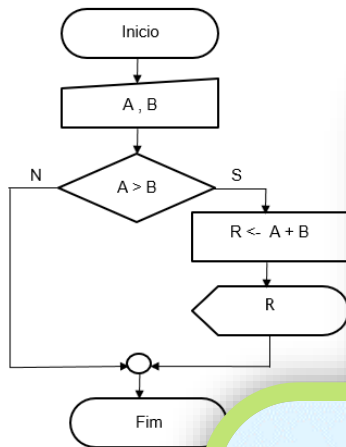


Técnicas e Lógica de Programação

Narrativa, Fluxograma e Pseudocódigo



```
Algoritmo "Nome do Algoritmo"
2 // Disciplina : Técnicas e Lógica de Programação
3 // Professor : Flavio Mota da Cruz
4 // Descrição :
5 // Autor(a) : Flavio Mota da Cruz
6 // Data atual :
7
8 Var
9
10 Nota1, Nota2:real // Exemplos de variáveis
11 Media:real
12 Inicio
13
14
15 Leia(Nota1) // Comando para ler a variável Nota1
16 Leia(Nota2) // Comando para ler a variável Nota2
17
18 Media <- (Nota1 + Nota2) / 2 // Processamento de dados
19
20 // Exemplo de saída de dados
21 Escreva(Media)
```

VisuALG



Professor: Flávio Mota

Apresentação do professor.....	2
Introdução.....	3
Lógica de Programação.....	3
O que é Algoritmo?.....	4
Tipos de Algoritmos.....	5
Principais fluxos dos algoritmos.....	5
Algoritmo narrativo sequencial	7
Algoritmo narrativo com decisão (Condicional).....	8
Algoritmo narrativo estrutura de repetição	9
É hora dos exercícios	11
Fases de um ALGORITMO computacional	12
É hora dos exercícios.	14
Fluxograma ou Diagrama de Blocos.....	16
Fluxograma/Diagrama de blocos Sequencial	17
Fluxograma/Diagrama de Blocos - decisão simples.	17
Fluxograma/Diagrama de Blocos - decisão composta.....	18
Fluxograma/Diagrama de Blocos - laço de repetição com condição no inicial.....	18
Fluxograma/Diagrama de Blocos - laço de repetição com condição no final.....	19
Fluxograma/Diagrama de Blocos - laço iterativo.	19
Pseudocódigo ou Portugol.....	23
O que é o interpretador de pseudocódigo VisualG 3.0?.....	24
Variáveis.....	25
Por que declarar variáveis e como nomeá-las?	26
Tipos de dados.....	27
Constantes	28
Operador de atribuição.....	28
Linearização de Expressões.....	29
Operadores Aritméticos	30
Operadores Relacionais	30

Operadores Lógicos	30
Comandos de ENTRADA e SAÍDA.....	32
Entrada de Dados.....	32
Saída de Dados.....	33
Comandos de desvio condicional (SE, SENAO, SENAO SE)	45
Comando de seleção múltipla (Escolha caso)	50
Comando de repetição (PARA DE ATE ...FAÇA)	52
Comando de repetição (ENQUANTO ...FAÇA)	56
Comando de repetição (REPITA ...ATE).....	59
VETOR: estrutura indexada unidimensional	62
MATRIZ: estrutura indexada bidimensional.....	65
PROCEDIMENTO: Subprograma, sub-rotina, método ou módulo	69
Procedimentos com parâmetros	72
Função: Subprograma, sub-rotina, método ou módulo com retorno.	73
Exercícios no CADERNO 01: Algoritmos nível 05 - PROCEDIMENTOS E FUNÇÕES	76

Apresentação do professor

Olá, meu nome é Flavio Mota, sou Bacharel em Sistemas de Informação, pós-graduado em Engenharia de Sistemas e licenciado pelo Centro Paula Souza, importante órgão do governo do estado de São Paulo. Esse órgão faz a Gestão das Escolas Técnicas e Faculdades de Tecnologia deste estado. Sou professor há mais de 15 anos da área de Tecnologia da Informação, ministro aulas nos cursos técnicos em Informática e Desenvolvimento de Sistemas nas escolas técnicas do estado de SP, já formei mais de 2000 alunos na área de Desenvolvimento de Sistemas. Desenvolvo minhas aulas em especial para os componentes voltados para Lógica de Programação, programação orientada a objetos e programação C# para desktop, web e mobile.

Sou casado com uma mulher maravilhosa, chamada Rochele Mota e pai de uma menina linda chamada Raquel Mota. Busco sempre estar em sintonia com os mandamentos do nosso Senhor Jesus Cristo, pois creio que Ele é nosso salvador e quem abençoa as boas intenções do nosso coração. Estou escrevendo esse material para que o mesmo possa ajudar e impactar muitas pessoas que buscam aprender a arte de programar sistemas computacionais, onde apresento os conceitos básicos e sólidos para qualquer linguagem de programação moderna, seja em qualquer plataforma das tecnologias atuais como: Desktop, Web ou Mobile.

Introdução

Programar sistemas seja ele para desktops (computadores de mesa/notebook), celulares, tablets ou para web é uma tarefa um pouco complexa, mas muito prazerosa, não importa sua idade, profissão ou objetivo, você precisa apenas de um pouco de dedicação e sentir prazer pelo momento de aprendizado que os desafios nas resoluções de problemas vão lhe trazer. Pois é, programar é na sua essência resolver pequenos, médios ou grandes problemas através do mundo computacional e tudo isso começa por aqui, pois para programar, antes vocês devem aprender as técnicas e a lógica de programação.

Esse material traz um pouco do meu conhecimento adquirido ao longo dos anos como professor e Analista de Sistemas. Fazer a máquina pensar e resolver problemas por você é extraordinário e além de tudo, pode ser muito útil.

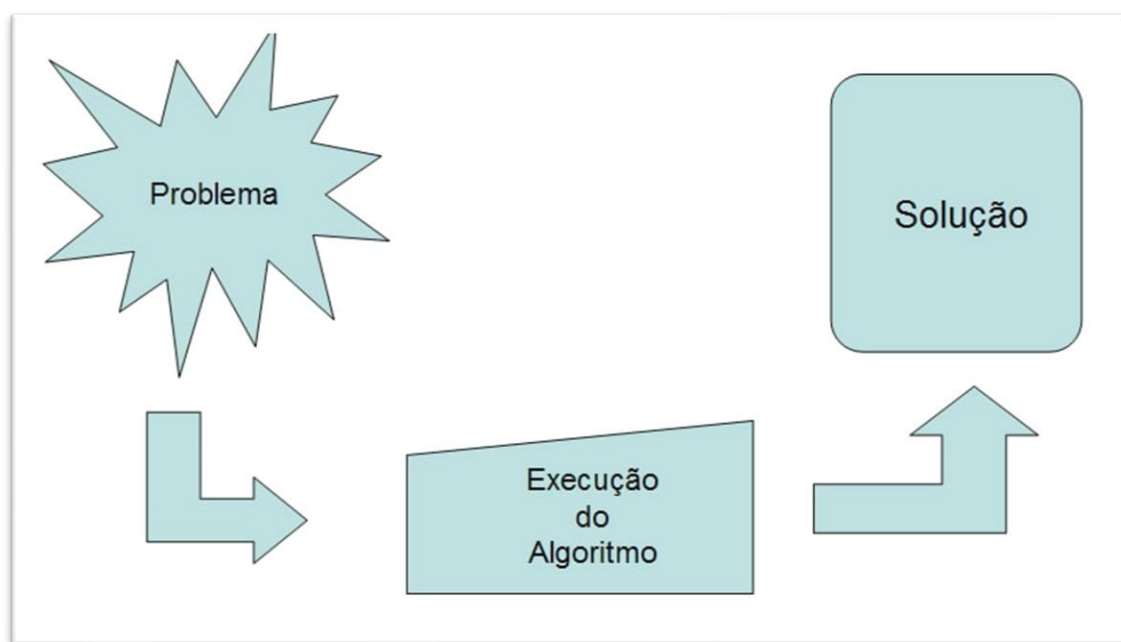
Lógica de Programação

É a técnica de desenvolver algoritmos (sequências lógicas) para atingir determinados objetivos dentro de regras baseadas na Lógica Matemática que são adaptados para as linguagens de programação. Utilizadas pelos programadores no desenvolvimento de sistemas computacionais baseado em múltiplas plataformas, como os computadores pessoais (Desktops/Notebooks), Web e Mobile e mais recentemente em diversos tipos de objetos onde o mercado chama de IOT (Internet das coisas). Podemos então definir a lógica como encadeamento de pensamentos para atingir um determinado objetivo ou realizar uma tarefa.

O que é Algoritmo?

Um algoritmo é uma sequência lógica de instruções que devem ser seguidas para a resolução de um problema ou para a execução de uma tarefa, usamos algoritmos mesmo sem saber em qualquer atividade do dia a dia, desde as atividades mais triviais a mais complexa, temos sempre que seguir passos lógicos para desenvolver nossas atividades, esses passos podemos chamar de algoritmos não computacional ou algoritmos narrativos.

Definição de um algoritmo



Para resolvermos qualquer problema, primeiro temos que entender o problema, depois precisamos executar alguns passos para que nosso problema seja resolvido e termos nossa solução mais adequada, pois para um problema podemos ter mais que uma solução e temos sempre que buscar a melhor solução.

Tipos de Algoritmos

- **Descrição narrativa** – utiliza linguagem natural para especificar os passos para a realização das tarefas ou solução de um problema.
- **Fluxograma** – é uma forma universal de representação, pois se utiliza de figuras geométricas para ilustrar os passos a serem seguidos para a resolução dos problemas.
- **Pseudocódigo** – utiliza linguagem estruturada e se assemelha na forma de um programa escrito em linguagem de programação estruturada, também conhecido como português estruturado ou Portugol, muito usado no meio computacional.

Principais fluxos dos algoritmos

- **Algoritmo sequencial:** é aquele onde os comandos são executados linha-a-linha, ou seja, na sequência pré-estabelecida na criação. No sequencial, há apenas um único fluxo de execução.
- **Algoritmo condicional:** é onde utilizamos condições (expressões lógicas). Neste tipo, o fluxo de instruções a ser seguido, será definido pela avaliação dessa condição, que pode ser uma ou mais.
- **Algoritmo de repetição:** é aquele onde um mesmo processamento (conjunto de instruções) é utilizado mais de uma vez. Essas repetições, também chamadas de laços.

Descrição Narrativa



Utiliza linguagem natural para especificar os passos na solução de problemas e na realização das tarefas.

Algoritmo narrativo sequencial

Problema observado: Preparar um bolo (Exemplo 01)

Início

- Bata quatro claras em neves;
- Adicione duas xícaras de açúcar;
- Adicione duas xícaras de farinha de trigo, quatro, gemas, uma colher de fermento e duas colheres de chocolate;
- Bata por três minutos;
- Coloque em uma assadeira com margarina e farinha de trigo;
- Coloque o bolo no forno para assar durante vinte minutos;
- Retire o bolo do forno.

Fim

Problema observado: Trocar uma lâmpada (Exemplo 02)

Início

- Pegue uma escada;
- Posicione a escada embaixo da lâmpada;
- Pegue uma lâmpada nova;
- Suba na escada;
- Retire a lâmpada velha;
- Coloque a lâmpada nova.

Fim

Algoritmo narrativo com decisão (Condicional)

Podemos ter sequências de tarefas onde alguns passos podem ou não serem executados dependendo de algumas condições que o processo pode nos oferecer, para esses casos podemos usar os algoritmos com decisões ou condicionais, no exemplo anterior fizemos um algoritmo para trocar uma lâmpada, mas não verificamos se a lâmpada precisaria ser trocada, segue um exemplo para essa situação usando uma condição.

Descrição narrativa usando condicional (Exemplo 01)

Início

- Ligue o interruptor
- Se a lâmpada não acender:
 - Pegue uma escada;
 - Posicione a escada embaixo da lâmpada;
 - Pegue uma lâmpada nova;
 - Suba na escada;
 - Retire a lâmpada velha;
 - Coloque a lâmpada nova

Fim

Descrição narrativa usando condicional (Exemplo 02)

Problema Observado: Fazer uma ligação Telefônica

Início

- Pegue o telefone
- Abra a agenda / últimos contatos
- Procure o contato
- Selecione o número / Digite
- Escute o som de chamada
- Se atender
 - Fale com seu contato
 - Encerre a chamada
 - Feche o aplicativo de discagem
- Se não atender
 - Feche o aplicativo de discagem

Fim

Algoritmo narrativo estrutura de repetição

Note que, apesar de nosso algoritmo de trocar lâmpada estar verificando a necessidade de trocar a lâmpada antes de fazê-lo, em momento algum verificamos se a lâmpada nova que foi instalada funciona. Assim, vamos tentar alterar o nosso algoritmo a fim de garantir que ao fim de sua execução tenhamos uma lâmpada funcionando. Para isso, vamos incluir um novo teste em seu final:

Problema observado: Trocar uma lâmpada (Exemplo 01)

Início

- Ligue o interruptor;
- Se a lâmpada não acender:

- Pegue uma escada;
- Posicione a escada embaixo da lâmpada;
- Pegue uma lâmpada nova;
- Suba na escada;
- Retire a lâmpada velha;
- Coloque a lâmpada nova.
- Enquanto a lâmpada não acender:
 - Retire a lâmpada;
 - Coloque uma outra lâmpada.

Fim

Assim, neste algoritmo, enquanto a condição definida for verdadeira (ou seja, enquanto a lâmpada não acender), as ações serão repetidas. Abstraímos os fatos de ter de descer da escada para pegar uma lâmpada nova e subir novamente.

Algoritmo para fazer uma prova (Exemplo 02)

Início

- ler a prova
- pegar a caneta
- enquanto ((houver questão em branco) e (tempo não terminou)) faça
 - se (souber a questão)
 - resolvê-la
 - senão
 - pular para outra
- Entregar a prova.

Fim

É hora dos exercícios

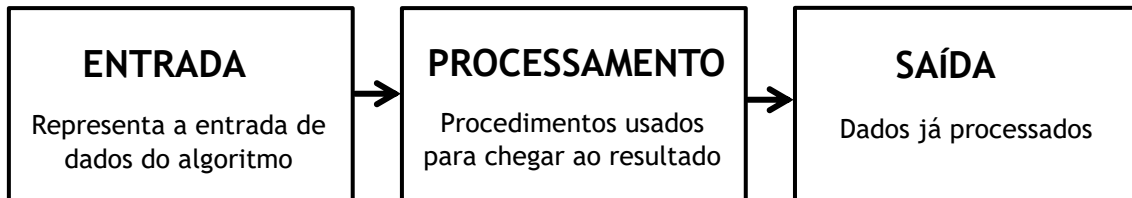
Usando os conceitos de algoritmos em sua forma narrativa resolva os seguintes problemas.

- 1 – Criar um algoritmo narrativo para fazer um bolo.
- 2 – Criar um algoritmo narrativo para trocar o pneu de um carro.
- 3 – Criar um algoritmo narrativo para uma partida de jogo da velha.
- 4 – Criar um algoritmo narrativo para o dia do seu aniversário.
- 5 – Criar um algoritmo narrativo para uma compra no mercado.

Resolução no CADERNO 02 - Resolução Algoritmos Narrativos.

Fases de um ALGORITMO computacional

Quando pensamos em algoritmos computacionais, podemos ter basicamente três passos:



Entrada de dados: geralmente quando vamos desenvolver algoritmos direcionadas para um sistema computacional podemos usar dados de entrada para que o sistema faça o processamento desses dados, esses dados serão digitados por uma pessoa ou enviados por outro sistema computacional.

Processamento: Quando o sistema recebe os dados, logo começa o processamento desses dados, transformando-os em informações muitas vezes de saída para os usuários ou para outro sistema.

Saída de dados: Podemos ter diversas formas de saída de dados, seja em tela, impressão ou enviado para outro sistema que a solicitou, são as informações já processadas.

Vamos fazer dois exemplos para o melhor entendimento dos conceitos de entrada, processo e saída de dados.

Usaremos nesses exemplos dois comandos muito utilizados nos algoritmos de pseudocódigos que vamos usar com mais frequência nos capítulos posteriores. Esses comandos são o **LEIA** (Entrada de dados) que representa quando o usuário digita um valor e o **ESCREVA** (Saída de dados) quando o sistema mostra alguma informação para o usuário.

Exemplo 01

Nesse algoritmo será feito o cálculo da média de 4 notas.

1 - Quais os dados de entrada?

Entraremos com 4 notas representadas por – **N1, N2, N3, N4.**

2 - Qual será o processamento?

O processamento será a soma das 4 notas e o resultado dividido por quatro.

3 - Quais os dados de saída?

O dado de saída será a média ou resultado final.

ENTRADA DE DADOS

Leia N1 (5)

Leia N2 (6)

Leia N3 (6)

Leia N4 (7)

PROCESSO

Soma = $N1 + N2 + N3 + N4 = (24)$

Média = $Soma / 4 = (24 / 4) = (6)$

SAÍDA

Escreva Média = (6)

Exemplo 02

Nesse algoritmo será feito o cálculo para converter um valor em REAL para o seu valor correspondente em DÓLAR, considerando que o dólar vale R\$ 4,00.

1 - Quais os dados de entrada?

Entraremos com um valor **X** representando o valor em real.

2 - Qual será o processamento?

O processamento será a divisão do valor X por 4 que é o valor de cada Dólar.

