Linguagens de Programação I

Introdução a Algoritmos e Lógica de Programação

INTRODUÇÃO

- Que é um programa de computador?
- Um programa de computador é o produto resultante da atividade intelectual de um **programador**.
- Para criar um programa de computador devemos ter conhecimentos em:
  - abstração e modelagem de problemas
  - linguagens e ferramentas de programação
  - uso da lógica na verificação das soluções

INTRODUÇÃO

- Um **programa de computador** é um conjunto de **instruções** e **dados** que algum ser humano define e que ao serem executadas por um **computador** cumprem algum objetivo.
- **instruções**: pequenas tarefas ou operações que a máquina deve realizar, geralmente modificam dados.

INTRODUÇÃO

- **dados**: valores armazenados no computador, utilizados para alcançar o objetivo do programa,
- **dados de entrada**: são fornecidos ao programa por um ser humano ou dispositivo,
- **dados de saída**: resultados oferecidos pelo computador após o processamento dos dados de entrada.

ENGENHARIA DE SOFTWARE

- O desenvolvimento de um programa ou software deve ser encarado como um processo bem definido de engenharia.
- O desenvolvimento de um software e definido nas seguintes etapas:

1. **Análise**: criam-se as especificações que detalham como o software vai a funcionar,

ENGENHARIA DE SOFTWARE

2. **Projeto**: criam-se especificações que detalham o resultado da análise em termos próximos da implementação do software (criação do algoritmo),
3. **Implementação**: utilizando-se uma linguagem de programação e as especificações de projeto, o software é construído,
4. **Testes**: após a construção do software, são realizados testes para conferir a conformidade com os requisitos iniciais.

ALGORITMO

- Um algoritmo representa um conjunto de regras que fornecem a solução de um problema (definição geral), pode ser aplicada a qualquer problema.
- Ex: fritar um ovo.
- Em programação, um algoritmo especifica com clareza e forma correta as instruções que um software devera conter, para que, ao ser executado, forneça os resultados esperados.

ALGORITMO

- Como criar um algoritmo?
- **Modelagem e Implementação**
- **Modelagem**
  - conhecer o problema a ser resolvido (entender o problema),
  - extrair todas as informações ao respeito do problema (dados e operações),
  - se necessário, buscar informações em outras fontes

ALGORITMO

- Como criar um algoritmo?
- **Implementação**
  - descrever claramente os passos para chegar a solução,
  - organizar os passos segundo uma seqüência lógica que leve a solução.

ALGORITMO

- **Exemplo**: crie um algoritmo para calcular a área de um triangulo de base *b* e altura *h*.

$$A = bh/2$$

Inicio

1. **Pedir ao usuário fornecer os valores de *b* e *h***
2. **Calcular a área *A* usando a formula**
3. **Exibir o valor de *A* na tela**

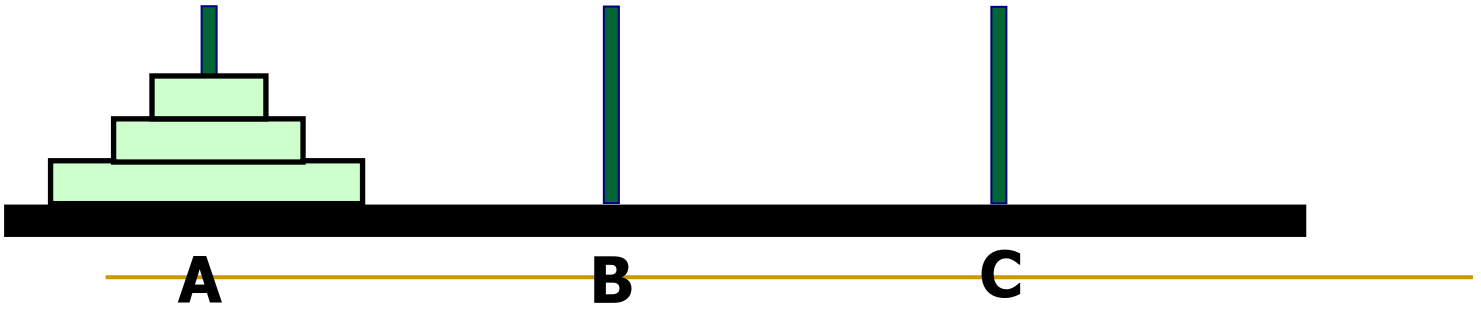
Fim

ALGORITMO

- **Exercício 1**: *Escreva um algoritmo para, dado um jogo de cartas, o usuário escolhe um naipe e vc mostra sua posição no baralho.*

