Estrutura de Dados
Lista Encadeada

Profa. Ana Cristina dos Santos
email: ana.csantos@sp.senac.br

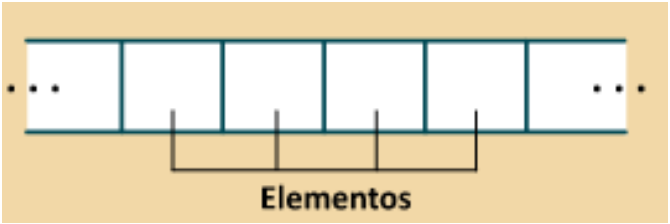
Tópicos da Aula

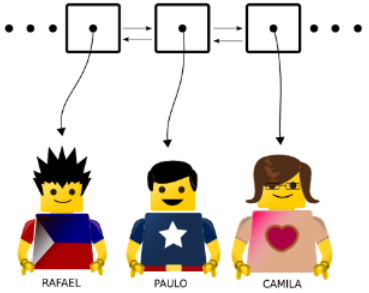
- Alocação de memória sequencial e encadeada
- Tipo Abstrato de Dados – Lista Ligada
- Operações sobre lista ligada

Alocação Encadeada X Alocação Sequencial

Classificação por critério de tipo de armazenamento.

- **Listas sequenciais:** Os elementos são vizinhos lógicos e físicos (contíguos)

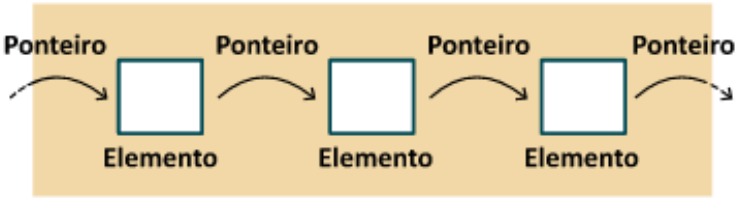


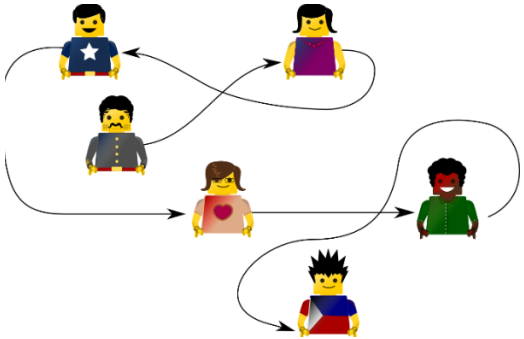


Alocação Encadeada X Alocação Sequencial

Classificação por critério de tipo de armazenamento.

- **Listas encadeadas:** Os elementos que são vizinhos lógicos, podem não ser vizinhos físicos.





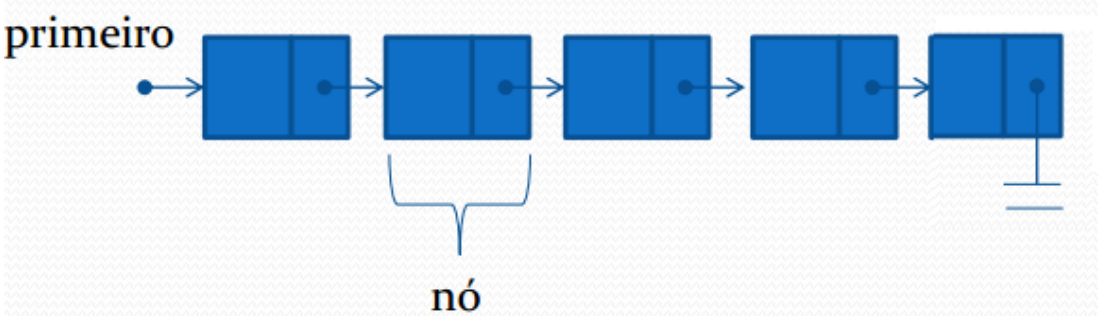
Ponteiro: Variável que diz qual o endereço do próximo Elemento da lista.

Alocação Encadeada X Alocação Sequencial

- A alocação encadeada se caracteriza pela alocação dos elementos na medida que são necessários (ou dispensados), sendo isso um recurso muito vantajoso em relação a alocação sequencial.
- Permite que as operações de inserção e remoção sejam mais eficientes.
- As desvantagens nesse tipo de alocação é o gasto de memória maior, em razão da forma como é implementado.

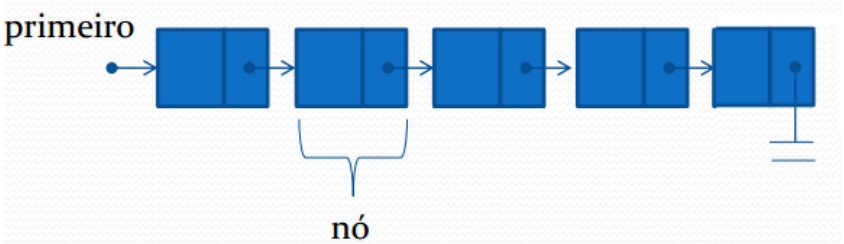
Tipo de Dados Abstrato: Lista Ligada

- Uma lista encadeada (= *linked list* = lista ligada) é uma representação de uma sequência de objetos na memória do computador.

