

```
mirror_mod = modifier_ob.  
set mirror object to mirror.  
mirror_mod.mirror_object =  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
  
#selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
  
print("please select exactly  
  
-- OPERATOR CLASSES ----  
  
types.Operator):  
on X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
  
context):  
context.active_object is n
```

Programação de computadores

Prof. Dr. Josenalde B. Oliveira

josenalde.oliveira@ufrn.br

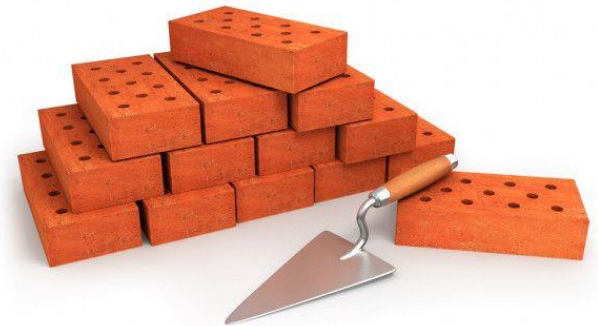
Nosso plano de curso...

- 28.02 – dez (120h aula)

Modelagem de problemas; Estrutura geral de um programa;
Características da **linguagem adotada**, do ambiente de programação e rastreamento de erros; Operadores; Atribuição; Operadores aritméticos; Operadores lógicos; Operadores relacionais; Variáveis simples e tipos primitivos; Comandos de entrada e saída

Estruturas de controle de fluxo de execução de programas; Comandos de seleção; Comandos de repetição; Estruturas de dados homogêneas;

Estruturas de dados heterogêneas, persistência de dados



Nosso plano de curso...

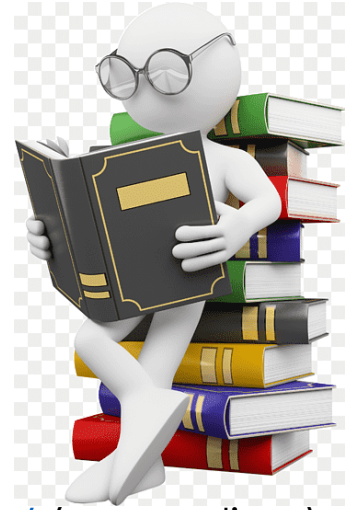
Bibliografia básica

- Javascript Tutorial. Disponível em <https://www.w3schools.com/js/>
- BAZILIO, C. Programando na cozinha. Disponível em <https://carlosbazilio.gitbooks.io/programando-na-cozinha/content/pt-br/> (acesso livre)
- MEDINA, M. Algoritmos e programação: teoria e prática. São Paulo: Novatec, 2006.

Bibliografia complementar

LEPSEN, E. F. Lógica de Programação e Algoritmos com JavaScript. São Paulo: Novatec, 2018.

Lista de exercícios em PRATIQUE OBI: <https://olimpiada.ic.unicamp.br/pratique/> (Júnior, P1,...)





Nosso plano de curso...

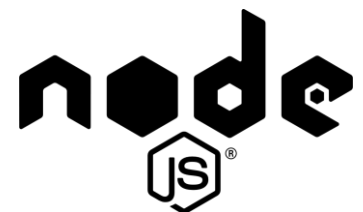
Ambientes de desenvolvimento

- Qualquer navegador interpreta código fonte escrito em Javascript
- Chrome (F12), Firefox, Edge: CTRL+SHIFT+I (Ferramentas do Desenvolvedor, Console)

Do ponto de vista WEB, explorado em Desenvolvimento Web, junto com HTML e CSS

Em Programação de Computadores, pode-se integrar progressivamente a alguma interface com o usuário em navegador, mas o foco é **lógica de programação**. Portanto, podemos escrever nossos códigos para serem interpretados fora do navegador. Numa instalação local, teríamos o **node.js**: nodejs.org/en

Escrever o código fonte num editor de texto qualquer, salvar com extensão .js e abrir um terminal (shell) e digitar node <arquivo.js>





Nosso plano de curso...