3 - Quais os dados de saída?

O dado de saída será o valor do Real convertido em Dólar.

ENTRADA DE DADOS

Leia X (20)

PROCESSO

Dolar = $X / 4 = (5)$

SAÍDA

Escreva Dolar = (5) - Resultado da conversão de R\$ 20,00 para Dólar.

É hora dos exercícios.

Para os exercícios a seguir, use as etapas de entrada, processo e saída conforme os exemplos anteriores.

1 – Leia dois valores, faça as quatro operações básicas da matemática (adição, subtração, divisão e multiplicação) com os dois valores e mostre os resultados como saída.

2 – Fui ao mercado com R\$ 50,00, fiz uma compra de R\$ 27,50, mostre como saída o troco que sobrou das compras.

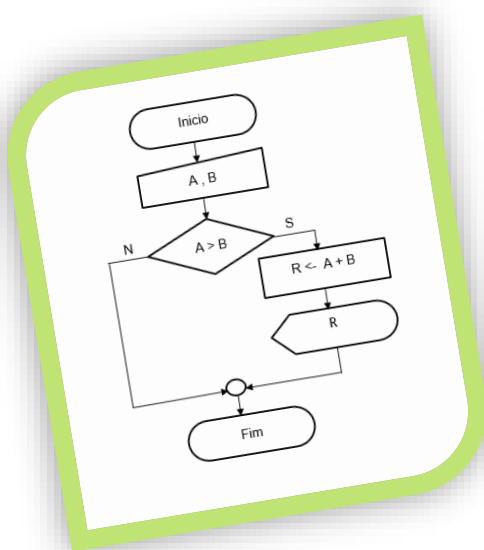
3 - Escreva um algoritmo para ler as dimensões de um retângulo (base e altura), calcular e escrever a área do retângulo.

4 - Faça um algoritmo que leia a idade de uma pessoa expressa em anos, meses e dias e escreva a idade dessa pessoa expressa apenas em dias. Considerar ano com 365 dias e mês com 30 dias.

5 - Escreva um algoritmo para ler o salário mensal atual de um funcionário e o percentual de reajuste. Calcular e escrever o valor do novo salário.

Resolução no CADERNO 02 - Resolução Algoritmos Narrativos.





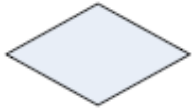


Fluxograma ou Diagrama de Blocos



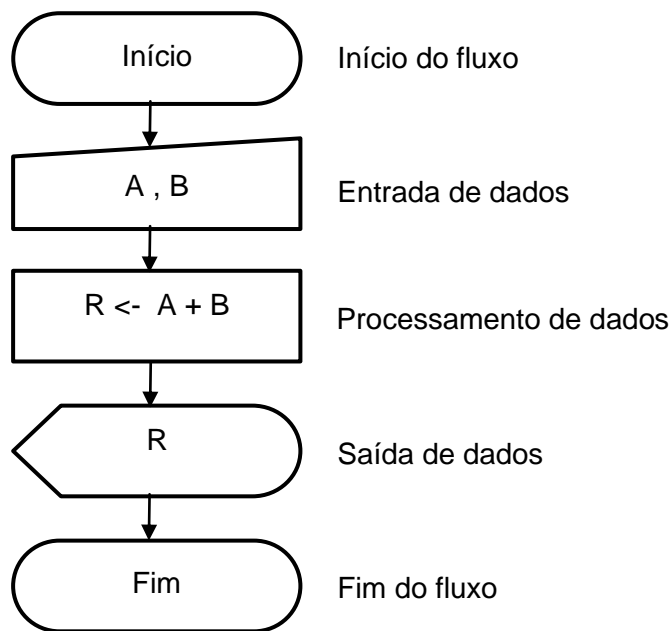
É uma forma universal de representação, pois se utiliza de figuras geométricas para ilustrar os passos a serem seguidos na resolução dos problemas.

Fluxograma ou Diagrama de Blocos

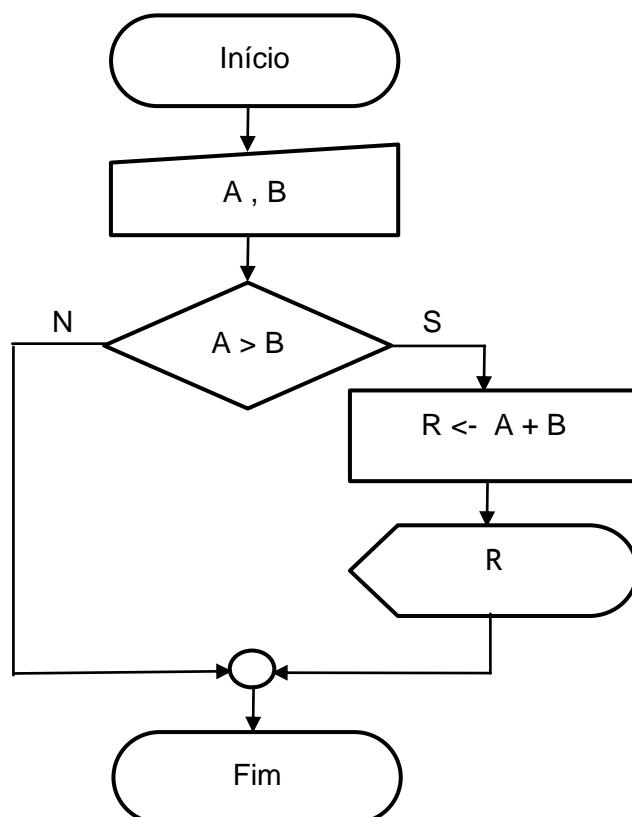
Esta forma de representação, utiliza símbolos gráficos para representar algoritmos. Existem símbolos padronizados para cada tipo de instrução, como por exemplo, início, entrada de dados, processamento (cálculos), saída de dados, fim, decisão, etc. Aqui estão alguns símbolos mais comuns, utilizados em Fluxogramas:

SÍMBOLO	NOME	FUNÇÃO
	TERMINAL	Indica o início ou fim de um algoritmo.
	PROCESSAMENTO	Representa um processamento
	ENTRADA MANUAL DE DADOS	Indica entrada de dados manual, através de um teclado por exemplo.
	EXIBIR	Representa a exibição dos dados e informações, através de um monitor por exemplo.
	DECISÃO	Representa um teste lógico que escolhe qual instrução será executado.
	PREPARAÇÃO	Representa uma ação de preparação para o processamento.
	CONECTOR	Utilizado para interligar partes de um fluxograma ou para desviar o fluxo

Fluxograma/Diagrama de blocos Sequencial

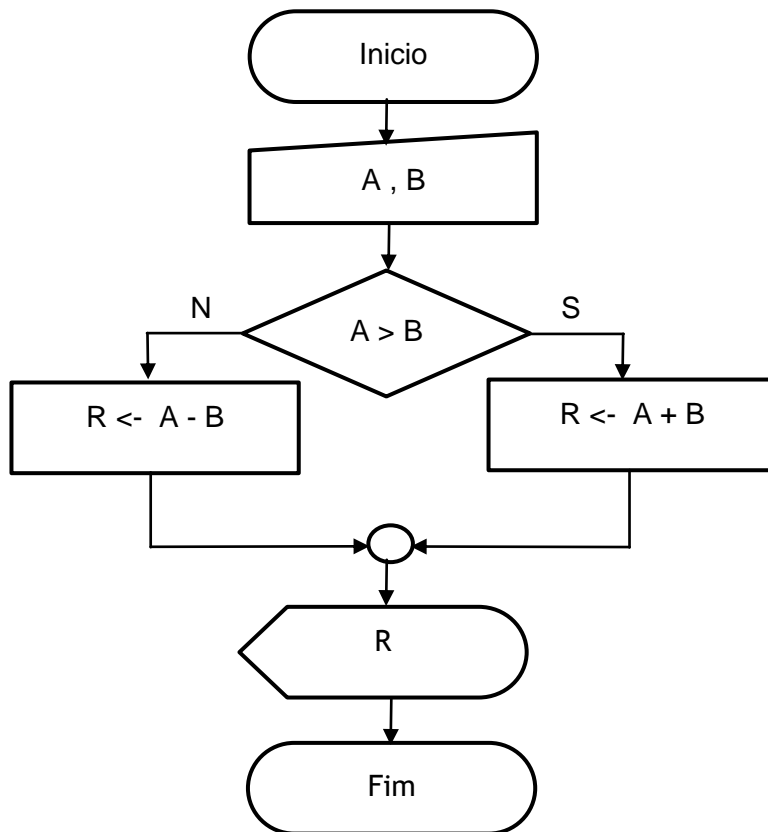


Fluxograma/Diagrama de Blocos - decisão simples.



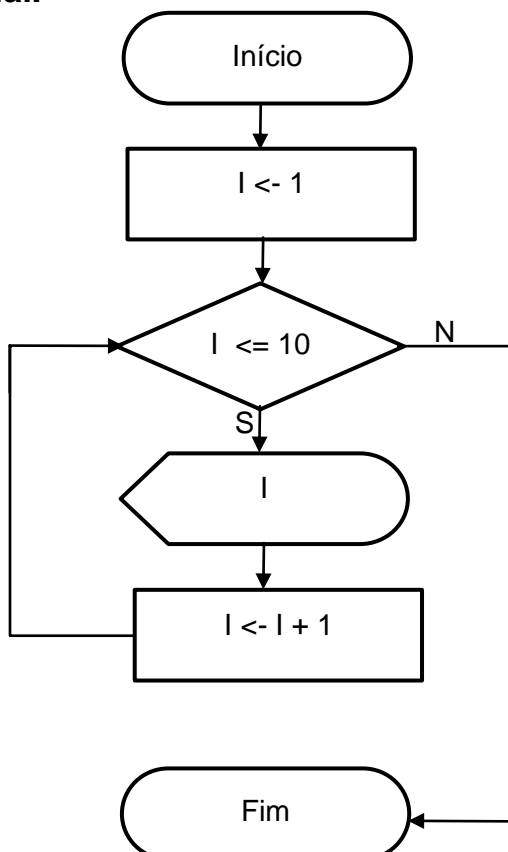
Nesse fluxograma, serão lidas duas variáveis **A** e **B**, o próximo passo é feito uma pergunta, SE **A** maior que **B**, se a resposta for verdadeira, então o fluxo será desviado para o lado da letra "**S**" e será feito a atribuição para a variável "**R**" que recebe a soma de **A + B**, logo após mostra a letra "**R**" e finaliza o fluxograma, caso contrário **A** menor que **B** o fluxo seguirá para o lado da letra "**N**" e será finalizado o fluxograma.

Fluxograma/Diagrama de Blocos - decisão composta.



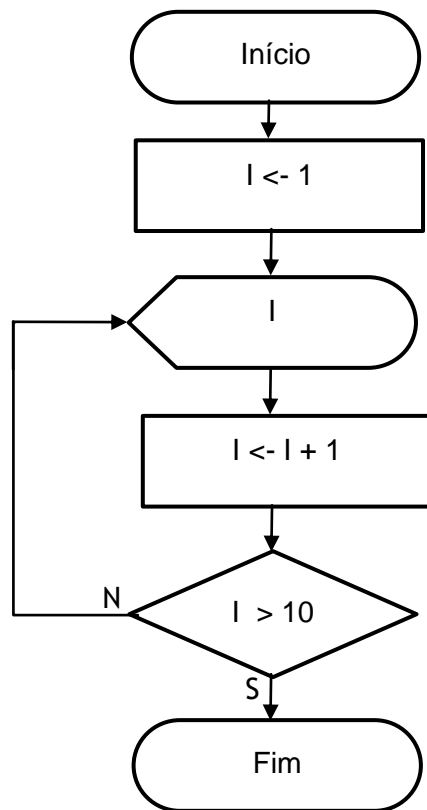
Nesse fluxograma, serão lidas duas variáveis **A** e **B**, o próximo passo é feito uma pergunta, SE **A** maior que **B**, se a resposta for verdadeira, então o fluxo será desviado para o lado da letra "**S**" e será feito a atribuição para a variável "**R**" que recebe a soma de $A + B$, logo após mostra a letra "**R**" e finaliza o fluxograma, caso contrário se **A** menor que **B** o fluxo seguirá para o lado da letra "**N**" e será feito a atribuição para a variável "**R**" que recebe a soma de $A - B$, logo após mostra a letra "**R**" e finaliza o fluxograma.

Fluxograma/Diagrama de Blocos - laço de repetição com condição no inicial.



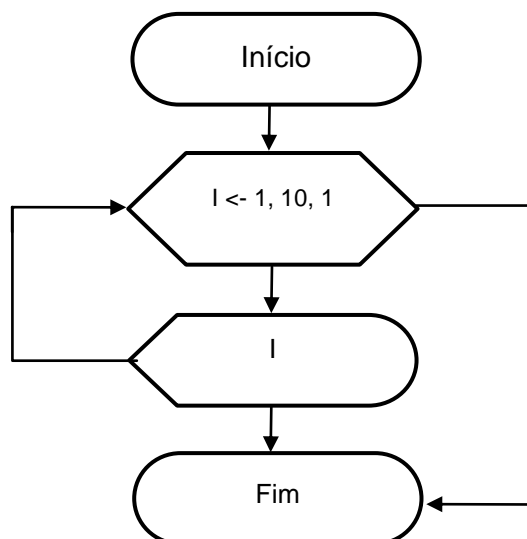
Esse fluxograma representa a repetição de um bloco de código que precisamos repetir por 10 vezes, para isso a variável "**I**" recebe o valor inicial de 1, em seguida é feita uma condição para verificar se o valor de **I** é menor ou igual a 10, caso verdadeiro o fluxo segue mostrando o valor da variável **I** e acrescenta mais 1 ao seu valor anterior que era 1 e agora vale 2 e volta para a mesma condição comparando o valor de **I** menor igual a 10, esse processo vai acontecer 10 vezes até a variável **I** for maior que 10.

Fluxograma/Diagrama de Blocos - laço de repetição com condição no final.



Esse fluxograma representa a repetição de um bloco de código que precisamos repetir por 10 vezes de acordo com a condição especificada, para isso a variável "I" recebe o valor inicial de 1, em seguida mostra o valor da variável I que nesse primeiro momento é 1 em seguida acrescenta mais 1 ao seu valor anterior que era 1 e agora vale 2, no próximo passo é feita uma condição para verificar se o valor de I é maior que 10, caso falso o fluxo segue mostrando o valor da variável I e acrescenta mais 1 ao seu valor anterior, esse processo ocorre 10 vezes até a variável I for maior que 10, quando a condição for verdadeira então é finalizado o algoritmo.

Fluxograma/Diagrama de Blocos - laço interativo.

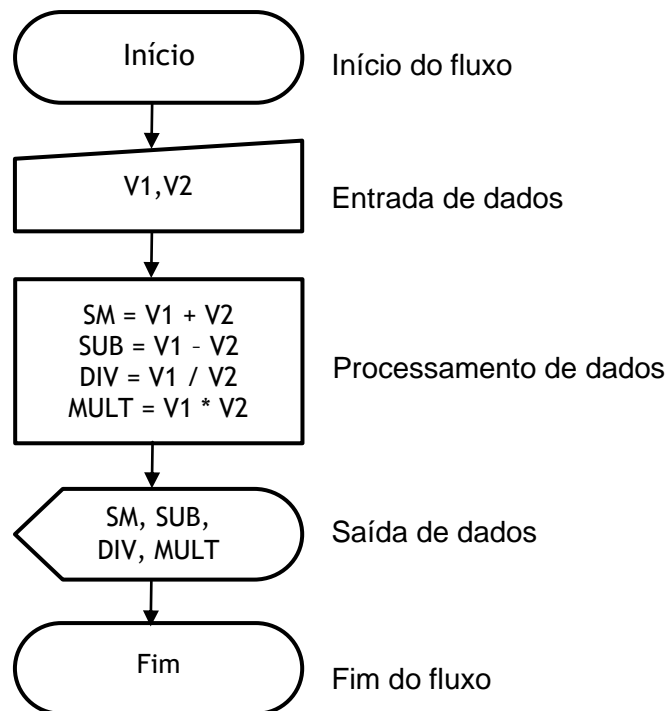


Nosso último exemplo de fluxograma funciona da seguinte forma.:

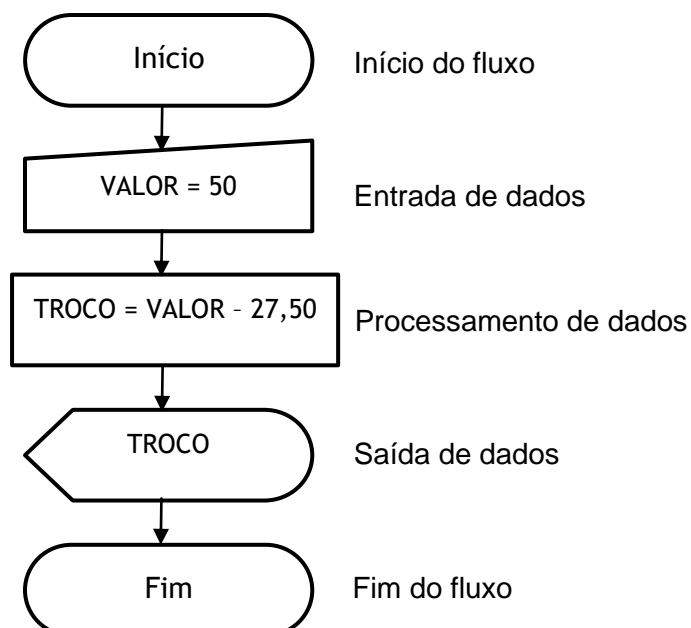
Na figura logo após a figura do início temos a seguinte marcação "I <- 1, 10, 1" onde a variável I representa o contador que recebe o valor inicial de 1 e vai até o valor de 10 incrementando a variável I de 1 em 1 e para cada interação é exibido o valor de I com seu valor atual, quando o valor de I for 10 as repetições são encerradas e o fluxograma vai para o fim.

