Introdução à Lógica de Programação

Tecbh.blogspot.com.br

E-mail – contatotecbh@gmail.com

Índice

Definição de lógica 3
Algoritimizando a lógica 3
Exemplo de Algoritmo 3
Pseudo-Liguagem ou Pseudo-Codigo 4
Tipos de dados 4
Constantes 5
Variável 5
Declaração de Variável 5
Atribuindo Valores as Variáveis 6
Operadores 6
Operadores Aritméticos 6
Operadores Relacionais ou Comparativos 6
Operadores Lógicos 7
Construindo Expressões 7
Modularização de Expressões 8
Prioridades 8
Comandos de Entrada e Saída 8
Primeiros Algoritmos 8
Estruturas Condicionais 14
SE (IF) 14
ESCOLHA CASO (SWITCH CASE) 16
Estruturas de repetição 17
Estrutura PARA FACA(FOR) 17
Estrutura ENQUANTO FACA (WHILE) 19
Vetor e Matriz 20

DEFINIÇÃO DE LÓGICA

Logica trata da correção do pensamento, como a filosofia, procura saber por que pensamos assim e não daquele jeito. Com a arte ou técnica, nos ensina a usar corretamente as leis do pensamento.

Poderíamos dizer também que a logica é a arte de pensar corretamente e, visto que a forma mais complexa do pensamento é o raciocínio, a logica estuda ou tem em vista a "correção do raciocínio". Podemos ainda dizer que logica tem em vista a "ordem da razão". Isto dá a entender que a nossa razão pode funcionar desordenadamente. Por isso também se pode dizer que a logica ensina a colocar Ordem no Pensamento.

ALGORITIMIZANDO A LÓGICA

Construir algoritmo é o objetivo fundamental de toda a programação, mas afinal o que é algoritmo?

Algoritmo é uma sequencia de passos que visam atingir um objetivo bem definido. Algoritmo é a descrição de um conjunto de ações que obedecidas, resultam numa sucessão finita de passos, atingindo o objetivo.

Em geral, um algoritmo destina-se a resolver um problema: fixa um padrão de comportamento a ser seguido, uma norma de execução a ser trilhada, com vista a alcançar, como resultado final a solução de um problema.

Resumindo algoritmo nada mais é que um passo a passo, de forma ordenada para executar uma determinada finalidade.

Exemplo de Algoritmo

Inicio

Objetivo: Fazer uma ligação

Passo 1: Pegar o telefone e destravar

Passo 2: Discar o numero desejado

Passo 3: Mandar chamar

Passo 4: Ser der sinal de chamar

4.1: Conversar

4.2: Desligar

4.3: Guardar Telefone

Passo 5: Senão

5.1 repetir

Fim

Este foi um exemplo simples de como criar um algoritmo, e todo programa é simplesmente isso um algoritmo em uma determinada linguagem de programação para executar uma tarefa cotidiana.

PSEUDO-LIGUAGEM OU PSEUDO-CODIGO

Este é um tipo de linguagem utilizada para aprendizado de logica de programação, geralmente na língua nativa de quem está aprendendo, facilitando o entendimento das funcionalidades do algoritmo. Aqui nesta apostila será utilizado a pseudo-linguagem portugol.

Uma pseudo-linguagem não pode ser interpretada por um ambiente computacional, o portugol e estudado em um programa chamado VISUAL G que pode ser facilmente tanto para PC quanto para celular, o link para baixar os dois estão disponíveis no blog, ou podem baixar no GOOGLE PLAY (Android Visual) e no BAIXAKI (Visual G). Também estará disponível o link do manual do programa visual g.

TIPOS DE DADOS

Antes começar a criar algoritmos é primordial entender algumas coisas como fundamentais, e uma dessas coisas são os tipos dados primitivos da logica de programação. Estes são divididos em 4 tipos.

Inteiro: Toda informação numérica que faça parte do conjunto de números inteiros, sendo ele negativo ou positivo.

Exemplos:
Ela tem **15** anos.
Hoje esta fazendo **-2** graus.

Real: Toda informação que faça parte do conjunto dos números reais, sendo ele negativo ou positivo.

Exemplos:
Minha altura é **1,73** cm.
Seu troco e R$ **5,65**.

Caractere: Toda e qualquer informação composta por um conjunto caracteres alfanuméricos incluindo caracteres especiais.

Exemplos:
Sua comissão é 10%

Logico: Toda informação que pode assumir apenas duas situações.

Exemplos:
Verdadeiro ou Falso
Ligado ou Desligado
0 ou 1

CONSTANTES

É definido como constante toda informação que não sofre nenhuma variação no decorrer do tempo.

Exemplo:

π 3,1416

VARIÁVEL

É definido como variável toda informação que pode ser alterada em algum instante no decorrer do tempo.

Exemplo:

Temperatura, peso

DECLARAÇÃO DE VARIÁVEL

As variáveis são armazenadas na memoria do ambiente computacional (memória ram), para declarar uma variável deve-se seguir uma regra.

Uma variável não pode começar com números e nem pode conter caracteres especiais, mas após a primeira letra pode-se colocar números.

Exemplos:

Declarações Validas	Declarações Invalidas
X, y, nome, x2	2x, nome@,a-b

Uma variável ao ser declarada pode ser utilizada durante o algoritmo varias vezes mas não pode ser repetida em uma declaração.

Além disso a variável deve também receber um tipo, ou seja, ao declarar uma variável além de dar o nome para ela e preciso informar ao sistema que tipo de informação ela vai receber.

Exemplos:

idade: inteiro;
nome: caractere;
peso,troco:real;
controle:logico;

Sempre se segue este conceito

Nome variável: tipo;

OBS: Depois de declarar uma ou uma lista de variáveis usa-se dois pontos (:) para dar o tipo da variável.

