Introdução à Lógica de Programação

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.

Seqüência Lógica

Estes pensamentos, podem ser descritos como uma seqüência de instruções, que devem ser seguidas para se 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 uma omelete de batatas, precisaremos colocar em prática uma série de instruções: descascar as batatas, bater os ovos, fritar as batatas, etc...

É evidente que essas instruções tem que ser executadas em uma ordem adequada – não se pode descascar as batatas depois de fritá-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.

Pedro F. Carvalho
Analista de Sistemas
contato@pedrofcarvalho.com.br
S.J. RIO PRETO - 2009

Algoritmos.

Algoritmo é a descrição de um conjunto de comandos que, obedecidos, resultam numa sucessão finita de ações.

Um Algoritmo é uma seqüência de instruções ordenadas de forma lógica para a resolução de uma determinada tarefa ou problema.

Um algoritmo é formalmente uma seqüência finita de passos que levam a execução de uma tarefa. Podemos pensar em algoritmo como uma receita, uma seqüência de instruções que dão cabo de uma meta específica. Estas tarefas não podem ser redundantes nem subjetivas na sua definição, devem ser claras e precisas.

Como exemplos de algoritmos podemos citar os algoritmos das operações básicas (adição, multiplicação, divisão e subtração) de números reais e decimais. Outros exemplos seriam os manuais de aparelhos eletrônicos, como um videocassete, que explicam passo-a-passo como, por exemplo, gravar um evento.

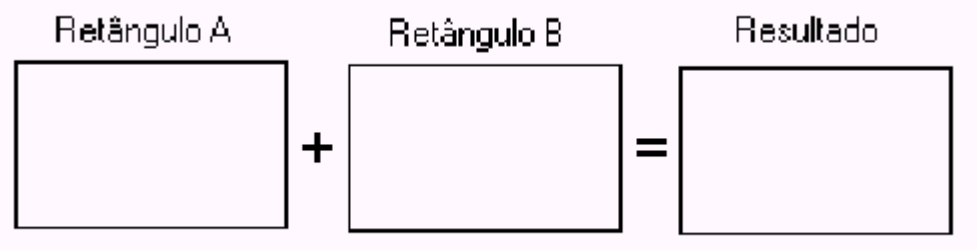
Até mesmo as coisas mais simples, podem ser descritas por seqüências lógicas. Por exemplo:

"Chupar uma bala".

- Pegar a bala;
- Retirar o papel;
- Chupar a bala;
- Jogar o papel no lixo.

"Somar dois números quaisquer".

- Escreva o primeiro número no retângulo A;
- Escreva o segundo número no retângulo B;
- Some o número do retângulo A com número do retângulo B e coloque o resultado no retângulo C.



Programas

Os programas de computadores nada mais são do que algoritmos escritos numa linguagem de computador (Pascal, C, Cobol, Fortran, Visual Basic, Java 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.

Exercícios

Obs: Abra um editor de texto para fazer os exercícios a seguir:

1. Crie uma seqüência lógica para tomar banho:
2. Descreva com detalhes a seqüência lógica para trocar um pneu de um carro.
3. Faça um algoritmo para trocar uma lâmpada. Descreva com detalhes:

Desenvolvendo algoritmos

Pseudocódigo

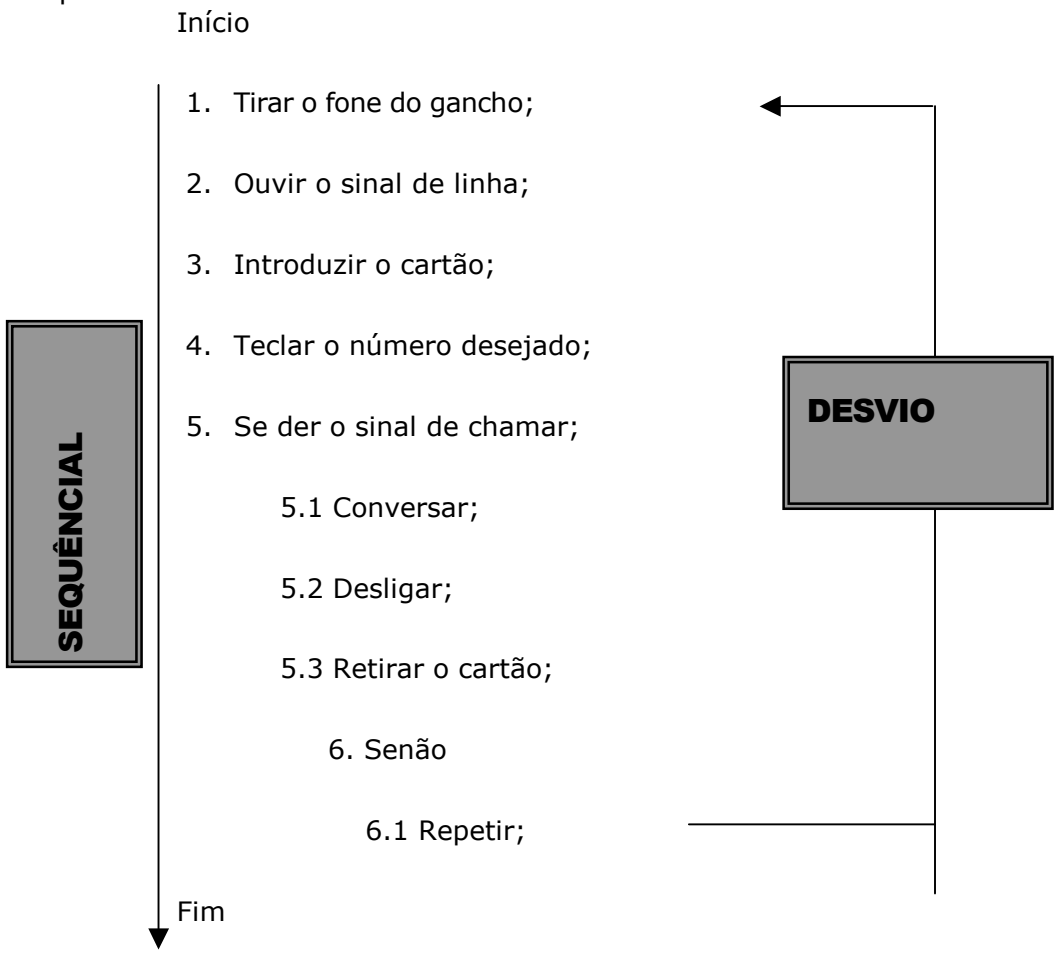
Os algoritmos são descritos em uma linguagem chamada **pseudocódigo**. Este nome é uma alusão à posterior implementação em uma linguagem de programação, ou seja, quando formos programar em uma linguagem, por exemplo Visual Basic, estaremos gerando código em Visual Basic. Por isso os algoritmos são independentes 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 de se interpretar e fácil de codificar. Ou seja, ele deve ser o intermediário entre a linguagem falada e a linguagem de programação.

Algoritmo não Computacional.

A seguir é apresentado um Algoritmo não computacional cujo objetivo é usar um telefone público.

Exemplo:



Este algoritmo, pode ser muito aperfeiçoado.

Pedro F. Carvalho
Analista de Sistemas
contato@pedrofcarvalho.com.br
S.J. RIO PRETO - 2009

Regras para construção do Algoritmo

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 usar palavras que não tenham sentido dúbio.

Fases

No item anterior vimos que 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 então

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:

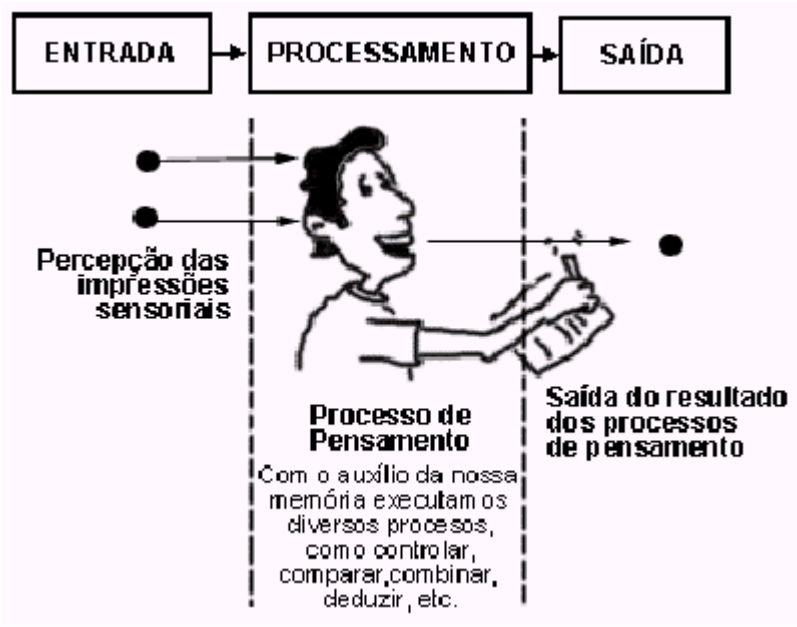


Figura 1

Exemplo de Algoritmo

Imagine o seguinte problema: Calcular a média final dos alunos da 3ª Série. Os alunos realizarão quatro provas: P1, P2, P3 e P4.

Para montar o algoritmo proposto, faremos três perguntas:

a) Quais são os dados de entrada?

R: Os dados de entrada são P1, P2, P3 e P4

b) Qual será o processamento a ser utilizado?

R: O procedimento será somar todos os dados de entrada e dividi-los por 4 (quatro)

$$\frac{P1 + P2 + P3 + P4}{4}$$

c) Quais serão os dados de saída?

R: O dado de saída será a média final

Algoritmo

Receba a nota da prova1

Receba a nota de prova2

Receba a nota de prova3

Receba a nota da prova4

Some todas as notas e divida o resultado por 4

Mostre o resultado da divisão

Teste de Mesa

Após desenvolver um algoritmo ele deverá sempre ser testado. Este teste é chamado de ***TESTE DE MESA***, que significa, seguir as instruções do algoritmo de maneira precisa para verificar se o procedimento utilizado está correto ou não.

Veja o exemplo:

Nota da Prova 1

Nota da Prova 2

Nota da Prova 3

Nota da Prova 4

Utilize a tabela abaixo:

P1	P2	P3	P4	Média

EXERCÍCIOS

Obs: Abra um editor de texto para fazer os exercícios a seguir:

1) Identifique os dados de entrada, processamento e saída no algoritmo abaixo

□□□Receba 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) Faça um algoritmo para "Calcular o estoque médio de uma peça", sendo que ESTOQUEMÉDIO = (QUANTIDADE MÍNIMA + QUANTIDADE MÁXIMA) /2

3) Teste o algoritmo anterior com dados definidos por você.

Pedro F. Carvalho
Analista de Sistemas
contato@pedrofcarvalho.com.br
S.J. RIO PRETO - 2009

Algoritmos em "PORTUGOL"

Durante nosso curso iremos aprender a desenvolver nossos Algoritmos em uma pseudo-linguagem conhecida como "Portugol" ou Português Estruturado.

"Portugol" é derivado da aglutinação de Português + Algol. Algol é o nome de uma linguagem de programação estruturada usada no final da década de 50.

Constantes, Variáveis e Tipos de Dados

Variáveis e constantes são os elementos básicos que um programa manipula.

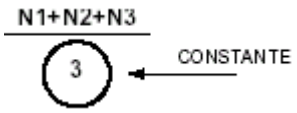
Uma variável é um espaço reservado na memória do computador para armazenar um tipo de dado determinado.

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 às vezes um valor inicial. Tipos podem ser por exemplo: inteiros, reais, 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 numérica, lógica e literal.

Exemplo de constantes:



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

Pedro F. Carvalho
Analista de Sistemas
contato@pedrofcarvalho.com.br
S.J. RIO PRETO - 2009

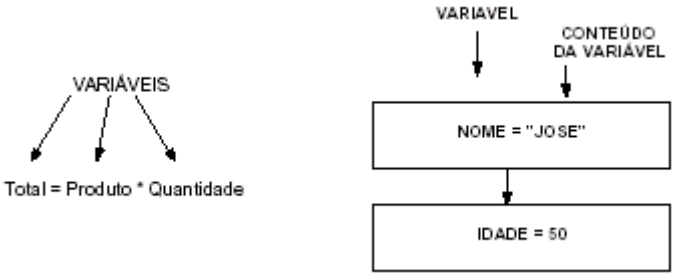


Figura 2

Tipos de Variáveis

As variáveis e as constantes podem ser basicamente de quatro tipos: Numéricas, Caracteres,

Alfanuméricas ou Lógicas.

