

Algoritmos – Introdução



Introdução à Ciência da Computação

Rosane Minghim

Apoio na confecção: Carlos Elias A. Zampieri

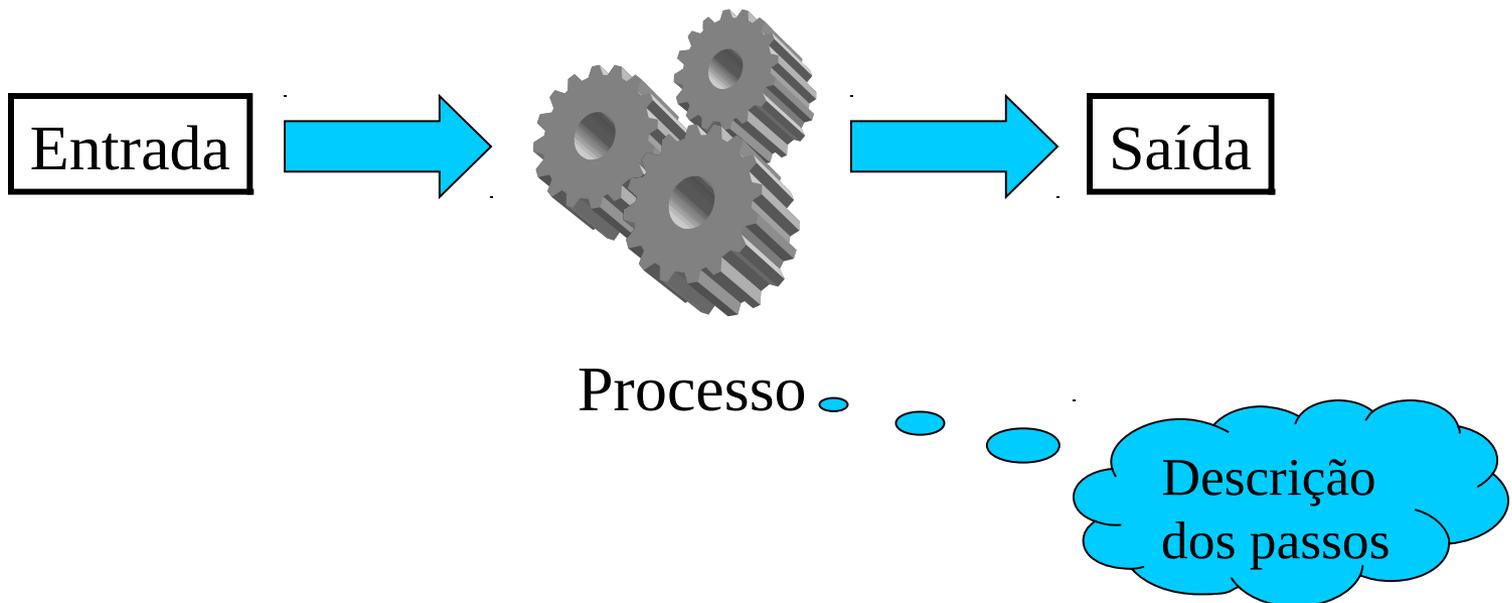
Danilo Medeiros Eler

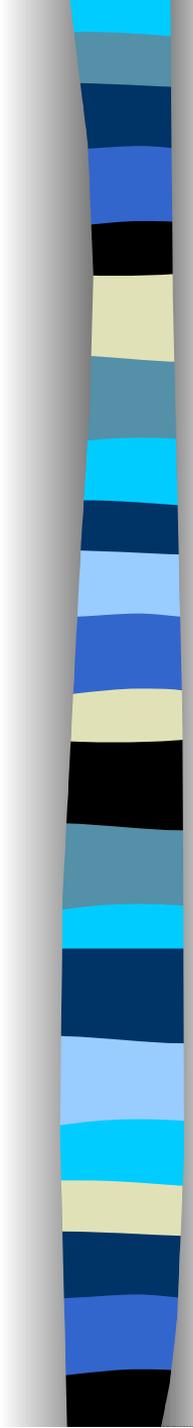
Renato Rodrigues

Rogério Eduardo Garcia

Algoritmo

- ***Sequência de passos para a execução de uma tarefa***
 - Ex: Receita de Bolo

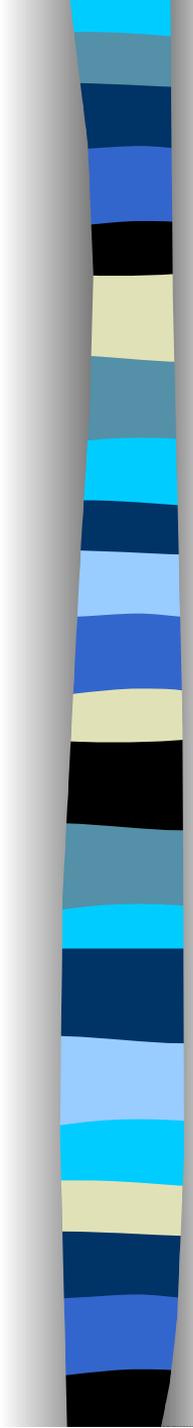




Exemplo: *Bolinhos de Chuva*

***Em uma tigela, bata o açúcar, a manteiga e o ovo.
Em outro recipiente misture a farinha, o fermento, a
canela, uma pitada de sal, o leite e a outra mistura.
Misture bem.***

***Aqueça óleo e pingue colheradas da massa,
fritando os bolinhos até dourar.
Escorra bem, polvilhe açúcar e sirva.***



Detalhamento do Algoritmo

Faça uma massa da maneira tradicional com o açúcar, a manteiga, o ovo a farinha, o fermento, a canela e uma pitada de sal. Frite os bolinhos e polvilhe com açúcar.

Em uma tigela, bata o açúcar e a manteiga. Depois, adicione um ovo.

Em outro recipiente, peneire a farinha e adicione o fermento, a canela e uma pitada de sal. Vá juntando essa mistura da farinha à outra mistura, alternando com leite. Misture bem. Em uma panela, aqueça uns 5 centímetros de óleo. Quando o óleo estiver quente, pingue colheradas da massa e frite os bolinhos até dourar. Escorra bem, polvilhe açúcar e sirva.

Detalhamento do Algoritmo

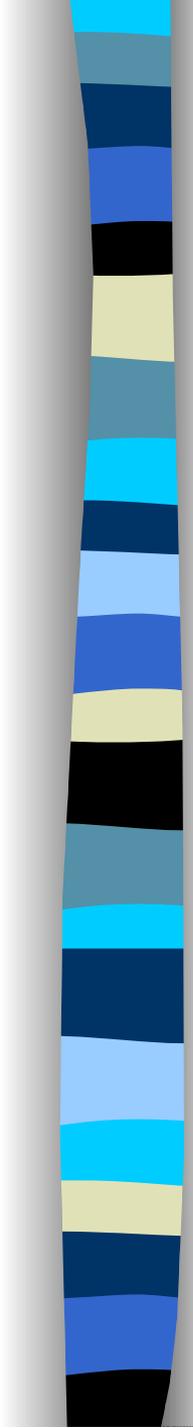
Faça uma massa da maneira tradicional com o açúcar, a manteiga, o ovo a farinha, o fermento, a canela e uma pitada de sal. Frite os bolinhos e polvilhe com açúcar.

Em uma tigela, bata o açúcar e a manteiga. Depois, adicione um ovo.

