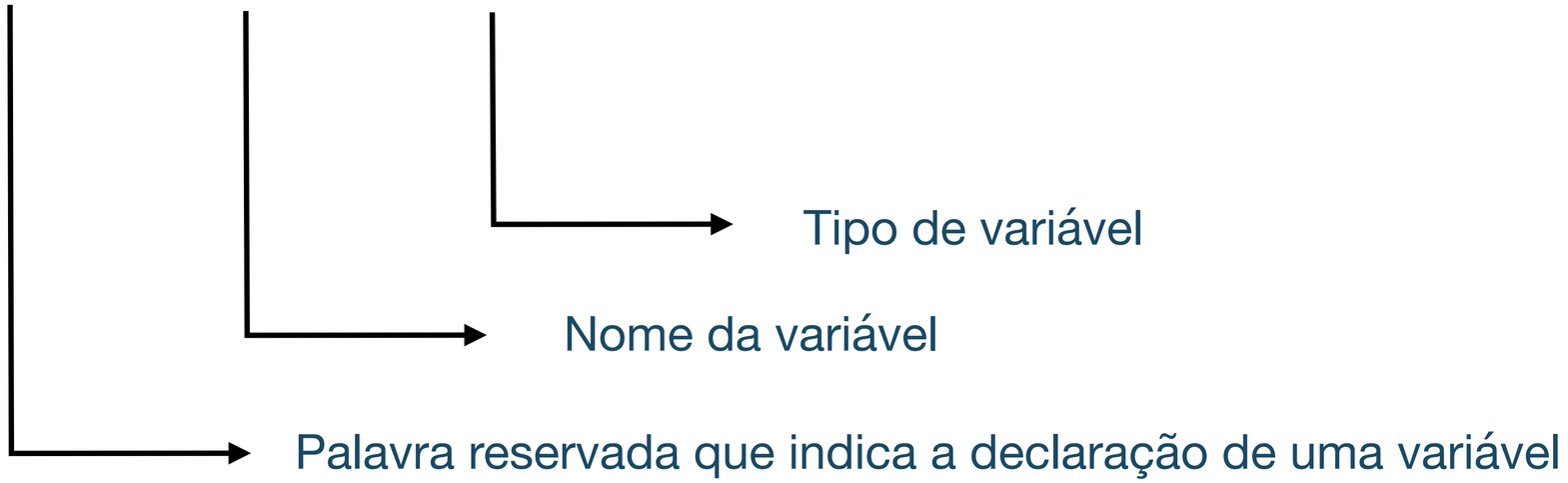
A Importância da Nomenclatura

- Código é lido por humanos.
- Nomes ruins:
  - a, b, temp, x1
- Nomes bons:
  - precoProduto, estoqueMinimo, usuarioLogado

