

MÉTODOS COMPUTACIONAIS

DR. MARCOS NAPOLEÃO RABELO

DR. WANDERLEI M. PEREIRA JUNIOR

Introdução à criação de algoritmos

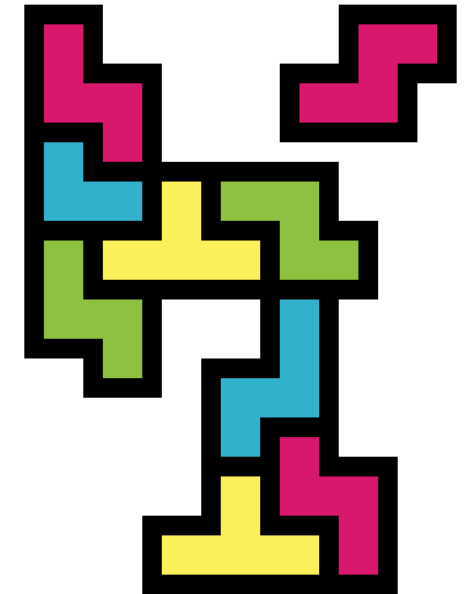
Grupo de Pesquisa e Estudos em Engenharia (GPÉE)



REPRESENTAÇÃO DE ALGORITMOS

O **lógica** pode ser explicada como **uma sequência coerente de ideias**, ou um modo pelo qual acontecimentos se encadeiam naturalmente [1]. Então o que seria a lógica de programação?

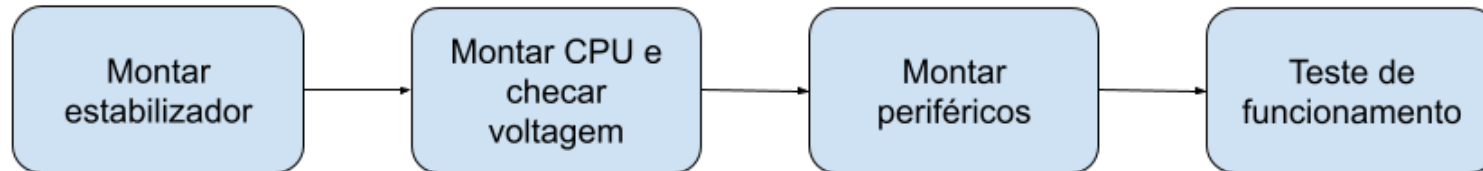
Segundo Forbellone e Eberspächer [2] a lógica de programação **poderia ser descrita como um uso correto, da ordem da razão** e de processos de raciocínio e simbolizações formais na programação de computadores. Logicamente essa definição pode ser traduzida como a **busca por uma solução adequada e válida para determinados problemas** que envolvam a programação de computadores.



Normalmente a utilização de computadores sempre envolve de forma direta ou indireta o uso dos chamados algoritmos. Estes podem ser definidos como um **procedimento computacional** bem definido que **toma** algum valor ou conjunto de valores como **entrada** e **produz** algum valor ou conjunto de valores como **saída** [3].

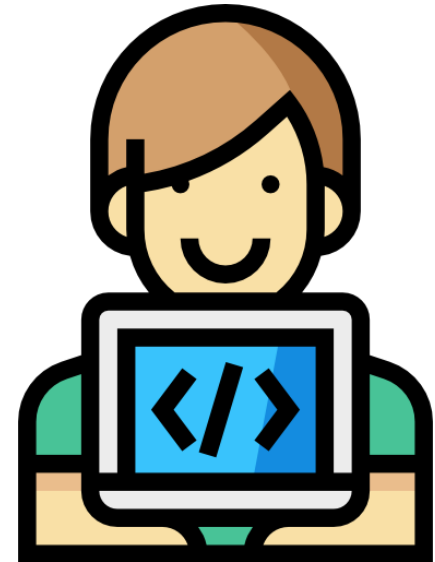
Imaginemos que desejamos montar um computador *desktop* [2], Qual seria uma forma lógica de fazer esta montagem do equipamento ?

Figura 1 – Fluxograma de um algoritmo de montagem de computador.



Os algoritmos podem ser escritos em **diversos paradigmas**. Nesse curso utilizaremos na maioria dos casos o **paradigma procedural** (também chamado de **paradigma imperativo**) por ser de uso geral e amplamente empregado para aqueles que desejam iniciar na área de métodos numéricos.