Numéricas Específicas para armazenamento de números, que posteriormente poderão serutilizados para cálculos. Podem ser ainda classificadas como Inteiras ou Reais. As variáveis do tipo inteiro são para armazenamento de números inteiros e as Reais são para o armazenamento de números que possuam casas decimais.

Caracteres Específicas para armazenamento de conjunto de caracteres que não contenham números (literais). Ex: nomes.

Alfanuméricas Específicas para dados que contenham letras e/ou números. Pode em determinados momentos conter somente dados numéricos ou somente literais. Se usado somente para armazenamento de números, não poderá ser utilizada para operações matemáticas.

Lógicas Armazenam somente dados lógicos que podem ser Verdadeiro ou Falso.

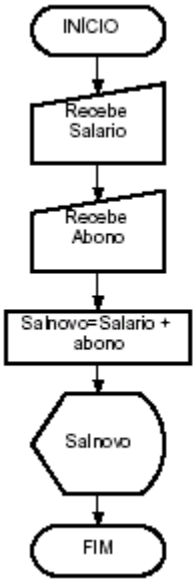
Declaração de Variáveis

As variáveis só podem armazenar valores de um mesmo tipo, de maneira que também são classificadas como sendo numéricas, lógicas e literais.

EXERCÍCIOS

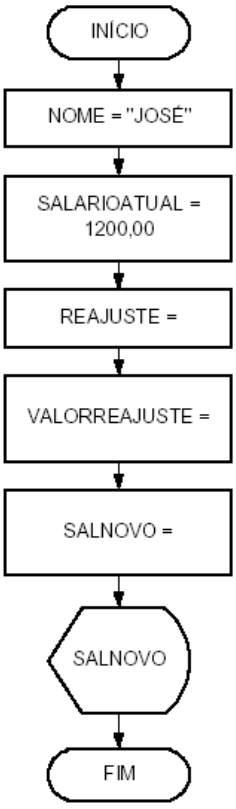
Obs: Abra um editor de texto para fazer os exercícios a seguir:

1) O que é uma constante? Dê dois exemplos.
2) O que é uma variável? Dê dois exemplos.
3) Faça um teste de mesa no diagrama de bloco abaixo e preencha a tabela ao lado com os
dados do teste:



Salário	Abono	Salnovo
600,00	60,00	
350,00		

4) Sabendo-se que José tem direito a 15% de reajuste de salário, complete o diagrama abaixo:



LINGUAGENS DE PROGRAMAÇÃO

São Softwares que permitem o desenvolvimento de programas. Possuem um poder de criação ilimitado, desde jogos, editores de texto, sistemas empresariais até sistemas operacionais.

Existem várias linguagens de programação, cada uma com suas características próprias.

Exemplos:

- Pascal;
- Clipper;
- C;
- Visual Basic;
- Delphi;
- Java;
- Cobol e etc.

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:

+ -	Operadores unários, isto é, são aplicados a um único operando. São os operadores aritméticos de maior precedência. Exemplos: -3, +x. Enquanto o operador unário - inverte o sinal do seu operando, o operador + não altera o valor em nada o seu

	valor.
\	Operador de divisão inteira. Por exemplo, 5 \ 2 = 2. Tem a mesma precedência do operador de divisão tradicional.
+ - * /	Operadores aritméticos tradicionais de adição, subtração, multiplicação e divisão. Por convenção, * e / têm precedência sobre + e -. Para modificar a ordem de avaliação das operações, é necessário usar parênteses como em qualquer expressão aritmética.
%	Operador de módulo (isto é, resto da divisão inteira). Por exemplo, 8 % 3 = 2. Tem a mesma precedência do operador de divisão tradicional.
^	Operador de potenciação. Por exemplo, 5 ^ 2 = 25. Tem a maior precedência entre os operadores aritméticos binários (aqueles que têm dois operandos).

OPERADORES RELACIONAIS

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

= < > <= >= <>	Respectivamente: igual, menor que, maior que, menor ou igual a, maior ou igual a, diferente de. São utilizados em expressões lógicas para se testar a relação entre dois valores do mesmo tipo. Exemplos: 3 = 3 (3 é igual a 3?) resulta em VERDADEIRO ; "A" > "B" ("A" está depois de "B" na ordem alfabética?) resulta em FALSO.

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

Expressão	Resultado
A = B	Falso
A <> B	Verdadeiro
A > B	Verdadeiro
A < B	Falso
A >= B	Verdadeiro
A <= B	Falso

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	Falso
A <> B	OR	B < C	Verdadeiro
A > B	NOT		Verdadeiro
A < B	AND	B > C	Verdadeiro
A >= B	OR	B = C	Falso
A <= B	NOT		Falso

EXERCÍCIOS

Obs: Abra um editor de texto para fazer os exercícios a seguir:

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
100,00	0,00	100	(SALLIQ >= 100,00)	
200,00	10,00	190,00	(SALLIQ < 190,00)	
300,00	15,00	285,00	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) ( )

VARIÁVEIS

Variáveis são endereços de memória destinados a armazenar informações temporariamente.

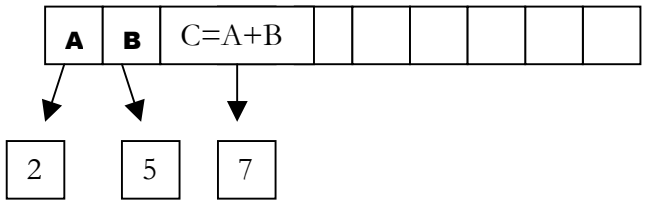
* Todo Algoritmo ou programa deve possuir variável!

VARIÁVEIS DE ENTRADA E SAÍDA

Variáveis de Entrada armazenam informações fornecidas por um meio externo, normalmente usuários ou discos.

Variáveis de Saída armazenam dados processados como resultados.

Exemplo:



De acordo com a figura acima A e B são Variáveis de Entrada e C é uma Variável de Saída.

CONSTANTES

Constantes são endereços de memória destinados a armazenar informações fixas, inalteráveis durante a execução do programa.

Exemplo:

PI = 3.1416

IDENTIFICADORES

São os nomes dados a variáveis, constantes e programas.

Regras Para construção de Identificadores:

Não podem ter nomes de palavras reservadas (comandos da linguagem);

Devem possuir como 1º caractere uma letra ou Underscore (_);

Ter como demais caracteres letras, números ou Underscore;

Ter no máximo 127 caracteres;

Não possuir espaços em branco;

A escolha de letras maiúsculas ou minúsculas é indiferente.

Exemplos:

NOME	TELEFONE	IDADE_FILHO
NOTA1	SALARIO	PI
UMNOMEMUITOCOMPRIDOEDIFICILDELER UM_NOME_MUITO_COMPRIDO_E_FACIL_DE_LER		

TIPOS DE DADOS

Todas as Variáveis devem assumir um determinado tipo de informação.

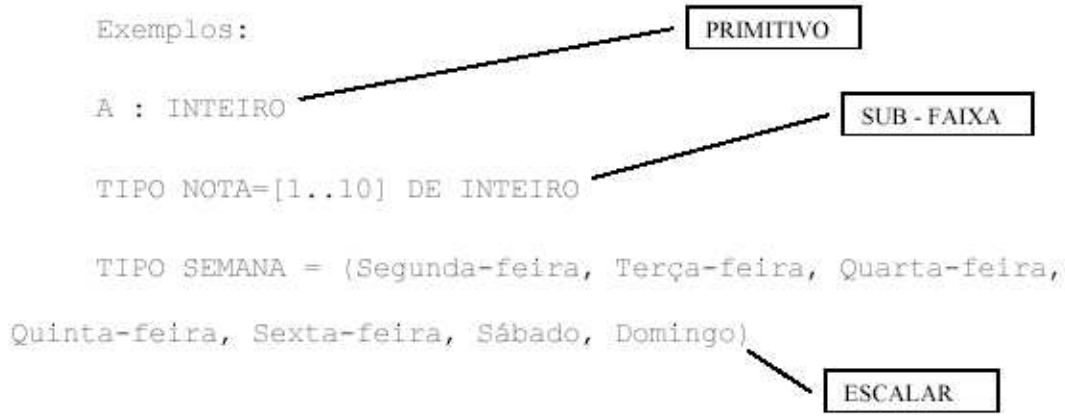
O tipo de dado pode ser:

Primitivo ➔ Pré-definido pela linguagem;

Sub-Faixa ➔ É uma parte de um tipo já existente;

Escalar ➔ Definidos pelo programador.

Exemplos:

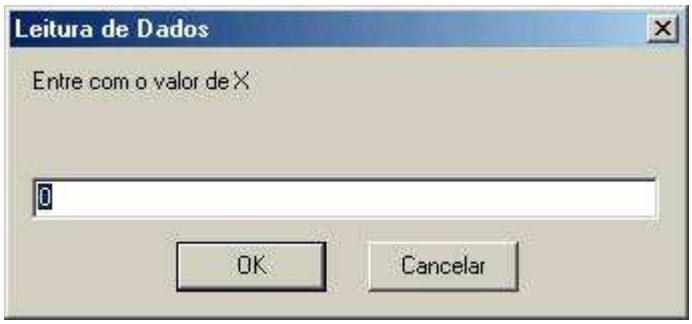


TIPOS PRIMITIVOS DE DADOS

INTEIRO	ADMITE SOMENTE NÚMEROS INTEIROS. GERALMENTE É UTILIZADO PARA REPRESENTAR UMA CONTAGEM (QUANTIDADE).

REAL	ADMITE NÚMEROS REAIS (COM OU SEM CASAS DECIMAIS). GERALMENTE É UTILIZADO PARA REPRESENTAR UMA MEDIÇÃO.
CARACTER	ADMITE CARACTERES ALFANUMÉRICOS. OS NÚMEROS QUANDO DECLARADOS COMO CARACTERES TORNAM SE REPRESENTATIVOS E PERDEM A ATRIBUIÇÃO DE VALOR.
LOGICO	ADMITE SOMENTE VALORES LÓGICOS(VERDADEIRO/FALSO).

Comando de Entrada de Dados



leia (<*lista-de-variáveis*>)

Recebe valores digitados pelo usuário, atribuindo-os às variáveis cujos nomes estão em <lista-de-variáveis> (é respeitada a ordem especificada nesta lista). É análogo ao comando read do Pascal.

Veja no exemplo abaixo o resultado:

algoritmo "exemplo 1”

var

x: inteiro
inicio
leia (x)
escreva (x)
fimalgoritmo

Comandos de Saída de Dados

escreva (<*lista-de-expressões*>)

Escreve no dispositivo de saída padrão (isto é, na área à direita da metade inferior da tela do VisuAlg) o conteúdo de cada uma das expressões que compõem <lista-de-expressões>. As expressões dentro desta lista devem estar separadas por vírgulas; depois de serem avaliadas, seus resultados são impressos na ordem indicada. É equivalente ao comando write do Pascal.

De modo semelhante a Pascal, é possível especificar o número de espaços no qual se deseja escrever um determinado valor. Por exemplo, o comando escreva (x:5) escreve o valor da variável x em 5 espaços, alinhado-o à direita. Para variáveis reais, pode-se também especificar o número de casas fracionárias que serão exibidas. Por exemplo, considerando y como uma variável real, o comando escreva (y:6:2) escreve seu valor em 6 espaços colocando 2 casas decimais.

escreval (<*lista-de-expressões*>).