ALGORITMO

- **Exercício 2**: *Escreva um algoritmo para resolver o seguinte problema. Têm-se três hastes A, B e C, na haste A repousam três anéis de diâmetros diferentes, em ordem crescente por diâmetro. Transfira os anéis de A para B, usando C se necessário. Considere: deve-se mover um único anel por vez, um anel de diâmetro maior no pode repousar sobre outro de diâmetro menor.*



ALGORITMO

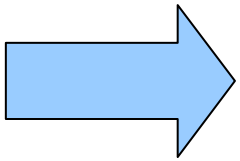
- Segundo o dicionário Aurélio:

***Matemática**: Processo de cálculo ou de resolução de um grupo de problemas semelhantes, em que se estipulam, com generalidade e sem restrições, regras formais para o obtenção do resultado ou da solução do problema.*

ALGORITMO

- Segundo o dicionário Aurélio:

***Informática**: Conjunto de regras e operações bem-definidas e ordenadas, destinadas à solução de um problema ou de uma classe de problemas, em um número finito de passos.*

Algoritmo  ***Caminho de solução para um problema***

ALGORITMO

- Características de um algoritmo:

1. Um algoritmo representa uma seqüência de regras,
2. Essas regras devem ser executadas em uma ordem preestabelecida,
3. Cada algoritmo possui um conjunto finito de regras,
4. Essas regras devem possuir um significado e ser formalizadas segundo alguma convenção.

PROPRIEDADES - ALGORITMO

- **Valores de entrada**: Todo algoritmo deve possuir zero, uma o mais entradas. [Ex]
- **Valores de saída**: Todo algoritmo possui uma o mais saídas que simboliza(m) seu(s) resultado(s) [Ex]
- **Finitude**: Toda tarefa a ser realizada possui um início, meio e fim. Os algoritmos representam a solução de um problema, também possuem início, meio e fim.

PROPRIEDADES - ALGORITMO

- **Finitude** ... Todo algoritmo deve ser finito, deve possuir um conjunto de passos que ao serem executados, levarão sempre ao seu término ou fim.

  Deve-se prestar especial atenção a esta propriedade, freqüentemente criamos algoritmos que nunca chegaram a um resultado, tornando-se infinito. [Ex]

PROPRIEDADES - ALGORITMO

- **Passos elementares**: Um algoritmo deve ser explicitado por meio de *operações elementares*, sem que possa haver diferenças de interpretação, da forma tal que possa ser executado por máquinas sem inteligência (computador).

- *Operações elementares?*

PROPRIEDADES - ALGORITMO

- **Correção**: Um algoritmo deve ser correto, deve permitir que ao ser executado, se chegue às saídas com resultados coerentes com as entradas.

  Para verificar se um algoritmo é correto ou não deve-se fazer uma simulação, testes com diversos valores de entrada, cujas saídas se conhecem a priori e, então, comparar estes resultados com os produzidos pelo algoritmo.

FORMALIZANDO ALGORITMOS

- A tarefa de especificar os algoritmos consiste em detalhar os dados que serão processados pelo programa e as instruções que vão a operar sobre esses dados.
- É importante formalizar a descrição dos algoritmos segundo alguma convenção, para que todas as pessoas possam entendê-lo da mesma forma.
- Para formalizar um algoritmo precisamos definir a sintaxe e a semântica.

FORMALIZANDO ALGORITMOS

- **Regras de sintaxe**: regras que regulam a escrita do algoritmo,
- A sintaxe de um algoritmo resume-se nas regras para escrevê-lo corretamente,
- Essas regras indicam quais são os tipos de comando que podem ser utilizados e também como neles escrever expressões,
- Os tipos de comandos de um algoritmo são também denominados estruturas de programação.