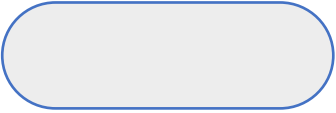
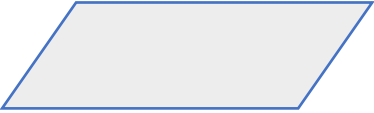

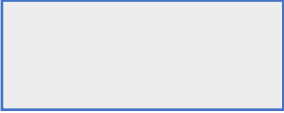
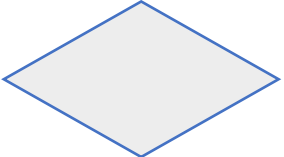
Basicamente neste modelo de programação nosso algoritmo conterá uma **lista de instruções que são executadas passo a passo na ordem previamente descrita**. Logo para um algoritmo atingir o seu funcionamento pleno deve ser baseado em uma lógica, ou um conjunto de passos que tornem aquela sequência adequada ao problema estudado.



A **representação dos algoritmos** pode ser feita de diversas formas, são exemplos: (a) **Fluxograma**; (b) **Pseudocódigo**; e (c) Diagrama de Chapin. Neste texto as formas mais utilizadas serão o fluxograma e o pseudocódigo.

A representação da **Figura 1** é o modelo de fluxograma e os “balões” representam uma finalidade dentro do algoritmo. O **Quadro 1** apresenta algumas das possibilidades.

Quadro 1 – Elementos mais utilizados no fluxograma [4].

				
Terminal: Demarca pontos de início e fim	Entrada de dados	Linhas que indicam fluxo do algoritmo	Passo de operação	Condição: Mudança de fluxo

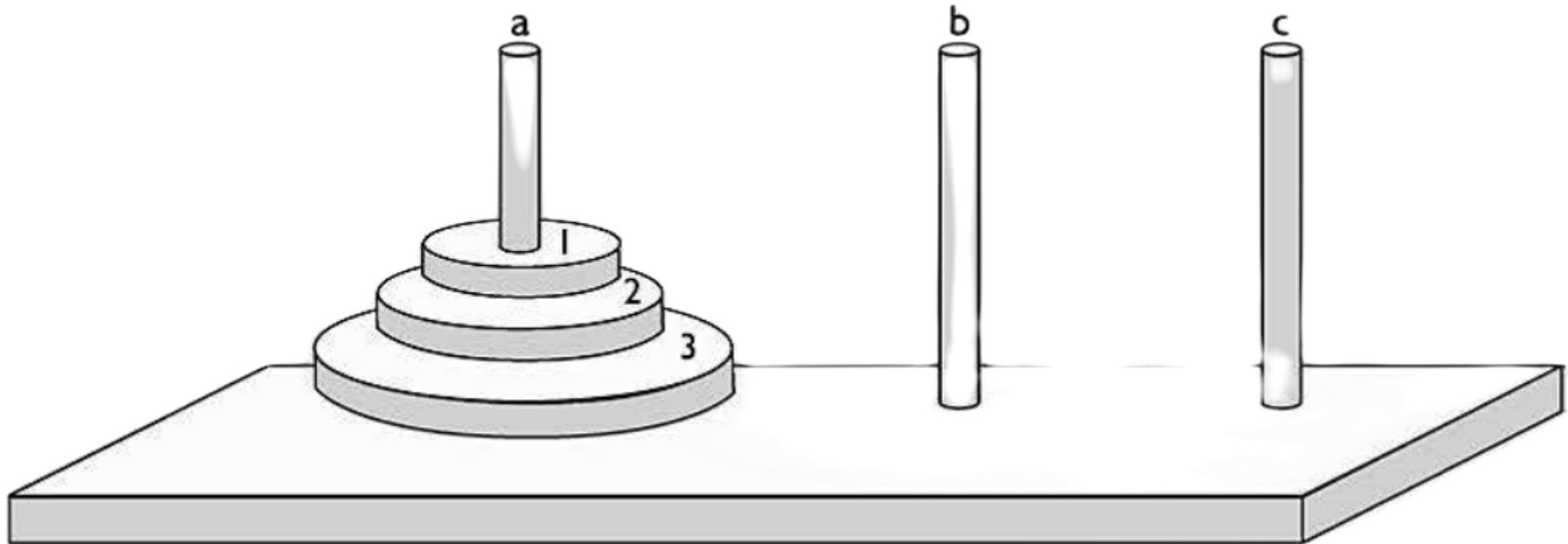
A outra forma seria a utilização dos chamados pseudocódigos que podem ser visualizados na *Figura 2*.

