Estruturas de Decisão Condicional Apresentação

Estruturas de Decisão Condicional

Uma **estrutura de decisão condicional** é empregada para que uma tarefa ou ação seja executada dependendo de uma condição especificada, avaliada por meio de um teste lógico, que é basicamente uma pergunta.

Desvio Condicional Simples: SE

Desvio Condicional Simples

se <condições> então <instruções> fim

- Essa instrução tem por finalidade tomar uma decisão de acordo com o resultado de uma condição (teste lógico).
- Se o teste retorna verdadeiro, as instruções contidas entre os comandos então e fim serão executadas; caso contrário, nada ocorre.

Sintaxe

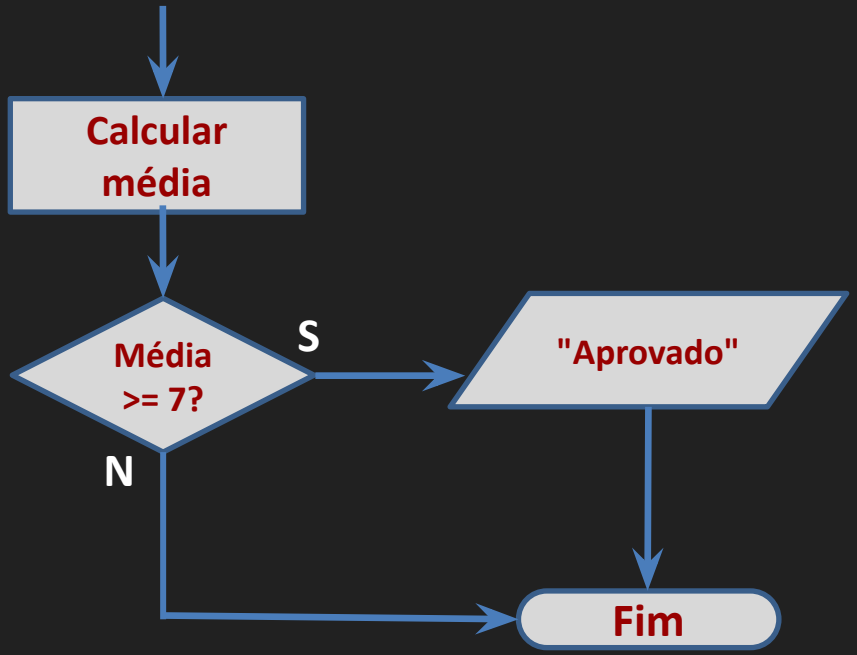
se (condição) **{**

instruções caso condição retorne verdadeiro

}

instruções executadas quando condição falsa ou após executar as instruções de condição verdadeiro

Desvio Condicional Simples



Exemplo

```
programa {
    real n1, n2
    real media
  funcao inicio() {
    escreva("Digite a primeira nota: ")
    leia(n1)
    escreva("Digite a segunda nota: ")
    leia(n2)
    media = (n1 + n2) / 2
    se (media >= 7) {
       escreva("Resultado: Aprovado \n")
    }
     /* se (media < 7) {
       escreva("Resultado: Reprovado \n")
    } */
    escreva("Sua média é ", media)
  }
```

Exemplo usando operador lógico

```
programa {
  real n1, n2
  real media
  funcao inicio() {
      escreva("Digite a primeira nota: ")
      leia(n1)
      escreva("Digite a segunda nota: ")
      leia(n2)
      media = (n1 + n2) / 2
      se ((media >= 5) e (media < 7)) {
        escreva("Resultado: Recuperação")
      }
      escreva("\nSua média é ", media)
  }
}
```

