Capítulo 6: Tomando decisões entre muitas opções

Olá querido leitor, conseguiu resolver o exercício do último capítulo? No final deste capítulo você poderá ver a minha solução para você conferir com o seu algoritmo.

No último capítulo nós falamos da estrutura **SE-ENTÃO-SENÃO**, que é usada para fazer os nossos programas tomarem decisões por si só.

Dei o exemplo do caixa eletrônico, em que o programa deveria verificar se o valor que desejamos sacar é menor que o saldo disponível.

Neste capítulo vamos ver qual estrutura de controle de fluxo devemos utilizar quando temos muitas opções para tomar decisão.

Antes de aprender a estrutura ESCOLHA-CASO, vamos ver uma coisa que a princípio não tem nada a ver com o nosso assunto, mas vai te ajudar a entender como funciona esta estrutura.

Equipamentos de rede de computadores

Talvez você já saiba mais ou menos algumas coisas sobre rede de computadores. Existem várias topologias de redes: estrela, barramento, anel, etc...

Mas quero chamar a sua atenção para dois equipamentos utilizados nas redes de computadores. Um é o **HUB** e o outro é o **SWITCH**.

Esses dois equipamentos são muito parecidos, algumas pessoas até pensam que são a mesma coisa. Mas há uma pequena diferença entre eles.

A tarefa é a mesma, transferir dados entre as portas, a diferença é a forma com que a tarefa é realizada por cada equipamento.

Basicamente o **HUB é burro** e o **SWITCH é inteligente**. Como assim?

Simples, quando o HUB recebe dados por uma porta, ele reenvia esses dados para TODAS as portas.

Por exemplo, se o computador ligado na porta 1 enviou um pacote de dados para o computador ligado na porta 5, o HUB enviará os dados para todas as portas, 1, 2, 3, 4, 5, 6... O computador de destino que vai descobrir se o pacote de dados é pra ele ou não, caso não o seja ele vai ignorar o pacote de dados.

Isso significa que os HUBs deixam a rede lenta, pois haverá muito congestionamento de dados na rede e processamento desnecessário pelos computadores. Além disso, apenas um pacote estará trafegando na rede por vez.

Ou seja, o **HUB é burro**!

Já o SWITCH é mais inteligente. Quando o SWITCH recebe um pacote de dados, ele identifica a porta correta para encaminhar aquele pacote de dados.

Por exemplo, se o computador ligado na porta 1 enviou um pacote de dados para o computador ligado na porta 5, o SWITCH enviará os dados apenas para a porta 5.

Dessa forma há menos congestionamento na rede e é possível trafegar vários pacotes na rede paralelamente.

Lembra que inglês é importante?

Talvez você esteja se perguntando o que tem a ver o HUB e o SWITCH com o assunto deste capítulo. Tudo!

A estrutura ESCOLHA-CASO funciona da mesma forma que o SWITCH das redes de computadores só que ao invés de enviar um pacote de dados para uma determinada porta, vamos enviar o fluxo do algoritmo para um determinado ponto do código. A ideia é a mesma!

A propósito, como eu disse no primeiro capítulo, inglês é essencial para trabalhar com programação, EMBORA NÃO SEJA IMPEDITIVO. E quando você estiver programando em inglês verá que ESCOLHA-CASO é conhecido como SWITCH-CASE.

A estrutura ESCOLHA-CASO

Lembra do SE-ENTÃO-SENÃO do capítulo passado? Imagine que você tem um menu de opções e o usuário deve escolher uma opção, dentre várias. Como você identificaria qual opção o usuário digitou? Talvez você faria algo assim ...

```
SE opção = 1 ENTÃO
    "instruções a serem executadas caso opção = 1"
SENÃO
    SE opção = 2 ENTÃO
        "instruções a serem executadas caso opção = 2"
    SENÃO
        SE opção = 3 ENTÃO
            "instruções a serem executadas caso opção = 3"
        SENÃO
            ...
        FIM-SE
    FIM-SE
FIM-SE
```

Ou seja, vários SE-ENTÃO-SENÃO aninhados, um no SENÃO do outro...