Em outro recipiente, peneire a farinha e adicione o fermento, a canela e uma pitada de sal. Vá juntando essa mistura da farinha à outra mistura, alternando com leite. Misture bem. Em uma panela, aqueça

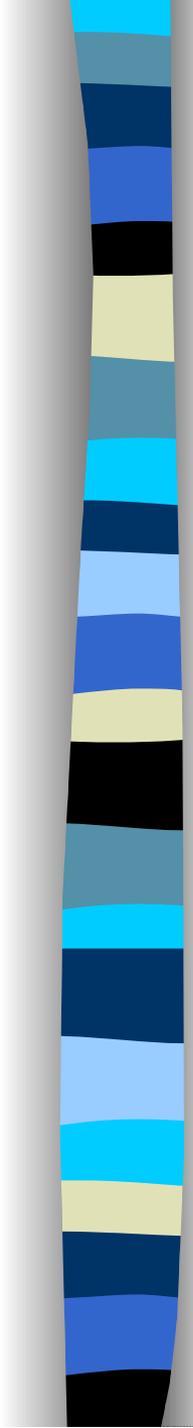
Nível de Detalhe: pode ser diferente, dependendo, por exemplo, da experiência da cozinheira que está lendo a receita... Em nosso caso, o computador deve ser capaz de executar a tarefa!

Escorra bem, polvilhe açúcar e sirva.



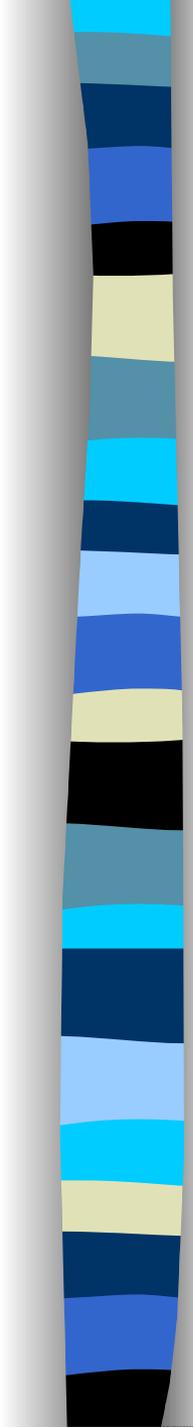
Algoritmos Computacionais

- O **computador** deve **executar** a tarefa
- Precisamos de uma linguagem de programação para construir um programa executável
- É preciso **transformar** a ideia da tarefa (receita) em um programa



Linguagens de Programação

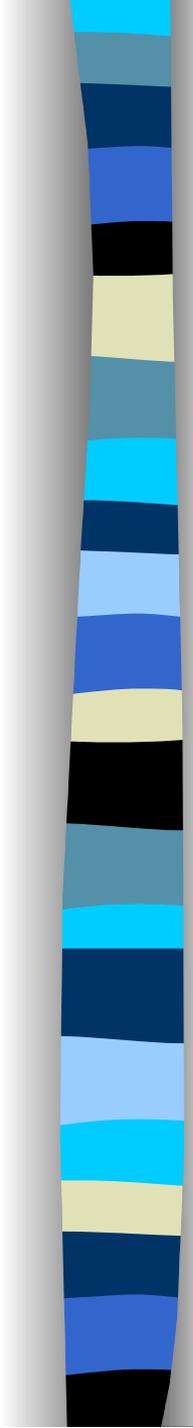
- As **operações** são **limitadas** a um pequeno conjunto
- A forma de escrever um algoritmo, sua **sintaxe**, deve seguir um certo padrão bem definido
- A **entrada** de dados e os dados que o programa manipula deve ser bem especificados



Linguagens de Programação

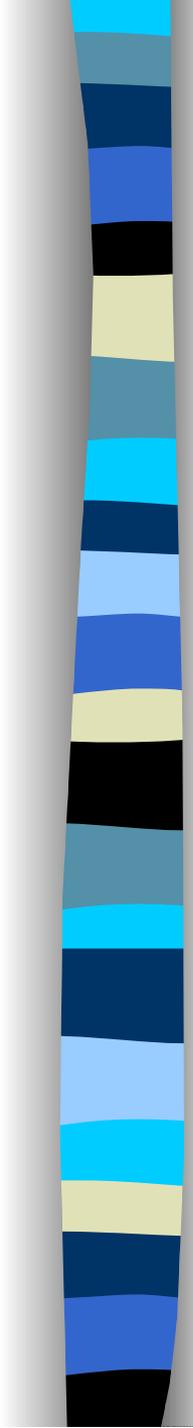
- C é uma linguagem **estruturada**, assim como Python*, Pascal, Modula 2, Perl e outras.
- Ao invés de estudar C ou outra linguagem diretamente, vamos definir uma **linguagem padrão** para construir algoritmos computacionais chamada de **pseudo-código**
- Vamos usar esse pseudo-código para apresentar os conceitos comuns às linguagens estruturadas

* É também uma linguagem orientada a objetos, assim como C++ e Java



Vantagens do Pseudo-Código

- **Sintaxe** mais **flexível** que a de uma linguagem de programação real
 - Permite que pensemos nos passos que o algoritmo computacional deve descrever sem nos preocuparmos demais com a forma de escrevê-los
- Ênfase nas ideias, e **não** nos **detalhes**



Vantagens do Pseudo-Código

- Poderemos construir um programa em uma linguagem estruturada com facilidade se tivermos um algoritmo em pseudo-código estruturado adequadamente
 - Os elementos do pseudo-código são os mesmos das linguagens estruturadas. Isto é, depois de desenvolver as ideias, a tradução para linguagem de programação é um processo simples e mecânico

Passos de um Programa

Algoritmo Raízes

Sejam a , b e c os coeficientes da equação do segundo grau

Calcule delta

Se delta for negativo, imprima a mensagem "não há raízes reais"

Se delta for positivo, calcule as raízes e imprima

fim

Um Programa em Pseudo-Código

Algoritmo Raízes

{Algoritmo para calcular as raízes reais de uma equação do segundo grau}

variável

a, b, c: real

delta: real

x1, x2: real

leia(a, b, c)

*delta ← b*b - 4*a*c*

se delta < 0 então

escreva('Esta equação não possui raízes reais.')

