UNIASSELVI

GRADUAÇÃO E PÓS

PROFESSOR JOSÉ PAULO | TURMA FLC15770BES

UNIASSELVI

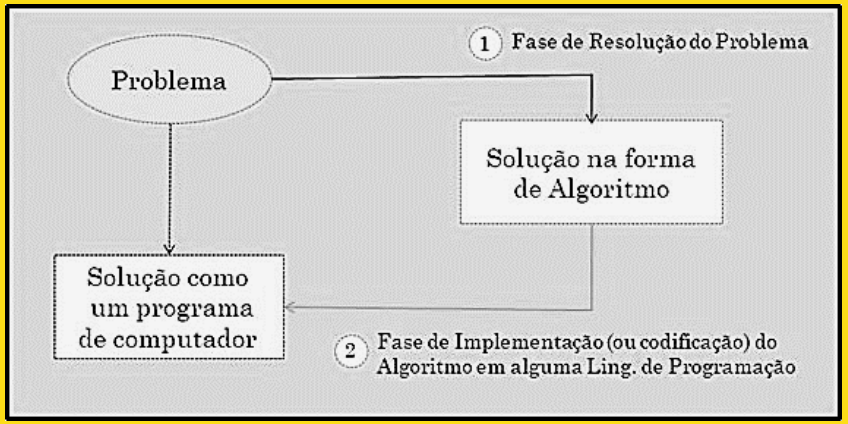
#ADistanciaImporta

A **distância nunca foi tão importante** em um momento como esse. A **UNIASSELVI**, referência na metodologia EAD semipresencial, ajustou o seu modelo de ensino para que **todas as aulas e provas sejam totalmente virtuais.** Assim você não precisa sair de casa para estudar. ***Com atitudes como essa, você colabora para a integridade da saúde de todos.***

Ah, e fique atento: quando a situação se resolver, voltaremos para o nosso modelo semipresencial – a metodologia mais indicada para uma educação de qualidade. ***#ADistânciaImporta***

LÓGICA E ALGORITMOS

- O algoritmo não é a solução de um problema, mas o meio para se chegar à forma mais adequada para a solução

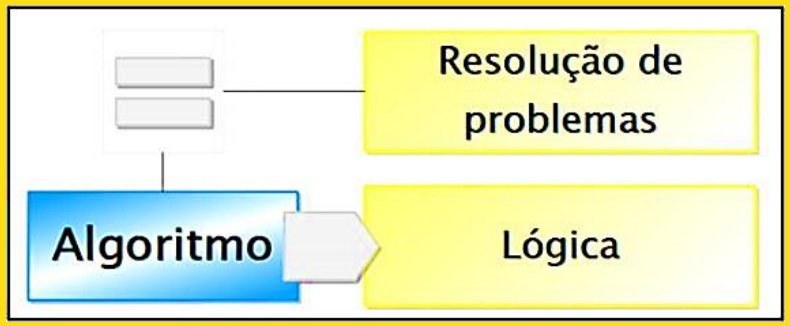


- A construção de programas é conhecida como ação de Programar e Programar é basicamente construir **Algoritmos**.
- **Algoritmo** é um conjunto das regras e procedimentos lógicos perfeitamente definidos que levam à solução de um problema em um número finito de etapas. Para criarmos o algoritmo devemos abstrair os procedimentos necessários.
- Abstrair é observar (um ou mais elementos de um todo), avaliando características e propriedades em separado.
- Para automatizar um procedimento manual, precisamos reproduzir seu comportamento na execução de um programa de computador. Identificar e reproduzir a Ação. Esse comportamento talvez seja a maior dificuldade na concepção e no entendimento do algoritmo.
- Uma ação é um evento que ocorre num período de tempo finito, estabelecendo um efeito intencionado e bem definido.
- Em programação as ordens são dadas no imperativo:
  - “Caminhe até a porta”
  - “Sente na Cadeira“
  - “Morda o cotovelo”

UNIASSELVI
GRADUAÇÃO E PÓS

LÓGICA E ALGORITMOS

- Para desenvolver adequadamente um algoritmo, é necessário usar a lógica, que consiste em organizar o pensamento para resolver problemas usando a mesma sequência adotada pelo computador, ou seja, usando a mesma lógica.