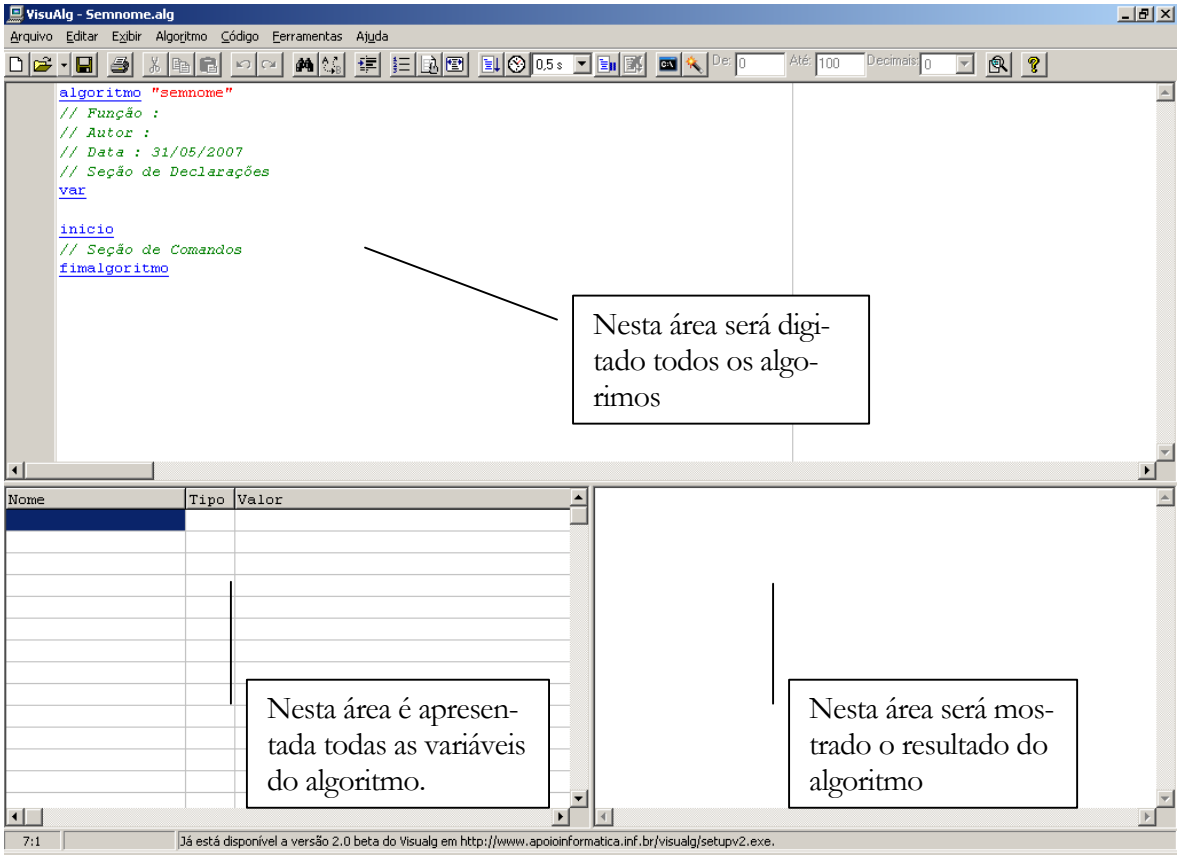
Idem ao anterior, com a única diferença que pula uma linha em seguida. É como se apertasse o botão ENTER e mudasse de linha.

CORPO GERAL DE UM PROGRAMA

```
algoritmo "semnome" // nome do programa ou algoritmo

var

 // seção das variáveis

inicio

 // Seção de Comandos

fimalgoritmo
```

UTILIZANDO O VISUALG

Programa usado para testar todos os algoritmos utilizando um linguagem mais próxima da nossa linguagem visando facilitar o entendimento usando o nosso idioma.



Para executar os algoritmos pressione F9.

ESTRUTURAS SEQÜÊNCIAIS

Como pode ser analisado no tópico anterior, todo programa possui uma estrutura seqüencial determinada por um ÍNICIO e FIM.

PRIMEIRO ALGORITMO

Obs: Abra o Visualg para realizar as seguintes execuções.

Execute 1.

Segue um Algoritmo que lê quatro números e calcular a média entre quatro números. Digite na área destinado ao algoritmo utilizando o visualg.

```
algoritmo "medianumeros"
var
      // declaração das variáveis
      a,b,c,d,media: real

inicio

      // entrada: leitura das variáveis
      leia (a,b,c,d)
     // processamento: calculo
     media <- (a + b + c + d) / 4
     // saída: resultado do cálculo
     escreva (media)

fimalgoritmo
```

Execute 2.

Segue programa que receba um valor inteiro e mostre na tela o quadrado deste valor.

```
algoritmo "calculaquadrado"


var
    valor: inteiro
    quadrado:inteiro
inicio
     leia (valor)
    quadrado <- valor * valor
    escreva (quadrado)
fimalgoritmo
```

Execute 3.

Segue um algoritmo que receba um valor que foi depositado e exiba o valor com rendimento após um mês. Considere fixo o juro da poupança em 0.70% ao mês.

```
algoritmo "valoreajustado"


var
   valor:real;
   valorreajuste:real;
inicio
   leia(valor)
   valorreajuste <- valor + (valor * 0.70 /100)
   escreva (valorreajuste)
fimalgoritmo
```

Execute 4.

Segue um programa que receba duas notas de um aluno e mostre na tela a média ponderada dessas notas, sendo que a primeira tem peso 10 e a segunda tem peso 6.

```
algoritmo "mediaponderada"


var
   nota1:real
   nota2:real
   media:real
inicio
   leia(nota1)
   leia(nota2)
   media <- ((nota1 * 10) + (nota2 * 6)) / 2
   escreva (media)
fimalgoritmo
```

Execute 5.

Dado o cardápio de uma pizzaria. Calcule a conta de acordo com o pedido: (diga quanto cada pessoa vai pagar). Para fazer este algoritmo mude a execução para Executar algoritmo como DOS (clique no menu algoritmo, escolha a opção Executar como DOS).

Mesa 02		
Pizzas	**Quantidade**	**Valor**
Pizza Portuguesa	2	R$ 18,00
Pizza Calabresa	0	R$ 17,00
Pizza Mussarela	1	R$ 16,00
Pizza Chocolate	0	R$ 17,00
Refrigerante	2	R$ 3,00

Total: R$ 58,00

Quantidade de Pessoas: 10

Total por pessoa: R$ 5,8

```
algoritmo "pizzaria"
var
   numeromesa:inteiro    // variável do numero da mesa
   qtpizzaport:inteiro   // variável indicando a quantidade pizza portuguesa
   qtpizzacala:inteiro   // variável indicando a quantidade pizza calabresa
   qtpizzamussa:inteiro  // variável indicando a quantidade pizza mussarela
   qtpizzachoco:inteiro  // variável indicando a quantidade pizza chocolate
   qtrefri:inteiro       // variável indicando a quantidade de refrigerante
   qtpessoas:inteiro     // variável indicando a quantidade de pessoas
   totporpessoa:real     // variável indicando o total a pagar por pessoa
   totalmesa:real        // variável indicando a total a pagar

inicio
   escreval("-------------------------------------------------------")
   escreval("                Pizzaria come bem")
   escreval("-------------------------------------------------------")
   escreva("Mesa: ")
   leia(numeromesa)
   escreval("-------------------------------------------------------")
   escreval("Pizza Portuguesa")
   escreva ("Quantidade: ")
   leia(qtpizzaport)
   escreval("Pizza Calabresa")
   escreva ("Quantidade: ")
   leia(qtpizzacala)
   escreval("Pizza Mussarela")
```

```
            escreva ("Quantidade: ")
            leia(qtpizzamussa)
            escreval("Pizza Chocolate")
            escreva ("Quantidade: ")
            leia(qtpizzachoco)
            escreval("Refrigerante")
            escreva ("Quantidade: ")
            leia(qtrefri)
            escreval("---------------------------------------------------------")
          totalmesa <- (qtpizzaport * 18) + (qtpizzacala * 17) + (qtpizzamussa * 16) +
(qtpizzachoco * 3) + (qtrefri * 3)
            escreval("Total a pagar: R$",totalmesa)
            escreval("---------------------------------------------------------")
            escreva("Quantidade de pessoas: ")
            leia(qtpessoas)
            escreval("---------------------------------------------------------")
            totporpessoa <- totalmesa / qtpessoas
            escreva("Total por pessoa: ",totporpessoa)
            fimalgoritmo
```

Execute 6.

Segue um programa que receba um valor x e calcule a expressão (x * x) – x + 7. Para executar esse algoritmo desmarque a opção Executar em modo DOS (clique no menu algoritmo, escolha a opção Executar em modo DOS então ela será desmarcada)

```
algoritmo "expressao"


var
  x:inteiro
inicio
  leia(x)
```

```
    x <- (x * x) - x + 7

    escreva(x)

fimalgoritmo
```

Exercícios

Obs: Abra o Visualg para realizar os seguintes exercícios.

1. Faça um algoritmo, que lê o número de um funcionário, seu número de horas trabalhadas e o valor que recebe por hora. O algoritmo deve calcular e mostrar o salário deste funcionário.
2. Escrever um algoritmo que lê o código da peça 1, a quantidade vendida de peças 1, o valor unitário da peça 1, o código da peça 2, a quantidade vendida de peças 2, o valor unitário da peça 2 e a percentagem do IPI a ser acrescentada. O algoritmo deve calcular o valor total a ser pago.
3. Escreva um algoritmo para ler dois inteiros (variáveis A e B) e efetuar as operações de adição, subtração, multiplicação e divisão de A por B apresentando ao final os quatro resultados obtidos.
4. Escreva um algoritmo para calcular a área de um triângulo, exibindo o resultado final. A base e a altura são dados que devem ser lidos como entrada.

$$\text{ÁREA} = \frac{\text{BASE . ALTURA}}{2}$$

5. Um loja de animais precisa de um algoritmo para calcular os custos de criação de coelhos. O custo é calculado com a fórmula CUSTO=(NRO_COELHOS*0.70)/18+10. O algoritmo tem como entrada o número de coelhos, devendo fornecer, como saída, o custo.
6. Escreva um algoritmo para ler os seguintes números: A, B e C. Após, calcular o valor de D segundo a expressão: D = B2 - 4AC e mostrar os valores lidos e o resultado.
7. Faça um algoritmo que leia a idade de uma pessoa em anos, meses e dias e mostre-a expressa em dias. (Nota: considere todos os anos com 365 dias e todos os meses com 30 dias).
8. Escrever um algoritmo para efetuar o cálculo da quantidade de litros de combustível gasta em uma viagem, utilizando-se um automóvel que faz 12 km por litro. Para obter o cálculo, o usuário deverá fornecer o tempo gasto na viagem e a velocidade média durante a mesma. Desta forma, será possível obter a distância percorrida com a fórmula DISTÂNCIA=TEMPO*VELOCIDADE. Tendo o valor da distância, basta calcular a quantidade de litros de combustível utilizada na viagem com a fórmula: LITROS_USADOS=DISTÂNCIA/12. O algoritmo deverá apresentar os valores da velocidade média, tempo gasto na viagem, a distância percorrida e a quantidade de litros utilizada na viagem.
9. Escreva um algoritmo para calcular o consumo médio de um automóvel (medido em Km/l), dado que são conhecidos a distância total percorrida e o volume do combustível consumido para percorrê-la (medidos em litros).
10. Escreva um algoritmo para dar o total, em reais, de um cofrinho que contenha:

n1 moedas de 1 real;
n2 moedas de 50 centavo\s;
n3 moedas de 25 centavos;
n4 moedas de 10 centavos; e

n5 moedas de 5 centavos.

11. Escreva um algoritmo que efetue e calculo do salário líquido de um professor. Para fazer este programa você deverá possuir alguns dados, tais como: valor da hora aula, número de aulas dadas no mês e percentual de desconto do INSS. Em primeiro lugar, deve-se estabelecer qual será o seu salário bruto para efetuar o desconto e ter o valor do salário líquido. Ao final do algoritmo devem ser mostrados o salário bruto e o salário líquido do professor.

ESTRUTURAS DE DECISÃO

SE..entao..senao

Executa uma seqüência de comandos de acordo com o resultado de um teste.

A estrutura de decisão pode ser Simples ou Composta, baseada em um resultado lógico.

se <*expressão-lógica*> entao
   <*seqüência-de-comandos*>
fimse

Ao encontrar este comando, o VisuAlg analisa a <expressão-lógica>. Se o seu resultado for VERDADEIRO, todos os comandos da <seqüência-de-comandos> (entre esta linha e a linha com fimse) são executados. Se o resultado for FALSO, estes comandos são desprezados e a execução do algoritmo continua a partir da primeira linha depois do fimse.

se <*expressão-lógica*> entao
   <*seqüência-de-comandos-1*>
senao
   <*seqüência-de-comandos-2*>
fimse

Nesta outra forma do comando, se o resultado da avaliação de <expressão-lógica> for VERDADEIRO, todos os comandos da <seqüência-de-comandos-1> (entre esta linha e a linha com senao) são executados, e a execução continua depois a partir da primeira linha depois do fimse. Se o resultado for FALSO, estes comandos são desprezados e o algoritmo continua a ser executado a partir da primeira linha depois do senao, executando todos os comandos da <seqüência-de-comandos-2> (até a linha com fimse).

Estes comandos equivalem ao if...then e if...then...else do Pascal. Note que não há necessidade de delimitadores de bloco (como begin e end), pois as seqüências de comandos já estão delimitadas pelas palavras-chave senao e fimse. O VisuAlg permite o aninhamento desses comandos de desvio condicional.

Execute 7.

Segue um programa para achar o maior numero entre dois números informados:

```
algoritmo "acharmaior"
var
x,y: inteiro
inicio
 leia (x)
 leia (y)
se x > y entao
   escreva (x," é maior do que",y)
senao
   escreva (x," é menor do que",y)
fimse
fimalgoritmo
```

Execute 8.