Vamos resolver dois exercícios usando fluxograma da lista anterior de exercícios.

1 – Leia dois valores, faça as quatro operações básicas da matemática (adição, subtração, divisão e multiplicação) com os dois valores e mostre os resultados como saída.



2 – Fui ao mercado com R\$ 50,00, fiz uma compra de R\$ 27,50, mostre como saída o troco que sobrou das compras.



Agora vocês devem fazer os próximos exercícios usando fluxogramas/Diagrama de blocos.

1 - Escreva um algoritmo usando fluxograma para ler as dimensões de um retângulo (base e altura), calcular e escrever a área do retângulo.

2 - Faça um algoritmo usando fluxograma que leia a idade de uma pessoa expressa em anos, meses e dias e escreva a idade dessa pessoa expressa apenas em dias. Considerar ano com 365 dias e mês com 30 dias.

3 – Faça um algoritmo usando fluxograma para ler o salário mensal atual de um funcionário e o percentual de reajuste. Calcular e escrever o valor do novo salário.

4 - Faça um algoritmo que mostre o novo preço de um produto sabendo-se que este terá um desconto de 15%, mostrando também a classificação do produto segundo as informações abaixo.

Novo preço: de 500 para cima = CARO

Abaixo de 500 = BARATO

5 – Ler um valor N e mostrar todos os números inteiros de 1 até N. Usando o laço de repetição.

Resolução no CADERNO 02 - Resolução Algoritmo Fluxograma.

Pseudocódigo ou português estruturado



Utiliza linguagem estruturada e se assemelha na forma de um programa escrito em linguagem de programação, também conhecido como português estruturado ou Portugal.

Pseudocódigo ou Portugol.

Visando a criação de um algoritmo com uma linguagem flexível e intermediária entre a linguagem natural e a linguagem de programação, utilizamos o pseudocódigo.

O Pseudocódigo ou Portugol é uma linguagem que une o formalismo das linguagens de programação à facilidade de compreensão da linguagem natural, utiliza linguagem estruturada e se assemelha na forma de um programa escrito em uma linguagem de programação como a linguagem C e Pascal, linguagens modernas como C#, Java, Python, Javascript entre outras, também usam esse conceito de programação estruturada no momento que usamos o fluxo de entrada, processo e saída de dados, condicionais, estrutura de repetição e estrutura de dados como vetores e matrizes.

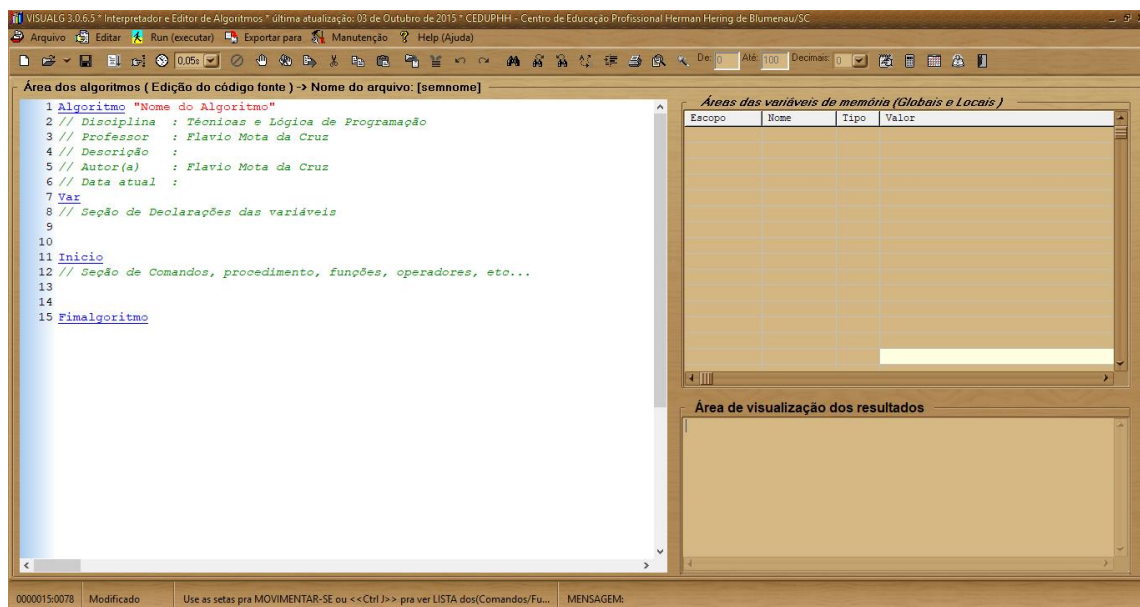
Usamos essa linguagem para construir algoritmos a serem interpretados por computadores e que seja de fácil entendimento para quem está desenvolvendo seus primeiros algoritmos, usando softwares que tenha uma sintaxe bem próximo do nosso idioma, desta forma a compreensão inicial dos programas escritos em português estruturado facilita o aprendizado nos primeiros passos na programação de computadores.

Nessa apostila usaremos o interpretador de pseudocódigo VisualG 3.0, software desenvolvido por dois brasileiros: Cláudio Morgado de Souza e Antônio Carlos Nicolodi. Nossos primeiros códigos serão nessa linguagem de pseudocódigo bastante utilizada no meio acadêmico.

O que é o interpretador de pseudocódigo VisualG 3.0?

O VisualG é um programa que permite criar, editar, interpretar e que também executa os algoritmos em português estruturado (portugol) como se fosse um “programa” normal de computador. É um programa de livre uso e distribuição GRÁTIS, e DOMÍNIO PÚBLICO, usado para o ensino de lógica de programação em várias escolas e universidades no Brasil e no exterior, disponível no seguinte endereço: <http://visualg3.com.br>, ao baixar o VisualG, faça a descompactação do arquivo no disco C do seu computador, será criado uma pasta chamada VISUALG3 e dentro da pasta será criado vários arquivos e pastas, para executar o programa clique duas vezes no ícone com o nome visualg30, o software não precisa ser instalado, o mesmo será executado direto da pasta.

Essa é a tela do software VisualG 3.0



Antes de começar a usar o software VisualG 3.0 temos que compreender os conceitos de variáveis, constantes, tipos de dados e operadores, conhecer os conceitos de comandos de atribuição, entrada e saída de dados.

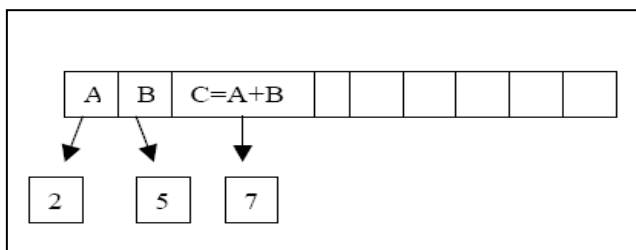
Variáveis

O primeiro passo para que um programa seja executado em um computador é o carregamento desse programa para a memória. A memória é utilizada para armazenar tanto as instruções dos programas quanto os dados utilizados por eles. As variáveis são endereços de memória destinados a armazenar informações temporariamente, pois ao término da execução dos programas todos os dados utilizados serão perdidos caso não sejam salvos no computador. Um dado é classificado como variável quando tem a possibilidade de ser alterado no decorrer da execução do programa.

Variáveis de **ENTRADA**: armazenam informações fornecidas por um meio externo, normalmente usuários ou outro sistema.

Variáveis de **SAÍDA**: armazenam dados processados como resultados.

Exemplo simples de como funciona uma variável na memória de um computador:



Nesse exemplo cada quadrado representa um espaço na memória do computador e cada espaço recebe apenas uma informação por vez, repare que

as variáveis são representadas por letras e cada letra possui um valor $A = 2$, $B = 5$ e C é igual a soma de $A + B$ ($C = A + B$) ($C = 7$), isso quer dizer que uma variável pode receber além de valores também pode receber uma expressão matemática, seja ela básica ou mais complexa.

Por que declarar variáveis e como nomeá-las?

Quando criamos algoritmos ou programas de computador, os criamos para resolver problemas observados, por exemplo, para fazer um ajuste de um salário em 10% para um funcionário, temos que usar dados que represente um valor para o salário do funcionário e uma taxa para representar o aumento que esse salário deverá ter, a partir dessas informações podemos declarar variáveis para armazenar essas informações na memória do computador, geralmente sugerimos que o nome das variáveis deva ser algo que lembre a informação que vamos guardar, por exemplo:

Para representar o salário do funcionário podemos declarar da seguinte forma: SalFunc ou SalarioFunc ou ainda Salario_Funcionario, além de outras combinações.

Para declarar a taxa de aumento podemos fazer da seguinte forma: Taxa, Aumento, Reajuste, Sal_Aumento entre outras combinações, além dessas sugestões temos que seguir algumas regras conforme a tabela abaixo.

Regras para nomear Variáveis

Regra	Exemplo
Inicie sempre por um caractere alfabético, nunca por um número.	Nome (correto) - 1nome (errado)
Não utilize caracteres especiais como “ , () / * ; + .	Nome(M); N*B

Não coloque espaços em branco ou hífen entre nomes.	Salario-bruto
Utilize, se necessário, underline.	Salario_bruto
Crie suas variáveis com nomes sugestivos.	Se vai guardar salário de funcionários, dê à variável o nome salario.

Tipos de dados

Quando declaramos as variáveis devemos atribuir para cada uma delas um tipo específico de dados, esses dados definem a forma que o computador vai tratar a informação recebida ou processada.

Por exemplo: um cálculo deve ser feito por dados do tipo numérico, seja ele real ou do tipo inteiro, caso o sistema precise armazenar um nome de uma pessoa esse valor deve ser do tipo caractere, e se precisamos guardar uma situação do tipo verdadeiro ou falso podemos usar uma variável do tipo lógica.

Inteiro: define variáveis numéricas do tipo inteiro, ou seja, sem casas decimais positivas ou negativas.

Ex. idade, número de filhos, números de gols.

Real: define variáveis numéricas do tipo real, ou seja, com casas decimais.

Ex. salário, peso, temperatura.

Caractere: define variáveis do tipo texto, ou seja, cadeia de caracteres.

Ex. nome, endereço, frase.

Lógico: define variáveis do tipo lógica, ou seja, com valor VERDADEIRO ou FALSO.

Constantes

Constantes são dados que durante a execução do programa, permanecem com os seus valores inalterados. Tem-se como definição de constante tudo aquilo que é fixo ou estável. Uma variável pode ser alterada ao longo de seu algoritmo. Mas, às vezes, precisamos armazenar valores que não se alteram. Para isso existem as constantes.

As constantes são criadas obedecendo às mesmas regras de nomenclatura já vistas em variáveis. Diferem apenas no fato de armazenar um valor constante, ou seja, que não se modifica durante a execução de um programa.

Exemplo de algoritmo (pseudocódigo), declaração de variáveis e constantes usando VISUALG 3.0

```

1 Algoritmo "Nome do Algoritmo"
2 // Disciplina : Técnicas e Lógica de Programação
3 // Professor : Flavio Mota da Cruz
4 // Descrição :
5 // Autor(a) : Flavio Mota da Cruz
6 // Data atual :
7
8 const
9
10 Var
11 ValorDolar, Nota1, Nota2: real // Exemplos de variaveis
12 Fgts: real
13 Inicio
14
15 Fgts <- 8 // Exemplo de constante
16
17 Fimalgoritmo

```

Operador de atribuição

No pseudocódigo quando queremos atribuir valor a uma variável ou a uma constante usamos o operador de atribuição “<-” sinal de “< menor que” é sinal

de “ - menos“, juntos formam o operador de atribuição, em linguagens de programação como C# ou Java esse operador geralmente é o sinal de igual “=”.

Exemplos de atribuição a variáveis ou constantes

Fgts <- 8 // Fgts contém o valor de 8

Dolar <- 4.00 //Dolar contém valor de 4.00

Idade <- 20 // Idade contém o valor de 20

Nome <- “Flavio Mota” // Nome contém o valor de “Flavio Mota”

Quando atribuímos o valor para a variável isso quer dizer que agora elas contêm um valor estabelecido provisoriamente, pois esses dados podem mudar a qualquer momento dependendo do processo que o sistema deverá realizar, por isso são chamadas de variáveis, pois seus valores podem variar durante a execução dos algoritmos ou programas. Diferente das constantes onde seus valores não mudam durante a execução do algoritmo ou programa.

Linearização de Expressões

Para a construção de algoritmos que realizam cálculo matemáticos, todas as expressões aritméticas devem ser linearizadas, ou seja, colocadas em linhas, devendo também ser feito o mapeamento dos operadores da aritmética tradicional para os do Português Estruturado.

$\left\{ \left[\frac{2}{3} - (5 - 3) \right] + 1 \right\} . 5$	$((2/3 - (5 - 3)) + 1) * 5$
Tradicional	Computacional

Operadores Aritméticos

OPERADORES ARITMÉTICOS	PORTUGUÊS ESTRUTURADO
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Divisão Inteira	\
Exponenciação	^ ou Exp (<base>, <expoente>)
Módulo (resto da divisão)	%

Operadores Relacionais

Os operadores relacionais realizam a comparação entre dois operandos ou duas expressões e resultam em valores lógicos (VERDADEIRO ou FALSO).

OPERADORES RELACIONAIS	PORTUGUÊS ESTRUTURADO
Maior	>
Menor	<
Maior ou igual	>=
Menor ou igual	<=
Igual	=
Diferente	<>

Exemplo:

$(7 + 2) > 10$ = Falso

$(5 <> 3)$ = Verdadeiro

Operadores Lógicos

Os operadores lógicos atuam sobre expressões e resultam em valores lógicos, VERDADEIRO ou FALSO.

OPERADORES LÓGICOS	PORTUGUÊS ESTRUTURADO	SIGNIFICADO
Multiplicação lógica	E	Resulta VERDADEIRO se ambas as partes forem verdadeiras.
Adição lógica	Ou	Resulta VERDADEIRO se uma das partes é verdadeira.
Negação	Nao	Nega uma afirmação, invertendo o seu valor lógico: se for VERDADEIRO torna-se FALSO , se for FALSO torna-se VERDADEIRO .

A tabela abaixo – chamada tabela-verdade – mostra os resultados das aplicações dos operadores lógicos conforme os valores dos operadores envolvidos.

A	B	A E B	A OU B	NÃO A	NÃO B
VERDADEIRO	VERDADEIRO	VERDADEIRO	VERDADEIRO	FALSO	FALSO
VERDADEIRO	FALSO	FALSO	VERDADEIRO	FALSO	VERDADEIRO
FALSO	VERDADEIRO	FALSO	VERDADEIRO	VERDADEIRO	FALSO
FALSO	FALSO	FALSO	FALSO	VERDADEIRO	VERDADEIRO

De acordo com a necessidade, as expressões podem ser unidas pelos operadores lógicos.

Exemplo:

$(2+5>4)$ e $(3<>3)$ resulta FALSO, pois VERDADEIRO e FALSO resulta FALSO.

Os parênteses indicam quais sub-expressões, dentro de uma expressão, serão executadas primeiro. A princípio, a execução é da esquerda para direita, mas além dos parênteses, existem prioridades entre os operadores envolvidos na expressão. Tais prioridades são mostradas nas tabelas seguintes.

OPERADOR ARITMÉTICO	PRIORIDADE
Exponenciação	3 (maior)
Multiplicação	2
Divisão	2
Adição	1
Subtração	1 (menor)

Exemplo:

$(2 + 2)/2$ resulta 2 e $2 + 2/2$ resulta 3

OPERADOR LÓGICO	PRIORIDADE
e	3
ou	2
nao	1

Exemplo:

$(2 > 3)$ ou $(3 < 2)$ e $(2 < 3)$ // resultado seria Falso

$(2 > 3)$ e $(3 < 2)$ ou $(2 < 3)$ // resultado seria Verdadeiro

Entre as categorias de operadores também há prioridades, conforme mostrado na tabela abaixo.

OPERADOR	PRIORIDADE
Operadores aritméticos	3
Operadores relacionais	2
Operadores lógicos	1

O software VisuAlg 3.0 não possui relacionamento de categorias.

$2 * 5 > 3$ ou $5 + 1 < 2$ e $2 < 7 - 2$ // resulta em erro.

$(2 * 5 > 3)$ ou $(5 + 1 < 2)$ e $(2 < 7 - 2)$ // modo correto de fazer.

Comandos de ENTRADA e SAÍDA

Entrada de Dados

Frequentemente, na construção de algoritmos, precisamos solicitar que usuários informem, por meio do teclado, alguns valores a serem utilizados durante a execução. Por exemplo: se fizermos um algoritmo para calcular a média das notas de um aluno, precisaremos solicitar quais foram as notas, em seguida calcularmos a média. Esses valores informados devem ser armazenados em variáveis para que sejam utilizados quando necessário.

