Estrutura de Dados – 2º semestre de 2019

Professor Mestre Fabio Pereira da Silva

Lista Duplamente Encadeada

- Cada elemento armazena um ou vários dados (estrutura homogênea ou heterogênea) e dois ponteiros; o primeiro para o próximo elemento, e o segundo para o anterior.
- Esses ponteiros permitem o duplo encadeamento e mantêm a estrutura linear.

Operações

- Inicializar a lista
- Inserir um elemento no início
- Inserir um elemento no final
- Inserir um elemento em uma posição específica da lista
- Realizar a pesquisa de um elemento
- Remover um elemento do início da lista
- Remover um elemento do final da lista
- Remover um elemento de uma dada posição da lista
- Ordenar a lista
- Realizar a composição de outras estruturas de dados a partir de uma lista duplamente encadeada
- Substituição de um valor por outro

Vantagens

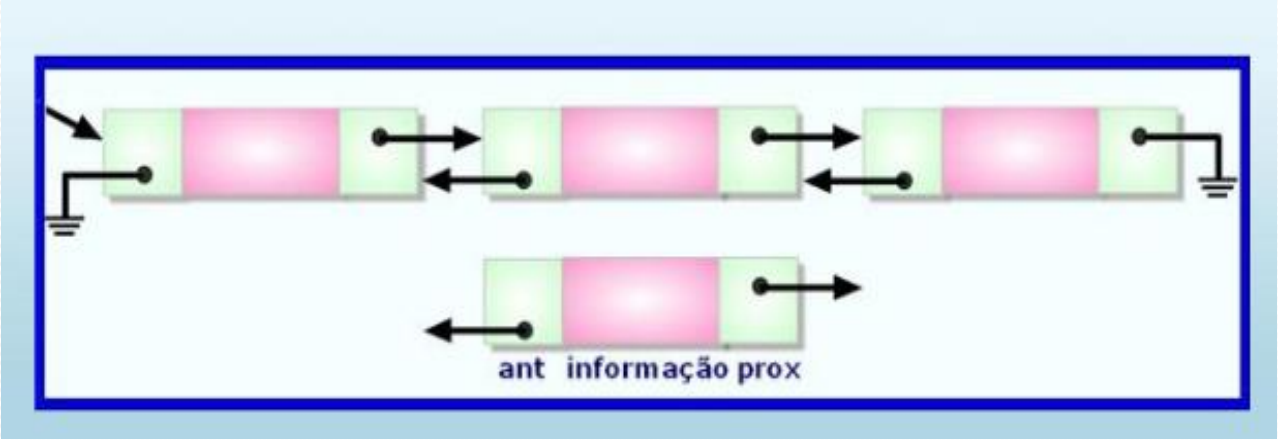
- Facilita a inserção/eliminação no interior de uma lista.
- Quando a busca ocorre segundo o valor do campo correspondente, ao encontrá-lo podemos inserir/eliminar sem a necessidade de ponteiro auxiliar.
- No encadeamento duplo pode ocorrer todo tipo de variação em sua implementação.

Lista Encadeada x Lista Duplamente Encadeada

Lista Encadeada	Lista Duplamente Encadeada
Menor dificuldade de implementação	Maior facilidade de navegação, utilizando os dois sentidos da lista
Uso de um único ponteiro para verificação dos elementos da lista	Diminuição na necessidade do uso de variáveis auxiliares para realizar operações de inclusão e remoção de Nós
Quando não há necessidade de percorrer a lista de trás para frente, a lista simplesmente encadeada é uma melhor opção	Necessidade de uma quantidade menor de variáveis auxiliares para percorrer a lista

Lista Duplamente Encadeada

- Nas listas duplamente encadeadas cada nó possui dois ponteiros, sendo que um aponta para o nó anterior e o outro para o nó posterior. Sendo assim, a lista pode ter seus elementos percorridos a partir de qualquer extremidade.
- Um ponteiro **ant** aponta para o Nó que precede enquanto o ponteiro **prox** aponta para o Nó que sucede.