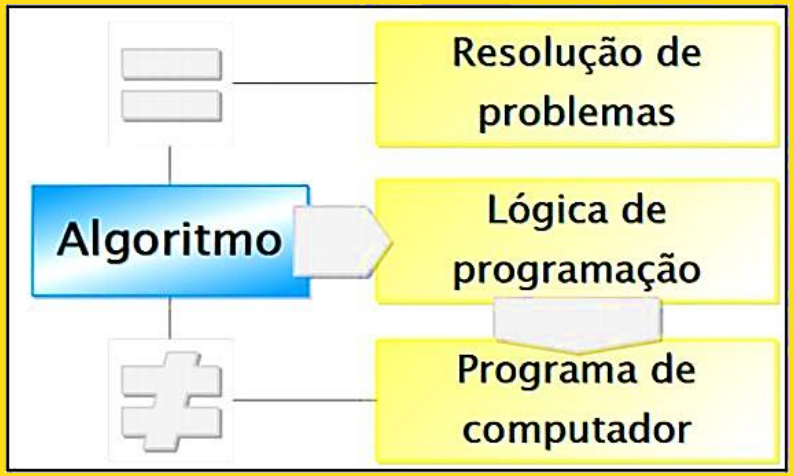


- Principais tipos de **lógica**
- **Lógica** aristotélica: tem como objeto de estudo o pensamento correto, assim como as leis e regras que o controlam. Para Aristóteles, os elementos constituintes da lógica são o conceito, juízo e raciocínio e as relações que existem entre eles.
- **Lógica** de argumentação: permite verificar a validade ou se um enunciado é verdadeiro ou não. São proposições tangíveis cuja validade pode ser verificada, por exemplo: O Fubá é um cachorro. Todos os cachorros são mamíferos. Logo, o fubá é um mamífero.
- **Lógica matemática** (ou lógica formal): estuda a lógica segundo a sua estrutura ou forma. A lógica matemática consiste em um sistema dedutivo de enunciados que tem como objetivo criar um grupo de leis e regras para determinar a validade dos raciocínios, que é considerado válido se for possível alcançar uma conclusão verdadeira a partir de premissas verdadeiras.
- **Lógica proposicional**: área da lógica que examina os raciocínios de acordo com as relações entre orações (proposições), as unidades mínimas do discurso, que podem ser verdadeiras ou falsas.
- **Lógica de programação**: é a estrutura usada para criar um programa de computador. A lógica de programação é essencial para desenvolver programas e sistemas informáticos, pois ela define o encadeamento lógico para esse desenvolvimento. Os passos para esse desenvolvimento são conhecidos como algoritmo, que consiste em uma sequência lógica de instruções para que a função seja executada.

UNIASSELVI
GRADUAÇÃO E PÓS

CARACTERÍSTICAS E FASES DE UM ALGORITMO

- Algoritmo é uma sequência de ações finitas encadeadas e lógicas que descrevem como um determinado problema deve ser resolvido.
- Um algoritmo é formalmente uma sequência finita de passos que levam à execução de uma tarefa.



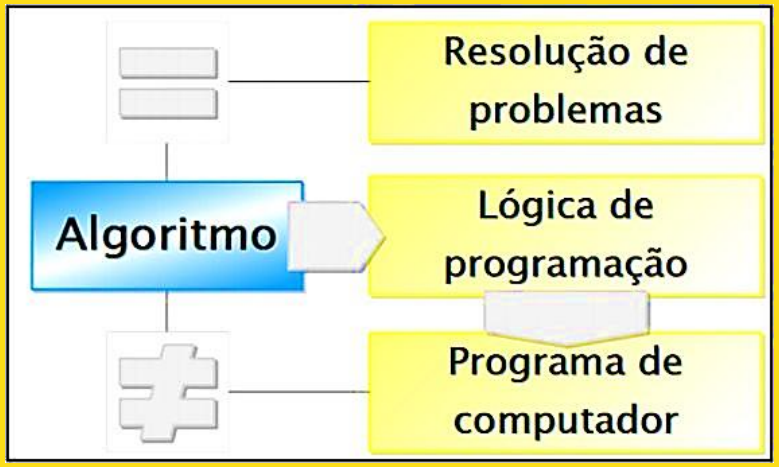
- Todo algoritmo precisa possuir as seguintes características:
  - Entrada: são as informações que alimentam a construção, geralmente usados como parâmetros ou filtros na busca das informações em uma base de dados. Um algoritmo pode não conter valores de entrada. Assim, como poderá apresentar um ou mais valores de tipos de dados distintos como entrada para a lógica construída.
  - Saída: todo algoritmo deve produzir um resultado.
  - Clareza ou definição: cada passo/instrução/etapa de um algoritmo deve ser claro e não gerar duplo entendimento.
  - Efetividade: cada passo/instrução/etapa de um algoritmo deve ser executável.
  - Finitude: o algoritmo deve ter uma condição para sair de sua execução. Isso evitará que entre em loop. O loop traduz a incapacidade do algoritmo de interromper a sua execução.