senão

*x1 ← (-1*b - raiz(delta, 2)) / (2*a)*

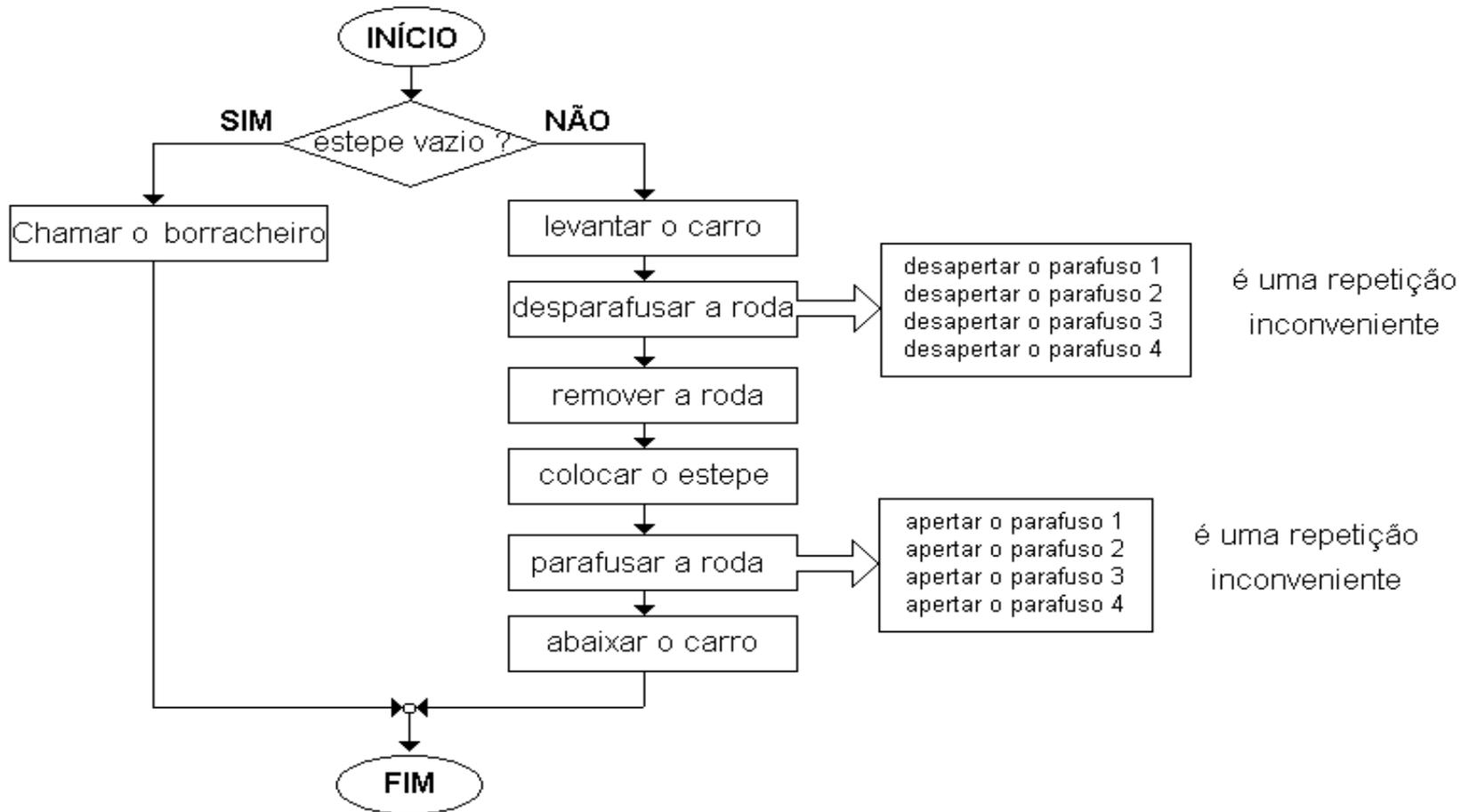
*x2 ← (-1*b + raiz(delta, 2)) / (2*a)*

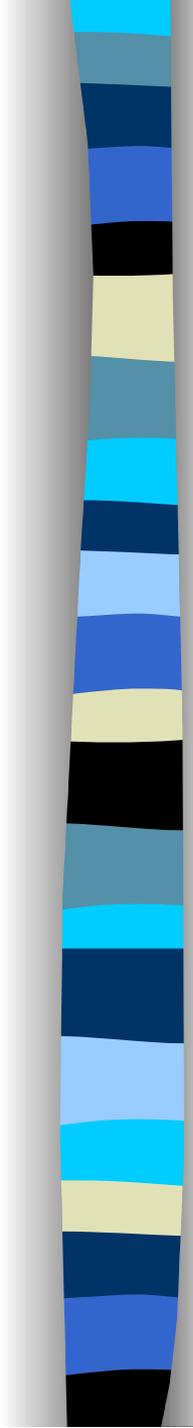
escreva('As raízes são ', x1, ' e ', x2)

fim se

fim

Outra representação





Elementos Básicos de um Algoritmo

- Um algoritmos deve expressar os principais elementos de um programa
- Os principais elementos são
 - Dados (constantes e variáveis)
 - Tipo de dados
 - Operadores
 - Comandos
 - Funções
 - Comentários

Algoritmo

```
Algoritmo <Identificador>
```

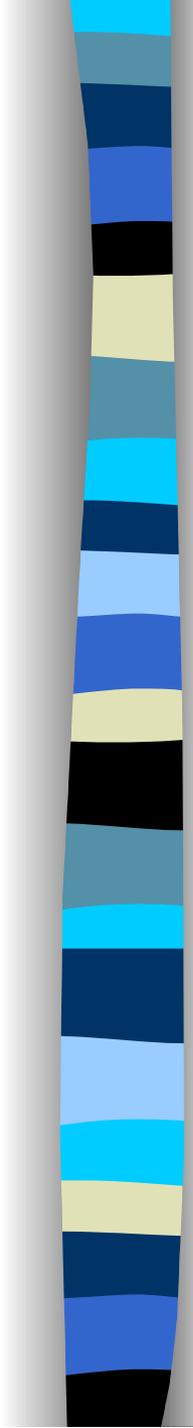
```
  <Declarações>
```

```
  <Comandos>
```

```
fim
```

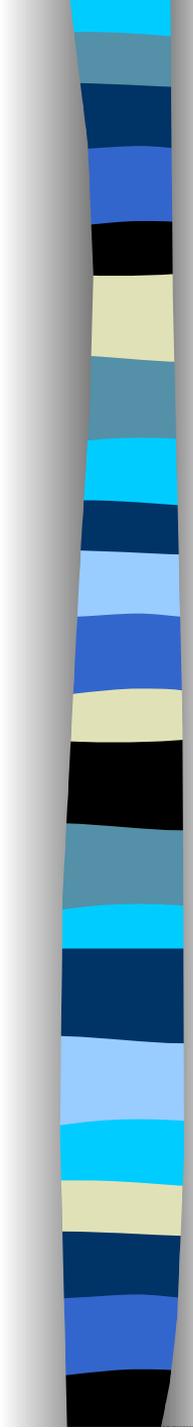


Constantes
Tipos e
Variáveis



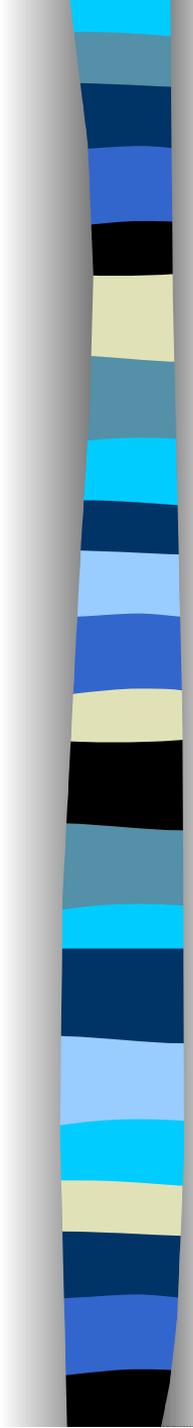
Constantes Literais

- Uma **constante** é um dado que aparece **literalmente** em um algoritmo
- Números, valores lógicos, letras, palavras e frases podem ser expressos como constantes em um algoritmo
- Exemplos:
 - 6,45
 - 'h'
 - 21
 - 'segunda-feira'



Identificadores

- Vários elementos de um algoritmo podem ser identificados através de um **nome**. Este nome é chamado de **identificador**.
- Em pseudo-código um identificador é uma **única palavra** com qualquer quantidade de letras, letras acentuadas, dígitos e símbolos que **não** sejam **operadores ou comandos**.



Identificadores

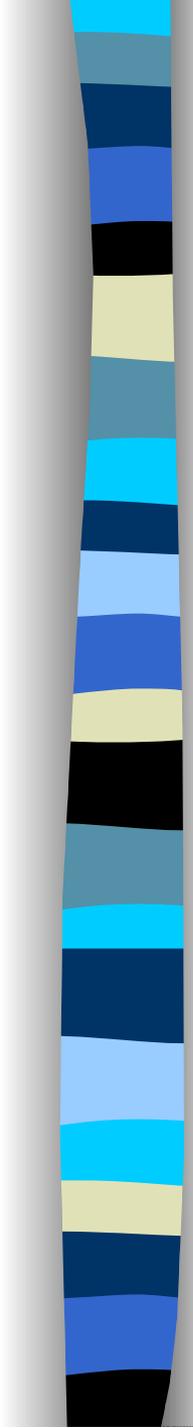
- Exemplos:
 - Nome, idade1, preço, preço_de_fábrica, kW
- Operadores e comandos têm sentido por si mesmo, por isso não devem ser usados;
- Não há **diferenciação** entre minúsculas e maiúsculas*:
 - Nome, NOME, nome

* **Atenção:** Em C **HÁ** diferença entre maiúsculas e minúsculas. Recomendação: estabeleça um padrão e **NUNCA** repita nomes.



