UNINOVE

Conceitos sobre Algoritmos: decisão e repetição

APRESENTAR AO ALUNO OS CONCEITOS SOBRE AS ESTRUTURAS DE DECISÃO E REPETIÇÃO UTILIZADAS NA CONSTRUÇÃO DE ALGORITMOS.

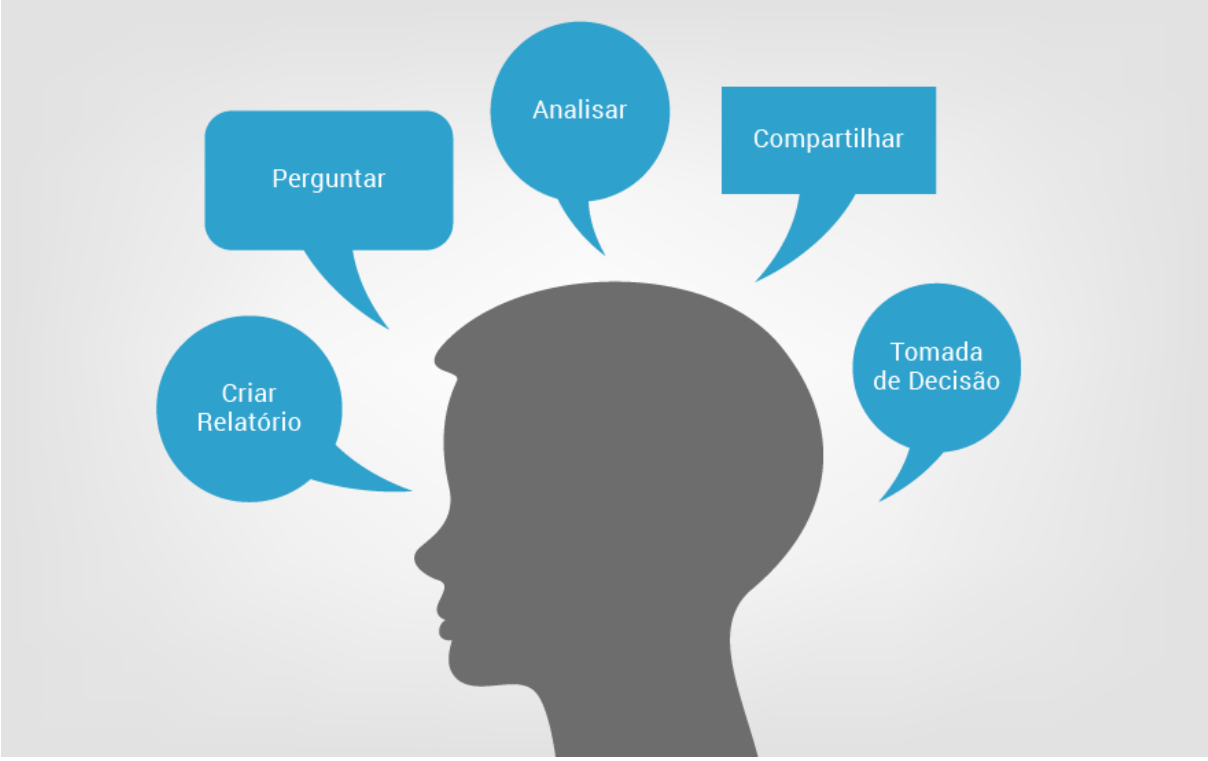
AUTOR(A): PROF. EDSON MELO DE SOUZA

1. Introdução

Durante a criação de um algoritmo é comum que surjam situações onde há a necessidade tomarmos uma decisão e também de repetirmos uma determinada ação. Durante nossa vida tomamos decisões a todo momento sem mesmo que tenhamos a percepção do que estamos fazendo, além de repetirmos diversas tarefas para atingir um objetivo.
Neste tópico serão apresentadas as estruturas de Decisão e de Repetição, com o objetivo de fornecer mais condições para a escrita de algoritmos mais complexos.