- A construção de um algoritmo apresenta três etapas distintas:
  - Entrada: são os dados que serão processados pelo algoritmo.
  - Processamento: representa os procedimentos necessários de manipulação das informações no intuito de produzir o resultado esperado.
  - Saída: é o resultado esperado; são os dados produzidos na etapa de processamento.

UNIASSELVI
GRADUAÇÃO E PÓS

MÉTODO PARA A CONSTRUÇÃO DE ALGORITMOS

- Várias são as práticas adotadas para a construção de algoritmos, com a adoção de algumas regras que precisam ser seguidas .



- Práticas adotadas, prioritariamente, para a construção de algoritmos:
  - Entender o problema a ser resolvido. Um exemplo de problema: somar dois números. Neste caso, imagine o que você precisa para executar a solução.
  - Identificar e definir as entradas do algoritmo. No caso do problema proposto, você teria como entrada dois números.
  - Descrever os passos para resolver o problema. Basicamente descrever o processo de soma dos dois números.
  - Definir os dados de saída. Na situação proposta, o resultado da soma dos dois números usados como entrada do problema.
  - Construir o algoritmo para representar a sequência de execução dos passos.
  - Transcrever o algoritmo para uma linguagem interpretada por computador.
  - Testar a lógica, bem como os passos de execução.
- Algumas regras que precisam ser seguidas:
  - Usar somente um verbo por passo/instrução/etapa.
  - Abusar da simplicidade e objetividade em relação aos termos e frases.
  - Escrever de uma forma simples para que possa ser entendido facilmente, inclusive por pessoas que não trabalham na área.
  - Evitar termos ou palavras que permitam duplo entendimento.

UNIASSELVI
GRADUAÇÃO E PÓS

As formas mais conhecidas para a representação de algoritmos são:

Descrição narrativa.
Fluxograma convencional.
Diagrama de Chapin.
Pseudocódigo, também conhecido como linguagem estruturada ou Portugol.

UNIASSELVI
GRADUAÇÃO E PÓS

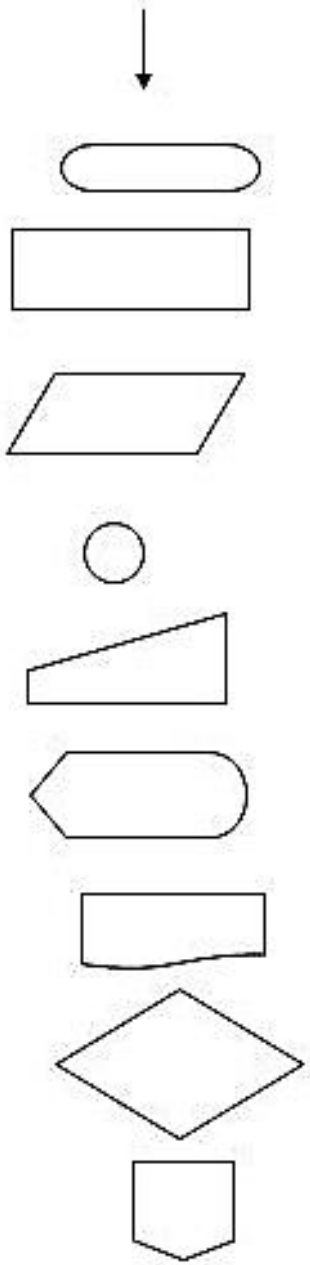
DESCRIÇÃO NARRATIVA

Ex: Algoritmo de como fazer uma laranjada

1) “Pegue três laranjas”
2) “Pegue uma faca”
3) “Corte cada uma laranja ao meio”
4) “Pegue um copo”
5) “Pegue o espremedor de laranjas”
6) “Coloque o espremedor sobre o copo”
7) “esprema cada parte da laranja de cada vez”
8) “retire o espremedor do copo”
9) “Pegue o vidro de água da geladeira”
10) “Coloque água no copo para completar”
11) “Pegue o Açúcar”
12) “Pegue uma colher de sopa”
13) “Coloque duas colheres de açúcar”
14) “Mexa até misturar completamente”

- Esta forma é conhecida como linguagem natural. É usada sempre quando se deseja que o receptor da mensagem entenda o que será feito, mesmo não tendo domínio acerca da elaboração de algoritmos.