FORMALIZANDO ALGORITMOS

- Existem três tipos de estruturas que podem ser utilizadas: estruturas seqüências, de decisão e de repetição.
- **Regras de semântica**: são as regras que permitem interpretar um algoritmo,
- Os símbolos ou comandos de um algoritmo por se só não tem um significado a menos que este seja bem definido.
- A semântica de um algoritmo sempre acompanha sua sintaxe, fornecendo um significado.

FORMALIZANDO ALGORITMOS

- A importância de formalizar um algoritmo (sintaxe e semântica) pode ser resumida assim:

1. Evitar ambigüidades, pois definimos regras que sempre são interpretadas da mesma forma,
2. Impedir a criação de símbolos ou comandos desnecessários na representação de um algoritmo (conjunto mínimo de regras),
3. Permitir uma aproximação das regras as linguagens de programação, facilitando a codificação do algoritmo.

FORMALIZANDO ALGORITMOS

- Existem diversos mecanismos que formalizam a representação de algoritmos, em nosso curso abordaremos:

1. **Fluxogramas**: representação gráfica.
2. **Portugol**: linguagem para representar algoritmos.

- Introduzimos ambas técnicas através de exemplos.

REPRESENTAÇÂO DE ALGORITMOS

- **Exemplo**: crie um algoritmo para calcular a área de um triangulo de base *b* e altura *h*.

$$A = bh/2$$

- Algoritmo informal:

Inicio

1. Pedir ao usuário fornecer os valores de *b* e *h*

2. Calcular a área *A* usando a formula

3. Exibir o valor de *A* na tela

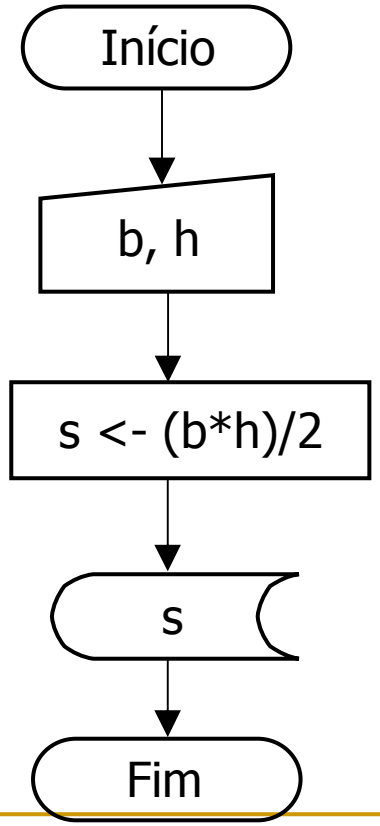
Fim

REPRESENTAÇÂO DE ALGORITMOS

- Portugol:

```
Inicio
  1. Leia(b, h)
  2. s <- (b*h)/2
  3. Exiba(s)
Fim
```

- Fluxograma:



REPRESENTAÇÂO DE ALGORITMOS

- **Exemplo**: *Compraram-se 30 canetas iguais, que foram pagas com uma nota de R$ 100,00; obtendo-se R$ 67,00 como troco. Quanto custou cada caneta?*
- Se paguei R$ 100,00 e recebi como troco R$ 67,00, o custo total das canetas foi

  R$ 100,00 – R$ 67,00 = R$ 33,00
- Para saber quanto paguei por cada caneta dividimos R$ 33,00 por 30,
- Assim, cada caneta custou R$ 1,10

REPRESENTAÇÂO DE ALGORITMOS

- **Exemplo**...

- Podemos ilustrar este raciocínio matematicamente
- Seja *x* o custo de cada caneta
- *quantogastei = 30x*
- *quantogastei + troco = 100,00*
- Resolvemos a equação anterior (quadro)

REPRESENTAÇÂO DE ALGORITMOS

- **Exemplo**...
- Algoritmo informal:

Inicio

1. **Pegar os valores 30, 100 e 67**
2. **Subtrair 67 de 100 e dividir o resultado por 30**
3. **Mostrar o resultado final**

Fim

- Deve-se observar que esse algoritmo resolve apenas uma **instância particular do problema.**

