AULA DE ALGORITMO - 01

Profa. M. Sc. Valéria Maria Volpe

Conceito de Lógica

Conceito de lógica

- Pode-se dizer que lógica é a “**correção do pensamento**”, pois uma das preocupações que se tem ao usar a lógica é **determinar** quais **operações são válidas e quais não são**, fazendo análise das formas e leis do pensamento. A lógica nos ensina a **usar corretamente as leis do pensamento**.

Conceito de Lógica

- **Conceito de lógica**
  - Também pode-se dizer que lógica é a **arte de bem pensar**, que é a **ciência das formas do pensamento** visto que a forma mais complexa do pensamento é o raciocínio, a lógica é a **correção do raciocínio**.
  - Pode-se dizer também que lógica é a **ordem da razão**. Portanto, a lógica nos permite **colocar ordem no pensamento**.

Conceito de Lógica de Programação

- **Lógica de programação**
  - Significa o **uso correto das leis do pensamento**, da ordem da razão e do processo do raciocínio e **simbolização formais** na **programação de computadores**, objetivando o desenvolvimento de técnica que cooperam para a produção de **soluções logicamente válidas e coerentes**, que resolvam com **qualidade** os problemas.

Conceito de Algoritmo

- **ALGORITMO**:
  - É uma sequência de passos, descritos de forma lógica, que visam atingir um objetivo bem definido.
- O principal objetivo da lógica de programação é a **construção de algoritmo válidos, coerentes e com qualidade**.
- Uma vez que um algoritmo representa o **raciocínio envolvido na lógica programação**, ele nos permite abstrair detalhes computacionais que podem ser **programados em qualquer linguagem de programação**.

Algoritmo - Regras

- **Regras para a construção de algoritmo:**

1. Identificar o início e o fim do algoritmo.
2. Descrever cada passo em uma linha.
3. Usar verbos no infinitivo/impessoal.

Algoritmo - Exemplo

- **Exemplo:** Faça um algoritmo para trocar uma lâmpada queimada.

Inicio

Pegar uma escada

Posicionar a escada debaixo da lâmpada queimada

Pegar uma lâmpada nova

Subir na escada

Retirar a lâmpada queimada

Colocar a lâmpada nova

Descer da escada

Jogar a lâmpada queimada no lixo

Guardar a escada

Fim

Algoritmo - Exercício

1. Elabore um ALGORITMO para resolver a Torre de Hanói. A Torre de Hanói consiste em três hastes e vários discos, mas aqui faremos com três discos. O objetivo é mover os discos de uma haste para outra respeitando as seguintes regras:

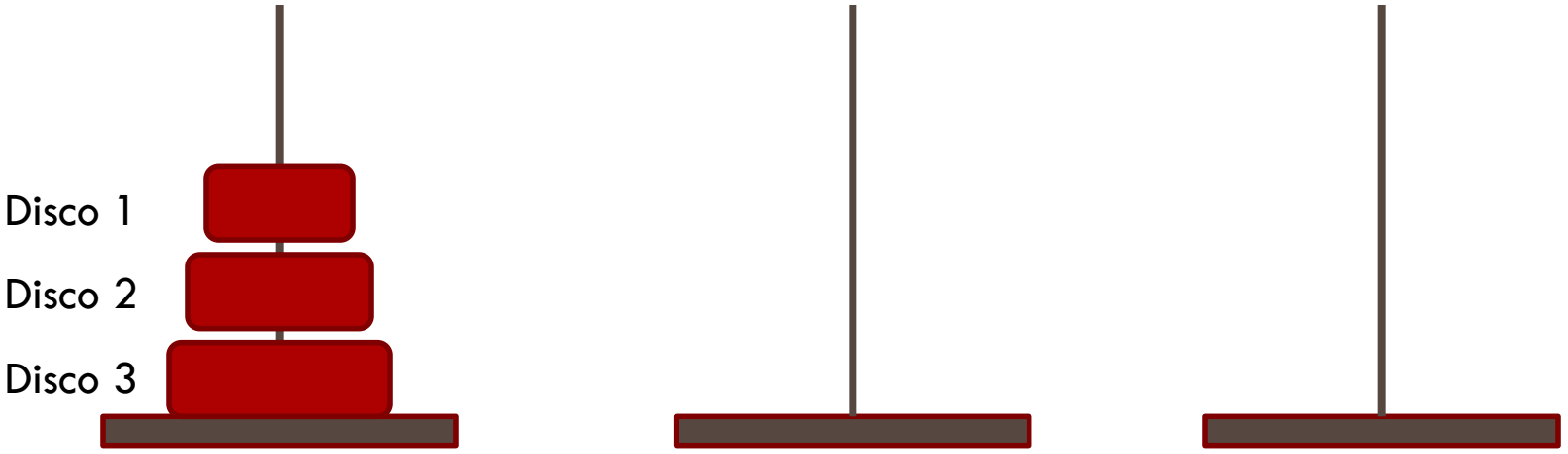
   a) Mover um disco de cada vez;

   b) Nunca colocar um disco maior sobre o disco menor;

   c) Levar os discos da haste A para a C.