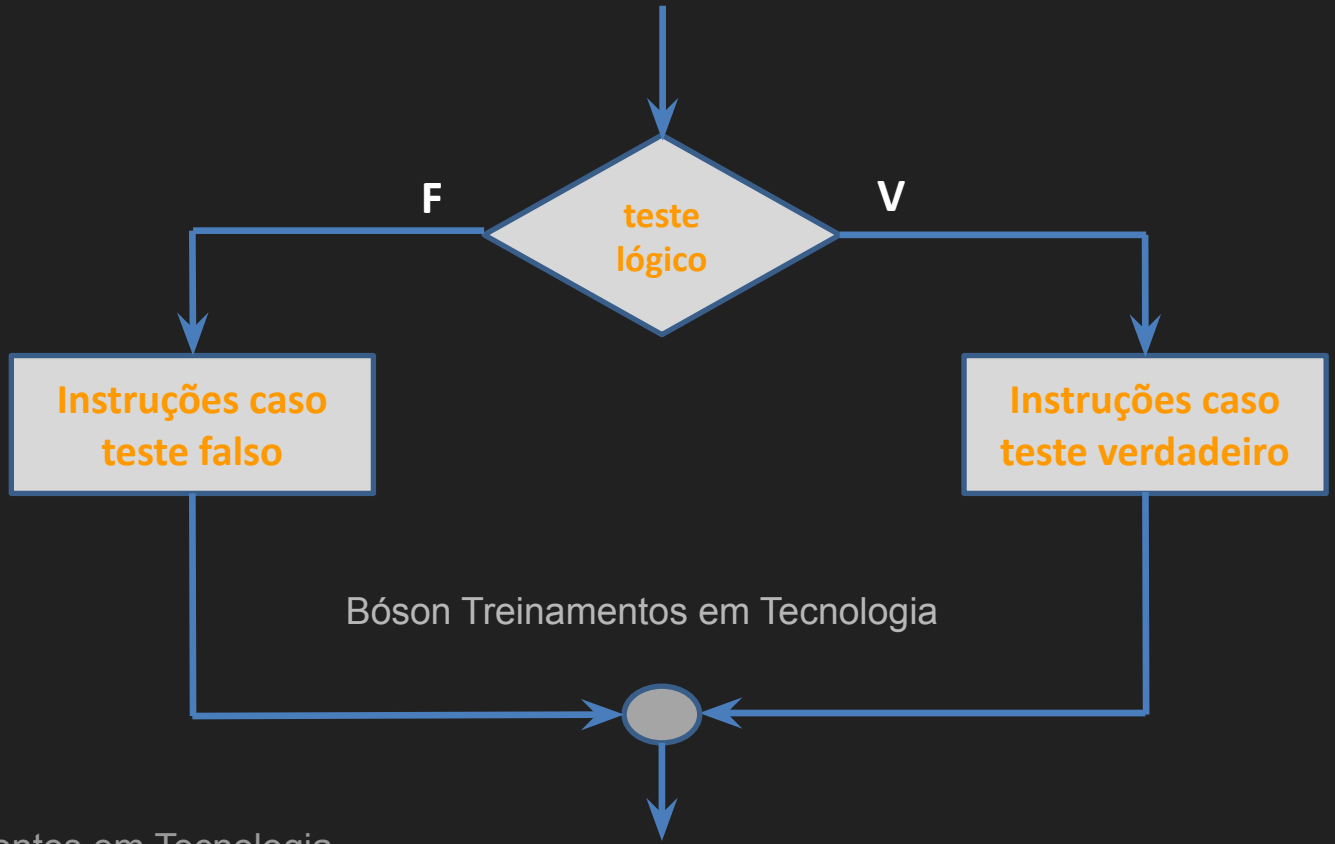
Desvio Condicional Composto

SE ... SENÃO

Desvio Condicional Composto

- Essa instrução toma uma decisão de acordo com o resultado de uma condição (teste lógico).
- Se o teste retornar verdadeiro, um grupo de instruções é executado
- Mas agora se o teste retornar falso, um grupo diferente de instruções será executado.