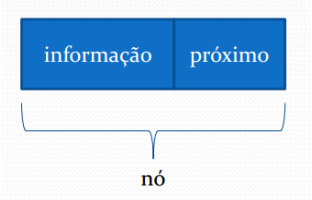


Tipo de Dados Abstrato: Lista Ligada

- Cada elemento da sequência é armazenado em uma célula (=Nós) da lista: o primeiro elemento na primeira célula, o segundo na segunda e assim por diante.



- Cada Nó contém um elemento de algum tipo e o endereço da célula seguinte.



Tipo de Dados Abstrato: Lista Ligada

- Vamos supor que os objetos armazenados nas células são do tipo int, assim cada Nó da lista um contém um elemento da sequência e a referência (=endereço) do Nó seguinte (prox).


```
public class No {
   private int elemento;
   private No prox;

   public No(int elemento, No prox ){
     this.elemento = elemento;
     this.prox = prox;
```

Tipo de Dados Abstrato: Lista Ligada

- Para se ter uma lista ligada basta ter a referência do primeiro Nó da lista, ou seja, o endereço de sua primeira célula.

```
public class listaLigada {
    private No inicio;   // endereço inicial da lista

  public listaLigada(){...}
  public void addInicio(int elemento ){...}
  public int remInicio(){...}
  public boolean isEmpty(){...}
  }
```

Tipo de Dados Abstrato: Lista Ligada

- Agora podemos definir as seguintes operações sobre a lista ligada:
  - addInicio: insere um elemento no início da lista
  - remInicio: retira o primeiro elemento da lista
  - isEmpty: verifica se a lista está vazia, ou seja, se `ini==null`.

Inserção no Início da Lista

- addInicio:
- Para inserir no começo da Lista basta criarmos um novo Nó, e este novo nó apontará para o atual primeiro Nó da lista.
- Depois atualizamos o atributo ini para referenciar ao novo Nó criado.

Remoção no Início da Lista

- remInicio:
- Antes de remover devemos verificar se temos pelo menos um Nó na lista, não faz sentido remover algo que não existe.
- Caso a lista esteja vazia seria interessante gerar uma exceção conveniente.
- Depois, basta "avançar" a referência que aponta para o primeira Nó, em seguida retornar o elemento Nó.

Percorrendo a lista ligada

- Para executar alguns testes, precisamos imprimir o conteúdo da nossa Lista.
- Para isso vamos sobrescrever o método toString() da classe para que ele concatene todos os elementos de dentro da Lista Ligada em uma única String.

Percorrendo a lista ligada

```
public String toString(){
        String strLista ="";
        No temp=ini;
        while(temp!=null){
            strLista += temp.getElemento()+”,”;
            temp=temp.getProx();
        }
         return strLista;
}
```

Exercícios

1) Qual seria a **complexidade de tempo** para as operações e **inserção e remoção no início** para lista ligada?
2) Modifique a implementação da lista ligada para que se torne uma **implementação genérica**.
3) Implemente um método que faz a **inserção no final** da lista ligada. Faça uma versão recursiva e uma iterativa para esse exercício.
4) Implemente um método que faz a **remoção no final** da lista ligada. Faça uma versão recursiva e uma iterativa para esse exercício.

Exercícios

5) Implemente um método que faz a **busca linear** um lista ligada. Caso o elemento esteja na lista o método retorna true, caso contrário false. Faça uma versão recursiva e uma iterativa para esse exercício.

Exercícios

6) Implemente um método que faz a **inserção de um elemento em uma lista ordenada**, seu método deve funcionar para as seguintes situações:

- lista vazia,
- lista com somente um elemento,
- lista com vários elementos.

Faça uma versão recursiva e uma iterativa

Exercícios

7) Implemente um método que faz a **remoção de um elemento em uma lista ordenada**, seu método deve funcionar para as seguintes situações:

- lista vazia,
- lista com somente um elemento,
- lista com vários elementos.

Faça uma versão recursiva e uma iterativa

Exercícios

8) Considere uma **lista elementos ordenados**, comentar as vantagens e desvantagens, para as operações de busca, remoção e inserção, considerando uma implementação de armazenamento sequencial e armazenamento encadeado.

Exercícios

9) Implemente um método que **inverta os Nós** de uma lista ligada, alterando as referências (prox) de cada Nó da lista, sem mover suas informações e nem alocando novos nós.

Exercícios

10) Intercalação de duas listas ordenadas com menor complexidade
11) União de duas listas quaisquer
12) Intersecção de duas listas quaisquer