RAFAEL BONATI CALIXTO

Material de apoio lógica de programação

OPERADORES LÓGICOS

DEFINIÇÃO

Operador lógico, assim como um operador aritmético, é uma classe de operação sobre variáveis ou elementos pré-definidos.

Operador	Nome	Exemplo
==	Igual	$a == $b
!=	Diferente	$a != $b
<>	Diferente	$a <> $b
<	Menor que	$a < $b
>	Maior que	$a > $b
<=	Menor ou igual	$a <= $b
>=	Maior ou igual	$a >= $b

AND, NAND, OR, XOR e NOT são os principais operadores lógicos, base para a construção de sistemas digitais e da Lógica proposicional, e também muito usado em linguagem de programação. Os operadores AND, NAND, OR e XOR são operadores binários, ou seja, necessitam de dois elementos, enquanto o NOT é unário. Na computação, esses elementos são normalmente variáveis binárias, cujos possíveis valores atribuídos são 0 ou 1. Porém, a lógica empregada para essas variáveis serve também para sentenças (frases) da linguagem humana, onde se esta for verdade corresponde ao valor 1, e se for falsa corresponde ao valor 0.

EXEMPLOS

```java
import java.util.Scanner;

/**
 * Created by RafaelCalixto on 05/05/16.
 */
public class OperadoresLogicos {

    public static void main(String[] args){

        int x = 0;

        Scanner imput = new Scanner(System.in);

        System.out.println("Insira um número inteiro: ");
        x = imput.nextInt();

        if(x % 2 == 0){

         System.out.println("PAR");

        }else{

         System.out.println("IMPAR");

        }

    }
}
```

Acima temos um exemplo de como verificar se um número é PAR ou ÍMPAR. E utilizamos o operador IGUAL == para verificar.

```java
import java.util.Scanner;

/**
 * Created by RafaelCalixto on 05/05/16.
 */
public class OperadoresLogicos {

    public static void main(String[] args) {

        int x = 0;
        int y = 0;

        Scanner imput = new Scanner(System.in);

        System.out.println("Insira dois número inteiro: ");
        x = imput.nextInt();
        y = imput.nextInt();

        if((x >= y) || (x != 10)){

            System.out.println("VERDADEIRO");
        }else{

            System.out.println("FALSO");
        }
    }
}
```

Acima temos um outro exemplo utilizando o operador MAIOR OU IGUAL >= e o DIFERENTE != para comparar se o valor de X era maior que o de Y e que X era diferente de 10.

LAÇOS DE REPETIÇÕES

DEFINIÇÃO

Na linguagem Java, possuímos três blocos de repetição. Na verdade, possuímos mais um recurso, que é a recursividade, mas que já passa a ser uma estrutura de algoritmo do que uma estrutura da linguagem.

Os três comandos são: while, do/while e for.

while(enquanto)

```
while(numeroA > numeroB){

}
```

O comando **`while`** verifica uma condição lógica primeiro e depois executa as ações que estão contidas em seu bloco. sua sintaxe é a seguinte:

```
while(<condição lógica>){
        ...comandos...
}
```

Sendo assim, tudo vai ser repetido até a condição se tornar falsa. No **`while`** não está garantido que seus comandos serão executados pelo menos uma vez porque antes da execução existe uma condição que verifica se o bloco pode ser executado.
A utilização do **`while`** se torna mais atraente quando precisamos repetir algo que não seja fixo, algo irregular. Por exemplo, um comando que controla um menu de usuário, onde não sabemos quando o usuário vai fechar o menu.

do/while(repita)

```
do{

}while(numeroA == numeroB);
```

O comando **do/while** executa primeiro o bloco e depois verifica a condição lógica. A sintaxe é a seguinte:

```
do{
        ...comandos...
}while(<condição lógica>);
```

Todo o bloco se repetirá até a condição se tornar falsa. No **do/while**, pelo menos uma vez o bloco será executado. Ele executará todo o bloco e depois verificará a condição, diferente do **while**, que verificava antes de executar.

for(para)

```
for ( int i = 0; i < numeroB; i++){

}
```

O comando `for` é uma estrutura onde possui todo o controle na sua assinatura. Nela podemos declarar variáveis, verificar condições e incrementar valores à variável de controle. A sintaxe é a seguinte:

for (<declara variável>; <condição de controle>; <incrementa a variável>){

...comandos...

}

A aplicação do comando `for` seria ideal quando possuímos um controle exato da repetição, como por exemplo, quando sabemos que iremos percorrer um vetor ou um número certo de repetições. Caso contrário, o ideal seria o `while`.

```java
import java.util.Scanner;

/**
 * Created by RafaelCalixto on 13/05/16.
 */
public class LacoDeRepeticao {

    public static void main(String[] args){

        int n = 0;

        Scanner imput = new Scanner(System.in);

        System.out.println("Digite um valor de n: ");

        n= imput.nextInt();

        for(int i=n;i<=10;i++){

            System.out.println(i);
        }

        while (n<=10){

            System.out.println(n);
            n++;
        }
    }
}
```

Acima temos um exemplo utilizando o FOR e um utilizando o WHILE. Tanto um quanto o outro fazem a mesma coisa porém com um raciocínio diferente.

Referências:

https://pt.wikipedia.org/wiki/Operador_l%C3%B3gico

http://auxiliojava.blogspot.com.br/2010/09/lacos-de-repeticao.html