ALGORITMOS

Introdução à Programação

BSI – DEINFO - UFRPE

Elementos de um Algoritmo

- Dados (variáveis e constantes)
- Tipos de dados
- Operadores
- Comandos
- Funções
- Comentários

Exemplo

```
algoritmo "perímetro_circunferência"
// declaração de variáveis
Var
perim, raio: real
inicio
    leia (raio)
    perim <- 2* PI* raio
    escreval ("o perímetro é:", perim)
fimalgoritmo
```

comentário

variáveis

tipo de uma variável

operador

função

Constantes e Variáveis

- Constantes
  - O valor de uma constante não se altera após sua definição.
  - Exemplos:
    - const_pi ← 3,1415

- Variáveis
  - Elemento que têm a função de associar um nome a uma área da memória onde um dado pode ser armazenado

Tipos

- Definem:
  - a **natureza** do dado
  - as **operações** que podem ser realizadas com o dado
  - O **espaço** a ser ocupado na memória
- Declarações:
  - **a, b, c, maior: real**
  - **x,y: inteiro**
  - **achou: logico**
  - **texto: caractere**

Tipos

- Tabela de tipos Visualg:

Tipo	Descrição
inteiro	Representa valores inteiros. Exemplos: 10, 5, -5, -10
real ou numerico	Representa valores reais (com ponto separador da parte decimal). Exemplos: 10, 15.5, -14.67
literal ou caractere	Representa texto (seqüência ou cadeia de caracteres) entre aspas duplas. Exemplo "Esta é uma cadeia de caracteres", “B”, “1234”
logico	Representa valores lógicos (VERDADEIRO ou FALSO)

Tipos

- Exemplos:
  - Inteiro: 10 -5 -128
  - Real (ponto flutuante): 1.34 13.4 -5.0
  - String de caracteres: “quarta-feira” Abril”
  - Lógico: VERDADEIRO (1)
    FALSO (0)

Entrada e Saída

- Algoritmos objetivam transformar informações
  - Algoritmo = **Entrada** + **Processamento** + **Saída**
- **Entrada**: obtenção de dados provenientes do meio externo
  - Comando: **leia**
  - Exemplos:
    - **leia** (X)
    - **leia** (A, NOTA)

Entrada e Saída

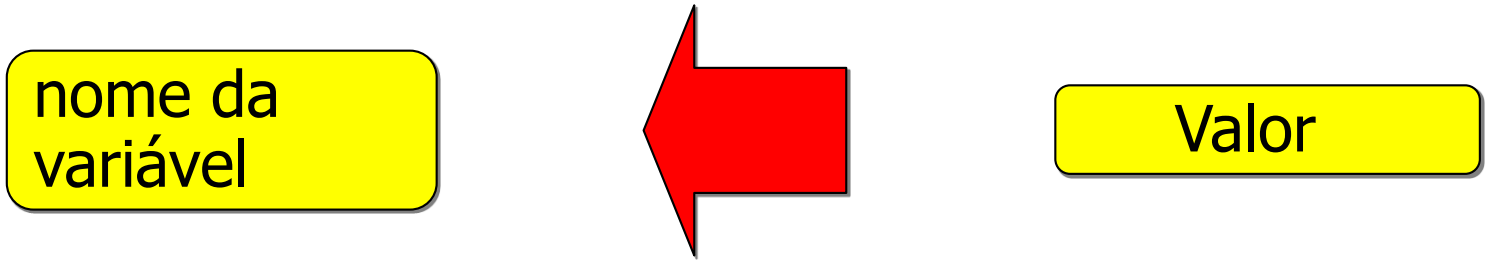
- **Saída**: entrega dos resultados ao meio externo
- Comandos: **escreva** ou **escreval**
- Exemplos:
  - **escreva** (X)
  - **escreva** (B, MEDIA, 2+2)
  - **escreval** (“cliente cadastrado com sucesso”)

