UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Estruturas de controle

QXD0001 - Fundamentos de Programação

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

Agenda

- Introdução
- Expressões booleanas
- if / else
- switch

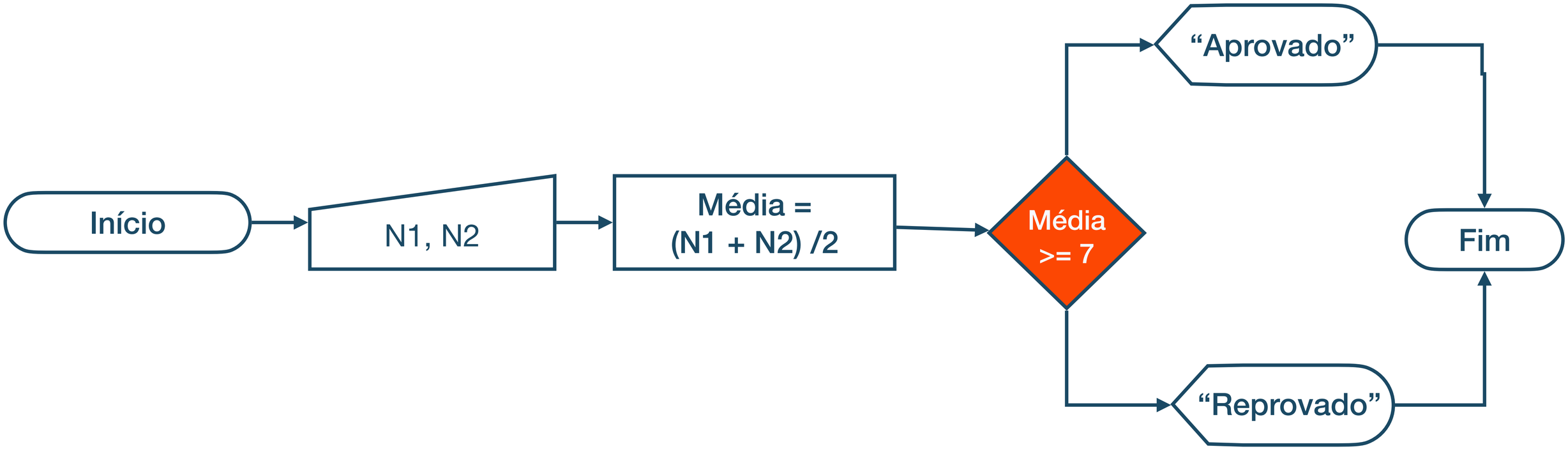
Introdução

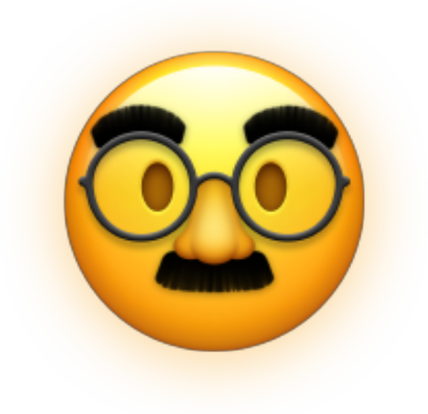
Introdução

- Com que sabemos até agora, construímos programas que sempre “fazem a mesma coisa”
  - A instruções sempre segue a mesma sequência de execução
- Em algumas situações, precisamos executar caminha alternativos
  - Baseado em dada condição
- Para isso, programas precisam:
  - comparar valores e tomar decisões

Introdução

Algoritmo para: Determinar se o aluno foi aprovado





Alguma coisa estabelecida ou combinada como requisito para que outra coisa seja feita ou entre em vigor; estipulação.