FLUXOGRAMA CONVENCIONAL



FLUXO DE DADOS

Indica o sentido do fluxo de dados

Conecta os demais símbolos

TERMINAL

Indica o INICIO ou FIM de um processamento

Exemplo: Início do algoritmo

PROCESSAMENTO

Processamento em geral

Exemplo: Cálculo de dois números

ENTRADA/SAIDA (Genérica)

Operação de entrada e saída de dados

Exemplo: Leitura e Gravação de Arquivos

DESVIO (conector)

Permite o desvio para um ponto qualquer do programa

ENTRADA MANUAL

Indica entrada de dados via Teclado

Exemplo: Digite a nota da prova 1

EXIBIR

Exibe informações

Exemplo: Exiba informações do cálculo

SAIDA

Representa operação de saída dos dados

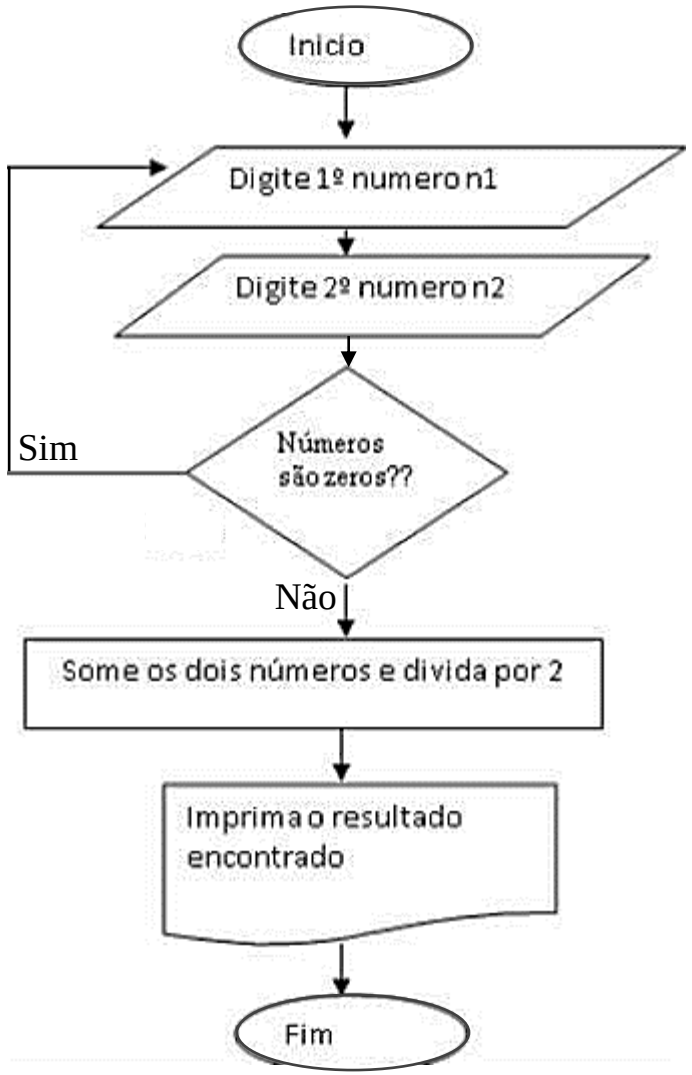
Exemplo: Imprima o relatório

DECISÃO

Permite elaborar processos de decisão

CONECTOR DE PAGINA

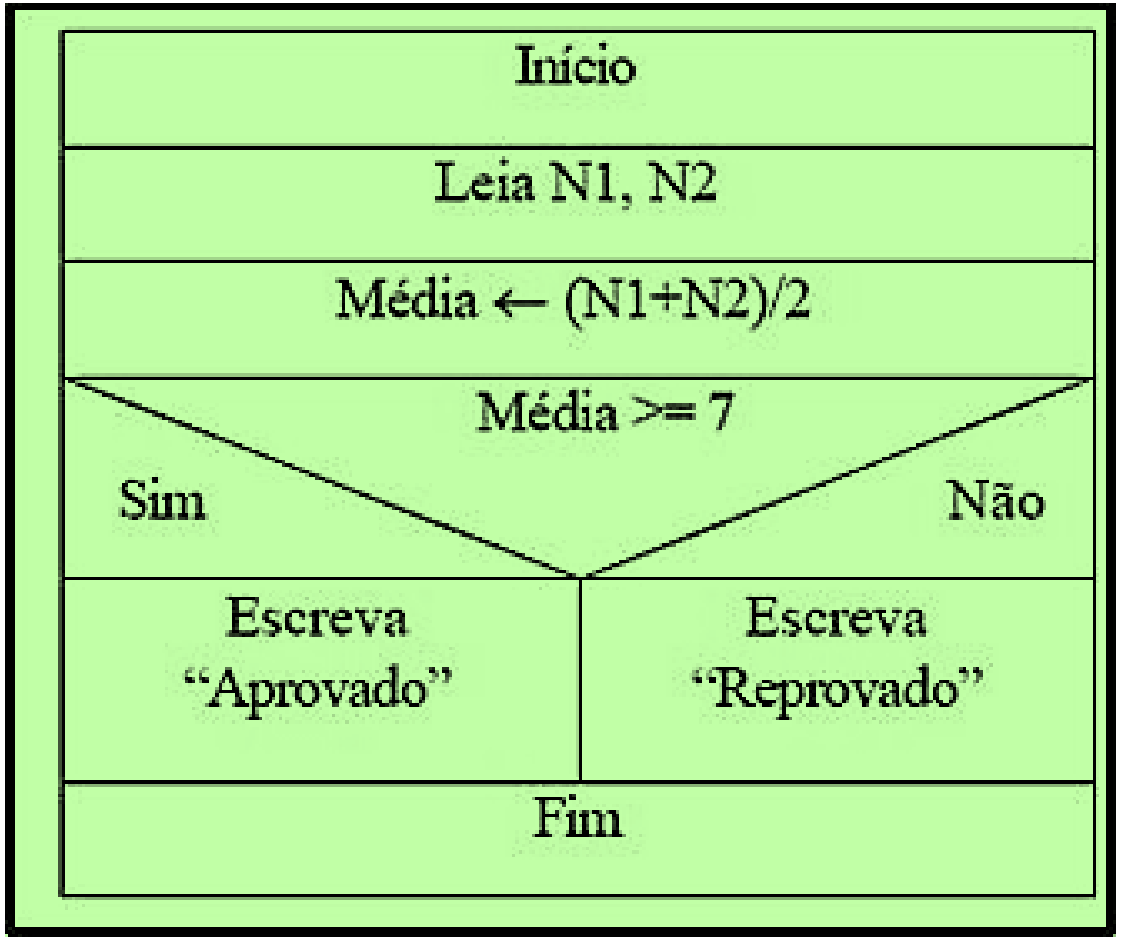
Permite informar de qual página vem o fluxograma