2. Estrutura de Decisão

Uma "decisão" ou "seleção" é o ato de fazermos uma "escolha" entre duas ou mais opções, sendo o mais comum as decisões entre duas opções disponíveis. Embora possam haver mais opções, o processo acaba se tornando mais complicado.



Legenda: PENSAMENTOS DECISóRIOS

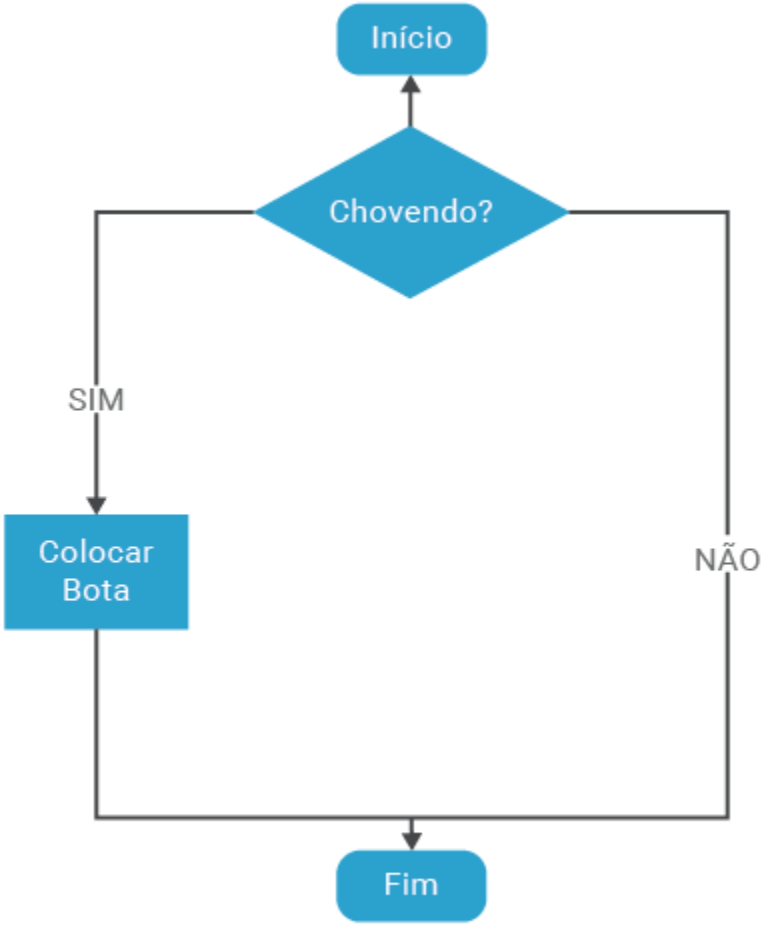
2.1 Decisões Simples

Como dito anteriormente, uma decisão é normalmente tomada com base em uma condição e, para escrevermos a decisão de uma determinada situação, vamos utilizar uma forma de escrita convencionada que permite representar a situação, conhecida como: “Se ... então” ou “Se ... então ... senão”
O “se ... então” é utilizado quando a decisão é feita de acordo com uma condição atendida. Vejamos:

Estrutura Se ... então

Pergunta	Condição	Ação
Se	estiver chovendo	colocar bota

Perceba que neste caso há uma decisão a ser tomada e só será realizada alguma ação caso esteja chovendo (condição).
A figura a seguir mostra o fluxograma que representa a situação:



Vamos escrever um fragmento em pseudocódigo para situação anterior, mas utilizando números, pois desta forma fica mais fácil compreender e também implementar em JavaScript.

```
início
    se (10 > 5) então
        escreva "O número 10 é maior que o número 5"
    fimse
fim
```

Vamos escrever em JavaScript o exemplo acima em que o número 10 é comparado com o número 5 e, se for "*verdadeira*" a comparação, será mostrada uma mensagem na tela informando que o número 10 é maior que o número 5.