Significado de Condição segundo o Dicionário Michaelis

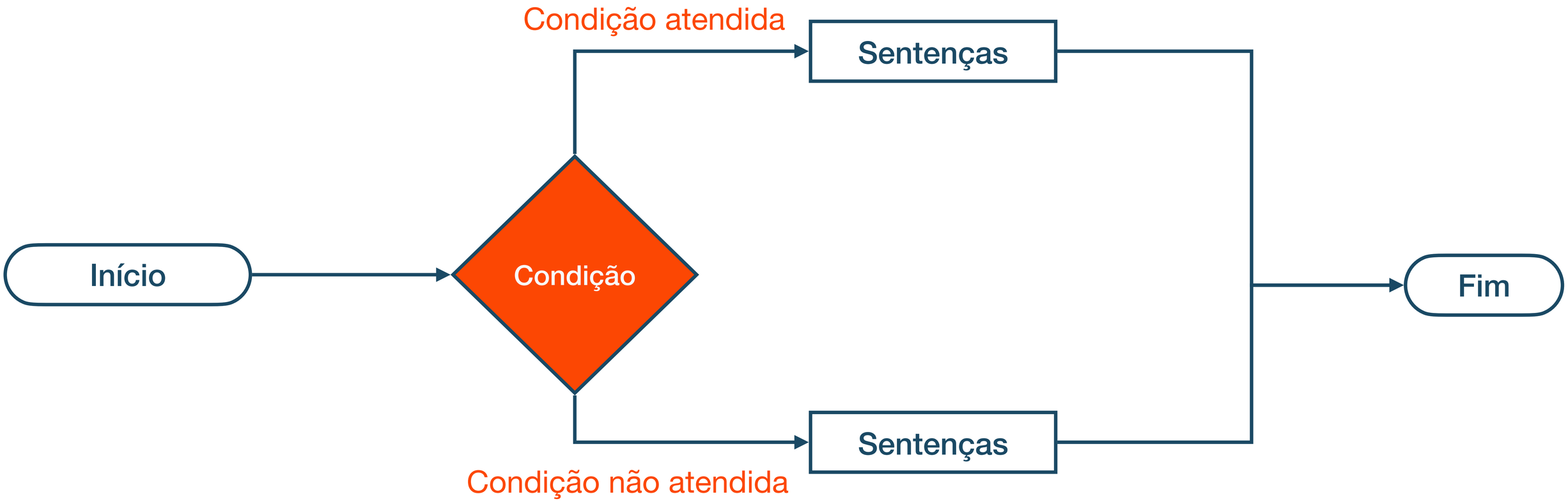
Se $idade < 18$

Senão

Fonte: dw.com

Introdução

Generalizando



Em linguagens de programação definimos as condições usando **expressões booleanas**

Expressões booleanas

Expressões booleanos

O tipo bool

- Em Go, utilizamos o tipo de dados bool para representar os resultados de condições
  - Só existem apenas dois valores possíveis
    - true
    - false
- Por sua vez, as condições são representadas por expressões booleanos

Expressões booleanos

O tipo bool

- Uma expressão booleana representa uma pergunta com duas possíveis resposta:
  - sim (true) ou não (false)
  - Sua avaliação resulta em um valor booleano
- São compostas por um ou mais operadores relacionais

Expressões booleanos

Operadores relacionais

Operador	Significado
==	igual
!=	diferente
>	maior
<	menor
>=	maior ou igual
<=	menor ou igual

Expressões booleanos

Operadores lógicos

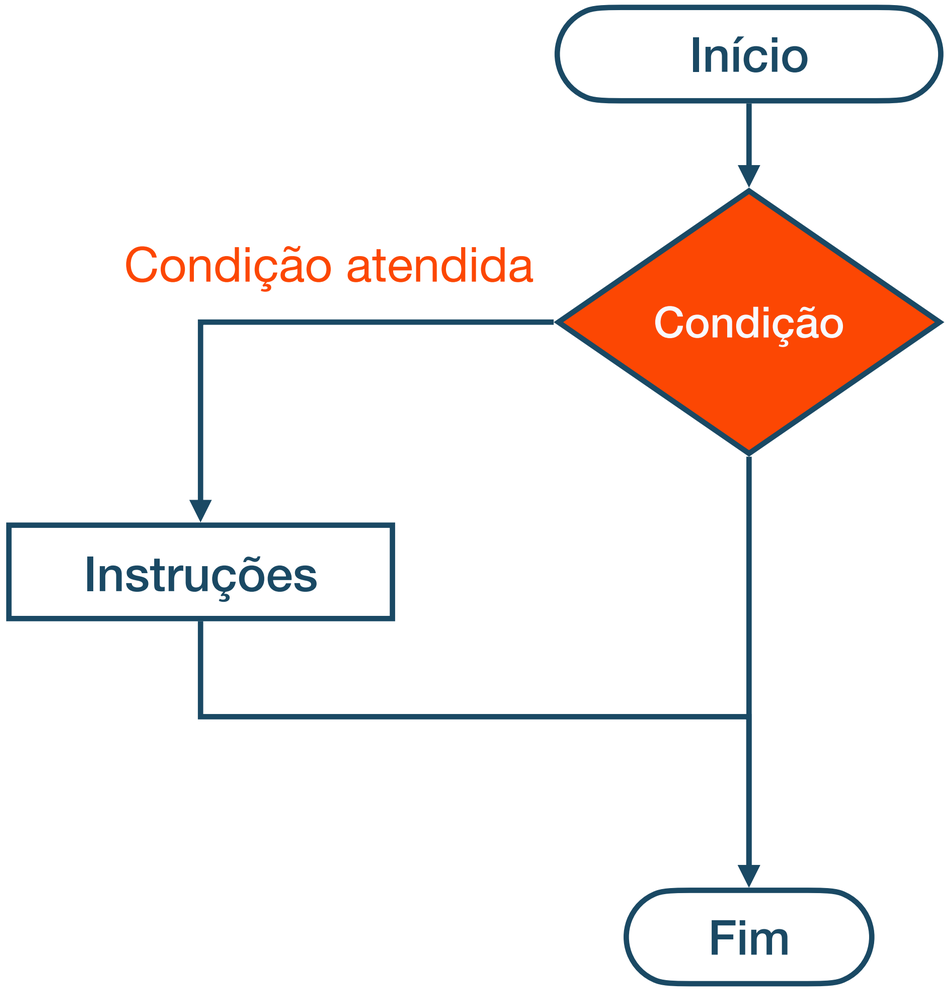
Operador	Significado
&&	E
||	Ou
!	Negação