Ambientes de desenvolvimento

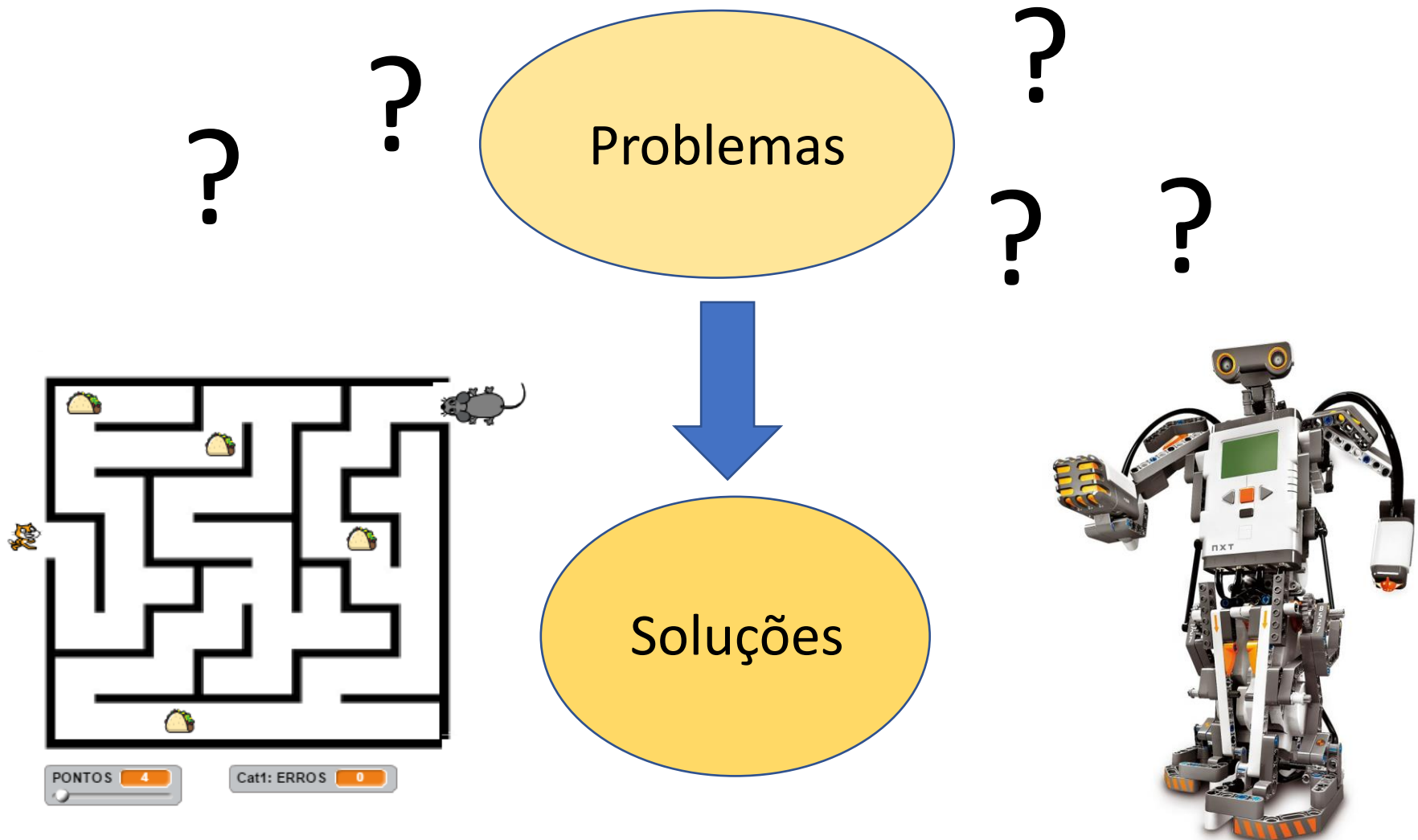
Ou escrever e executar códigos em ambientes online, como:

REPL.IT

https://olimpiada.ic.unicamp.br/saci/cursos/intro_js/1/

<https://www.w3schools.com/js/>

Motivação...



Motivação...

- Podem existir inúmeras soluções para um mesmo problema;
- Em termos computacionais, procuramos explorar as que necessitem do menor número possível de “passos” (linhas de código);
- O estudante de informática/computação é um solucionador em potencial dos mais variados problemas;
- Raciocínio lógico, encadeamento de ideias, enfim, LÓGICA.

