Lógica de programação

Conceitos iniciais

Prof. Kleber Petry

Metodologia de avaliação

Carga horaria 120H

- Prova 50%
- Trabalho 40%
- Presença em sala 10%

kleber.petry@pr.senai.br

Afinal o que é informática

Cada vez mais estamos conectados a informática, em casa o trabalho ou em momento de lazer.

Muitas vezes nem percebemos a quantidade de tempo que destinamos a informática

Curiosidade:

Informática = informação + automática

Era da informação

Com o avanço tecnológico cada vez mais criamos dependência de dados e informações precisas e em tempo real.

Entendemos que **Dados** são coleções de itens ou valores simplesmente armazenados sem qualquer tratamento ou processamento.

Informação: é o resultado do processamento, manipulação e organização de dados de forma que represente uma modificação (quantitativa e qualitativa) no conhecimento do sistema (seja pessoa, sistema ou máquina)

Como funciona o computador

Esquema básico de funcionamento do computador

Parte física = Hardware

Parte lógica= Software

Números e a invenção do computador

Vídeo bit e bytes

Prof. Kleber Petry

Fundamentos de programação

- Computadores são máquinas de propósito geral que não tem uma finalidade especifica.
- O computador deve ser programado, ou seja deve ter uma sequencia de comandos que faça ele realizar atividades uteis ou resolver um problema computacional.

Lógica

Lógica é a técnica de encadear pensamentos para atingir determinado objetivo.

Prof. Kleber Petry

Noções de Lógica

- Exemplos de aplicação da lógica
  - O quarto está fechado e que meu livro está no quarto. Então, preciso primeiro abrir o quarto para pegar o livro
  - Rosa é mãe de Ana, Paula é filha de Rosa, Júlia é filha de Ana. Então, Júlia é neta de Rosa e sobrinha de Paula
  - Todo mamífero é animal e todo cavalo é mamífero. Então, todo cavalo é animal
  - Todo mamífero bebe leite e o homem bebe leite. Então, todo homem é mamífero e animal (mas não é um cavalo)

Instruções

Instruções são um conjunto de regras ou normas definidas para a realização ou emprego de algo. Em informática, é o que indica a um computador uma ação elementar a executar.

- INSTRUÇÃO:
  - Cada um dos *passos*, cada uma das ações a tomar (obedecendo a *seqüência lógica*) para ir resolvendo o problema, ou para ir executando a tarefa
  - Em informática, é a informação que indica a um computador uma *operação elementar* a executar
    - Ex.: “somar”, “subtrair”, “comparar se é maior”, etc
  - Uma só instrução não resolve problemas

  - Executar um *conjunto de instruções*
  - Executar em uma *seqüência lógica*

Seqüência Lógica e Algoritmo

- Seqüência Lógica são passos executados até atingir um objetivo ou solução de um problema.

- Um algoritmo é formalmente uma seqüência finita de passos que levam a execução de uma tarefa. Podemos pensar em algoritmo como uma receita, uma seqüência de instruções que dão cabo de uma meta específica.

Exemplo de algoritmo

- “Chupar uma bala”
    - Pegar a bala
    - Retirar o papel
    - Jogar o papel no lixo
    - Chupar a bala
- Pode-se escrever este algoritmo de outra forma?
- Como ficaria se o pote de balas tivesse balas de diversos sabores?

Como desenvolver um programa?

Um programa usualmente vai ter a finalidade especifica de automatizar algum processo ou resolver um problema.

Exemplo: Em uma loja, cada item tem um preço. Como calcular o valor a ser cobrado por *N* itens de produtos? Para resolver isso sem o uso de um computador, podemos simplesmente multiplicar o preço unitário do produto pelo números de itens.

E como podemos transportar essa solução para um programa ou linguagem de programação.

Critérios para um bom programa

- Clareza: assim como no português, existem inúmeras maneiras de transmitir uma informação.
- Endentação: a endentação se refere a organização visual do código, na língua portuguesa utilizamos parágrafos, recuos, marcadores e numerações para facilitar o entendimento de um texto.
- Comentários no código: a maioria das linguagens permite que o programador inclua comentários juntamente as linhas de códigos, esses comentários são ignorados pelo computador e somente utilizados para aumentar a clareza do código.
- Modularidade: manter o código limpo, claro e objetivo, refatorar sempre!

Como se tornar um bom programador?

- Pratique
- Pratique bastante
- Pratique bastante e tente errar pouco
- Pratique bastante, tente errar pouco e leia muitos outros códigos
- Pratique bastante, tente errar pouco, leia muitos códigos e sempre fique atento a novas tendências.
- Um bom programador pode levar até 10 anos para obter um lógica perspicaz e automática.

Tipos de algoritmo

- Naturais: conjunto de passos necessários para para realizar determinadas ações no nosso dia a dia.
- Computacionais: conjunto de passos automatizados que resolvem problemas do nosso dia a dia.

Como um algoritmo pode ser representado?

- Nós podemos representar um algoritmo da maneira que acharmos melhor, desde que tal representação seja bem estruturada e organizada. Porém, as representações mais utilizadas são a de Fluxograma e de Pseudocódigo

Algoritmo Textual Informal

- Modo de preparo:
  - Bata a margarina, as gemas e o açúcar até ficar cremoso
  - Junte o leite, o coco e a farinha e continue batendo
  - Acrescente o fermento e, por último, as claras em neve
  - Unte uma forma com manteiga e leve ao forno para assar

Qução cremoso?!?