Atribuindo Valores as Variáveis

Para passar um valor para uma variável usa-se (<-) sinal de menor que mais traço, lembrando que uma variável só pode receber um valor de acordo com seu tipo, por exemplo, uma variável do tipo inteiro só pode receber numero inteiros.

Exemplo:

```
idade:inteiro;
idade<-15;
```

No exemplo dado a variável idade está recebendo o numero 15, uma variável pode receber outra variável do mesmo tipo.

Exemplo:

```
idade,anos:inteiro;
x<-15;
idade<-x;
```

Neste exemplo a variável x recebe o valor 15 e logo depois passa este valor para a variável idade.

OPERADORES

Operadores são símbolos que servem para efetuar operações aritméticas ou relacionais (comparativas) no algoritmo.

Operadores Aritméticos

Este é o conjunto de símbolos matemáticos que servem para efetuar operações matemáticas.

SIMBOLO	NOME
+	Adição
*	Multiplicação
**	Potenciação
-	Subtração
/	Divisão
//	Radiciação

Operadores Relacionais ou Comparativos

Este é o conjunto de símbolos que servem para comparar valores.

SIMBOLO	NOME
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a
=	Igual
<>	Diferente

Ainda existe o operador **mod** e **div** que só podem ser utilizados em números inteiros.

MOD – Retorna o resto de uma divisão

DIV – Retorna quociente da divisão inteira

Exemplo:

9 MOD 2 = 1

9 DIV 2 = 4

Operadores Lógicos

Estes operadores sempre irão retornar valores lógicos, ou seja, verdadeiro ou falso. São utilizados em expressões.

E (AND)- Retorna verdadeiro com as duas partes da expressão são verdadeiras, neste caso se qualquer uma das partes for falsa o resultado será falso.

OU (OR) – Retorna verdadeiro quando uma das partes da expressão for verdadeira, neste caso só é preciso que uma dê o valor verdadeiro em uma das expressões.

NÃO (NOT)– Inverte o resultado da operação, caso o resultado seja Verdadeiro ele passa ser falso e se for falso passa a ser verdadeiro.

CONSTRUINDO EXPRESSÕES

Quando se fala em expressões logo se lembra de matemática, e basicamente é isso mesmo, mas com uma diferença toda expressão feita em um algoritmo é linear. Na matemática temos colchetes chaves parentes frações e tudo mais, mas quando se fala de uma expressão em linguagem computacional só sobram os parênteses, falando assim e meio complicado de entender por e melhor trabalhar com exemplos.

$$\left[\frac{10}{2} + (5-3) + 1\right]$$

Tradicional

(2/3 + (5-3) +1)

Computacional

Modularização de Expressões

Quando se fala em modularização de expressão é a mesma coisa que falar em dividir a expressão em partes, proporcionando a resolução da expressão. Como já foi falado na computação é utilizado somente parênteses, mas também pode ser utilizados parênteses dentro de parênteses.

Exemplo:

(2+2)/2=2 Neste exemplo a operação e feita assim 2+2 e depois se divide por dois resolve primeiro o que esta dentro do parênteses e depois o que esta fora.

2+2/2=3 Neste caso a operação e feita assim 2/2 e depois soma 2 + 1 que daria o resultado 3.

Por este motivo a Modularização e muito importante, pois um pequeno erro muda todo o resultado final.

Prioridades

Na resolução das expressões aritméticas existe uma hierarquia para resolver.

1. Parênteses mais externos
2. Funções matemáticas
3. Operadores aritméticos
4. Operadores relacionais
5. Operadores lógicos

Será mais fácil de entender isso trabalhando com a construção de algoritmos.

Comandos de Entrada e Saída

Comando de entrada serve para receber uma informação, enquanto comandos de saída servem para retirar a informação ou exibir.

LER – comando de entrada serve para levar uma informação digitada para uma variável.

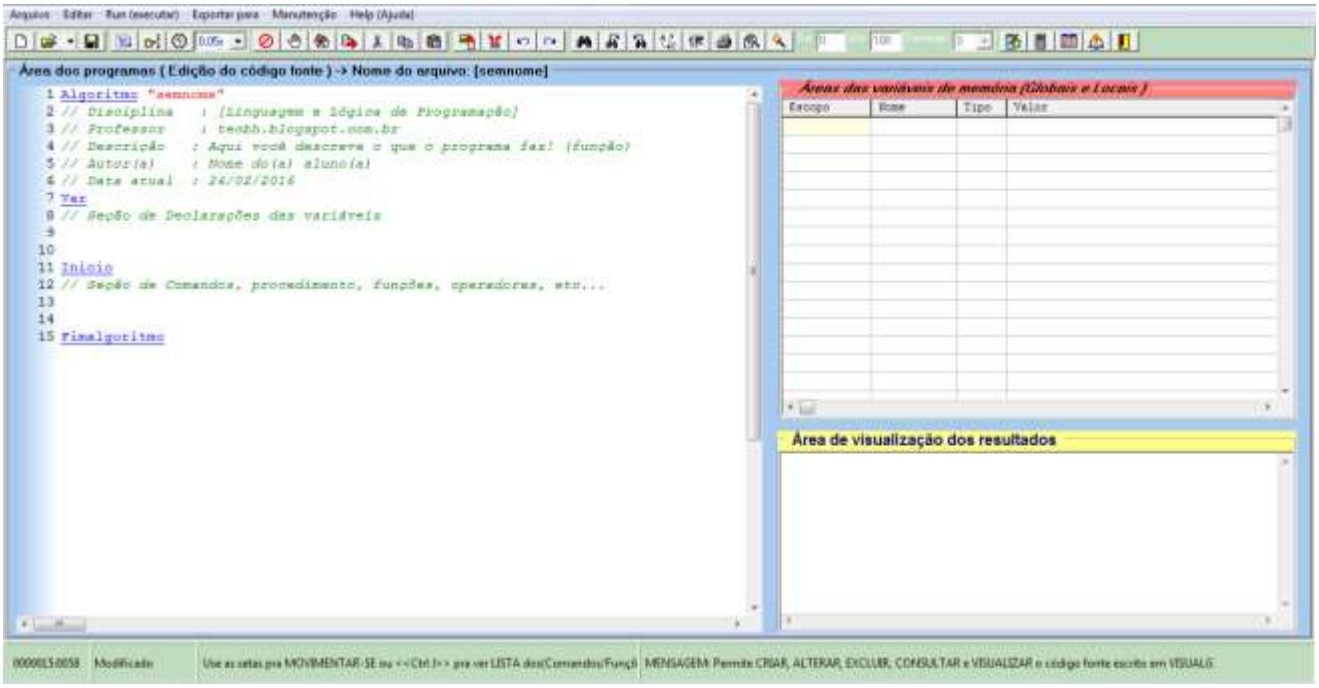
ESCREVA OU ESCREVAL – Comando de saída que serve para exibir uma informação na tela.