Operadores

- **Atribuição**
- **Aritméticos**
- **Relacionais**
- **Lógicos**

Operador de Atribuição

- Utilizado para atribuir um valor a uma variável (“=“ ou “:=“ ou “← ”):



- Visualg: **“:=“ ou “← ”**
- Notação:

```
x1    ←  23;
temp  ←  x1;
nome  ←  “Carlos da Silva”;
```

Operadores Aritméticos

- **Dados de entrada**: tipo numérico (inteiro ou real)
- **Resultado**: tipo numérico (inteiro ou real)

- Exemplos:
  - x_2 ← 2 + 3;
  - x_3 ← 3 / 2;
  - alfa ← 1 div 5;
  - resto ← 10 % 3;
  - resto ← 1 % 4;
  - delta ← 5*5–4*1*4;

Operadores Aritméticos

Operação	Operador
Adição	**+**
Subtração	**-**
Multiplicação	*****
Divisão	**/**
Divisão Inteira	**\ ou div**
Exponenciação	**^ ou Exp (<base>,<expoente>)**
Resto da Divisão	**%**

Operadores Relacionais

- **Dados de entrada**: tipo numérico (int ou float) ou caracteres
- **Resultado**: tipo lógico

Operador	Operação
a < b	**a** é menor que **b**
a <= b	**a** é menor ou igual a **b**
a > b	**a** é maior que **b**
a >= b	**a** é maior ou igual a **b**
a = b	**a** é igual a **b**
a <> b	**a** não é igual a **b**

Operadores Relacionais

- Exemplos:
  - cond1 ← 2 = 3 // (falso)
  - cond2 ← 1.6 <> 5.0 // (verdadeiro)
  - cond3 ← 1 > 5 // (falso)
  - cond4 ← (1 + 2) < 5 // (verdadeiro)
  - cond5 ← 10 >= 3 // (verdadeiro)
  - cond6 ← 1 <= 4 // (verdadeiro)
  - cond7 ← “café” < “expresso” // (verdadeiro)
  - cond8 ← “café” =“café” // (verdadeiro)
  - cond9 ← “café” >= “mocha” // (falso)

Operadores Lógicos

- **Dados de entrada**: tipo lógico
- **Resultado**: tipo lógico
- E (AND), OU (OR), NAO (NOT)

Operação	Resultado
a E b	VERDADEIRO se ambas as partes (**a** e **b**) forem verdadeiras
a OU b	VERDADEIRO se apenas uma das partes (**a** ou **b**) é verdadeira.
NAO a	Nega uma afirmação, invertendo o seu valor lógico: se **a** for VERDADEIRO retorna FALSO, se **a** for FALSO retorna VERDADEIRO.

Operadores Lógicos

- Exemplos:
  - cond1 ← verdadeiro E falso // (falso)
  - cond2 ← falso OU falso //(falso)
  - cond3 ← NAO cond1 // (verdadeiro)
  - cond4 ← (verdadeiro E falso) OU (5 > 3) // (verdadeiro)

Tabela Verdade

a	b	a E b	a OU b	NAO a	NAO b
V	V	V	V	F	F
V	F	F	V	F	V
F	V	F	V	V	F
F	F	F	F	V	V

Prioridade dos Operadores

Operador Aritmético	Prioridade
Exponenciação	3 (maior)
Multiplicação	2
Divisão	2
Adição	1
Subtração	1 (menor)

Operador Lógico	Prioridade
E	3
OU	2
NAO	1

Funções

- Podem ser pré-definidas da linguagem ou definidas pelo programador

- Exemplos:
  - **seno(angulo)**
  - **exp(x,y)**
  - **ler(var1,var2,...)**
  - **escrever(resul1,result2,...)**