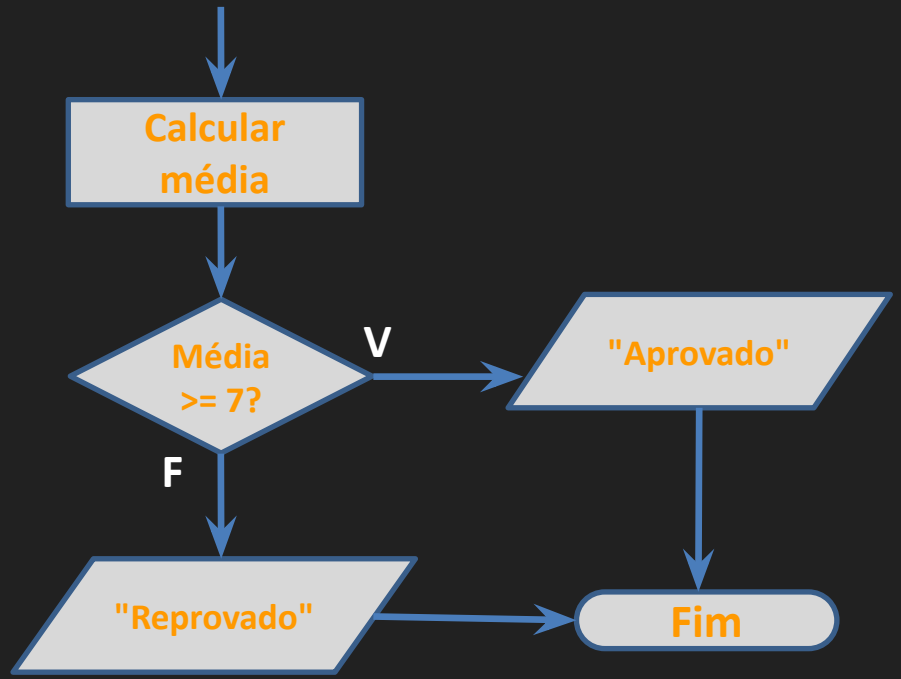
Desvio Condicional Composto



Sintaxe

```
se (condição) {
    Instruções caso condição retorne verdadeiro
}
senao  {
    Instruções caso condição retorne falso
}
Instruções após executar um dos blocos condicionais
(verdadeiro ou falso)
```

Desvio Condicional Composto



Exemplo

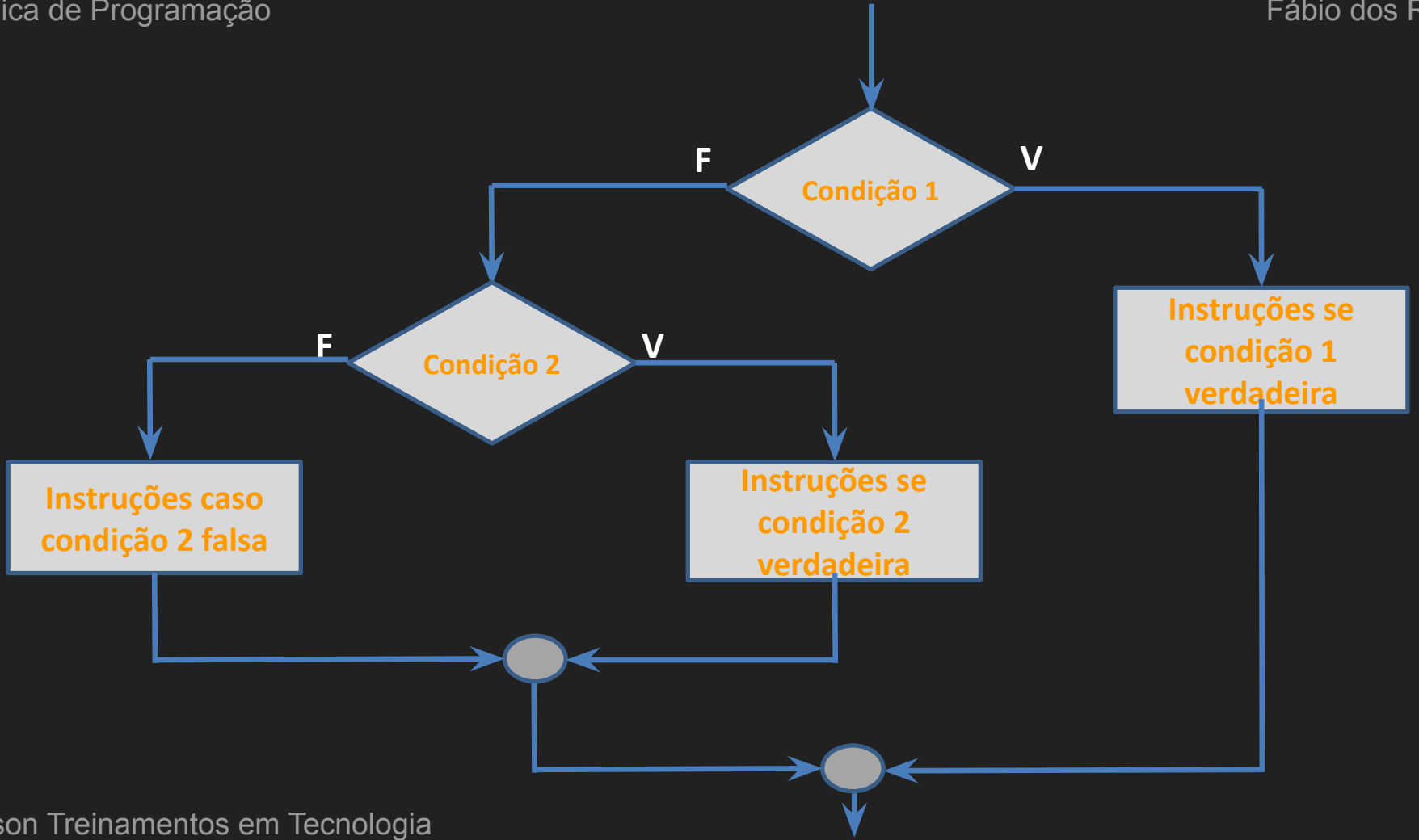
```
programa {
      real n1, n2
      real media
  funcao inicio() {
      escreva("Digite a primeira
       nota: ")
      leia(n1)
      escreva("Digite a segunda
       nota: ")
      leia(n2)
      media = (n1 + n2) / 2
      se (media >= 7) {
        escreva("Aprovado\n")
      }
      senao {
         escreva("Reprovado\n")
      }
         escreva ("Sua média é ",
media)
   }
}
```