O comando de entrada de dados será responsável pela leitura e armazenamento desses dados na variável que indicarmos. A sintaxe do comando de entrada de dados em Portugol, é exibida abaixo e no nosso caso usando a linguagem de pseudocódigo do VISUALG 3.0.

LEIA(Variavel) ou Leia(Variavel), vamos usar um exemplo no VisualG 3.0

```

1 Algoritmo "Nome do Algoritmo"
2 // Disciplina : Técnicas e Lógica de Programação
3 // Professor : Flavio Mota da Cruz
4 // Descrição :
5 // Autor(a) : Flavio Mota da Cruz
6 // Data atual :
7
8 Var
9
10 Nota1, Nota2: real // Exemplos de variaveis
11
12 Inicio
13
14
15 Leia(Nota1) // Comando para ler a variável Nota1
16 Leia(Nota2) // Comando para ler a variável Nota2
17
18 Fimalgoritmo

```

Saída de Dados

Por meio da utilização do comando de saída de dados, conseguimos exibir mensagens ou valores para o usuário de nossos programas. É através desse comando que nosso algoritmo consegue se comunicar com o usuário para solicitar a entrada de dados ou para fornecer saídas de dados.

O comando de saída de dados exibe no monitor valores de constantes, variáveis ou expressões.

Exemplo:

Escreva(Variavel, Expressões , “Texto”) ou **Escreval**(Variavel, Expressões , “Texto”)

No VisualG 3.0 usamos o comando ESCREVA conforme o exemplo abaixo.

```

1 Algoritmo "Nome do Algoritmo"
2 // Disciplina : Técnicas e Lógica de Programação
3 // Professor : Flavio Mota da Cruz
4 // Descrição :
5 // Autor(a) : Flavio Mota da Cruz
6 // Data atual :
7
8 Var
9
10 Nota1, Nota2:real // Exemplos de variaveis
11 Media:real
12 Inicio
13
14
15 Leia(Nota1) // Comando para ler a variável Nota1
16 Leia(Nota2) // Comando para ler a variável Nota2
17
18 Media <- (Nota1 + Nota2) / 2 // Processamento de dados
19
20 Escreval(Media) // Exemplo de saída de dados
21 Escreva("A média das notas é = ", Media) // Exemplo de saída de dados
22
23
24 Fimalgoritmo

```

Nesse exemplo usamos dois comandos de saídas de dados, o primeiro comando “**Escreval**”, escreva + a letra L, além de mostrar na tela o resultado da variável Media, esse comando quebra a linha fazendo com que o próximo comando seja feito na linha de baixo. O segundo “**Escreva**” mostra a frase “A média das notas é = ” + a variável Media, chamamos de concatenação quando juntamos um texto entre aspas mais uma variável.

Pseudocódigo VISUALG



A linguagem que o VisualG interpreta é bem simples: é uma versão portuguesa dos pseudocódigos largamente utilizados nos livros de introdução à programação, conhecida como "Portugol". Inicialmente, pensava-se em criar uma sintaxe muito simples e "liberal", para que o usuário se preocupasse apenas com a lógica da resolução dos problemas e não com as palavras-chave, pontos e vírgulas, etc. No entanto, chegou-se à conclusão de que alguma formalidade seria não só necessária como útil, para criar um sentido de disciplina na elaboração do "código-fonte".

A linguagem do VisualG permite apenas um comando por linha: desse modo, não há necessidade de tokens separadores de estruturas, como o ponto e vírgula em Pascal. Também não existe o conceito de blocos de comandos (que correspondem ao begin e end do Pascal e ao { e } do C).

Importante: para facilitar a digitação e evitar confusões, todas as palavras-chave do VisualG foram implementadas sem acentos, cedilha, etc. Portanto, o tipo de dados lógico é definido como lógico, o comando se..então..senão é definido como se..entao..senao, e assim por diante. O VisualG também não distingue maiúsculas e minúsculas no reconhecimento de palavras-chave e nomes de variáveis.

Estrutura básica de um algoritmo em VisualG

```

1 Algoritmo "Nome_Algoritmo"
2 // Professor : Flávio Mota
3 Var
4
5     // Seção de Declarações das variáveis
6
7 Inicio    // Inicio do Algoritmo
8
9 leia()    // Entrada de dados
10 escreva() // Saída de dados
11 escreval() // Saída de dados com quebra de linha
12
13 Fimalgoritmo // Fim do algoritmo

```

A primeira linha é composta pela palavra-chave `algoritmo` seguida do seu nome delimitado por aspas duplas. Este nome será usado como título nas janelas de leitura de dados. A seção que se segue é a de declaração de variáveis, que termina com a linha que contém a palavra-chave `início`. Deste ponto em diante está a seção de comandos, que continua até a linha em que se encontre a palavra-chave `fimalgoritmo`. Esta última linha marca o final do pseudocódigo: todo texto existente a partir dela é ignorado pelo interpretador.

O VisualG permite a inclusão de comentários: qualquer texto precedido de `"//"` é ignorado, até se atingir o final da sua linha. Por este motivo, os comentários não se estendem por mais de uma linha: quando se deseja escrever comentários mais longos, que ocupem várias linhas, cada uma delas deverá começar por `"//"`.

Tipos de Dados

O VisualG prevê quatro tipos de dados: inteiro, real, cadeia de caracteres e lógico (ou booleano). As palavras-chave que os definem são as seguintes (observe que elas não têm acentuação):

Inteiro: define variáveis numéricas do tipo inteiro, ou seja, sem casas decimais positivos ou negativos.

Ex. idade, número de filhos, números de gols.

Real: define variáveis numéricas do tipo real, ou seja, com casas decimais.

Ex. salário, peso, temperatura.

Caractere: define variáveis do tipo texto, ou seja, cadeia de caracteres.

Ex. nome, endereço, frase.

Lógico: define variáveis do tipo lógica, ou seja, com valor VERDADEIRO ou

FALSO.

Vamos abordar os primeiros exemplos de algoritmos usando a ferramenta VisualG com seus conceitos iniciais de **ENTRADA, PROCESSO E SAÍDA** de dados.

Exemplo 01

<pre> 1 Algoritmo "Ex01" 2 3 4 Var 5 valor:inteiro // recebe numeros inteiros 6 x:real // recebe numeros fracionados 7 nome:caracter // recebe textos 8 status:logico // recebe verdadeiro ou falso 9 idade:inteiro // recebe numeros inteiros 10 11 Inicio 12 13 // entrada de dados 14 15 valor <- 10 // atribuição de dados 16 x <- 2.5 // atribuição de dados 17 nome <- "Flavio Mota" // atribuição de dados 18 status <- falso // atribuição de dados 19 idade <- 40 // atribuição de dados 20 21 // processos 22 Esse algoritmo não tem processo, somente entrada/atribuição e saída 23 de dados. 24 // saidas 25 26 escreva(nome , " voce tem ", idade, " anos ") 27 28 Fimalgoritmo </pre>	<p>Declaração de variáveis e os tipos de dados que elas receberão.</p> <p>Atribuição de valores para as variáveis de acordo com seus tipos definidos na declaração</p> <p>Saída de dados com o comando ESCREVA()</p>
--	--

Ao executar o algoritmo teremos essa saída no console

```

C:\> Console simulando o modo texto do MS-DOS

Flavio Mota voce tem 40 anos
>>> Fim da execução do programa !

```


Exemplo 02

<pre> 1 <u>Algoritmo</u> "Ex02" 2 3 // Professor : Flavio Mota 4 <u>Var</u> 5 6 valor:<u>inteiro</u> // receber numeros inteiros 7 8 <u>Inicio</u> 9 10 // entrada de dados 11 escreva("Digite um valor: ") 12 leia(valor) 13 14 // processos 15 Esse algoritmo não tem processo, somente entrada/atribuição e saída 16 de dados. 17 // saidas 18 escreva("o valor da variavel ", valor) 19 <u>Fimalgoritmo</u> </pre>	<div style="border-left: 1px solid red; border-right: 1px solid red; height: 100px; margin: 0 auto; width: 50px;"></div> <div style="border-left: 1px solid red; border-right: 1px solid red; height: 100px; margin: 0 auto; width: 50px;"></div> <div style="border-left: 1px solid red; border-right: 1px solid red; height: 100px; margin: 0 auto; width: 50px;"></div>	<p>Declaração da variável valor com seu tipo de dados inteiro.</p> <p>Entrada de dados usando o comando LEIA()</p> <p>Saída de dados usando o comando ESCREVA()</p>
---	--	---

Ao executar o algoritmo teremos essa saída no console

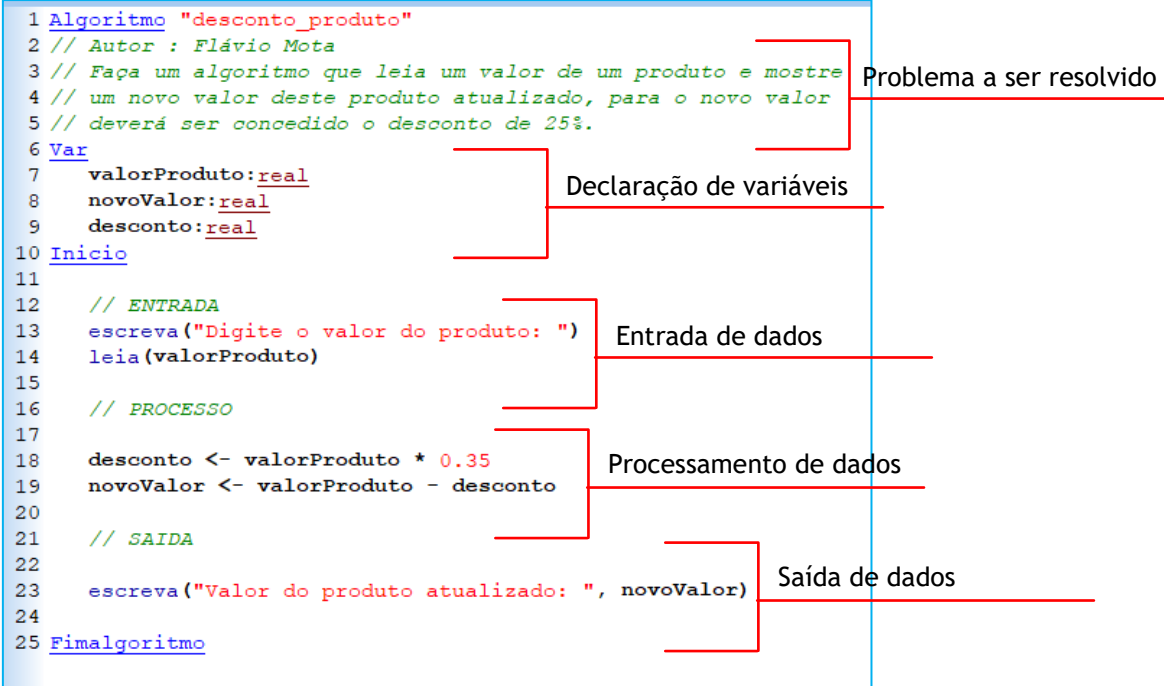
```

C:\> Console simulando o modo texto do MS-DOS

Digite um valor: 35
o valor da variavel 35
>>> Fim da execução do programa !

```

Exemplo 03



Ao executar o algoritmo teremos essa saída no console

```

C:\> Console simulando o modo texto do MS-DOS

Digite o valor do produto: 100
Valor do produto atualizado: 65
>>> Fim da execução do programa !
  
```

Para o melhor entendimento dos nossos problemas (Algoritmos) criamos uma tabela que vai nos ajudar a desenvolver nossos algoritmos. Todo algoritmo é um problema a ser resolvido, em busca da solução, temos que seguir alguns passos que devem ser entendidos para chegarmos ao resultado esperado.

1 - Um comerciante deseja saber qual é o lucro percentual que ele está tendo com a venda de mercadorias. Calcule o lucro percentual de uma mercadoria ao serem fornecidos o preço de compra e o preço de venda da mesma.	
O problema	Identificar o lucro obtido com a venda de uma mercadoria.
	Observações complementares
	Nenhuma.
Solução esperada	Obter o percentual de lucro.
Dados de Entrada	Mercadoria; Valor de compra; Valor de venda.
	Detalhamento dos Dados de Entrada
	✓ Lucro Percentual É obtido através da subtração do valor de compra do valor de venda e, posteriormente multiplicando esse valor por cem e dividindo-o pelo valor de compra. Uma regra de três básica.
Dados de Saída	Percentual de Lucro
Etapas encontradas	<ul style="list-style-type: none"> ✓ Identificar a mercadoria; ✓ Solicitar o valor de compra; ✓ Solicitar o valor de venda; ✓ Cálculo do lucro; ✓ Informar o Lucro percentual.
Descrição Narrativa da solução encontrada	<ol style="list-style-type: none"> 1. Solicitar a mercadoria; 2. Solicitar o valor de compra; 3. Solicitar o valor de venda; 4. Calcular o Lucro; 5. Informar ao comerciante o resultado.

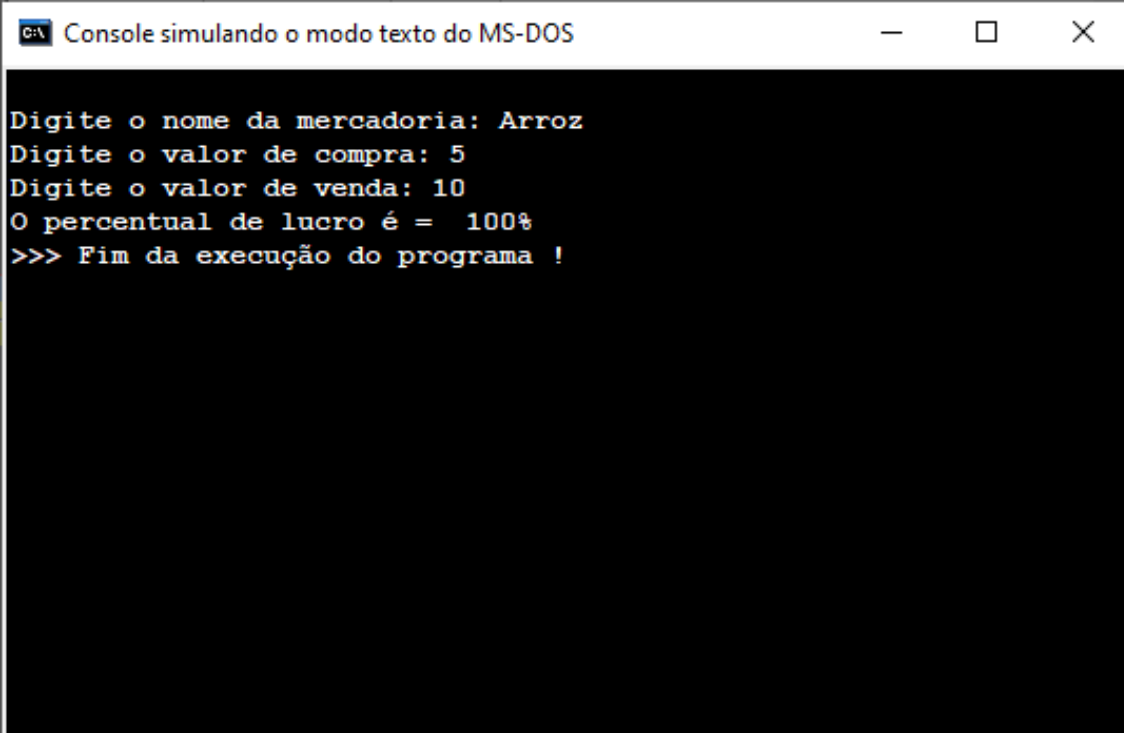
Resolução em VisualG

```

1 Algoritmo "EPS_ex01"
2 // Autor: Flávio Mota
3
4 // Um comerciante deseja saber qual é o lucro percentual
5 // que ele está tendo com a venda de mercadorias. Calcule
6 // o lucro percentual de uma mercadoria ao serem fornecidos
7 // o preço de compra e o preço de venda da mesma.
8 Var
9     mercadoria:caracter
10    valorCompra:real
11    valorVenda:real
12    percentualLucro:real
13 Inicio
14
15     // ENTRADAS
16     Escreva("Digite o nome da mercadoria: ")
17     leia(mercadoria)
18     Escreva("Digite o valor de compra: ")
19     leia(valorCompra)
20     Escreva("Digite o valor de venda: ")
21     leia(valorVenda)
22     // PROCESSO
23     percentualLucro <- (valorVenda - valorCompra) * 100 / valorCompra
24     // SAÍDA
25     Escreva("O percentual de lucro é = ",percentualLucro)
26
27
28 Fimalgoritmo

```

Saída no console



The screenshot shows a console window titled "C:\> Console simulando o modo texto do MS-DOS". The window has standard Windows window controls (minimize, maximize, close). The text inside the console is as follows:

```

Digite o nome da mercadoria: Arroz
Digite o valor de compra: 5
Digite o valor de venda: 10
O percentual de lucro é = 100%
>>> Fim da execução do programa !