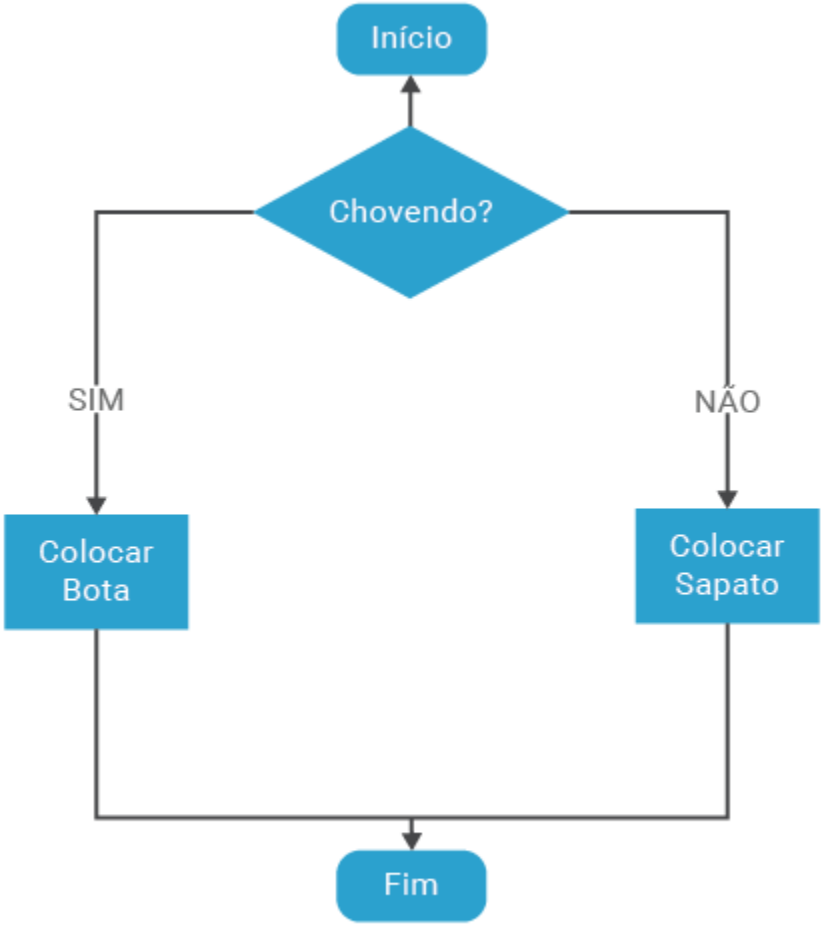
```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.         <title></title>
5. </head>
6. <body>
7.         <script type="text/javascript">
8.                 // Estrutura de decisão: se ... fimse
9.                 // Em JavaScript esta estrutura é definida pela instruçã
10.                if(10 > 5){
11.                        document.write("O número 10 é maior que o número
12.                }
13.        </script>
14. </body>
15. </html>
```

Podem ocorrer casos em que existam duas possibilidades a serem executadas, dependendo do resultado da condição. Vejamos:

Se ... então ... senão

Pergunta	Condição	Ação para SIM	Senão
Se	estiver chovendo	colocar bota	colocar sapato

Como você pode perceber, a decisão foi tomada sobre uma condição, mas existem duas opções possíveis de execução, ao contrário do primeiro exemplo que só há uma opção possível a ser executada.
Na figura a seguir é apresentado um fluxograma que representa a situação apresentada:



O pseudocódigo para este problema, novamente utilizando números, pode ser escrito da seguinte forma:

```
início
        se (10 < 5) então
                escreva "O número 10 é menor que o número 5"
        senão
                escreva "O número 10 é maior que o número 5"
        fimse
fim
```

Você pode perceber que agora é possível apresentar duas mensagens dependendo do resultado da avaliação dos números. Claramente será escrita a mensagem “O número 10 é maior que o número 5”, ou seja, a segunda opção “senão”, mas, se invertermos o sinal “<” (sinal de maior), teremos impressa a outra mensagem.