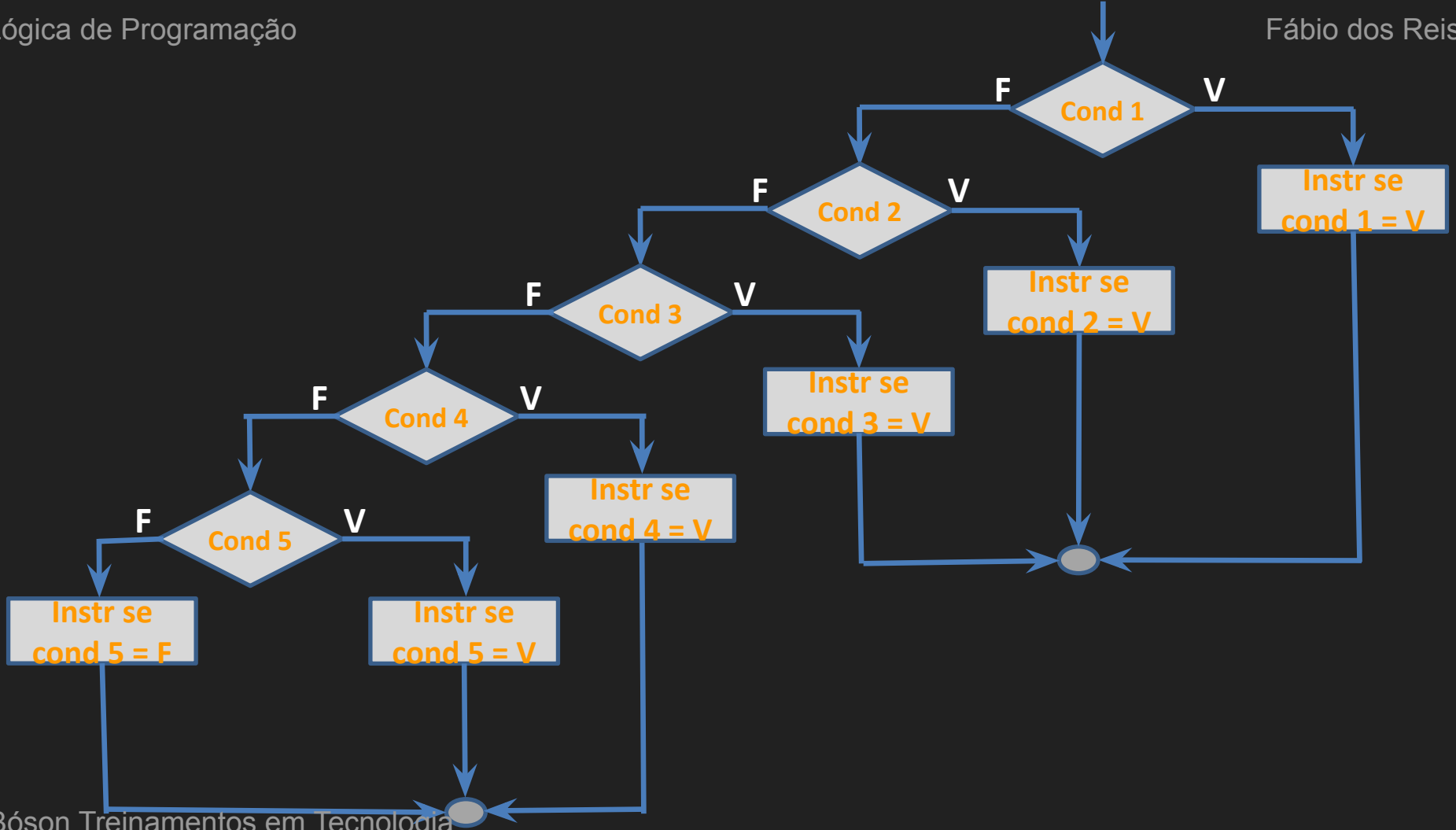
Desvio Condicional Encadeado

Desvio Condicional Encadeado

Ou *Desvio Condicional Aninhado*

- Usado quando é necessário verificar condições sucessivas onde uma ação será executada dependendo de um conjunto anterior de condições testadas.


F
Condição 1
V
F
Condição 2
V
Instruções caso condição 2 falsa
Instruções se condição 2 verdadeira
Instruções se condição 1 verdadeira


Cond 1
F
V
Instr se cond 1 = V