Funções Pré-definidas

Função	Descrição
Abs (valor : real) : real	Valor absoluto
Asc (s : caracter) : inteiro	Retorna o código ASCII
Compr (c : caracter) : inteiro	Retorna o tamanho da cadeia de caracteres
Cos (valor : real) : real	Cosseno
Exp (<base>,<expoente>)	Exponenciação
Int (valor : real) : inteiro	Converte o valor em inteiro
Log (valor : real) : real	Logaritmo de base 10
Pi : real	A constante PI
Sen (valor : real) : real	Seno
Raizq (valor : real) : real	Raiz quadrada
Quad (valor : real) : real	Elevado quadrado

Pressionando (CTRL+J) o visualg mostra uma Lista de funções predefinidas

Comentários

- Utilizados para descrever texto esclarecendo trechos do código
  - **# (Python) ou**
  - ***//* (C++, Visualg) ou**
  - ***/** (Java)**

COMANDOS

Forma Geral do Algoritmo

- Algoritmo <Nome do Algoritmo>

```
  <declaração_de_variáveis>
Início
  <lista_de_comandos>
  Fim
```

Forma Geral

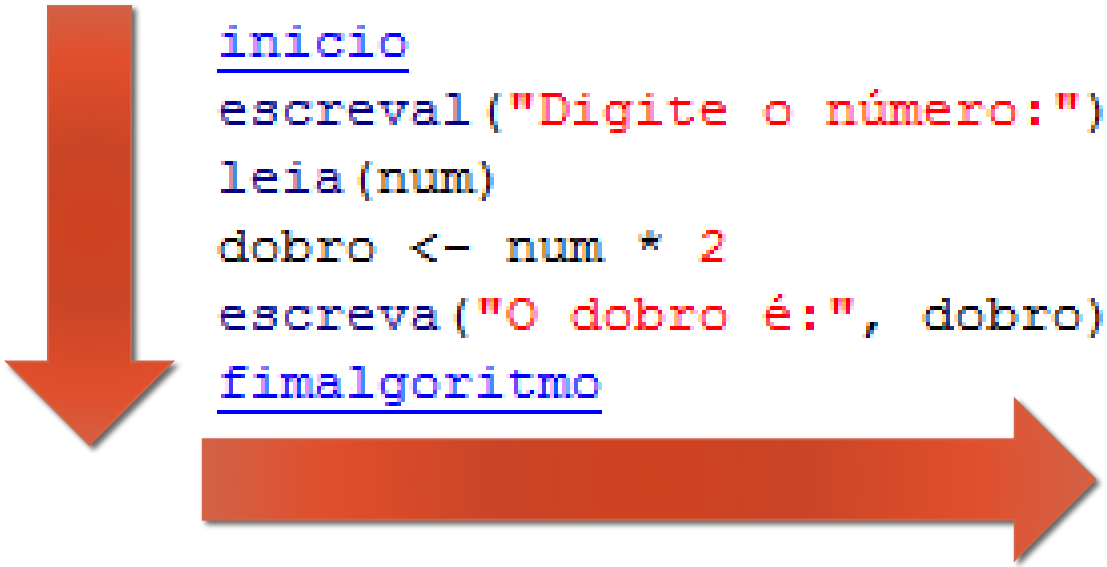
```
algoritmo "dobro"

// Seção de Declarações
var
num, dobro :inteiro

// comandos
inicio
escreval("Digite o número:")
leia(num)
dobro <- num * 2
escreva("O dobro é:", dobro)
fimalgoritmo
```

Iniciando em algoritmos...