IMPRIMA – Comando de saída que serve para imprimir algo.

Com estas informações já é possível começar a escrever algoritmos.

PRIMEIROS ALGORITMOS

Abra o Visual G, este programa e autoexecutável e só extrair e depois dar dois clicks no aplicativo ele aparecera esta tela.



Da linha 2 até a linha 6 temos comentários, comentários são partes do código que não serão compiladas.

A linha 7 é onde começa o espaço para declaração das variáveis, não se pode declarar variáveis na linha 7 onde se encontra a palavra **var**

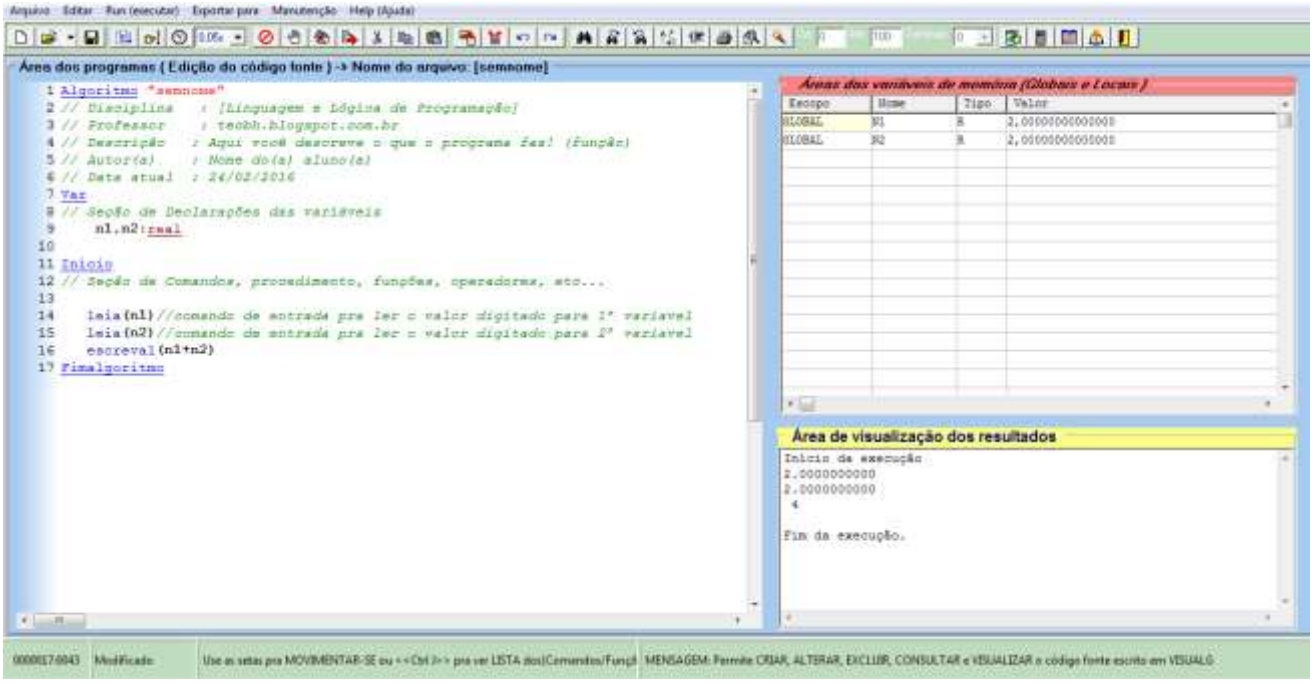
A linha 11 e onde começa o algoritmo as instruções ordenadas.

A linha 15 e onde termina o algoritmo, ou seja, nenhuma parte do algoritmo pode estar depois da palavra **fimalgoritmo**.