- O diagrama de blocos ou fluxograma é uma forma padronizada eficaz para representar os passos lógicos de um determinado processamento usando símbolos universais.

UNIASSELVI
GRADUAÇÃO E PÓS

DIAGRAMA DE CHAPIN



- Foi criado com a intenção de substituir os diagramas tradicionais. O objetivo era apresentar uma visão mais hierárquica e estruturada da lógica do sistema. A vantagem do uso consiste no fato de que é mais fácil representar as estruturas que tem um ponto de entrada e um ponto de saída e são compostas pelas estruturas básicas de controle de sequência, seleção e repartição.

PSEUDOCÓDIGO

```
Programa Nome_Prog;
Var
    real: nota_N1, nota_N2, media;
Inicio
    leia (nota_N1);
    leia (nota_N2);
    media := (nota_N1 + nota_N2)/2;
    Escreva (“A média é: “, media);
Fim
```

- O pseudocódigo ou portugol é uma representação muito parecida com a forma de escrita dos programas para a versão computacional. Esta técnica de algoritmização é baseada em uma PDL – Program Design Language (Linguagem de Projeto de Programação), muito parecida com a notação da linguagem PASCAL.
- Os algoritmos são independentes das linguagens de programação. Ao contrário de uma linguagem de programação, não existe um formalismo rígido de como deve ser escrito o algoritmo.

A variável é um “local” reservado na memória RAM que abriga um conteúdo tal que pode ter o seu valor mudado durante a execução do algoritmo.

A memória pode ser imaginada como uma “estante” com diversas gavetas, em que os dados e as instruções a serem executadas são armazenados temporariamente.

UNIASSELVI
GRADUAÇÃO E PÓS

DADOS E VARIÁVEIS

- Os tipos básicos de dados definem como e o que pode ser salvo (valor a ser atribuído) em cada variável e, também, as operações que podem ser executadas com eles.
- Tipos Primitivos de dados:
  - **Inteiro** – O tipo inteiro define qualquer número, positivo ou negativo, pertencente ao conjunto dos números inteiros. (..., -2, -1, 0, 1, 2, ...).
  - **Real** – O tipo real define qualquer número pertencente ao conjunto dos números reais. Também são chamados de ponto flutuante.
  - **Texto** – Pode armazenar elementos alfanuméricos, pode ser uma sequência de um ou mais caracteres, representados entre “ ” (aspas duplas) ou ‘ ’ (aspas simples), dependendo da linguagem. (a, b, casa, ricardo10, 11, @, $, ...).
  - **Lógico** – Pode armazenar apenas os valores verdadeiro ou falso (valores booleanos), representado apenas um bit (que aceita apenas 1 ou 0) .

DECLARAÇÃO DE VARIÁVEL

- Os nomes das variáveis devem ser formados por caracteres pertencentes ao seguinte conjunto: {A,B,C, ..., X,Y,W,Z,0,1,...,8,9, .,_} e o primeiro caractere deve sempre ser uma letra;
- O nome escolhido deve explicar seu conteúdo.
- Exemplo: Total_Alunos, Total_Pagar, MediaFinal.
- Declarar uma variável significa reservar uma “gaveta” na memória;
- As variáveis, a princípio, devem ser declaradas antes da sua utilização, no local específico para isto.