Variável

Declarando uma variável em GO

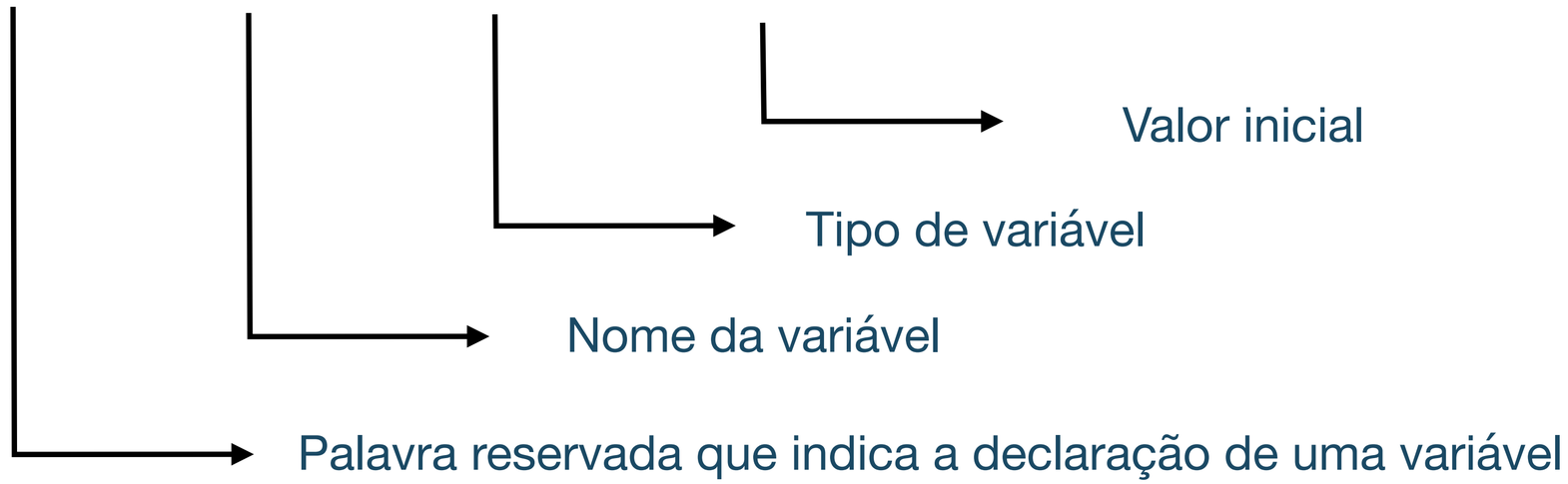
```
var idade int
```



Variável

Declarando e inicializando uma variável em GO

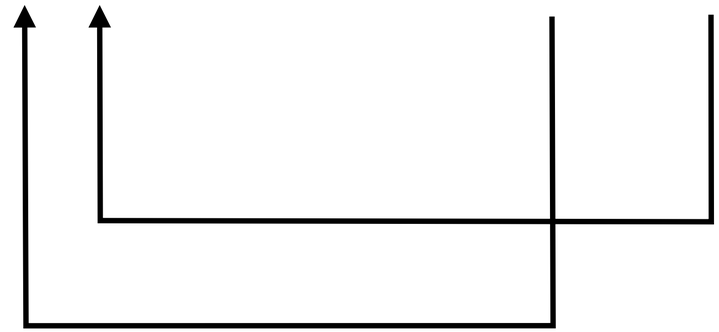
```
var idade int = 20
```



Variável

Declarando e inicializando mais de uma variável GO

```
var y, z float32
var a, b float32 = 1.0, 3.3
```



Variável

Tipos de dados

Tipo	Exemplo
int	10
float64	3.14
string	"Olá"
bool	true

Variável

Tipos de dados com exemplos

```
var idade  int = 20
var altura  float32 = 1.75
var nome string = "Ana"
var aprovado bool = true
```

❗ Cada variável declarada deve ser usada pelo menos uma vez ❗

Variável

Inteiros

Tipo	Descrição	Intervalo
int8	8-bit inteiro com sinal	(-128 até 127)
int16	16-bit inteiro com sinal	(-32768 até 32767)
int32	32-bit inteiro com sinal	(-2147483648 até 2147483647)
int64	64-bit inteiro com sinal	(-9223372036854775808 até 9223372036854775807)
uint8	8-bit inteiro sem sinal	(0 até 255)
uint16	16-bit inteiro sem sinal	(0 até 65535)
uint32	32-bit inteiro sem sinal	(0 até 4294967295)
uint64	64-bit inteiro sem sinal	(0 até 18446744073709551615)
int	32 ou 64 bit	
uint	32 ou 64 bit	
byte	It is a synonym of uint8.	

Variável

Ponto flutuante e números complexos

Tipo	Descrição	Exemplo
float32	32-bit IEEE 754 floating-point number	3.14159
float64	64-bit IEEE 754 floating-point numbe	2.718281828459045
complex64	Número complexo com a parte real e imaginária sendo um float32	complex(1.0, 2.0)
complex128	Número complexo com a parte real e imaginária sendo um float64	complex(3.0, 4.0)

Variável

Usando o tipo complexo

```
package main

import "fmt"

func main() {
    var comp1 = 13 + 33i
    fmt.Println("Complex number 1 is :", comp1)
    realNum := real(comp1)
    fmt.Println("Real part of complex number 1:", realNum)
    imaginary := imag(comp2)
    fmt.Println("Imaginary part of complex number 2:", imaginary)

}
```