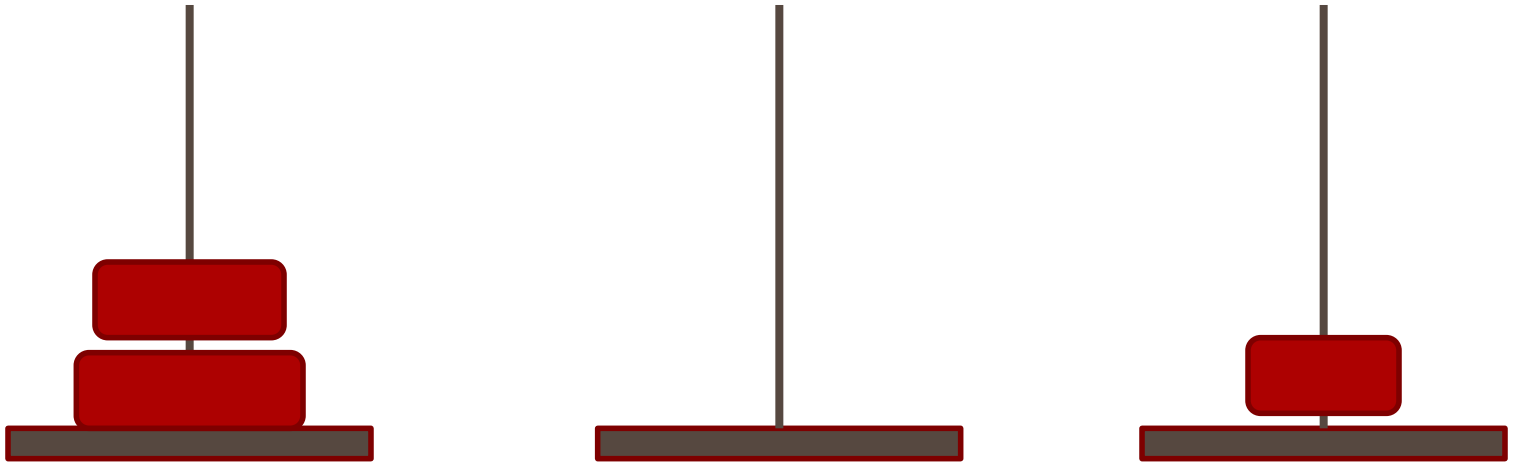
Algoritmo - Exercício

- **Torre de Hanói**



Algoritmo - Exercício

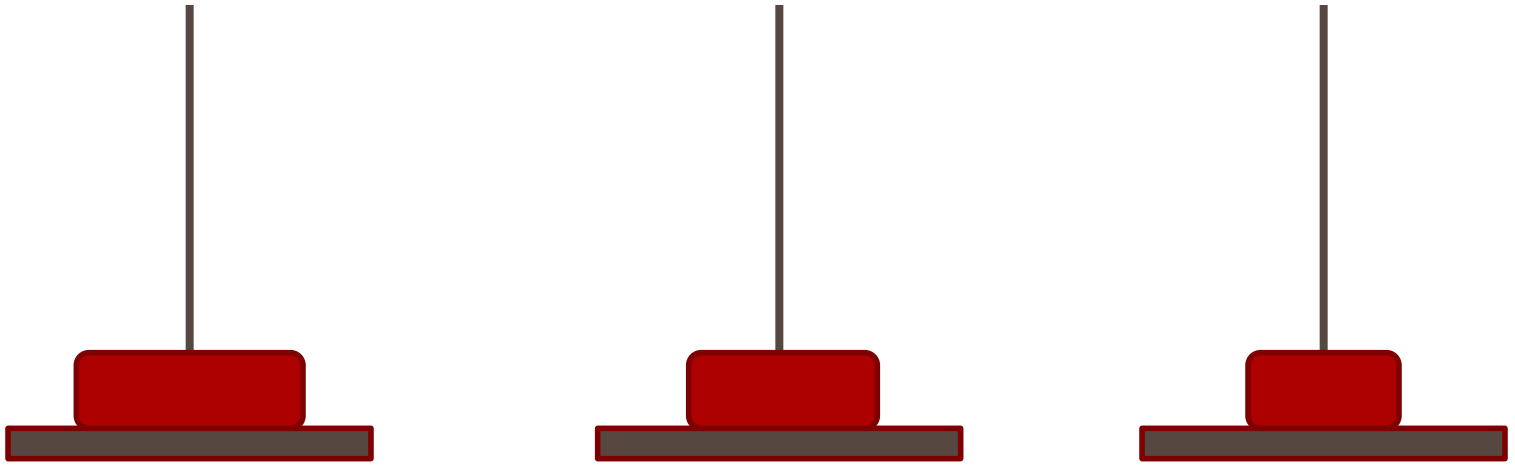
- **Torre de Hanói**



- **Haste A** **Haste B** **Haste C**

Algoritmo - Exercício

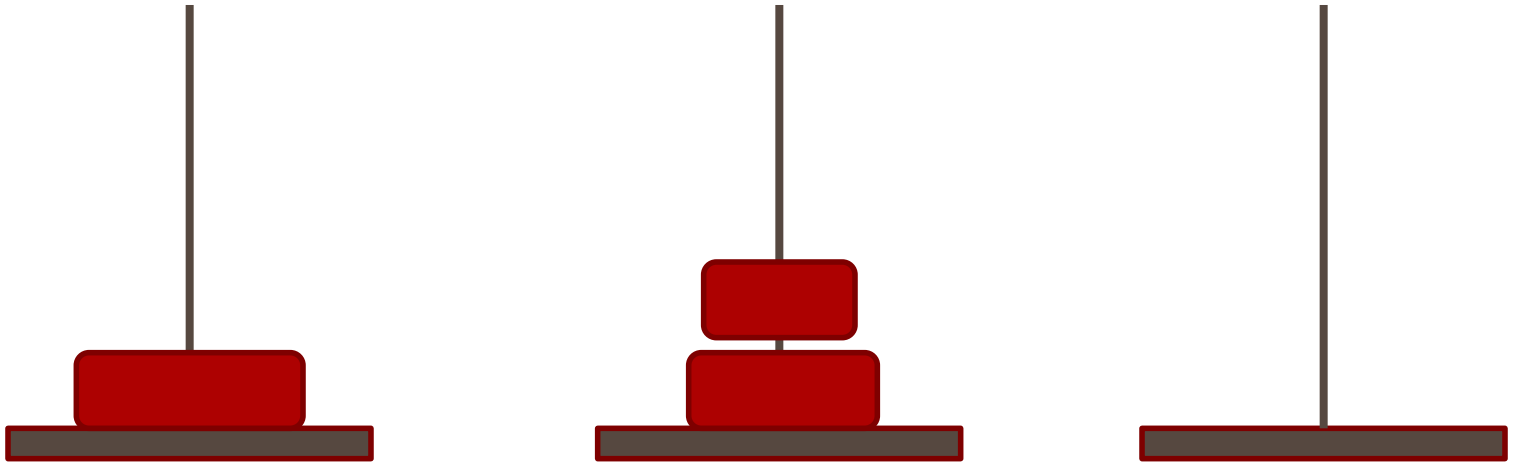
- **Torre de Hanói**



- **Haste A** **Haste B** **Haste C**

Algoritmo - Exercício

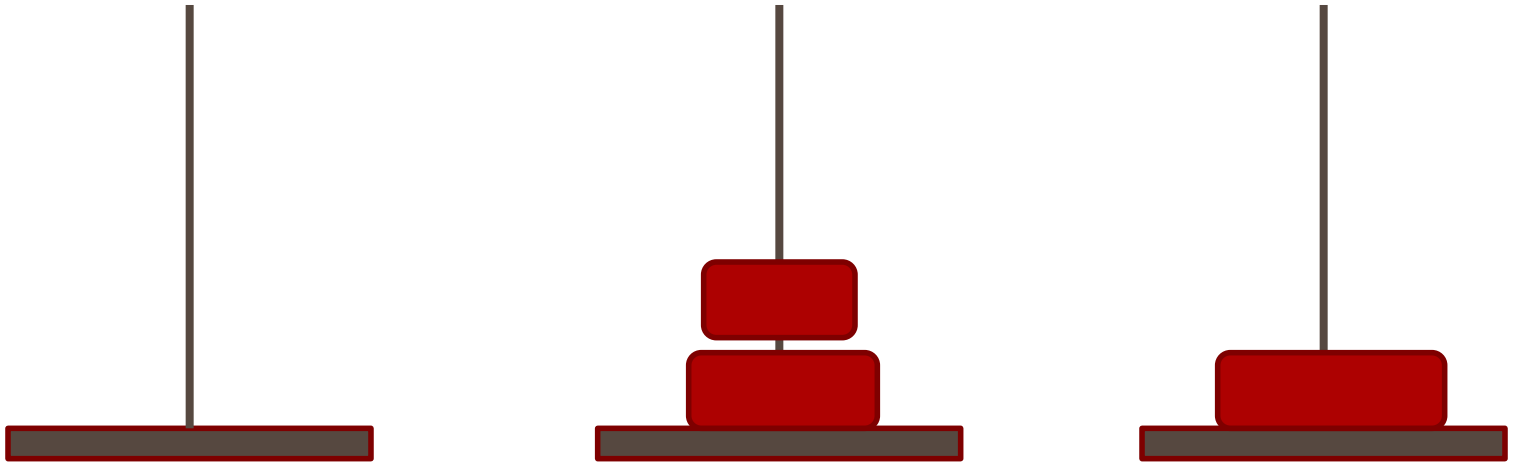
- **Torre de Hanói**



- **Haste A** **Haste B** **Haste C**

Algoritmo - Exercício

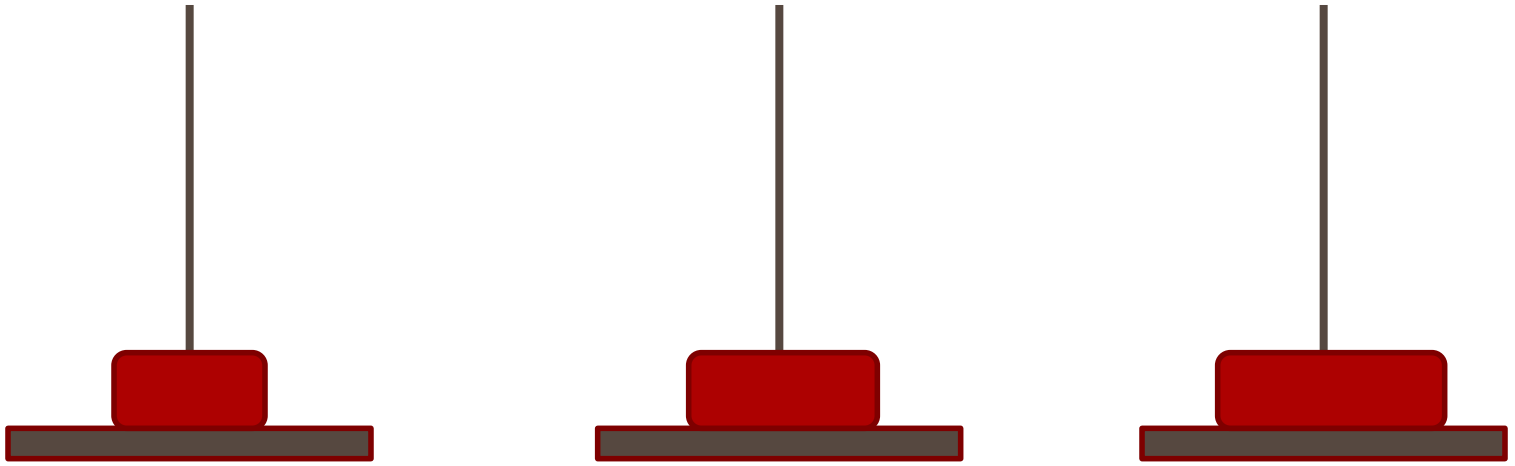
- **Torre de Hanói**



- **Haste A** **Haste B** **Haste C**

Algoritmo - Exercício

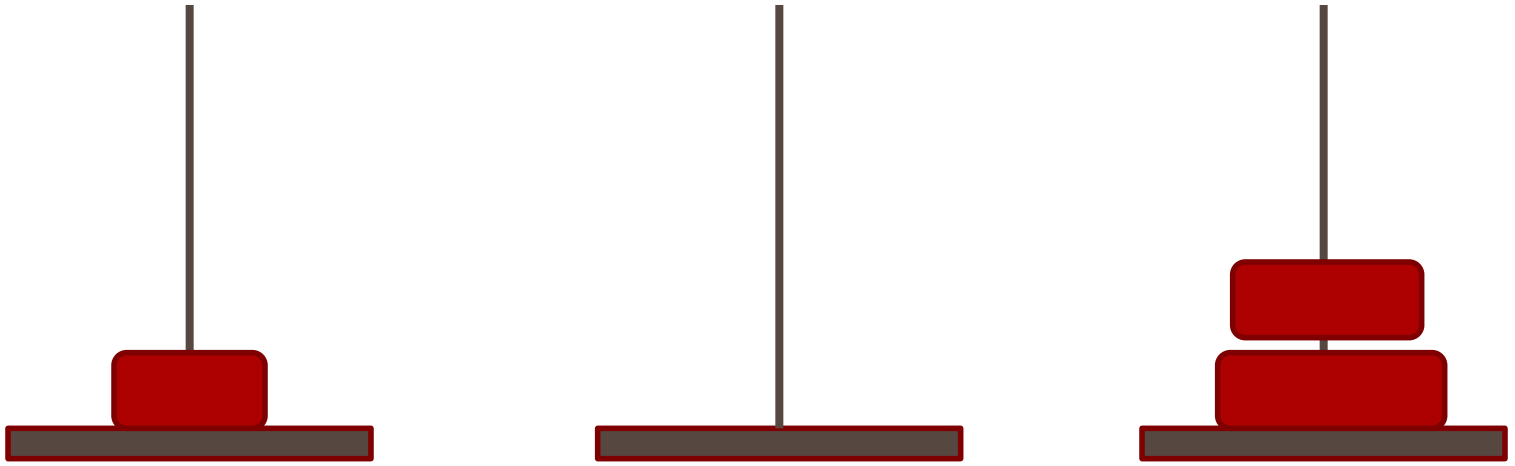
- **Torre de Hanói**



- **Haste A** **Haste B** **Haste C**

Algoritmo - Exercício

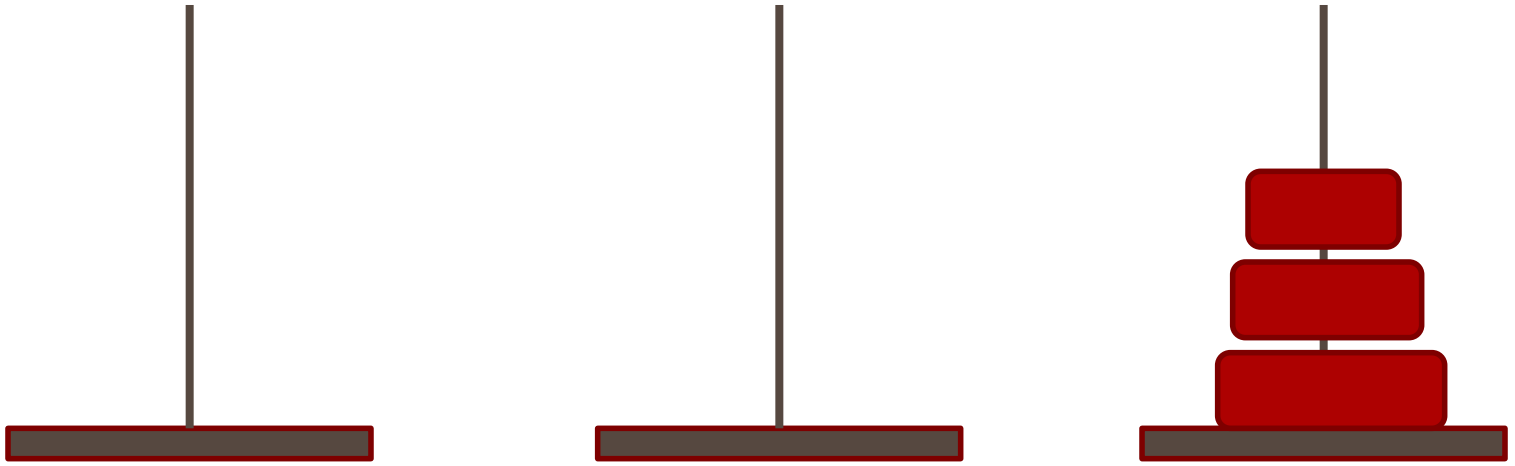
- **Torre de Hanói**



- **Haste A** **Haste B** **Haste C**

Algoritmo - Exercício

- **Torre de Hanói**



- **Haste A** **Haste B** **Haste C**

Algoritmo - Exercício

- **Solução:**

Início
Retirar o Disco 1 da Haste A
Colocar o Disco 1 na Haste C
Retirar o Disco 2 da Haste A
Colocar o Disco 2 na Haste B
Retirar o Disco 1 da Haste C
Colocar o Disco 1 na Haste B
Retirar o Disco 3 da Haste A
Colocar o Disco 3 na Haste C
Retirar o Disco 1 da Haste B
Colocar o Disco 1 na Haste A
Retirar o Disco 2 da Haste B
Colocar o Disco 2 na Haste C
Retirar o Disco 1 da Haste A
Colocar o Disco 1 na Haste C
Fim

Algoritmo - Exercícios

1. Um homem precisa atravessar um rio com um barco que possui capacidade apenas para carregar ele mesmo e mais uma carga, que são: um maço de alfafa, um bode e um lobo. O que o homem deve fazer para conseguir atravessar o rio sem perder suas cargas? Sabendo que o lobo come o bode e o bode come o maço de alfafa. Faça um algoritmo para resolver o problema.

2. Três jesuítas e três canibais precisam atravessar um rio; para tal dispõem de um barco com capacidade para duas pessoas. Por medida de segurança, não se deve permitir que em algumas margens a quantidade de jesuítas seja menor que à de canibais. Qual a solução para realizar a travessia em segurança? Faça um algoritmo para resolver este problema.

