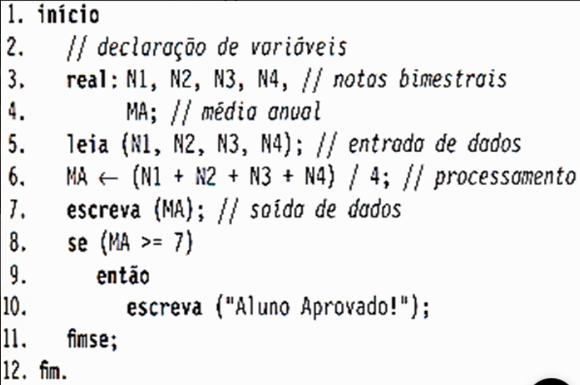
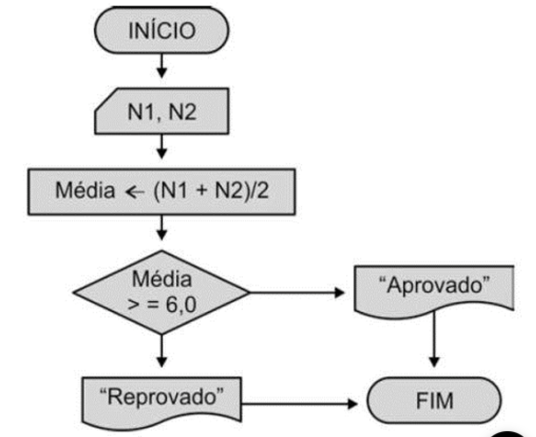
**Algoritmos**

🡪 Representados de três maneiras básicas: narração descritiva; fluxogramas e pseudocódigo.

Podemos representar algoritmos pela linguagem comum desde que os passos sejam claros e bem definidos.



Outra forma de representação de um algoritmo é por meio de um fluxograma, espécie de diagrama em que a lógica de um algoritmo é representada por meio de um fluxo ligando elementos gráficos.



Durante muito tempo, a representação de algoritmos por meio de fluxogramas foi utilizada. Entretanto, novas ferramentas e pesquisas mostraram que fluxogramas ocultam representações que podem ser importantes (a estrutura de dados, por exemplo) e a ferramenta foi caindo em certo descrédito.

**Estruturas de controle**

O fluxo de execução de um algoritmo está sujeito a desvios dependendo do atendimento ou não de certas condições. Chamamos os elementos responsáveis por essas mudanças e desvios de estruturas de controle.

* A condição “se-(então)(-senão)” é uma estrutura de seleção comum em muitas linguagens de programação. Utiliza expressões booleanas (do tipo lógica verdadeiro/falso) para desviar a execução do código para um fluxo ou outro, dependendo do valor avaliado ser verdadeiro ou falso.
* A estrutura ENQUANTO repete uma sequência de comandos enquanto uma determinada condição ou expressão lógica, for satisfeita. A expressão é avaliada antes de cada repetição do laço interno. Quando seu resultado for VERDADEIRO, a <sequência-de-comandos> é executada. Quando a condição não for mais satisfeita, o fluxo desvia do laço de execução.
* A estrutura de repetição REPITA…ATÉ é utilizada quando um conjunto de comandos deve ser executado repetidamente até que uma condição (expressão lógica) seja verdadeira. Note a diferença para a estrutura ENQUANTO, aqui a condição é testada ao final do laço!

É interessante observar que estes algoritmos embora ligeiramente diferentes, geram o mesmo resultado. Pode-se usar outras estruturas de controle quando se sabe, a priori, o número de repetições como:

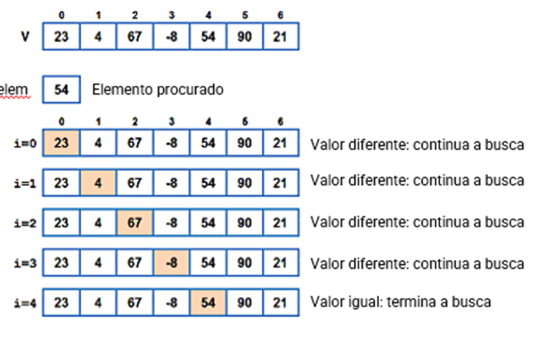
PARA <variável contadora> DE <valor inicial> ATE <valor final> [PASSO <valor de incremento>] FAÇA

<Instruções a serem executadas repetidamente até a <variável contadora> atingir o valor final>

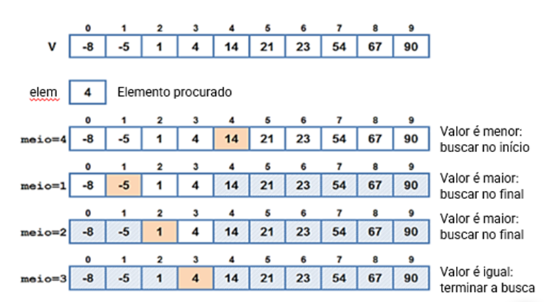
FIM-PARA

**Análise e comparação entre algoritmos: BUSCA de um elemento em uma lista**

Busca linear: o método da busca linear (ou sequencial) faz com que o elemento em cada posição da lista seja comparado ao elemento que se busca.



Busca binária: tudo começa com a lista ordenada, só é possível busca binária em **listas ordenadas**. O valor a ser buscado é comparado com o valor do meio da lista. Se o valor a ser procurado é maior do que o valor do meio da lista, descartamos a metade inferior da lista, afinal ele não estará lá com certeza.



Ao comparar os dois métodos, busca linear e busca sequencial, é importante observar:

* Se a lista não está ordenada, a busca binária não se aplica.
* A busca sequencial é ineficiente para conjuntos grandes de valores de entrada.
* Se o valor a ser buscado estiver logo no começo da lista, a busca sequencial é rápida, mas isso é raro acontecer, casos ótimos ou exceções não costumam ser muito considerados em análises de algoritmos.
* Olhando o comportamento matemático dos processos computacionais (complexidade assintótica) podemos enxergar os algoritmos por meio de gráficos de funções. No eixo X a quantidade n de elementos, no eixo Y o número de comparações.
* A busca sequencial tem complexidade n e a busca binária tem complexidade log n.
* Como a função log n cresce significativamente mais devagar do que a busca linear em função do tamanho da entrada, dizemos que faz menos comparações, sendo mais eficiente do que a busca sequencial.