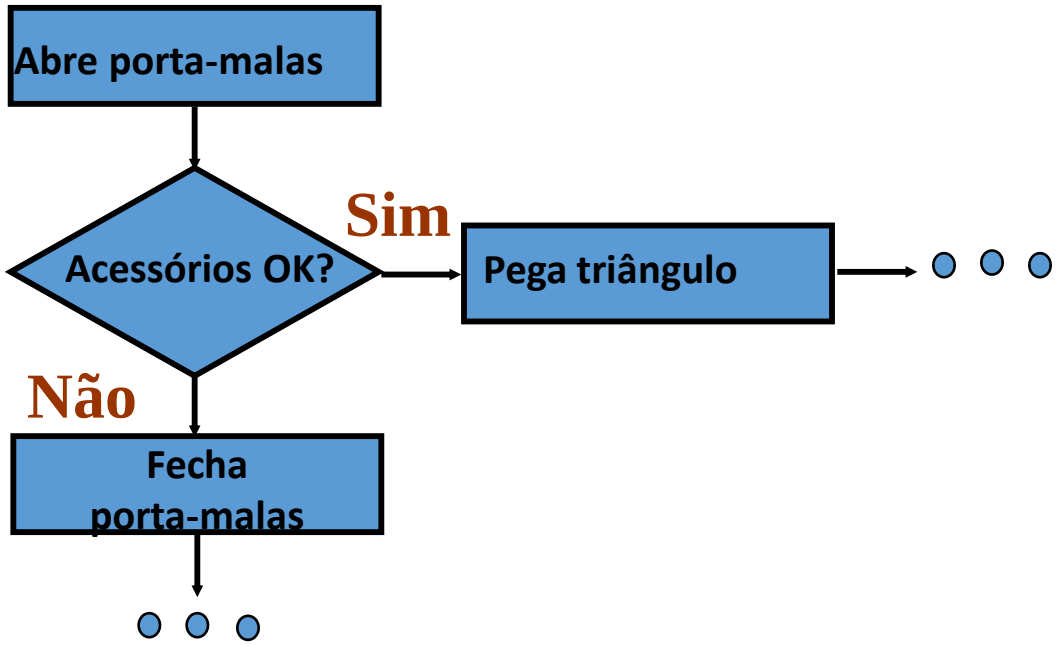
Quanto tempo?!?

De uma vez só?!?

Quanto tempo?!?

Algoritmo Gráfico Semi-formal

- Troca de pneu (Fluxograma)



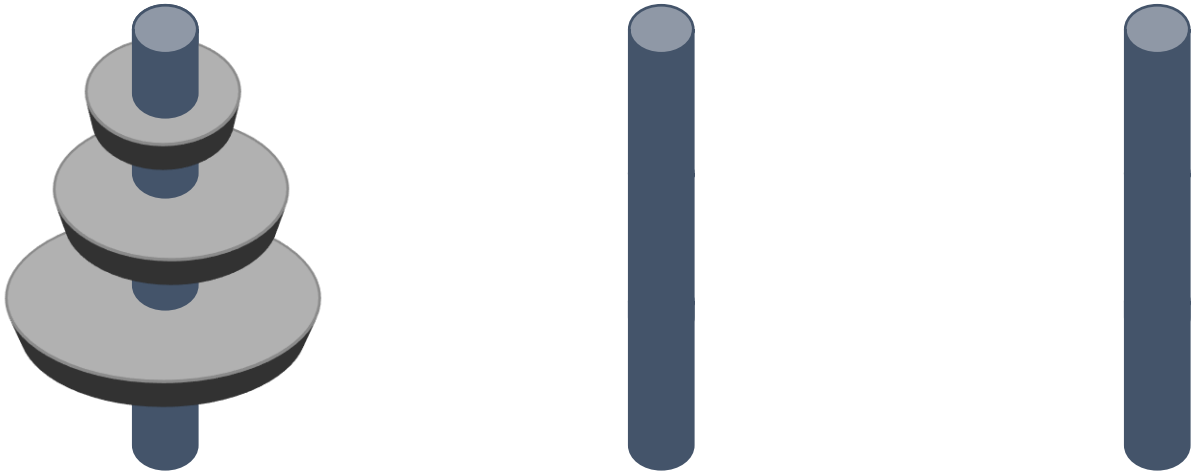
Algoritmo Textual Formal

- Troca de pneu

```
abre(porta_malas)
Se acessorio_ok = FALSO
Então
   fecha(porta_malas)
   espera_carona()
Senão
   pega_triangulo()
                  ...
```

Algoritmo: Problemas Complexos

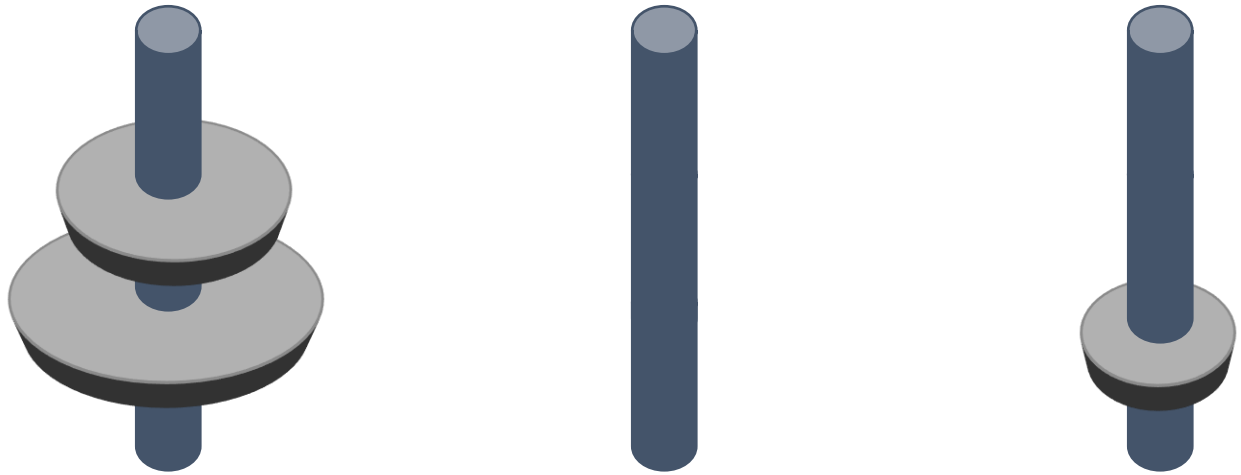
- Problema da Torre de Hanói
  - Seja a seguinte situação:
    - deve-se mover todos os discos do primeiro eixo para o terceiro mantendo-se a ordem original
    - em cada movimento, pode-se mover apenas um disco
    - um disco nunca poderá ser sobreposto por outro maior