```

<p>2 - Sobre o salário bruto de um funcionário são descontados 8% de INSS, 10% de IR (Imposto de Renda) e sobre o restante 0,5% (meio por cento) referente à filiação sindical. Ao ser fornecido o valor do salário bruto do funcionário, calcule:</p> <ul style="list-style-type: none"> • Os descontos de INSS, IR e Filiação Sindical; • O Total dos Descontos; • O Salário Líquido. 	
O problema	Calcular os descontos de INSS, IR e Filiação Sindical; O Total dos Descontos e o Salário Líquido.
	Observações complementares
	Descontar a taxa referente à filiação sindical apenas sobre o restante salário, ou seja, sobre o que sobrar após o desconto do INSS e IR.
Solução esperada	Obter os valores de descontos de INSS, IR e Filiação Sindical, o total dos descontos e o salário líquido.
Dados de Entrada	Salário bruto;
	Detalhamento dos Dados de Entrada
	Nenhum.
Dados de Saída	O total descontado de INSS, IR, Filiação Sindical; total de descontos e salário líquido.
Etapas encontradas	<ul style="list-style-type: none"> ✓ Solicitar o valor do salário bruto; ✓ Calcular o valor do INSS; ✓ Calcular o valor do IR; ✓ Calcular o valor da Filiação Sindical; ✓ Calcular o total de impostos; ✓ Obter o salário líquido; ✓ Informar o valor encontrado de cada imposto, o total de impostos e do salário líquido.
Descrição Narrativa da solução encontrada	<ol style="list-style-type: none"> 1. Solicitar o valor do salário bruto; 2. Calcular o desconto de INSS com a fórmula: $(\text{Salário} * 8) / 100$; 3. Calcular o desconto de IR com a fórmula: $(\text{Salário} * 10) / 100$; 4. Calcular a taxa Filiação Sindical com a fórmula: $((\text{Salário} - (\text{INSS} + \text{IR})) * 0.5) / 100$; 5. Calcular o total do INSS + IR + FILIAÇÃO SINDICAL; 6. Descontar o total de impostos do Salário Bruto ($\text{Salário} - (\text{INSS} + \text{IR} + \text{FILIAÇÃO SINDICAL})$); 7. Informar o valor de INSS; 8. Informar o valor do IR; 9. Informar o total de Impostos; 10. Informar o Salário Líquido;

Resolução em VisualG

```

1 Algoritmo "EPS_ex02"
2 // Autor: Flávio Mota
3
4 // Sobre o salário bruto de um funcionário são descontados 8% de INSS,
5 // 10% de IR (Imposto de Renda) e sobre o restante 0,5% (meio por cento)
6 // referente à filiação sindical. Ao ser fornecido o valor do salário bruto
7 // do funcionário, calcule:
8 // " Os descontos de INSS, IR e Filiação Sindical;
9 // " O Total dos , Descontos
10 // " O Salário Líquido
11 Var
12     salarioBruto, INSS, IR, filiacaoSindical: real
13     totalImpostos, salarioLiquido: real
14 Inicio
15     // ENTRADAS
16     Escreva("Digite o salário bruto: ")
17     leia(salarioBruto)
18     // PROCESSO
19     INSS <- (salarioBruto * 8) / 100
20     IR <- (salarioBruto * 10) / 100
21     filiacaoSindical <- (salarioBruto - (INSS+IR)) * 0.05 / 100
22     totalImpostos <- INSS + IR + filiacaoSindical
23     salarioLiquido <- salarioBruto - totalImpostos
24     // SAÍDA
25     Escreval("INSS = ", INSS)
26     Escreval("IR = ", IR)
27     Escreval("Total de imposto = ", totalImpostos)
28     Escreval("Salário Líquido = ", salarioLiquido)
29
30 Fimalgoritmo

```

Saída no console

```

C:\ Console simulando o modo texto do MS-DOS

Digite o salário bruto: 1000
INSS = 80
IR = 100
Total de imposto = 221
Salário Líquido = 779

>>> Fim da execução do programa !

```

Exercícios no CADERNO 01: Algoritmos nível 01 – ENTRADA, PROCESSO E SAÍDA DE DADOS

Comandos de desvio condicional (SE, SENAO, SENAO SE)

Muitas vezes precisamos tomar decisões que podem interferir diretamente no andamento do algoritmo. A representação dessas decisões em nossos programas é feita através do uso de estruturas de seleção, ou estruturas de decisão. A estrutura de seleção que utilizaremos em Portugol (VisualG) será a estrutura **SE...ENTAO... SENAO**. A sintaxe dessa estrutura é exibida abaixo:

Sintaxe:

```
SE <expressão lógica> ENTAO
    <bloco de instruções verdade>
FIMSE
```

Exemplo VisualG

```
Se media > 6 entao
    Escreva("Aprovado")
fimse
```

Sintaxe:

```
SE <expressão lógica> ENTAO
    <bloco de instruções verdade>
SENAO
    <bloco de instruções falso>
FIMSE
```

Exemplo VisualG

```
Se media > 6 entao
    Escreva("Aprovado")
Senao
    Escreva("Reprovado")
fimse
```

Como funciona?

A palavra reservada **SE** indica o início da estrutura de seleção. Após essa palavra, vem a condição que definirá o bloco a ser executado. Qualquer expressão lógica poderá ser utilizada como condição, pois deverá retornar verdadeiro ou falso. Caso a expressão da condição seja verdadeira, o bloco de instruções **ENTAO** será executado. Caso contrário, o bloco **SENAO** o será. A palavra reservada **FIMSE** indica o final da estrutura de seleção. Vale ressaltar

que o bloco **SENAO** não é obrigatório: caso só queiramos executar algumas instruções se a expressão lógica for verdadeira, podemos omitir o bloco **SENAO**.

Acompanhe o desenvolvimento de um algoritmo para encontrar a média de duas notas e mostrar se o aluno foi aprovado ou reprovado, sendo que, o critério para a aprovação é ter média superior a 6.

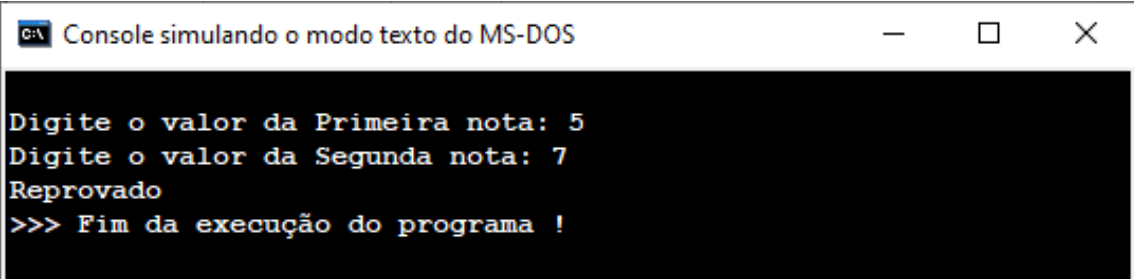
Resolução em VisualG

```

1 Algoritmo "Condicional01"
2
3 // Professor : Flávio Mota da Cruz
4
5 // Desenvolver um algoritmo para encontrar a média
6 // de duas notas e mostrar se o aluno foi aprovado ou reprovado,
7 // sendo que, o critério para a aprovação é ter média superior a 6.
8
9 Var
10 nota1,nota2,media:real
11 Inicio
12
13 Escreva("Digite o valor da Primeira nota: ")
14 leia(nota1)
15 Escreva("Digite o valor da Segunda nota: ")
16 leia(nota2)
17 media <- (nota1 +nota2)/ 2 // Calculando a média
18 se (media > 6) entao // Verificando a condição da média
19     escreva("Aprovado") // Valor se verdadeiro
20 senao
21     escreva("Reprovado") // Valor se Falso
22 fimse // Fim da condição
23
24 Fimalgoritmo

```

Saída no console



```

C:\> Console simulando o modo texto do MS-DOS
Digite o valor da Primeira nota: 5
Digite o valor da Segunda nota: 7
Reprovado
>>> Fim da execução do programa !

```


Para esse exemplo vamos usar o operador lógico “OU”, para que uma sentença seja verdadeira basta que uma das opções seja verdadeira.

2 - Sendo dado um dia da semana (Segunda, Terça, Quarta, Quinta, Sexta, Sábado e Domingo), faça um algoritmo que escreva “Casa” para “Sábado” e “Domingo” e escreva “Trabalho” para os demais dias.	
O problema	Exibir ao usuário “Casa” caso o dia da semana informado seja Sábado ou Domingo, ou exibir “Trabalho” para os demais casos.
	Observações complementares
	Nenhuma.
Solução esperada	Informar se o dia informado refere-se a estar em casa ou no trabalho.
Dados de Entrada	Dia da semana;
	Detalhamento dos Dados de Entrada
	Nenhum.
Dados de Saída	Local onde o usuário estará no dia informado.
Etapas encontradas	<ul style="list-style-type: none"> ✓ Solicitar o dia da semana ao usuário; ✓ Escrever a frase correspondente ao dia informado.
Descrição Narrativa da solução encontrada	<ul style="list-style-type: none"> 1. Solicitar o dia da semana; 2. Se o dia informado for igual a Sábado ou Domingo, apresentar a mensagem “Casa”, e se não for, apresentar a mensagem “trabalho”.

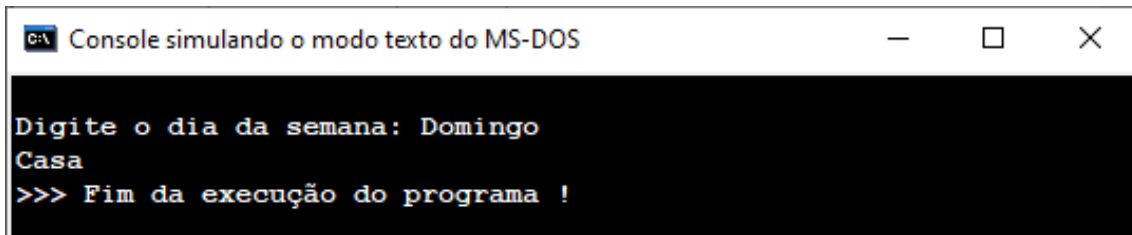
Resolução em VisualG

```

1  Algoritmo "Condicional01"
2
3  // Professor : Flávio Mota da Cruz
4
5  // Sendo dado um dia da semana (Segunda, Terça, Quarta,
6  // Quinta, Sexta, Sábado e Domingo), faça um algoritmo
7  // que escreva "Casa" para "Sábado" e "Domingo" e escreva
8  // "Trabalho" para os demais dias.
9
10 Var
11     diaSemana: caracter
12 Inicio
13
14     Escreva("Digite o dia da semana: ")
15     Leia(diaSemana)
16
17     se ((diaSemana = "Sabado") ou (diaSemana = "Domingo")) entao
18         escreva("Casa") // Valor se verdadeiro
19     senao
20         escreva("Trabalho") // Valor se Falso
21     fimse // Fim da condição
22
23 Fimalgoritmo

```

Saída no console

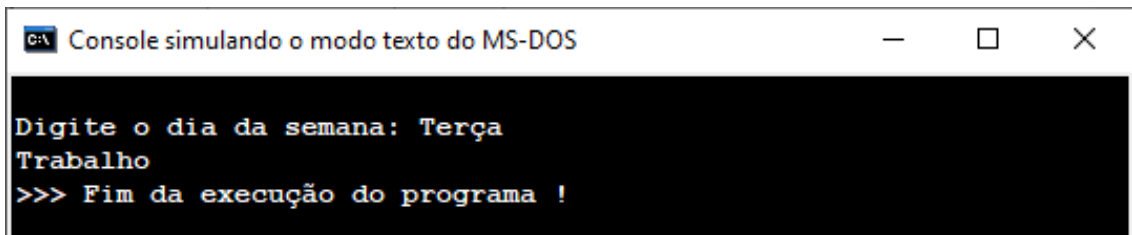


```

C:\> Console simulando o modo texto do MS-DOS

Digite o dia da semana: Domingo
Casa
>>> Fim da execução do programa !
  
```

Saída no console



```

C:\> Console simulando o modo texto do MS-DOS

Digite o dia da semana: Terça
Trabalho
>>> Fim da execução do programa !
  
```

Para esse exemplo vamos usar o operador lógico “E”, para que uma sentença seja verdadeira todas as opções desta sentença devem ser verdadeiras.

3 - Faça um algoritmo que receba como argumento de entrada o total de ganhos de uma pessoa, em reais, e que calcule o desconto do imposto de renda, segundo a tabela a seguir:

Faixa Salarial em R\$		Alíquota de Desconto	
Até 500,00		Isento (0%)	
De 500,00 até 1.500,00		10%	
De 1.500,00 até 2.500,00		15%	
Acima de 2.500,00		25%	
O problema	Calcular o desconto do imposto de renda de uma pessoa, baseado em seu rendimento.		
	Observações complementares		
	Para o correto cálculo do desconto, deve-se utilizar a tabela fornecida pelo enunciado.		
Solução esperada		Conhecer o valor referente ao imposto de renda a ser descontado do salário de uma pessoa.	
Dados de Entrada	Total de ganhos da pessoa;		
	Detalhamento dos Dados de Entrada		
	Nenhum.		
Dados de Saída		Valor do desconto:	

Etapas encontradas	<ul style="list-style-type: none"> • Solicitar o total de ganhos da pessoa; • Verificar na tabela o percentual de desconto a que o salário tem direito; • Fazer o cálculo; • Mostrar o resultado ao usuário.
Descrição Narrativa da solução encontrada	<ol style="list-style-type: none"> 1. Solicitar o total de ganhos da pessoa; 2. Se o total for menor ou igual a R\$500,00 então, o desconto será de R\$0,00; 3. Se o total for maior que R\$500,00 e menor ou igual a R\$1.500,00, então o desconto será obtido através da fórmula : renda * 10 / 100; 4. Se o total for maior que R\$1.500,00 e menor ou igual a R\$2.500,00, então o desconto será obtido através da fórmula : renda * 15 / 100; 5. Se o total for maior que R\$2.500,00 o desconto será obtido através da fórmula : renda * 25 / 100; 6. Tendo feito o cálculo, exibir ao usuário o valor do desconto.

Resolução em VisualG

```

1 Algoritmo "Condicional03"
2 // Professor : Flávio Mota da Cruz
3 // Faça um algoritmo que receba como argumento de entrada o total
4 // de ganhos de uma pessoa, em reais, e que calcule o desconto do
5 // imposto de renda, segundo a tabela a seguir:
6
7 // Faixa Salarial em R$           Aliquota de Desconto
8 // Até 500,00 ----- Isento (0%)
9 // De 500,00 até 1.500,00 ---- 10%
10 // De 1.500,00 até 2.500,00 --- 15% Acima de 2.500,00 ----- 25%
11 Var
12     renda,desconto: Real
13 Inicio
14     Escreva("Digite o total de ganhos: ")
15     leia(renda)
16     se renda <= 500 entao
17         desconto <- 0
18     senao
19         se ((renda > 500) e (renda <=1500)) entao
20             desconto <- renda * 10 / 100
21         senao
22             se ((renda > 1500) e (renda <=2500)) entao
23                 desconto <- renda * 15 / 100
24             senao
25                 desconto <- renda * 25 / 100
26             fimse
27         fimse
28     fimse
29     escreva("Total do desconto: ", desconto)
30 Fimalgoritmo

```

Comando de seleção múltipla (Escolha caso)

O VisuAlg implementa (com certas variações) o comando switch case das linguagens modernas como C#, Java, Javascript entre outras. 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 os exemplos a seguir, que ilustram bem o que faz este comando:

Exemplo 01

```

1 Algoritmo "Meses"
2 // Autor : Flávio Mota
3 var
4   mes : inteiro
5 inicio
6   escreva ("Entre com um valor de 01 a 12:")
7   leia (mes)
8   escolha mes
9   caso 1
10    escreval ("Janeiro")
11  caso 2
12    escreval ("Fevereiro")
13  caso 3
14    escreval ("Março")
15  caso 4
16    escreval ("Abril")
17  caso 5
18    escreval ("Maio")
19  caso 6
20    escreval ("Junho")
21  // aqui, os outros meses .....
22  outrocaso
23    escreval ("Mês inválido.")
24  fimescolha
25 fimalgoritmo
  
```

Nesse exemplo, o valor mostrado pelo algoritmo depende do valor digitado para a variável mês: caso 1 o sistema mostra "janeiro", caso 2 o sistema mostra fevereiro e assim por diante.