if / else

if / else

Seletor de caminho único

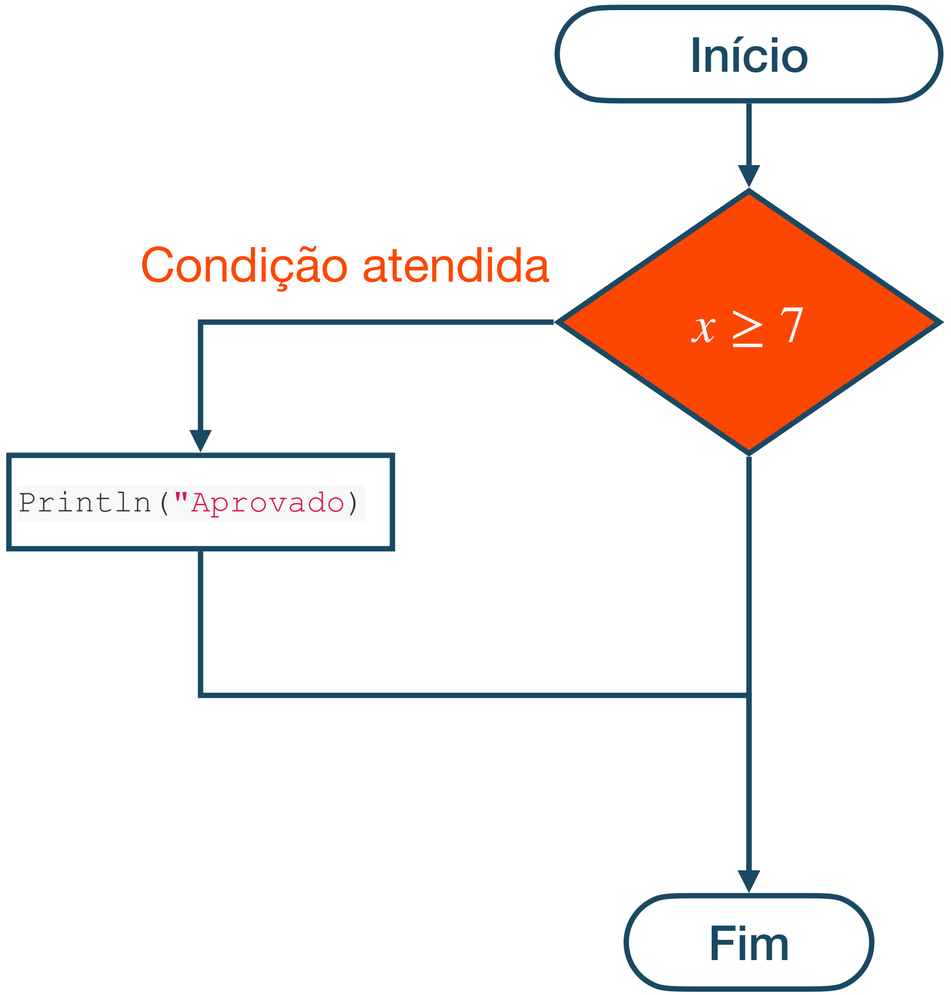
- Também chamado de seletor unário
- O bloco de instruções do seletor é executado quando a condição é atendida