Vamos ver então este problema escrito em JavaScript:

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.         <title></title>
5. </head>
6. <body>
7.         <script type="text/javascript">
8.                 // Estrutura de decisão SE ... ENTAO ... SENAO
9.                 // Em JavaScript estra estrutura é definida pela instruç
10.                // a estrutura mostrada abaixo
11.                if(10 < 5){
12.                        document.write("O número 10 é menor que o número
13.                } else {
14.                        document.write("O número 10 é maior que o número
15.                }
16.
17.        </script>
18. </body>
19. </html>
```

2.1 Decisões Encadeadas

Uma decisão encadeada é quando ocorre uma situação que depende de outra anterior. Veja a situação a seguir, mostrada em um fragmento de pseudocódigo:

Início
    se (estiver chovendo) então
        se (chuva fraca) então
            colocar uma bota
        senão
            pegar um guarda-chuva
        fimse
    senão
        colocar um sapato
    fimse
fim

Você pode perceber que se a primeira condição for verdadeira, será testada novamente outra condição. Em diversas situações isto pode ocorrer, mas, este tipo de decisão requer muito cuidado, pois uma pequena desatenção pode causar grandes erros na sequência lógica do algoritmo.
Vamos ver um exemplo em JavaScript para um problema deste tipo:

```
<!DOCTYPE html>
<html>
<head>
        <title></title>
</head>
<body>
        <script type="text/javascript">
                // Estrutura de decisão SE ... ENTAO ... SENAO
                // Em JavaScript estra estrutura é definida pela instruç
                // a estrutura mostrada abaixo
                if(10 > 5){
                        if(10 > 8){
                                document.write("O número 10 é maior que
                        }else{
                                document.write("O número 10 é menor que
                        }
                } else {
                        document.write("O número 10 é menor que o número
                }

        </script>
</body>
</html>
```

No exemplo anterior você pode perceber como é muito importante, pois, nele estamos usando apenas números fixos. Agora imagine que você utilizará números que podem ser informados pelo usuário, portanto, todo cuidado é pouco.

2.2 Decisões com mais de uma Condição

Também existem situações em que pode ocorrer a existência de mais de uma condição para a realização de uma ação. No primeiro exemplo queríamos saber se 10 é maior que 5. E se tivermos uma situação onde necessitamos comparar dois números ao mesmo tempo? Por exemplo: 10 e 8 são maiores que 5?

Vamos ver como fica isso em JavaScript:

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.         <title></title>
5. </head>
6. <body>
7.         <script type="text/javascript">
8.                 // Estrutura de decisão: se ... fimse
9.                 // Em JavaScript esta estrutura é definida pela instruçã
10.                if( (10 > 5) || (8 > 5) ){
11.                        document.write("O número 10 é maior que o número
12.                }
13.        </script>
14. </body>
15. </html>
```

SAIBA MAIS!

Estruturas de Decisão: http://www.dicasdeprogramacao.com.br/estrutura-de-decisao-se-entao-senao/ (http://www.dicasdeprogramacao.com.br/estrutura-de-decisao-se-entao-senao/)
Algoritmos: http://www.dicasdeprogramacao.com.br/o-que-e-algoritmo/ (http://www.dicasdeprogramacao.com.br/o-que-e-algoritmo/)
Fluxogramas: https://www.citisystems.com.br/fluxograma/ (https://www.citisystems.com.br/fluxograma/)
Fluxo de Processos: http://www.venki.com.br/blog/fluxograma-de-processos/ (http://www.venki.com.br/blog/fluxograma-de-processos/)

No exemplo anterior, foram utilizadas duas barras “||” para realizar a comparação. Este é o símbolo que permite fazer a comparação do tipo "ou" entre valores em JavaScript.

Você percebeu como podemos fazer muitas “escolhas” com base em mais de uma comparação e, dependendo do problema a ser resolvido, a escrita do algoritmo vai ganhando complexidade.

Portanto, deve ser realizado um aprofundamento no assunto. Na caixa em destaque estão disponíveis alguns links para este propósito.

