PROCESSO
DE
DESENVOLVIMENTO
JAVA WEB

INSTRUTOR: TIAGO DA ROSA VALÉRIO

E-MAIL: tiago_valerio_betha@hotmail.com

LÓGICA DE PROGRAMAÇÃO (ALGORITMOS)

“Eu não procuro saber as respostas, procuro compreender as perguntas.” Confúcio

INTRODUÇÃO

Para o desenvolvimento de qualquer programa, deve-se seguir basicamente as seguintes etapas, conhecidas como Ciclo de Vida do Sistema:

1) Estudo de Viabilidade (Requisitos da Aplicação)
2) Analise Detalhada do Sistema (Arquitetura)
3) Projeto Preliminar (Protótipos)
4) Aceitação do Cliente (Demonstração do Projeto)
5) Implementação ou Codificação
6) Testes do Sistema
7) Instalação e Manutenção

Podemos encontrar nas literaturas de informática, várias formas de representação das etapas que compõem o ciclo de vida de um sistema. Essas formas de representação podem variar, tanto na quantidade de etapas, quanto nas atividades a serem realizadas em cada fase.

Em nossa imagem, temos o exemplo de ciclo de vida do sistema (com sete fases). Apresentando na quinta etapa o desenvolvimento de um programa. Na verdade, os algoritmos estão presentes no nosso dia a dia sem que saibamos, pois uma receita culinária, as instruções de uso de um equipamento ou as indicações de um instrutor sobre como estacionar um carro por exemplo, nada mais são do que algoritmos.

No desenvolvimento de um sistema, quanto mais tarde um erro é detectado mais dinheiro e tempo se gasta para repará-lo. Assim, a responsabilidade do programador é maior na criação dos algoritmos do que na sua própria implementação. Quando o algoritmo é bem projetado, não teremos retrabalho para implementá-los novamente, e ainda testar e implantar no cliente.

Um algoritmo pode ser definido como um conjunto de regras (instruções), bem definidas, para a solução de um determinado problema. Segundo o dicionário Michaelis, o conceito de algoritmo é a “utilização de regras para definir ou executar uma tarefa específica ou para resolver um problema específico”.

A partir desses conceitos de algoritmos, pode-se perceber que a palavra algoritmo não é um termo computacional, ou seja, não se refere apenas à área de informática. É uma definição ampla que agora que você já sabe o que significa, talvez a utilize no seu cotidiano normalmente.

Na informática, o algoritmo é o "projeto do programa", ou seja, antes de se fazer um programa (software) na Linguagem de Programação desejada (Pascal, C, Delphi, Java), deve-se fazer o algoritmo do programa. Já um programa, é um algoritmo escrito numa forma compreensível pelo computador (através de uma Linguagem de Programação), onde todas as ações a serem executadas devem ser especificadas nos mínimos detalhes e de acordo com as regras de sintaxe da linguagem escolhida.

Um algoritmo não é a solução de um problema. Se assim fosse, cada problema teria um único algoritmo. Um algoritmo é um "caminho" para a solução de um problema e, em geral, existem muitos caminhos que levam a uma única solução satisfatória, ou seja, para resolver o mesmo problema pode-se obter vários algoritmos diferentes.

O objetivo ao final da disciplina é que você tenha adquirido capacidade de transformar qualquer problema em um algoritmo de boa qualidade, ou seja, a intenção é que você aprenda a Lógica de Programação dando uma base teórica e prática suficientemente boa, para que você domine os algoritmos e esteja habilitado a aprender uma Linguagem de Programação posteriormente.

LÓGICA

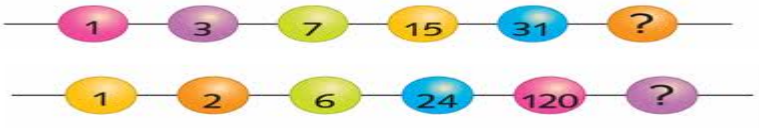
A lógica de programação é necessária para pessoas que desejam trabalhar com desenvolvimento de sistemas e programas. Ela permite definir a seqüência lógica para o desenvolvimento. Então, o que é lógica ?

Lógica de programação é a técnica de encadear pensamentos para atingir determinado objetivo.

SEQUÊNCIA LÓGICA

Estes pensamentos podem ser descritos como uma seqüência de instruções, que devem ser seguidas para cumprir uma determinada tarefa.

Seqüência Lógica são passos executados até atingir um objetivo ou solução de um problema.



INSTRUÇÕES

Na linguagem comum, entende-se por instruções “um conjunto de regras ou normas definidas para a realização ou emprego de algo”.

Em informática, porém, instrução é a informação que indica a um computador uma ação elementar a executar. Convém ressaltar que, uma ordem isolada não permite realizar o processo completo. Para isso é necessário um conjunto de instruções colocadas em ordem seqüencial lógica.

Por exemplo, se quisermos fazer um omelete de batatas, precisaremos colocar em prática uma série de instruções: descascar batatas, bater os ovos, fritar as batatas, etc. É evidente que essas instruções devem ser executadas em uma ordem adequada, não se pode descascar as batatas depois de frita-las.

Dessa maneira, uma instrução tomada em separado não tem muito sentido; para obtermos o resultado, precisamos colocar em prática o conjunto de todas as instruções na ordem correta. Instruções são um conjunto de regras ou normas definidas para a realização ou emprego de algo. Em informática, é o que indica a um computador uma ação elementar a executar.

ALGORITMO

Algoritmo é a forma organizada de expressar uma seqüência de passos que visam atingir a um objetivo definido. Algoritmo é a lógica necessária para o desenvolvimento de um programa.

Apesar do nome estranho, os algoritmos são muito comuns no nosso cotidiano, como por exemplo, em uma receita de bolo. Nela estão escritos os ingredientes necessários e a seqüência de passos ou ações a serem cumpridos para que se consiga fazer um determinado bolo.

De um modo geral, um algoritmo segue determinado padrão de comportamento, com objetivo de alcançar a solução de um problema. Padrão de comportamento: imagine a seqüência de números 1, 6, 11, 16, 21, 26, ... Para determinar qual será o sétimo elemento dessa série, precisamos descobrir qual a regra de formação, isto é, qual seu padrão de comportamento.

Como a seqüência segue certa constância facilmente determinada, somos capazes de determinar qual seria o sétimo termo ou outro termo qualquer. Descrevemos então, uma atividade bem cotidiana: trocar uma lâmpada, apesar de parecer óbvia, muitas vezes fazemos este tipo de atividade inconscientemente, sem percebermos os detalhes.