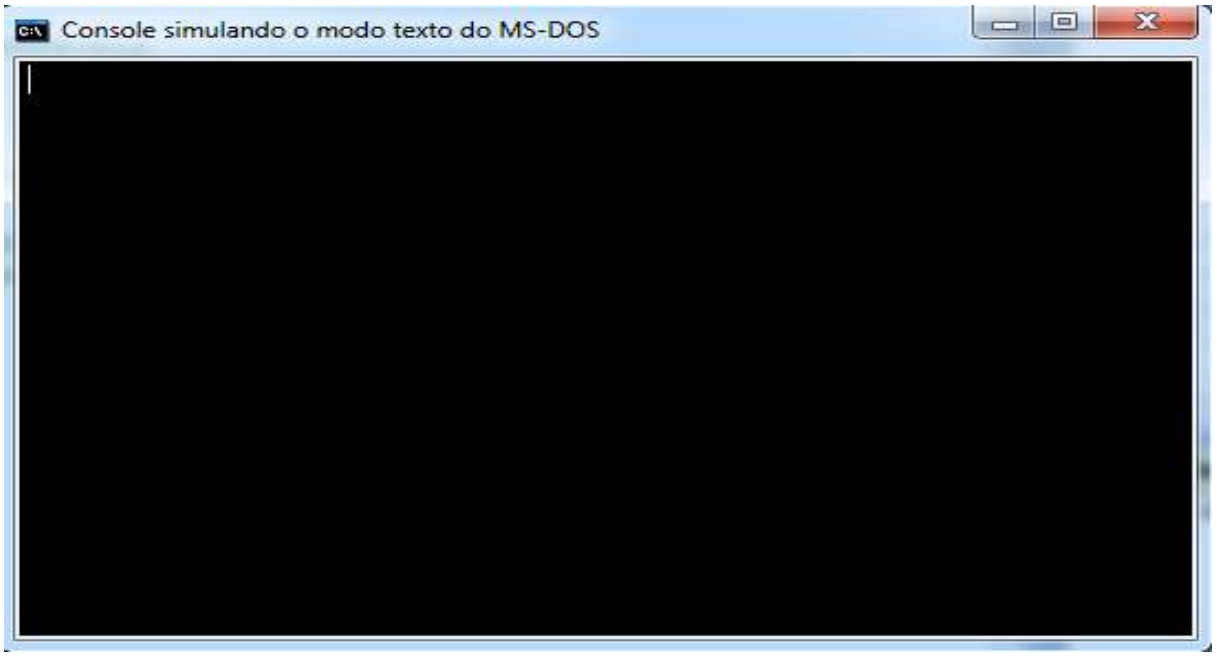
O objetivo deste primeiro algoritmo e pegar dois números e depois soma-los, pensando nisso é hora começar a pensar na logica do algoritmo. Este é bem simples e pegar dois números e soma-lo, as variáveis podem ser do tipo inteiro ou real. Veja como o algoritmo fica.

```
Algoritmo "semnome"
// Disciplina   : [Linguagem e Lógica de Programação]
// Professor    : tecbh.blogspot.com.br
// Descrição    : Aqui você descreve o que o programa faz! (função)
// Autor(a)     : Nome do(a) aluno(a)
// Data atual   : 24/02/2016
Var
// Seção de Declarações das variáveis
   n1,n2:real
```

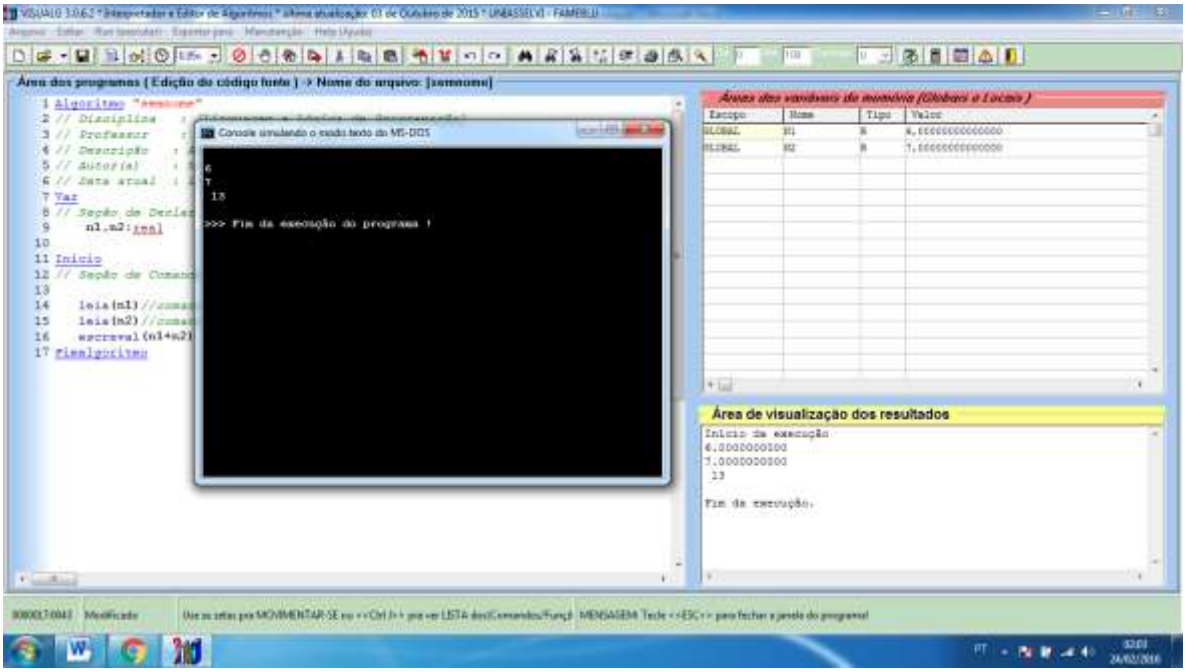
```
Inicio
// Seção de Comandos, procedimento, funções, operadores, etc...

   leia(n1)//comando de entrada pra ler o valor digitado para 1º variavel
   leia(n2)//comando de entrada pra ler o valor digitado para 2º variavel
   escreval(n1+n2)
Fimalgoritmo
```



Para executar o programa aperte a tecla F9 ira aparecer uma tela do MS DOS



Digite o primeiro numero aperte enter, depois o segundo numero e aperte enter que terá o resultado da soma.



Mas este simples algoritmo pode melhorar um pouco, adicione um escreval antes de cada leitura de variável.

escreval(“Digite o primeiro numero”)

leia(n1)

escreval(“Digite o segundo numero”)

leia(n2)

O texto escrito entre “” significa que é do tipo caractere (string), e sempre que tiver um texto ele deve estar entre aspas.