Cond 2
F
V
Instr se cond 2 = V
Cond 3
F
V
Instr se cond 3 = V
Cond 4
F
V
Instr se cond 4 = V
Cond 5
F
V
Instr se cond 5 = F
Instr se cond 5 = V

Sintaxe

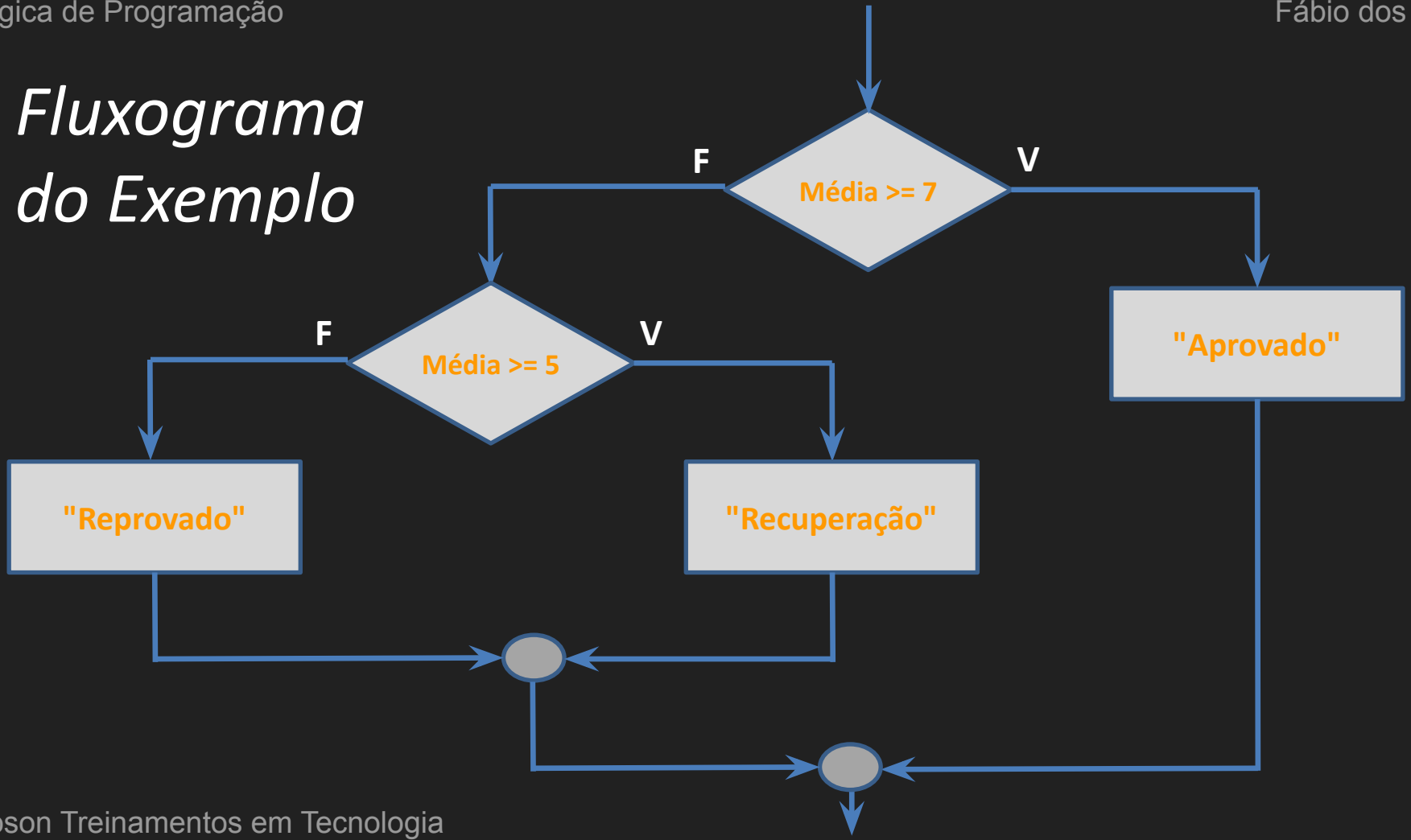
```
se (condição 1) {
     instruções caso condição 1 retorne verdadeiro
}
senao se (condição 2) {
     instruções caso condição 1 retorne falso e 2 retorne
 verdadeiro
}
senao {
     instruções caso todas as condições retornem falso
}
```