Vejamos como seria descrevê-la passo-a-passo:

-Pegar uma escada;

-Posicionar a escada embaixo da lâmpada;

-Buscar uma lâmpada nova;

-Subir na escada;

-Retirar lâmpada velha;

-Colocar a lâmpada nova.

Para se trocar a lâmpada, é seguida uma determinada seqüência de ações, representadas através desse algoritmo. Como isso pode ser seguido por qualquer pessoa, estabelece-se um padrão de comportamento. A ordenação tem por objetivo reger o fluxo de execução, determinando qual ação vem a seguir.

O algoritmo anterior tem um objetivo bem específico: trocar uma lâmpada. E se a lâmpada não estiver queimada? O algoritmo faz com que ela seja trocada do mesmo modo, não prevendo essa situação. Para solucionar este problema, podemos efetuar um teste seletivo, verificando se a lâmpada está ou não queimada.

-Pegar uma escada;

-Posicionar embaixo da lâmpada;

-Buscar uma lâmpada nova;

-Ligar o interruptor;

-Se a lâmpada não ascender, então:

-Subir na escada;

-Retirar a lâmpada velha;

-Colocar a lâmpada nova.

Dessa forma, algumas ações estão ligadas à condição (lâmpada não ascender). No caso da lâmpada ascender, as três linhas:

-Subir na escada;

-Retirar a lâmpada velha;

-Colocar a lâmpada nova.

Não serão executadas. Em algumas situações, embora o algoritmo resolva o problema proposto, a solução pode não ser a mais eficiente.

Exemplo: três alunos devem resolver um determinado problema:

- O aluno A conseguiu resolver o problema executando 35 linhas de programa;

-O aluno B resolveu o problema executando 10 linhas de programa;

-O aluno C resolveu o problema executando 54 linhas de programa.

Obviamente, o algoritmo desenvolvido pelo aluno B é menor e mais eficiente que os demais. Isso significa que há código desnecessário nos demais programas. Dessa forma, podemos otimizar o algoritmo anterior, uma vez que buscando a escada e a lâmpada sem saber se serão necessárias:

-Ligar o interruptor;

-Se a lâmpada não ascender, então:

-Pegar uma escada;

-Posicionar a escada embaixo da lâmpada;

-Buscar uma lâmpada nova;

-Subir na escada;

-Retirar a lâmpada velha;

-Colocar a lâmpada nova.

ITENS AVALIADOS NA CONSTRUÇÃO DE UM ALGORITMO

COMPLEXIDADE

A medida que colocávamos situações novas no problema a ser resolvido, aumentamos a complexidade do algoritmo. Esse certamente é o maior problema envolvido na construção de algoritmos. A complexidade pode ser vista como um sinônimo de variedade (quantidade de situações diferentes que um problema pode apresentar), as quais devem ser previstas na sua solução. Conviver com a complexidade é um mal necessário, é saudável fazer o possível para diminuí-la ao máximo, controlando o problema e encontrando a solução.

Devemos sempre diferenciar as seguintes questões O que ? e Como ? Muitos programadores aumentam a complexidade de um devido problema desnecessariamente. A forma errada de interpretação de um problema pode levar a respostas irrelevantes a solução almejada ou até mesmo a nenhuma solução, gerando algoritmos mais complexos do que o necessário.

Exemplo:

Perguntamos a um leigo sobre um relógio – Como é relógio?

Resposta – É um instrumento com três ponteiros concêntricos.

Então perguntamos – Um relógio com dois ponteiros. É possível?

Resposta – É, pode ser.

Continuamos – E um relógio com apenas um ponteiro, poderia ser uma possibilidade?

Resposta – O relógio pode ter três, dois ou ponteiro.

Nova pergunta – E sem ponteiro?

Resposta – Sim, pode ser digital.

Com todos estes questionamentos concluímos que, o relógio é um instrumento cuja finalidade é marcar o decorrer do tempo. Então estas variações poderão aumentar ou diminuir a complexidade de um sistema quando forem bem ou mal utilizadas.

LEGIBILIDADE

Mede a capacidade de compreensão de um algoritmo por qualquer observador (que não o construiu); a clareza com que sua lógica esta exposta. Quanto mais legível for um algoritmo, menor será sua complexidade.

PORTABILIDADE

Devido a quantidade enorme de linguagens de programação existentes, não será adotada nenhuma linguagem específica para trabalhar os algoritmos (ex: C, Pascal, Java, etc.). Isso porque a solução do problema fica ligada as características e recursos no qual a linguagem foi concebida.

Utilizaremos uma pseudo-linguagem (linguagem fictícia) que visa a permitir a representação dos algoritmos através da língua portuguesa (português estruturado). Esses algoritmos poderão ser convertidos facilmente para qualquer linguagem de programação usual.

TÉCNICA DE RESOLUÇÃO POR MÉTODO CARTESIANO

A famosa frase de Descartes “Dividir para conquistar” é muito importante dentro da programação. É um método que ataca um problema grande, de difícil solução, dividindo-o em problemas menores, de solução mais fácil. Se necessário, pode-se dividir novamente as partes não compreendidas. Este método pode ser esquematizado em passos:

Dividir o problema em partes.

Analisar a divisão e garantir a coerência entre as partes.

Reaplicar o método, se necessário.

PROGRAMAS

Os programas de computadores nada mais são do que algoritmos escritos numa linguagem de computador (Pascal, C, Cobol, Fortran, Visual Basic entre outras) e que são interpretados e executados por uma máquina, no caso um computador. Notem que dada esta interpretação rigorosa, um programa é por natureza, muito específico e rígido em relação aos algoritmos da vida real.

É a tradução para o inglês do algoritmo feito em português. O mais importante de um programa é a sua lógica, o raciocínio utilizado para resolver o problema, que é exatamente o algoritmo.

A forma de escrever um algoritmo em pseudocódigo (algoritmo que não usa nenhuma linguagem de programação) vai variar de autor para autor, pois, um traduz ao pé da letra a linguagem C, outro, o Pascal, outro, mistura as duas linguagens e assim por diante. É importante lembrar que estas variações vão sempre ocorrer, podemos dizer que é uma variação de autores adotados.

EXERCÍCIOS

1) Crie uma seqüência lógica para tomar banho:

2) Descreva com detalhes a seqüência lógica para Trocar um pneu de um carro.

PSEUDOCÓDIGO

Os algoritmos são descritos em uma linguagem chamada pseudocódigo. Este nome é uma alusão a posterior implementação em uma linguagem de programação, ou seja, quando formos programar em uma linguagem, por exemplo Visual Basic, estaremos gerando código em Visual Basic. Por isso, os algoritmos são independentes das linguagens de programação. Ao contrário de uma linguagem de programação, não existe um formalismo rígido de como deve ser escrito o algoritmo.