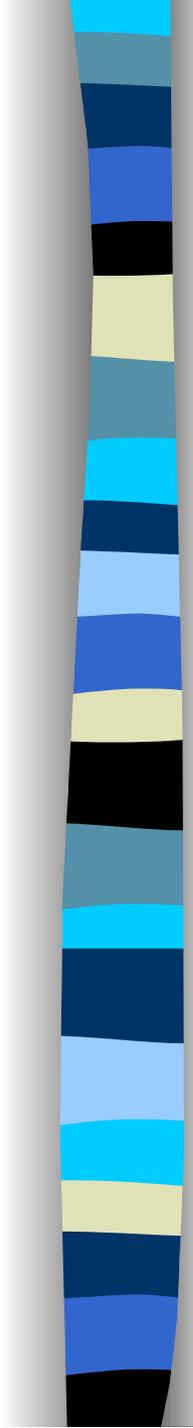
Dados e Tipos de Dados

- Um **dado** é uma informação que um algoritmo recebe ou manipula
- Exemplos de dados são nomes, datas, valores (preços, notas, coeficientes, quantidades, etc.) e condições (verdadeiro e falso). Mas há outros, muitos outros, principalmente coleções desses.



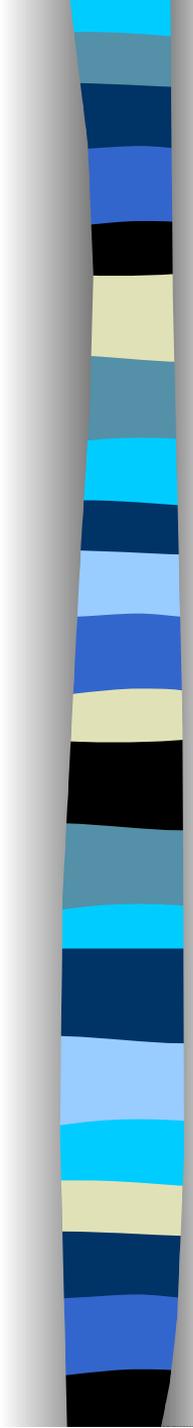
Dados e Tipos de Dados

- Todo dado é de um certo **tipo** que define sua natureza (p. ex., um nome é diferente de um valor), identificando seu uso, e define as operações que podem ser realizadas com o dado.
- Por exemplo, podemos somar dois valores numéricos, ou concatenar duas frases, mas não podemos somar um número a uma frase, ou somar duas frases, ou concatenar dois números.



Dados e Tipos de Dados

- Os tipos de dados mais **básicos** em algoritmos são o caractere, o numérico, o lógico e a enumeração.
- Tipos de dados básicos podem ser **estruturados** em tipos mais complexos;
 - Por exemplo, palavras e frases são construídas a partir de caracteres, podemos ter números complexo.



Tipos de Dados: Numérico

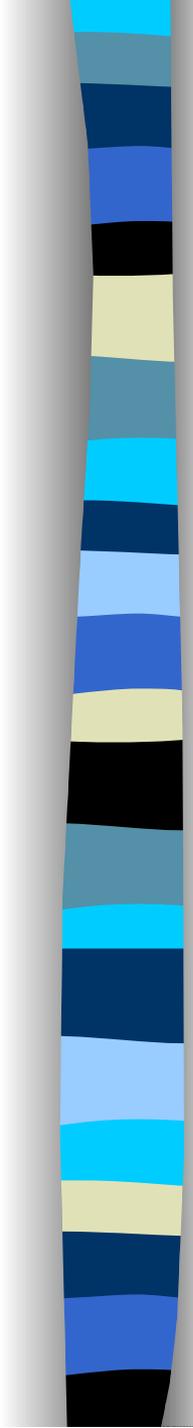
- **Inteiro**: representa um número inteiro. Por exemplo -1, 0, 1, e 26 são dados inteiros.
- Dados deste tipo podem ser usados para idade em anos, número de filhos etc.



Tipos de Dados: Numérico

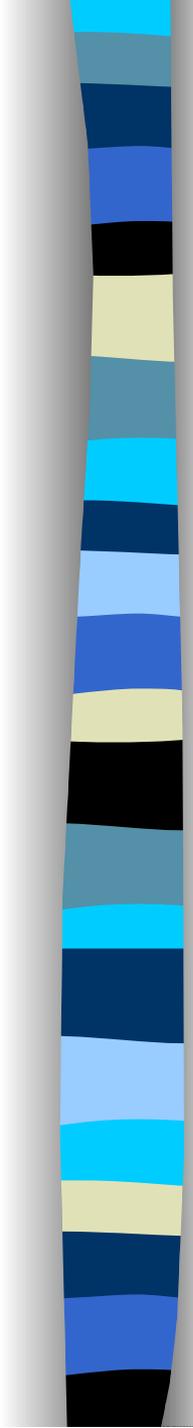
- **Ponto flutuante:** também chamado real, representa um número real. Por exemplo 1,2; 0,0; 26,4 e -2,49 são dados reais*
- Dados deste tipo podem ser usados para saldo bancário, altura, peso, temperatura, etc

* **Atenção:** Em C a notação para delimitação de casas decimais é determinada pelo uso do “.” (**ponto**). Exemplo: 1.2; 0.0; 26.4 e -2.49.



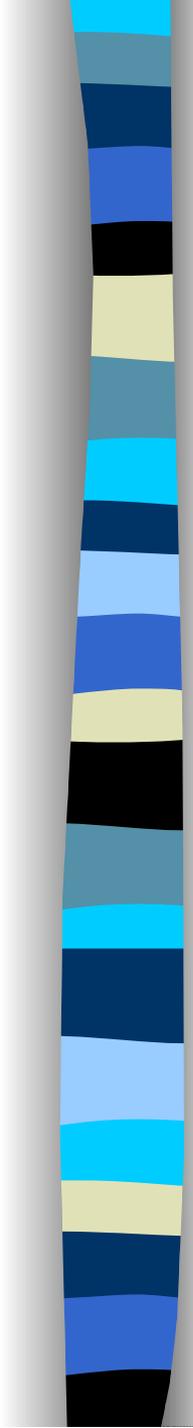
Tipos de Dados: Numérico

- No projeto de um algoritmo devemos utilizar o **tipo** numérico **mais adequado**, ou seja, não devemos usar um número real quando um número inteiro resolve o problema.
- Devemos **balancear as necessidades** do problema com a economia de espaço de armazenamento.



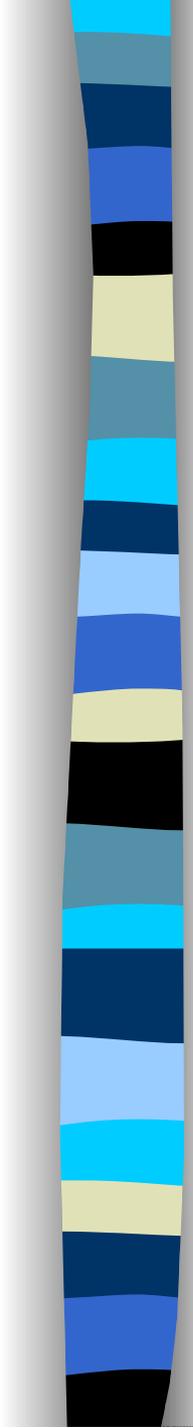
Tipos de Dados: Caractere

- Dados que representam valores alfanuméricos unitários são do tipo caractere
 - Por exemplo, 'A', 'a', '*'
- Caracteres podem ser usados para a codificação de algum item, como sexo ('m', 'f', 'o'), estado civil ('s', 'c', 'd', 'v', 'u'), estado de funcionamento (ligado/desligado), etc..



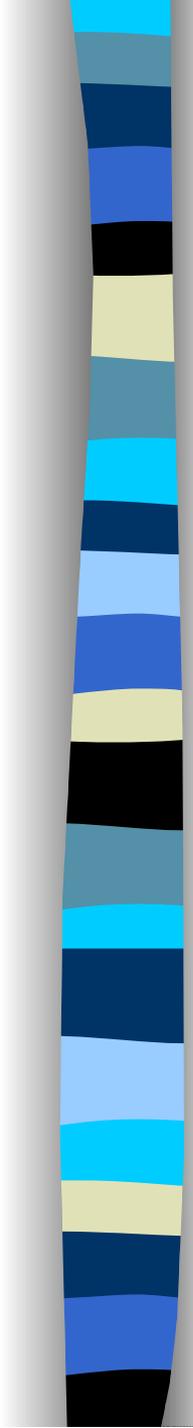
Tipos de Dados: Caractere

- Valores **alfanuméricos** incluem letras, algarismos e símbolos.
- Por exemplo, '1' é um caractere se consideramos apenas o símbolo '1' e não o valor 1.