Exemplo 02

```

1 Algoritmo "Notas"
2 // Autor : Flávio Mota
3 var
4 nota : inteiro
5 inicio
6     escreva("Entre com a nota do aluno:")
7     leia(nota)
8     escolha nota
9     caso 0,1,2,3
10        escreval("Reprovado.")
11    caso 5 ate 7, 4
12        // A lista não precisa estar em uma ordem específica
13        // Só na cláusula ATE o primeiro valor precisam ser menor
14        // que o segundo
15        escreval("Em recuperação.")
16    caso 8 ate 10
17        escreval("Aprovado")
18    outrocaso
19        escreval("Nota inválida.")
20    fimescolha
21 fimalgoritmo

```

Exemplo 03

```

1 Algoritmo "Times"
2 // Autor : Flávio Mota
3
4 var time : caracter
5 inicio
6
7     escreva("Entre com o nome de um time de futebol:")
8     leia(time)
9     escolha time
10    caso "Fluminense", "Flamengo", "Vasco", "Botafogo"
11        escreva("É um time carioca.")
12    caso "São Paulo", "Palmeiras", "Santos", "Corinthians"
13        escreva("É um time paulista.")
14    outrocaso
15        escreva("É de outro estado.")
16    fimescolha
17
18 fimalgoritmo

```

Exercícios no CADERNO 01: Algoritmos nível 02 – CONDICIONAIS – SE, SENAO, SENAO SE

Comando de repetição (PARA DE ATE ...FAÇA)

Esta estrutura repete uma sequência de comandos um determinado número de vezes.

para <variável> de <valor-inicial> ate <valor-limite> [passo <incremento>] faça
 <sequência-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 >	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
fimpara	Indica o fim da sequência 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 sequência 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.

```

1 Algoritmo "Estrutura_Repetição_PARA"
2 // Professor: Flavio Mota
3 Var
4
5     I:inteiro
6
7 Inicio
8
9     // inicial final
10    para i de 1 ate 10 faca
11
12        escreval(i) // o valor de i será atualizado a cada loop
13
14    fimpara
15
16 Fimalgoritmo

```

No exemplo a seguir, os números de 1 a 10 são exibidos em ordem decrescente, este outro exemplo, no entanto, funcionará por causa do **passo -1** e repare que o valor inicial é o 10.

```

1 Algoritmo "Estrutura_Repetição_PARA"
2 // Professor: Flavio Mota
3 Var
4
5     I:inteiro
6
7 Inicio
8
9     // inicial final
10    para i de 10 ate 1 passo -1 faca
11
12        escreval(i) // o valor de i será atualizado a cada loop
13
14    fimpara
15
16 Fimalgoritmo

```

Vamos criar uma **tabuada** onde será repetido um bloco de código 10 vezes, nesse caso deveremos usar a estrutura de repetição **PARA**, pois o código fica mais limpo e profissional.

Exemplo: dado um valor o algoritmo deve criar uma tabuada desde valor formatando as informações como se fosse uma tabuada real: "5 X 1 = 5 , 5 X 2

= 10, $5 \times 3 = 15$ e assim por diante, caso o valor digitado seja o 5, porém o algoritmo deve servir para qualquer valor que o usuário digitar.

```

1 Algoritmo "Estrutura_Repeticao_PARA"
2 // Professor: Flavio Mota
3 // Tabuada
4 Var
5
6     i:inteiro
7     valor:inteiro
8 Inicio
9
10    Escreva("Digite o valor para a tabuada: ")
11    leia(valor)
12    para i de 1 ate 10 faca
13
14        escreval(valor, " x " , i, " = ", valor * i)
15
16    fimpara
17 Fimalgoritmo

```

Efetuar a leitura de 10 números inteiros e apresentar a soma desses valores	
O problema	Informar ao usuário a soma dos valores digitados.
	Observações complementares
	✓ Todos os valores informados devem ser acumulados em uma variável ✓ Serão informados 10 números inteiros.
Solução esperada	Informar ao usuário a soma dos valores digitados.
Dados de Entrada	Dez números inteiros informados pelo usuário;
	Detalhamento dos Dados de Entrada
	Os números informados devem ser do tipo inteiro, obrigatoriamente.
Dados de Saída	A soma dos valores digitados
Etapas encontradas	✓ Solicitar dez números ao usuário; ✓ Efetuar o cálculo para obter a soma dos valores
Descrição Narrativa da solução encontrada	1. Solicitar dez números ao usuário; 2. Para cada número informado, realizar o cálculo do valor atual com o valor acumulado anteriormente; 3. Após a leitura dos dez valores o algoritmo deve mostrar a soma desses valores ao usuário.

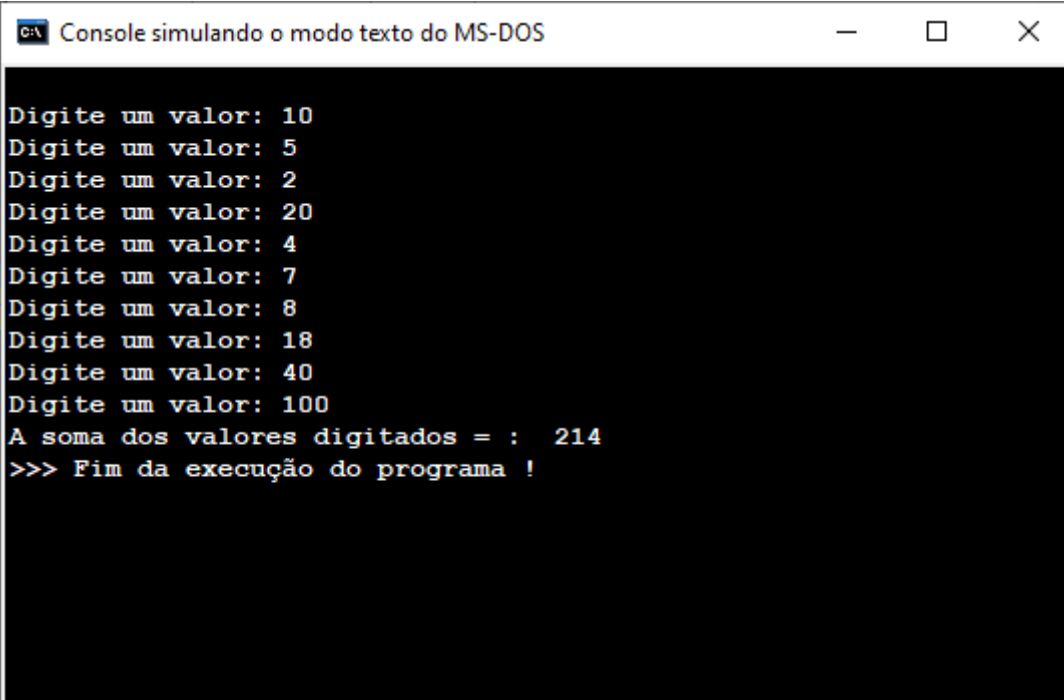
Solução em VisualG

```

1 Algoritmo "Estrutura_Repeticao_PARA"
2 // Professor: Flavio Mota
3
4 Var
5
6     i:inteiro
7     valor:inteiro
8     soma:inteiro
9 Inicio
10
11
12     para i de 1 ate 10 faca
13
14         Escreva("Digite um valor: ")
15         leia(valor)
16         soma <- soma + valor // acumulador
17
18     fimpara
19     escreva("A soma dos valores digitados = : ", soma)
20
21 Fimalgoritmo

```

Resultado no console, foi digitado 10 valores e ao final o algoritmo mostrou a soma dos valores digitados pelo usuário.



```

C:\> Console simulando o modo texto do MS-DOS

Digite um valor: 10
Digite um valor: 5
Digite um valor: 2
Digite um valor: 20
Digite um valor: 4
Digite um valor: 7
Digite um valor: 8
Digite um valor: 18
Digite um valor: 40
Digite um valor: 100
A soma dos valores digitados = : 214
>>> Fim da execução do programa !

```

Comando de repetição (ENQUANTO ...FAÇA)

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

```
enquanto <expressão-lógica> faça
    <sequência-de-comandos>
fimenquanto
```

<expressão-lógica> - Esta expressão que é avaliada antes de cada repetição do laço. Quando seu resultado for VERDADEIRO, <sequência-de-comandos> é executada.

Fimenquanto - Indica o fim da <sequência-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 <sequência-de-comandos> será executada mais uma vez; caso contrário, a execução prosseguirá a partir do primeiro comando após fimenquanto.

Importante: Como o laço enquanto...faça testa sua condição de parada antes de executar sua sequência de comandos, esta sequência poderá ser executada zero ou mais vezes.

Exemplo 01

```
1 Algoritmo "ENQUANTO 01"
2 // Professor : Flávio Mota
3 Var
4
5     i:inteiro
6 Início
7     i <- 1
8     enquanto i <= 10 faça
9         escreva(i)
10        i<-i+1
11    fimenquanto
12 Fimalgoritmo
```

Nesse exemplo, ENQUANTO a variável *i* for menor ou igual a 10 a condição será satisfeita e o loop continua rodando mostrando o valor de *i*, mas quando o valor de *i* for 11 o algoritmo para a estrutura de repetição e passa para o próximo bloco de código ou finaliza o algoritmo como mostra esse exemplo.

Exemplo 02 – Vamos usar o exemplo anterior da tabuada, agora vamos usar a estrutura de repetição **ENQUANTO**:

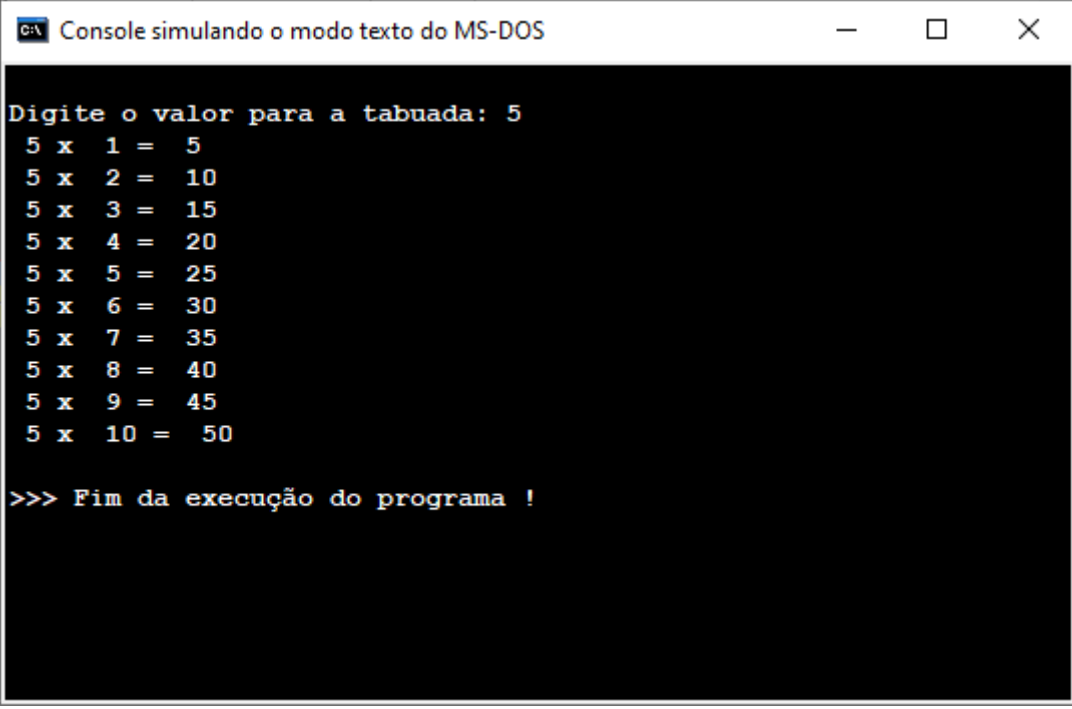
Exemplo: dado um valor o algoritmo deve criar uma tabuada desde valor.

```

1 Algoritmo "Estrutura_Repeticao_ENQUANTO"
2 // Professor: Flavio Mota
3 // Tabuada
4 Var
5
6     i:inteiro
7     valor:inteiro
8 Inicio
9
10    Escreva("Digite o valor para a tabuada: ")
11    leia(valor)
12    i <- 1
13    enquanto i <= 10 faca
14        escreval(valor, " x " , i, " = ", valor * i)
15        i <- i + 1
16    fimenquanto
17 Fimalgoritmo

```

Nesse exemplo foi digitado o valor 5 e o algoritmo apresenta a tabuada formatada conforme o exemplo abaixo.



Console simulando o modo texto do MS-DOS

```

Digite o valor para a tabuada: 5
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50

>>> Fim da execução do programa !

```

Sendo dado um número positivo, faça um algoritmo que escreva todos os números positivos menores que esse número.	
O problema	Escrever todos os números positivos menores que o informado.
	Observações complementares
	<ul style="list-style-type: none"> ✓ O número deve ser positivo e diferente de ZERO; ✓ A contagem dever terminar sempre em ZERO.
Solução esperada	Escrever todos os números positivos menores que o informado.
Dados de Entrada	Número positivo informado pelo usuário.
	Detalhamento dos Dados de Entrada
	Nenhum.
Dados de Saída	Números positivos menores que o informado.
Etapas encontradas	<ul style="list-style-type: none"> ✓ Solicitar um número positivo ao usuário; ✓ Enquanto o número for maior que zero, diminuir 1 e exibir seu novo valor ao usuário;
Descrição Narrativa da solução encontrada	<ol style="list-style-type: none"> 1. Solicitar um número positivo ao usuário; 2. Se o número informado for menor ou igual a ZERO, informar que o número deve ser positivo e maior que ZERO e solicitar um outro número; 3. Se o número for maior que ZERO, para cada número, subtraia em 1 até ser igual a ZERO, mostrando sempre o resultado ao usuário.

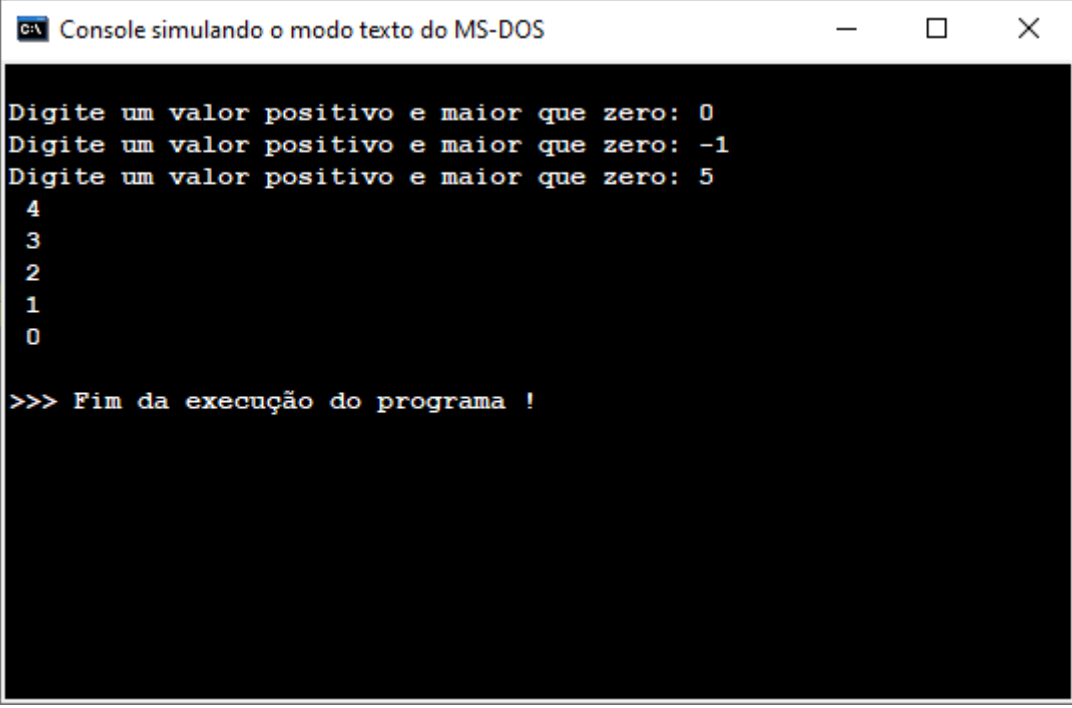
Resolução em VisualG

```

1 Algoritmo "Estrutura_Repeticao_ENQUANTO"
2 // Professor: Flavio Mota
3 Var
4     i:inteiro
5     valor:inteiro
6 Inicio
7     Escreva("Digite um valor positivo: ")
8     leia(valor)
9
10    enquanto valor <= 0 faca
11        Escreva("Digite um valor positivo e maior que zero: ")
12        leia(valor)
13    fimenquanto
14
15    enquanto valor > 0 faca
16        valor <- valor - 1
17        escreval(valor)
18    fimenquanto
19 Fimalgoritmo

```

Resultado no console.



```
C:\> Console simulando o modo texto do MS-DOS

Digite um valor positivo e maior que zero: 0
Digite um valor positivo e maior que zero: -1
Digite um valor positivo e maior que zero: 5
4
3
2
1
0

>>> Fim da execução do programa !
```

No exemplo acima foi digitado o valor **0** e a estrutura de repetição verificou que a condição é verdadeira e pediu para digitar um novo valor, o próximo valor digitado foi **-1** e como a condição continua verdadeira o bloco dentro da estrutura enquanto foi executado novamente, pois a condição continua sendo atendida, somente quando o valor **5** foi digitado e foi verificado que a condição não foi mais atendida e o algoritmo passou para o próximo passo onde é feito uma nova verificação executando o segundo enquanto, nesse caso, enquanto o valor digitado for maior que 0, será mostrado todos os valores menores que esse valor.

Comando de repetição (REPITA ...ATE)

Esta estrutura repete uma sequência 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 <sequência-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.

```

1 Algoritmo "REPITA_01"
2 // Professor : Flávio Mota
3 Var
4
5     i:inteiro
6 Inicio
7     i <- 1
8     repita
9
10        escreval(i)
11        i<-i+1
12
13     ate i > 10
14 Fimalgoritmo

```

Nesse exemplo, usando a estrutura REPITA, enquanto a variável *i* for menor e igual a 10 a condição será satisfeita e o loop continua rodando mostrando o valor de *i*, mas quando o valor de *i* for maior que 10 o algoritmo para a estrutura de repetição e passa para o próximo bloco de código ou finaliza o algoritmo como mostra nosso exemplo.