O algoritmo deve ser fácil interpretação e codificação. Ou seja, ele será o intermediário entre a linguagem falada e a linguagem de programação.

Para escrever um algoritmo precisamos descrever a seqüência de instruções, de maneira simples e objetiva. Para isso, utilizaremos algumas técnicas:

- Usar somente um verbo por frase;

- Imaginar que você está desenvolvendo um algoritmo para pessoas que não trabalham com informática;

- Usar frases curtas e simples;

- Ser objetivo;

- Procurar não usar palavras que não tenham sentido dúbio.

FASES

No capítulo anterior vimos que o algoritmo é uma seqüência lógica de instruções que podem ser executadas. É importante ressaltar que, qualquer tarefa que siga determinado padrão, pode ser descrita por um algoritmo, como por exemplo: COMO FAZER ARROZ DOCE ou CALCULAR O SALDO FINANCEIRO DE UM ESTOQUE.

Entretanto ao montar um algoritmo, precisamos primeiro dividir o problema apresentado em três fases fundamentais.

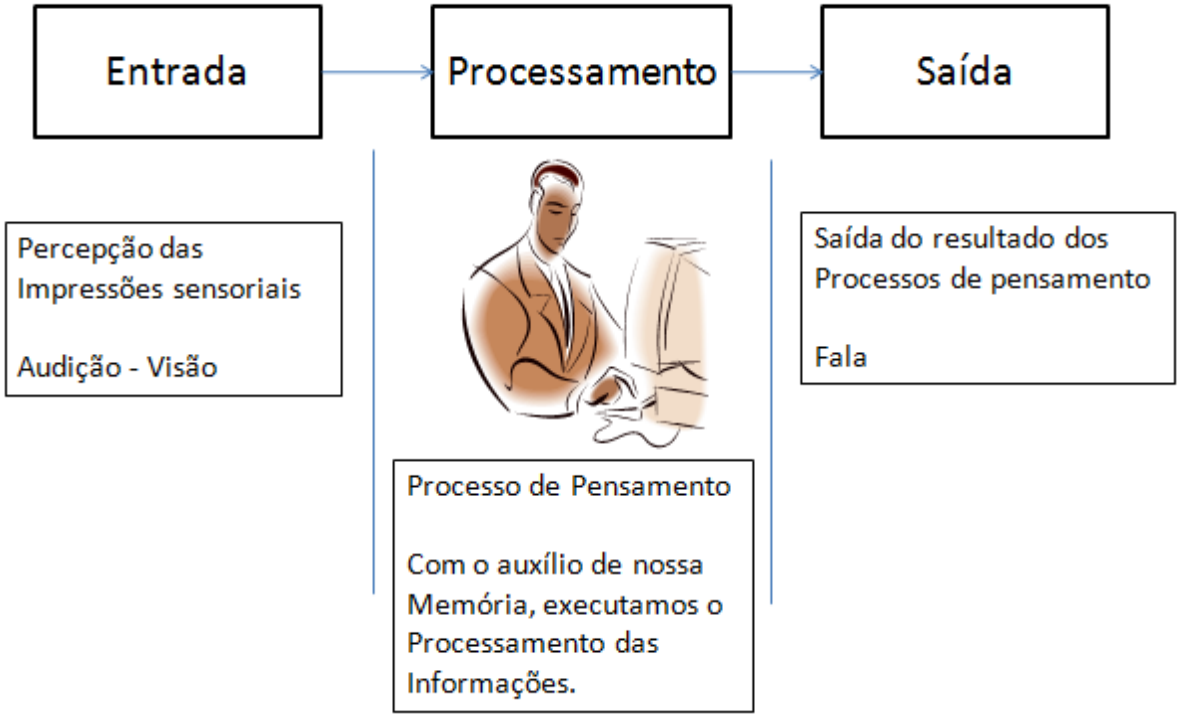
Onde temos:

ENTRADA: São os dados de entrada do algoritmo.

PROCESSAMENTO: São os procedimentos utilizados para chegar ao resultado final.

SAÍDA: São os dados já processados.

Analogia com o homem



EXEMPLO DE ALGORITMO

Imagine o seguinte problema: calcular a média final dos alunos da 3° série. Os alunos realizaram quatro provas: P1, P2, P3 e P4. Onde:

$$\text{Média Final} = \frac{P1 + P2 + P3 + P4}{4}$$

Para montar o algoritmo proposto, faremos três perguntas:

1) Qual são os dados de Entrada ?
2) Qual será o processamento a ser utilizado ?
3) Quais serão os dados de saída ?

1) Os dados de entrada são P1, P2, P3 e P4.

2) O procedimento será somar os dados de entrada e dividir por 4.
3) Os dados de saída serão a média final.

Algoritmo

1) Receba a nota da prova 1.
2) Receba a nota da prova 2.
3) Receba a nota da prova 3.
4) Receba a nota da prova 4.
5) Some todas as notas e divida por 4.
6) Mostre o resultado da divisão.

EXERCÍCIOS

1) Identifique os dados de entrada, processamento e saída no algoritmo abaixo:
   - Receba o código da peça
   - Receba valor da peça
   - Receba quantidade de peças
   - Calcule o valor total da peça (Quantidade * Valor da Peça)
   - Mostre o código da peça e seu valor total.

2) Defina os dados de entrada saída e processamento:
   a) Construção de uma casa.
   b) Realização do curso de programação.
   c) Ter uma árvore no quintal.

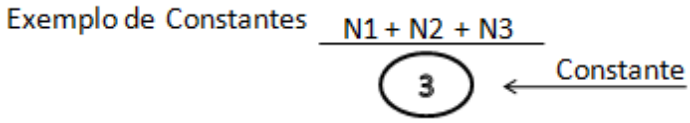
CONSTANTES, VARIÁVEIS E TIPOS DE DADOS

Variáveis e constantes são os elementos básicos que um programa manipula. Uma variável é um espaço reservado na memória do computador para armazenar um tipo de dado determinado.

Variáveis devem receber nomes para poderem ser referenciadas e modificadas quando necessário. Um programa deve conter declarações que especificam de que tipo são as variáveis que ele utilizará e as vezes um valor inicial. Tipos podem ser por exemplo: inteiros, decimais, caracteres, etc. As expressões combinam variáveis e constantes para calcular novos valores.

CONSTANTES

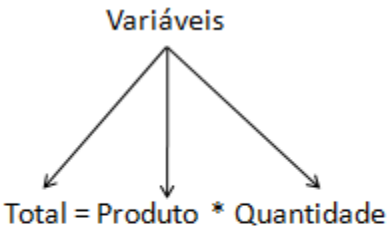
Constante é um determinado valor fixo que não se modifica ao longo do tempo, durante a execução de um programa. Conforme o seu tipo, a constante é classificada como sendo decimal, inteira, boleana e caracter.