Segue um Algoritmo que lê o nome e as 4 notas bimestrais de um aluno. Em seguida o Algoritmo calcula e escreve a média obtida pelo aluno escrevendo também se o aluno foi aprovado ou reprovado.

Média para aprovação é 6

```
algoritmo "mediaaprovacao"
var
      nota1, nota2, nota3, nota4, media: real
inicio
 leia (nota1)
 leia (nota2)
 leia (nota3)
```

```
leia (nota4)
media <- (nota1 + nota2 + nota3 + nota4) / 4
se media > 6 entao
   escreva ("Aprovado")
senao
   escreva ("Reprovado")
fimse
fimalgoritmo
```

Execute 9.

Segue um Algoritmo que lê 3 números e escreve o maior.

```
algoritmo "mediaaprovacao"
var
 a,b,c:real
inicio
 leia(a,b,c)
 se (a > b) e (a > c) entao
   escreva (a, " é maior")
 senao
   se (b > a) e (b > c) entao
      escreva (b, " é maior")
   senao
      escreva (c, " é maior")
   fimse
 fimse
fimalgoritmo
```

Execute 10.

Segue um algoritmo que lê a idade de uma pessoa. E ao final imprima uma frase dizendo se essa pessoa é Adulta ou é de Menor.

```
algoritmo "veridade"
var
   idade:inteiro
inicio
escreva("Digite sua idade: ")
leia (idade)
   se (idade < 18) entao
       escreval("é menor de idade")
      senao
       escreval("é adulta")
   fimse
fimalgoritmo
```

Execute 11.

Segue um algoritmo que lê uma senha. Caso a senha informada for igual "12345X" imprima a frase Senha Correta, caso contrário imprima a frase Senha Inválida.

```
algoritmo "veridade"
var
   senha:caracter
```

```
inicio
   leia (senha)
   se (senha = "12345X" ) entao
      escreval("Senha correta")
     senao
      escreval("Senha invalida")
   fimse
fimalgoritmo
```

Execute 12.

Dado o salário de um funcionário, calcular e imprimir o valor do INSS de acordo com a tabela abaixo:

Salário	**%INSS**
Até 465.85	7.65%
Acima 465.85 até 635.11	8.65%
Acima 635.11 até 913.14	9%
Acima 913.14	11%

```
algoritmo "versalario"


var
   salario:real
   inss:real
inicio
   leia (salario)
   se (salario <= 465.85) entao
      inss <- salario * 7.65 / 100
   fimse
```

```
        se (salario > 465.85) e (salario <= 635.11) entao
          inss <- salario * 8.65 / 100
        fimse
        se (salario > 635.11) e (salario <= 913.14) entao
          inss<-salario * 9 / 100
        fimse
        se (salario > 913.14) entao
          inss <- salario * 11 / 100
        fimse

        escreval("O valor do inss é :",inss)
      fimalgoritmo
```

Exercícios

Obs: Abra o Visualg para realizar os seguintes exercícios.

Faça um algoritmo que leia dois números N1 e N2, nesta ordem e imprima:
PRIMEIRO: se N1<N2
SEGUNDO: se N1>N2
TERCEIRO: se N1=N2

Escrever um algoritmo para ler e imprimir três números. Se o primeiro for positivo, imprimir sua raiz quadrada, caso contrário, imprimir o seu quadrado; se o segundo número for maior que 10 e menor que 100, imprimir a mensagem: "NÚMERO ESTÁ ENTRE 10 E 100 - INTERVALO PERMITIDO"; se o terceiro número for menor que o segundo, calcular e imprimir a diferença entre eles, caso contrário, imprimir calcular e imprimir a soma entre eles.

A concessionária de veículos "Carro Zero e Velho" está vendendo os seus veículos com desconto. Faça um algoritmo que calcule e exiba o valor do desconto e o valor a ser pago pelo cliente. O desconto deverá ser calculado de acordo com o ano do veículo. Até 2000 (12% de desconto), acima 2000 (7% de desconto).

Escrever um algoritmo para ler dois valores: NUM1 e NUM2, e se NUM1 for maior que NUM2 executa a soma de NUM1 e NUM2; caso contrário, executa uma subtração.

Escrever um algoritmo que leia os dados de uma pessoa (nome, sexo, idade, e saúde) e informe se ela está apta ou não para cumprir o serviço militar.

Faça um algoritmo eu receba o preço de custo e o preço de venda de um produto. Mostre como resultado se houve lucro, prejuízo ou empate.

Faça um algoritmo que leia um número de 1 a 5 e escreva por extenso. Caso o usuário digite um número que não esteja neste intervalo, exibir mensagem: número inválido.

A concessionária de veículos "Carro Bom" esta vendendo os seus veículos com desconto. Faça um algoritmo que calcule e exiba o valor do desconto e o valor a ser pago pelo clien-

te. O desconto deverá ser calculado sobre o valor do veículo de acordo como combustível (álcool –25%, gasolina – 21% ou diesel – 14%).
Elaborar um algoritmo que lê dois valores e escreve cada um juntamente com a mensagem: "É múltiplo de 2" ou "Não é múltiplo de dois".
Faça um algoritmo que receba um número e mostre o mês correpondente.
Escrever um algoritmo que leia três valores inteiros distintos e os escreva em ordem crescente.

Escolha....caso

O VisuAlg implementa (com certas variações) o comando case do Pascal. A sintaxe é a seguinte:

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

Veja o exemplo a seguir, que ilustra bem o que faz este comando. Abra o Visualg para realizar as seguintes execuções.

Execute 13.

Segue um algoritmo que lê o nome de um time e responde o estado correspondente desse time.

```
algoritmo "Times"
var
    time: caracter
inicio
    escreva ("Entre com o nome de um time de futebol: ")
    leia (time)
    escolha time
        caso "Flamengo", "Botafogo", "Fluminense", "Vasco"
            escreval ("É um time carioca.")
        caso "Corinthians",  "Santos", "São Paulo", "Palmeiras"
```

```
                escreval ("É um time paulista.")
            outrocaso
                escreval ("É de outro estado.")
        fimescolha
    fimalgoritmo
```

Execute 14.

A escola Aprender faz o pagamento de seus professores por hora aula. Faça um algoritmo que calcule e exiba o salário de um funcionário. Sabe-se que o valor da hora aula segue a tabela abaixo:

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

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

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

```
algoritmo "porextenso"
var
   nivel:inteiro
   qthora:real
   salario:real
inicio
   escreva("Digite o nivel do professor: ")
   leia (nivel)
   escreva("Digite a quantidade de horas trabalhadas: ")
   leia (qthora)
   escolha nivel
     caso 1     // recebe R$12,00 por hora
        salario <- qthora * 12
     caso 2     // recebe R$17,00 por hora
        salario <- qthora * 17
     caso 3     // recebe R$25,00 por hora
        salario <- qthora * 25
     outrocaso
       escreval ("nível inválido")
       salario <-0
   fimescolha

   escreval("O salario é: ",salario)

fimalgoritmo
```

Execute 15.

Faça um algoritmo que leia um número. Escreva por extenso os números de (1 a 5) caso um número fora do intervalo escreva "número inválido".

```
algoritmo "porextenso"
var
numero: inteiro

inicio
leia (numero)
escolha numero
caso 1
   escreval ("Um")
caso 2
   escreval ("Dois")
caso 3
   escreval ("Três")
caso 4
   escreval ("Quatro")
caso 5
   escreval ("Cinco")
outrocaso
   escreval ("número inválido")
fimescolha
fimalgoritmo
```

Exercícios

Obs: . Abra o VisuAlg para realizar os seguintes exercícios.

Escrever um algoritmo que lê um conjunto de 4 valores i, a, b, c, onde i é um valor inteiro e positivo e a, b, c, são quaisquer valores reais e os escreva. A seguir:

Se i=1 escrever os 3 valores a, b, c em ordem crescente;
Se i=2 escrever os 3 valores a, b, c em ordem decrescente;
Se i=3 escrever os 3 valores de forma que o maior valor entre a, b, c fica entre os outros dois.
Escrever um algoritmo que leia um código e três valores: a, b e c. Os códigos válidos são 1, 2, 3, 4 e 5. Se o código for diferente destes, apresentar a mensagem "CÓDIGO INVÁLIDO" e terminar o programa.
código = 1: multiplicar os três valores;
código = 2: somar os três valores;
código = 3: subtrair os três valores;
código = 4: somar o cubo dos 3 valores;
código = 5: somar o quadrado dos 3 valores.
Escreva um algoritmo que tendo como dados de entrada o preço de um produto e um código de origem emita o preço junto de sua procedência. Caso o código não seja nenhum dos especificados o produto é considerado importado. Os códigos de origem são os seguintes:

1 - Sul	5 ou 6 - Nordeste
2 – Norte	7, 8 ou 9 - Sudeste
3 – Leste	10 até 20 - Centro Oeste
4 – Oeste	25 até 50 – Noroeste

Um vendedor necessita de um algoritmo que calcule o preço total devido por um cliente. O algoritmo deve receber o código de um produto e a quantidade comprada e calcular o preço total, usando a tabela abaixo:

Código Produto	Preço Unitário
1001	5,32
1324	6,45
6548	2,37
0987	5,32
7623	6,45

ESTRUTURA DE REPETIÇÃO DETERMINADA

Quando uma seqüência de comandos deve ser executada repetidas vezes, tem-se uma estrutura de repetição.

A estrutura de repetição, assim como a de decisão, envolve sempre a avaliação de uma condição.

Na repetição determinada o algoritmo apresenta previamente a quantidade de repetições.

Esta estrutura repete uma seqüência de comandos em um determinado número de vezes, sendo especificada a condição inicial e a final.

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

<variável >	É a variável contadora que controla o número de repetições do laço. Na versão atual, deve ser necessariamente uma variável do tipo inteiro, como todas as expressões deste comando.
<valor-inicial>	É uma expressão que especifica o valor de inicialização da variável contadora antes da primeira repetição do laço.
<valor-limite >	É uma expressão que especifica o valor máximo que a variável contadora pode alcançar.
<incremento >	É opcional. Quando presente, precedida pela palavra passo, é uma expressão que especifica o incremento que será acrescentado à variável contadora em cada repetição do laço. Quando esta opção não é utilizada, o valor padrão de <*incremento*> é 1. Vale a pena ter em conta que também é possível especificar valores negativos para <*incremento*>. Por outro lado, se a avaliação da expressão <*incremento* > resultar em valor nulo, a execução do algoritmo será interrompida, com a impressão de uma mensagem de erro.
fimpara	Indica o fim da seqüência de comandos a serem repetidos. Cada vez que o programa chega neste ponto, é acrescentado à variável contadora o valor de <*incremento* >, e comparado a <*valor-limite* >. Se for menor ou igual (ou maior ou igual, quando <*incremento* > for negativo), a seqüência de comandos será executada mais uma vez; caso contrário, a execução prosseguirá a partir do primeiro comando que esteja após o fimpara.

<valor-inicial >, <valor-limite > e <incremento > são avaliados uma única vez antes da execução da primeira repetição, e não se alteram durante a execução do laço, mesmo que variáveis eventualmente presentes nessas expressões tenham seus valores alterados.

No exemplo a seguir, os números de 1 a 10 são exibidos em ordem crescente.

```
algoritmo "Números de 1 a 10"
var j: inteiro
inicio
para j de 1 ate 10 faca
  escreva (j:3)
fimpara
fimalgoritmo
```

Importante: Se, logo no início da primeira repetição, <valor-inicial > for maior que <valor-limite > (ou menor, quando <incremento> for negativo), o laço não será executado nenhuma vez. O exemplo a seguir não imprime nada.

```
algoritmo "Numeros de 10 a 1 (não funciona)"
var j: inteiro
inicio
para j de 10 ate 1 faca
  escreva (j:3)
fimpara
fimalgoritmo
```

Este outro exemplo, no entanto, funcionará por causa do passo -1:

```
algoritmo "Numeros de 10 a 1 (este funciona)"
var j: inteiro
inicio
para j de 10 ate 1 passo -1 faca
   escreva (j:3)
fimpara
fimalgoritmo
```

Obs: . Abra o VisuAlg para realizar as seguintes execuções.

Execute 16.

Segue um algoritmo que escreve 10 vezes a frase "Hytec Informática"

```
algoritmo "repetefrase"

var

   i:inteiro

inicio

   para i de 1 ate 10 faca

        escreval("Hytec Informática")

   fimpara

fimalgoritmo
```

Execute 17.