if / else

Seletor de caminho único

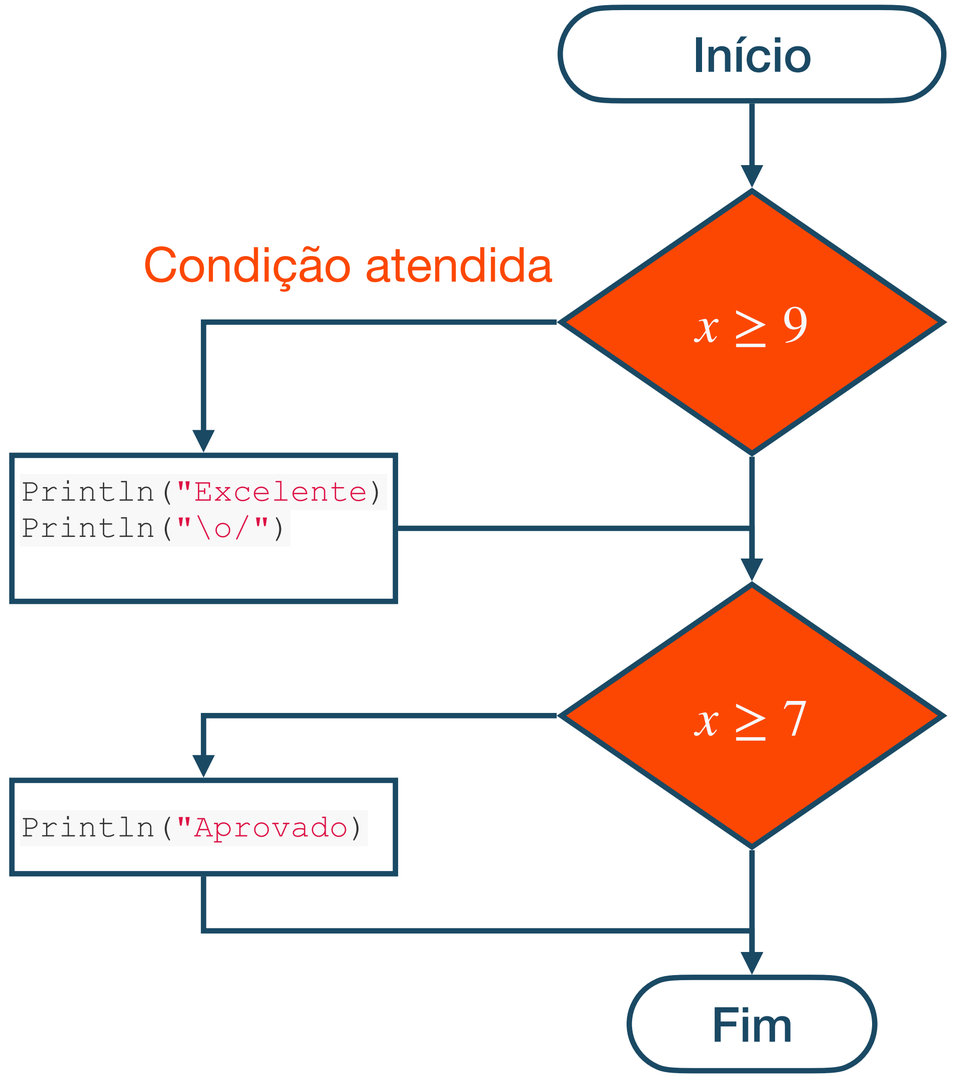
```
var nota int = 8

if nota >= 7 {
   fmt.Println("Aprovado")
}

fmt.Println(“Até a próxima")
```