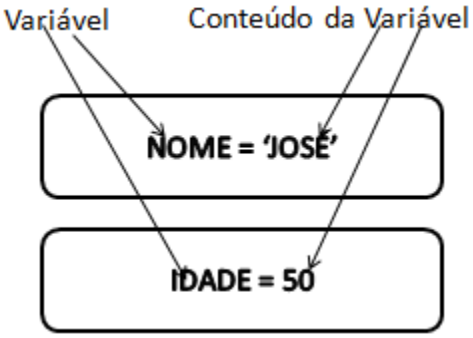


VARIÁVEIS

Variável é a representação simbólica dos elementos de um certo conjunto. Cada variável corresponde a uma posição de memória, cujo conteúdo pode se alterado ao longo do tempo durante a execução de um programa. Embora uma variável possa assumir diferentes valores, ela só pode armazenar um valor a cada instante.

Exemplos de variáveis:





TIPOS DE VARIÁVEIS

Inteiro: são aqueles que não possuem componentes decimais ou fracionários, podendo ser positivos ou negativos.

Exemplos:

10 número inteiro positivo

-10 número inteiro negativo

Decimal: são aqueles que podem possuir componentes decimais ou fracionários, podendo também ser positivos ou negativos.

Exemplos:

25.03 número real positivo com duas casas decimais

235. número real positivo com zero casas decimais

-10.5 número real negativo com uma casa decimal

Cadeia ou String ou Alfanuméricas: são aqueles que possuem letras e/ou números. Pode em determinados momentos conter somente dados numéricos ou somente letras. Se usado somente para armazenamento de números, não poderá ser utilizado para operações matemáticas.

Exemplos:

“Maria” String de comprimento 5

“123” String de comprimento 3

“A” String de comprimento 1

Lógico também conhecido como booleano. É representado no algoritmo pelos dois únicos valores lógicos possíveis: verdadeiro ou falso. Porém é comum encontrar em outras referências outros pares de valores lógicos como: sim/não, 1/0, true/false, verdadeiro/falso.

DECLARAÇÃO DAS VARIÁVEIS

As variáveis só podem armazenar valores de um mesmo tipo, de maneira que também são classificadas como sendo decimais, inteiras, boleanas e alfanumérica.

EXERCÍCIOS

1) O que é uma constante? Dê dois exemplos.
2) O que é uma variável? Dê dois exemplos.
3) Identifique os tipos de dados, sendo (I) Inteiro, (D)Decimal, (A)Alfanumérico e (B)Boleana:

( ) 1.23 ( ) 'tomates' ( ) Verdadeiro ( ) 1 + 2.1

( ) 'Falso' ( ) '1' ( ) 3 – 2 ( ) '1 + 1'

4) Identifique o tipo de dados para cada uma das situações abaixo:
   a) Nome do aluno.
   b) Saldo da conta bancária.
   c) Idade do cliente do mercado.
   d) Nota do aluno.

OPERADORES

Os operadores são meios pelo qual incrementamos, decrementamos, comparamos e avaliamos dados dentro do computador. Temos três tipos de operadores:

· Operadores Aritméticos

· Operadores Relacionais

· Operadores Lógicos

OPERADORES ARITMÉTICOS

Os operadores aritméticos são os utilizados para obter resultados numéricos. Além da adição, subtração, multiplicação e divisão, podem utilizar também o operador para exponenciação. Os símbolos para os operadores aritméticos são:

OPERAÇÃO	SIMBOLO
Adição	+
Subtração	-
Multiplicação	*
Divisão	/

Exponenciação	**

Hierarquia das Operações Aritméticas

1 º ( ) Parênteses

2 º Exponenciação

3 º Multiplicação, divisão (o que aparecer primeiro)

4 º + ou – (o que aparecer primeiro)

Exemplo:

TOTAL = PRECO * QUANTIDADE

1 + 7 * 2 ** 2 – 1 = 28

3 * (1 – 2) + 4 * 2 = 5

EXERCÍCIOS

1 – Resolva as expressões matemáticas:

a) 1 + 9 * 3
b) 2 / 1 * (3 + 1)
c) 2 ** 3 + (2 + (6 / 3))
d) 3 + 4 ** 2

VISUAL G

O VISUAL G é uma ferramenta criada para quem está iniciando seus aprendizados em Lógica Computacional ou Construção de Algoritmos Computacionais. Nele você poderá digitar seus comandos em português e ver seu programa funcionando.

Supondo que temos a seguinte problemática.

João tem um aviário e precisa do valor de suas vendas pela quantidade de ovos vendidos para cada cliente. Neste caso ele vai informar ao sistema a quantidade de ovos e o valor da unidade, sendo demonstrado o valor de venda, desta forma teremos o seguinte algoritmo:

```
Inteiro qtdeOvo;
Numerico valorUnid, valorVenda;

Escreva('Informe a quantidade de ovos vendidos: ');
Ler(qtdeOvo);
Escreva('Informe o valor da unidade R$: ');
Ler(valorUnid);
valorVenda = qtdeOvo * valorUnid;
Escreva('Valor da venda R$: ' + valorVenda);
```

Para resolução deste problema inicialmente foi verificado quais as entradas que seriam informadas pelo usuário, no caso a quantidade de ovos e o valor da unidade. Depois foi verificado o processamento a ser realizado (quantidade de ovos * preço da unidade) e por último demonstrado saída para o usuário final, que seria o preço da venda.

```
algoritmo "Aviario"
// Função : Calcular lucro do aviario
// Autor : Tiago da Rosa Valério
// Data : 21/04/2015
// Seção de Declarações
var
qtdeOvos : Inteiro
valorUnid, valorVenda : Numerico

inicio
// Seção de Comandos
Escreva("Informe a quantidade de ovos: ")
Leia(qtdeOvos)
Escreva("Informe o valor da unidade do ovo R$:")
Leia(valorUnid)
valorVenda <- (qtdeOvos * valorUnid)
Escreva("Receita do Aviário R$ : " , valorVenda)

fimalgoritmo
```

EXERCÍCIOS

1) Construa um algoritmo para solucionar a seguinte situação :

· Leia a cotação do dólar

· Leia um valor em dólares

· Converta esse valor para Real

· Mostre o resultado

2) Desenvolva um algoritmo que:

· Leia 4 (quatro) números

· Calcule o quadrado para cada um

· Somem todos e

· Mostre o resultado

3) Construa um algoritmo para pagamento de comissão de vendedores de peças, levando-se em consideração que sua comissão será de 5% do total da venda e que você tem os seguintes dados:

· Identificação do vendedor

· Código da peça

· Preço unitário da peça

· Quantidade vendida

OPERADORES RELACIONAIS

Os operadores relacionais são utilizados para comparar String de caracteres e números. Os valores a serem comparados podem ser constantes ou variáveis.

