=======================================================

***Listas Encadeadas***

=======================================================

# **Características importantes:**

* Podem crescer e diminuir dinamicamente;
* Tamanho máximo não precisa ser definido previamente;
* Provêm flexibilidade, permitindo que os itens sejam reorganizados eficientemente;
* Perda no tempo de acesso a qualquer item arbitrário da lista, comparando com vetores;
* Também chamadas de **Listas Ligadas**;
* A sequência de elementos é especificada explicitamente, onde cada um contém um (link) para o próximo da lista 🡪 Encadeamento;
* Cada elemento é chamado de nó da lista;
* A lista deve possuir um ponteiro para o primeiro elemento (ou nó) e um para o último. Mas, pode haver implementações sem o ponteiro para o último elemento;
* Do primeiro elemento, pode-se alcançar o segundo seguindo o encadeamento e assim sucessivamente.
* Para cada novo elemento inserido na estrutura, um espaço na memória é alocado dinamicamente, mas a alocação do espaço não é contígua.

# **Detalhe que deve ser considerado:**

Cada elemento possui pelo menos dois campos: um para armazenar a informação e outro para o endereço do próximo (ponteiro/referência).

Simplesmente encadeada 🡪 Utiliza “ponteiros” em uma única direção.

**Início**

**Fim**

# Inserção

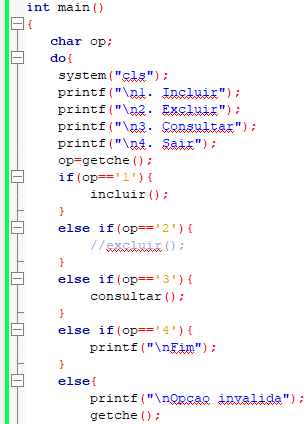
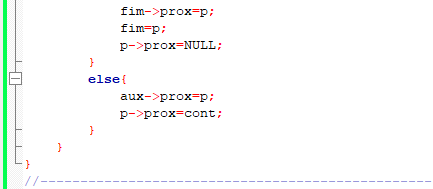
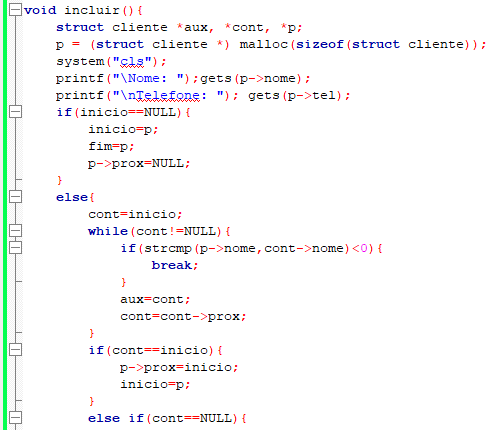
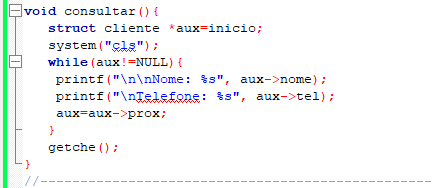
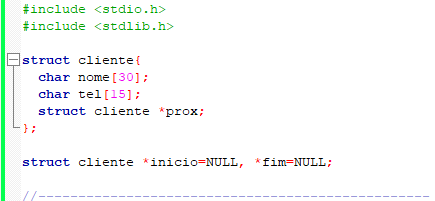
Em listas ordenadas, podemos ter 4 situações:

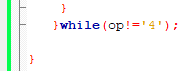
1. A lista está vazia;
2. O novo elemento se tornará o primeiro da lista;
3. O novo elemento se tornará o último da lista;
4. O novo elemento será inserido entre dois elementos da lista.