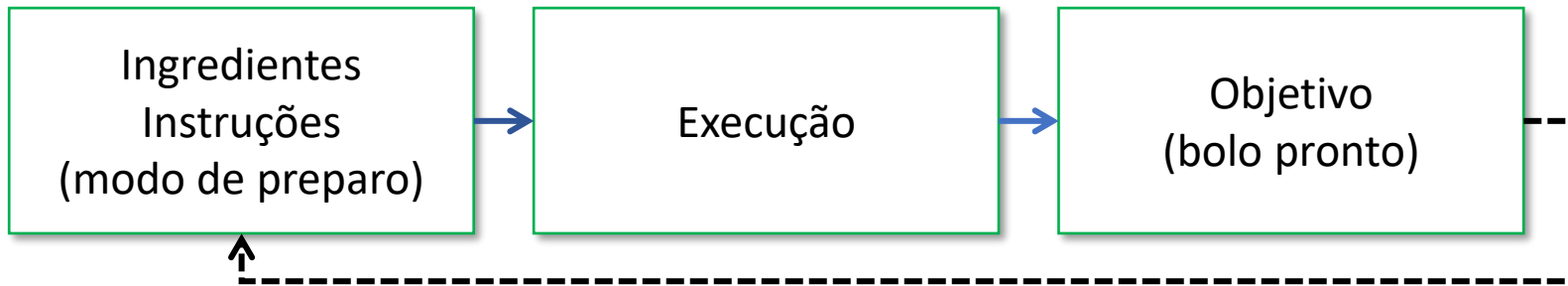
EXEMPLO...

- Você está numa margem de um rio, com três animais: uma galinha, um cachorro e uma raposa. Somente pode atravessar com um animal por vez e nunca deixar a raposa e o cachorro sozinhos nem a raposa e a galinha. Descreva uma forma de conseguir atravessar os três animais, obedecendo a essas condições.



SURGE ENTÃO UM ALGORITMO

- Um **algoritmo** representa um conjunto de **regras** para a **solução** de um problema (descrição geral). Neste sentido, uma receita de bolo é um exemplo de algoritmo.



SURGE ENTÃO UM ALGORITMO

Algoritmo para fazer um bolo simples

- 1 - pegar os ingredientes;
- 2 - se (roupa branca)
 colocar avental;
- 3 - se (tiver batedeira)
 bater os ingredientes na batedeira;
 senão
 bater os ingredientes à mão;
- 4 - colocar a massa na forma;
- 5 - colocar a forma no forno;
- 6 - aguardar o tempo necessário;
- 7 - retirar o bolo;

Outros exemplos...

Algoritmo para trocar lâmpadas

- 1 - se (lâmpada estiver fora de alcance)
 pegar a escada;
- 2 - pegar a lâmpada;
- 3 - se (lâmpada estiver quente)
 pegar pano;
- 4 - tirar lâmpada queimada;
- 5 - colocar lâmpada boa;

Outros exemplos...

Algoritmo para descascar batatas

- 1 - pegar faca, bacia e batatas;
- 2 - colocar água na bacia;
- 3 - enquanto (houver batatas)
 descascar batatas;

Outros exemplos...

Algoritmo para fazer uma prova

```
1 - ler a prova;  
2 - pegar a caneta;  
3 - enquanto ((houver questão em branco) E (tempo  
    não terminou)) faça  
    se (souber a questão)  
        resolvê-la;  
    senão  
        pular para outra;  
4 - entregar a prova;
```

PODERÍAMOS ENTÃO DIZER...

- Que um **programa de computador** nada mais é do que um ou mais algoritmos escritos numa linguagem de programação (C, C++, Pascal, Fortran, Delphi, Java, Javascript, Python, Ruby, Basic, PHP entre outras);
- O mais importante de um programa é sua LÓGICA, o RACIOCÍNIO utilizado para resolver o problema, que é exatamente o **ALGORITMO**.

VAMOS “BRINCAR”...