- De forma genérica, a construção de um algoritmo se resume às seguintes **etapas**:
  1. **entendimento do problema**
  2. **elaboração da solução algorítmica**
  3. **codificação da solução em pseudo-código ou linguagem de programação**
- Geralmente a etapa 2 é a mais complexa

Estruturas Sequenciais

- O fluxo de controle segue a mesma seqüência
- linear da nossa escrita, ou seja:
  - De cima para baixo;
  - Da esquerda para direita



```
inicio
escreval("Digite o número:")
leia(num)
dobro <- num * 2
escreva("O dobro é:", dobro)
fimalgoritmo
```

Exemplo

- Enunciado: Fazer um programa que leia dois valores reais, e calcule e exiba a média aritmética

- Uma boa prática seria decompor o problema em problemas menores!! “Dividir para conquistar”

Exemplo

- **Passo 1**
  - Qual a fórmula da média?
  - A média aritmética de dois valores é calculada como (a+b)/2
- **Passo 2**
  - Os dados necessários serão os dois valores, que colocaremos em duas variáveis A e B do tipo real, e uma terceira variável, que chamaremos MEDIA, para armazenar a média aritmética calculada.

Exemplo

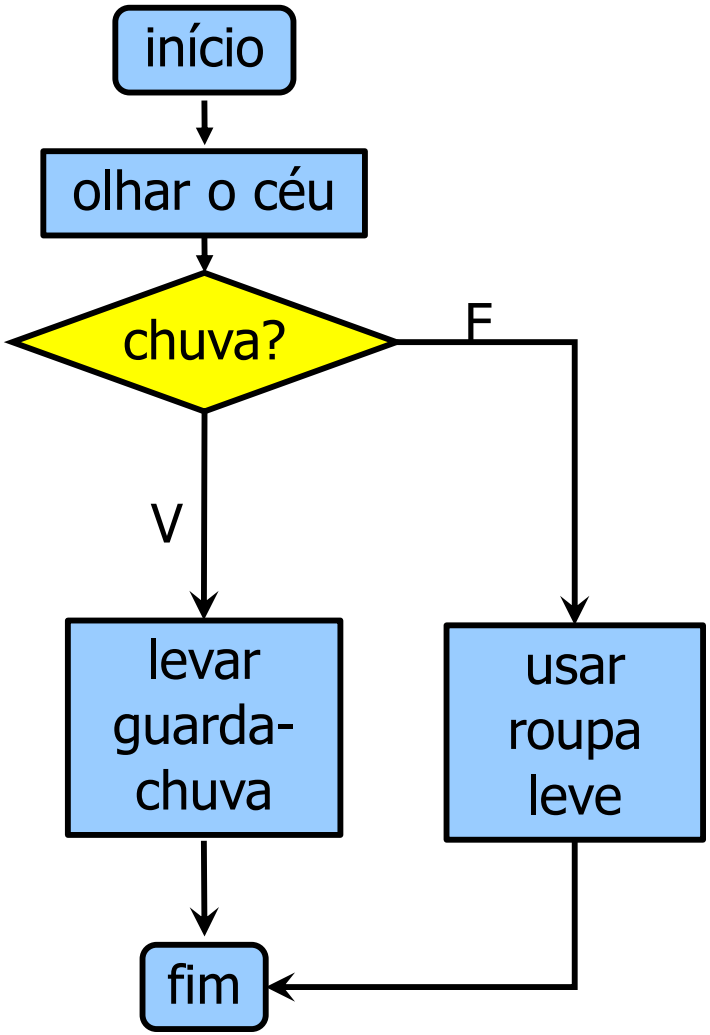
- **Passo 3**
  - A entrada dos dados neste programa é simples e direta.
  - Pedir ao usuário que digite os dois valores
- **Passo 4**
  - O processamento é o cálculo da média, usando a fórmula mencionada na etapa 1.
  - O resultado do cálculo será armazenado na variável MEDIA.

Estruturas Sequenciais
Exemplo

```
Algoritmo "Cálculo de Média Aritmética"
var
A,B,Media : REAL
inicio
      escreval ("Programa que calcula a média aritmética de dois valores.")
      escreval ("Digite um valor : ")
      leia (A)
      escreval ("Digite outro valor : ")
      leia (B)
      media <- (A+B)/2
      escreval ("A média dos dois valores é : ", Media)
fimalgoritmo
```

Estrutura Condicional

- Execução seletiva ou condicional
  - **Depende da avaliação da condição**
- Permite a escolha de um grupo de ações (bloco), quando certas condições são (ou não são) satisfeitas.