Segue um algoritmo que escreve os 100 primeiros números pares.

```
algoritmo "paresate100"

var

   i:inteiro

inicio

   para i de 2 ate 100 passo 2 faca

        escreval(i)

   fimpara

fimalgoritmo
```

Execute 18.

Segue um algoritmo que os números de 10 a 1

```
algoritmo "Numeros10a1"
var j: inteiro
inicio
para j de 10 ate 1 passo -1 faca
  escreva (j:3)
fimpara
fimalgoritmo
```

Execute 19.

Segue um algoritmo que receba a nota obtida por uma turma de 30 alunos. Calcule a média obtida pela turma.

```
algoritmo "media 30 alunos"

var

nota, media, soma: real

i: inteiro

inicio

soma <- 0

para i de 1 ate 30 faca

  escreva ("Digite a nota do aluno: ")

  leia (nota)

  soma <- soma + nota

  escreva ("O valor da soma das notas dos alunos é: ")

  escreval (soma)

fimpara

media <- soma / 30

escreva ("O valor da média das notas de todos os alunos: ")

escreva (media)

fimalgoritmo
```

Exercícios

Obs: . Abra o VisuAlg para realizar os seguintes exercícios.

1. Escreva um algoritmo para escrever a palavra PROGRAMACAO 5 vezes utilizando uma estrutura de repetição.
2. Escreva um algoritmo para ler um número inteiro e escrevê-lo na tela 10 vezes utilizando uma repetição.
3. Escreva um algoritmo que imprima na tela os 10 primeiros números inteiros maiores que 100 utilizando uma estrutura de repetição.
4. Escreva um algoritmo que imprima os números ímpares existentes de entre 1(inclusive) e 9(inclusive).
5. Escreva um algoritmo para ler um valor N (validar para aceitar apenas valores positivos) e imprimir a palavra PROGRAMACAO N vezes.
6. Escreva um algoritmo para ler um valor N (validar para aceitar apenas valores positivos) e imprimir os N primeiros números inteiros.
7. Escreva um algoritmo que imprima a tabuada do 8 utilizando uma estrutura de repetição.
8. Escreva um algoritmo para ler um valor X (validar para aceitar apenas valores entre 1(inclusive) e 10(inclusive)). Escrever a tabuada de X.
9. Ler 10 valores e escrever quantos destes valores são negativos.
10. Ler 10 valores e contar quantos estão no intervalo [10,20] e quantos deles estão fora deste intervalo. Escrever o resultado das duas contagens.
11. Ler 10 valores, calcular e escrever a média aritmética destes valores.
12. Ler o número de alunos existentes em uma turma, ler as notas destes alunos, e calcular a média aritmética destas notas.
13. Ler um valor A e um valor N. Imprimir a soma dos N números a partir de A(inclusive). Caso N seja negativo ou ZERO, deverá ser lido um novo N(apenas N).

Valores para teste:

A	N	SOMA	
3	2	7	(3+4)
4	5	30	(4+5+6+7+8)

ESTRUTURA DE REPETIÇÃO INDETERMINADA COM VALIDAÇÃO INICIAL

É usada para repetir N vezes uma ou mais instruções. Tendo como vantagem o fato de não ser necessário o conhecimento prévio do número de repetições.

Esta estrutura repete uma seqüência de comandos enquanto uma determinada condição (especificada através de uma expressão lógica) for satisfeita.

enquanto <*expressão-lógica*> faca
  <*seqüência-de-comandos*>
fimenquanto

<expressão-lógica>	Esta expressão que é avaliada antes de cada repetição do laço

	Quando seu resultado for VERDADEIRO, *<seqüência-de-comandos>* é executada.
fimenquanto	Indica o fim da *<seqüência-de-comandos>* que será repetida. Cada vez que a execução atinge este ponto, volta-se ao início do laço para que *<expressão-lógica>* seja avaliada novamente. Se o resultado desta avaliação for VERDADEIRO, a *<seqüência-de-comandos>* será executada mais uma vez; caso contrário, a execução prosseguirá a partir do primeiro comando após fimenquanto.

Obs: Abra o VisuAlg para realizar as seguintes execuções.

Execute 20.

Segue um algoritmo que imprime de 1 a 10.

```
algoritmo "Númerosde1a10"
var
j:inteiro
inicio
j <- 1
enquanto j <= 10 faca
  escreva (j:3)
  j <- j + 1
fimenquanto
fimalgoritmo
```

Execute 21.

Segue um algoritmo que calcule a soma dos salários dos funcionários de uma empresa. O programa termina quando o usuário digitar um salário igual a zero

```
algoritmo "somasalario"
var
salario:inteiro
```

```
soma:real
inicio
salario <- 1
enquanto salario <> 0 faca
  leia(salario)
  soma <- soma + salario
fimenquanto
escreval("A soma dos salarios é: ",soma)
fimalgoritmo.
```

Execute 22.

Segue um algoritmo que receba o peso de 1000 porcos, ao final mostre o maior peso.

```
algoritmo "maiorporco"
var
 peso:real
 maior:real
 qtporcos:inteiro

inicio
 maior <- 0
 qtporcos <- 1
enquanto qtporcos <= 5 faca
   leia(peso)
   se (peso > maior) entao
      maior <- peso
   fimse
   qtporcos <- qtporcos + 1
fimenquanto
escreval("o Maior peso é: ",maior)
fimalgoritmo
```

Execute 23.

Segue um algoritmo que entre com vários números positivos e imprima a média dos números digitados.

```
algoritmo "numerospositivos"
var
   numero:inteiro
```

```
    soma:inteiro  // variável usada para calcular a soma dos numeros
    media:real    // variável usada para calcular a media dos numeros
    qtnum:inteiro // variável usada para contar a quantidade de numeros
inicio
    qtnum <- 0
    soma <- 0
    leia(numero)
    enquanto numero > 0 faca
        soma <- soma + numero
        qtnum <- qtnum + 1
        leia(numero)
    fimenquanto
    media <- soma / qtnum
    escreval("A média dos numeros positivos digitados é: ",media)
fimalgoritmo
```

Execute 24.

Segue um algoritmo que entre com o sexo de várias pessoas e imprima quantas são do sexo masculino e quantas são do sexo feminino.

```
algoritmo "verificarsexoquantidade"


var
  sexo:caracter
  qtmasc:inteiro
  qtfem:inteiro
inicio
  escreva("Entre com o sexo da pessoa ou digite s para sair:")
  leia(sexo)
```

```
enquanto (sexo <> "s") ou (sexo <> "S") faca
  se (sexo = "f") ou (sexo = "F") entao
    qtfem <- qtfem + 1
  fimse
  se (sexo = "m") ou (sexo = "M") entao
        qtmasc <- qtmasc + 1
  fimse
  escreva("Entre com o sexo da pessoa ou digite s para sair:")
  leia(sexo)
 fimenquanto
 escreval("A quantidade de pessoas com sexo masculino é: ",qtmasc)
 escreval("A quantidade de pessoas com sexo feminino é:",qtfem)
fimalgoritmo
```

ESTRUTURA DE REPETIÇÃO INDETERMINADA COM VALIDAÇÃO FINAL

Assim como a estrutura ENQUANTO É usada para repetir N vezes uma ou mais instruções.

Sua validação é final fazendo com que a repetição seja executada pelo menos uma vez.

Esta estrutura repete uma seqüência de comandos até que uma determinada condição (especificada através de uma expressão lógica) seja satisfeita.

repita
  <*seqüência-de-comandos*>
ate <*expressão-lógica*>

repita	Indica o início do laço.
ate <expressão-lógica>	Indica o fim da <*seqüência-de-comandos*> a serem repetidos. Cada vez que o programa chega neste ponto, <*expressão-lógica*> é avaliada: se seu resultado for FALSO, os comandos presentes entre esta linha e a linha repita são executados; caso contrário, a execução prosseguirá a partir do primeiro comando após esta linha.

Obs: Abra o VisuAlg para realizar as seguintes execuções.

Execute 25.

Segue um algoritmo que calcule a soma dos salários dos funcionários de uma empresa. O programa termina quando o usuário digitar um salário menor que 0.

```
algoritmo "somasalario"
var
salario:inteiro
soma:real
inicio
salario <- 1
repita
   leia(salario)
   soma <- soma + salario
ate salario = 0
escreval("A soma dos salarios é: ",soma)
fimalgoritmo
```

Execute 26.

Segue um algoritmo que escreve os 100 primeiros números pares.

```
algoritmo "paresate100"
var
 numero:inteiro

inicio
 numero <- 2
repita
   escreval(numero)
   numero <- numero + 2
ate numero > 100
fimalgoritmo
```

Execute 27.

Segue um algoritmo que receba o peso de 1000 porcos, ao final mostre o maior peso.

```
algoritmo "maiorporco"
var
 peso:real
 maior:real
 qtporcos:inteiro

inicio
 maior <- 0
 qtporcos <- 1
repita
```

```
    leia(peso)
    se (peso > maior) entao
        maior <- peso
    fimse
    qtporcos <- qtporcos + 1
ate qtporcos > 100
escreval("o Maior peso é: ",maior)
fimalgoritmo
```

Execute 28.

Segue um algoritmo que entre com vários números positivos e imprima a média dos números digitados.

```
algoritmo "numerospositivos"
var
   numero:inteiro
   soma:inteiro  // variável usada para calcular a soma dos numeros
   media:real    // variável usada para calcular a media dos numeros
   qtnum:inteiro // variável usada para contar a quantidade de numeros
inicio
   qtnum <- 0
   soma <- 0
   leia(numero)
   repita faca
      soma <- soma + numero
      qtnum <- qtnum + 1
      leia(numero)
   ate numero <=  0
   media <- soma / qtnum
```

```
    escreval("A média dos numeros positivos digitados é: ",media)
fimalgoritmo
```

Execute 29.

Segue um algoritmo que entre com o sexo de várias pessoas e imprima quantas são do sexo masculino e quantas são do sexo feminino.

```
algoritmo "verificarsexoquantidade"
var
  sexo:caracter
  qtmasc,qtfem:inteiro
inicio
  escreva("Entre com o sexo da pessoa ou digite s para sair:")
  leia(sexo)
  repita
    se (sexo = "f") ou (sexo = "F") entao
       qtfem <- qtfem + 1
    fimse
    se (sexo = "m") ou (sexo = "M") entao
              qtmasc <- qtmasc + 1
    fimse
    escreva("Entre com o sexo da pessoa ou digite s para sair:")
    leia(sexo)
  ate (sexo = "s") ou (sexo = "S")
  escreval("A quantidade de pessoas com sexo masculino é: ",qtmasc)
  escreval("A quantidade de pessoas com sexo feminino é:",qtfem)
fimalgoritmo
```

O algoritmo descrito anteriormente poderia ter sido criado com qualquer estrutura de repetição. Portanto podemos ter algoritmos que são escritos de maneiras diferentes, mas, funcionam realizando o mesmo objetivo.

Exercícios

Obs: . Abra o VisuAlg para realizar os seguintes exercícios.

Escreva um algoritmo para imprimir os números de 1 a 10 utilizando uma estrutura ENQUANTO e um contador.

Escreva um algoritmo para imprimir os número de 1 a 10 utilizando uma estrutura REPITA/ATÉ e um contador.

Escreva um algoritmo para repetir a leitura de um número enquanto o valor fornecido for diferente de 0. Para cada número fornecido, imprimir se ele é NEGATIVO ou POSITIVO. Quando o número 0 for fornecido a repetição deve ser encerrada sem imprimir mensagem alguma.

OBS: Utilize uma estrutura de repetição condicional no **final (Repita - Até**).

[Para os dados de entrada abaixo]	[Deve ser gerada a seguinte saída]
4	Positivo
-1	Negativo
2	Positivo
6	Positivo
-7	Negativo
-2	Negativo
0	

Escreva outra versão do algoritmo para resolver o problema anterior utilizando a estrutura com teste de saída no **início** da repetição (**Enquanto**).
Escreve um algoritmo para ler uma quantidade indeterminada de valores inteiros. Para cada valor fornecido escrever uma mensagem que indica se cada valor fornecido é PAR ou ÍMPAR. O algoritmo será encerrado imediatamente após a leitura de um valor NULO (zero) ou NEGATIVO.