if / else

Seletor de caminho único

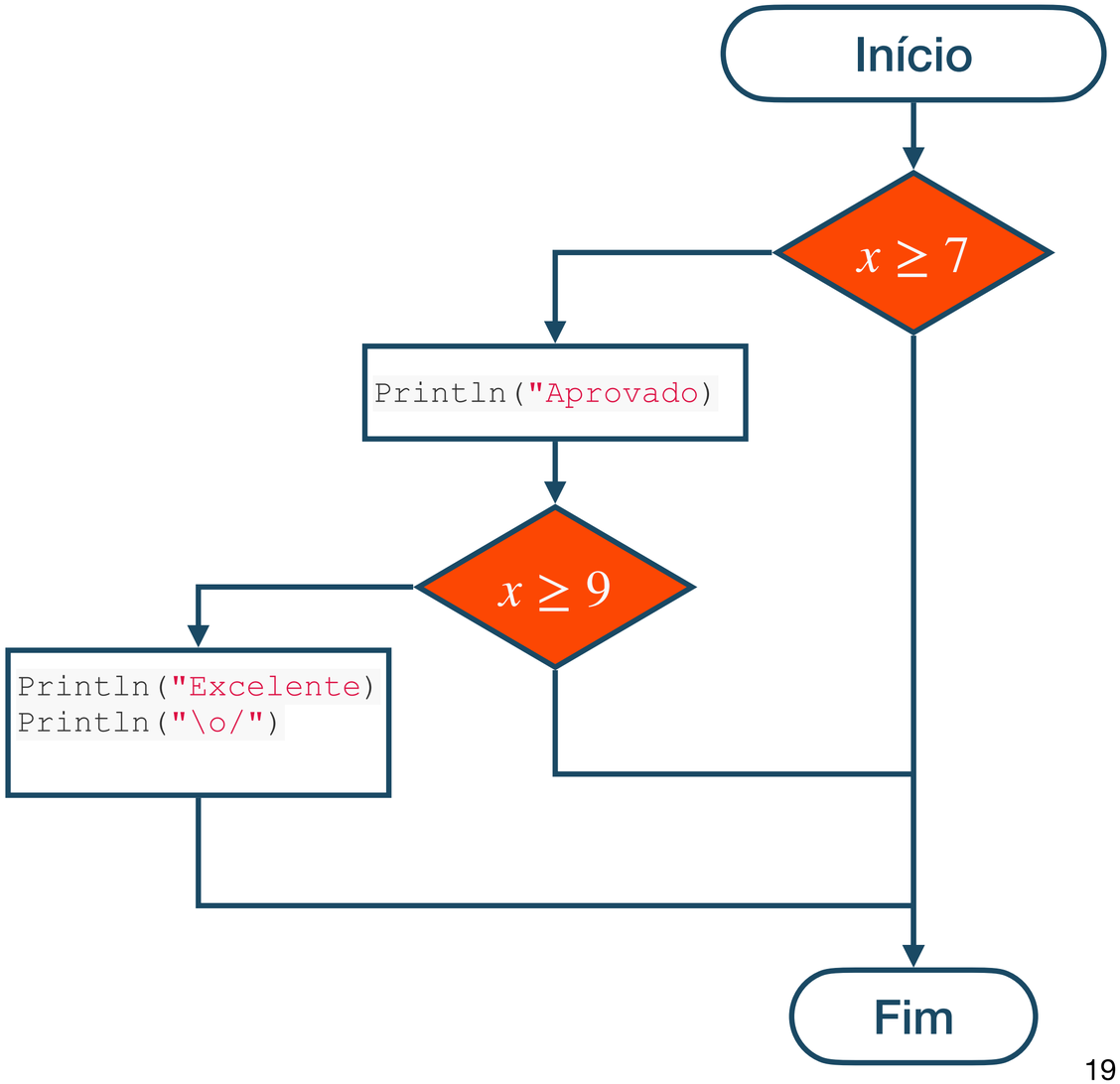
```
var nota int = 8

if nota >= 9 {
    fmt.Println("Excelente")
    fmt.Println("\o/")
}

if nota >= 7 {
    fmt.Println("Aprovado")
}
fmt.Println(“Até a próxima”)
```



if / else

Seletores aninhados

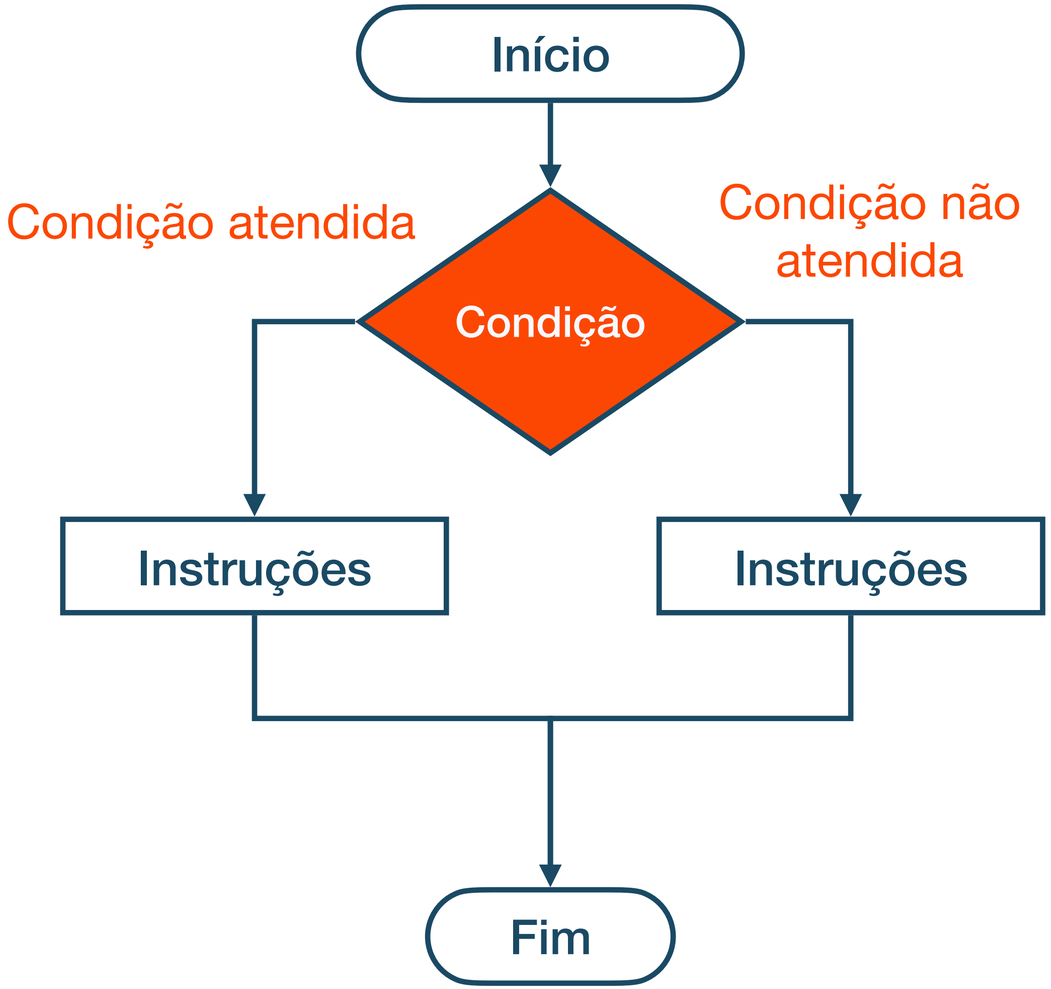
```
var nota int = 8

if nota >= 7 {
   fmt.Println("Aprovado")
   if nota >= 9 {
      fmt.Println("Excelente")
      fmt.Println(“\o/")
   }
}
fmt.Println(“Até a próxima”)
```