Algoritmo: Problemas Complexos

- Passo 1:

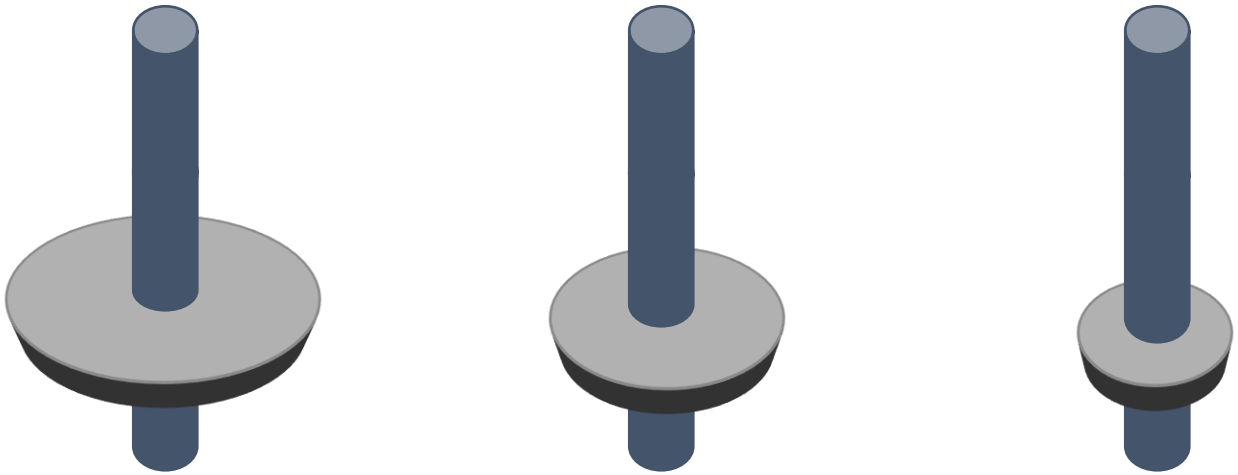
```
mova disco menor para terceiro eixo
```



Algoritmo: Problemas Complexos

- Passo 2:

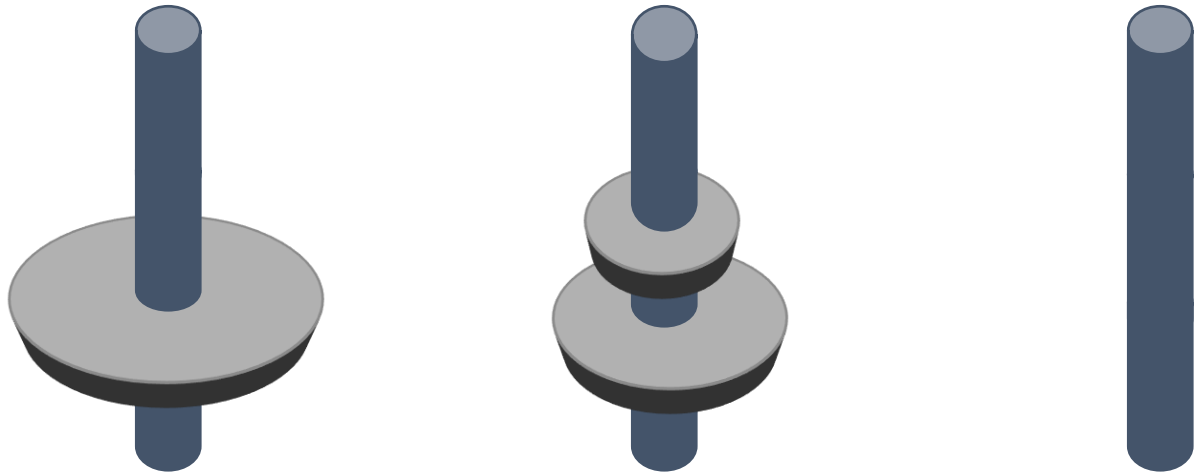
```
mova disco médio para segundo eixo
```



Algoritmo: Problemas Complexos

- Passo 3:

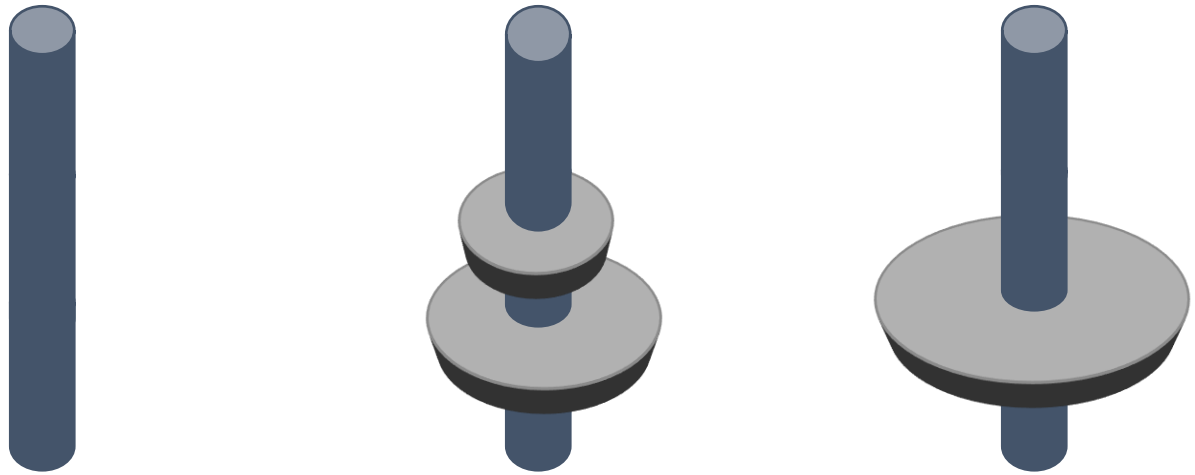
```
mova disco menor para segundo eixo
```