REPRESENTAÇÂO DE ALGORITMOS

- **Exemplo**...
- Se quisermos solucionar o caso geral teríamos o seguinte:
- *Compraram-se N canetas iguais que foram pagas com uma nota de Z reais, obtendo-se Y reais de troco. Quanto custou cada caneta?*
- Portugol:

```
Inicio
  1. Leia(N, Z, Y)
  2. C <- (Z-Y)/N
  3. Exiba(C)
Fim
```

REPRESENTAÇÂO DE ALGORITMOS

- **Exemplo**...
- O algoritmo apresentado tem uma serie de restrições (problemas com os dados de entrada – analisar)
- Para obter um algoritmo consistente devemos levar em consideração as precondições do problema:

1. O valor pago pelas canetas tem que ser maior que o troco recebido
2. O valor pago e a quantidade de canetas tem que ser maiores que zero
3. O troco tem que ser maior ou igual a zero

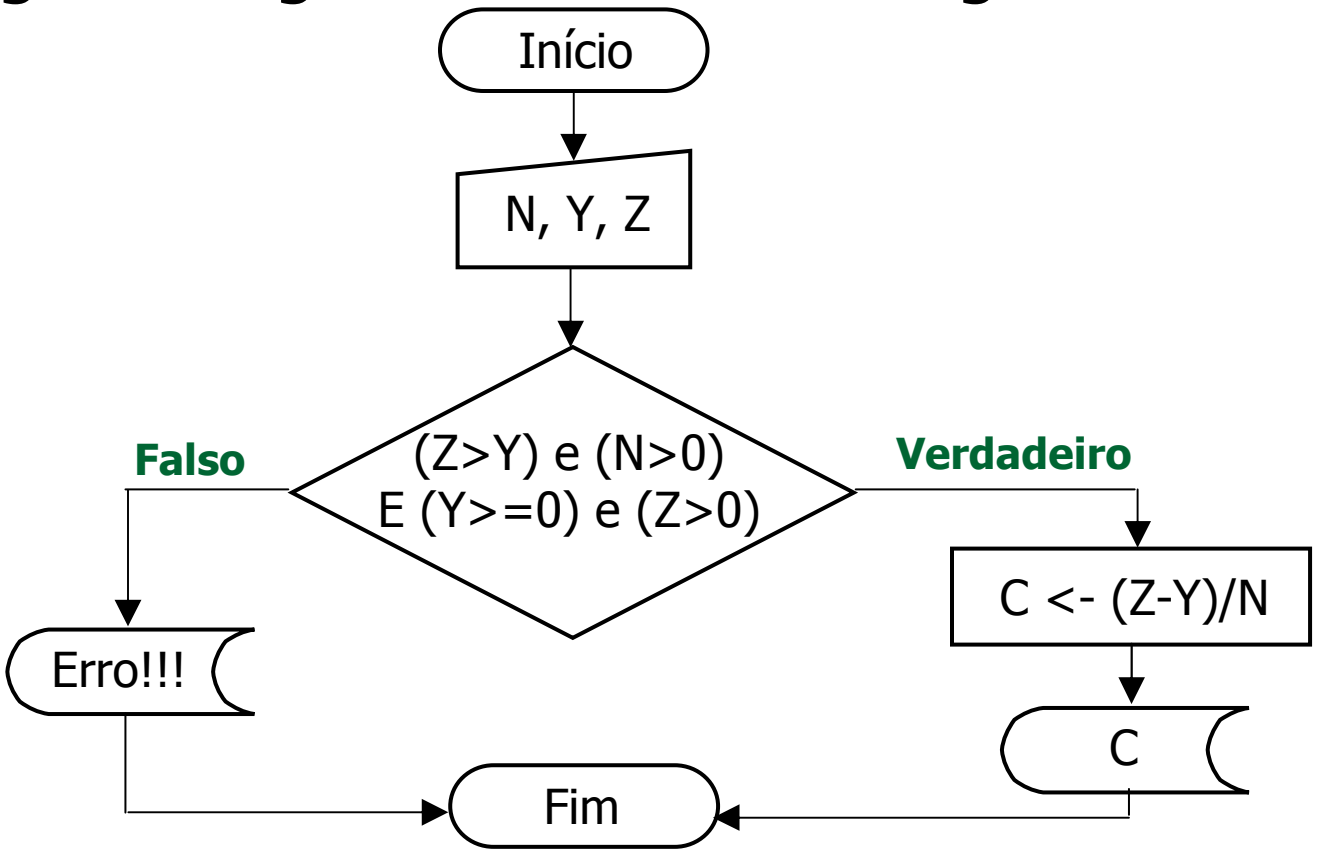
REPRESENTAÇÃO DE ALGORITMOS

- **Exemplo**...
- Algoritmo geral e correto em portugol

```
Inicio
 1. Leia(N, Z, Y)
 2. Se (Z>Y) e (N>0) e (Z>0) e (Y>=0) Então
 3.    C <- (Z-Y)/N
 4.    Exiba(C)
 5. Senão
 6.    Exiba("Erro: Valores inconsistentes!")