Exemplo

Criar um algoritmo para cálculo da média aritmética das notas de um aluno em uma disciplina, que mostre como resultado:

- "**Aprovado**", se a média for maior ou igual a 7.0
- "**Recuperação**", se a média for maior ou igual a 5.0 e menor que 7.0
- "**Reprovado**", se a média ficar abaixo de 5.0

Fluxograma do Exemplo



Exemplo

```
programa
{
    real n1, n2
    real media
    funcao inicio() {
        escreva("Digite a primeira
    nota: ")
        leia(n1)
        escreva("Digite a segunda
    nota: ")
        leia(n2)
        media = (n1 + n2) / 2
        se (media >= 7) {
          escreva("Aprovado\n")
          }
          senao se (media >= 5) {
           escreva("Recuperação\n")
        }
        senao {
            escreva("Reprovado\n")
        }
           escreva ("Média: ", media)
    }
}
```

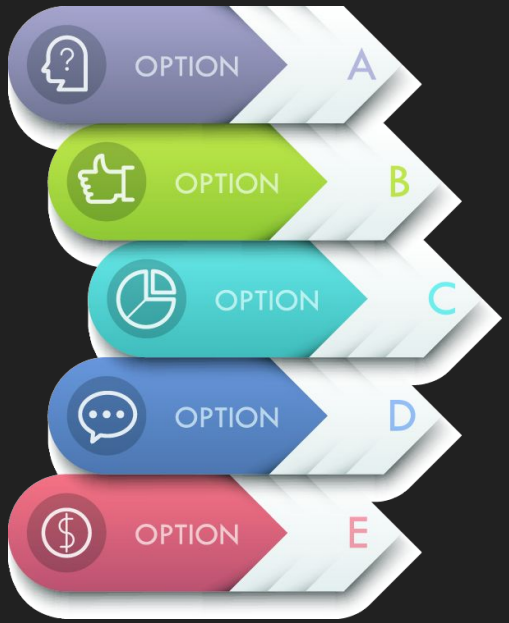
Desvio Condicional: Comando Escolha Caso

Comando Escolha CASO

- O comando caso é utilizado para que seja possível escolher uma opção dentre várias existentes, eliminando a necessidade de se usar diversos SE ... ENTÃO encadeados.
- Isso simplifica a codificação do algoritmo, e facilita sua depuração e atualização.
- Esse comando testa uma condição, e dependendo do resultado, executará os códigos associados.

Sintaxe do ESCOLHA CASO

```
escolha (opção) {
    caso valor1:
                comandos a executar
                pare
    caso valor2:
                comandos a executar
                pare
    caso contrario:
                comandos-padrão
}
```



Exemplo de ESCOLHA CASO

```
programa
{
   caracter opcao
   funcao inicio()       {
     escreva ("Digite uma letra: ")
     leia(opcao)

     escolha (opcao) {
       caso 'a':
         escreva("Letra digitada:  a")
         pare
       caso 'b':
         escreva("Letra digitada:   b")
         pare
       caso contrario:
         escreva("Você digitou outra
letra")
     }
   }
}
```