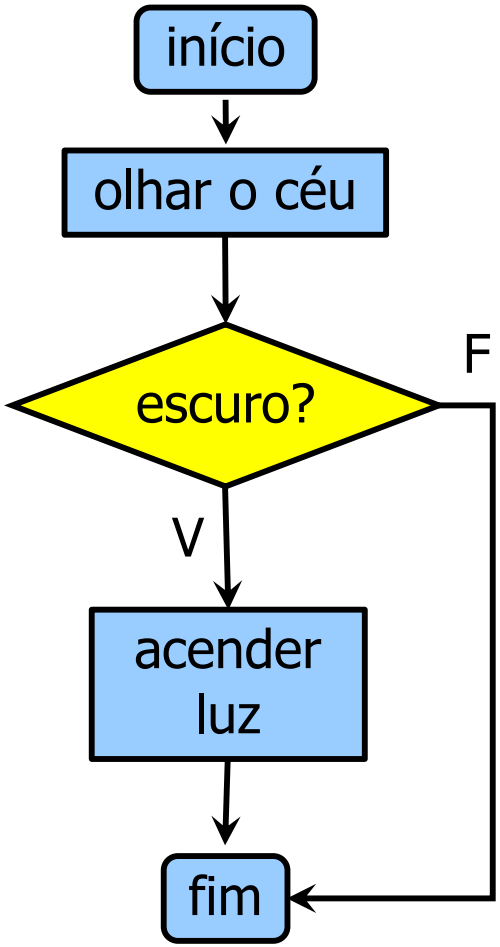


Estrutura Condicional

- **Simples**

- **Composta**

Estrutura Condicional Simples



Estrutura Condicional Simples

- Utilizada quando precisamos testar uma certa condição antes de executar uma ação

```
se <condição> entao
    <bloco de ações>
fimse
```

Exemplo

```
Algoritmo "Media"
var
   // declaração de variáveis:
   N1, N2, NF, media : real
inicio
   // início do programa
   leia(N1,N2,NF)

   media ← (N1 + N2 + NF) / 3.0


   se (media ≥ 7.0) entao
       escreva("Aluno aprovado")
   fimse

fimalgoritmo
```

Exercício

- Crie um algoritmo que leia um valor e mostre esse valor caso ele seja maior que zero.

Exercício

- Crie um algoritmo que leia um valor e mostre esse valor caso ele seja maior que zero.

```
Algoritmo "MostraMaiorZero"
var
   // declaração de variáveis:
   num : inteiro
inicio
   // início do programa:
   leia(num)

   se (num > 0) entao
    escreva("numero maior que zero:", num)
   fimse

fimalgoritmo
```

Estrutura Condicional Composta

- Utilizada em situações em que duas alternativas dependem da mesma condição, uma da condição verdadeira (então) e a outra da condição falsa (senão).

```
se <condição> entao
   <bloco de ações1>
senão
   <bloco de ações2>
fimse
```

Exemplo

```
Algoritmo "Media2"
var
   // declaração de variáveis:
   N1, N2, NF, media : real
inicio
   // início do programa:
   leia(N1,N2,NF)

   media ← (N1 + N2 + NF) / 3.0


   se (media ≥ 5.0) entao
       escreva("Aluno aprovado")
   senão
       escreva("Aluno reprovado")
   fimse

fimalgoritmo
```

Exercício

- Crie um algoritmo que leia dois números inteiros e calcule a diferença entre eles.

- A diferença é o maior número menos o menor

Exercício

- Crie um algoritmo que leia dois números inteiros e calcule a diferença entre eles.