Estes operadores sempre retornam valores lógicos (verdadeiro ou falso/ True ou False). Para estabelecer prioridades no que diz respeito a qual operação executar primeiro, utilize os parênteses.

Os operadores relacionais são:

DESCRIÇÃO	SIMBOLO
Igual a	'='
Diferente de	<> ou #
Maior que	>
Menor que	<
Maior ou Igual a	>=
Menor ou Igual	<=

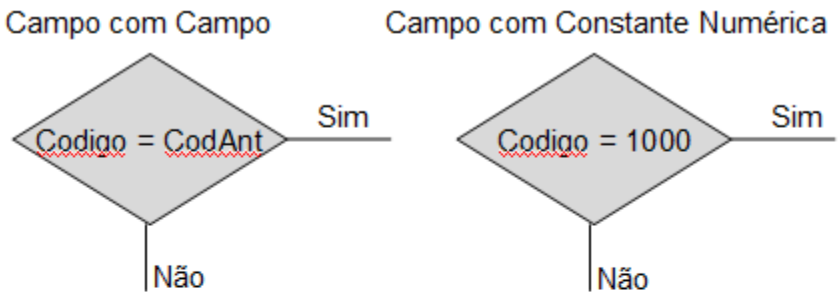
Exemplo:

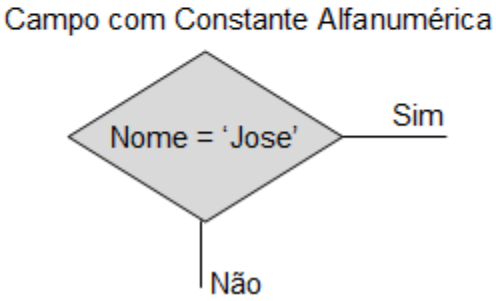
Tendo duas variáveis A = 5 e B = 3. Os resultados das expressões seriam:

OPERAÇÃO	SIMBOLO
A = B	False
A <> B	True
A > B	True
A < B	False

A >= B	True
A <= B	False

Símbolo Utilizado para comparação entre expressões.





OPERADORES LÓGICOS

Os operadores lógicos servem para combinar resultados de expressões, retornando se o resultado final é verdadeiro ou falso.

Os operadores lógicos são:

E	AND
OU	OR
NÃO	NOT

E / AND - Uma expressão AND (E) é verdadeira se todas as condições forem verdadeiras.

OR/OU - Uma expressão OR (OU) é verdadeira se pelo menos uma condição for verdadeira.

NOT - Um expressão NOT (NÃO) inverte o valor da expressão ou condição, se verdadeira inverte para falsa e vice-versa.

A tabela abaixo mostra todos os valores possíveis criados pelos três operadores lógicos (AND, OR e NOT).

1º Valor	Operador	2º Valor	Resultado
T	AND	T	T
T	AND	F	F
F	AND	T	F
F	AND	F	F
T	OR	T	T
T	OR	F	T
F	OR	T	T
F	OR	F	F
T	NOT		F
F	NOT		T

Exemplos:

Suponha que temos três variáveis A = 5, B = 8 e C =1. Os resultados das expressões seriam:

Expressões			Resultado
A = B	AND	B > C	False
A <> B	OR	B < C	True
A > B	NOT		True
A < B	AND	B > C	True
A >= B	OR	B = C	False
A <= B	NOT		False

EXERCÍCIOS

1) Tendo as variáveis SALARIO, IR e SALLIQ, e considerando os valores abaixo. Informe se as expressões são verdadeiras ou falsas.

SALARIO	IR	SALLIQ	EXPRESSÃO	V ou F
100	0	100	(SALLIQ >= 100)	
200	10	190	(SALLIQ < 190)	
300	15	285	SALLIQ = SALARIO - IR	

2) Sabendo que A=3, B=7 e C=4, informe se as expressões abaixo são verdadeiras ou falsas.

a) (A+C) > B ( )

b) B >= (A + 2) ( )

c) C = (B –A) ( )

d) (B + A) <= C ( )

e) (C+A) > B ( )

3) Sabendo que A=5, B=4 e C=3 e D=6, informe se as expressões abaixo são verdadeiras ou falsas.

a) (A > C) AND (C <= D) ( )

b) (A+B) > 10 OR (A+B) = (C+D) ( )

c) (A>=C) AND (D >= C) ( )

ESTRUTURA DE DECISÃO E REPETIÇÃO

Como vimos no capítulo anterior em “Operações Lógicas”, verificamos que na maioria das vezes necessitamos tomar decisões no andamento do algoritmo. Essas decisões interferem diretamente no andamento do programa. Trabalharemos com dois tipos de estrutura. A estrutura de Decisão e a estrutura de Repetição.

COMANDOS DE DECISÃO

Os comandos de decisão ou desvio fazem parte das técnicas de programação que conduzem a estruturas de programas que não são totalmente seqüenciais. Com as instruções de SALTO ou DESVIO pode-se fazer com que o programa proceda de uma ou outra maneira, de acordo com as decisões lógicas tomadas em função dos dados ou resultados anteriores. As principais estruturas de decisão são: “Se Então”, “Se então Senão” e “Caso Selecione”.

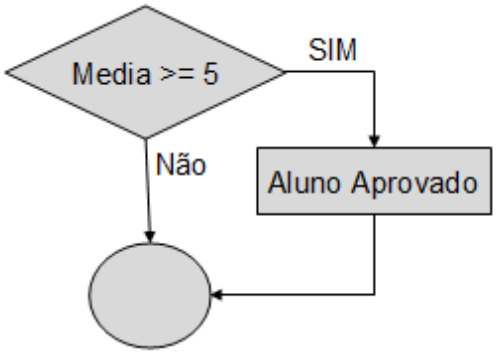
SE ENTÃO / IF ... THEN

A estrutura de decisão “SE/IF” normalmente vem acompanhada de um comando, ou seja, se determinada condição for satisfeita pelo comando SE/IF então execute determinado comando.

Imagine um algoritmo que determinado aluno somente estará aprovado se sua média for maior ou igual a 5.0, veja no exemplo de algoritmo como ficaria.