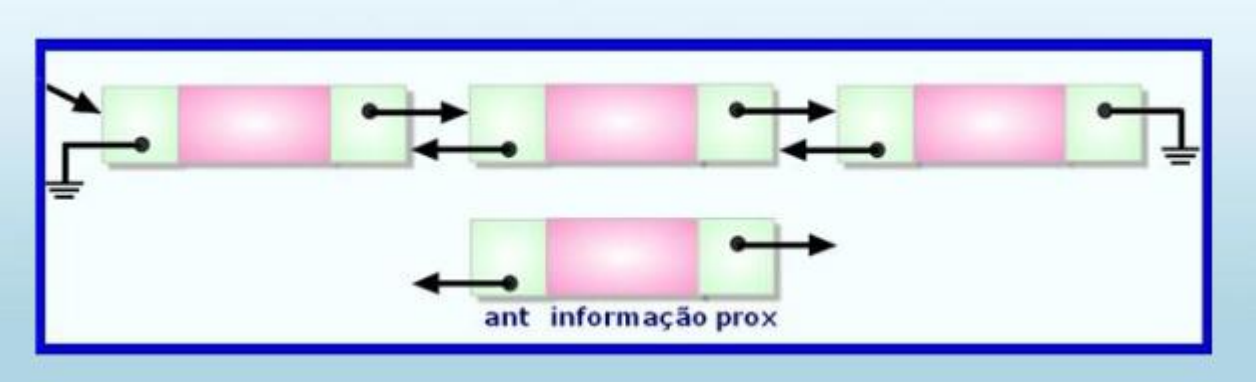


Lista Duplamente Encadeada

- Quando cada nó referência tanto o próximo nó da lista quanto o nó anterior.
- O importante é que, neste tipo de lista, o ponteiro externo pode apontar para qualquer nó da lista, pois é possível caminhar para a direita ou para a esquerda com igual facilidade.
- Uma lista duplamente encadeada pode ser circular ou não e ainda, pode estar em ordem (crescente/decrescente) ou não.

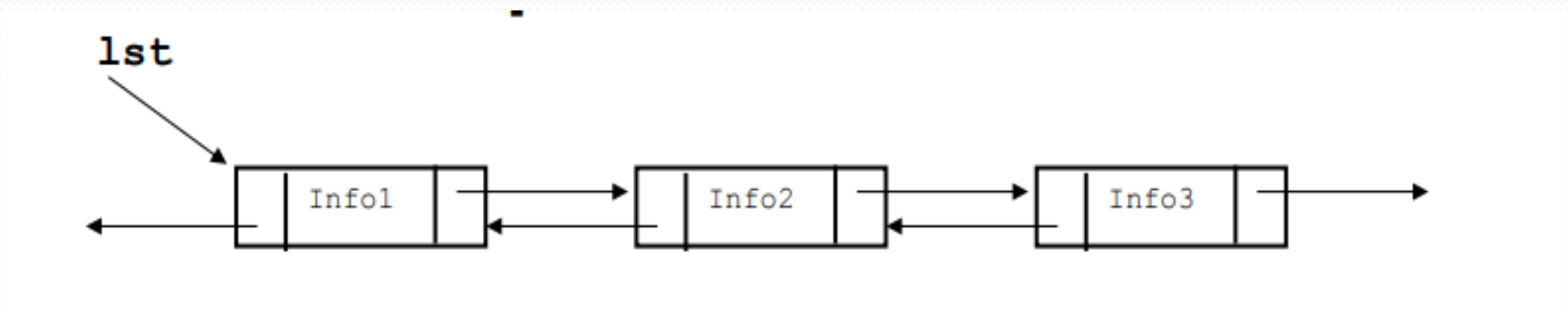
Lista Duplamente Encadeada

- Nas listas duplamente ligadas não existe necessidade de dimensionar o número de Nós, porque tal como ocorre nas Listas Ligadas a alocação vai sendo feita de acordo com a necessidade, de maneira dinâmica na memória.
- As listas duplamente ligadas são recomendadas quando precisamos percorrer uma lista do fim para o início.