[Para os dados de entrada abaixo]	[Deve ser gerada a seguinte saída]
11	Ímpar
3	Ímpar
2	Par
10	Par
5	Ímpar
-2	

Ler dois valores inteiros e imprimir o resultado da divisão do primeiro pelo segundo. Se o segundo valor informado for ZERO, deve ser impressa uma mensagem de VALOR INVÁLIDO e ser lido um novo valor. Ao final do programa deve ser impressa a seguinte mensagem: VOCE DESEJA OUTRO CÁLCULO (S/N). Se a resposta for S o programa devera retornar ao começo, caso contrário deverá encerrar a sua execução imprimindo quantos cálculos foram feitos.
OBS: O programa só deverá aceitar como resposta para a pergunta as letras S ou N.
Ler uma quantidade indeterminada de duplas de valores (2 valores de cada vez). Escrever para cada dupla uma mensagem que indique se ela foi informada em ordem crescente ou decrescente. A repetição será encerrada ao ser fornecido, para os elementos da dupla, valores iguais.

[Para os dados de entrada abaixo] [Deve ser gerada a seguinte saída]

5 4	Decrescente
7 2	Decrescente
3 8	Crescente
2 2	

Escreva um algoritmo para repetir a leitura de uma senha até que ela seja válida. Para cada leitura da senha incorreta informada escrever a mensagem "SENHA INVÁLIDA". Quanto a senha for informada corretamente deve ser impressa a mensagem "ACESSO PERMITIDO" e o algoritmo encerrado. Considere que a senha correta é o valor **2003.**

[Para os dados de entrada abaixo]	[Deve ser gerada a seguinte saída]
2200	Senha Inválida
1020	Senha Inválida
2022	Senha Inválida
2002	Acesso Permitido

Escreva um algoritmo para ler as coordenadas (X,Y) de uma quantidade indeterminada de pontos no sistema cartesiano. Para cada ponto escrever o quadrante a que ele pertence. O algoritmo será encerrado quando pelo menos uma de duas coordenadas for NULA (nesta situação sem escrever mensagem alguma).

[Para os dados de entrada abaixo]	[Deve ser gerada a seguinte saída]
2 2	primeiro
3 -2	quarto
4 7	primeiro
-8 -1	terceiro
-7 1	segundo
0 2	

Para que a divisão entre 2 números possa ser realizada, o divisor não pode ser nulo. Escreva um algoritmo para ler 2 valores e imprimir o resultado da divisão do primeiro pelo segundo.
OBS: O algoritmo deve validar a leitura do segundo valor (que não deve ser nulo). Enquanto for fornecido um valor nulo a leitura deve ser repetida. Utilize a estrutura REPITA/ATE na construção da repetição de validação.
Altere a solução do exercício 7 para que seja impressa a mensagem Valor inválido caso o segundo valor informado seja ZERO.
Reescreva o algoritmo para o problema 7 utilizando a estrutura ENQUANTO na construção da repetição de validação.
Altere a solução do exercício 9 para que seja impressa a mensagem Valor inválido caso o segundo valor informado seja 0.
Escreva um algoritmo para ler as notas da 1a. e 2a. avaliações de um aluno, calcular e imprimir a média semestral. Faça com que o algoritmo só aceite notas válidas(uma nota válida deve pertencer ao intervalo [0,10]. Cada nota deve ser validada separadamente. Deve ser impressa a mensagem "Nota inválida" caso a nota informada não pertença ao intervalo [0,10].
Reescreva o algoritmo para o problema 11 para que no final seja impressa a mensagem Novo cálculo (1.sim 2.não) solicitando ao usuário que informe um código (1 ou 2) indicando se ele deseja ou não executar o algoritmo novamente. Se for informado o código 1 deve ser repetida a execução de todo o algoritmo para permitir um novo cálculo, caso contrário ele deve ser encerrado.
Reescreva o algoritmo do exercício 12 validando a resposta do usuário para a pergunta Novo Cálculo (1.sim 2.não)? (aceitar apenas os código 1 ou 2).
Escreva um algoritmo para ler 2 notas de um aluno, calcular e imprimir a média final. Logo após escrever a mensagem "Calcular a média de outro aluno 1.Sim 2.Não?" e solicitar uma

resposta. Se a resposta for 1, o algoritmo deve ser executado novamente, caso contrário deve ser encerrado imprimindo a quantidade de alunos aprovados.
Reescreva o algoritmo do exercício 14 para que seja impresso no final, a quantidade de alunos aprovados, reprovados ou que ficaram em exame.

8.4.Escreva um algoritmo que verifique a validade de uma senha fornecida pelo usuário. A senha válida é o número 1234.

OBS: Se a senha informada pelo usuário for inválida, a mensagem "ACESSO NEGADO" deve ser impressa e repetida a solicitação de uma nova senha até que ela seja válida. Caso contrário deve ser impressa a mensagem "ACESSO PERMITIDO" junto com um número que representa quantas vezes a senha foi informada.

A Federação Gaúcha de Futebol contratou você para escrever um programa para fazer uma estatística do resultado de vários GRENAIS. Escreva um algoritmo para ler o número de gols marcados pelo Inter, o número de gols marcados pelo GRÊMIO em um GRENAL, imprimindo o nome do time vitorioso ou a palavra EMPATE. Logo após escrever a mensagem "Novo GRENAL 1.Sim 2.Não?" e solicitar uma resposta. Se a resposta for 1, o algoritmo deve ser executado novamente solicitando o número de gols marcados pelos times em uma nova partida, caso contrário deve ser encerrado imprimindo:

-Quantos GRENAIS fizeram parte da estatística.
-O número de vitórias do Inter.
-O número de vitórias do Grêmio.
-O número de Empates.
- Uma mensagem indicando qual o time que venceu o maior número de GRENAIS (ou NÃO HOUVE VENCEDOR).

Um Posto de combustíveis deseja determinar qual de seus produtos tem a preferência de seus clientes. Escreva um algoritmo para ler o tipo de combustível abastecido (codificado da seguinte forma: 1.Álcool 2.Gasolina 3.Diesel 4.Fim). Caso o usuário informe um código inválido (fora da faixa de 1 a 4) deve ser solicitado um novo código (até que seja válido). Ao ser informado o código do combustível, o seu respectivo nome deve ser impresso na tela. O programa será encerrado quando o código informado for o número 4 escrevendo então a mensagem: "MUITO OBRIGADO" e a quantidade de clientes que abasteceram cada tipo de combustível.

Arrays (Vetor e Matriz)

As variáveis compostas homogêneas, mais conhecidas como arrays, correspondem a conjuntos de elementos de um mesmo tipo, representados por um único nome.

Os arrays podem variar quanto a sua dimensão, isto é, a quantidade de índices necessária para a individualização de cada elemento do conjunto. O array unidimensional também é conhecido por vetor, enquanto o array bidimensional é denominado de matriz.

EXEMPLOS:

Vetor

V =

4	7	2	5	3

Matriz

M =

3	8	1	5
0	2	4	7
2	5	9	3

Array Tridimensional

T =

3	8	1	5	9	4	0	3
0	2	4	7	1	5		
2	5	9	3				

6	3	8	1
7	3	0	2	5	2

Cada elemento dos arrays podem ser referenciados através de índices. Exemplos:

V[1] = 4	M[1,1] = 3	T[1,1,1] = 3
V[2] = 7	M[2,3] = 4	T[2,3,2] = 9
V[5] = 3	M[3,1] = 2	T[1,2,3] = 3

VETORES

Vetores são arrays que necessitam de apenas um índice para individualizar um elemento do conjunto.

Vetores são arrays que necessitam de apenas um índice para individualizar um elemento do conjunto.

Declaração:

Para definirmos uma variável do tipo vetor, utilizamos a seguinte sintaxe:

variavel:vetor[índice-inicial..índice-final] de tipo
onde:
Variável são os nomes das variáveis que se deseja declarar;
índice-inicial é o limite inferior do intervalo de variação do índice;
índice-final é o limite superior do intervalo de variação do índice;
tipo é o tipo dos componentes da variável (inteiro, real, lógico e caracter

O índice-inicial e o índice-final devem ser do mesmo tipo escalar (inteiro, caracter ou logico).

EXEMPLO:

Declarar uma variável composta de 8 elementos numéricos de nome NOTA.

```
var
  NOTA : vetor[1..8] de real;
```

Fará com que passe a existir um conjunto de 8 elementos do tipo real, individualizáveis pelos índices 1, 2, 3, ..., 8.

nota[1]	nota[2]	nota[3]	nota[4]	nota[5]	nota[6]	nota[7]	nota[8]

Outros exemplos de declarações de vetores:

```
var
    IDADE : vetor[1..20] de inteiro;
        NOME : vetor[1..30] de caracter;
        QUANT : vetor['A'..'Z'] de inteiro;
        RECEITA,DESPESA : vetor[90..96] de real;
        VETOR: vetor[-5..5] de caracter;
```

Para processarmos individualmente todos os componentes de um vetor, é aconselhável o uso da estrutura PARA, para que possamos variar o índice do vetor.

Obs: . Abra o VisuAlg para realizar as seguintes execuções.

Execute 30.

Segue um algoritmo que dado um vetor A, preenchê-lo com o valor 30.

```
algoritmo "vetorde1a100"


var
  A : vetor[1..100] de inteiro
  indice:inteiro
inicio


  // preenchendo um vetor de 100 elementos com o valor 30
  para indice de 1 ate 100 faca
      A[indice] <- 30
  fimpara


  // imprimindo o conteúdo do vetor
  para indice de 1 ate 100 faca
      escreval("A[",indice,"]: ", A[indice])
  fimpara
fimalgoritmo
```

Execute 31.

Segue um algoritmo que dado um vetor A, preenchê-lo com os números inteiros 1,2,3,...,100.

```
algoritmo "vetorde1a100"


var
  A : vetor[1..100] de inteiro
  indice:inteiro
inicio


  para indice de 1 ate 100 faca
      A[indice] <- indice
  fimpara


  // imprimindo o conteúdo do vetor
  para indice de 1 ate 100 faca
      escreval("A[",indice,"]: ",A[indice])
  fimpara
fimalgoritmo
```

Execute 32.

Segue um programa que leia um vetor A contendo 30 números inteiros, calcule e exiba: maior elemento e a posição (índice) do maior elemento.

```
algoritmo "vetormaiorelemento"


var
  A : vetor[1..100] de inteiro
  maior:inteiro
  indice:inteiro
  posicao:inteiro
inicio
  maior <- 0
  para indice de 1 ate 30 faca
      leia(A[indice])
      se A[indice] > maior entao
         maior <- A[indice]
         posicao <- indice
      fimse
  fimpara
  escreval("O maior elemento é: A[",posicao,"]: ",maior)


fimalgoritmo
```

Execute 33.

Segue um programa para ler 20 números, calcular a média dos mesmos e exibir os números que estiverem acima da média.

```
algoritmo "medianumero"

```

```
var
  numero : vetor[1..20] de inteiro
  soma:inteiro
  indice:inteiro
  media:real
inicio
  soma <- 0
  // calculando a media dos numeros
  para indice de 1 ate 20 faca
      leia(numero[indice])
      soma <- soma + numero[indice]
  fimpara
  media <- soma / 20
  escreval("A média dos números: ",media)

  // imprimir os numero acima da media
  para indice de 1 ate 20 faca
      se (numero[indice] > media) entao
          escreval(numero[indice])
      fimse
  fimpara
 fimalgoritmo
```

Exercícios

Obs: . Abra o VisuAlg para realizar as seguintes execuções.

1. Dado um vetor A contendo 100 elementos inteiros, gerar e exibir um vetor B cujos elementos estão na ordem inversa de A.

Exemplo:

	1	2		99	100
A =	23	37	. . .	20	26

B =	26	20	. . .	37	23

2. Dado dois vetores A e B contendo 20 elementos inteiros cada, gerar e exibir um vetor C do mesmo tamanho cujos elementos sejam a soma dos respectivos elementos de A e B.
Exemplo:

	1	2	3		19	20
A =	23	37	30	. . .	45	35
B =	30	32	46	. . .	33	42
C =	53	69	76	. . .	88	77

3. Dado dois vetores A e B contendo 25 elementos inteiros cada, gerar e exibir um vetor C de 50 elementos, cujos elementos sejam a intercalação dos elementos de A e B.
Exemplo:

	1	2	3		24	25
A =	23	37	30	. . .	38	55
B =	30	32	46	. . .	43	49

	1	2	3	4	5	6		47	48	49	50
C =	23	30	37	32	30	46	. . .	38	43	55	49

4. Um time de basquete possui 12 jogadores. Deseja-se um programa que, dado o nome e a altura dos jogadores, determine:

o nome e a altura do jogador mais alto;
a média de altura do time;
a quantidade de jogadores com altura superior a média, listando o nome e a altura de cada um.