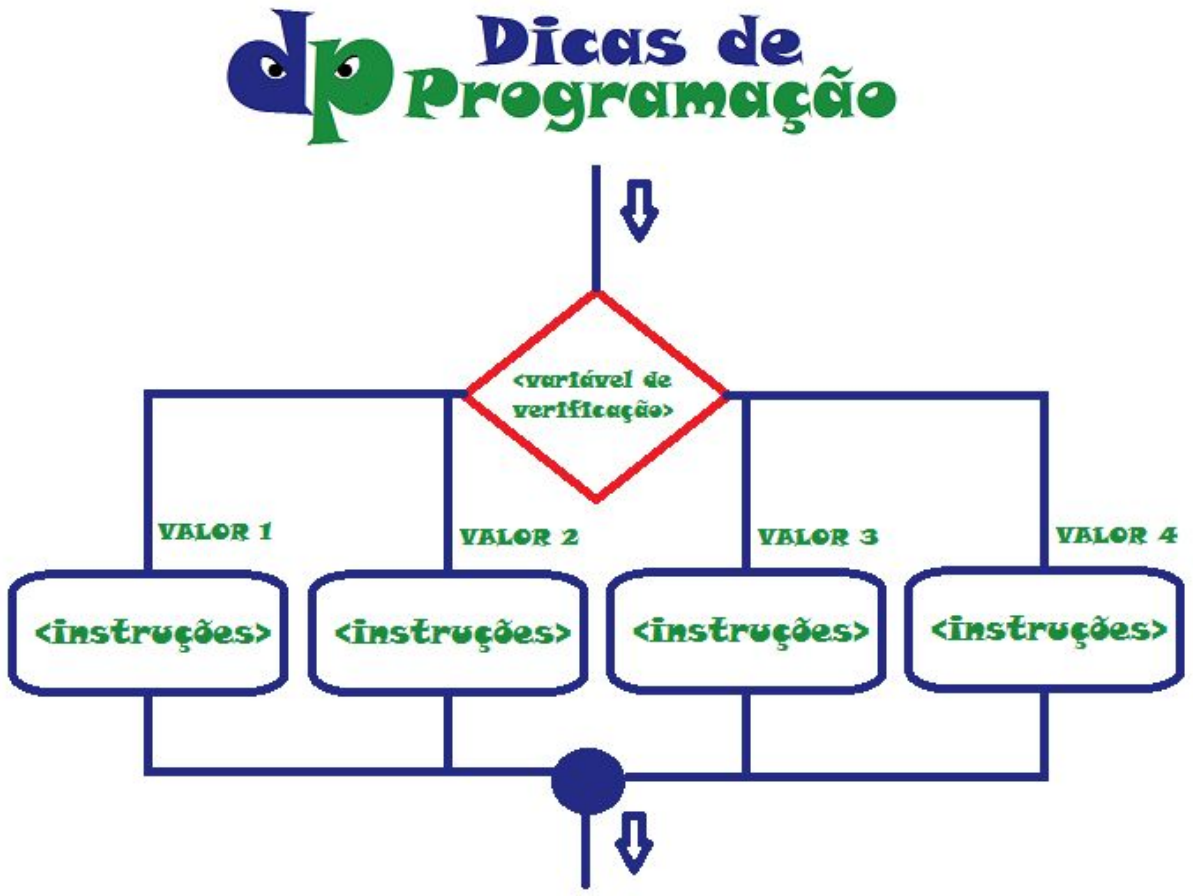
A proposta do ESCOLHA-CASO é ser uma solução mais elegante para este caso. Levando o fluxo do programa direto ao bloco de código correto (igual o switch), dependendo do valor de uma variável de verificação.

Essa é a estrutura ESCOLHA-CASO.

```
ESCOLHA <variável de verificação>
    CASO <valor1> FAÇA
                "instruções a serem executadas caso <variável de
verificação> = <valor1>"
    CASO <valor2> FAÇA
                "instruções a serem executadas caso <variável de
verificação> = <valor2>"
    CASO <valor3> FAÇA
                "instruções a serem executadas caso <variável de
verificação> = <valor3>"
    ...
```

```
FIM-ESCOLHA
```

O esquema visual do fluxograma desta estrutura é como a figura abaixo:



ESCOLHA-CASO na prática!

Nada melhor para aprender programação do que praticar. Bastante! Então vamos ver um exemplo prático da utilização do ESCOLHA-CASO em comparação ao SE-ENTÃO-SENÃO.