Tipos de Dados: Lógico

- Dados **lógicos** podem assumir apenas dois valores: **VERDADEIRO** OU **FALSO**.
- São usados para expressar uma **condição**:
 - o fato de que $4 > 5$ é falso ou
 - se o cheque número 00425 já foi compensado ou não

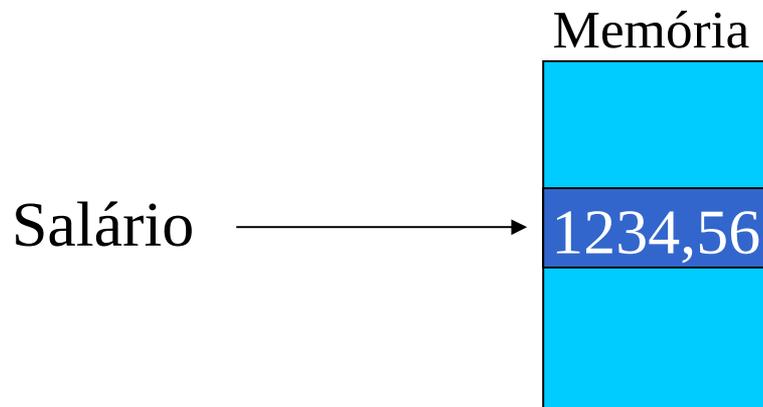


Tipos de Dados: Enumeração

- Um dado que pode assumir um valor dentre os valores de um conjunto é uma enumeração ou tipo enumerado
- Por exemplo, um dado que pode assumir qualquer valor dentro do conjunto de frutas
 - {banana, maçã, pêra, uva, jaca}

Variáveis

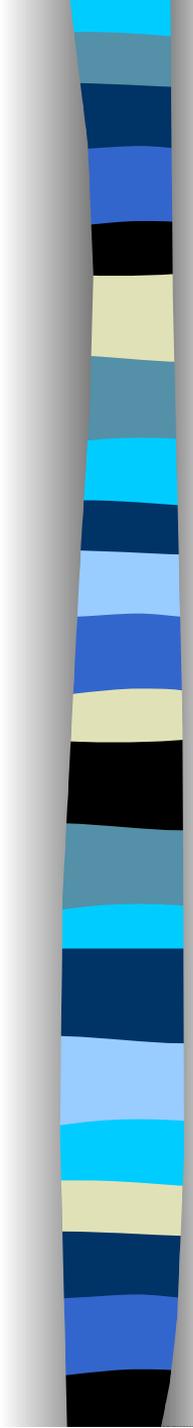
- Uma variável é um elemento de algoritmos que tem a função de **associar um nome a uma porção da memória** onde um dado pode ser armazenado



Variáveis

- A variável possui, além do nome, um **tipo**, responsável por definir como o dado vai ser armazenado e recuperado da memória.
- Em pseudo-código as variáveis são declaradas na seção de declarações, antes da seção de comandos, na cláusula **variável**

```
variável  
salário: real
```

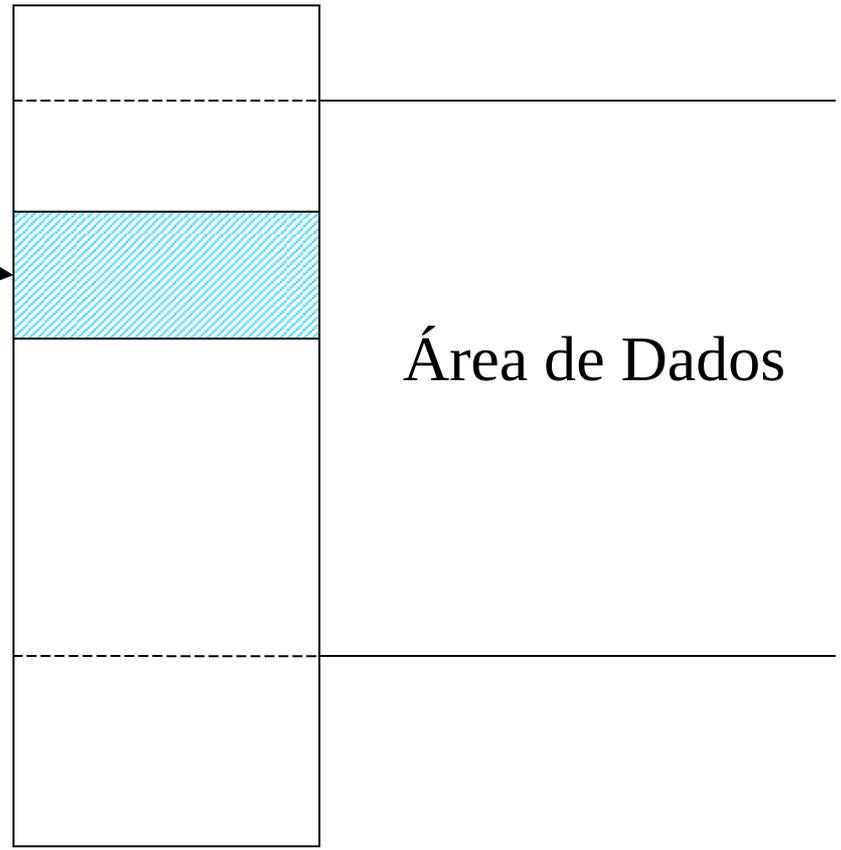


Atribuição

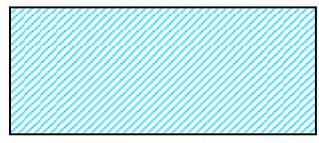
- Pode-se atribuir uma dado a uma variável pelo operador ' \leftarrow '
- Exemplos:
 - idade \leftarrow 51
 - válido \leftarrow VERDADEIRO
 - sexo \leftarrow 'f'
 - idade_mínima \leftarrow idade

Memória

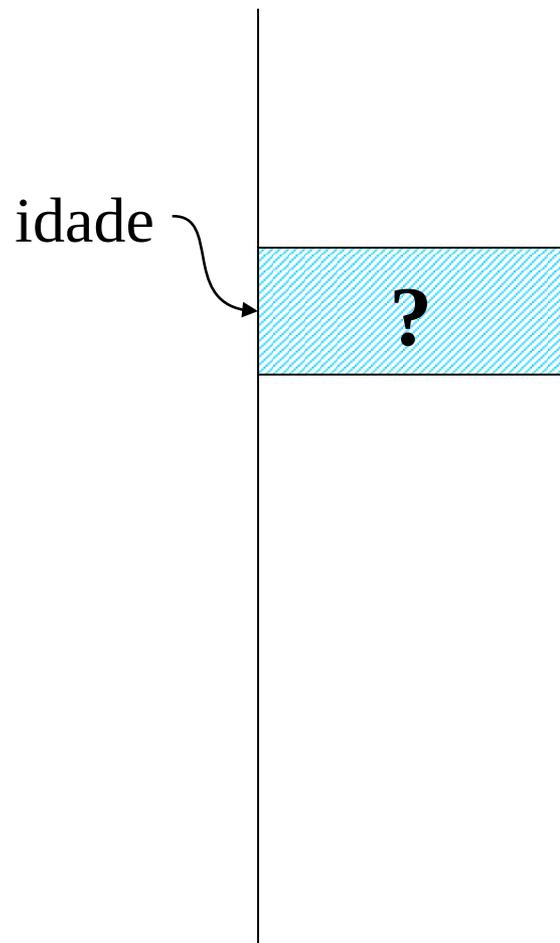
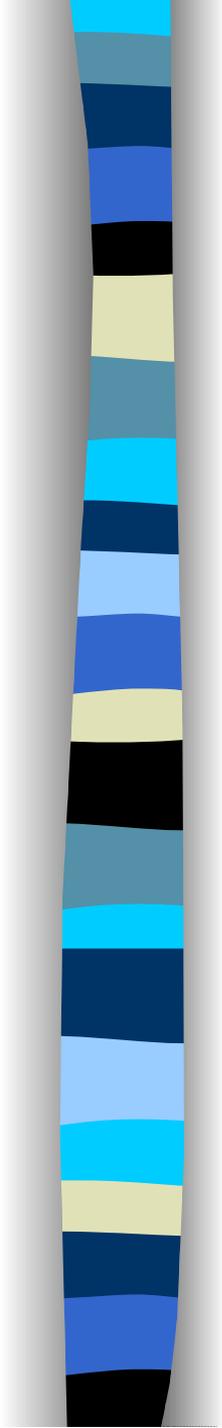
idade