if / else

Seletor de dois caminhos

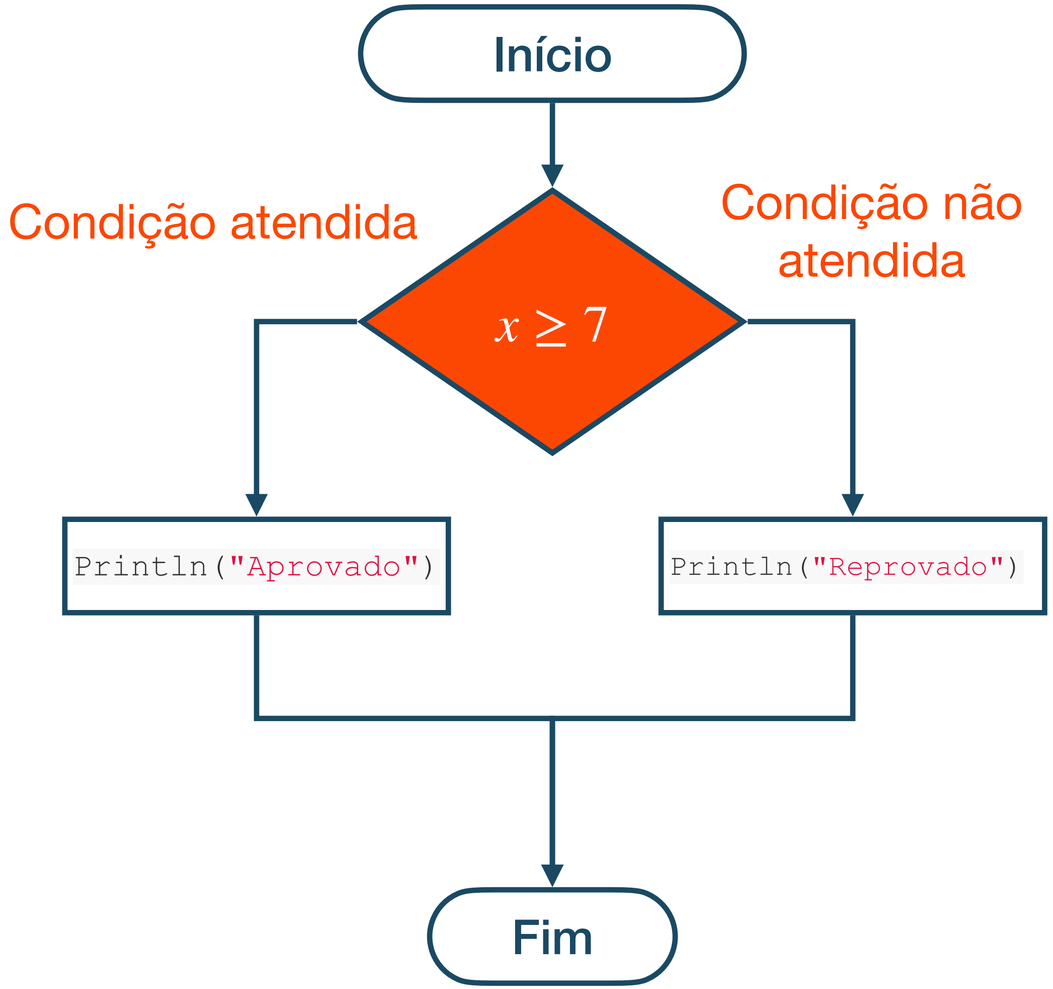
- Às vezes queremos fazer algo quando a condição é verdadeira e outra coisa quando a condição é falsa
- Similar a um entroncamento em uma estrada