Área dos programas (Edição do código fonte) -> Nome do arquivo: [semnome]

```
1  Algoritmo "semnome"
2  // Disciplina   : [Linguagem e Lógica de Programação]
3  // Professor    : tecbh.blogspot.com.br
4  // Descrição    : Aqui você descreve o que o programa faz! (função)
5  // Autor(a)     : Nome do(a) aluno(a)
6  // Data atual   : 24/02/2016
7  Var
8  // Seção de Declarações das variáveis
9      n1,n2:real
10
11 Inicio
12 // Seção de Comandos, procedimento, funções, operadores, etc...
13    escreval("Digite o primeiro numero")
14    leia(n1)//comando de entrada pra ler o valor digitado para 1° variavel
15    escreval("Digite o segundo numero")
16    leia(n2)//comando de entrada pra ler o valor digitado para 2° variavel
17
18    escreval(n1+n2)
19 Fimalgoritmo
```

Execute novamente o algoritmo e veja a diferença.

```
Digite o primeiro numero
5,5
Digite o segundo numero
111
 116.5

>>> Fim da execução do programa !
```

O próximo algoritmo será bem simples também, ele ira pegar 3 números e calcular a media entre os 3.

Para calcular uma media em primeiro lugar devemos pegar os 3 valores e depois dividir esta soma por 3.

Veja como ele ficara:

```
Algoritmo "semnome"
// Disciplina   : [Linguagem e Lógica de Programação]
// Professor    : tecbh.blogspot.com.br
// Descrição    : Aqui você descreve o que o programa faz! (função)
// Autor(a)     : Nome do(a) aluno(a)
// Data atual   : 24/02/2016
Var
// Seção de Declarações das variáveis
n1,n2,n3,resultado:real
Inicio
// Seção de Comandos, procedimento, funções, operadores, etc...
escreval("Digite o primeiro numero")
// Mostra na tela o pedido pra digitar o numero
leia(n1)//recebe o numero digitado
escreval("Digite o segundo numero")
// Mostra na tela o pedido pra digitar o numero
leia(n2)//recebe o numero digitado
escreval("Digite o terceiro numero")
// Mostra na tela o pedido pra digitar o numero
leia(n3)//recebe o numero digitado
resultado<-n1+n2+n3
// a variavel resultado recebe o a soma dos valores de n1,n2,n3
escreval("A soma dos tres valores é ",resultado)
//exibe o resultado da soma,a virgula serve para concatenar os valores
resultado<- resultado/3
//Nesta parte a variavel resultado recebe o valor dela mesmo divido por 3
escreval("A media dos valores é ", resultado)
// Exibe o valor da media que está armazenado na variavel reaproveitada
Fimalgoritmo
```

Este algoritmo pode ser escrito de varias formas diferentes, como por exemplo, utilizando somente uma variável, ou retirando somente a variável resultado, ou ate mesmo deixando ela, mas fazendo parte final diferente. Agora veja o exemplo usando somente 2 variáveis.

```
1 Algoritmo "semnome"
2 // Disciplina   : [Linguagem e Lógica de Programação]
3 // Professor    : tecbh.blogspot.com.br
4 // Descrição    : Aqui você descreve o que o programa faz! (função)
5 // Autor(a)     : Nome do(a) aluno(a)
6 // Data atual   : 24/02/2016
7 Var
8 // Seção de Declarações das variáveis
9 n,res:real
10 Inicio
11 // Seção de Comandos, procedimento, funções, operadores, etc...
12 escreval("Digite o primeiro numero")
13 // Mostra na tela o pedido pra digitar o numero
14 leia(n)//recebe o numero digitado
15 res<-n
16 escreval("Digite o segundo numero")
17 // Mostra na tela o pedido pra digitar o numero
18 leia(n)//recebe o novo numero digitado
19  res<-res+n
20  // a variavel resultado recebe o valor que ja tem nela mais o novo digitado
21  escreval("Digite o terceiro numero")
22 // Mostra na tela o pedido pra digitar o numero
23 leia(n)//recebe o novo numero digitado
24  res<-(res+n)/3
25  // a variavel resultado recebe o valor que ja tem nela mais o novo digitado
26  // divido por 3
27  escreval("A media dos numeros digitados é ",res)
28 Fimalgoritmo
```

Voltando a falar de logica, uma da principais funções da logica de programação e otimizar seu código de forma que ele seja escrito de forma mais limpa (menos código), neste caso usando uma estrutura de repetição o código fica mais enxuto.

ESTRUTURAS CONDICIONAIS

As estruturas condicionais são as utilizadas para tomadas de decisões, estas sãos as estruturas SE e a estrutura CASO.

SE (IF)

A sintaxe desta estrutura funciona da seguinte forma

se <expressão-lógica> entao

<seqüência-de-comandos>

Fimse

Sempre que expressão logica da estrutura for verdadeira a sequencia de comandos dentro da estrutura e executada, caso seja falsa e estrutura e ignorada, neste exemplo foi o utilizado o SE simples existe a opção do SE composto.

se <expressão-lógica> entao

<seqüência-de-comandos>

senão

<seqüência-de-comandos>

Fimse

Nesta a regra é a mesma a única diferença é que quando a expressão logica for falsa será executada a sequencia de comandos dentro do **senao,** enquanto no exemplo anterior ele sai da estrutura.

OBS: Dentro de estrutura e possível adicionar outras estruturas.

Para os exemplos desta estrutura irei aproveitar o algoritmo feito anteriormente, imagine a situação.

Um professor precisa saber a media do aluno nos 4 bimestres e se a media for maior ou igual a 70 o aluno está aprovado se for abaixo disso o aluno está reprovado.