Algoritmo: Problemas Complexos

- Passo 4:

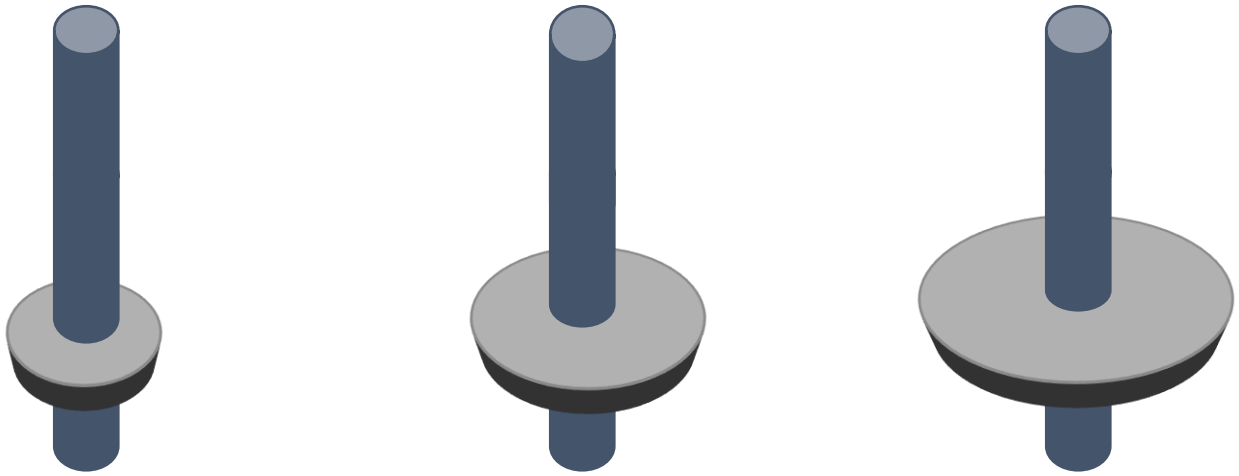
```
mova disco maior para terceiro eixo
```



Algoritmo: Problemas Complexos

- Passo 5:

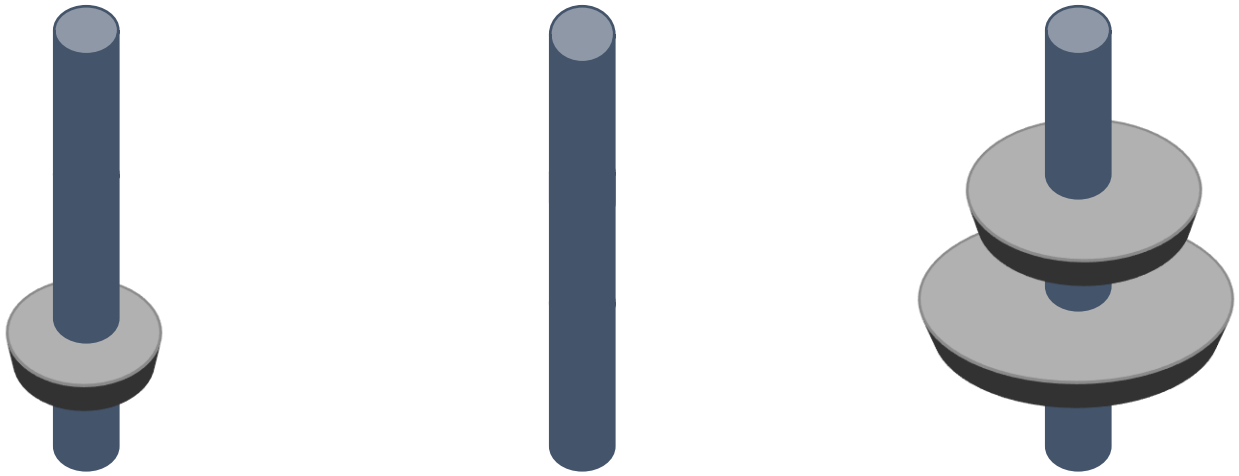
```
mova disco menor para primeiro eixo
```



Algoritmo: Problemas Complexos

- Passo 6:

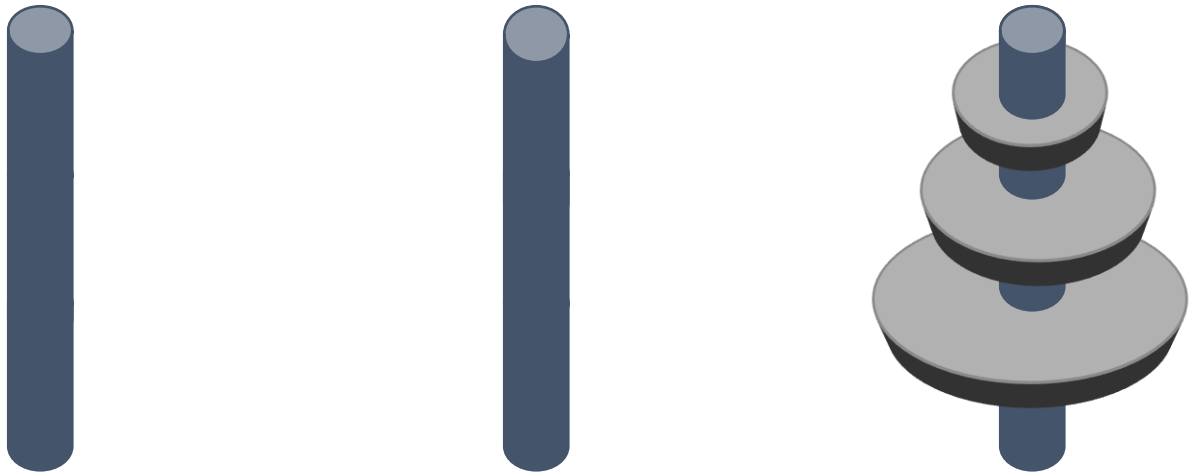
`mova disco médio para terceiro eixo`



Algoritmo: Problemas Complexos

- Passo 7:

```
mova disco menor para terceiro eixo
```



Fluxograma

- O fluxograma é um dos métodos mais utilizados para se representar um algoritmo. Trata-se de uma espécie de diagrama e é utilizado para documentar processos (simples ou complexos). Tal tipo de diagrama ajuda o leitor a visualizar um processo, compreendê-lo mais facilmente e encontrar falhas ou problemas de eficiência.