SE MEDIA >= 5.0 ENTÃO ALUNO APROVADO

Em diagrama de blocos ficaria assim:



Em Visual Basic:

IF MEDIA >= 5 Then

Text1 = “APROVADO”

ENDIF

Com estas estruturas poderemos executar determinadas ações conforme a necessidade. Podemos imaginar o que ocorreu no Japão. Vamos construir uma algoritmo que seja informado o índice de abalos sísmicos por um técnico e se chegar a determinada escala que seja gerado alerta máximo.

Numerico indiceSismico;

Escreva(‘Informe o índice de atividades sísmicas:’);

Ler(indiceSismico);

Se indiceSismico >= 10 então

Escreva(‘Alerta máximo! Perigo de tsunami!!!’);

Fim se;

Visualg

```
algoritmo "Alerta_Tsunami"
// Função : Avisar população do perigo de Tsunami
// Autor : Tiago Valério
// Data : 26/04/2015
// Seção de Declarações
var
indiceSismico : Numerico

inicio
// Seção de Comandos
Escreva("Informe o indice de atividades sismicas: ")
Leia(indiceSismico)

se indiceSismico >= 10 entao
     Escreva("Alerta Máximo. Perigo de Tsunami!!!")
fimse

fimalgoritmo
```

Exercícios

1) Faça um algoritmo que receba um número e mostre uma mensagem caso este número seja maior que 10.

2) Faça um algoritmo que receba um número e diga se este número está no intervalo entre 100 e 200.

3) João Papo-de-Pescador, comprou um microcomputador para controlar o rendimento diário de seu trabalho. Toda vez que ele traz um peso de peixes maior que o estabelecido pelo regulamento de pesca do estado de São Paulo (50 quilos) deve pagar uma multa de R$ 4,00 por quilo excedente. O algoritmo deve verificar se excedeu a quantidade permitida, neste caso deve retornar a quantidade excedida e o valor de multa a ser paga.

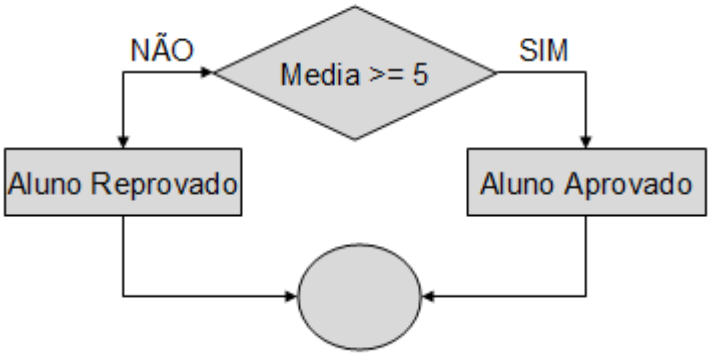
SE ENTÃO SENÃO / IF ... THEN ... ELSE

A estrutura de decisão "SE/ENTÃO/SENÃO", funciona exatamente como a estrutura "SE", com apenas uma diferença, em "SE" somente podemos executar comandos caso a condição seja verdadeira, diferente de "SE/SENÃO" pois sempre um comando será executado independente da condição, ou seja, caso a condição seja "verdadeira" o comando da condição será executado, caso contrário o comando da condição "falsa" será executado.

Em algoritmo ficaria assim:

```
SE MÉDIA >= 5.0 ENTÃO
    ALUNO APROVADO
SENÃO
    ALUNO REPROVADO
FIM SE
```

Em diagrama:



Em Visual Basic

IF MEDIA >= 5 Then

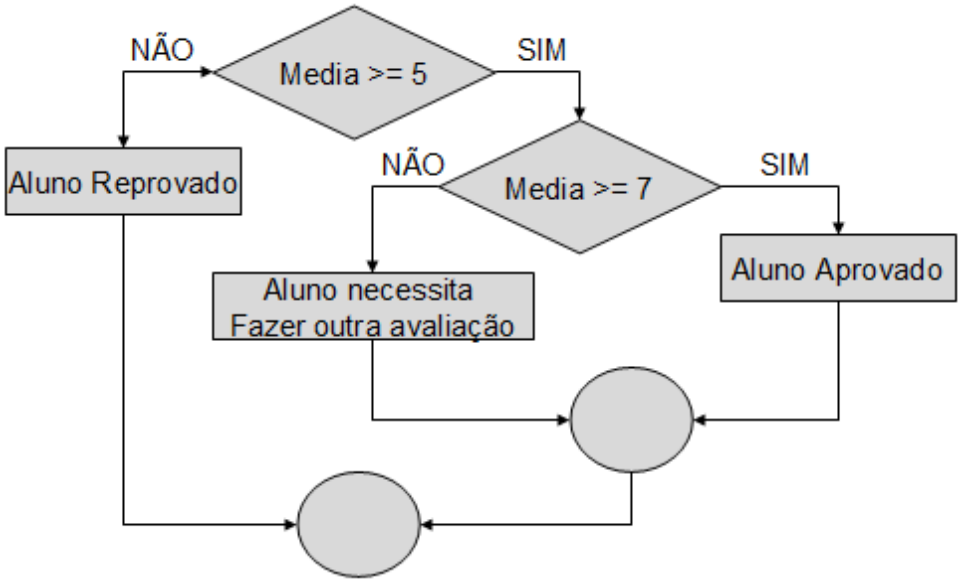
Text1 = "APROVADO"

ELSE

Text1 = "REPROVADO"

ENDIF

No exemplo acima está sendo executada uma condição que, se for verdadeira, executa o comando "APROVADO", caso contrário executa o segundo comando "REPROVADO". Podemos também dentro de uma mesma condição testar outras condições. Como no exemplo abaixo:



Em Visual Basic

IF MEDIA >= 5 Then

IF MEDIA >= 7.0 then

Text1 = “Aluno APROVADO”

ELSE

Text1 = “Aluno Necessita fazer outra Avaliação”

ENDIF

ELSE

Text1 = “Aluno REPROVADO”

ENDIF

Vamos melhorar nosso exemplo anterior, agora vamos adicionar uma mensagem, caso o índice sísmico não representa problemas:

Numerico indiceSismico;

Escreva(‘Informe o índice de atividades sísmicas:’);

Ler(indiceSismico);

Se indiceSismico >= 10 então

Escreva(‘Alerta máximo! Perigo de tsunami!!!’);

Senão

Escreva(‘Indice não apresenta problemas.’);

Fim se;

VisualG

```
algoritmo "Alerta_Tsunami"
// Função : Avisar população do perigo de Tsunami
// Autor : Tiago Valério
// Data : 26/04/2015
// Seção de Declarações
var
indiceSismico : Numerico

inicio
// Seção de Comandos
Escreva("Informe o indice de atividades sismicas: ")
Leia(indiceSismico)

se (indiceSismico >= 10) entao
     Escreva("Alerta Máximo. Perigo de Tsunami!!!")
senao
     Escreva("Indice não apresenta problemas...")
fimse

fimalgoritmo
```

EXERCÍCIOS

1) Implemente um algoritmo para calcular o imposto de renda de um trabalhador. No caso o sistema deverá solicitar a informação do salário do funcionário. Se o salário for superior a R$: 3000,00 deve pagar 15% de imposto para os demais valores deve ser 2%.

