# Teoria

Podem ser ordenadas de vários modos sem ser ordem de chegada.

Vantagem de usar vetor com n° de elementos definidos

-> Já tme lugar definido para o elemento

Alocação dinâmica precisa de um tempo de alocação na memória

-> atributo 'prox'

Lista encadeada simples

-> Final para null

Lista encadeada circular

-> Final para o primeiro da lista

-> Vantagem: posso voltar para o inicio

-> Bom para pesquisa

Lista duplamente encadeada

-> Posso retornar entre os elementos na lista

-> Tenho um atributo 'ante' que é o elemento anterior

# PRÁTICA

1 caso: lista vazia

-> Referencia lista -> null

lista recebe 5

-> novo = aloca NO

-> novo.dado = 5

-> lista = novo

-> novo.prox = null

2 caso: lista possui nós e o nó a ser inserido torna-se o primeiro

-> Lista crescente por conteúdo

-> Quero inserir 2 na lista, ou seja, ele vira o primeiro

-> se (novo.dado < lista.dado)

-> novo aponta para o novo NO com dado 2

-> o atributo prox do novo recebe a referência lista

-> lista.dado

-> lista precisa se referenciar ao novo dado colocado

-> lista = novo

3 caso: o novo nó deve ser inserido no ‘meio’ da lista

-> aux precisa percorrer a lista para achar a referencia de onde vai ficar o novo NO (enquanto)

-> precisamos ver o dado do nó sucessor ao aux (aux.prox.dado)

-> para não perdemos o resto da lista

PRIMEIRO: Atributo prox do novo nó vai receber a referencia do prox do no ‘aux’

-> DEPOIS: o prox do aux recebe o novo NO.

4 caso: inserir o valor 10

-> procurar o local para inserir

...

-> novo.prox recebe null (final da fila)

-> aux.prox recebe o novo NO