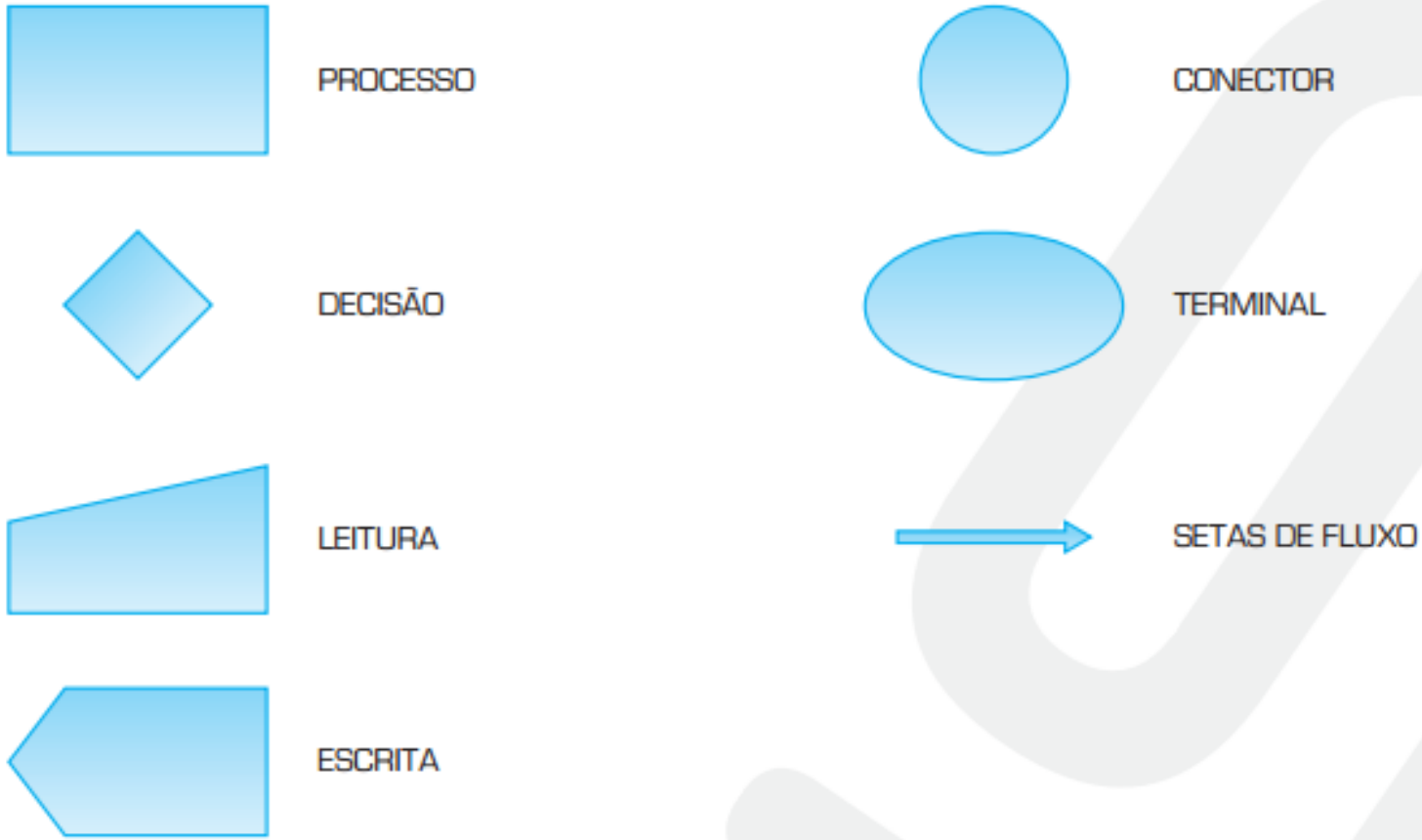


Figura 2.2: Símbolos utilizados em um fluxograma

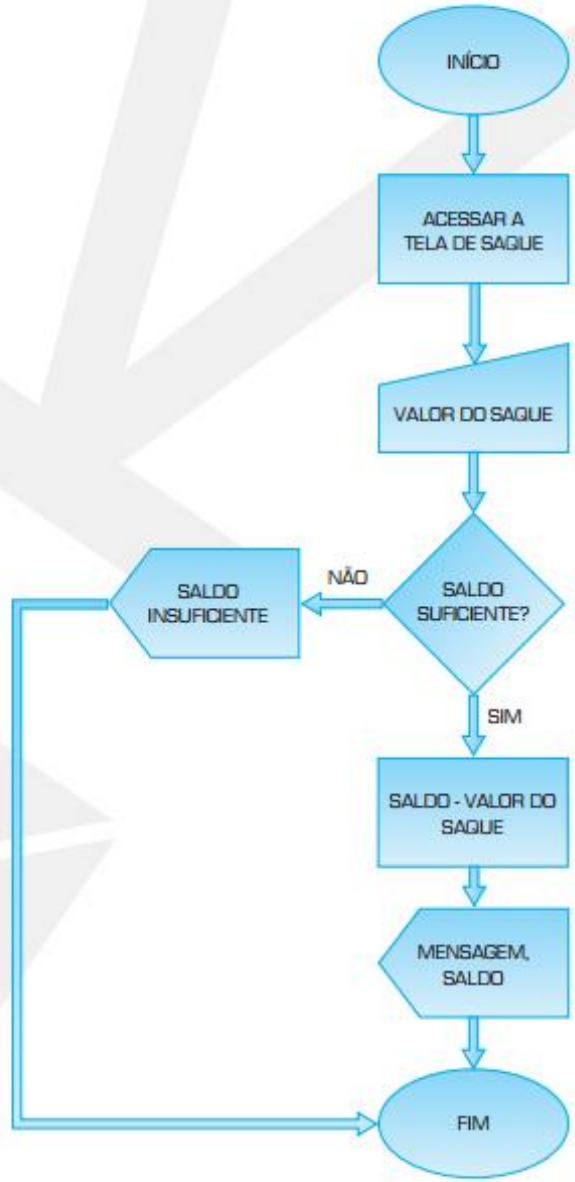


Figura 2.3: Exemplo de fluxograma para a operação de saque em um caixa eletrônico de um banco

Pseudocódigo

- Escrever um algoritmo em pseudocódigo é outra forma muito utilizada por autores de livros que tratam de algoritmos, pois dessa forma o leitor não precisa ter o conhecimento prévio de nenhuma linguagem de programação. Nos países cujo idioma principal é o português, muitos se referem ao pseudocódigo como português estruturado ou portugol.

Pseudocódigo - Exemplo

- Achar o maior de dois números A e B.

```
Início
    Declare A,B;   { Declaração de variáveis }
    Leia(A,B);
    Se A = B Então Escreva("A e B iguais");
    Senão  Se A>B Então Escreva("A é maior");
                Senão Escreva("B é maior");
                Fim-Se
    Fim-Se
Fim.
```

```
1 INICIO
2    LER(ValorDoSaque)
3    SE ValorDoSaque > 0 E ValorDoSaque <= Saldo ENTÃO
4       Saldo = Saldo - ValorDoSaque;
5       ESCREVER("Saque efetuado com sucesso. Saldo atual: ", Saldo);
6    SENÃO