Algoritmo – Exercícios 1 - Solução

Início
Pegar o Bode
Subir no barco
Atravessar o rio da margem ESQ para margem DIR
Descer do barco
Deixar o Bode na margem DIR
Subir no barco
Atravessar o rio da margem DIR para margem ESQ
Descer do barco
Pegar o maço de alfafa
Subir no barco
Atravessar o rio da margem ESQ para margem DIR
Descer do barco
Deixar o maço de alfafa na margem DIR
Pegar o Bode
Subir no barco
Atravessar o rio da margem DIR para margem ESQ

Algoritmo – Exercícios 1 - Solução

Descer do barco
Deixar o Bode na margem ESQ
Pegar o Lobo
Subir no barco
Atravessar o rio da margem ESQ para margem DIR
Descer do barco
Deixar o Lobo na margem DIR
Subir no barco
Atravessar o rio da margem DIR para a margem ESQ
Descer do barco
Pegar o Bode
Subir no barco
Atravessar o rio da margem ESQ para a margem DIR
Descer do barco
Deixar o Bode na margem DIR
Fim.

Algoritmo – Exercícios 2 - Solução

Início
- Subir no barco 2 canibais
- Atravessar o rio da margem ESQ para a margem DIR
- Descer do barco um canibal
- Atravessar o rio da margem DIR para a margem ESQ
- Subir no barco 1 canibal
- Atravessar o rio da margem ESQ para a margem DIR
- Descer do barco um canibal
- Atravessar o rio da margem DIR para a margem ESQ
- Descer do barco 1 canibal
- Subir no barco 2 jesuítas
- Atravessar o rio da margem ESQ para a margem DIR
- Descer do barco dois jesuítas
- Subir no barco 1 canibal em 1 jesuíta
- Atravessar o rio da margem DIR para a margem ESQ
- Descer do barco 1 canibal e 1 jesuíta
- Subir no barco 2 jesuítas

Algoritmo – Exercícios 2 - Solução

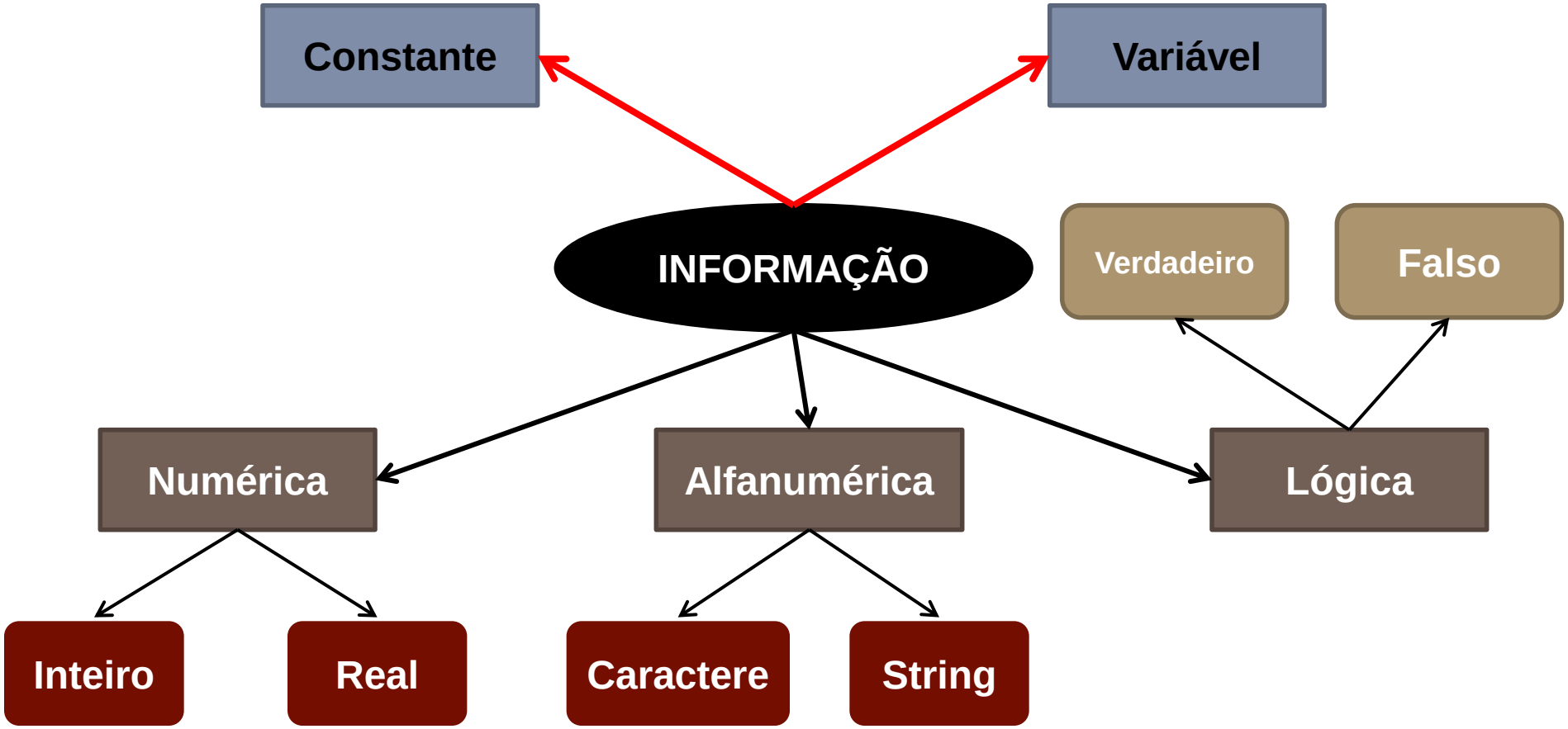
Atravessar o rio da margem ESQ para a margem DIR
Descer do barco 2 jesuítas
Subir no barco 1 canibal
Atravessar o rio da margem DIR para a margem ESQ
Subir no barco 1 canibal
Atravessar o rio da margem ESQ para a margem DIR
Descer do barco 1 canibal
Atravessar o rio da margem DIR para a margem ESQ
Subir no barco 1 canibal
Atravessar o rio da margem ESQ para a margem DIR
Descer do barco 2 canibais