Exemplo SE:

ea dos programas (Edição do código fonte) -> Nome do arquivo: [media 2.alg]

```
11 // Seção de Comandos, procedimento, funções, operadores, etc...
12 escreval("Digite o primeiro numero")
13 // Mostra na tela o pedido pra digitar o numero
14 leia(n)//recebe o numero digitado
15 res<-n
16 escreval("Digite o segundo numero")
17 // Mostra na tela o pedido pra digitar o numero
18 leia(n)//recebe o novo numero digitado
19  res<-res+n
20  // a variavel resultado recebe o valor que ja tem nela mais o novo digitado
21  escreval("Digite o terceiro numero")
22 // Mostra na tela o pedido pra digitar o numero
23 leia(n)//recebe o novo numero digitado
24 res<-res+n
25  escreval("Digite o quarto numero")
26 // Mostra na tela o pedido pra digitar o numero
27 leia(n)//recebe o novo numero digitado
28  res<-(res+n)/4
29  // a variavel resultado recebe o valor que ja tem nela mais o novo digitado
30  // divido por 3
31  escreval("A media do aluno é ",res)
32  //Mostra a media do aluno
33  //------------------------------//
34  // nesta parte será adicionada estrutura se
35  se res >= 70 entao//inicio da estrutura
36  escreval("Aluno Aprovado")
37  fimse//fim da estrutura
38  se res<70 entao
39  escreval("Aluno reprovado")
40  fimse
41 Fimalgoritmo
```

Neste exemplo foi preciso criar duas estruturas para obter o resultado, este algoritmo pode ser simplificado utilizando o SE composto veja o exemplo.

Exemplo SE SENAO (composto):

```
10 Inicio
11 // Seção de Comandos, procedimento, funções, operadores, etc...
12 escreval("Digite o primeiro numero")
13 // Mostra na tela o pedido pra digitar o numero
14 leia(n)//recebe o numero digitado
15 res<-n
16 escreval("Digite o segundo numero")
17 // Mostra na tela o pedido pra digitar o numero
18 leia(n)//recebe o novo numero digitado
19  res<-res+n
20  // a variavel resultado recebe o valor que ja tem nela mais o novo digitado
21  escreval("Digite o terceiro numero")
22 // Mostra na tela o pedido pra digitar o numero
23 leia(n)//recebe o novo numero digitado
24 res<-res+n
25  escreval("Digite o quarto numero")
26 // Mostra na tela o pedido pra digitar o numero
27 leia(n)//recebe o novo numero digitado
28  res<-(res+n)/4
29  // a variavel resultado recebe o valor que ja tem nela mais o novo digitado
30  // divido por 3
31  escreval("A media do aluno é ",res)
32  //Mostra a media do aluno
33  //------------------------------//
34  // nesta parte será adicionada estrutura se
35  se res >= 70 entao//inicio da estrutura
36  escreval("Aluno Aprovado")
37  senao   // so será executada se a resposta logica for falsa
38  escreval("Aluno reprovado")
39  fimse//fim da estrutura
40 Fimalgoritmo
```

Repare que foi utilizado duas linhas de comando a menos no algoritmo anterior, ou seja, já deu uma melhorada na logica.

ESCOLHA CASO (SWITCH CASE)

A estrutura CASE tem a mesma função que o SE, mudando o jeito de sua sintaxe.

escolha <expressão-de-seleção>
caso <exp11>, <exp12>, ..., <exp1n>
<seqüência-de-comandos-1>
caso <exp21>, <exp22>, ..., <exp2n>
<seqüência-de-comandos-2>
...
outrocaso
<seqüência-de-comandos-extra>
Fimescolha

Exemplo:

```
1 Algoritmo "semnome"
2 // Disciplina: [Linguagem e Lógica de programaçao]
3 // Função :
4 // Autor :
5 // Data : 24/02/2016
6 // Seção de Declarações
7 var
8 nota : inteiro
9 inicio
10     escreva("Entre com a nota do aluno:")
11     leia(nota)
12     escolha nota
13     caso 0 ate 69
14         escreval("Reprovado.")
15     caso 70 ate 100
16         // A lista não precisa estar em uma ordem específica
17         // Só na cláusula ATE o primeiro valor precisam ser menor que o  segund
18         escreval("Aprovado.")
19     outrocaso
20         escreval("Nota inválida.")
21     fimescolha
22 fimalgoritmo
23
```

Neste foi simplificado os exemplos anteriores não foi feito o calculo das notas, para simplificar o algoritmo, mas nos vídeos vou mostrar com os cálculos de notas.

ESTRUTURAS DE REPETIÇÃO

Estruturas de repetição são utilizadas para criação loop finito, ou seja, o bloco de comando (sequencia de comando), vai ser executado até que a condição seja satisfeita.

Estrutura PARA FACA(FOR)

Está estrutura repete uma sequencia de comandos um determinado numero de vezes, veja a sintaxe da estrutura.

para <variável> de <valor-inicial> ate <valor-limite> [passo <incremento>] faca
<seqüência-de-comandos>
Fimpara

Exemplo:

```
1  Algoritmo "semnome"
2  // Disciplina   : [Linguagem e Lógica de Programação]
3  // Professor    : tecbh.blogspot.com.br
4  // Descrição    : Aqui você descreve o que o programa faz! (função)
5  // Autor(a)     : Nome do(a) aluno(a)
6  // Data atual   : 24/02/2016
7  Var
8  // Seção de Declarações das variáveis
9  i:inteiro
10 nota,resultado:real
11
12 Inicio
13 // Seção de Comandos, procedimento, funções, operadores, etc...
14 para i de 1 ate 4 faca
15       escreval("Digite a nota do",i,"° bimestre do aluno")
16       leia(nota)
17       resultado<-resultado+nota
18 fimpara
19 resultado<-resultado/4
20 escreval("A media final do aluno foi",nota)
21 se nota >= 70 entao
22 escreval("Aluno aprovado")
23 senao
24 escreval("Aluno reprovado")
25 fimse
26 Fimalgoritmo
```

Ainda utilizando o exemplo de notas de aluno, foi criado um algoritmo que faz a soma das notas depois tira a media e retorna se o aluno foi aprovado ou não, na primeira vez que foi utilizado este mesmo exemplo, o algoritmo foi escrito com mais ou menos 30 linhas de código neste exemplo utilizando a estrutura de repetição, com isso foi utilizado mais ou menos 12 linhas de código, ou seja, otimizamos o código.

OBS: Esta estrutura precisa ter uma variável de controle no exemplo foi utilizado a variável i, e ela deve ser do tipo inteiro. E caso o valor inicial seja maior que o valor final a estrutura não se executa, para que funcione com o valor inicial maior que final é preciso usar um PASSO -1, para ele decrementar.