ESCOLHA CASO: Criar menu de acesso

```
programa
{
  caracter menu

  funcao inicio()      {
      escreva("Escolha uma opção:")
      escreva("1. Abrir\n")
      escreva("2. Salvar\n")
      escreva("3. Fechar\n")

      leia(menu)
      limpa()
```

```
escolha (menu) {
        caso '1':
            escreva("O programa será aberto.
Aguarde.")
            pare
        caso '2':
            escreva("O arquivo será salvo")
            pare
        caso '3':
            escreva("Fechando a aplicação. Tchau!")
            pare
        caso contrario:
        escreva("Opção incorreta.")
  }  }  }
```

Bibliotecas

Biblioteca

Uma biblioteca é um arquivo que organiza código pré-definido para o uso em aplicações.
É uma *coleção de recursos* usados no desenvolvimento de software. Pode incluir:

- Dados de configuração
- Documentação
- Procedimentos
- Classes
- Funções
- Templates
- Especificações de tipos

entre outros recursos.

Vantagens de usar Bibliotecas

- Funções comuns são pré-definidas para programação modular.
- Rapidez no desenvolvimento da aplicação, pois podem ser usadas por vários programas distintos, mas que necessitem de funcionalidades similares. Assim, implementamos o ***reuso de código*** nas aplicações.
- Com bibliotecas podemos escrever códigos menores e mais organizados, poupando tempo precioso no desenvolvimento e diminuindo a ocorrência de erros.
- Facilitam a atualização de funções do programa.
- Ao usar uma biblioteca, um programa passa a ser capaz de executar suas funções sem que seja necessário implementá-las no programa em si.

Como usar uma Biblioteca - Exemplo

Para usar uma biblioteca em um programa, primeiramente é preciso importá-la, ou seja, declarar que desejamos usar o código contido nela. Para tal usamos a seguinte sintaxe:

```
inclua biblioteca nome_biblioteca [--> apelido]
```

A instrução ***inclua*** diz ao interpretador para usar o código da biblioteca informada, disponibilizando suas funcionalidades.

Podemos, opcionalmente, criar um ***apelido*** *('alias')*, que nada mais é que uma palavra pequena ou abreviada para facilitar a escrita de código ao usar a biblioteca.

Exemplo 01: Calcular raiz quadrada com uma biblioteca

```
programa {

    inclua biblioteca Matematica
    real num, raiz

    funcao inicio() {
        escreva("Digite um número entre 10 e 100: ")
        leia(num)

        raiz = Matematica.raiz(num, 2.0)

        escreva("A raiz quadrada de " + num + " é " + raiz)
    }
}
```

Exemplo 02: Texto em maiúsculas com biblioteca

```
programa {

    inclua biblioteca Texto --> t
    cadeia pais, maiusculas
    inteiro letras

    funcao inicio() {
        escreva("Digite o nome de um país: ")
        leia(pais)
        maiusculas = t.caixa_alta(pais)
        escreva("O país escolhido foi: " + maiusculas)
        letras = t.numero_caracteres(palavra)
       escreva("\nO nome da cidade possui " + letras + " caracteres.")
    }
}
```

Números Aleatórios

Números Aleatórios

- Um número aleatório é um valor numérico gerado automaticamente em um programa, por meio do emprego de algoritmos específicos.
- "*Alea*", em latim, significa "jogo", "dado" ou "sorte", daí a ideia de um número "jogado" ou "sorteado".

alea jacta est: o dado (*a sorte*) foi lançado

Função sorteia()

Geramos números aleatórios inteiros no Portugol Studio utilizando a biblioteca Util e o método sorteia().

Sintaxe:

var = Util.sorteia(inicial, final)

onde *inicial* e *final* indicam o intervalo de valores a serem sorteados

Gerar Números Aleatórios

Método *sorteia* da biblioteca *Util* no Portugol Studio

Sintaxe:
Util.sorteia(valor_mínimo, valor_máximo)

valor_mínimo e ***valor_máximo*** são números inteiros que definem a faixa de valores a ser empregada.

Exemplo:

```
inclua biblioteca Util
inteiro valor = Util.sorteia(10, 25) // Gera números entre 10 e 25
```

Números Aleatórios - Exemplo

```
inclua biblioteca Util → u
inteiro inicial, final, alea

  funcao inicio() {
      inicial = 10
      final = 30

      alea = u.sorteia(inicial,final)

      escreva("Número aleatório gerado: " + alea)
  }
```

Gerar um número aleatório no intervalo de 10 a 30