7       ESCREVER("Saldo Insuficiente.");
8    FIM SE
9 FIM
```

Pseudocódigo 2.1: Exemplo de pseudocódigo para a operação de saque em um caixa eletrônico.

Desafio

Atribua uma lógica do seguinte jogo “Transporte pelo rio”.

O jogo consiste em atravessar todos os personagens de uma margem à outra do rio seguindo as seguintes regras:

- 1. Lobo não pode ficar sozinho com a cabra.
- 2. Cabra não pode ficar sozinho com o maço de alface.

Sites complementares

- Code.org

Código de tutor

QYRXNB

Khan academy

pt.khanacademy.org

Código de tutor

PKX5C7

Fluxograma

Ele deve conter INICIO E FIM

Identificadores: são os nomes dados as variáveis, constantes, entre outras, e devem seguir algumas regras;

- Deve se iniciar por letra
- Pode conter números ou _ mas não pode conter caractere especial
- Não pode ser uma palavra reservada, exemplo: inicio, fim, se etc
- Não existe distinção para maiúsculo ou minisculo
- Não deve ser utilizado acento

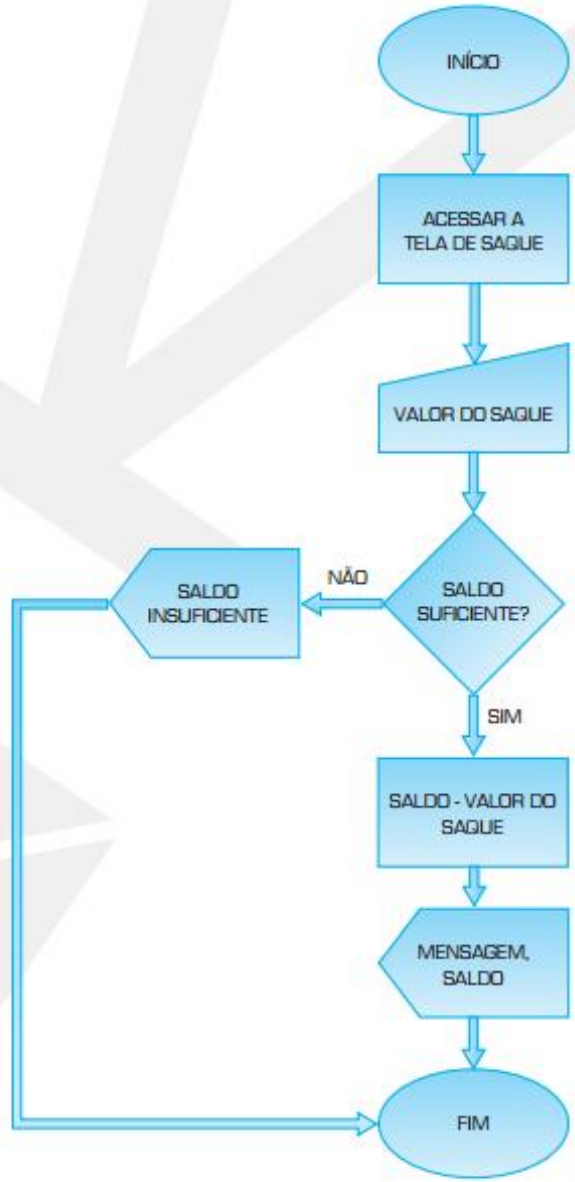


Figura 2.3: Exemplo de fluxograma para a operação de saque em um caixa eletrônico de um banco

Oque é uma Variável?