Área de Dados

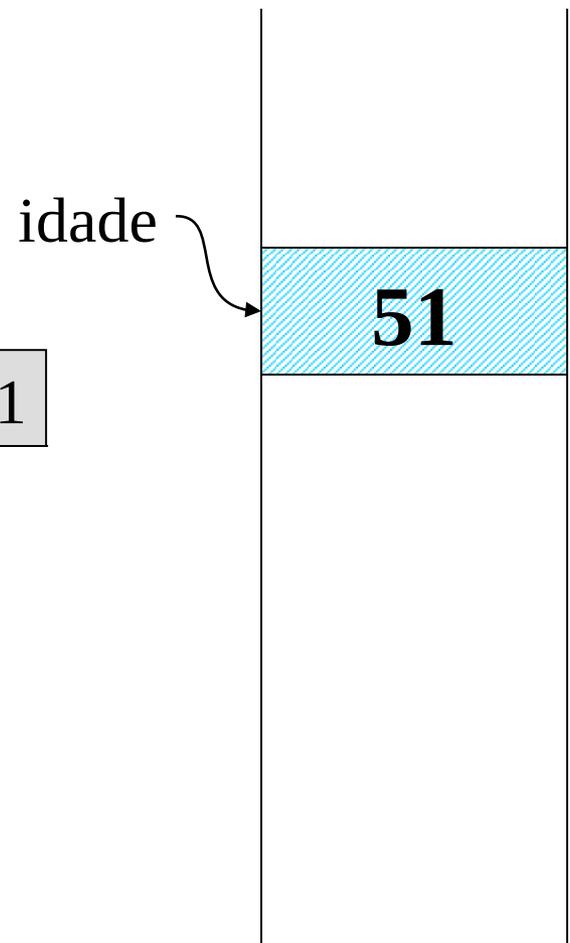


Área de memória
equivalente ao
armazenamento de
um dado inteiro



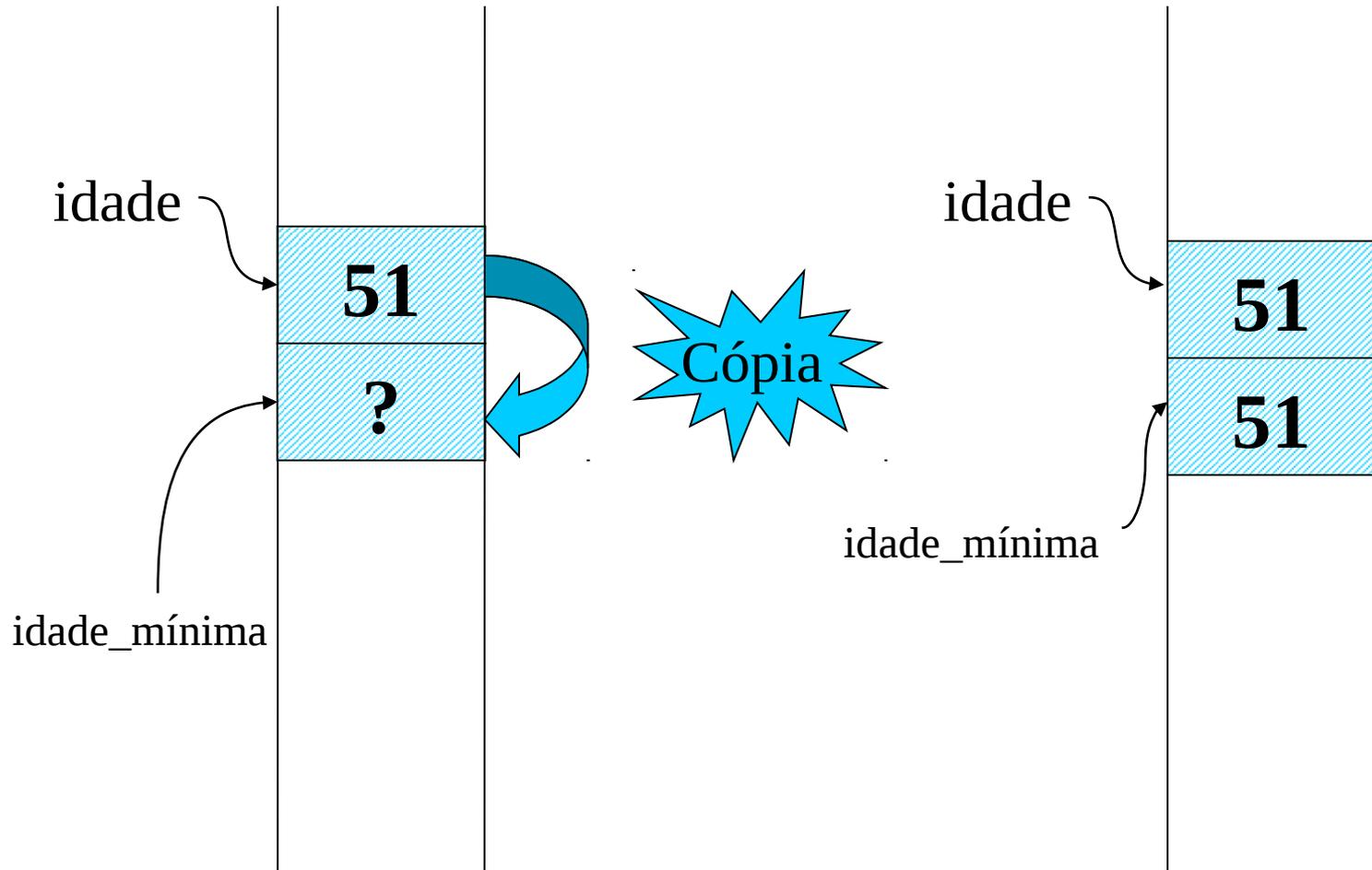
Antes

idade ← 51



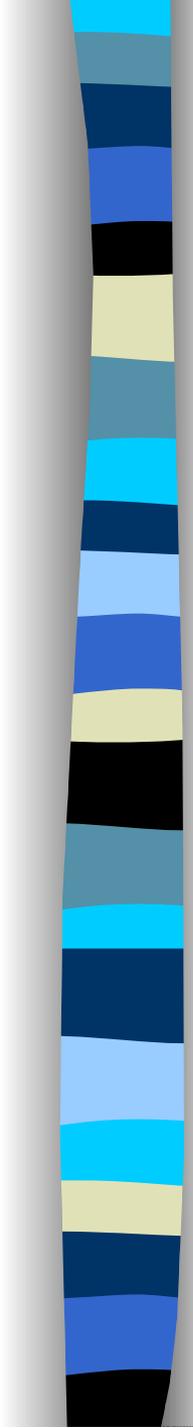
Depois

idade_mínima ← idade



Depois de idade ← 51

Depois de
idade_mínima ← idade



Variáveis

- O **tipo** de uma variável **não muda*** durante todo o algoritmo que a utiliza
- As **atribuições** entre variáveis podem ser feitas **apenas com e constantes variáveis** de mesmo tipo ou de tipo que seja compatível
- Dentre os tipos definidos até o momento só existe compatibilidade entre inteiro e real

* **Atenção:** em algumas linguagens como Python o tipo de uma variável pode mudar, mas isso não é uma boa prática.