5. Fazer um programa para corrigir provas de múltipla escolha. Cada prova tem 10 questões e cada questão vale 1 ponto. O primeiro conjunto de dados a ser lido será o gabarito para a correção da prova. Os outros dados serão os números dos alunos e suas respectivas respostas, e o último número, do aluno fictício, será 0 (zero). O programa deverá calcular e imprimir:
para cada aluno, o seu número e sua nota;
o percentual de aprovação, sabendo-se que a nota mínima de aprovação é 6.
a nota que teve maior freqüência absoluta, ou seja, a nota que apareceu maior número de vezes (supondo a inexistência de empates).

A estrutura de dados para este programa de ser a seguinte:

GABARITO NUMERO NOTA

RESPOSTAS APROVADOS TOTAL

FREQUENCIA MAIOR PORCENT

SCRS 502 Bloco C - Entrada 22 (W2) ***Tel.: (61) 3225-7784***
CEP – 70330-530 ***Brasília - DF***

SCRS 502 Bloco C - Entrada 22 (W2) *Tel.: (61) 3225-7784*
CEP – 70330-530 *Brasília - DF*

MATRIZES

Matrizes são arrays que necessitam de dois índices para individualizar um elemento do conjunto. O primeiro índice representa as linhas e o segundo as colunas.

Declaração:

Para definirmos uma variável do tipo matriz, utilizamos a seguinte sintaxe:

variável :vetor[índice1-inicial..índice1-final,
índice2-inicial..índice2-final] de tipo

onde:
variável são os nomes das variáveis que se deseja declarar;
índice1-inicial é o limite inferior do intervalo de variação do primeiro índice;
índice1-final é o limite superior do intervalo de variação do primeiro índice;
índice2-inicial é o limite inferior do intervalo de variação do segundo índice;
índice2-final é o limite superior do intervalo de variação do segundo índice;
tipo é o tipo dos componentes da variável

o índice1-inicial e o índice1-final devem ser do mesmo tipo escalar (inteiro, caracter ou logico). O índice2-inicial também deve ser do mesmo tipo escalar do índice2-final.

EXEMPLO:

Declarar uma matriz M, de 4 linhas por 3 colunas, constituída de elementos numéricos inteiros.

var M : vetor[1..4,1..3] de inteiro;

fará com que passe a existir uma estrutura de dados agrupada denominada M, com 4x3=12 elementos inteiros, endereçáveis por um par de índices, com o primeiro indicando a linha e o outro, a coluna.

M =

m_{11}	m_{12}	m_{13}
m_{21}	m_{22}	m_{23}
m_{31}	m_{32}	m_{33}
m_{41}	m_{42}	m_{43}

Outros exemplos de declarações de matrizes:

```
var
M1 : vetor[1..4,80..90] de real;
     M2 : vetor['A'..'E',0..10] de caracter;
     M3 : vetor[-3..3,1..3] of inteiro;
```

Execute 34.

Segue um programa para ler uma matriz 3 x 5 de números inteiros e escrevê-la após ter multiplicado cada elemento por 2.

```
algoritmo "medianumero"
var
   M : vetor[1..3,1..5] de inteiro;
   nl:inteiro    // numero de linhas
   nc:inteiro    // numero de colunas
inicio
   // digitando os valores dentro da matriz
   para nl de 1 ate 3 faca
       para nc de 1 ate 5 faca
           leia(M[nl,nc])
       fimpara
   fimpara
   // imprimindo os valores multiplicado por dois
   para nl de 1 ate 3 faca
       para nc de 1 ate 5 faca
           escreva(M[nl,nc]*2," ")
       fimpara
       escreval("")
   fimpara

fimalgoritmo
```

Execute 35.

Dada uma matriz de 4 x 5 elementos inteiros leia os elementos e calcular a soma de cada linha. Obs: é usado um vetor para armazenar da soma de cada linha.

```
algoritmo "medianumero"
```

```
var
   M : vetor[1..4,1..5] de inteiro
   nl:inteiro    // numero de linhas
   nc:inteiro    // numero de colunas
   vsomalin:vetor[1..4] de inteiro
   somalin:inteiro
inicio

   // digitando os valores dentro da matriz
   para nl de 1 ate 4 faca
      somalin <- 0
      para nc de 1 ate 5 faca
         leia(M[nl,nc])3
         somalin <- somalin + M[nl,nc]
      fimpara
      vsomalin[nl] <- somalin
   fimpara

   // imprimindo os valores e a soma das linhas
   para nl de 1 ate 3 faca
      para nc de 1 ate 5 faca
         escreva(M[nl,nc]3," ")
      fimpara
      escreva("|",vsomalin[nl]
      escreval("")
   fimpara
fimalgoritmo
```

Exercícios

Obs: . Abra o VisuAlg para realizar as seguintes execuções.

1. Dado duas matrizes A e B, com 2 x 3 elementos inteiros cada, gerar e exibir uma matriz C do mesmo tamanho que resultará da soma da matriz A com a matriz B.

2. Faça um programa que leia uma matriz de ordem 3 x 5 de elementos inteiros, calcular e exibir:
o maior elemento da matriz;
a soma dos elementos da matriz;
a média dos elementos da matriz;

3. Dado uma matriz quadrada de ordem N, de elementos inteiros, exibir os elementos da diagonal principal, isto é, os elementos onde i = j. Obs: N será lido ($N \leq 10$).

4. Dado uma matriz A com 3 x 4 elementos inteiros, gerar e exibir uma matriz B que será a matriz transposta de A.

5. Faça um programa que leia o nome e as 3 notas dos 50 alunos de uma turma e:
calcule:
a média aritmética de cada aluno;
a situação de cada aluno; (aprovado se média superior ou igual a 7.0)
o número de alunos aprovados;
a média geral da turma;
exiba:
o nome e a situação de cada aluno;
o número de alunos aprovados;
a média geral da turma;
o nome e a média dos alunos com média superior ou igual à média geral da turma.
Use vetores para armazenar nome, média e situação, e uma matriz para armazenar as notas.

6. A tabela abaixo demonstra a quantidade de vendas dos fabricantes de veículos durante o período de 1993 a 1998, em mil unidades.

Fabricante / Ano	1993	1994	1995	1996	1997	1998
Fiat	204	223	230	257	290	322
Ford	195	192	198	203	208	228
GM	220	222	217	231	245	280
Wolkswagen	254	262	270	284	296	330

Faça um programa que:
leia os dados da tabela;
determine e exiba o fabricante que mais vendeu em 1996;
determine e exiba o ano de maior volume geral de vendas.
determine e exiba a média anual de vendas de cada fabricante durante o período.

MODULARIZAÇÃO

A modularização consiste num método utilizado para facilitar a construção de grandes programas, através de sua divisão em pequenas etapas, que são os módulos ou subprogramas. A primeira delas, por onde começa a execução do trabalho, recebe o nome de programa principal, e as outras são os subprogramas propriamente ditos, que são executados sempre que ocorre uma chamada dos mesmos, o que é feito através da especificação de seus nomes.

Vantagens da utilização de subprogramas:

Economia de código: escreve-se menos;
Desenvolvimento modularizado: pensa-se no algoritmo por partes;
Facilidade de depuração (correção/acompanhamento): é mais fácil corrigir/detectar um erro apenas uma vez do que dez vezes;
Facilidade de alteração do código: se é preciso alterar, altera-se apenas uma vez;
Generalidade de código com o uso de parâmetros: escreve-se algoritmos para situações genéricas.

Há duas espécies de subprogramas: PROCEDIMENTO e FUNÇÃO.

PROCEDIMENTO

Um subprograma do tipo PROCEDIMENTO é, na realidade, um programa com vida própria, mas que, para ser processado, tem que ser solicitado pelo programa principal que o contém, ou por outro subprograma, ou por ele mesmo.

Declaração:

```
PROCEDIMENTO nome

  declaração dos objetos locais ao Procedimento

  INICIO

    comandos do Procedimento

  FIMPROCEDIMENTO
```

onde: nome é o identificador associado ao procedimento.

EXEMPLO:

O programa abaixo calcula a média aritmética entre 2 notas, sem o uso de procedimentos.

```
algoritmo "CALCULA_MÉDIA"; {sem o uso de procedimentos}

var
  NOTA1,NOTA2,MEDIA : real
```

```
inicio
{lê as notas}
   escreva("Digite a primeira nota: ")
   leia(NOTA1)
   escreva("Digite a segunda nota: ")
   leia(NOTA2)
{calcula a media}
   MEDIA <- (NOTA1 + NOTA2) / 2
{escreve o resultado}
   escreval("Media = ",MEDIA)
fimalgoritmo
```

Execute 36.

Mostraremos agora o mesmo programa, utilizando um procedimento.

```
algoritmo "CALCULA_MÉDIA" {usando procedimento}
var
   NOTA1,NOTA2,MEDIA : real

{declaração do procedimento}
procedimento LER_NOTAS
inicio
     escreva("Digite a primeira nota: ")
     leia(NOTA1)
     escreva("Digite a segunda nota: ")
     leia(NOTA2)
fimprocedimento

{Programa Principal}
inicio
   LER_NOTAS {ativação do procedimento LER_NOTAS}
   MEDIA <- (NOTA1 + NOTA2) / 2 {calcula a media}
   escreval("Media = ",MEDIA) {escreve o resultado}
fimalgoritmo
```

FUNÇÃO

As funções, embora bastante semelhantes aos procedimentos, têm a característica especial de retornar ao programa que as chamou um valor associado ao nome da função. Esta característica permite uma analogia com o conceito de função da Matemática.

Declaração:

```
FUNCAO nome : tipo
  declaração dos objetos locais à Função
  INICIO
    Comandos da Função
  FIMFUNCAO
```

onde: ***nome*** é o identificador associado à função.
tipo é o tipo da função, ou seja, o tipo do valor de retorno.

EXEMPLO:

O programa abaixo calcula a média dos elementos de um vetor, sem uso de Procedimentos ou Funções.

```
algoritmo "SOMA_VETOR" {sem o uso de procedimentos ou funções}


var

    V : vetor[1..30] de inteiro

    I,soma: inteiro

    media : real

inicio

   {lê os valores do vetor}

   para I de 1 ate 30 faca

      leia(V[I])

   fimpara


   soma <- 0

   para I de 1 ate 30 faca

      soma <- soma + V[I]

   fimpara


   {calcula a media}
```

```
    media <- soma / 30


    {escreve o resultado}
    escreval(media)
fimalgoritmo
```

Execute 37.

Mostraremos agora o mesmo programa, utilizando um procedimento para ler os valores do vetor e uma função para efetuar o cálculo da média.

```
algoritmo "SOMA_VETOR" {usando uma função e um procedimento}


var
  V : vetor[1..3] de inteiro


{declaração do procedimento}
  procedimento LER_DADOS
    var I : inteiro
    inicio
      para I de 1 ate 3 faca
          leia(V[I])
            fimpara
  fimprocedimento


  {declaração da função}
  funcao CALCULAMEDIA : REAL
  var
   I,SOMA : inteiro
```

```
        MEDIA:REAL
        inicio
           SOMA := 0;
           para I de 1 ate 3 faca
               SOMA <- SOMA + V[I]
           fimpara
           MEDIA <- SOMA / 3
         RETORNE MEDIA
      fimfuncao

    {Programa Principal}
    inicio
    {ativa o procedimento LER_DADOS}
       LER_DADOS
    {escreve o resultado, chamando a função MEDIA}
       escreval(CALCULAMEDIA)
    fimalgoritmo
```

VARIÁVEIS GLOBAIS E VARIÁVEIS LOCAIS

Observe que, no exemplo anterior, declaramos uma variável no programa principal e outras nos subprogramas. Podemos dizer que a variável VETOR, que foi declarada no programa principal é uma variável global aos subprogramas, enquanto que a variável I é dita variável local ao procedimento LER_DADOS e as variáveis I e SOMA são locais à função MEDIA. É importante ressaltar que a variável I do procedimento LER_DADOS é diferente da variável I da função MEDIA, embora possuam o mesmo identificador.

o uso de variáveis globais dentro de procedimentos e funções serve para implementar um mecanismo de transmissão de informações de um nível mais externo para um mais interno.

As variáveis locais dos procedimentos e funções são criadas e alocadas quando da sua ativação e automaticamente liberadas quando do seu término.

A utilização de variáveis globais não constitui, no entanto, uma boa prática de programação. Assim, todos subprogramas devem apenas utilizar as variáveis locais, conheci-

das dentro dos mesmos, e a transmissão de informações para dentro e fora dos subprogramas deve ser feita através dos parâmetros de transmissão, que serão apresentados a seguir.

PARÂMETROS

Quando se deseja escrever um subprograma que seja o mais genérico possível, deve-se usar a passagem de parâmetros.

A passagem de parâmetros formaliza a comunicação entre os módulos. Além disto, permite que um módulo seja utilizado com operandos diferentes, dependendo do que se deseja do mesmo.