- Os dados manipulados por um programa são armazenados em variáveis. Normalmente, uma variável é associada a uma posição da memória RAM. Nas variáveis é possível armazenar dados de vários tipos: numéricos, strings (texto), booleanos (verdadeiro ou falso), referências, entre outros.
- Toda variável possui um nome (um identificador). Os nomes das variáveis são utilizados para manipular os dados contidos nelas. Como, normalmente, as variáveis são associadas à posições da memória RAM, os identificadores das variáveis funcionam como nomes simbólicos dos endereços da memória RAM.

Tipos de dados para uma variável

Mais utilizados

- Float : Valores numéricos
- String: Valores texto
- Char: Valores condições

String

- O tipo string é um dos mais importantes e mais utilizados. O tipo string é usado para o armazenamento de texto (sequência de caracteres).
- Observe, nos exemplos abaixo, que o texto que deve ser armazenado nas variáveis é definido dentro de aspas duplas.

VARIÁVEIS

```java
String texto = "K19 Treinamentos";
```

Código Java 3.3: Tipo String em Java

```csharp
string texto = "K19 Treinamentos";
```

Código C# 3.3: Tipo string em C#

Constantes

Alguns valores apesar de variáveis, podem ser fixos para uma execução do algoritmo.

Exemplo: o valor do imposto que, para um determinado algoritmo é sempre 15%, pode ter seu valor alterado para uma nova lei. Caso o valor de 15% tenha sido inserido diretamente no programa, haverá necessidade de alterar diversos pontos do programa para atualiza-lo, dificultando a sua manutenção.

Para isso podemos fazer o uso de constantes, que são valores que não se alteram ao longo da execução do programa e devem ser declarados na seção especifica de declaração de constantes.

Declaração de variáveis

A variável dependendo da linguagem pode ser identificado de varias formas, no geral ela se estrutura da seguinte forma.

<identificador [,identificador...]>:<tipo>

Var Idade: inteiro

Declarando variáveis

Para criar uma variável em Java ou C#, é necessário declará-la. Nessas duas linguagens de programação, para declarar uma variável é necessário informar o seu tipo e o seu nome (identificador).

```java
int numeroDaConta;
double saldo;
boolean contaAtiva;
```

Código Java 3.1: Declaração de variáveis em Java.

```csharp
int numeroDaConta;
double saldo;
bool contaAtiva;
```

Código C# 3.1: Declaração de variáveis em C#.

Atribuição

- Após declararmos uma variável e antes de utilizá-la, devemos inicializá-la para evitarmos um erro de compilação.

```java
int numeroDaConta;
numeroDaConta = 3466;

boolean contaAtiva = true;
```

Código Java 3.2: Declaração e inicialização de variáveis em Java.

```csharp
int numeroDaConta;
numeroDaConta = 3466;

bool contaAtiva = true;
```

Código C# 3.2: Declaração e inicialização de variáveis em C#.

Tipos de Operadores

- Para manipular as variáveis de uma aplicação, devemos utilizar os operadores oferecidos pela linguagem de programação que estamos utilizando. As linguagens possuem diversos operadores. Os principais operadores são:
- Aritmético
- Atribuição
- Relacional
- Lógico

Operadores Aritméticos

Os operadores aritméticos funcionam de forma muito semelhante aos operadores da matemática. Os operadores aritméticos são:

- Adição +
- Subtração –
- Multiplicação *
- Divisão /
- Potenciação ^,**
- Divisão inteira Div
- Resto da divisão Mod

Comando de entrada

O comando **Ler** permite que as informações digitadas no teclado sejam armazenadas nas variáveis do algoritmo.

Ex:

Ler(<variável>,..,variável>)

Ler (nome,endereço)

Ler(salario)

Comando de saída

O comando **escrever** é utilizado para apresentar mensagens, conteúdo de variáveis ou resultado de expressões na tela do computador e pode ser escrito da seguinte forma:

Ex:

Escrever(<mensagem, variável, expressão>)

Escrever(‘Resultado:’, resultado)

Escrever(‘informe um valor’)

Teste de mesa

Após escrever o seu algoritmo é fundamental testá-lo para ver se funciona corretamente antes de considera-lo pronto. O teste de mesa é a execução passo a passo do algoritmo como se ele fosse executado no computador.

- Facilita o entendimento do fluxo de execução do algoritmo
- Permite identificar erros lógicos na construção do algoritmo
- Possibilita a verificação da evolução do conteúdo das variáveis.

Algoritmo sequencial

Em um algoritmo sequencial, os comandos são executados na sequência em que aparecem. Cada comando é executado após o termino do anterior sem omissões nem repetições.

- Atribuição
- Entrada
- Saída de dados

Operadores

- Usados para incrementar, decrementar, comparar e avaliar dados, que são operações básicas em processamento de dados.
- Tipos:
  - Aritméticos (+, -, *, /, ** ou ^)
    - Resultados numéricos
  - Relacionais (>, <, >=, <=, =, <> ou #)
    - Resultados lógicos (V ou F)
  - Lógicos (e, ou, não)
    - Combinam resultados lógicos

Precedência dos operadores

- Operadores **relacionais** são muito usados quando temos que tomar decisões nos algoritmos. Com eles fazemos testes, comparações, que resultam em valores lógicos (verdadeiro ou falso):

Descrição	Símbolo
Igual a	=
Diferente de	<> ou #
Maior que	>
Menor que	<
Maior ou igual a	>=
Menor ou igual a	<=

Exemplo:
tendo duas variáveis, A = 5 e B = 3:

Expressão	Resultado
A = B	Falso
A <> B	Verdadeiro
A > B	Verdadeiro
A < B	Falso
A >= B	Verdadeiro
A <= B	Falso

Exercícios

Sabendo que A=3, B=7 e C=4, informe se as expressões abaixo são verdadeiras ou falsas.

a) (A+C) > B ( )

b) B >= (A + 2) ( )

c) C = (B –A) ( )

d) (B + A) <= C ( )

e) (C+A) > B ( )

- Operadores **lógicos** combinam resultados lógicos, gerando novos valores lógicos (verdadeiro ou falso). A “tabela-verdade” abaixo mostra todos os valores possíveis de se obter com oper. lógicos:

			Resultado
T	AND	T	T
T	AND	F	F
F	AND	T	F
F	AND	F	F
T	OR	T	T
T	OR	F	T
F	OR	T	T
F	OR	F	F
	NOT	T	F
	NOT	F	T

T = Verdadeiro
F = Falso
AND = E
OR = OU
NOT = NÃO

Considere a seguinte atribuição de valores para as variáveis:
A=3, B=4, C=8. Avalie as expressões a seguir indicando o resultado final: verdadeiro ou falso.