Constantes Identificadas

- É possível dar nome às constantes utilizadas nos algoritmos

```
constante
```

```
pi = 3,1415926
```

```
salário_mínimo = 640,00
```

- As **constantes identificadas**, assim como as constantes literais, podem ser atribuídas a variáveis
- O Valor de uma **constante não se altera** após a sua definição

Definição de Tipos de Dados

- É possível **definir tipos** de dados a partir dos tipos já existentes e dar nome a eles
- Exemplos:

```
tipo  
  Booleano = lógico  
  eixo = 'x' até 'z'  
  dezena = 1 até 12
```

Expressões: Aritméticas e Lógicas

- Podemos **combinar valores** pela aplicação de operadores

$$\rightarrow 3 + 7 * 2 - 15$$

- Verdadeiro E Falso OU Verdadeiro

$$\rightarrow 3 + 2 < 5$$

- Pode-se armazenar o resultado de uma expressão em uma variável:

$$\rightarrow \textit{imposto} = \textit{valor} * 0,18$$

Operadores Aritméticos

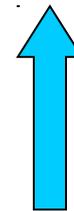
operador	primeiro operando	segundo operando	resultado	notação
+	a	b	$a + b$	$a + b$
+	a	—	$+a$	$+a$
-	a	b	$a - b$	$a - b$
-	a	—	$-a$	$-a$
*	a	b	$a \times b$	$a * b$
/	a	b	$\frac{a}{b}$	a/b

Precedência:

+ - unários

* /

+ - binários



Exemplos

$$3 + \underbrace{7 * 2}_{14} - 15$$

$$\underbrace{3 + 14}_{17} - 15$$

$$\underbrace{17 - 15}_{2}$$

2

$$\underbrace{(3 + 7)}_{10} * (2 - 15)$$

$$10 * \underbrace{(2 - 15)}_{-13}$$

$$\underbrace{10 * -13}_{-130}$$

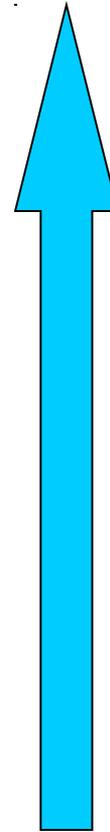
-130

Operadores Lógicos

verdadeiro OU verdadeiro = verdadeiro
verdadeiro OU falso = verdadeiro
falso OU verdadeiro = verdadeiro
falso OU falso = falso

verdadeiro E verdadeiro = verdadeiro
verdadeiro E falso = falso
falso E verdadeiro = falso
falso E falso = falso

NÃO verdadeiro = falso
NÃO falso = verdadeiro



Precedência: NÃO, E, OU

Operadores Relacionais

$=, >, <, \geq, \leq e \neq$

idade \leftarrow 28

valor \leftarrow 1000,00

fator \leftarrow 0,05

segurado \leftarrow idade $<$ 30 e valor*fator \leq 500,00

segurado \leftarrow idade $<$ 30 e valor*fator \leq 500,00

segurado \leftarrow idade $<$ 30 e 50 \leq 500,00

segurado \leftarrow verdadeiro e 50 \leq 500,00

segurado \leftarrow verdadeiro e verdadeiro

segurado \leftarrow verdadeiro

Precedência entre os Operadores

+ - unários

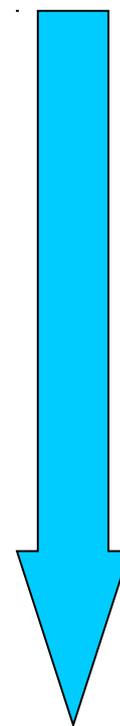
* /

+ - binários

não e ou

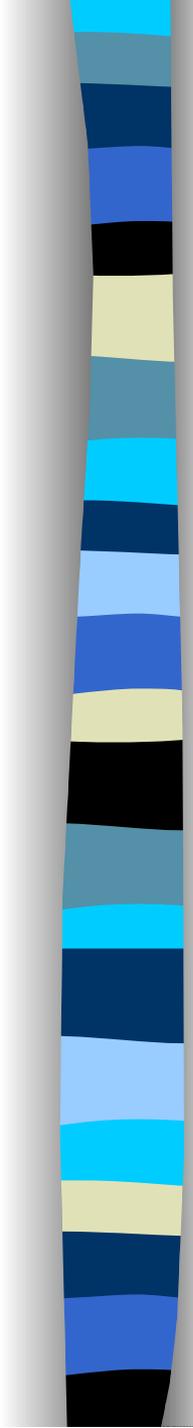
= < > ≤ ≥ ≠

←



Funções Pré-definidas

Função	Tipo dos Parâmetros	Resultado
raiz(x,n)	x: real, n: real	A n-ésima raiz de x
seno(x)	x: real	O seno de x dado em graus
cosseno(x)	x: real	O cosseno de x dado em graus
tangente(x)	x: real	A tangente de x dado em graus
exp(x)	x: real	e^x
abs(x)	x:real	O valor absoluto de x
arredonda(x)	x:real	Aproxima para o inteiro mais próximo

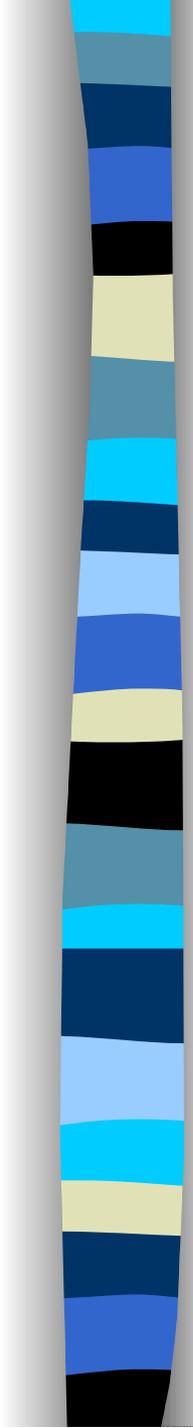


Entrada e Saída

- Um algoritmo pode receber dados através de dispositivos como teclado, mouse, discos e placas de rede, e pode enviar dados para o monitor de vídeo, discos e outros.
- Este tipo de operações em que dados são recebidos por um algoritmo ou são enviados por um algoritmo para um dispositivo são chamados de operações de entrada e saída

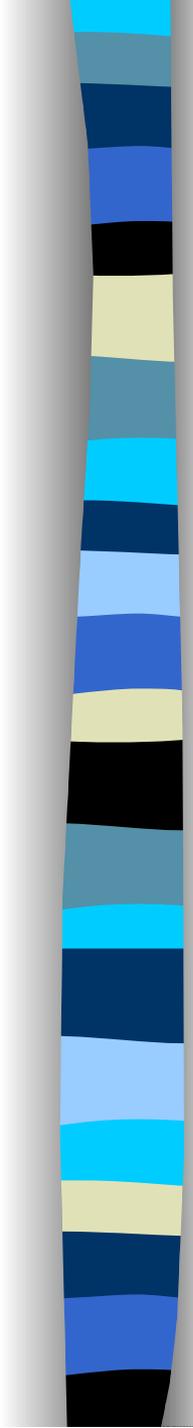
Entrada e Saída

Função	Parâmetros	Resultado
leia(x1,x2,...)	variáveis de qualquer tipo básico	Os valores digitados no teclado são armazenados em x1, x2, ...
escreva(y1,y2,...)	variáveis, constantes ou expressões de qualquer tipo básico	Os valores de y1, y2, ... são escritos no monitor.