Vamos criar o mesmo exemplo da tabuada usada na estrutura **PARA E ENQUANTO** usando a estrutura de repetição **REPITA...ATE**

```

1 Algoritmo "Estrutura_Repeticao_REPITA"
2 // Professor: Flavio Mota
3 // Tabuada
4 Var
5
6     i:inteiro
7     valor:inteiro
8 Inicio
9
10    Escreva("Digite o valor para a tabuada: ")
11    leia(valor)
12    i <- 1
13    repita
14        escreval(valor, " x " , i, " = ", valor * i)
15        i <- i + 1
16    ate i > 10
17 Fimalgoritmo

```

Vamos replicar o exercício anterior que usava a estrutura de repetição **ENQUANTO** agora usando a estrutura de repetição **REPITA**.

Exercício: Sendo dado um número positivo, faça um algoritmo que escreva todos os números positivos menores que esse número até o valor zero.

```

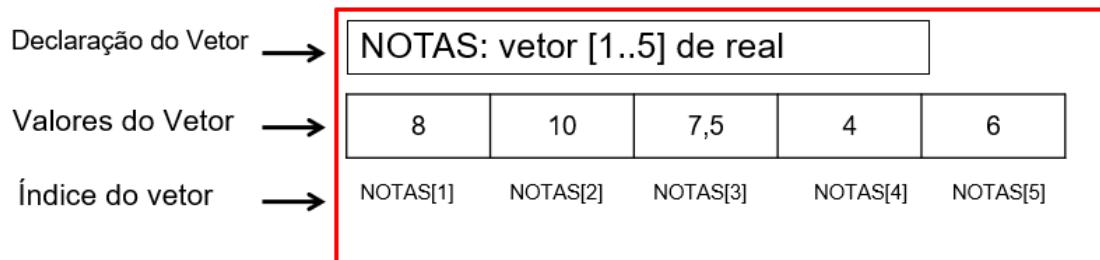
1 Algoritmo "Estrutura_Repeticao_REPITA"
2 // Professor: Flavio Mota
3 Var
4   i: inteiro
5   valor: inteiro
6 Inicio
7
8
9   repita
10     Escreva("Digite um valor positivo e maior que zero: ")
11     leia(valor)
12   ate valor > 0
13
14
15   repita
16     valor <- valor - 1
17     escreval(valor)
18   ate valor = 0
19
20 Fimalgoritmo

```

Exercícios no CADERNO 01: Algoritmos nível 03 – ESTRUTURA DE REPETIÇÃO – PARA, ENQUANTO, REPITA.

VETOR: estrutura indexada unidimensional

Os tipos homogêneos são conjuntos do mesmo tipo básico. A utilização desse tipo de estrutura de dados recebe diversos nomes, tais como: variáveis indexadas, compostas, arranjos, tabelas em memória, Arrays (do inglês) vetores e matrizes. Quando possuímos uma ordem e um índice de acesso aos elementos de um conjunto então temos caracterizado um **VETOR**.



Nesse exemplo temos um vetor declarado no interpretador de pseudocódigo Visualg, representado pelo nome NOTAS[], com o tamanho de 5 índices representado por vetor[1..5] que possui índices representado por NOTAS[1], NOTAS[2], NOTAS[3] ... e seus valores correspondentes 8,10, 7,5. Para cada índice podemos alocar um valor para um acesso posterior, desta forma podemos organizar de forma adequada uma quantidade de dados do mesmo tipo.

Imagine a seguinte situação, onde queremos armazenar 5 nomes de 5 pessoas

```

1 algoritmo "Exemplo_variáveis_simples"
2
3 var
4     nome1:caracter
5     nome2:caracter
6     nome3:caracter
7     nome4:caracter
8     nome5:caracter
9 inicio
10    nome1 <- "Flavio"
11    nome2 <- "Rochele"
12    nome3 <- "Jesus"
13    nome4 <- "Antonia"
14    nome5 <- "João"
15    escreval (nome1)
16    escreval (nome2)
17    escreval (nome3)
18    escreval (nome4)
19    escreval (nome5)
20 fimalgoritmo
  
```

Exemplo com variáveis simples

```

1 algoritmo "Exemplo_Vetor"
2
3 var
4     texto: vetor [1..5] de caracter
5 inicio
6     texto[1] <- "Flavio"
7     texto[2] <- "Rochele"
8     texto[3] <- "Jesus"
9     texto[4] <- "Antonia"
10    texto[5] <- "João"
11    escreval (texto[1])
12    escreval (texto[2])
13    escreval (texto[3])
14    escreval (texto[4])
15    escreval (texto[5])
16 fimalgoritmo
  
```

Exemplo com variável indexada do tipo vetor

No exemplo a seguir vamos mostrar como trabalhar de forma eficiente com vetores usando a estrutura de repetição PARA. Sempre que trabalharmos com vetores é uma boa prática usar uma estrutura de repetição para percorrer os índices da estrutura indexada.

```

1 algoritmo "Exemplo_Vetor"
2 // Autor: Flávio Mota
3 var
4   nomes: vetor [1..5] de caracter // Declarando o vetor
5   i: inteiro
6 inicio
7
8   para i de 1 ate 5 faca
9     escreva ("Digite um nome: ")
10    leia (nomes[i]) // Lendo o Vetor
11  fimpara
12
13  escreval(" ===== NOMES ARMAZENADOS NO VETOR ===== ")
14  para i de 1 ate 5 faca
15    escreval (nomes[i]) // Mostrando o Vetor
16  fimpara
17
18 fimalgoritmo

```

Nesse exemplo o usuário vai digitar 5 nomes e armazenar no vetor **nomes[]**, conforme mostra a primeira estrutura de repetição PARA, logo após ler os 5 valores para o vetor o algoritmo mostra os valores armazenados na segunda estrutura de repetição PARA.

```

C:\ Console simulando o modo texto do MS-DOS

Digite um nome: Flavio
Digite um nome: Rochele
Digite um nome: Raquel
Digite um nome: Antonia
Digite um nome: João
===== NOMES ARMAZENADOS NO VETOR =====
Flavio
Rochele
Raquel
Antonia
João

>>> Fim da execução do programa !

```

Criar um algoritmo que tenha 2 vetores, um para receber 5 nomes de alunos e outro para receber 5 notas dos alunos, mostrar os nomes e as notas digitadas, mostrar também a média total das notas.	
O problema	Ler e mostrar 5 nomes e 5 notas de alunos e mostrar também a média total dessas notas.
	Observações complementares
	<ul style="list-style-type: none"> ✓ O vetor nome deve ser do tipo caracter; ✓ O vetor notas deve ser do tipo real.
Solução esperada	Escrever todos os nomes, notas e a média de todas as notas dos alunos.
Dados de Entrada	5 nomes e 5 notas em vetores
	Detalhamento dos Dados de Entrada
	Nenhum.
Dados de Saída	Todos os nomes, notas e a média de todas as notas dos alunos.
Etapas encontradas	✓ Solicitar cinco nomes e cinco notas ao usuário;
	✓ Mostrar cinco nomes e cinco notas mais a média das notas.
Descrição Narrativa da solução encontrada	<ol style="list-style-type: none"> 1. Solicitar o nome e notas de cinco alunos ao usuário; 2. Acumular em uma variável a somatória de todas as notas do vetor nota[]; 3. Fazer a média das notas; 4. Mostrar os nomes e notas dos alunos; 5. Mostrar a média das notas.

Resolução em Visualg

```

1 algoritmo "Exemplo_Vetor"
2 var
3     nome: vetor [1..5] de caracter
4     nota: vetor [1..5] de real
5     i: inteiro
6     soma: real
7     media: real
8 inicio
9     soma <- 0
10    para i de 1 ate 5 faca
11        escreva ("Digite o nome: ")
12        leia (nome[i])
13        escreva ("Digite a nota: ")
14        leia (nota[i])
15        soma <- soma + nota[i]
16    fimpara
17    escreval(" ===== NOME E NOTAS DOS ALUNOS ===== ")
18    para i de 1 ate 5 faca
19        escreval ("Nome: ", nome[i])
20        escreval ("Nota: ", nota[i])
21    fimpara
22    media <- soma / 5
23    escreval (" A media das notas = ", media)
24 fimalgoritmo

```

Entrada e saída de dados no console do Visualg

```

C:\> Console simulando o modo texto do MS-DOS

Digite o nome: Flavio
Digite a nota: 10
Digite o nome: Rochele
Digite a nota: 10
Digite o nome: Raquel
Digite a nota: 9
Digite o nome: Antonia
Digite a nota: 8
Digite o nome: João
Digite a nota: 8
===== NOME E NOTAS DOS ALUNOS =====
Nome: Flavio
Nota: 10
Nome: Rochele
Nota: 10
Nome: Raquel
Nota: 9
Nome: Antonia
Nota: 8
Nome: João
Nota: 8
A media das notas = 9

>>> Fim da execução do programa !
  
```

MATRIZ: estrutura indexada bidimensional

Uma matriz é uma variável composta homogênea bidimensional formada por uma sequência de variáveis, todas do mesmo tipo, com o mesmo identificador (mesmo nome) e alocadas sequencialmente na memória.

Declaração de uma Matriz (Visualg)

matrizA: vetor [1..2,1..4] de inteiro

1	2	45	98
33	71	23	59

Nesse exemplo, temos uma matriz chamada **matrizA** representada por linhas e colunas **vetor[1..2 linhas, 1..4 colunas]** toda matriz bidimensional é representada por linhas e colunas, no exemplo **matrizA[1,2] <- 2** o valor 2 está localizado na linha 1 coluna 2 e no exemplo **matrizA[2,3] <- 23** o valor 23 está localizado na linha 2 e coluna 3.

Nesse exemplo vamos criar uma matriz chamada **matrizA com duas linhas e duas colunas** e atribuir o valor **zero** a todas as posições desta matriz e logo depois mostramos essa matriz.

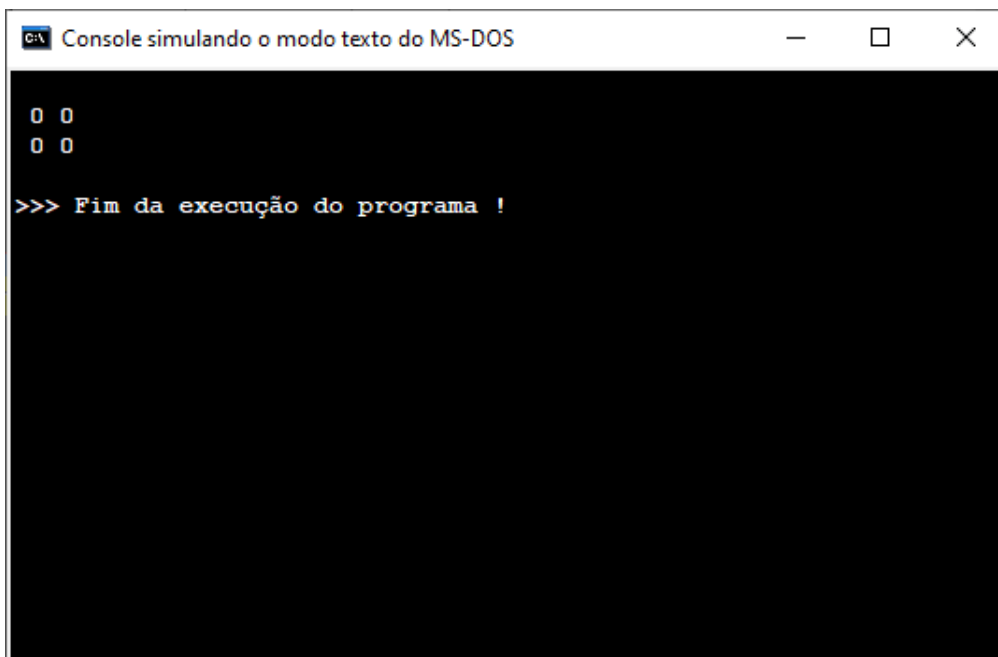
Repare que nesse algoritmo temos duas variáveis de controle, uma que controla a linha da matriz “i” e outra que controla a coluna da matriz “j”, nesse caso temos que usar uma estrutura de repetição “PARA” externo que controla a linha e uma estrutura de repetição “PARA” interno que controla a coluna.

```

1 algoritmo "matriz"
2 var
3   matrizA: vetor[1..2,1..2] de inteiro
4   i,j: inteiro
5 inicio
6   para i <- 1 ate 2 faca //percorre a linha da matriz
7     para j <- 1 ate 2 faca //percorre a coluna da matriz
8       matrizA[i,j]<-0 // atribuindo Zero a todas as
9       fimpara           // posições da matrizA
10    fimpara
11    para i <- 1 ate 2 faca
12      para j <- 1 ate 2 faca
13        escreva(matrizA[i,j])//Mostrando a matriz
14      fimpara
15      escreval("")
16    fimpara
17 fimalgoritmo

```

Quando executar o algoritmo, esse será o resultado no console.



```

0 0
0 0

>>> Fim da execução do programa !

```

Criar um algoritmo que armazene o nome de 5 estados e suas capitais e depois mostre os dados na tela USANDO MATRIZ.	
O problema	Ler e mostrar 5 estados e 5 capitais em uma estrutura do tipo matriz
	Observações complementares
	<ul style="list-style-type: none"> ✓ A matriz deve ser do tipo caracter; ✓ Com 5 linhas e duas colunas
Solução esperada	Mostrar de forma organizada os nomes dos estados e suas capitais.
Dados de Entrada	Matriz com cinco linhas e duas colunas
	Detalhamento dos Dados de Entrada
	Nenhum.
Dados de Saída	Mostrar de forma organizada os nomes dos estados e suas capitais.
Etapas encontradas	<ul style="list-style-type: none"> ✓ Solicitar cinco estados e cinco capitais ao usuário; ✓ Mostrar cinco estados e suas capitais
Descrição Narrativa da solução encontrada	<ol style="list-style-type: none"> 1. Solicitar o nome e capitais de cinco estados ao usuário; 2. Organizar em uma matriz de cinco linhas e duas colunas 3. Mostrar as informações ao usuário de forma organizada

Resolução no VisualG

```

1 algoritmo "ESTADO_CAPITAL"
2 var
3     Estado_Capital:vetor[1..5,1..2] de caracter
4     i,j:inteiro
5 inicio
6     // LENDO A MATRIZ
7     para i de 1 ate 5 faca
8         para j de 1 ate 2 faca
9             se j = 1 entao
10                Escreva("Entre com o nome do Estado: ")
11                Leia(Estado_Capital[i,j])
12            fimse
13            se j = 2 entao
14                Escreva("Entre com o nome do Capital: ")
15                Leia(Estado_Capital[i,j])
16            fimse
17        fimpara
18    fimpara

```

Nesse exemplo para cada linha que a estrutura de repetição representada por "i" percorrer, serão percorridas duas colunas, a condicional "se" está sendo usada para identificar em qual coluna o usuário está no momento da digitação.

```

19  // MOSTRANDO A MATRIZ
20  para i de 1 ate 5 faca
21      para j de 1 ate 2 faca
22          se j = 1 entao
23              Escreva("Estado: ",Estado_Capital[i,j])
24          fimse
25          se j = 2 entao
26              Escreva(" - Capital: ",Estado_Capital[i,j])
27          fimse
28      fimpara
29      escreval("")
30  fimpara
31 fimalgoritmo

```

Quando executar e terminar o algoritmo, esse será o resultado no console do VisualG.

```

C:\ Console simulando o modo texto do MS-DOS

Entre com o nome do Estado: São Paulo
Entre com o nome do Capital: São Paulo
Entre com o nome do Estado: Bahia
Entre com o nome do Capital: Salvador
Entre com o nome do Estado: Rio de Janeiro
Entre com o nome do Capital: Rio de Janeiro
Entre com o nome do Estado: Minas Gerais
Entre com o nome do Capital: Belo Horizonte
Entre com o nome do Estado: Ceara
Entre com o nome do Capital: Fortaleza
==== Dados na Matriz ====
Estado: São Paulo - Capital: São Paulo
Estado: Bahia - Capital: Salvador
Estado: Rio de Janeiro - Capital: Rio de Janeiro
Estado: Minas Gerais - Capital: Belo Horizonte
Estado: Ceara - Capital: Fortaleza

>>> Fim da execução do programa !

```

Exercícios no CADERNO 01: Algoritmos nível 04 – ESTRUTURA DE DADOS – VETORES E MATRIZES

PROCEDIMENTO: Subprograma, sub-rotina, método ou módulo

Subprograma é um programa que auxilia o programa principal através da realização de uma determinada subtarefa. Também costuma receber os nomes de sub-rotina, procedimento, método ou módulo. Os subprogramas são chamados dentro do corpo do programa principal como se fossem comandos. Após seu término, a execução continua a partir do ponto onde foi chamado. É importante compreender que a chamada de um subprograma simplesmente gera um **desvio provisório no fluxo de execução**.

Cada subprograma, além de ter acesso às variáveis do programa que o chamou (são as variáveis globais), pode ter suas próprias variáveis (são as variáveis locais), que existem apenas durante sua chamada.

Ao se chamar um subprograma, também é possível passar-lhe determinadas informações que recebem o nome de parâmetros (são valores que, na linha de chamada, ficam entre os parênteses e que estão separados por vírgulas). A quantidade dos parâmetros, sua sequência e respectivos tipos não podem mudar: devem estar de acordo com o que foi especificado na sua correspondente declaração.