 7. Fim Se
Fim
```

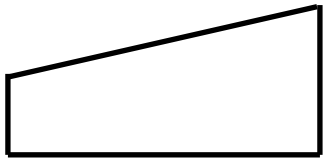
REPRESENTAÇÂO DE ALGORITMOS

- **Exemplo**...
- Algoritmo geral e correto Fluxograma



REPRESENTAÇÂO DE ALGORITMOS

- Símbolos - Fluxograma

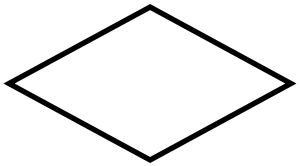
	terminador	Representa a saída ou entrada do ambiente externo
	processo	Representa qualquer tipo de processo (funções, operações)
	linha	Representa o fluxo de dados ou controles
	entrada manual	Representar os dados que sejam fornecidos em tempo de processamento

REPRESENTAÇÂO DE ALGORITMOS

- Símbolos - Fluxograma

exibição

Representa dados que serão mostrados (tela, impressora)



decisão

Representa uma decisão ou desvio (uma entrada; saídas: uma, duas, múltiplas) de acordo com a decisão se tomara apenas uma saída.

REPRESENTAÇÂO DE ALGORITMOS

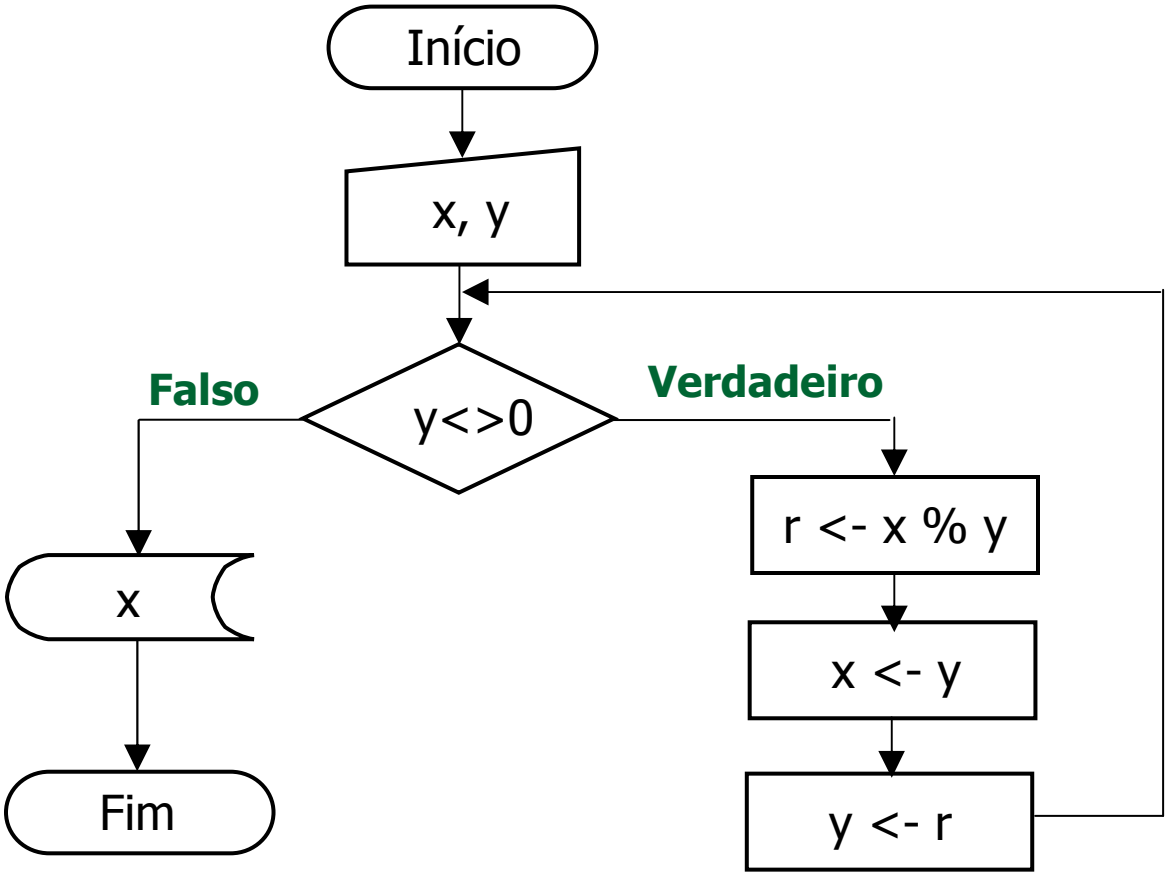
- **Exercício**: Que faz o seguinte algoritmo

```
Inicio
 1. Leia(x, y)
 2. Enquanto (y<>0) Faça
 3.    r <- x % y
 4.    x <- y
 5.    y <- r
 6. Fim Enquanto
 7. Exiba(x)
Fim
```