Fim.

Dados X Informação

- **Dados e Informação:**
  - Informação é a matéria prima da computação, pois os computadores são capazes de manipular um grande volume de informação em pouco tempo.
  - **Dados**: Conjunto de valores manipulados por sistemas de informação.
  - **Informação**: Conjunto de dados que agregam valores aos sistemas de informação, ou seja, tem algum significado.
- **Exemplo**:
  - Dia 19 de abril => dado
  - Dia 19 de abril dia do índio => informação

Informação


Constante
Variável
INFORMAÇÃO
Verdadeiro
Falso
Numérica
Alfanumérica
Lógica
Inteiro
Real
Caractere
String

Tipos Primitivos

- **Tipos Primitivos:**
  - Os tipos primitivos serão usados para classificar as informações manipuladas pelas aplicações que serão desenvolvidas.
- **Os tipos Primitivos são**:
  - **Inteiro:** Todo e qualquer valor numérico pertencente ao conjunto dos números inteiros (negativo, nulo, positivo).
- **Exemplo:**
  - Nº de filhos
  - Nº de faltas

Tipos Primitivos

- **Os tipos Primitivos são**:
  - **Real:** Todo e qualquer valor numérico que pertence ao conjunto dos números reais (negativos, nulo, positivo).
- **Exemplo**
  - Altura de uma pessoa
  - Salário do funcionário

Tipos Primitivos

- **Os tipos Primitivos são**:
  - **Caracter:** Toda e qualquer informação composta por um conjunto de caracteres alfanumérico (‘A’;....;’Z’;’a’;....;’z’; 0.....9; símbolos especiais #, !, ?).
  - **Caracter:** Apenas 1 caracter alfanumérico.
  - **String:** Mais de 1 caracter alfanumérico
- **Exemplo:**
  - Nome do cliente
  - Sexo de uma Pessoa (M ou F)

Tipos Primitivos

- **Os tipos Primitivos são**:
  - **Lógico:** Toda e qualquer informações que pode assumir valores lógicos verdadeiro ou falso.
- **Exemplo**
  - Lâmpada acesa ou apagada.

Tipos Primitivos

- **Tabela Resumo**

Algoritmo	Linguagem C	Uso de Memória	Valores (Limite)
inteiro	int	2 bytes	-32.768 a 32.767
real	float	4 bytes	$3,4*10^{-38}$ a $3,4*10^{38}$
caractere	char	1 byte	-128 a 127 (0 a 255)
string	char [tamanho]	1 byte por caractere	-128 a 127 (0 a 255)
lógico	bool (Linguagem C++)	1 byte	V (true) ou F (false)

Constante e Variável

- **Constantes e Variáveis:**
  - Uma informação deve ser classificada como constante ou variável.

- **Constante:** Uma informação constante é uma informação que, após receber um valor inicial, este valor **não pode** ser alterado no decorrer do tempo (durante a execução do programa).
- **Exemplo:**
  - π = 3.14

Constante e Variável

- **Variável:** Uma informação variável é uma informação que **pode ser** alterada no decorrer do tempo (durante a execução do programa).
- **Exemplo:**
  - Valor do dólar,
  - peso de uma pessoa,
  - etc...

Constante e Variável

- **Identificador de constante e variável:**
  - Uma informação, para ser utilizada por nossos programas, deve receber um nome.
  - Este nome será seu identificador, que será a referência ao valor que a informação armazena e será a forma de acesso ao espaço de memória reservado para a informação.
  - Ao utilizarmos o nome da informação estaremos acessando seu conteúdo, que pode ser para usar o valor armazenado, assim como, armazenar um novo valor alterando o valor que já está armazenado. É de responsabilidade dos programadores dar nomes às informações (constantes / variável).

Constante e Variável

- **Regras para criação dos identificadores**
  - Devem começar sempre com uma **Letra**
  - Os demais caracteres podem ser **Letras ou Números**
  - Não é permitido usar símbolos especiais.
- **Exemplo:**
- nomeAluno, dataNasc, nro1, nr02, valor_Dolar

Constante e Variável

- **Declaração de Constante e Variável**
  - Declarar uma constante ou variável significa identificar seu tipo primitivo, identificar se a informação é constante ou variável e criar seu identificador (Nome/Referência). Com a declaração o computador irá alocar um espaço de memória suficiente para armazenar a informação. Esse espaço de memória será reservado de acordo com o tipo primitivo da informação. Também com a declaração, o computador irá relacionar o identificador da constante ou variável com o endereço do espaço de memória reservado transformando o identificador em referência de memória permitindo o acesso ao conteúdo da informação.

Constante e Variável

- **Declaração de Constante**
  - Para declararmos uma informação como sendo uma **constante** usamos a palavra reservada **const** tanto nos algoritmos como nos programas em linguagem C.

Constante e Variável

- **Sintaxe – Algoritmo**

const

tipo primitivo: ID_Const = Valor_Inicial;

inteiro, real, caracter, lógico — Identificador da constante — Valor Constante atribuído

- **Exemplo**

const

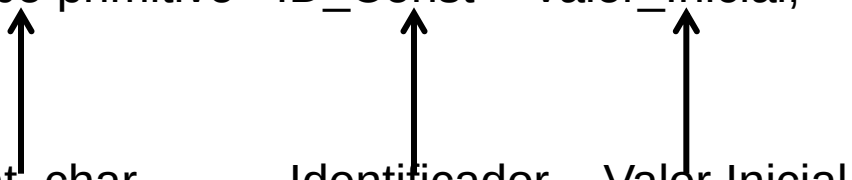
real: PI = 3.14;

inteiro: PFALTAS = 25;

Constante e Variável

- **Sintaxe – Linguagem C**

const tipo primitivo ID_Const = Valor_Inicial;



- **Exemplo**

const float PI=3.14;

const int PFALTAS = 25;

Constante e Variável

- **Declaração de Variável**
  - Para declararmos uma informação como sendo **variável** usamos a palavra reservada **var** em nossos algoritmos. Em linguagem C não existe uma palavra que identifique as variável, basta declará-las.

Constante e Variável

- **Sintaxe – Algoritmo**

  var

  tipo primitivo: ID_Var;

  inteiro, real, caracter, lógico

  Identificador da variável

- **Exemplos:**

  var

  inteiro: idade, nroFaltas, anoNasc;

  real: notaProva01, notaProva02, valor_Dolar;

Constante e Variável

- **Sintaxe – Linguagem C**

  tipo primitivo ID_Var;

  int, float, char Identificador da variável

- **Exemplos:**

  int idade, nroFaltas, anoNasc = 1982;

  float notaProva01, notaProva02, valor_Dolar;

Exercícios

1) Assinale os identificadores válidos

• a) (x)	b) v2	C) há	D) AH!
• E) “nomeAluno”	F) nomeAluno	G) vyt	H) ABC*D
• I) 234xyz	J) O&0	K) rua	L) cep
• M) diaNasc?	N) dia_Nasc	O) mês	P) nota.Prova

2) Identifique o tipo primitivo das informações grifadas. Faça a declaração em algoritmo e Linguagem C.