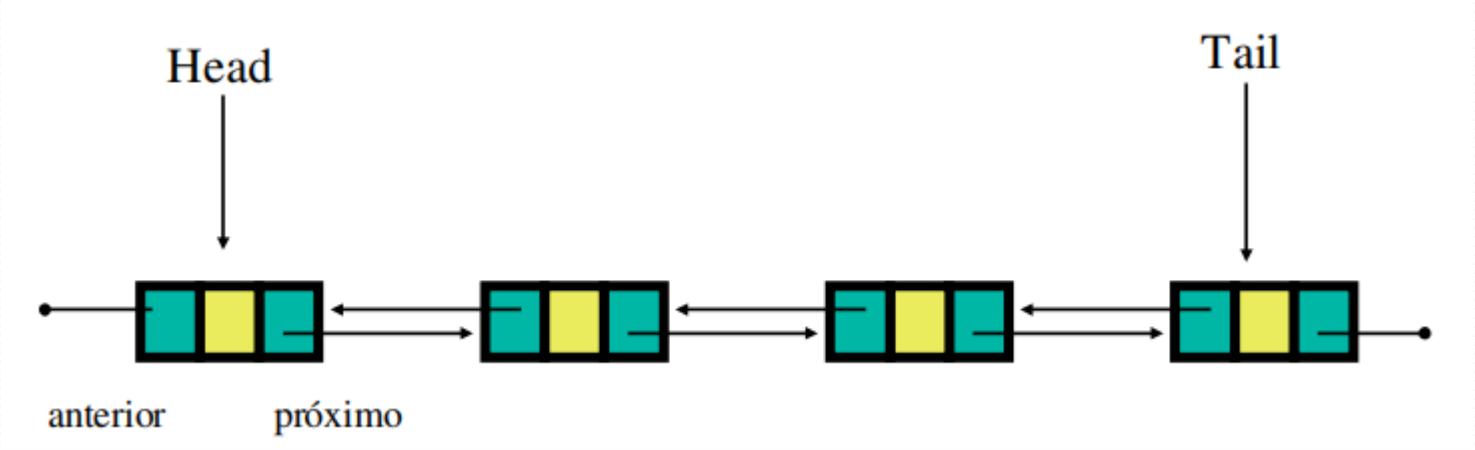
Lista Duplamente Encadeada

- Cada elemento tem um ponteiro para o próximo elemento e um ponteiro para o elemento anterior
- Dado um elemento, é possível acessar o próximo e o anterior
- Dado um ponteiro para o último elemento da lista, é possível percorrer a lista em ordem inversa