2) Faça um algoritmo que leia um número inteiro e mostre uma mensagem indicando se este número é par ou ímpar, e se é positivo ou negativo.

3) Escrever um algoritmo que leia quatro valores inteiro distintos e informe qual é o maior e o menor valor.

4) Tendo como dados de entrada a altura e o sexo de uma pessoa, construa um algoritmo que calcule seu peso ideal, utilizando as seguintes fórmulas:

Para homens: (72.7*h) - 58

Para mulheres: (62.1*h) - 44.7 (h = altura)

5) Escrever um algoritmo que leia três valores inteiros e verifique qual o tipo de triângulo que eles formam: equilátero, isóscele ou escaleno.

Triângulo Equilátero: aquele que tem os comprimentos dos três lados iguais;

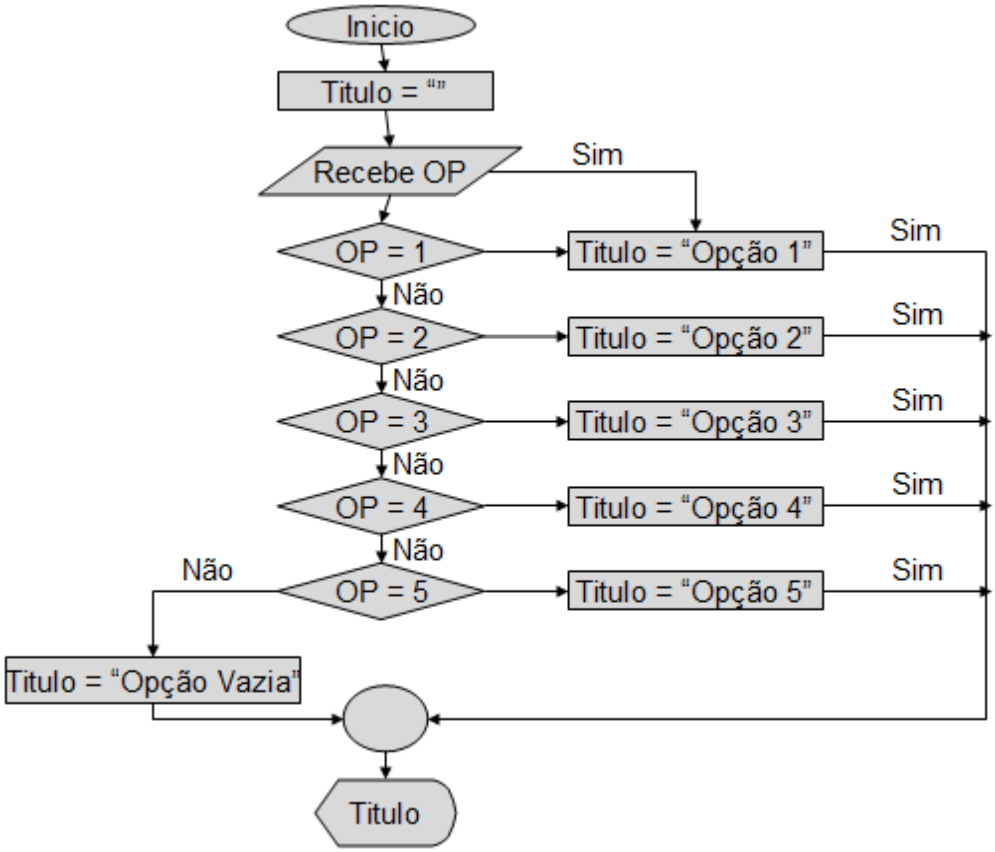
Triângulo Isóscele: aquele que tem os comprimentos de dois lados iguais. Portanto, todo triângulo equilátero é também isóscele;

Triângulo Escaleno: aquele que tem os comprimentos de seus três lados diferentes.

CASO SELECIONE / SELECT ... CASE

A estrutura de decisão CASO/SELECIONE é utilizada para testar, na condição, uma única expressão, que produz um resultado, ou, então, o valor de uma variável, em que está armazenado um determinado conteúdo. Compara-se, então, o resultado obtido no teste com os valores fornecidos em cada cláusula "Caso".

No exemplo do diagrama de blocos abaixo, é recebido uma variável "**Op**" e testado seu conteúdo, caso uma das condições seja satisfeita, é atribuído para a variável Titulo a String "Opção X", caso contrário é atribuído a string "Opção Errada".



Em Visual Basic utilizamos a seguinte seqüência de comandos para representar o diagrama anterior.

```
TITULO = “”
OP = INPUTBOX(“DIGITE A OPÇÃO”)
SELECT CASE OP
    CASE 1
        TITULO = “OPÇÃO 1”
    CASE 2
        TITULO = “OPÇÃO 2”
    CASE 3
        TITULO = “OPÇÃO 3”
    CASE 4
        TITULO = “OPÇÃO 4”
    CASE 5
        TITULO = “OPÇÃO 5”
    CASE ELSE
        TITULO = “OPÇÃO ERRADA”
END SELECT
LABEL1.CAPTION = TITULO
```

VisualG

```
algoritmo "opcoes"
// Função : Verificar opção informada pelo usuario
// Autor : Tiago Valério
// Data : 26/04/2015
// Seção de Declarações
var
opcao : Inteiro
inicio
// Seção de Comandos
Escreva("Informa opção de 1 - 5: ")
Leia(opcao)
escolha(opcao)
  caso 1
    Escreva("Opção Hum")
  caso 2
    Escreva("Opção dois")
  caso 3
    Escreva("Opção três")
  caso 4
    Escreva("Opção quatro")
  caso 5
    Escreva("Opção cinco")
  outrocaso
    Escreva("Opção inválida")
  fimescolha
fimalgoritmo
```

Exercícios

1) Elabore um algoritmo que dada a idade de um nadador classifique-o em uma das seguintes categorias:

Infantil A = 5 a 7 anos

Infantil B = 8 a 11 anos

Juvenil A = 12 a 13 anos

Juvenil B = 14 a 17 anos

Adultos = Maiores de 18 anos

2) Faça um algoritmo que calcule o valor da conta de luz de uma pessoa. Sabe-se que o cálculo da conta de luz segue a tabela abaixo:

Tipo de Cliente Valor do KW/h

1 (Residência) 0,60

2 (Comércio) 0,48

3 (Indústria) 1,29

3) A escola “APRENDER” faz o pagamento de seus professores por hora/aula. Faça um algoritmo que calcule e exiba o salário de um professor. Sabe-se que o valor da hora/aula segue a tabela abaixo:

Professor Nível 1 R$12,00 por hora/aula

Professor Nível 2 R$17,00 por hora/aula

Professor Nível 3 R$25,00 por hora/aula

COMANDOS DE REPETIÇÃO

Utilizamos os comandos de repetição quando desejamos que um determinado conjunto de instruções ou comandos sejam executados um número definido ou indefinido de vezes, ou enquanto um determinado estado de coisas prevalecer ou até que seja alcançado.