- Ana tem 1,73m de altura.
- O saldo bancário de João é R$2734,57
- Estava escrito na prova: Marque com X a alternativa correta
- Pedro tem 2 filhos. Uma menina e um menino
- Fui bem na prova tirei 7,75
- A fórmula para calcular o comprimento do circulo é 2πR.

Exercícios

- Solução Ex02 - Algoritmo:

```
var
        real: altura, saldo, notaProva, raio;
        inteiro: numFilhos;
        caracter: sexo;
        string: nome1, nome2;
const
        real: PI = 3.14;
        caracter: MARCADOR = ‘X’;
```

Exercícios

- Solução Ex02 – Linguagem C:

```
float altura, saldo, notaProva, raio;
int numFilhos;
char sexo, nome1[40], nome2[50];

const  float PI = 3.14;
const char MARCADOR = ‘X’;
```

Operadores

- Os Operadores, como o próprio nome diz, são usados para realizar operações. Essas operações podem ser:
  - Armazenamento de dados (operador de atribuição)
  - Processamento (operadores aritméticos, relacionais e lógicos)
- Os operadores só funcionam com valores de mesmo tipo.

Operador de Atribuição

- **Operador de Atribuição** (←):
  - O Operador de atribuição (←) é usado para armazenar (atribuir) um Valor em uma variável. Caso seja atribuído uma expressão à uma variável, esta será resolvida e o resultado será armazenado.
- **Sintaxe – Algoritmo**

Nome_Var ← Valor;

Identificar Da variável

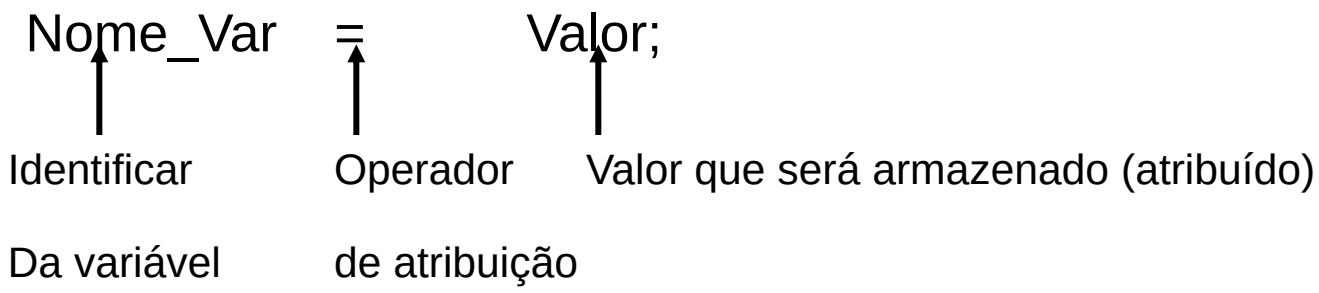
Operador de atribuição

Valor que será armazenado (atribuído)

- Exemplos
  - x ← 10;
  - y ← 6.3;
  - sexo ← ‘F’;

Operador de Atribuição

- **Sintaxe – Linguagem C**



- Exemplos
  - x = 10;
  - y = 6.3;
  - sexo = ‘F’;

Operador de Atribuição

- Só é possível atribuir um valor à uma variável, este deve ser do mesmo tipo da variável, ou seja, uma variável declarada como sendo do tipo inteiro só armazena números inteiros.
- Uma variável só armazena um valor, sempre o último que lhe foi atribuído.
- Se uma expressão for atribuída a uma variável, ela será resolvida e seu resultado será armazenado.

Operadores

- **Operadores Aritméticos**:
  - Os Operadores aritméticos são usado para realizar as operações aritméticas básicas de:
    - Adição
    - Subtração
    - Multiplicação
    - Divisão

Operadores Aritméticos

Algoritmo	Linguagem C	Função	Conjuntos Válidos	Exemplo Algoritmo
+	+	Usado para calcular a adição entre dois números	real inteiro	5.0 + 2.0 = 7.0 5 + 2 = 7
-	-	Usado para calcular a subtração entre dois números	real inteiro	5.0 - 2.0 = 3.0 5 - 2 = 3
*	*	Usado para calcular a multiplicação entre dois números	real inteiro	5.0 * 2.0 = 10.0 5 * 2 = 10
/	/	Usado para calcular a divisão entre dois números	real inteiro	5.0 / 2.0 = 2.5 5 / 2 = 2
mod	%	Usado para calcular o resto da divisão entre dois números inteiros	inteiro	5 mod 2 = 1

Operadores Aritméticos

- Prioridade de operações
  - 1º) Parênteses do interno para o externo
  - 2º) Funções matemáticas (exponenciação, radiciação, etc.)
  - 3º) * / mod
  - 4º) + -
  - 5º) Operador de atribuição ←

Operadores Aritméticos

- **Exercício:** Resolva a expressão dada:

3 * (7 + 2 * (**16 – 6** + 5) / 4) + 7

3 * (7 + 2 * (**10 + 5**) / 4) + 7

3 * (7 + **2 * 15** / 4) +7

3 * (7 + **30 / 4**) +7

3 * (**7 + 7**) +7

3 * 14 + 7

42 + 7

49

Operadores Aritméticos

- **Exercício:** Resolva a expressão dada:

3.0 * (7.0 + 2.0 * (**16.0 – 6.0** + 5.0) / 4.0) + 7.0

3.0 * (7.0 + 2.0 * (**10.0 + 5.0**) / 4.0) + 7.0

3. 0* (7.0 + **2.0 * 15.0** / 4.0) +7.0

3.0 * (7.0 + **30.0 / 4.0**) +7.0

3.0 * (**7.0 + 7.5**) +7.0

3.0 * 14.5 + 7.0

43.5 + 7.0

50.5

Exercícios

- Resolva as expressões aritmética:
  a) (3*5+2+10MOD3*(17/2))+(7*(3*10/(5*2+4)–3)–1)*2
  b) 12.0*3.0/7.0+10.0*3.5/2.7–((17.0*4.2+5.0)–3.0)
  c) 17+3-2+10* $((\sqrt{81} * \sqrt{225})/\sqrt[3]{27})$

Exercícios

- Solução:

(3*5+2+10MOD3*(17/2))+(7*(3*10/(5*2+4)–3)–1)*2

(3*5+2+10MOD3* 8)+(7*(3*10/(10+4)–3)–1)*2

(15+2+1*8)+(7*(3*10/14–3)–1)*2

(15+2+8)+(7*(30/14–3)–1)*2

(17+8)+(7*(2–3)–1)*2

25+(7*(-1)–1)*2

25+(-7 – 1)*2

25+(-8)*2

25+(-16)

25 - 16

9

Exercícios

- Solução:

12.0*3.0/7.0+10.0*3.5/2.7–((17.0*4.2+5.0)–3.0)

36.0/7.0+35.0/2.7–((71.4+5.0)–3.0)

5.14+12.96–(76.40–3.0)

18.10 – 73.40

-55.30

Exercícios

- Solução:

$$17+3-2+10*((\sqrt{81}*\sqrt{225})/\sqrt[3]{27})$$

$$20-2+10*(9*15/3)$$

$$18+10*(135/3)$$

$$18+10*45$$

$$18+450$$

$$468$$

Exercício

- Calcule o valor final das variáveis X, Y, Z, A e K. Sabendo que as atribuições abaixo representam um bloco de comandos de um **Algoritmo**. Faça a declaração das variáveis X, Y, Z, A e K em **Algoritmo** e **Linguagem C**.

a)
```
X ← 10;
Y ← 15;
Z ← 32;
X ← X+Y;
Y ← Z–X;
A ← 25;
Z ← A+14MOD3;
K ← 0;
K ← K+1;
K ← K+1;
K ← K+1;
K ← K+A;
```

Memória

X	**25**
Y	**7**
Z	**27**
A	**25**
K	**28**

b)
```
X ← 12.0;
X ← X+2.0*3.0;
Y ← 5.0;
Z ← 6.3;
A ← 12.98;
A ← A+Y;
Z ← X*2-(Z+Y);
K ← 2.6+A;
K ← K*K;
Z ← Z+2.5*A;
X ← X/2.0*3.5+((A*3.0)-Y*2);
Y ← X+Y+Z+A;
```

Memória

X	**75.44**
Y	**168.07**
Z	**69.65**
A	**17.98**
K	**423.53**

Solução a)

a)

```
X ← 10;
Y ← 15;
Z ← 32;
X ← X+Y;
Y ← Z–X;
A ← 25;
Z ← A+14MOD3;
K ← 0;
K ← K+1;
K ← K+1;
K ← K+1;
K ← K+A;
```

Memória

X	25
Y	7
Z	27
A	25
K	28

Solução a)

- Declaração das variáveis inteiras
  - Algoritmo
  - var
    - inteiro: X, Y, Z, K, A;
  - Linguagem C
    - int X, Y, Z, K, A;

Solução b)

- Declaração das variáveis reais
  - Algoritmo
  - var
    - real: X, Y, Z, K, A;
  - Linguagem C
    - float X, Y, Z, K, A;

Operadores Relacionais

- **Operadores Relacionais:**
  - Os operadores relacionais são usados para comparar valores de mesmo tipo.
  - O uso dos operadores relacionais criará uma expressão lógica. Portanto, o resultado de uma expressão lógica será verdadeiro ou falso.
  - Geralmente se usa os operadores relacionais nas estruturas de controle de decisão e repetição.