if / else

Seletor de dois caminhos

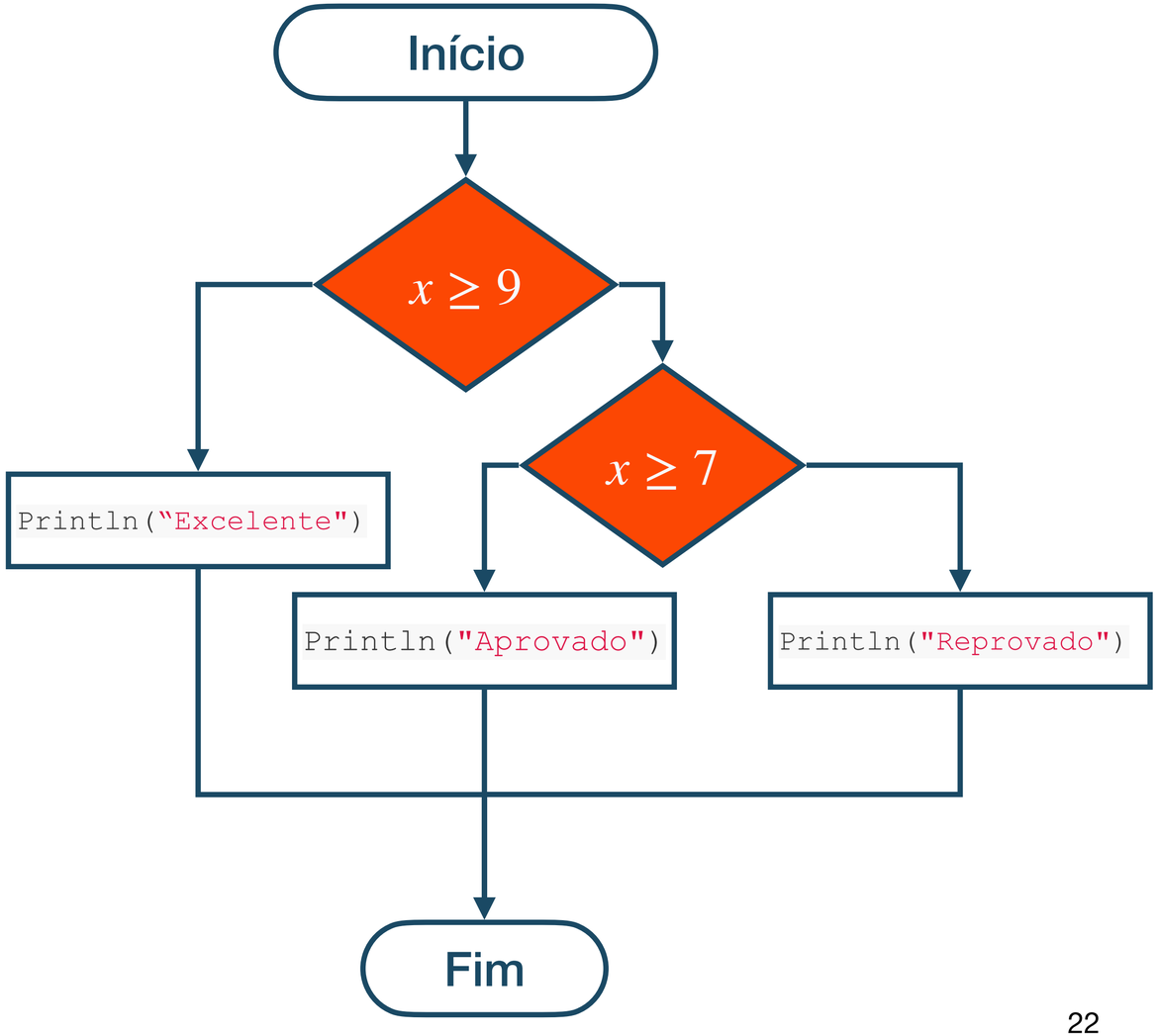
```
var nota int = 8

if nota >= 7 {
    fmt.Println("Aprovado")
} else {
    fmt.Println("Reprovado")
}

fmt.Println("Até a próxima")
```



if / else

Seletor de múltiplos caminhos

```
var nota int = 8

if nota >= 9 {
    fmt.Println("Excelente")
} else if nota >= 7 {
    fmt.Println("Aprovado")
} else {
    fmt.Println("Reprovado")
}
```



if / else

Qual instrução nunca será executada?

```
if x < 2 {
   fmt.Println("Abaixo de 2")
} else if x >= 2 {
   fmt.Println("Dois ou mais")
} else {
   fmt.Println("Algo mais")
}
```

if / else

Qual instrução nunca será executada?

```
if x < 2 {
   fmt.Println("Abaixo de 2")
} else if x < 20 {
   fmt.Println("Abaixo de 20")
} else if x < 10 {
   fmt.Println("Abaixo de 10")
} else {
    fmt.Println("Algo mais")
}
```