- Considere *x* e *y* valores inteiros. Elabore o fluxograma do algoritmo anterior.

REPRESENTAÇÂO DE ALGORITMOS

- **Exercício**... Fluxograma



REPRESENTAÇÂO DE ALGORITMOS

- **Exercício**: Crie um algoritmo que calcule quantas notas de 50, 10, 5 e 1 são necessárias para pagar uma conta cujo valor é fornecido. Considere valores inteiros. Utilize alguma das técnicas de representação de algoritmos estudadas (fluxograma ou portugol).

COMO TER SUCESSO EM UM CURSO DE PROGRAMAÇÂO?

- O grande **problema** apresentado pelos estudantes em um primeiro **curso de programação**, não são as características da linguagem, mas sim a **dificuldade** em se **abstrair e descrever** as soluções de problemas contando com poucas e simples estruturas.
- Um novo problema pode ser gerado a partir de um já existente, alterando-se apenas poucos elementos de seu enunciado (Ex).

COMO TER SUCESSO EM UM CURSO DE PROGRAMAÇÂO?

- É um erro decorar as soluções em computação, elas não servem para outros problemas que com certeza serão diferentes.
- Procure o entendimento de como foi obtida uma solução, guarde-lo na memória e utilize essa experiência adaptando-la a outras situações, por **analogia**, **generalização** ou **especialização**.
- Acumule experiência e use-la em novos desafios.

DICAS

1. Ao se deparar com um problema novo, tente entendê-lo:

- O que se deve descobrir ou calcular? (Objetivo)
- Quais são os dados disponíveis? São suficientes?
- Quais as condições necessárias e suficientes para resolver o problema?
- Se possível, modele o problema de forma matemática.

DICAS

2. Crie um plano com a solução:

- Consulte sua memória e verifique se você já resolveu algum problema similar (analogia, generalização, especialização)
- Verifique se é necessário introduzir algum elemento novo no problema, como um problema auxiliar.
- Se o problema for muito complicado, tente quebrá-lo em partes menores e solucionar essas partes.

DICAS

3.Formalize a solução:

- Crie um algoritmo informal com os passos que resolvam o problema.
- Verifique se cada passo do algoritmo esta correto.
- Escreva um algoritmo formalizado (fluxograma ou portugol)

DICAS

4.Exame dos resultados:

- Teste o algoritmo com diversos dados e verifique os resultados (teste de mesa)
- Se o algoritmo não gerou resultado algum. Volte e tente encontrar o erro.
- Se o algoritmo gerou resultados, estes estão corretos?
- Se não estão corretos, alguma condição, operação ou a ordem, estão incorretas. Volte e tente encontre o erro.

DICAS

5.Otimização da solução:

- É possível melhorar o algoritmo?
- É possível reduzir o número de passos ou dados?
- É possível conseguir uma solução ótima?