=======================================================

***Inclusão Ordenada em Lista Encadeada***

=======================================================





=======================================================

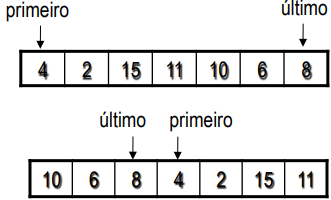
***Pilhas e Filas***

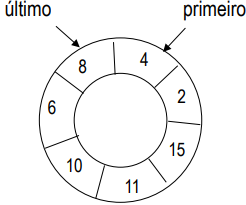
=======================================================

Tanto na estrutura Pilha como na Fila, podemos construir de forma **Estática** ou **Dinâmica** na Memória.

# **Filas**

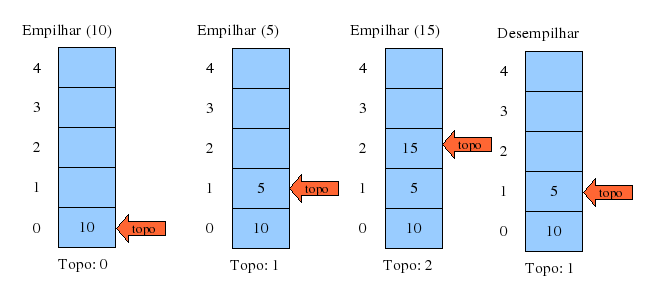
Uma Fila ou queue em inglês é uma estrutura de dados que usa o método **FIFO**(acrônimo para **First In**, **First Out**, que em português significa primeiro a entrar, primeiro a sair).

A ideia fundamental da fila é criar uma disciplina de acesso, onde só podemos inserir um novo elemento no final da fila e só podemos retirar o elemento do início.



# **Pilhas**

Pilha ou stack é uma lista linear em que todas as inserções e remoções de elementos só podem ser feitos em uma extremidade chamada Topo. As pilhas também são chamadas de estruturas **LIFO** (**Last In First Out**) ou seja, o último elemento inserido é o primeiro a ser removido.



# **Pilha Dinâmica**

A estrutura de Pilha é análoga ao conceito que temos de Pilhas em geral, a ideia fundamental da pilha é que todo o acesso a seus elementos é feito através do seu topo.

Assim, quando um elemento novo é introduzido na pilha, passa a ser o elemento do topo, e o único elemento que pode ser removido da pilha é o do topo.

1. Inicialmente criamos a variável Topo contendo null.
2. Para inserir o primeiro elemento, criaremos um Objeto que conterá a informação e uma referência na memória para o objeto anterior da Pilha.
3. No nosso caso criaremos um objeto do tipo No.
4. Criar um Objeto do tipo No e colocar sua referência em P.
5. Agora a variável “anterior” do Objeto recebe o conteúdo de Topo.
6. Como a pilha estava vazia, a variável recebe null (A variável anterior está contida no objeto e armazenará o endereço do elemento anterior da Pilha. Neste caso o anterior é null.)
7. Na sequência a variável Topo recebe o conteúdo de P e passa a referenciar o Topo da Pilha.
8. Quando a rotina de inclusão for concluída a variável P é destruída e nossa Pilha fica desta forma.
9. As próximas inclusões seguirão a mesma ordem.
10. Ou seja, a variável anterior do novo No recebe Topo e Topo recebe a referência contida em P.
11. Após a inclusão a variável P é destruída.
12. Mais um Nó colocado na Pilha.

# **Remoção em Pilhas**

Conforme exposto anteriormente as retiradas só podem ocorrer no Topo da Pilha.

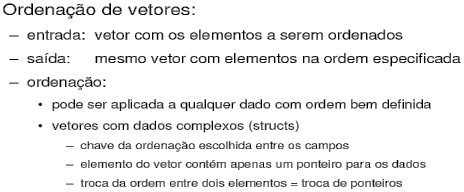
1. O No que será retirado. E será aquele que o Topo está referenciando.
2. Após, o Topo recebe o conteúdo da Variável anterior...
3. O No que será retirado é aquele que o Topo está referenciando.