Trabalharemos com modelos de comandos de repetição:

· Enquanto x, processar (**Do While ...Loop**);

· Processar ..., Até que x (**Do ... Loop Until**)

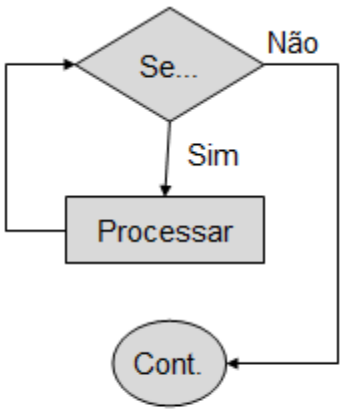
· Para ... Até ... Seguinte (**For ... To ... Next**)

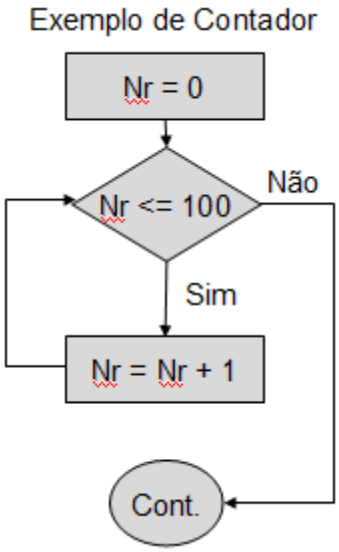
ENQUANTO X, PROCESSAR (Do While ... Loop)

Neste caso, o bloco de operações será executado enquanto a condição x for verdadeira. O teste da condição será sempre realizado antes de qualquer operação.

Enquanto a condição for verdadeira o processo se repete. Podemos utilizar essa estrutura para trabalharmos com contadores.

Em diagrama de bloco a estrutura é a seguinte:





Em Visual Basic:

Nr = 0

Do While Nr <= 100

Nr = Nr + 1

Loop

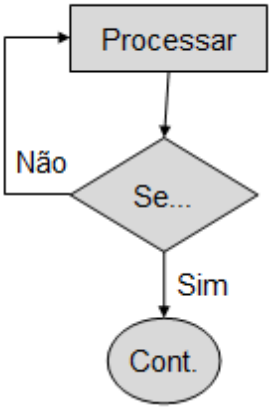
```
algoritmo "enquanto_proc"
// Função :
// Autor :
// Data : 26/04/2015
// Seção de Declarações
var
numero : inteiro
inicio
// Seção de Comandos
numero <- 0
enquanto numero <= 100 faca
         numero <- numero + 1
fimenquanto
fimalgoritmo
```

EXERCÍCIOS

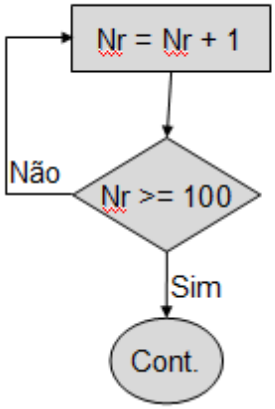
1) Faça um algoritmo que determine o maior entre N números. A condição de parada é a entrada de um valor 0, ou seja, o algoritmo deve ficar calculando o maior até que a entrada seja igual a 0 (ZERO).

PROCESSAR ... ATÉ QUE X (Do ... Loop Until)

Neste caso, executa-se primeiro o bloco de operações e somente depois é realizado o teste de condição. Se a condição for verdadeira, o fluxo do programa continua normalmente. Caso contrário é processado novamente os comandos antes do teste da condição. Em diagrama de Bloco



Exemplo de até diagrama



Em Visual Basic

nr = 0

Do

nr = nr + 1

Loop Until nr >= 100

Label1.Caption = nr

```
algoritmo "ateque"
// Função :
// Autor :
// Data : 26/04/2015
// Seção de Declarações
var
numero : inteiro
inicio
// Seção de Comandos
numero <- 0
repita
      numero <- numero + 1
ate numero = 100
fimalgoritmo
```

Exercícios

1) Construa um algoritmo que recebe números, sendo que a sua condição de para é valor 0. Depois deverá ser demonstrada a soma dos números informados pelos usuários.

PARA X ATE Y FAÇA (FOR ... NEXT)

Esta estrutura de repetição é a mais utilizada em linguagens de programação, a principal diferença é na sua declaração que indicamos o valor inicial da variável, sua condição de parada e como a mesma será implementada. No exemplo abaixo, iremos imprimir 10 vezes o nome do curso Cidadãos Ligados na Rede.

```
algoritmo "para"
// Função :
// Autor :
// Data : 26/04/2015
// Seção de Declarações
var
numero : inteiro
inicio
// Seção de Comandos
Para numero de 1 ate 10 faca
       Escreval("Cidadãos Ligados na Rede")
fimpara
fimalgoritmo
```

EXERCÍCIOS

1) Escrever um algoritmo que leia o nome e o sexo de 6 pessoas e informe o nome e se ela é homem ou mulher. No final informe total de homens e de mulheres.

Teste seus conhecimentos

1) Implemente algoritmo que receba o nome do usuário e o seu ano de nascimento. Deverá ser demonstrado ao usuário a idade atual e quantos anos ele terá em 2050.

2) Faça um algoritmo que conte de 1 a 100 e a cada múltiplo de 10 emita uma mensagem:

“Múltiplo de 10”.

3) Elabore um algoritmo que gera e escreve os números ímpares dos números lidos entre 100 e 200.

4) Elabore um algoritmo que receba a quantidade de KM e informe o valor correspondente em metros.

5) Elabore um algoritmo e recebe o nome e idade de uma pessoa, se a pessoa for maior de idade 18 anos, deve demonstrar o nome da pessoa e a mensagem que é maior de idade, caso contrário deve demonstrar a mensagem com o nome e idade da pessoa, mas informando que a mesma não é maior de idade.