(Novamente vamos usar o Visualg para criar os nossos algoritmos, você pode fazer com lápis e papel, mas caso queira baixar o Visualg, **clique aqui para fazer o download**)

Imagine a seguinte situação: Você deseja criar um algoritmo para uma calculadora, o usuário digita o primeiro número, a operação que deseja executar e o segundo número. Dependendo do que o usuário informar como operador, o

algoritmo executará um cálculo diferente (soma, subtração, multiplicação ou divisão).

Vejamos como seria este algoritmo utilizando a estrutura SE-ENTÃO-SENÃO:

```
algoritmo "CalculadoraBasicaComSE"
var
   numero1 : REAL
   numero2 : REAL
   operacao : CARACTERE
   resultado : REAL
inicio

      ESCREVA ("Digite o primeiro número: ")
      LEIA (numero1)
      ESCREVA ("Digite a operação: ")
      LEIA (operacao)
      ESCREVA ("Digite o segundo número: ")
      LEIA (numero2)

      SE operacao = "+" ENTAO
         resultado := numero1 + numero2
      SENAO
         SE operacao = "-" ENTAO
            resultado := numero1 - numero2
         SENAO
            SE operacao = "*" ENTAO
               resultado := numero1 * numero2
            SENAO
               SE operacao = "/" ENTAO
                  resultado := numero1 / numero2
               FIMSE
            FIMSE
         FIMSE
      FIMSE

      ESCREVA ("Resultado: ", resultado)
```

```
fimalgoritmo
```

Veja como os SEs aninhados (dentro dos SENÃOs) deixam o código mais complexo. Dá pra entender a lógica, mas não é muito elegante. Agora vamos ver como ficaria a mesma lógica com a estrutura ESCOLHA-CASO.

```
algoritmo "CalculadoraBasicaComESCOLHA_CASO"
var
   numero1 : REAL
   numero2 : REAL
   operacao : CARACTERE
   resultado : REAL
inicio

      ESCREVA ("Digite o primeiro número: ")
      LEIA (numero1)
      ESCREVA ("Digite a operação: ")
      LEIA (operacao)
      ESCREVA ("Digite o segundo número: ")
      LEIA (numero2)

      ESCOLHA operacao
         CASO "+"
            resultado := numero1 + numero2
         CASO "-"
            resultado := numero1 - numero2
         CASO "*"
            resultado := numero1 * numero2
         CASO "/"
            resultado := numero1 / numero2
      FIMESCOLHA

      ESCREVA ("Resultado: ", resultado)

fimalgoritmo
```

Bem mais bonito! Né? Agora a lógica tá mais visível e elegante. O resultado dos dois algoritmos é o mesmo. Mas o código com o ESCOLHA-CASO é mais fácil de entender.

CASO NÃO TRATADO NA ESTRUTURA (OUTROCASO)

Além das opções tratadas na estrutura, é possível identificar quando o valor da variável não é equivalente a nenhum valor informado como opção nos CASOs, ou seja, é um "OUTROCASO".

No algoritmo que fizemos anteriormente, imagine se o usuário digitasse um valor diferente de "+", "-", "*" e "/". Caso quiséssemos apresentar uma mensagem para o usuário informando que ele digitou uma opção inválida, utilizaríamos esse recurso da estrutura ESCOLHA-CASO. Veja:

```
ESCOLHA operacao
   CASO "+"
      resultado := numero1 + numero2
   CASO "-"
      resultado := numero1 - numero2
   CASO "*"
      resultado := numero1 * numero2
   CASO "/"
      resultado := numero1 / numero2
   OUTROCASO
      ESCREVA("A operação digitada é inválida!")
FIMESCOLHA
```

Como você pôde observar, em termos de organização de código a estrutura ESCOLHA-CASO é uma opção muito elegante quando se tem muitos SE-ENTÃO-SENÃO para verificar a mesma variável. Facilita a leitura do algoritmo e a manutenção do código.

Exercício

Aprender programação é como aprender matemática, tem que praticar muito fazendo exercícios. Portanto vou deixar mais um exercício para você resolver sozinho, com o assunto que vimos neste capítulo.

* Crie um algoritmo em que o usuário digita uma letra qualquer e o programa verifica qual a ordem dessa letra no alfabeto, por exemplo: se o usuário digitar a letra 'G' o programa deve imprimir na tela, "A letra G está na posição 7 do

alfabeto”. Implemente com a estrutura ESCOLHA-CASO e depois com a estrutura SE-ENTÃO-SENÃO para perceber a diferença gritante no código.

É muito importante que você tente fazer os algoritmos sozinho antes de ver a resposta. Ok?