=======================================================

***Métodos de Ordenação***

=======================================================

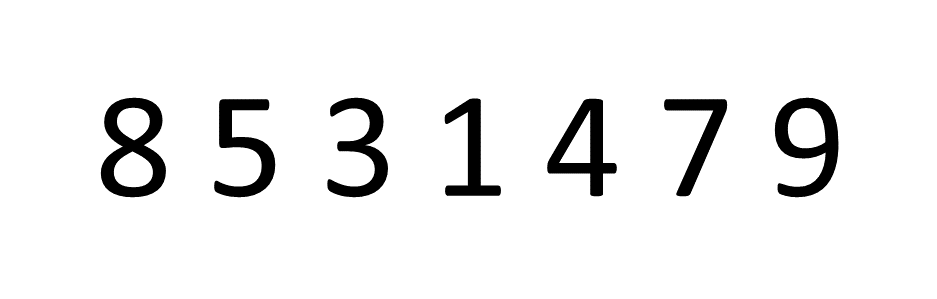
Ordenação é o processo de arranjar um conjunto de informações semelhantes numa ordem crescente ou decrescente.



Ao Analisar qualquer método de ordenação, você deve determinar quantas comparações e trocas são realizadas para o menor, médio e pior caso.

# **Ordenação Bolha (Bubble Sort)**

A ordenação mais conhecida e mais difamada é a Ordenação Bolha

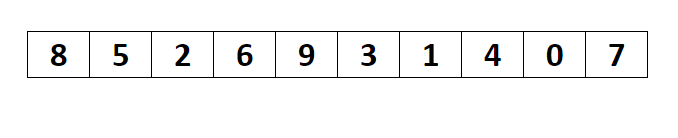


# **Ordenação por Seleção (Selection Sort)**

A ordenação por seleção seleciona o elemento de menor valor e troca-o pelo primeiro elemento.

Então, para os n-1 elementos restantes, é encontrado o elemento de menor chave, trocado pelo segundo elemento e assim por diante.

As trocas continuam até os dois últimos elementos.



# **Ordenação por Inserção (Insertion Sort)**

A ordenação por inserção é o terceiro e último dos algoritmos simples de ordenação. Inicialmente, ela ordena os dois primeiros membros da matriz.

Em seguida, o algoritmo insere o terceiro membro na sua posição ordenada com relação aos dois primeiros membros.

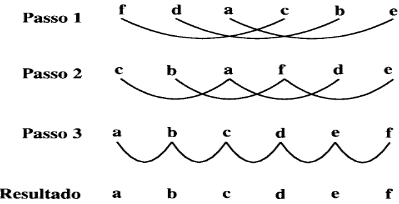
Então, insere o quarto elemento na lista dos três elementos, o processo continua até que todos os elementos tenham sido ordenados.



# **Ordenação Shell (Shell Sort)**

O método geral é derivado da ordenação por inserção e é baseado na diminuição dos incrementos. Considere o desenho abaixo. Primeiro, todos os elementos que estão três posições afastados são ordenados.

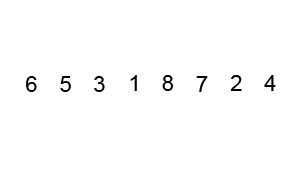
Em seguida, todos os elementos de estão a duas posições são ordenados. Finalmente, tosos os elementos adjacentes são ordenados.



Ou seja ele divide os elementos e troca as posições de seus opostos, o processo se repete até todos os elementos estarem na posição correta.

# **Ordenação QuickSort**

A quicksort é baseada na ideia de partições. O procedimento geral é selecionar um valor, chamado de comparando, e, então fazer a partição da matriz em duas seções, com todos os elementos maiores ou iguais ao valor de um lado e os menores de outro. Esse processo é repetido para cada seção restante até que a matriz esteja ordenada.



Ou seja ele seciona um elemento central e passa todos os elementos menores para um lado e todos os elementos maiores para o outro lado, o processo se repete até todos os valores estarem na sua devida posição.

=======================================================

**Questões**

=======================================================