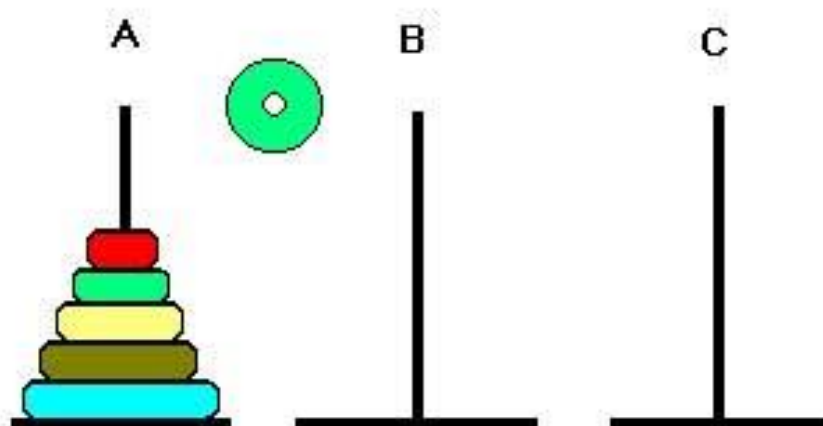
- Problema da TORRE DE HANOI



- Proposição: inicialmente tem-se três hastes: A, B e C, e na haste A repousam três anéis de diâmetros diferentes, em ordem decrescente de diâmetro.

VAMOS “BRINCAR”...

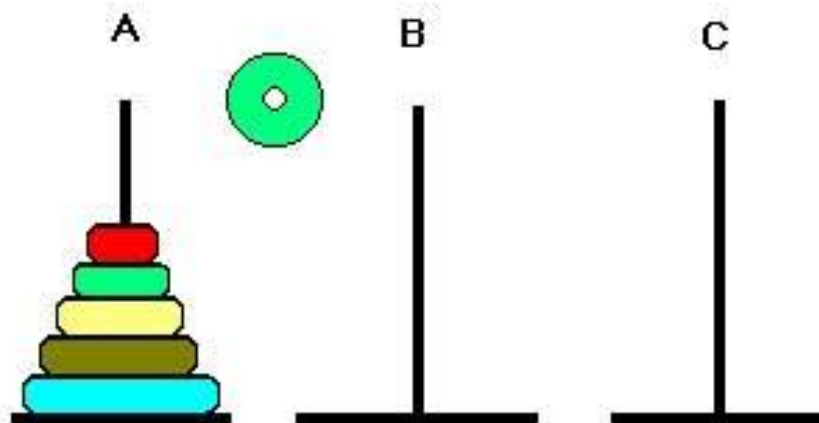
- O objetivo é transferir os três anéis da haste A para B, usando C se necessário. As regras do movimento são:
 - Deve-se mover um único anel por vez;
 - Um anel de diâmetro maior nunca pode repousar sobre algum outro de diâmetro menor.



VAMOS “BRINCAR”...

- Solução 1:

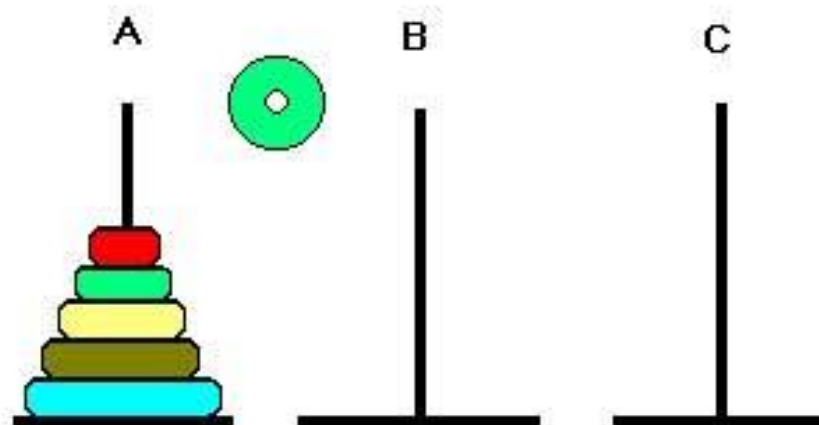
- 1: A -> B
- 2: A -> C
- 3: B -> C
- 4: A -> B
- 5: C -> A
- 6: C -> B
- 7: A -> B



VAMOS “BRINCAR”...

- Solução 2:

- 1: A -> C
- 2: A -> B
- 3: C -> B
- 4: A -> C
- 5: B -> C
- 6: B -> A
- 7: C -> A
- 8: C -> B
- 9: A -> C
- 10: A -> B
- 11: C -> B



PORTANTO...

Podem existir várias soluções para o mesmo problema!

- Devemos nos esforçar (raciocinar) por buscar a “melhor” solução!
- No caso da TORRE DE HANOI, a solução que utilize o MENOR NÚMERO DE MOVIMENTOS!