Exercícios

Exercícios

Par ou ímpar

- Escreva um programa que determine se um número é par ou ímpar.
  - O número deve ser informado pelo usuário

Exercícios

Par ou ímpar

- Escreva um programa que determina o vencedor de uma partida de par ou ímpar

Exercícios

Folha de pagamento

- Escreva um programa que peça as seguintes informações ao usuário:
  - Salário
  - Carga horária semanal
  - Quantidade de horas trabalhada no mês
- O programa deve mostrar na tela quanto deve ser pago ao funcionário ao final do mês.
  - O valor da hora extra, é igual 1.5 do valor da hora normal
  - A saída deve ter o seguinte formato:
  - O funcionário deve receber 1500.00 reais, sendo 500 referente a horas extras

Exercícios

Pedra, papel e tesoura

- Escreva um programa que determina o vencedor de uma pedra, papel e tesoura
- Regras:
  - Pedra ganha de tesoura
  - Tesoura ganha de papel
  - Papel ganha de pedra

Exercícios

Pedra, papel, tesoura, lagarto e spock.

- Problema: Algoritmo que determina o vencedor de uma pedra, papel, tesoura, lagarto e spock
- Regras:
  - Tesoura corta papel
  - Papel cobre pedra
  - Pedra esmaga lagarto
  - Lagarto envenena Spock
  - Spock esmaga (ou derrete) tesoura
  - Tesoura decapita lagarto
  - Lagarto come papel
  - Papel refuta Spock
  - Spock vaporiza pedra
  - Pedra quebra tesoura

Lógica Booleana

Lógica booleanas

Operadores lógicos

Operador	Significado
&&	E
||	Ou
!	Negação

Lógica Booleana

Operador && (E)

- Condição 1 e Condição 2
  - Se é maior de idade e tem habilitação
  - Só é verdadeiro quando ambas são verdadeiras

A	B	A && B
T	T	T
T	F	F
F	T	F
F	F	F

Lógica Booleana

Operador || (Ou)

- Condição 1 ou Condição 2
  - Se é cliente vip ou tem desconto
  - É verdadeiro quando pelo menos uma condição é verdadeira

A	B	A || B
T	T	T
T	F	T
F	T	T
F	F	F

Lógica Booleana

Operador ! (negação)

- negação de Condição 1

A	!A
T	F
F	T

switch

switch

Par ou ímpar

- Instrução que permite a seleção de múltiplos caminhos
  - Ideal para quando comparamos uma única variável com vários valores constantes
  - Alternativa elegante ao uso de vários if / else if /

switch

Dias da semana

```
switch dia {
  case 1:
    fmt.Println("Domingo")
  case 2:
    fmt.Println("Segunda")
  case 3:
    fmt.Println("Terça")
  case 4:
    fmt.Println("Quarta")
  case 5:
    fmt.Println("Quinta")
  case 6:
    fmt.Println("Sexta")
  case 7:
    fmt.Println("Sábado")
  default:
    fmt.Println("Dia inválido")
}
```

switch

Sem múltiplos casos

```
var letra string = "a"

switch letra {
case "a", "e", "i", "o", "u":
    fmt.Println("Vogal")
default:
    fmt.Println("Consoante")
}
```

switch

Aprovado ?

```
var nota uint = 8

switch nota {
  case 10, 9:
    fmt.Println("Excelente")
  case 8, 7:
    fmt.Println("Bom")
  case 6, 5:
    fmt.Println("Regular")
  default:
    fmt.Println("Reprovado")
}
```

switch

Menu de opções

```
var opcao uint = 3

switch opcao {
 case 1:
    fmt.Println("Cadastrar usuário")
 case 2:
    fmt.Println("Listar usuários")
 case 3:
    fmt.Println("Excluir usuário")
 case 0:
    fmt.Println("Sair")
 default:
    fmt.Println("Opção inválida")
}
```

switch

Sem variável

```
var idade uint = 3

switch {
  case idade < 12:
    fmt.Println("Criança")
  case idade < 18:
    fmt.Println("Adolescente")
  case idade < 60:
    fmt.Println("Adulto")
  default:
    fmt.Println("Idoso")
}
```

Erros comuns

Erros comuns

Erro 1: usar = em vez de ==

```
if nota = 7 // errado
```

Erros comuns

Erro 2: esquecer chaves

```
if nota >= 7
   fmt.Println("Aprovado")
```

Erros comuns

Erro 3: lógica invertida

```
if nota < 7 // errado
   fmt.Println("Aprovado")
```

Erros comuns

Erro 4: condições sempre verdadeiras

```
if 10 < 7 {
   // sempre é executado
}
```

Referências

- Go 101
- Data Types in Go