Variável

Operações básicas

Tipo	Exemplo	Requisito
+	Adição	Os dois operandos devem ser do mesmo tipo numerico.
-	Subtração	
*	Multiplicação	
/	Divisão	
%	Resto da divisão	

Variável

Operações básicas

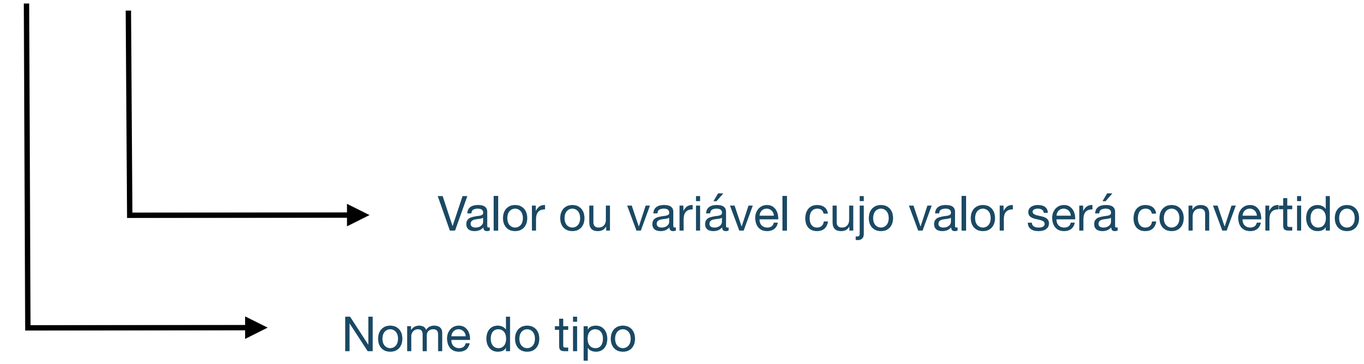
```
func main() {
   var i int = 10
   var f float32 = 3.3
   var j float32 = f - I

   var x int16 = 49
   var y int8 = 8
   var z int16 = x + y


}
```

Variável

Ás vezes precisamos converter os tipos

- Go não faz conversão automática como outras linguagens
- Sintaxe:

```
T(v)
```



Variável

Ás vezes precisamos converter os tipos

```
import "fmt"

func main() {
   var i int = 42
   var f float64 = float64(i)
   var u uint = uint(f)
}
```

Go é estaticamente tipado: Uma vez que uma variável não pode mudar de tipo.

Entrada e Saída

Entrada e Saída

Saída

- Existem três funções para imprimir dados no terminal, stdout (saída padrão)
- Todas providas pelo pacote fmt

```
func Print(a ...any)

func Println(a ...any)

func Printf(format string, a ...any)
```

Entrada e Saída

fmt.Print

- Print imprime texto sem adicionar quebra de linha no final

```
package main

import "fmt"

func main() {
    fmt.Print("Olá ")
    fmt.Print("mundo!")
}
```

Entrada e Saída

fmt.Println

- Println significa Print Line
  - imprime o conteúdo
  - adiciona quebra de linha automaticamente

```
func main() {
    fmt.Println("Olá ")
    fmt.Println("mundo!")
}
```

Entrada e Saída

fmt.Printf

- Printf permite formatar a saída.
  - Ele usa formatadores (placeholders)

```
import "fmt"

func main() {
  var nome string = "Maria"
  var idade  int = 20

  fmt.Printf("Nome: %s\n", nome)
  fmt.Printf("Idade: %d\n", idade)
}
```

Entrada e Saída

fmt.Printf

Formatador	Significado
%d	inteiro
%f	número decimal
%s	string
%t	boolean

Entrada e Saída

fmt.Printf

- Printf permite formatar a saída.
  - Ele usa formatadores (placeholders)

```
var nome string = "João"
var idade int = 22
var altura float32 = 1.75

fmt.Printf("Nome: %s\n", nome)
fmt.Printf("Idade: %d anos\n", idade)
fmt.Printf("Altura: %.2f m\n", altura)
```