PORTANTO...

$$\text{Número de movimentos} = 2^n - 1$$

Onde n é o número de discos.

Outra forma de pensar o problema é não estabelecer qual deve ser a haste destino e enxergar as hastes dispostas em círculo.

OLHAR DIFERENTE...



Se n for ímpar, os anéis serão transferidos para a primeira haste após a haste de início, no sentido horário;

Se n for par, os anéis serão transferidos para a primeira haste após a haste de início, no sentido anti-horário;

Algoritmo GERAL para as Torres de Hanoi

Início

Repita

1. Mova o menor anel de sua haste atual para a próxima, no sentido horário.
2. Execute o único movimento possível com um anel que não seja o menor de todos.

Até que todos os discos tenham sido transferidos para outra haste.

Fim

MODELAGEM DE PROBLEMAS

Considere o seguinte problema:

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?

Como poderíamos raciocinar:

Se eu tinha R\$ 100,00 e recebi como troco R\$ 67,00, o custo do total de canetas é a diferença entre os R\$ 100,00 que eu tinha e os R\$ 67,00 do troco. Ora, isto vale R\$ 33,00; portanto, esse valor foi o total pago pelas canetas. Para saber quanto custou cada caneta, basta dividir os R\$ 33,00 por 30, resultando em R\$ 1,10, ou seja, o preço de cada caneta.

Matematicamente (x o custo de cada caneta, então quantogastei=30x. Como quantogastei+troco = R\$ 100,00, tem-se):

$$30x + 67 = 100$$

$$30x = 100 - 67$$

$$30x = 33$$

$$X = 33/30$$

$$X = 1,1$$

MODELAGEM DE PROBLEMAS

Problema GERAL:

Compraram-se **N** canetas iguais, que foram pagas com uma nota de **Z** reais, obtendo-se **Y** reais como troco. Quanto custou cada caneta?

É NECESSÁRIO PENSAR SOBRE AS RESTRIÇÕES DO PROBLEMA:

- O que acontecerá se alguém tentar comprar 0 canetas?
- Ou -3 canetas? Faz sentido?
- Suponha que $N = 10$; $Z = 10$; $Y = 15$. Cada caneta será R\$ - 0,50.

OU SEJA, temos as restrições (que devem estar codificadas)

- O valor pago pelas canetas seja sempre maior que o troco recebido;
- Que o valor pago e a quantidade de canetas seja sempre maior que zero;
- Que o troco seja maior ou igual a zero.

LOGO, as restrições são: $Z > Y$, $N > 0$, $Z > 0$ e $Y \geq 0$

MODELAGEM DE PROBLEMAS

Algoritmo GERAL e CORRETO para o problema das canetas:

Início

Leia(N, Y, Z).

Se ($Z > Y$ **E** $N > 0$ **E** $Y \geq 0$ **E** $Z > 0$) **Então**

 TotalCanetas $\leftarrow (Z - Y) / N$.

Exiba(TotalCanetas).