Dá-se a designação de parâmetro real ou de chamada ao objeto utilizado na unidade chamadora/ativadora e de parâmetro formal ou de definição ao recebido como parâmetro no subprograma.

Dentre os modos de passagem de parâmetros, podemos destacar a passagem por valor e a passagem por referência.

Na passagem de parâmetros por valor, as alterações feitas nos parâmetros formais, dentro do subprograma, não se refletem nos parâmetros reais. O valor do parâmetro real é copiado no parâmetro formal, na chamada do subprograma. Assim, quando a passagem é por valor, isto significa que o parâmetro é de entrada.

Na passagem de parâmetros por referência, a toda alteração feita num parâmetro formal corresponde a mesma alteração feita no seu parâmetro real associado. Assim, quando a passagem é por referência, isto significa que o parâmetro é de entrada-saída.

PROCEDIMENTO nome (lista de parâmetros formais)

FUNCAO nome (lista de parâmetros formais) : tipo

A lista de parâmetros formais tem a seguinte forma:

parâmetro1 : tipo; parâmetro2 : tipo; ...; parâmetro n : tipo

Exemplos da lista de parâmetros:

PROCEDIMENTO testeproc(X,Y,Z:inteiro; K:real)

funcao testefunc(A,B:real; C:caracter) : inteiro

Na forma apresentada, a passagem dos parâmetros será por valor. Para se utilizar a passagem por referência, deve-se acrescentar a palavra VAR antes do nome do parâmetro.

Execute 38.

procedimento PROC(A:inteiro; var B,C:inteiro)

Na chamada de procedimentos ou funções utilizando parâmetros, devemos acrescentar após o nome do procedimento ou função uma lista de parâmetros reais (de chamada), os quais devem ser do mesmo tipo e quantidade dos parâmetros formais declarados.

O exemplo a seguir demonstra a diferença entre a passagem de parâmetros por referência e a passagem de parâmetros por valor:

```
algoritmo "EXEMPLO_PASSAGEM_PARÂMETROS"

var
  N1,N2 : inteiro

procedimento PROC(X:inteiro; var Y:inteiro)
   inicio
          X:=1
     Y:=1
fimprocedimento

inicio
   N1<-0
   N2<-0
   PROC(N1,N2)
   escreval(N1) {será exibido o valor 0}
   escreval(N2) {será exibido o valor 1}
fimalgoritmo
```

Execute 39.

Escrever uma função chamada MAIOR que receba dois números inteiros e retorne o maior deles. Escrever um programa para ler dois números inteiros e, utilizando a função MAIOR, calcular e exibir o maior valor entre os números lidos.

```
algoritmo "CALCULA_MAIOR"


var
   X,Y,M : inteiro


funcao MAIOR (NUM1,NUM2:inteiro) : inteiro
   inicio
         se NUM1 > NUM2 entao
        retorne NUM1
      senao
         retorne NUM2
   fimse
fimfuncao


inicio
   leia(X,Y)
   M <- MAIOR(X,Y)
   escreva(M)
fimalgoritmo
```

Execute 40.

Escrever um procedimento chamado DOBRA que multiplique um número inteiro (recebido como parâmetro) por 2. Escrever um programa para ler um valor inteiro e , utilizando o procedimento DOBRA, calcular e exibir o dobro do mesmo.

```
algoritmo "CALCULA_DOBRO"


var
   x: inteiro


procedimento DOBRA (var NUM:inteiro)
   inicio
         NUM := NUM * 2
fimprocedimento


inicio
   leia(x)
   DOBRA(x)
   escreval(x)
fimalgoritmo
```

Exercícios

01. Escreva um procedimento que exiba o seu nome.

02. Escreva um procedimento que receba um valor string S e um valor inteiro positivo N e exiba o string S por N vezes seguidas na tela.

03. Escreva uma função chamada CUBO que receba um valor do tipo real e retorne a potência elevado a 3 do mesmo.

04. Escreva um procedimento chamado TROCA que receba 2 variáveis inteiras (X e Y) e troque o conteúdo entre elas;

05. Escreva um procedimento chamado SINAL que receba como parâmetro um valor N inteiro e escreva a palavra POSITIVO se N for um número maior que zero, NEGATIVO se N for menor que zero, ou ZERO se N for igual a zero.
Escreva um programa que leia um número inteiro e, usando o procedimento SINAL, mostre se ele é maior, menor ou igual a zero.

06. Escreva um procedimento chamado METADE que divida um valor do tipo real (passado como parâmetro) pela metade.
Escreva um programa que leia um vetor A de 30 elementos reais e, usando o procedimento METADE, divida todos seus elementos pela metade.

07. Escreva uma função chamada MEDIA que retorne a média de 3 valores reais (X, Y e Z) passados como parâmetros.
Escreva um programa que, para um número indeterminado de alunos, faça para cada uma deles:
ler o nome e as 3 notas do aluno (a leitura do nome FIM indica o fim dos dados - flag);
calcule a média do aluno (usando a função MEDIA);
exiba o nome e a média do aluno.

08. Escreva uma procedimento chamado AUMENTO que receba dois valores reais X e Y como parâmetros e aumente o valor de X em Y%.
Escreva um programa que leia uma variável K do tipo real e, para um número indeterminado de funcionários de uma empresa, faça para cada uma delas:
ler a matrícula, o nome e o salário (a leitura da matrícula 0 (zero) indica o fim dos dados - flag);
aumente o salário em K% (usando o procedimento AUMENTO) e exiba o salário aumentado.

09. Escreva um programaque leia as 3 notas e o número de faltas de um aluno, calcule a sua média e determine e exiba a sua situação. Caso a aluno tenha mais de 10 faltas, ele está REPROVADO POR FALTA. Caso contrário, estará REPROVADO se sua média for menor que 5.0 ou APROVADO se sua média for superior a 5.0.
Observações:
utilize uma função para calcular a média e um procedimento para determinar e exibir a situação do aluno;
não utilize variáveis globais.

10. Escreva uma função chamada SEG para receber uma medida de tempo expressa em Horas, Minutos e Segundos e retornar esta medida convertida apenas para segundos.
Escreva um procedimento chamado HMS para receber uma medida de tempo expressa apenas em segundos em retornar esta medida convertida para horas, minutos e segundos.
Faça um programa que leia 2 medidas de tempo (expressas em horas, minutos e segundos) e, usando a função SEG e o procedimento HMS, calcule e exiba a diferença (também em horas, minutos e segundos) entre elas.

11. Escreva uma função chamada NOME_MES que receba um valor inteiro N (de 1 a 12) e retorne um string contendo o nome do mês correspondente a N.
Faça um programa que leia uma data (no formato dia, mês e ano) e, usando a função NOME_MES, exiba a data lida no formato abaixo:
EXEMPLO:
Entrada: 23 11 1998 Saída: 23 de novembro de 1998

12. Escreva uma função chamada DIAS_ANO que receba 3 valores inteiros (DIA, MES, ANO) e retorne o número de dias decorridos no ano até o dia/mês/ano fornecido.
Escreva um função booleana chamada DATA_VALIDA que receba uma data (DIA, MÊS, ANO) e verifique se a data é válida (considerando os anos bissextos).
Faça um programa que leia 2 datas, no formato dia, mês e ano (as datas devem ter o mesmo ano) verificando se as mesmas são válidas (através da função DATA_VALIDA), calcule e exiba a diferença de dias entre elas (usando a função DIAS_ANO).

RECURSIVIDADE

A recursividade é uma característica que alguns problemas apresentam: a de serem definidos em termos deles mesmos. Todo problema que se comporta assim é dito ser recursivo.

A recursão é uma técnica apropriada se o problema a ser resolvido tem as seguintes características:

a resolução dos casos maiores do problema envolve a resolução de um ou mais casos menores;
os menores casos possíveis do problema podem ser resolvidos diretamente;
a solução iterativa do problema (usando enquanto, para ou repita) é complexa.

Exemplo:

O problema do fatorial é recursivo por definição:

$N! = N \times (N - 1) \times (N - 2) \times (N - 3) \ldots \times 1$ (Equação 1)

Existe um caso especial: 0! é igual a 1, por definição.

A partir da equação 1, podemos concluir que o fatorial de N está expresso em termos do fatorial de N-1.

$$N! = N \times (N - 1) \times \underbrace{(N - 2) \times (N - 3) \times \ldots \times 1}_{(N - 1)!}$$

Resumindo:

$N! = N \times (N - 1)!$ (Equação 2)

A equação 2 é válida para todos os números inteiros com exceção do 0 (zero), sendo, portanto, necessário um tratamento especial.

Execute 41.

O programa recursivo em Visualg para o cálculo do fatorial de N ficaria assim:

```
algoritmo "FATORIAL"


var
   N,F : inteiro


{função recursiva que retorna o fatorial de N}


funcao FAT(N:inteiro):inteiro
   var
      F :inteiro
   inicio
      se N = 0 entao
         F <- 1
      senao
         F <- N * FAT (N - 1)
      fimse
      retorne F
fimfuncao
fimalgoritmo
```

Note que para cada chamada da função recursiva FAT deve ser um criado um novo nível de ativação, guardado em uma pilha, onde para cada um destes níveis de ativação têm-se um parâmetro de chegada e um valor de rechamada para uma nova ativação ou um valor de retorno.

Criaremos uma tabela para melhor expor a observação acima e para isso vamos calcular o fatorial de 3:

Número da ativação	Valor de chegada	Rechamada	Valor de retorno
1	3	3 * FAT(3-1)	
2	2	2 * FAT(2-1)	
3	1	1 * FAT(1-1)	
4	0		1
3			1
2			2
1			6

Observe que até ser atendida a condição de retorno (N = 0), têm-se apenas reativações de FAT. Quando o valor de chegada é igual a 0 (na ativação 4), começa o processo de retorno, onde cada nível imediatamente superior recebe o resultado do nível inferior. Quando um nível recebe o resultado do seu nível inferior ele o aplica para poder calcular a expressão FAT <- N * FAT(N-1) até então indefinida, retornando para o nível de cima o seu valor calculado.

É importante que seja observada a condição de encerramento da recursão para evitar que este processo continue indefinidamente.

CONCLUINDO:

A recursividade é uma técnica elegante e quem a domina, geralmente, demonstra experiência, porém possui um preço: a movimentação de dados na PILHA (controle interno da linguagem para possibilitar ativações recursivas). Essa movimentação de dados de controle, impõe à solução um tempo adicional que pode torná-la ineficiente.

Devido a esse tempo extra utilizado pelas soluções recursivas, deve-se dar preferência às soluções iterativas, deixando a utilização da recursividade para os casos apropriados (atentar para características básicas de um problema recursivo).

Arquivo

Muitas vezes é necessário repetir os testes de um programa com uma série igual de dados. Para casos como este, o VisuAlg permite o armazenamento de dados em um arquivo-texto, obtendo deles os dados ao executar os comandos leia.

Esta característica funciona da seguinte maneira:

Se **não existir** o arquivo com nome especificado, o VisuAlg fará uma leitura de dados através da digitação, armazenando os dados lidos neste arquivo, na ordem em que forem fornecidos.
Se o arquivo **existir**, o VisuAlg obterá os dados deste arquivo até chegar ao seu fim. Daí em diante, fará as leituras de dados através da digitação.
Somente um comando arquivo pode ser empregado em cada pseudocódigo, e ele deverá estar na seção de declarações (dependendo do "sucesso" desta característica, em futuras versões ela poderá ser melhorada...).
Caso não seja fornecido um caminho, o VisuAlg irá procurar este arquivo na pasta de trabalho corrente (geralmente, é a pasta onde o programa VISUALG.EXE está). Este comando não prevê uma extensão padrão; portanto, a especificação do nome do arquivo deve ser completa, inclusive com sua extensão (por exemplo, .txt, .dat, etc.).
A sintaxe do comando é:
arquivo <nome-de-arquivo>
<*nome-de-arquivo*> é uma constante caractere (entre aspas duplas).

Execute 42.

Veja o exemplo a seguir:

```
algoritmo "lendo do arquivo"
arquivo "teste.txt"
var x,y: inteiro
inicio
para x de 1 ate 5 faca
  leia (y)
fimpara
fimalgoritmo
```

Observe que é criado um arquivo dentro da pasta onde esta o programa com o nome de teste.txt contendo os números digitados no programa.

Execute 43.

Faça um algoritmo que leia (digitados) o nome de 20 pessoas e armazene em um arquivo chamado amigos.txt.

```
algoritmo "lendo do arquivo"
arquivo "amigos.txt"
var x: inteiro
   nome:caracter
inicio
para x de 1 ate 20 faca
  leia (nome)
fimpara
fimalgoritmo
```