- Alguns exemplos de declaração de variáveis:
- Inteiro : X1;
- Real : A, B, Total;
- Caractere : Nome_Cliente, endereco;
- Logico : Possui_Filhos, X;
- Obs : Note as regras de sintaxe para a criação do nome da variável, os tipos de dados e o término com o “;”. Observe também as declarações múltiplas separadas com a “,”.

UNIASSELVI
GRADUAÇÃO E PÓS

DECLARAÇÃO DE CONSTANTES

- Constantes têm valores fixos ou estáveis. São declaradas na mesma seção das variáveis. Sua utilidade consiste no fato de que ela será declarada uma única vez com valor fixo. Caso seja necessário alterar o valor, esta mudança será feita na sua declaração, não sendo necessário alterar o código de programação que referencia a mesma, isto é, o valor de uma constante permanece o mesmo durante toda a execução do programa.

- Exemplo de declaração de constantes:
- CONST <nome_da_constante> = <valor>

- Exemplo de definição de constantes:
- CONST pi = 3.14159

ATRIBUIÇÃO DE VALORES

<variável> ← <valor>

Para atribuirmos valor a uma variável usamos o seguinte comando:

Total := 10; (Lê-se: Total recebe 10)

Total := X1; (Lê-se: Total recebe o conteúdo de X1)

A expressão localizada no lado direito do sinal do operador de atribuição é avaliada e armazenado o valor resultante na variável à esquerda. O nome da variável aparece sempre sozinho, no lado esquerdo do sinal do operador de atribuição deste comando.

OPERADORES ARITMÉTICOS

» Utilizados para efetuar equações matemáticas com as variáveis.

Símbolo	Função	Tipos Disponíveis
+	Adição	inteiro, real
-	Subtração	inteiro, real
*	Multiplicação	inteiro, real
/	Divisão	inteiro, real
** ou ^	Exponenciação	inteiro, real
MOD	Resto da divisão	inteiro, real
DIV	Quociente da divisão	inteiro, real

OPERADORES ARITMÉTICOS

- Exemplos de Operadores Aritméticos:
- B := A + 2;
- Total := Valor * 18;
- Soma := Total + Valor_Anterior;
- Imposto := Valor_Vendido * 0,03;
- X := Soma + (Imposto * 12)

- As prioridade de operações são as mesmas das regras matemáticas, ou seja:
  - Primeiro calcula-se exponenciações e a raízes;
  - Depois as Multiplicações e divisões;
  - Por último, as somas e subtrações.
  - Se há a necessidade de alterar a ordem deve-se utilizar o “( )” para priorizar operções;
- Veja: a equação B := 2 + 3 * 4; é diferente da equação B := (2 + 3) * 4;

OPERADORES RELACIONAIS

» Utilizados para efetuar comparações entre as variáveis.

Símbolo	Função	Tipos Disponíveis
<=	Menor ou igual	inteiro, real
<	Menor	inteiro, real
>=	Maior ou igual	inteiro, real
>	Maior	inteiro, real
<>	Diferente	Todos
=	Igual	Todos

OPERADORES RELACIONAIS

- Exemplos de Operadores Relacionais
  - C > 30;
  - Valor = 45;
  - Exemplo <= 13;
-
- Os operadores relacionais comparam valores e retornam simplesmente verdadeiro ou falso, dependendo da relação testada.

Expressão	Resultado Lógico
1 = 1	Verdadeiro
1 > 5	Falso
1 < 5	Verdadeiro
“teste” <> “teste”	Falso

OPERADORES LÓGICOS

- Utilizados para aprimorar as comparações efetuadas entre as variáveis.
- Eles são chamados de conectivos e servem para formar novas proposições a partir de outras anteriores.

Símbolo	Função	Tipos Disponíveis	Prioridade de operação
não	Negação	Lógico	1
e	Conjunção	Lógico	2
ou	Disjunção	Lógico	3

OPERADORES LÓGICOS

- Exemplos de Lógica Relacional:
  - O número 6 é par e é menor do que 8;
  - O aluno pode ser aprovado ou reprovado.
  - Não está chovendo.

- Na matemática:
  - 6 < 3 e 4 > 2;
  - 8 <> 4 ou 5 = 5;
  - não (9 = 3);

Expressão	Resultado Lógico
(1 = 1) E (1 = 2)	Falso
(1 = 1) E (2 = 2)	Verdadeiro
(1 = 1) OU (1 = 2)	Verdadeiro
(1 = 3) OU (1 = 2)	Falso
NÃO (1 = 1)	Falso

OPERADORES LITERAIS

- Os operadores que atuam sobre caracteres variam muito de uma linguagem para outra. O operador mais comum e mais usado é o operador que faz a concatenação de strings: toma-se duas strings e acrescenta-se (concatena-se) a segunda ao final da primeira.