Figura 2 – Pseudocódigo do algoritmo de montagem de computador.

- 1 Montar o estabilizador
- 2 Montar CPU e checar voltagem do equipamento
- 3 Montagem dos periféricos
- 4 Teste de funcionamento

Exercício 1.1 [2]: Elabore um algoritmo que mova três discos de uma Torre de Hanói, que consiste em três hastes (a - b - c), uma das quais serve de suporte para três discos de tamanhos diferentes(1 - 2 - 3), os menores sobre os maiores. Pode-se mover um disco de cada vez para qualquer haste, contanto que nunca seja colocado um disco maior sobre um menor. O objetivo é transferir os três discos para outra haste.

Figura 3 – Torre de Hanói.



DADOS E OS TIPOS DE VARIÁVEIS

Trabalhar com algoritmos está intuitivamente ligado a manipulação de dados. Segundo Lopes e Garcia [5] uma **variável é um local na memória principal**, isto é, um **endereço que armazena um conteúdo**. Em linguagens de alto nível é permitido dar nome a esse endereço de forma a facilitar o processo de programação.

Por exemplo em **Python 3** se o usuário digitar o comando **id()** o algoritmo retornará o endereçamento de memória da variável *H*.

```
>>> H = 50
>>> ENDERECO_RAM = id(H)
>>> print("Endereço de memória = ", ENDERECO_RAM)
Endereço de memória = 94820313464864
```


De forma abstrata podemos entender que o **espaço de memória** (ou endereços) são reservados em **listas com endereços específicos** que **podem ou não estar preenchidos por variáveis** conforme **Figura 4**.

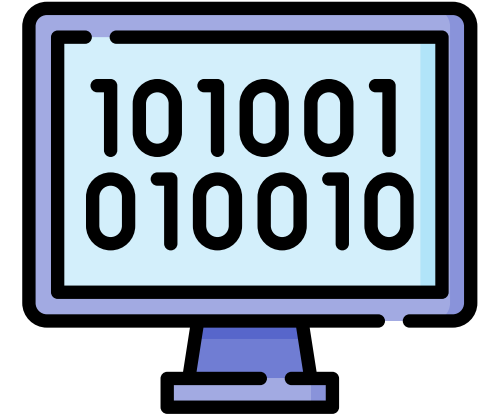
Figura 4 – Armazenamento abstrato de variáveis na memória RAM do computador.



END.: 394394803403	H = 10
END.: 394345803403	H = 25
END.: 394394803288	"vazio"
END.: 285594803403	"vazio"
END.: 285594801444	"vazio"
END.: 199594801444	"vazio"
END.: 199594801888	"vazio"

Espaço de memória

Salientamos aqui que o computador trabalha com a **codificação binária** em seu **cérebro**, logo todas informações são transformadas em um **sistema numérico binário**, sendo que os dígitos 0 ou 1 deverão ocupar uma porção do espaço da memória chamada **bit** ($8 \text{ bits} = 1 \text{ byte}$). Cada **byte** será especificado para um **determinado endereço de memória** [6].



Os **tipos de variáveis** podem ser divididos da seguinte forma:

- **Inteiros** (em inglês *integer*);
- **Reais** (em inglês *float*);
- **Caracteres** (em inglês *string*);
- **Lógica** (em inglês *logical*).

REFERÊNCIAS

- [1] Michaelis. Dicionário escolar língua portuguesa. 2018.
- [2] Forbellone ALV, Eberspächer HF. Lógica de programação: a construção de algoritmos e estruturas de dados. São Paulo: Pearson Prentice Hall; 2007.
- [3] Cormen TH. Algoritmos: teoria e prática. Rio de Janeiro: Campus; 2012.
- [4] Junior DP, Nakamiti GS, Engelbrecht A de M, Bianchi F. Algoritmos e Programação de Computadores. 2012.
- [5] Lopes A, Garcia G. Introdução à programação: 500 algoritmos resolvidos. Rio de Janeiro (RJ): Campus; 2002.
- [6] Ascencio AFG, Campos EAV de. Fundamentos da Programação de Computadores: Algoritmos, Pascal, C, C++ e Java. 3^a edição. Pearson Universidades; 2012.