Exemplo:

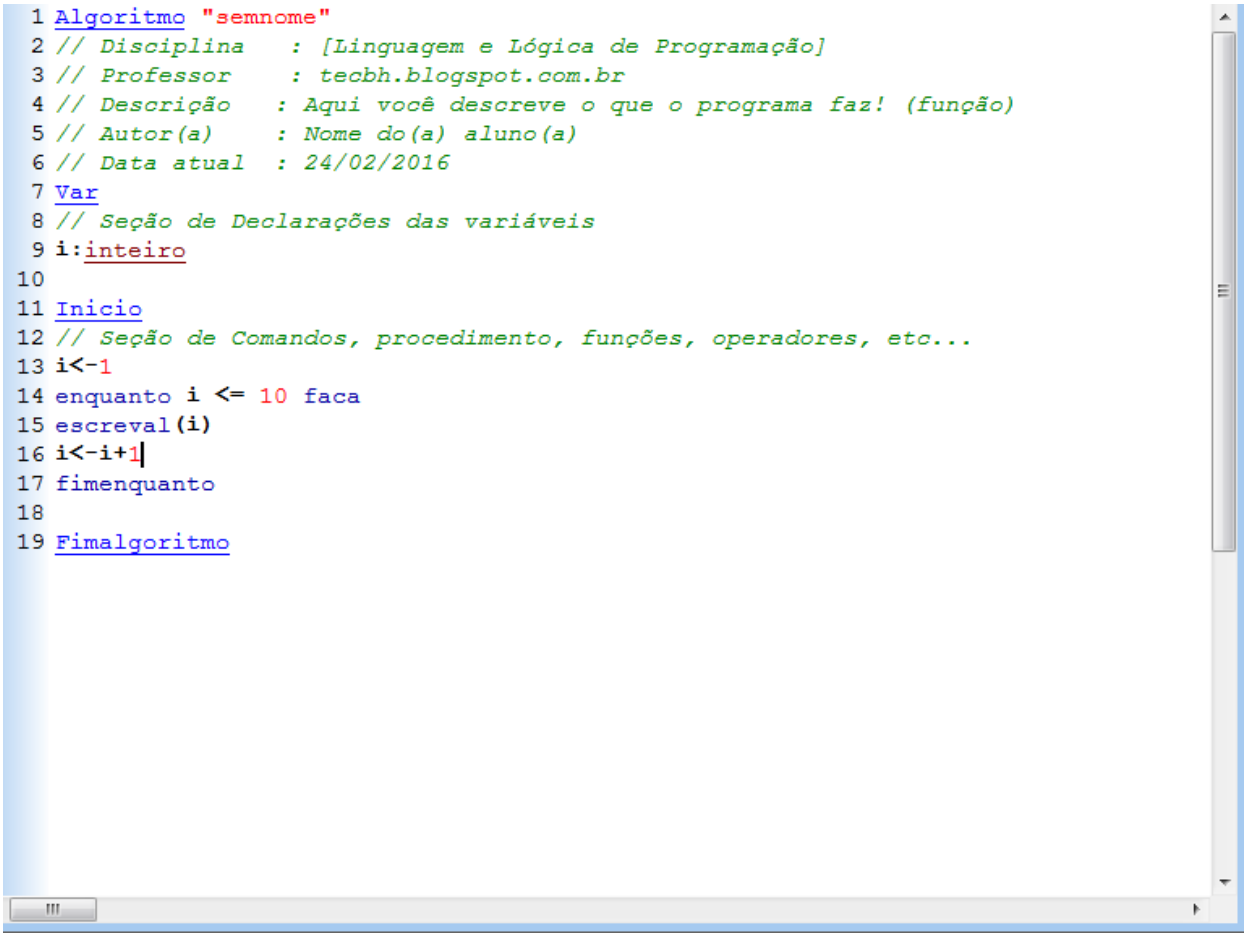
```
1  Algoritmo "semnome"
2  // Disciplina   : [Linguagem e Lógica de Programação]
3  // Professor    : tecbh.blogspot.com.br
4  // Descrição    : Aqui você descreve o que o programa faz! (função)
5  // Autor(a)     : Nome do(a) aluno(a)
6  // Data atual   : 24/02/2016
7  Var
8  // Seção de Declarações das variáveis
9  i:inteiro
10
11 Inicio
12 // Seção de Comandos, procedimento, funções, operadores, etc...
13          para i de 10 ate 1 passo -1 faca
14          escreval(i)
15          fimpara
16
17 Fimalgoritmo
```

Neste exemplo ele ira contar de 10 ate 1.

Estrutura ENQUANTO FACA (WHILE)

Está estrutura trabalha de forma que enquanto a expressão for verdadeira ela será executada.

enquanto <expressão-lógica> faca
<seqüência-de-comandos>
fimenquanto
Exemplo:



```
1 Algoritmo "semnome"
2 // Disciplina   : [Linguagem e Lógica de Programação]
3 // Professor    : tecbh.blogspot.com.br
4 // Descrição    : Aqui você descreve o que o programa faz! (função)
5 // Autor(a)     : Nome do(a) aluno(a)
6 // Data atual   : 24/02/2016
7 Var
8 // Seção de Declarações das variáveis
9 i:inteiro
10
11 Inicio
12 // Seção de Comandos, procedimento, funções, operadores, etc...
13 i<-1
14 enquanto i <= 10 faca
15 escreval(i)
16 i<-i+1
17 fimenquanto
18
19 Fimalgoritmo
```

Este foi um exemplo bem simples só pra explicar como a estrutura funciona, e a diferença entre uma e outra neste caso o incremento da variável está sendo feito no final da sequencia de instruções, enquanto no para e feito direto na estrutura, outro detalhe importante é que foi preciso dar um valor inicial da variável de controle.