- O operador que faz esta operação é: +

- Por exemplo, a concatenação das strings “ALGO” e “RITMO” é representada por:

  - “ALGO” + ”RITMO”

  - O resultado de sua avaliação é: “ALGORITMO”

INSTRUÇÕES PRIMITIVAS

- As instruções primitivas compõem os comandos básicos na funcionalidade dos computadores, como, por exemplo, a entrada e saída dos dados, fazendo a comunicação entre os usuários e a máquina.

- Conceitos importantes para a fixação:

  - Dispositivos de entrada: usado pelo usuário ou pela memória do computador para transferir informações **(comando LEIA)**. Exemplo: teclado, mouse, leitor de código de barras etc.
  - Dispositivo de saída: forma pela qual o computador transfere informações ao usuário **(comando ESCREVA)**. Ex: monitor de vídeo, impressoras etc.
  - **Sintaxe** é a forma como os comandos devem ser escritos, evitando erros de compilação dos programas. A sintaxe define a forma como deve ser escrito o código.
  - **Semântica** é o significado, ou seja, o conjunto de ações que serão exercidas pelo computador durante a execução do referido comando. A semântica é a lógica do conteúdo dessa escrita.

COMANDO DE DESVIO CONDICIONAL SIMPLES

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

- Exemplo SE-ENTÃO
- **se** <expressão-lógica> **entao**
- <sequência-de-comandos>
- **fimse**

UNIASSELVI
GRADUAÇÃO E PÓS

COMANDO DE DESVIO CONDICIONAL COMPOSTA

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

- Exemplo SE-ENTÃO
- **se** <expressão-lógica> **entao**
- <sequência-de-comandos-1>
- **senao**
- <sequência-de-comandos-2>
- **fimse**

UNIASSELVI
GRADUAÇÃO E PÓS

COMANDO DE DESVIO CONDICIONAL COMPOSTA

» Considerando um exemplo em que se calcula a média de um aluno, a solução em algoritmo seria construída assim:

SE (Média >= 5.0) **ENTAO**

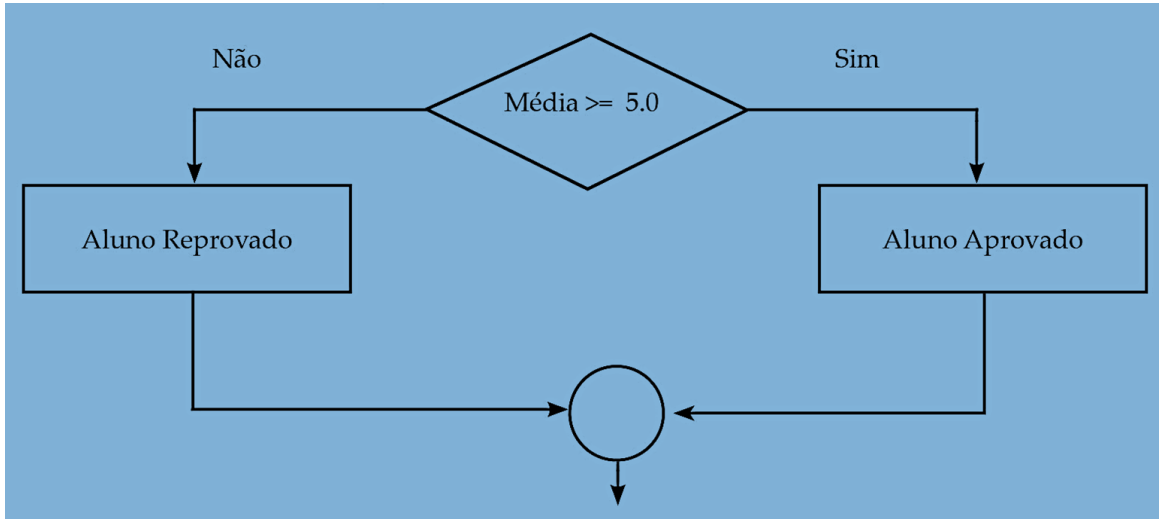
escreva ("aluno Aprovado“)

SENAO

escreva ("aluno Reprovado“)

FimSe

» Representação em fluxograma:



UNIASSELVI
GRADUAÇÃO E PÓS

DESVIO CONDICIONAL MÚLTIPLO – CASO

```
Programa caso_seja;
Var
     X : Inteiro;
  Inicio
    Escreva ('Escolha o canal da sua preferência:');
    Escreva ('Digite 1 para a Globo.');
    Escreva ('Digite 2 para a Record.');
    Escreva ('Digite 3 para a Band.');
    Escreva ('Digite 4 para o SBT.');
    Escreva ('Digite 0 para o Felipe Neto.');
    Leia (X);
    Caso X seja
        1 : Escreva ('Entrou no canal da Globo');
        2 : Escreva ('Entrou no canal da Record');
        3 : Escreva ('Entrou no canal da Band');
        4 : Escreva ('Entrou no canal do SBT');
        0 : Escreva ('Entrou no YouTube para ver Felipe Neto');
        Senao Escreva ('Entrou no Else. Não tinha Opção, né?');
    Fim;
  Fim.
```

- É usado para fazer um trabalho semelhante ao do comando IF, quando se pretende usar vários desvios condicionados à escolha do usuário.

UNIASSELVI
GRADUAÇÃO E PÓS

ESTRUTURAS DE REPETIÇÃO – ENQUANTO-FAÇA

- Escreva os números na tela de 0 a 10:

```
algoritmo "contador"
var
   contador:inteiro
inicio
// Seção de Comandos

   enquanto contador <=10 faca
              escreval(contador)
              contador <- contador + 1
   fimenquanto
fimalgoritmo
```

- Este comando repete seus comandos infinitamente até que uma condição seja satisfeita.

UNIASSELVI
GRADUAÇÃO E PÓS

ESTRUTURAS DE REPETIÇÃO ENQUANTO-FAÇA

```
algoritmo "notas"
var
   nota1, nota2, media: real
   aluno : caractere
   resposta : caractere
Inicio
      resposta <- "S"
      enquanto (resposta = "S") faca
         escreval("Digite o nome do aluno")
         leia(aluno)
         escreval("Digite a nota 1 ")
         leia(nota1)
         escreval("Digite a nota 2 ")
         leia(nota2)
         media <-(nota1 + nota2)/2
         se (media >=60) entao
            escreval("Aluno aprovado")
         senao
              escreval("Aluno reprovado")
         fimse
         escreval("Deseja a informar os dados de outro aluno? S ou N")
         leia(resposta)
      fimenquanto
fimalgoritmo
```

- Exemplo para o desenvolvimento de um algoritmo que lê duas notas de um aluno, calcula sua média e indica se foi aprovado ou reprovado. O algoritmo deverá ser executado até que o usuário diga que não temos mais alunos para avaliar.

ESTRUTURAS DE REPETIÇÃO – PARA-FAÇA

Exemplo de sintaxe:

```
Algoritmo Estrutura_de_repetição_para
    Para <variavel inicio> até <fim incremento> faça
        <bloco de comandos>
Fim para;
```

- Este comando repete seus comandos um número pré-determinado de vezes.

Exemplo de algoritmo:

```
Algoritmo repetiçco_para;
Var
   I = inteiro;
Inicio
   Para i de 1 até 100 faca
      Escreva (‘Seja otimista e vença na vida’);
   Fim para;
Fim
```

O resultado será a exibição de 100 linhas com a frase: Seja otimista e vença na vida.

UNIASSELVI
GRADUAÇÃO E PÓS

CONTADORES

- Os contadores são normalmente inicializados com valor 0 (zero) e incrementados em 1 (um) a cada vez que uma nova ocorrência (ou situação) é processada.
- Na expressão “para i de 1 ate 10 faca” i é o contador.

- O contador também pode vir dentro da estrutura de uma repetição enquanto, como um acumulador, por exemplo:
- Enquanto x <= i faca
  - Escreva (“Eu programo pra caramba!”);
  - i=i+1;
- Fim

ACUMULADORES

- Algoritmo Acumulador

```
Nome  pagamento;
Var
   Salario: real;
   Soma_salario: real
Inicio
   Soma_salario := 0;
    Para i de 1 ate 18 faca
       Escreva (‘digite o nome da pessoa’);
       Leia (nome);
       Escreva (‘digite o salario da pessoa’);
       Leia(salario);
       Soma_salario := soma_salario + salario;
    Fim para;
    Escreva (‘ A soma de todos os salarios é:’, soma_salario);
 Fim;
```

- Os acumuladores são utilizados em situações em que é necessário acumular uma soma. No caso dos somatórios, o acumulador é normalmente inicializado com o valor 0 e incrementado no valor de um outro termo qualquer, dependendo do problema em questão.

UNIASSELVI
GRADUAÇÃO E PÓS

Muito Obrigado!
Vejo vocês na próxima segunda-feira!

e-mail:
professorjosepauloviana@gmail.com

UNIASSELVI
GRADUAÇÃO E PÓS