Operadores Relacionais

Algoritmo	Linguagem C	Função	Exemplo Algoritmo	Resultado da comparação
>	>	Usado para comparar se um valor é maior que outro	5 > 5	F
>=	>=	Usado para comparar se um valor é maior ou igual a outro	5 >= 5	V
<	<	Usado para comparar se um valor é menor que outro	5 < 5	F
<=	<=	Usado para comparar se um valor é menor ou igual a outro	5 <= 5	V
=	==	Usado para comparar se um valor igual a outro	5 = 5	V
<>	!=	Usado para comparar se um valor é diferente de outro	5 <> 5	F

Operadores

- Prioridade de operações
  - 1º) Parênteses do interno para o externo
  - 2º) Funções matemáticas (exponenciação, radiciação, etc.)
  - 3º) * / mod
  - 4º) + -
  - 5º) > >= < <= = <>
  - 6º) Operador de atribuição ←

Operadores Relacionais

- Comparação entre **caracteres e string** significa verificação em **ordem alfabética** de acordo com a tabela ASCII, ou seja, letras maiúsculas vem antes de letras minúsculas.
- **Exemplos**
  - ‘A’ > ‘a’ F
  - ‘A’ >= ‘a’ F
  - ‘A’ < ‘a’ V
  - ‘A’ <= ‘a’ V
  - ‘A’ = ‘a’ F
  - ‘A’ <> ‘a’ V

Operadores Relacionais

- **Resolva a expressão dada:**

6 * 7 MOD 3 + 12 >= **7 * 6** MOD 3 + 12

42 MOD 3 + 12 >= **42 MOD 3** + 12

0 + 12 >= 0 + 12

12 >= 12

V

“ANA MARIA BRAGA” > “ana”

F

Exercícios

- Resolva as expressões a seguir:
  a) 16+7-2*(3 MOD 12) <> 12*7-12*3+5
  b) (12*3-2*(3MOD12))>=(12 + 23 * 13 MOD2)

Exercícios

- **Solução a):**

16+7-2*(3 MOD 12) <> 12*7-12*3+5

23 – 2 * 3 <> 84 – 36 + 5

23 – 6 <> 48 + 5

17 <> 53

V

Exercícios

- **Solução b):**

(12*3-2*(3MOD12))>=(12 + 23 * 13 MOD2)

(36 – 2 *3) >= (12 + 299 MOD 2)

(36 – 6) >= (12 + 1)

30 >= 13

V

Operadores Lógicos

- **Operadores Lógicos:**
  - Os operadores lógicos nos auxiliam na composição de expressões lógicas complexas, a partir de expressões lógicas simples.
  - Os operandos que compõem a expressão lógica, com o uso dos operadores lógicos, são sempre valores lógicos (V ou F). Portanto, o uso dos operadores lógicos resulta sempre em um valor lógico (V ou F).

Operadores Lógicos

Algoritmo	Linguagem C	Função	Resultado
E	&&	Conjunção	Basta um operando ser FALSO para que o resultado da expressão seja FALSO
OU	||	Disjunção	Basta um operando ser VERDADEIRO para que o resultado a expressão seja VERDADEIRA
NÃO	!	Negação (Inversão)	Inverte ou NEGA o valor do operando

Operadores

- Prioridade de operações
  - 1º) Parênteses do interno para o externo
  - 2º) Funções matemáticas (exponenciação, radiciação, etc.)
  - 3º) * / mod
  - 4º) + -
  - 5º) > >= < <= = <>
  - 6º) NÃO
  - 7º) E
  - 8º) OU
  - 9º) Operador de atribuição ←

Operadores Lógicos

Tabela Verdade

Operador Lógico **E**

Suponha A e B dois valores Lógicos

A	B	A **E** B
V	V	**V**
V	F	**F**
F	V	**F**
F	F	**F**

Operadores Lógicos

Tabela Verdade

Operador Lógico **OU**

Suponha A e B dois valores Lógicos

A	B	A **OU** B
V	V	**V**
V	F	**V**
F	V	**V**
F	F	**F**

Operadores Lógicos

Tabela Verdade

Operador Lógico **NÃO**

A	NÃO(A)
V	**F**
F	**V**

Suponha A um valor

Lógico

Operadores

- **Prioridade de operação – todos os operadores:**
  - 1º) Parênteses (interno para externo)
  - 2º) Operadores Aritméticos
  - 3º) Operadores Relacionais
  - 4º) Operadores lógicos
  - 5º) Operador de atribuição

Operadores

- **Resolva a expressão dada:**

Não (Não (**Não (V)**)) **E** “Carlos” >= “Ana” **OU** **3 * 5** – 7 >= 2 + **17 MOD 3**

Não (**Não (F)**) **E** “Carlos” >= “Ana” **OU** 15 – 7 >= **2 + 2**

Não (V) E “Carlos” >= “Ana” OU 8 >= **4**

F **E** V **OU** V

F **OU** V

V

Exercícios

1) **(12*3-2*7<=4*9-2*11) E NÃO(7*2+3<=34*2) OU NÃO(NÃO(VERDADEIRO))**
2) **NÃO(35MOD7*2 >= 12MOD3*16)**
3) **NÃO(22.1+13.7 = 3.4*5.0+12.2 OU NÃO(3.0*2.1<12.2))**

Exercícios

- Solução 1:

(12*3-2*7<=4*9-2*11) E NÃO(7*2+3<=34*2) OU NÃO(NÃO(VERDADEIRO))

(36 – 14 <= 36 – 22) E NÃO(14 + 3 <= 68) OU NÃO(F)

(22 <= 14) E NÃO(17 <= 68) OU V

F E NÃO(V) OU V

F E F OU V

F OU V

V

Exercícios

- Solução 2:

NÃO(35MOD7*2 >= 12MOD3*16)

NÃO(0 * 2 >= 0 * 16)

NÃO(0 >= 0)

NÃO(V)

F

Exercícios

- Solução 3:

NÃO(22.1+13.7 = 3.4*5.0+12.2 OU NÃO(3.0*2.1<12.2))

NÃO(35.8 = 17.0 + 12.2 OU NÃO(6.3 < 12.2))

NÃO(35.8 = 29.2 OU NÃO(V))

NÃO(F OU F)

NÃO(F)

V

Estrutura do Algoritmo

Identificação do Algoritmo	→	**Algoritmo nome algoritmo;**
Tipos construídos pelo usuário	→	Estrutura dos registros
Constantes/Variáveis globais	→	const/var
Módulos	→	Módulos
Inicio do algoritmo	→	**Início**
Declaração das constante	→	const
		Declaração das constante
Declaração das variáveis	→	var
		Declaração das variáveis
Comando e estrutura do algoritmo	→	Comando e estrutura do algoritmo
Fim do Algoritmo	→	**Fim.**

Estrutura do Programa – Linguagem C

Inclusão das Bibliotecas	→	**#include <biblioteca>**
Tipos construídos pelo usuário	→	struct
Constantes/Variáveis globais	→	const/Variáveis globais
Módulos	→	Módulos
Inicio do algoritmo	→	**main( )**
		{
Declaração das constante	→	const Declaração das constante
Declaração das variáveis	→	Declaração das variáveis
Comando e estrutura do do algoritmo	→	**Comando e estrutura programa principal**
Fim do Algoritmo	→	**}**

Inclusão de Bibliotecas

- A inclusão da biblioteca (#include <biblioteca>) deve ser a primeira linha de comando executável do programa.

- Bibliotecas são programas em Linguagem C que contêm funções pré-definidas da Linguagem e auxiliam o Compilador a entender cada uma dessas funções e executá-las corretamente.