Entrada e Saída

Comparação

Função	Quebra de linha	Formatação
Print	❌	❌
Println	✔	❌
Printf	❌	✔

Entrada e Saída

Entrada

- Existem três funções para imprimir dados no terminal, stdout (saída padrão)
- Todas providas pelo pacote fmt

```
func Scan(a ...any)

func Scanln(a ...any)

func Scanf(format string, a ...any)
```

Entrada e Saída

fmt.Scan

- Lê o que o usuário digita na ordem em que as variáveis aparecem
  - Separa os valores por espaços em branco ou quebras de linha.
  - Ignora espaços extras e pula para a próxima linha se necessário até encontrar um valor que combine com o tipo da variável

```
var nome string
var idade int
fmt.Print("Digite seu nome e idade: ")
fmt.Scan(&nome, &idade)
// O usuário pode digitar: "Ana 30" ou "Ana [ENTER] 30"
```

Entrada e Saída

fmt.Scanln

- Funciona de forma idêntica ao Scan, mas com uma regra de parada estrita:
  - Para de ler assim que encontra uma quebra de linha (tecla Enter)
  - Diferença Crucial: Se você pedir duas variáveis e o usuário digitar apenas uma e der Enter, o Scanln encerra a leitura ali mesmo, deixando a segunda variável vazia/zero

```
var nome string
var idade int
fmt.Print("Digite seu nome e idade: ")
fmt.Scanln(&nome, &idade)
// O usuário pode digitar: “João 30"
```

Entrada e Saída

fmt.Scanf

- O Scanf é a versão “cirúrgica".
  - Permite a definação de um formato esperado para a entrada
  - Usa-se os mesmos "verbos" do Printf (%d, %s, %f)
  - Só aceita a entrada se ela seguir exatamente o formato definido na string

```
var dia, mes, ano int
fmt.Print("Digite a data (DD/MM/AAAA): ")
fmt.Scanf("%d/%d/%d", &dia, &mes, &ano)
// Se o usuário digitar "28/03/2026", ele separa os
números pelas barras automaticamente.
```

10 erros clássicos com variáveis em Go

10 erros clássicos com variáveis em Go

1 — Declarar variável e nunca usar

```go
var idade int = 20
// Se a variável não for utilizada
// declared and not used
```

10 erros clássicos com variáveis em Go

2 — Usar variável antes de declarar

```
fmt.Println(nome)
var nome string = "Ana"
// undefined: nome
```

10 erros clássicos com variáveis em Go

3 — Confundir = com :=

```go
var idade int
idade := 20
// Isso gera erro porque := declara nova variável.
```

10 erros clássicos com variáveis em Go

4 — Misturar tipos incompatíveis

```go
var idade int = 20
var altura float64 = 1.75
resultado := idade + altura
// invalid operation
```

10 erros clássicos com variáveis em Go

5 — Esquecer o operador & no Scanln

```
var idade int
fmt.Scanln(idade)
// Correto abaixo
// Precisamos passar o endereço da variável.
fmt.Scanln(&idade)
```

10 erros clássicos com variáveis em Go

6 — Nomes de variáveis ruins

```go
// Exemplo ruim
a := 10
b := 20
c := a + b
// Melhor
preco := 10
quantidade := 20
total := preco * quantidade
```

10 erros clássicos com variáveis em Go

7 — Confundir maiúsculas e minúsculas

```
// Go é case-sensitve
idade := 20
fmt.Println(Idade)
// Erro
// undefined: Idade
```

10 erros clássicos com variáveis em Go

8 — Usar palavra reservada como variável

```
var type int
//type é palavra reservada.
```

10 erros clássicos com variáveis em Go

9 — Achar que int / int gera decimal

```
a := 5
b := 2
fmt.Println(a / b)
// Resultado: 2
```

10 erros clássicos com variáveis em Go

10 — Achar que variável guarda "expressão"

```
a := 10
b := a
a = 20
fmt.Println(b)
//Resultado:
10
// Porque b guarda o valor, não a expressão.
```

Referências

- Go 101
- Data Types in Go