Para se criar subprogramas, é preciso descrevê-los após a declaração das variáveis e antes do corpo do programa principal. O VisualG possibilita declaração e chamada de subprogramas nos moldes da linguagem Pascal, ou seja, procedimentos e funções com passagem de parâmetros por valor ou referência. Exemplo da declaração de um procedimento.

procedimento <nome-de-procedimento> [(<seqüência-de-declarações-de-parâmetros>)]

// Seção de Declarações Internas

inicio

// Seção de Comandos

fimprocedimento

O **<nome-de-procedimento>** obedece às mesmas regras de nomenclatura das variáveis. Por outro lado, a **<sequência-de-declarações-de-parâmetros>** é uma sequência de **[var] <sequência-de-parâmetros>: <tipo-de-dado>** separadas por ponto e vírgula. A presença (opcional) da palavra-chave **var** indica passagem de parâmetros por referência; caso contrário, a passagem será por valor.

Por sua vez, **<sequência-de-parâmetros>** é uma sequência de nomes de parâmetros (também obedecem a mesma regra de nomenclatura de variáveis) separados por vírgulas.

De modo análogo ao programa principal, a seção de declaração internas começa com a palavra-chave **var**, e continua com a seguinte sintaxe:

<lista-de-variáveis>: <tipo-de-dado>

Vamos a um exemplo:

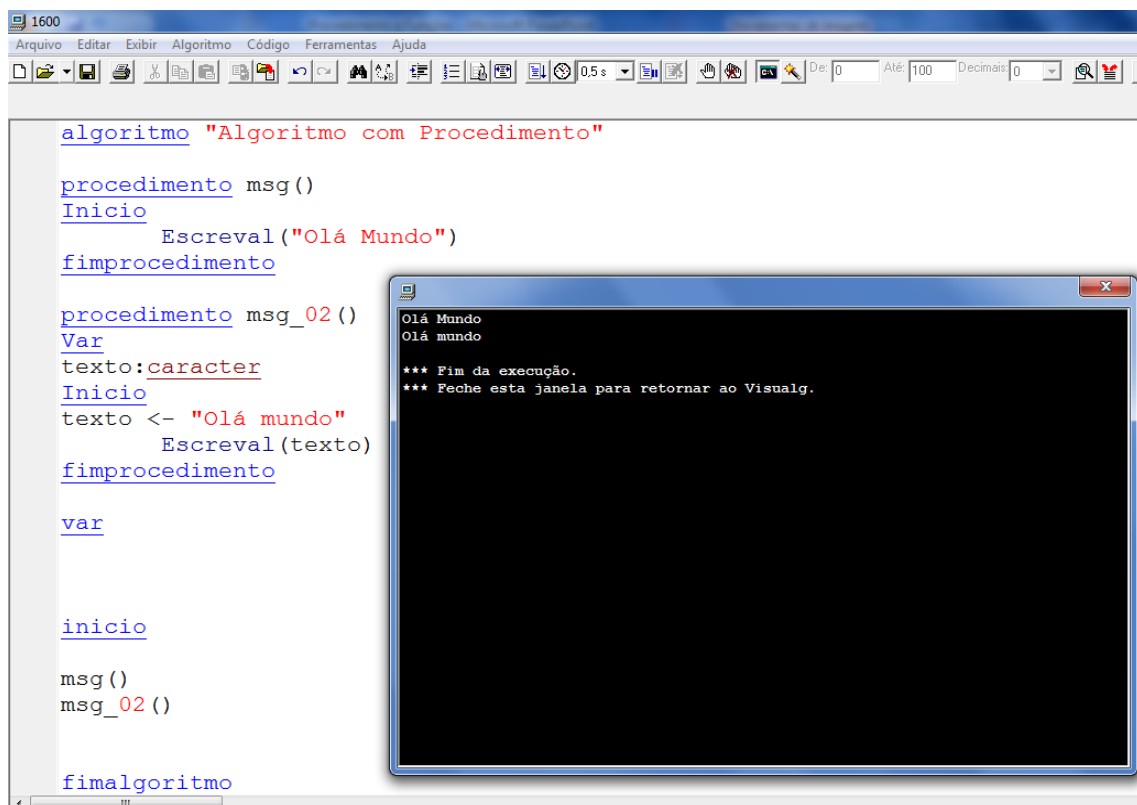
Através de um subprograma **soma**, será calculada a soma entre os valores 4 e 9 (ou seja, será obtido o resultado 13) que o programa principal imprimirá em seguida. No primeiro caso, um procedimento sem parâmetro utiliza uma variável local **aux** para armazenar provisoriamente o resultado deste cálculo (evidentemente, esta variável é desnecessária, mas está aí apenas para ilustrar o exemplo), antes de atribuí-lo à variável global **res**:


```

1 Algoritmo "Procedimento"
2 // Professor : Flavio Mota
3 procedimento soma() // inicio do procedimento
4 var
5   aux:inteiro // essa variavel é desnecessaria, está aqui
6   // somente para exemplo
7 inicio
8   aux<- n + m
9   res <- aux
10 fimprocedimento // fim do procedimento
11
12 Var
13   n,m,res:inteiro
14 Inicio
15   // algoritmo principal
16   n <- 4
17   m <- 9
18   soma()
19   escreva(res)
20 Fimalgoritmo

```

Vamos a outro exemplo usando procedimento



```

algoritmo "Algoritmo com Procedimento"

procedimento msg()
Inicio
  Escreval("Olá Mundo")
fimprocedimento

procedimento msg_02()
Var
  texto:caracter
Inicio
  texto <- "Olá mundo"
  Escreval(texto)
fimprocedimento

var

inicio

  msg()
  msg_02()

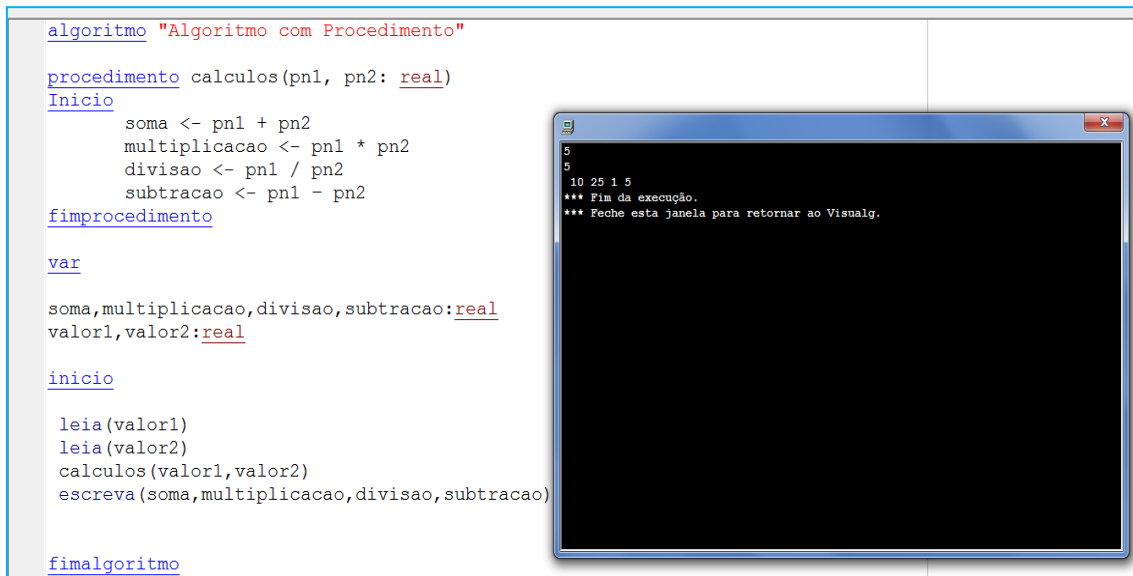
fimalgoritmo

```

Nesse exemplo criamos dois procedimentos: msg(), msg_02() e chamamos no algoritmo principal, ambos fazem a mesma coisa, porém o procedimento msg_02() usa uma variável local para guardar a frase "olá mundo".

Procedimentos com parâmetros

Os procedimentos e as funções podem receber dados através dos parâmetros declarados entre os parênteses ao lado de seus nomes, essas informações vêm de fora do bloco do procedimento e é usada como uma variável interna.



```

algoritmo "Algoritmo com Procedimento"
procedimento calculos(pn1, pn2: real)
início
    soma <- pn1 + pn2
    multiplicacao <- pn1 * pn2
    divisao <- pn1 / pn2
    subtracao <- pn1 - pn2
fimprocedimento

var
soma,multiplicacao,divisao,subtracao:real
valor1,valor2:real

início
    leia(valor1)
    leia(valor2)
    calculos(valor1,valor2)
    escreva(soma,multiplicacao,divisao,subtracao)

finalgoritmo
  
```

The screenshot shows a Turbo Pascal IDE window titled "Algoritmo com Procedimento". The code defines a procedure named **calculos** that takes two real parameters, **pn1** and **pn2**. Inside the procedure, four variables are calculated: **soma** (sum), **multiplicacao** (multiplication), **divisao** (division), and **subtracao** (subtraction). The procedure is then called from the main algorithm block with user input values. An output window titled "Visualg" is also shown, displaying the results of the calculations for inputs 5 and 10.

Nesse exemplo estamos criando um procedimento chamado **calculos** que recebe dois valores por parâmetros representados por **pn1** e **pn2**, esses valores serão processados internamente e atribuídos as variáveis globais **soma**, **multiplicacao**, **divisao** e **subtracao**. Na chamada do procedimento no algoritmo principal "**calculos(valor1,valor2)**" serão passados dois valores para a função que neste caso será **valor1** e **valor2**, esses valores serão substituídos internamente na função por **pn1** e **pn2** e então serão feitos os cálculos com os valores de entrada nas variáveis do algoritmo principal.

Função: Subprograma, sub-rotina, método ou módulo com retorno.

Uma função é um tipo especial de sub-rotina que retorna um resultado de volta ao ponto onde foi chamada.

As funções embora bastante semelhantes aos procedimentos tem a característica especial de retornar ao programa que a chamou um valor associado ao nome da função. Todas as regras de programação válidas para os procedimentos são aplicadas nas funções.

funcao <nome-de-função> [(<sequência-de-declarações-de-parâmetros>)]: <tipo-de-dado>

// Seção de Declarações Internas

inicio

// Seção de Comandos

fimfuncao

O **<nome-de-função>** obedece às mesmas regras de nomenclatura das variáveis. Por outro lado, a <sequência-de-declarações-de-parâmetros> é uma sequência de [var] <sequência-de-parâmetros>: <tipo-de-dado> separadas por ponto e vírgula. A presença (opcional) da palavra-chave var indica passagem de parâmetros por referência; caso contrário, a passagem será por valor.

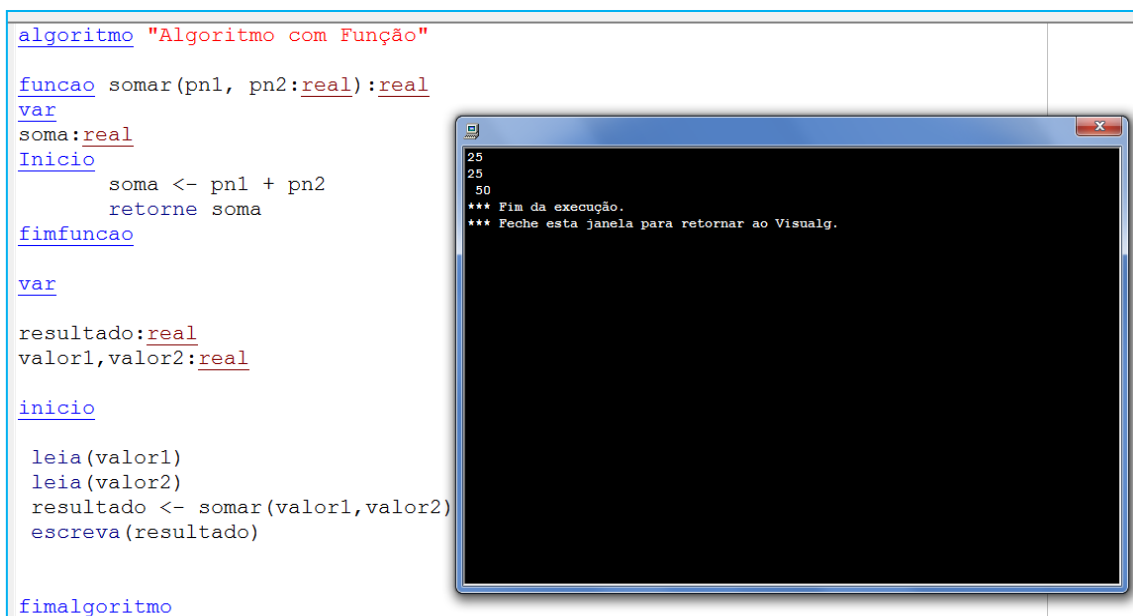
Por sua vez, <sequência-de-parâmetros> é uma sequência de nomes de parâmetros (também obedecem a mesma regra de nomenclatura de variáveis) separados por vírgulas.

O valor retornado pela função será do tipo especificado na sua declaração (logo após os dois pontos). Em alguma parte da função (de modo geral, no seu final), este valor deve ser retornado através do comando retorne.

De modo análogo ao programa principal, a seção de declaração interna começa com a palavra-chave `var`, e continua com a seguinte sintaxe: `<lista-de-variáveis> : <tipo-de-dado>`

Exemplo 01 - Função

Esse algoritmo tem uma função que recebe dois parâmetros e retorna a soma destes parâmetros.



```

algoritmo "Algoritmo com Função"

funcao somar(pn1, pn2:real):real
var
soma:real
Inicio
    soma <- pn1 + pn2
    retorne soma
fimfuncao

var
resultado:real
valor1,valor2:real

inicio

    leia(valor1)
    leia(valor2)
    resultado <- somar(valor1,valor2)
    escreva(resultado)

fimalgoritmo
  
```

The terminal window shows the following output:

```

25
25
50
*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
  
```

Exemplo 02 - Função

Nesse exemplo, nossa função recebe um valor como parâmetro “x” e retorna esse valor reajustado em 25% de aumento “valorReajustado”, quando a função é chamada passamos o valor 1000 e a função retorna o novo valor de 1250 exibido na tela dentro do comando `escreva()`. Mas poderíamos atribuir esse retorno a uma variável, por exemplo `y = reajuste(1000)` e depois mostrar a variável desta forma, `escreva(y)`, para o usuário o resultado seria o mesmo.

```

1 Algoritmo "Função_01"
2
3 // Professor : Flávio Mota
4 // Crie uma função que receba como parametro o valor de um produto
5 // e retorne esse valor com um aumento de 25%
6
7 funcao reajuste(x:real): real
8 var
9     valorReajustado:real
10    PercentualReajuste:real
11 inicio
12
13    PercentualReajuste <- (x * 0.25)
14    valorReajustado <- PercentualReajuste + x
15    retorne valorReajustado
16 fimfuncao
17
18 Var
19
20 Inicio
21     .
22     escreva(reajuste(1000))
23
24 Fimalgoritmo

```

Exemplo 03 – Procedimentos e Funções

Nesse exemplo vamos usar os conceitos de procedimentos e funções aprendidos em nosso curso, vamos dividir nosso algoritmo em módulos de leitura, processo e saída de dados de forma objetiva, usando procedimento e funções para o melhor entendimento de vocês.

Esse algoritmo tem um **procedimento** para a entrada de dois valores globais e um procedimento para a saída dos dados processados pelas funções, quatro **funções** que devolve o resultado das quatro operações básicas da matemática (soma, subtração, divisão e multiplicação) de forma individual com dois parâmetros de entrada para cada função.

```

1 Algoritmo "procedimento_função"
2
3 procedimento Leitura()
4 inicio
5     Escreva ("Digite o numero 1.:")
6     Leia(num1)
7     Escreva ("Digite o numero 2.:")
8     Leia(num2)
9 fimprocedimento
10
11 funcao somar(pn1, pn2:real):real
12 Inicio
13     soma <- pn1 + pn2
14     retorne soma
15 fimfuncao
16
17 funcao multiplicar(pn1, pn2:real):real
18 Inicio
19     multiplicacao <- pn1 * pn2
20     retorne multiplicacao
21 fimfuncao
22
23 funcao dividir(pn1, pn2:real):real
24 Inicio
25     divisao <- pn1 / pn2
26     retorne divisao
27 fimfuncao
28
29 funcao subtrair(pn1,pn2:real):real
30 Inicio
31     subtracao <- pn1 - pn2
32     retorne subtracao
33 fimfuncao
34
35 procedimento Saida()
36 inicio
37
38     Escreval ("A soma .: ",somar(num1, num2))
39     Escreval ("A multiplicacao .: ", multiplicar(num1, num2))
40     Escreval ("A subtração .: ", subtrair(num1, num2))
41     Escreval ("A divisão .: ", dividir(num1, num2))
42
43 fimprocedimento
44
45 // Algoritmo principal
46 var
47     // variaveis globais
48     num1, num2, soma, multiplicacao, divisao, subtracao: real
49 Inicio
50
51     Leitura() // chamada do procedimento Leitura()
52     Saida() // chamada do procedimento Saida()
53
54 fimalgoritmo
55

```

Exercícios no CADERNO 01: Algoritmos nível 05 – PROCEDIMENTOS E FUNÇÕES

Copyright © 2021 de **Flávio Mota da Cruz**

Todos os direitos reservados. Este ebook ou qualquer parte dele não pode ser reproduzido ou usado de forma alguma sem autorização expressa, por escrito, do autor ou editor, exceto pelo uso de citações breves em uma resenha do ebook.

Primeira edição, 2021

Entre em contato: fmotac@gmail.com

Para mais apostilas e cursos: <https://qikbyte.com.br/>