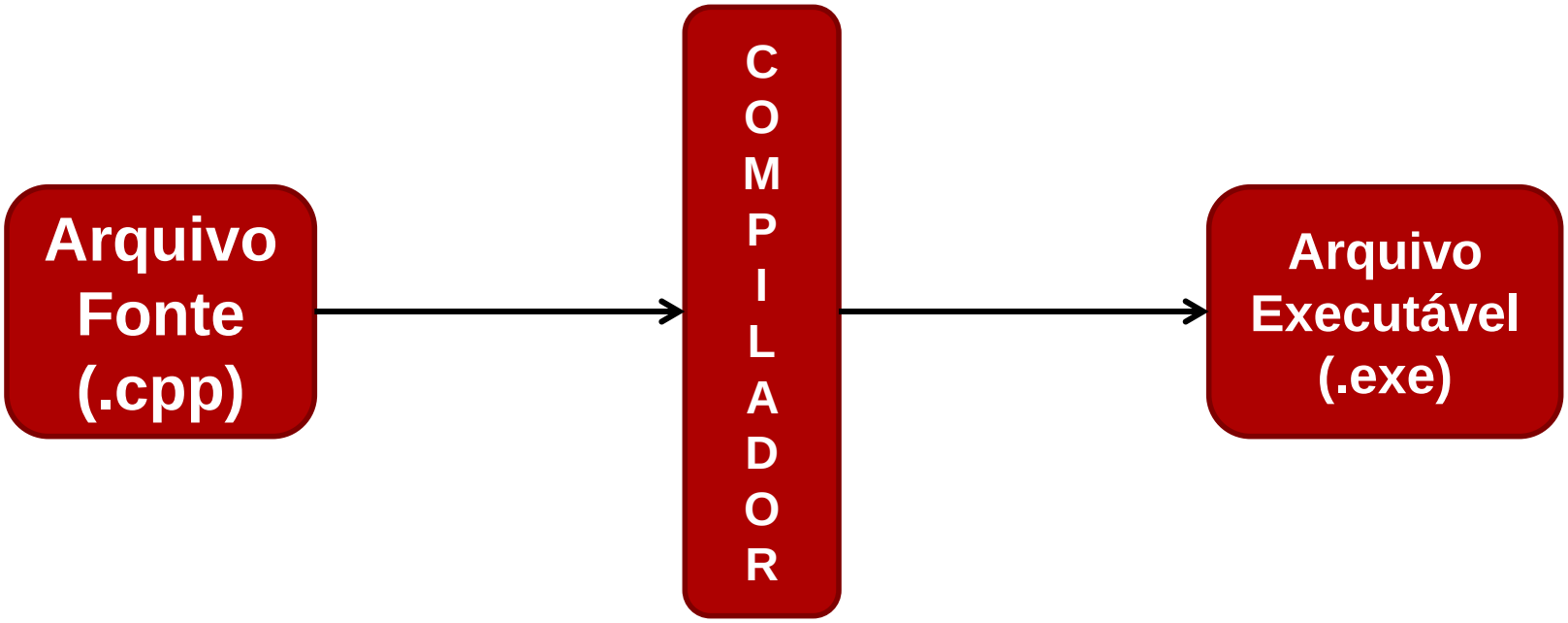
Tipos de Arquivos

- **Arquivo fonte:** é um arquivo não formatado que contém o programa escrito na Linguagem de Programação. É o “texto” do Programa.

- **Arquivo executável:** é o Arquivo Fonte compilado e transformado em Linguagem de Máquina. É o arquivo que o computador compreende e executa.

Processo de Compilação

- **Compilador:** é um programa que transforma o Arquivo Fonte em Arquivo executável. Sua função é “traduzir” cada um dos comandos da Linguagem em códigos executáveis (bits). Há um Compilador para cada Linguagem de Programação.



Estrutura Sequencial

- A estrutura sequencial dos algoritmos ou programas corresponde ao conjunto de ações (comandos e estrutura) que serão executados em uma sequência linear de cima para baixo, da esquerda para direita. Isto é, na mesma ordem em que foram escritas.
- Em linguagem C, a execução do programa começa sempre pelo **programa principal (main()).**

Comandos de Entrada e Saída de Dados

- Os Algoritmos/Programas necessitam de informações para que possam realizas processamento de operações e cálculos são necessário para obter os resultados desejados. Com esta finalidade utiliza-se os comandos de entrada e saída de dados.

- **Comando de saída de dados:** São usados para fazermos a comunicação da máquina com o humano. A função do comando de saída de dados é mostrar os resultados do processamento e também mostrar mensagem que orientam o usuário no uso da aplicação.
- Com este comando podemos:
  - Mostrar o valor armazenado em variável.
  - Mostrar mensagens de comunicação com o usuário.
  - Mostrar mensagens e valores armazenados em variável.

Comando de Saída de Dados

- **Sintaxe – Algoritmo**

escreva (x); // Mostra o Valor armazenado na variável X

escreva(“BOM DIA”); // Mostra a mensagem “BOM DIA”

escreva(“MEU NOME É ”, nome, ” nasci no ano ”, anoNasc);

// Mostra a mensagem mais o valor armazenado na variável

Comando de Saída de Dados

- **Comando de saída de dados – Linguagem C**
  - O comando de saída de dados padrão da Linguagem C tem como parâmetro apenas **uma** **String** e as variáveis que serão mostradas, respeitando a ordem de exibição da esquerda para direita.
  - Todos os valores armazenados nas variáveis, para serem exibidos pelo comando de saída de dados, deverão ser inseridos na mensagem (***string***) nos locais onde colocamos os **marcadores de tipo**.
- **Marcadores de tipo:**
  - **%d** – indica um número inteiro
  - **%f** – indica um número real
  - **%c** – indica um caracter
  - **%s** – indica uma String

Comando de Saída de Dados

- **Sintaxe – Linguagem C**

  printf(“%d”, x); // Mostra o Valor armazenado na variável X

  printf(“Bom Dia”); // Mostra a mensagem “BOM DIA”

  printf(“Meu nome é %s Nasci no ano %d”, nome, anoNasc);

  // Mostra a mensagem mais o valor armazenado na variável

- Na linguagem C os comandos de entrada e saída de dados pertencem a **biblioteca stdio.h**

Meu primeiro algoritmo

- Faça um Algoritmo para mostrar na tela a mensagem: “HELLO WORLD!!! MY FIRST ALGORITHM!!!”.

```
Algoritmo Hello World;
Início
    escreva(“HELLO WORLD!!! MY FIRST ALGORITHM!!!”);
Fim.
```

Meu primeiro programa

- Faça um Programa em Linguagem C para mostrar na tela a mensagem: “HELLO WORLD!!! MY FIRST PROGRAM!!!”.

```
#include<stdio.h>
main( )
{
    printf(“HELLO WORLD!!! MY FIRST PROGRAM!!!”);
}
```

Exercícios – Comando de Saída de Dados

1. Sabendo que A, B, C são três variáveis que armazenam respectivamente 9, 17 e -6. Faça um Algoritmo/Programa para calcular A + B, B * C, C – A, A + C / B. Mostre os resultados.
2. Sabendo que o salário de um funcionário é R$ 2.300,00 e que recebe ao mês seu salário mais 4,0% de comissão sobre o salário. Faça um Algoritmo/Programa para calcular e mostrar o valor da comissão e o valor do salário final desse funcionário.
3. Sabendo que uma criança pesa 23,5Kg. Faça um Algoritmo/Programa que calcule e mostre o peso em gramas. Sabendo que 1 Kg corresponde a 1000g.
4. Sabendo que uma pessoa nasceu em 1988 e que o ano atual é 2018, faça um Algoritmo/Programa que calcule e mostre:
   - A idade da pessoa em anos;
   - A idade da pessoa em meses;
   - A idade da pessoa em dias;
   - A idade da pessoa em semanas.

Exercícios – Saída de Dados

5) Faça um algoritmo para calcular a multiplicação entre as variáveis a, b e c, sabendo que elas armazenam respectivamente 12, -7, 15.

6) Sabendo que o salario de uma pessoa é R$ 2.500,00 e que este mês ela vendeu R$ 10.000,00, faça um algoritmo para calcular e mostrar o salário final sabendo que ele recebe 7,5% de comissão sobre as vendas realizadas e desconta 3% de imposto.

7) Faça um algoritmo para calcular e mostrar X^Y, sabendo que X armazena 12 e Y armazena 3.

8) Faça um algoritmo para calcular e mostrar a média de prova e a média final. Para a média de prova use o cálculo da média aritmética, para a média final use a média ponderada aplicada a nossa disciplina. Atribua valores para as notas de prova e média de trabalho.

Solução Exercício 1

```
Algoritmo Operações;
Início
    var
        inteiro: a, b, c, somaAB, multiplicaBC, subtraiCA, somaDivisaoACB;
    a ← 9;
    b ← 17;
    c ← -6;
    somaAB ← a + b;
    subtraiCA ← c – a;
    multiplicaBC ← b * c;
    somaDivisaoACB ← a + c / b;
    escreva(“Soma = “, somaAB, “Subtração = “, subtraiCA,
    “Multiplicação =“, multiplicaBC,  “Soma e Divisão = “,
    somaDivisaoACB);
Fim.
```

Solução Exercício 1

```
#include <stdio.h>
main()
{
        int a, b, c, somaAB, multiplicaBC, subtraiCA, somaDivisaoACB;
    a = 9;
    b = 17;
    c = -6;
    somaAB = a + b;
    subtraiCA = c – a;
    multiplicaBC = b * c;
    somaDivisaoACB = a + c / b;
    printf(“Soma =  %d \n  Subtração =  %d \n  Multiplicação =
    %d \n Soma e Divisão = %d \n“, somaAB,  subtraiCA,
    multiplicaBC,  somaDivisaoACB);
}
```

Solução Exercício 2

```
Algoritmo Comissão;
Início
    var
        real: salario, comissao, salarioFinal;
    salario ← 2300.00;
    comissao ← salario * 0.04;
    salarioFinal ← salario + comissao;
    escreva(“Salário R$ “, salario, “Comissão R$ “, comissao, “Salário Final R$ “,
    salarioFinal);
Fim.
```

Solução Exercício 3

```
Algoritmo Peso em Gramas;
Início
    const
        real: GRAMAS = 1000.00;
    var
        real: pesoKg, pesoG;
    pesoKg ← 23.5;
    pesoG ← pesoKg * GRAMAS;
    escreva("Peso da criança em Kg  ", pesoKg, " Kg");
    escreva("Peso da criança em G  ", pesoG, " G");
Fim.
```