3. Estruturas de Repetição (*loop ou laço*)

Como o próprio nome diz uma repetição é algo que deve ser realizado uma ou mais vezes e, em programação de computadores, é amplamente utilizada.

Dependendo da situação, muitas vezes precisamos repetir determinada ação até que uma condição seja atendida. Por exemplo, imagine que você vai ao supermercado e, ao chegar no caixa, faça uma soma para saber o valor gasto com os produtos que estão no carrinho.

Neste caso, haverá uma repetição, pois serão realizadas "diversas somas", que representam cada item do carrinho.

Um outro exemplo é quando você está procurando por um livro em uma biblioteca e vai lendo os títulos na prateleira até encontrar (ou não) o livro procurado.

Outro exemplo clássico de repetições é quando há a necessidade de imprimir um relatório com várias linhas (boletim escolar), ou seja, são impressas as linhas até que o fim do conteúdo seja atingido.

A seguir vamos ver alguns tipos de repetição.

3.1. Repetição do tipo: PARA

A repetição do tipo "para ... até ... faça" é o tipo de repetição realizada quando sabemos, antecipadamente, o número de vezes que será realizada a repetição. Este tipo de estrutura tem a seguinte representação:

```
para de até faça
        escreva "olá"
fimpara
```

Vamos ver um exemplo em JavaScript que realiza esta tarefa de mostrar 5 vezes a mensagem na tela:

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.         <title></title>
5. </head>
6. <body>
7.         <script type="text/javascript">
8.                 // Estrutura de repetição: para ... faça
9.                 // Em JavaScript esta estrutura é definida pela instruçã
10.                for( contador = 0; contador < 5; contador ++ ){
11.                        document.write(contador + " -> Olá<br>");
12.                }
13.        </script>
14. </body>
15. </html>
```

No exemplo é realizada uma contagem de 0 até 5, ou seja, serão contados 5 números, e o resultado será:

0 -> Olá

1-> Olá

2-> Olá

3-> Olá

4-> Olá

Perceba que o valor inicial é zero (0) e o final (objetivo) é cinco (5), ou seja, a ação escreva “olá” será repetida até que o objetivo seja alcançado, sempre aumentando a contagem em um (1).

A seguir é mostrada uma linguagem desta estrutura em linguagem C.

3.2. Repetição do tipo: ENQUANTO … FAÇA

Diferente do para, enquanto ... faça sempre fará o teste da condição antes de realizar qualquer tarefa e somente continuará a repetir se a condição for satisfatória (verdadeira).

Vejamos um exemplo dessa estrutura em JavaScript:

```
<!DOCTYPE html>
<html>
<head>
        <title></title>
</head>
<body>
        <script type="text/javascript">
                // Estrutura de repetição: enquanto ... faça
                // Em JavaScript esta estrutura é definida pela instruçã

                // Declara uma variável para ser comparada
                var contador = 0;

                // Início do laço de repetição
                while( contador < 5 ){
                        // Escreve a mensagem na tela, juntamente com o
                        document.write(contador + " -> Olá<br>");

                        // Adiciona (1) cada vez que é executado o laço
                        contador++;
                }
        </script>
</body>
</html>
```

Perceba que a condição (contador < 5), ou seja, que se o valor de contador em questão é menor que cinco, então é verdade.

Um outro exemplo, agora mostrado em bloco de código, repete a instrução até que haja dinheiro.

enquanto (dinheiro suficiente) faça

    compre maça

    compre carne

    compre leite

fimenqunato

Neste caso, somente poderá ocorrer a compra enquanto houver dinheiro suficiente, ou seja, antes de realizar qualquer operação, é realizada a comparação da variável contador.

A seguir é mostrada uma simulação, em linguagem C, do laço de repetição enquanto ... faça.

3.3. Repetição do tipo: FAÇA…ENQUANTO

Similar ao enquanto ... faça, este tipo de repetição realiza uma ação “ao menos uma vez” e somente realizará outras repetições, caso a condição seja verdadeira. Vamos ao exemplo em um fragmento de pseudocódigo:

faça

    beber o suco do copo

enquanto

Perceba que este tipo de repetição “não garante” que existe suco no copo (condição), pois só vai ser checada a condição depois que o suco foi “bebido”.

Vamos ver um exemplo em JavaScript:

```
<!DOCTYPE html>
<html>
<head>
        <title></title>
</head>
<body>
        <script type="text/javascript">
                // Estrutura de repetição: faça ... enquanto
                // Em JavaScript esta estrutura é definida pela instruçã

                // Declara uma variável para ser comparada
                var contador = 6;

                // Início do laço de repetição
                do {
                        // Escreve a mensagem na tela, juntamente com o
                        document.write(contador + " -> Olá<br>");

                        // Adiciona (1) cada vez que é executado o laço
                        contador++;
                } while( contador < 5 );
        </script>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
        <title></title>
</head>
<body>
        <script type="text/javascript">
                // Estrutura de repetição: faça ... enquanto
                // Em JavaScript esta estrutura é definida pela instruçã

                // Declara uma variável para ser comparada
                var contador = 6;

                // Início do laço de repetição
                do {
                        // Escreve a mensagem na tela, juntamento com o
                        document.write(contador + " -> Olá<br>");

                        // Adiciona (1) cada vez que é executado o laço
                        contador++;
                } while( contador < 5 );
        </script>
</body>
</html>
```

Perceba que no exemplo o valor inicial do contador é seis (6) e, portanto, o valor a ser apresentado na tela é 6 -> olá, pois, apesar da condição não ser atendida (falsa), o código foi executado uma vez, mostrando o resultado.

Você pode perceber a importância dos laços de repetição, além de verificar como é simples a implementação em JavaScript.

SAIBA MAIS!

DevMedia: http://www.devmedia.com.br/while-e-do-while-lacos-de-repeticoes-estrutura-da-linguagem-parte-1/18870 (http://www.devmedia.com.br/while-e-do-while-lacos-de-repeticoes-estrutura-da-linguagem-parte-1/18870)

PUCRS: http://www.inf.pucrs.br/~pinho/LaproI/ComandosDeRepeticao/Repeticao.html (http://www.inf.pucrs.br/~pinho/LaproI/ComandosDeRepeticao/Repeticao.html)

Resumo

Neste tópico foram abordados os assuntos estrutura de decisão e de repetição, além do conceito de fluxograma e sua representação em blocos de código e pseudocódigos para criação de algoritmos.

Conclusão

Utilizar os recursos de decisão e repetição em um algoritmo é algo essencial para que uma ideia seja colocada no papel de forma que possa ser "traduzida" ao computador. Qualquer algoritmo que realize alguma tarefa útil, sempre conterá algum tipo de tomada de decisão e repetições. Portanto, entender bem os conceitos de decisão e repetição são o passo fundamental para a criação de algoritmos e programas que realizem tarefas de forma correta e atendam a um objetivo esperado.

ATIVIDADE FINAL

O que são fluxogramas?

A. São representações gráficas de como um determinado processo (sequência lógica) deve ocorrer.
B. É um comando para criação de um algoritmo.
C. É a forma de tomar decisões mais rapidamente.
D. Técnica utilizada para criar lações de repetição.

Uma decisão pode conter, no mínimo:

A. Uma ou mais condições.
B. Apenas uma condição.
C. Apenas duas condições.
D. Apenas três condições.

A estrutura de repetição que permite analisar a condição depois da primeira execução é:

A. Faça ... enquanto
B. Enquanto ... faça.
C. Para ... faça.

D. Enquanto ... para.

REFERÊNCIA

ALVES, WILLIAN PEREIRA. Linguagem e lógica de programação. 2014.

MANZANO, JOSÉ AUGUSTO N. G. Lógica para Desenvolvimento de Programação de Computadores. São Paulo: Érica, 2016.

SOUZA, JOÃO. Lógica para ciência da computação. Elsevier Brasil, 2015.