Senão

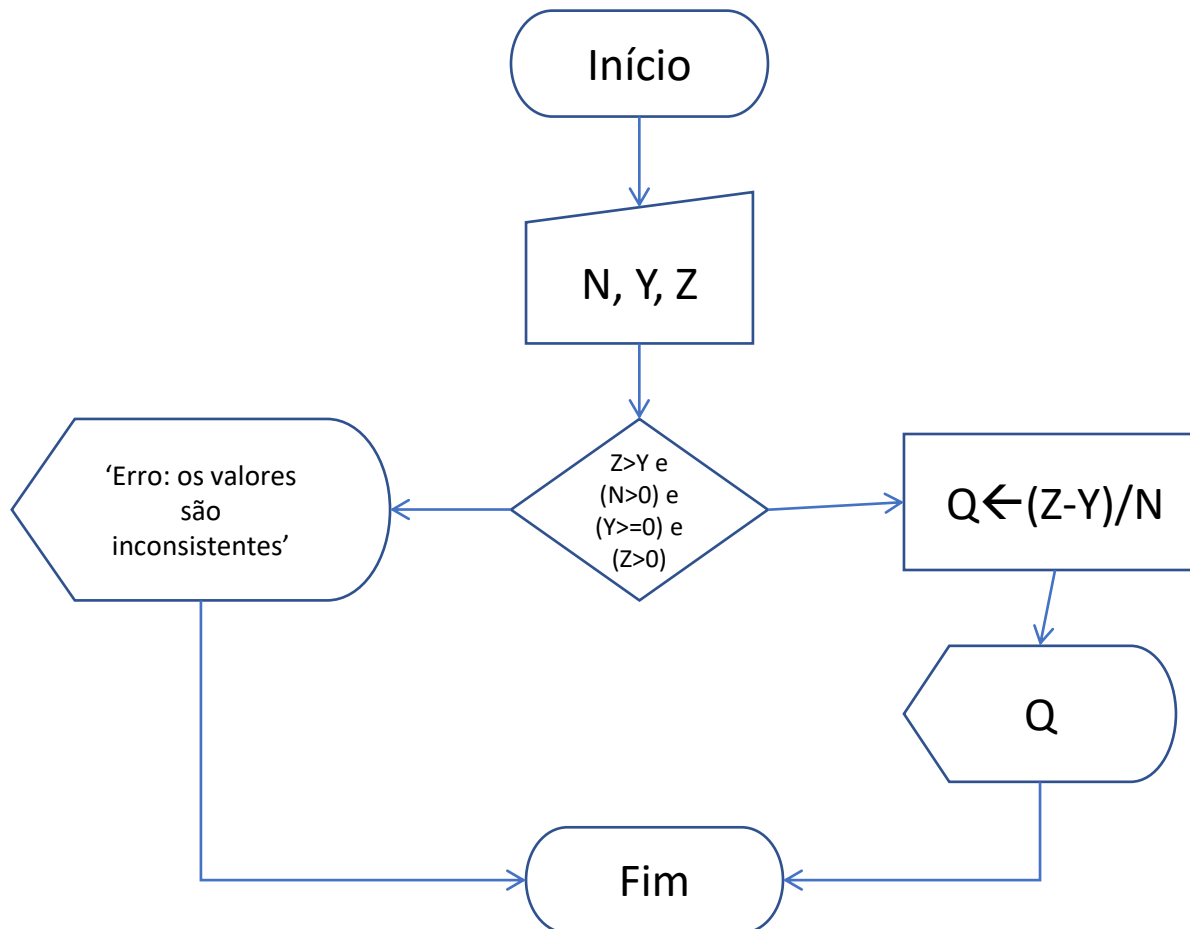
Exiba("Erro: os valores são inconsistentes!").

Fim Se

Fim

MODELAGEM DE PROBLEMAS

Fluxograma para resolver o problema das canetas:



MODELAGEM DE PROBLEMAS

Portanto, não **DECORAMOS** soluções, mas buscamos o **ENTENDIMENTO** de como foi obtida uma solução e usamos esta **EXPERIÊNCIA** adaptando-as a outras situações, por **ANALOGIA**, **GENERALIZAÇÃO** ou **ESPECIALIZAÇÃO**.

DICAS:

1. Ao se deparar com um problema novo, tente entendê-lo. Para auxiliar, pense no seguinte:

- o que se deve descobrir ou calcular? Qual é o 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.

MODELAGEM DE PROBLEMAS

2. Crie um plano com a solução:

- Consulte sua memória e verifique se você já resolveu algum problema similar;
- 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 estas partes;
- É possível enxergar o problema de outra forma, de modo que seu entendimento se torne mais simples?

MODELAGEM DE PROBLEMAS

3. Formalize a solução:

- Crie um algoritmo informal com passos que resolvam o problema;
- Verifique se cada passo desse algoritmo está correto;
- Escreva um algoritmo formalizado por meio de um fluxograma ou outra técnica de representação.

MODELAGEM DE PROBLEMAS

4. Exame dos resultados:

- Teste o algoritmo com diversos dados de entrada e verifique os resultados (teste de mesa);
- Se o algoritmo não gerou resultado algum, o problema está na sua sintaxe e nos comandos utilizados. Volte e tente encontrar o erro;
- Se o algoritmo gerou resultados, estes estão corretos? Analise sua consistência.
- Se não estão corretos, alguma condição, operação ou ordem das operações está incorreta. Volte e tente encontrar o erro.

MODELAGEM DE PROBLEMAS

5. Otimização da solução:

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

Encaremos os problemas de COMPUTAÇÃO como verdadeiros projetos de Engenharia. Os programas de computador são ferramentas empregadas para auxiliar às pessoas.