VETOR E MATRIZ

Voltando ao exemplo das notas de alunos imagine uma sala com 30 alunos e ter dar notas a todos eles, e ser exibido o nome de todos, para isso seria preciso 30 variáveis uma para cada nome, também tem as notas, ou seja, seria muitas variáveis. Neste caso usamos um vetor ou uma matriz.

Vetor ou Matriz pode ser considerado como uma mesa com varias gavetas, assim é possível armazenar vários dados em um mesmo local de uma só vez, mas um vetor do tipo inteiro só pode armazenar dados deste tipo. Veja o exemplo.

1	2	3	4	5	6	7	8	9	10

Este é um exemplo de um vetor com 10 posições seria o mesmo que declarar dez variáveis, veja como declarar um vetor.

```
7 Var
8 // Seção de Declarações das variáveis
9 nomes:vetor[1..10] de caractere
```

Neste caso foi declarado um vetor com 10 posições, ou seja, pode ser armazenado até dez nomes nele, para atribuir os valores em um vetor usamos uma estrutura de repetição.

Exemplo:

```
1 Algoritmo "semnome"
2 // Disciplina   : [Linguagem e Lógica de Programação]
3 // Professor    : tecbh.blogspot.com.br
4 // Descrição    : Aqui você descreve o que o programa faz! (função)
5 // Autor(a)     : Nome do(a) aluno(a)
6 // Data atual   : 24/02/2016
7 Var
8 // Seção de Declarações das variáveis
9 nomes:vetor[1..10] de caractere
10 i:inteiro
11 Inicio
12 // Seção de Comandos, procedimento, funções, operadores, etc...
13 para i de 1 ate 10 faca
14        escreval("Digite o nome do ",i,"° aluno")
15        //quando se manda ler o vetor e so colocar o nome do vetor e o
16        // numero da posição do vetor neste caso e colocar a variavel i
17        // pois ele se auto incrementa.
18        leia(nomes[i])
19 fimpara
20 limpatela//este comando serve para limpar a tela do DOS
21 //Nesta parte será feita a leitura do vetor para isso e so repetir
22 //o codigo anterior usando somente o comando escreval e chamando as posiçoes
23 // do vetor
24     para i de 1 ate 10 faca
25        escreval("O ",i,"°"," nome digitado foi ",nomes[i])
26 fimpara
27 Fimalgoritmo
```

Matriz é a mesma coisa que o vetor, mas e mais parecido com uma tabela.

1,1	1,2	1,3	1,4
2,1	2,2	2,3	2,4
3,1	3,2	3,3	3,4
4,1	4,2	4,3	4,4

Está é uma matriz 4 por 4.

Na matriz temos colunas e linhas enquanto o vetor e somente linha, na tabela mostrada possui um números, que indicam sua posição.

1,1 linha 1 coluna 1

1,2 linha 1 coluna 2

2,4 linha 2 coluna 4

Sempre será linha e depois coluna.

Declarando e preenchendo uma matriz

Exemplo:

```
1  Algoritmo "semnome"
2  // Disciplina   : [Linguagem e Lógica de Programação]
3  // Professor    : tecbh.blogspot.com.br
4  // Descrição    : Aqui você descreve o que o programa faz! (função)
5  // Autor(a)     : Nome do(a) aluno(a)
6  // Data atual   : 24/02/2016
7  Var
8  // Seção de Declarações das variáveis
9  numero:vetor[1..4,1..4] de inteiro
10 // veja que a declaração e quase igual a unica diferença e a inclusão
11 // das colunas antes da virgula são as linhas e depois as colunas
12 // se fosse [1..4,1..2] seria 4 linha e duas colunas
13 linha,coluna:inteiro
14 Inicio
15 // Seção de Comandos, procedimento, funções, operadores, etc...
16 // para preencher e so usar uma esttrura de repetição dentro de outra
17 para linha de 1 ate 4 faca
18       para coluna de 1 ate 4 faca
19       escreval("Digite um n° para linha ",linha," na coluna ",coluna)
20                 leia(numero[linha,coluna])
21       fimpara
22 fimpara
23
24 Fimalgoritmo
```

Para fazer a leitura dos dados gravados na matriz e só fazer o mesmo código igual foi feito no vetor. Veja

```
1  Algoritmo "semnome"
2  // Disciplina   : [Linguagem e Lógica de Programação]
3  // Professor    : tecbh.blogspot.com.br
4  // Descrição    : Aqui você descreve o que o programa faz! (função)
5  // Autor(a)     : Nome do(a) aluno(a)
6  // Data atual   : 24/02/2016
7  Var
8  // Seção de Declarações das variáveis
9  num:vetor[1..4,1..4] de inteiro
10 // veja que a declaração e quase igual a unica diferença e a inclusão
11 // das colunas antes da virgula são as linhas e depois as colunas
12 // se fosse [1..4,1..2] seria 4 linha e duas colunas
13 linha,coluna:inteiro
14 Inicio
15 // Seção de Comandos, procedimento, funções, operadores, etc...
16 // para preencher e so usar uma esttrura de repetição dentro de outra
17 para linha de 1 ate 4 faca
18   para coluna de 1 ate 4 faca
19      escreval("Digite um n° para linha ",linha," na coluna ",coluna)
20      leia(num[linha,coluna])
21   fimpara
22 fimpara
23
24 limpatela
25
26 para linha de 1 ate 4 faca
27   para coluna de 1 ate 4 faca
28    escreval("O num da linha ",linha," na coluna ",coluna," é",num[linha,coluna]
29   fimpara
30 fimpara
31 Fimalgoritmo
```

Com isso chego ao final desta etapa, lembrando que esta apostila foi só uma introdução a logica de programação, os exercícios para treinar estes estudos estão disponíveis no blog. Caso tenha duvida ou critica ou sugestão e só entrar em contato através do e-mail, ou ate mesmo nos comentários do blog.

Obrigado.