1) A > 3 **E** C = 8 ( )
2) A <> 2 **OU** B <= 5 ( )
3) A = 3 **OU** B >= 2 **E** C = 8 ( )
4) A = 3 **E NÃO** B <= 4 **E** C = 8 ( )
5) A <> 8 **OU** B = 4 **E** C > 2 ( )
6) B > A **E** C <> A ( )
7) A > B **OU** B < 5 ( )
8) A <> B **E** B = C ( )
9) C > 2 **OU** A < B ( )
10) A > B **OU** B > A **E** C <> B ( )

Sabendo que A=5, B=4 e C=3 e D=6, informe se as expressões abaixo são verdadeiras ou falsas.

a) (A > C) **AND** (C <= D) ( )

b) (A+B) > 10 **OR** (A+B) = (C+D) ( )

c) (A>=C) **AND** (D >= C) ( )

Sabe-se que o uso incorreto da precedência de operadores ocasiona erros. Pensando nisso, determine o resultado das expressões a seguir (valores: A= 8, B = 5, C = -4, D = 2)

a) Delta = $B^2 – 4 * A * C$

b) J = “Hoje” <> “HOJE”

c) Media = (A + B + C + D) / 4

d) Media = A + B + C + D / 4

e) Resultado = A + B – 10 * C

f) Y = A > 8 E B + C > D

g) Y = A > 3 * 2 OU B + C <> D

Comando de seleção IF

Comandos de seleção ou instrução de decisão

O comando de seleção **Se..Então** associa a execução de um comando a uma determinada condição.

Quando a condição associada à cláusula é verdadeira, o comando **então** é executado; se a condição for falsa, o comando não será executado.

Seleção composta IF - ELSE

A seleção composta nos permite, além de associar um fluxo de execução para o caso de uma condição ser verdadeira, associar também outro fluxo de execução no caso da condição ser falsa.

A seleção composta faz uso do comando **se..então..então**.

Comando de seleção ou instrução de decisão

- Considere um parque de diversões como os da Disney. Nesses parques, para garantir a segurança, alguns brinquedos possuem restrições de acesso. Em geral, essas restrições estão relacionadas à altura dos visitantes. Em alguns parques, a altura do visitante é obtida por sensores instalados na entrada dos brinquedos e um programa de computador libera ou bloqueia o acesso de acordo com altura obtida. Então, o programa deve decidir se executa um trecho de código de acordo com uma condição. Essa decisão pode ser realizada através das instruções de decisão oferecidas pelas linguagens de programação.

Instrução if

- A instrução if (se), é utilizada quando queremos testar uma condição antes de executarmos um
- ou mais comandos. A sintaxe da instrução if é a seguinte:

```
1 if(condição) {
2   // comando 1
3   // comando 2
4   // comando 3
5 }
6 // comando 4
7 // comando 5
```

Instrução else

Muitas vezes, queremos executar um bloco de comandos caso uma condição seja verdadeira e outro bloco de comandos caso essa condição seja falsa. Para isso, podemos utilizar as instruções if e else. Veja abaixo, a estrutura dessas instruções.

```
1  if(condição) {
2    // comando 1
3    // comando 2
4    // comando 3
5  } else {
6    // comando 4
7    // comando 5
8    // comando 6
9  }
10 // comando 7
```

Instruções de Decisão Encadeadas

- Considere um programa de computador que controla os saques efetuados nos caixas eletrônicos de um banco. Nesse banco, os saques efetuados das 6 horas até as 22 horas não podem ser superiores a R$ 5.000,00. Por outro lado, os saques efetuados depois das 22 horas e antes das 6 horas não podem ser superiores a R$ 400,00. Podemos implementar essa lógica utilizando as instruções de decisão oferecidas pelas linguagens de programação.

```
1  if(hora >= 6 && hora <= 22) {
2    if(valor <= 5000) {
3      System.out.println("Saque efetuado com sucesso");
4    } else {
5      System.out.println("Valor máximo de saque é R$ 5000,00");
6    }
7  } else {
8    if(valor <= 400) {
9      System.out.println("Saque efetuado com sucesso");
10   } else {
11     System.out.println("Valor máximo de saque é R$ 400,00");
12   }
13 }
```

Comando caso seja - CASE

O comando **caso-seja** é um comando de seleção. As alternativas do comando **caso** podem ser valores, lista de valores ou intervalos, e a clausula **senão** é opcional neste comando. Os comandos da causa **senão** serão executados se o resultado da variável ou expressão não for igual às alternativas anteriores.

- Exemplo:
  - Entre com número da opção (OP).
  - Selecione: caso OP igual a 1, título “opção 1”; caso OP igual a 2; título “opção 2”; caso OP igual a 3, título “opção 3”; caso OP igual a 4, título “opção 4”; caso OP igual a 5, título “opção 5”; senão titulo igual a “opção errada”.

```
TITULO = “”
OP = INPUTBOX(“DIGITE A OPÇÃO”)
SELECT CASE OP
    CASE 1
        TITULO = “OPÇÃO 1”
    CASE 2
        TITULO = “OPÇÃO 2”
    CASE 3
        TITULO = “OPÇÃO 3”
    CASE 4
        TITULO = “OPÇÃO 4”
    CASE 5
        TITULO = “OPÇÃO 5”
    CASE ELSE
        TITULO = “OPÇÃO ERRADA”
END SELECT
```