Comentários

- São usados para descrever o algoritmo
- Indicar o significado de variáveis e constantes e esclarecer trechos de código



Linhas em Branco e Alinhamento

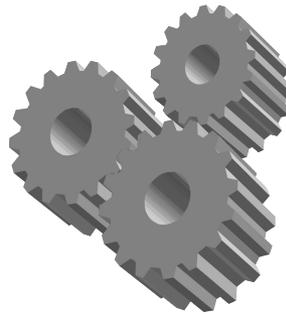
- Melhoram a legibilidade do programa
- Delimitam blocos de comandos do algoritmo, deixando claro quais comandos serão selecionados por uma alternativa

Resumindo

Um algoritmo é uma forma de organizar as idéias com o objetivo de construir um programa

Dados e Tipos
Comandos de Entrada

Entrada



Saída

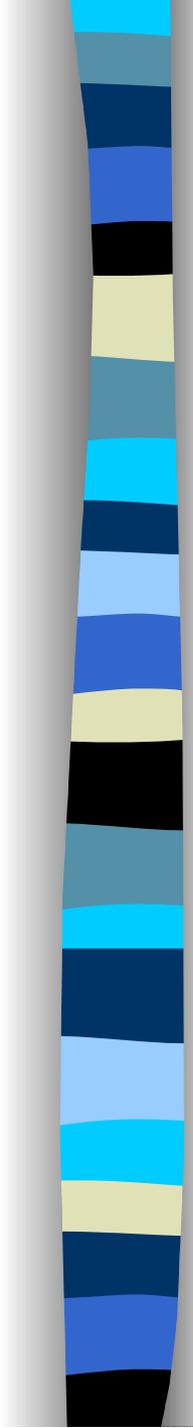
Dados e Tipos
Comandos de Saída

Processo

Expressões
Aritméticas, Relacionais
e Lógicas



Descrição
dos passos

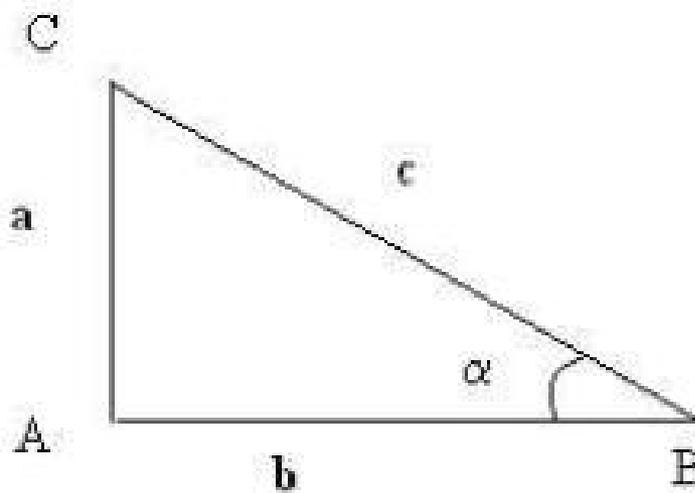


Sugestões

- Desenvolva o algoritmo em etapas
- Procure usar nomes de variáveis significativos, mesmo que eles fiquem longos
- Identifique se os passos individuais são suficientes independentes um dos outros
- Revise seu algoritmo em busca de possíveis erros e exceções que possam ser tratados

Exemplo: triângulo retângulo

Calcular dois lados de um triângulo retângulo, dados um ângulo e a hipotenusa



$$a = \text{sen}(\alpha) \times c \qquad b = \text{cos}(\alpha) \times c$$

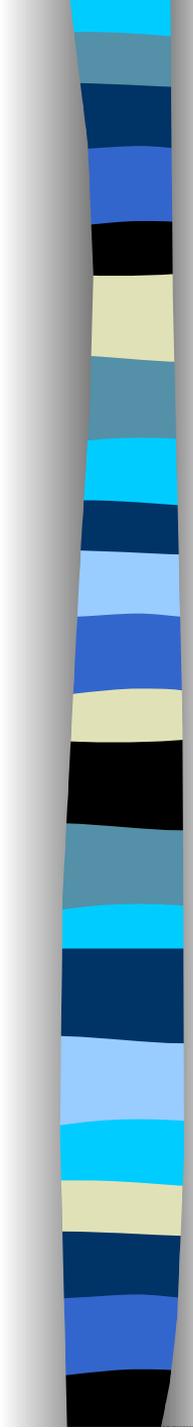
Exemplo: triângulo retângulo

```
Algoritmo lados_triângulo
{Este algoritmo calcula o valor dos lados de
um triângulo retângulo, dados um de seus
ângulos menores e a hipotenusa}

variável
    lado_oposto, lado_adjacente: real
    hipotenusa, alfa: real

leia(alfa)
leia(hipotenusa)
lado_oposto ← seno(alfa)*hipotenusa
lado_adjacente ← cosseno(alfa)*hipotenusa
escreva(lado_oposto)
escreva(lado_adjacente)

fim
```



Exemplo: triângulo retângulo

Suponha agora que o usuário digite um número negativo. Embora isso não fosse natural de acontecer, seria razoável que o algoritmo fosse capaz de tratar o problema. Nesse caso, o comprimento do lado oposto ficaria negativo, o que é errado, uma vez que o seno de um ângulo negativo é negativo.

Assim a solução seria fornecer, como resultado, o valor absoluto do cálculo do comprimento do lado oposto. A versão a seguir prevê esse caso, através do comando:

```
lado_oposto ← abs(seno(alfa)*hipotenusa)
```

Esse comando também ilustra a chamada de uma função passando como parâmetro uma expressão.

Exemplo: triângulo retângulo

```
Algoritmo lados_triângulo
{Este algoritmo calcula o valor dos lados de
um triângulo retângulo, dados um de seus
ângulos menores e a hipotenusa}

variável
    lado_oposto, lado_adjacente: real
    hipotenusa, alfa: real

leia(alfa)
leia(hipotenusa)
lado_oposto ← abs(seno(alfa)*hipotenusa)
lado_adjacente ← cosseno(alfa)*hipotenusa
escreva(lado_oposto)
escreva(lado_adjacente)

fim
```

Exemplo: Salário

Algoritmo salário

{Este algoritmo calcula o valor do salário de um funcionário dados o valor total de suas vendas e sua porcentagem de comissão}

constante

salário_base = 640,00

variável

salário: real

comissão: real

valor_vendido: real

leia(comissão, valor_vendido)

salário ← salário_base + comissão/100*valor_vendido

escreva(salário)

fim

Exercícios

Desenvolver algoritmos **EM PSEUDO-CÓDIGO** para:

1. Cálculo da área de um retângulo
2. Cálculo de área de um círculo
3. Cálculo da massa molecular de uma molécula com até três tipos de átomos (carbono, hidrogênio e oxigênio).

Exemplo: para a molécula $C_{12}H_{22}O_{11}$ (açúcar), a massa molecular será:

- 12 átomos de carbono = $12 \times 12,0111 \text{ u} = 144,133 \text{ u}$
- 22 átomos de hidrogênio = $22 \times 1,0079 \text{ u} = 22,174 \text{ u}$
- 11 átomos de oxigênio = $11 \times 15,9994 = 175,993 \text{ u}$ e a soma **342,3 u**.

Exercícios

4. Cálculo da vazão d'água em um canal

Exemplo: Seja $V = 0,2\text{m/s} = \mathbf{K}\sqrt{\mathbf{R}\mathbf{J}}$ (velocidade), $\mathbf{A} = 0,5\text{m}^2$ (área de secção), $\mathbf{R} = 0,18$ (raio médio), $\mathbf{J} = 0,0005$ (inclinação por metro) e $\mathbf{K} = 21$ (coeficiente de Bazin).

$$\mathbf{Q} = \mathbf{A}\mathbf{K}\sqrt{\mathbf{R}\mathbf{J}} = \mathbf{0,5} \times (\mathbf{21}\sqrt{\mathbf{0,18} \times \mathbf{0,0005}}) = \mathbf{0,0997\text{m}^3/\text{s}}$$

5. Implementar todos os algoritmos vistos em aula e os dos exercícios em C.