- A diferença é o maior número menos o menor

```
Algoritmo "Dif2Numeros"
var
   // declaração de variáveis:
   N1, N2, Dif : inteiro
inicio
   // início do programa:
   leia(N1,N2)

   se N1 > N2 entao
       Dif ← N1 - N2
   senao
       Dif ← N2 - N1
   fimse

   escreva("a diferença é", Dif)

fimalgoritmo
```

Estrutura Condicional Composta

```
Algoritmo Maior3Numeros
var
  N1, N2, N3 : real
inicio
  leia(N1,N2,N3)

  se N1 ≥ N2 e N1 ≥ N3 entao
   escreva(N1, "é o maior")
  senao
   se N2 ≥ N1 e N2 ≥ N3 entao
      escrever(N2, "é o maior")
   senao
      escrever(N3, "é o maior")
   fimse
  fimse
fimalgoritmo
```

Estruturas de Repetição

- Permitem que uma **sequência de comandos** seja executada **repetidamente**, até que determinada **condição de interrupção** seja satisfeita
- São também conhecidas como **`laços`** (loop) ou malhas
- Cada **`repetição do bloco`** de comandos é chamada **`iteração`**

Estruturas de Repetição

- A repetição de comandos em um laço pode seguir um dos seguintes critérios:
  - Por Condição (**Verificação no Início**)
  - Por Condição (**Verificação no Fim**)
  - **Por Contagem**

Verificação no Início

- Os comandos do bloco de ações são executados **enquanto** uma condição é atendida (**verdadeira**)

```
enquanto <condição> faca
  <ação 1>
  <ação 2>
  ...
  <ação n>
fimenquanto
```

Verificação no Início
Exemplo

- Dado o valor de N, calcular a soma dos números inteiros de 1 a N

```
...
soma ← 0
i ← 1
enquanto (i ≤ N) faca
   soma ← soma + i
   i ← i + 1
fimenquanto
escreval("a soma é:", soma)
```

Verificação no Fim

- Os comandos do bloco de ações são executados **até que** uma condição seja atendida (**verdadeira**)

```
repita
   <ação 1>
   <ação 2>
   ...
   <ação n>
ate <condição>
```

Verificação no Fim
Exemplo

- Dado o valor de N, calcular a soma dos números inteiros de 1 a N

```
...
soma ← 0
i ← 1
repita
   soma ← soma + i
   i ← i + 1
ate (i > N)
escreval("a soma é:", soma)
```

Verificação no Fim × Verificação no Início

Verificação no início

- Condição é verificada antes do conjunto de instruções

Verificação no fim

- O conjunto de instruções será executado pelo menos uma vez
- Condição é verificada depois do conjunto de instruções

Repetição por Contagem

- Permite que comandos sejam repetidos um determinado número de vezes.

```
para  variavel de início ate
fim passo <incremento> faca
   <ação 1>
   <ação 2>
      ...
   <ação n>
fimpara
```

Repetição por contagem

- **`início`**: indica a **variável de controle** do laço (contador) e seu valor inicial.

- **`fim`**: define o valor final da variável de controle

- **`incremento`**: define como a variável de controle se altera **a cada repetição**

Repetição por Contagem
Exemplo

- Dado o valor de N, calcular a soma dos números inteiros de 1 a N

```
...
soma ← 0
para i de 1 ate N passo 1
faca
   soma ← soma + i
fimpara
escreval("a soma é:", soma)
...
```

Repetição por Contagem
Exemplo

- Algoritmo que lê e escreve os números ímpares de 1 a 1000.

```
para  i de 1 ate 1000 passo 2 faca
   escreval (i, " é ímpar")
fimpara
```

Exercícios

- Faça um algoritmo que some os múltiplos de 5, de 0 até 100.
  - Faça um agoritmo usando repetição por verificação no início e outro algoritmo usando repetição no final.

Agradecimento

- Slides adaptados a partir dos originais preparados pelo prof. Leandro Galvão (galvao@dcc.ufam.edu.br)