Solução Exercício 4

```
Algoritmo Idade;
Início
   var
        inteiro: anoAtual, anoNasc, idadeAnos, idadeMeses, idadeDias,
        idadeSemanas;
    anoAtual ← 2018;
    anoNasc ← 1988;
    idadeAnos ← anoAtual - anoNasc;
    idadeMeses ← idadeAnos * 12;
    idadeDias ← idadeAnos * 365;
    idadeSemanas ← idadeDias / 7;
    escreva(“Idade em Anos = “, idadeAnos, “ Idade em Dias = ”,
    idadeDias, “Idade em Meses = “, idadeMeses, “Idade em Semanas =“,
    idadeSemanas);
Fim.
```

Solução Exercício 6

```
Algoritmo Salário Final;
Início
    var
        real: salario, vendas, comissao, imposto, salarioFinal;
    salario ← 2500.00;
    vendas ← 10000.00;
    comissao ← vendas * 0.075;
    salarioFinal ← salario + comissao;
    imposto ← salarioFinal * 0.03;
    salarioFinal ← salarioFinal - imposto;
    escreva(“Salário R$“, salario, “ Comissão R$”, comissao,
    “Imposto R$”, imposto, “Salário Final R$”, salarioFinal);
Fim.
```

Comando de Entrada de Dados

- **Comando de entrada de dados**: este comando permite ao usuário inserir informações que serão armazenadas e processadas pela aplicação. Portanto, a função do comando de entrada de dados é **“ler”** o valor que será armazenado na variável especificadas no comando.
- **Sintaxe – Algoritmo**

  leia (nomeVar);
- **Exemplos**

  leia(idade);
  leia (sexo);
  leia (nota);

Comando de Entrada de Dados

- **Linguagem C**
- Para que o comando de entrada de dados funcione corretamente, é necessário indicar o tipo da variável que está sendo **“lida”**. Isto é feito usando os **marcadores de tipos**:

  %d – número inteiro

  %f – número real

  %c – um caracter

  %s – uma string

Comando de Entrada de Dados

- Também é necessário indicar que o valor “lido” será armazenado no **endereço de memória** da variável especificada. O operador **&** indica o endereço de memória.
- **Sintaxe – Linguagem C**

  scanf(“marcador de tipo”, &nomeVar);
- **Exemplos:**

  scanf(“%d”, &idade);

  scanf(“%c”, &sexo);

  scanf(“%f”, &nota);

- Em linguagem C, o comando de entrada de dados scanf() pertence à biblioteca **stdio.h**.

Comando de Entrada de Dados

- **Exemplo**
- Faça um algoritmo para ler 3 Nº inteiros. Mostre na tela os 3 Nº na ordem inversa a ordem digitada.
- Ações:
  - Ler 3 números
  - Mostrar os números lidos em ordem inversa a ordem digitada
- Variáveis:
  - A, B e C inteiro

Comando de Entrada de Dados

```
Algoritmo Ordem Inversa;
Início
  var
      inteiro: a, b, c;
  leia (a);
  leia (b);
  leia (c);
  escreva("Ordem digitada:  ", a, b, c);
  escreva ("Ordem Inversa a ordem digitada:  ", c, b, a);
Fim.
```

Comando de Entrada de Dados

```
#include <stdio.h>
#include <stdlib.h>
main( )
{
   int a, b, c;
   printf(“\n Digite um número: ”);
   scanf(“%d”, & a);
   printf(“\n Digite outro número: ”);
   scanf(“%d”, & b);
   printf(“\n Digite outro número: ”);
   scanf(“%d”, & c);
   printf(“\n Ordem digitada: %d  %d  %d”, a, b, c);
   printf(“\n Ordem inversa: %d  %d  %d”, c,  b, a);
   system (“pause”);
}
```

Comando de Entrada de Dados

- Faça um algoritmo para ler um número real e mostrar na tela o valor lido e o endereço de memória da variável.
- **Ações**
  - Ler um número
  - Mostrar o valor lido e o endereço de memória
- **Variável**
  - num Real

Comando de Entrada de Dados

Algoritmo Endereço de Memória;

Início

```
  var
        real: num;
  escreva (“Digite um Nº real”);
  leia (num);
  escreva (“Valor armazenado”, num, ”Endereço de memória da variável ”, & num);
```

Fim.

Comando de Entrada de Dados

```
#include <stdio.h>
#include <stdlib.h>
main( )
{
  float num;
  printf(“Digite um Nº real”);
  scanf(“%f”,  &num);
  printf(“Valor armazenado: %f  \n Endereço de memória da variável: %d”, num ,&num);
  system(“pause”);
}
```

Exercícios - Comando de Entrada de Dados

1) Faça um algoritmo/programa para ler 3 números inteiros. Calcule e mostre:
   a) A soma do 1º número digitado pelo 2º
   b) A subtração do 2º número digitado pelo 3º
   c) A multiplicação do 3º número digitado pelo 1º

2) Faça um algoritmo/programa que leia duas notas, calcule e mostre a média ponderada dessas notas, sabendo que os pesos devem ser dados pelo usuário.

3) Faça um algoritmo/programa que leia o ano de nascimento de uma pessoa e o ano atual, calcule e mostre:
   a) A idade dessa pessoa em anos;
   b) A idade dessa pessoa em meses;
   c) A idade dessa pessoa em dias;
   d) A idade dessa pessoa em semanas.

Exercícios - Comando de Entrada de Dados

4) Faça um algoritmo para ler três números inteiros e calcule a multiplicação entre eles.

5) Faça um algoritmo para ler o salário base de um vendedor e o total de vendas realizadas no mês. Calcule e mostre o salário final sabendo que ele recebe 7,5% de comissão sobre as vendas realizadas e desconta 3% de imposto.

6) Faça um algoritmo para ler dois números reais X e Y. Calcule e mostre X^Y.

7) Faça um algoritmo para ler duas notas de prova, e a média de trabalho. Calcule e mostre a média de prova e a média final. Para a média de prova use o cálculo da média aritmética, para a média final use a média ponderada aplicada a nossa disciplina.

Solução Exercício 1

```
Algoritmo Operações;
Início
    var
        inteiro: a, b, c, somaAB, subtraiBC, multiplicaCA;
    leia(a);
    leia(b);
    leia(c);
    somaAB ← a + b;
    subtraiBC ← b – c;
    multiplicaCA ← c * a;
    escreva(“soma = “, somaAB, “subtracão = “, subtraiBC, “multiplicação = “,
    multiplicaCA);
Fim.
```

Solução Exercício 2

```
Algoritmo Média Ponderada;
Início
    var
        real: nota1, nota2, peso1, peso2,  mediaPonderada;
    leia(nota1);
    leia(nota2);
    leia(peso1);
    leia(peso2);
    mediaPonderada ← (nota1 * peso1 + nota2 * peso2) / (peso1 + peso2);
    escreva("média ponderada = ", mediaPonderada);
Fim.
```

Solução Exercício 3

```
Algoritmo Cálculo Idade;
Início
    var
        inteiro: anoNasc, anoAtual, idadeAnos, idadeMeses, idadeSemanas, idadeDias;
    leia(anoNasc);
    leia(anoAtual);
    idadeAnos← anoAtual - anoNasc;
    idadeMeses← idadeAnos * 12;
    idadeDias← idadeAnos * 365;
    idadeSemanas← idadeDias / 7;
    escreva("Idade = ", idadeAnos, "anos Idade = ", idadeMeses, "meses Idade = ",
    idadeSemanas, "semanas Idade = ", idadeDias, "dias");
Fim.
```

Solução Exercício 5

```
Algoritmo Salário Final;
Início
    var
        real: salarioBase, salarioFinal, totalVendas, comissao, impostos;
    leia(salarioBase);
    leia(totalVendas);
    comissao ← totalVendas * 0.075;
    salarioFinal ← salarioBase + comissao;
    impostos ← salarioFinal * 0.03;
    salarioFinal ← salarioFinal – impostos;
    escreva(“Salário Base = “, salarioBase, “Total de Vendas = “, totalVendas,
    “Comissão = “, comissao, “Impostos = “, impostos, “Salário Final = “,
    salarioFinal);
Fim.
```

Bibliografia

- **Básica**

  ASCENCIO, A. F. G, CAMPOS, E. A. V. **Fundamentos da Programação de Computadores**: algoritmos, Pascal e C/C++ e Java. Longman, 2007.

  FORBELLONE, L. V., EBERSPACHER, H. F. **Lógica de Programação:** a construção de algoritmos e estruturas de dados. Prentice Hall, 2005.

  ZIVIANI, Nivio. **Projeto de Algoritmos com Implementações em Pascal e C**. 2.ed. Thomson Pioneira, 2004.

- **Complementar**

  FARRER, H et al. **Algoritmos estruturados**. 3 ed. Rio de Janeiro: LTC, 1999. 284 p.

  MANZANO, J. A. N. G.; **Estudo dirigido de algoritmos**. 9. ed. São Paulo: Érica, 2004.

  LOUNDON, L. **Algoritmos em C**. São Paulo: Ciência Moderna, 2000.

  ASCENCIO, A. F. G.; CAMPOS, E.A.V. **Fundamentos da programação de computadores:** algoritmo, pascal e C++. São Paulo: Pearson Prentice Hall, 2002. 355 p