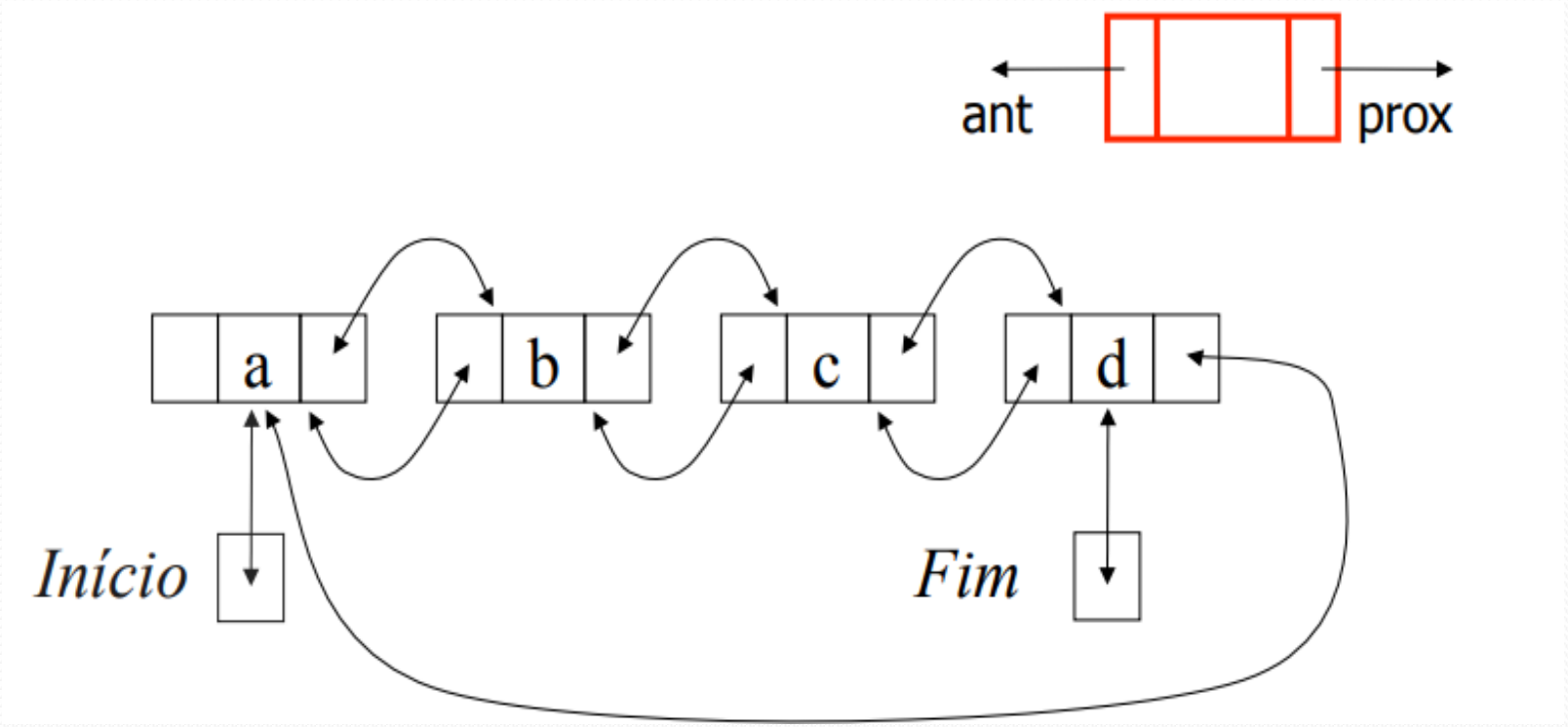
Lista Duplamente Encadeada

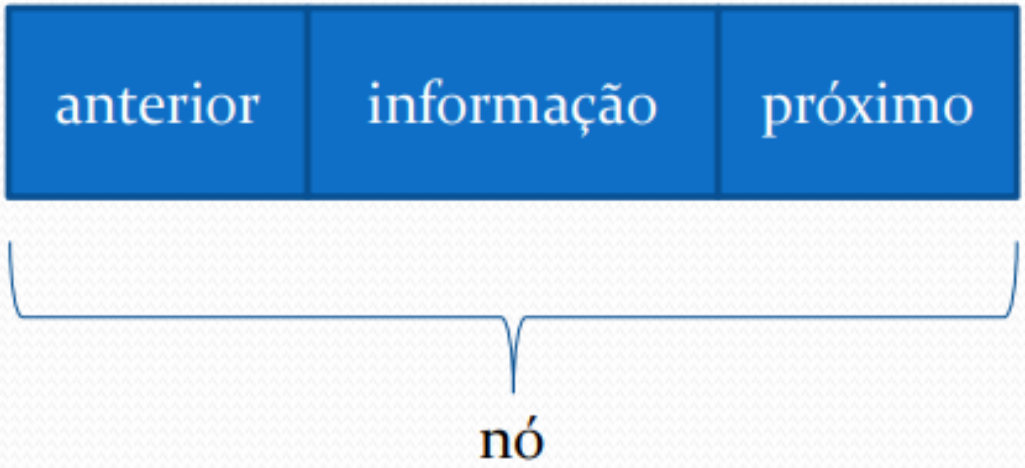
- Cada elemento possui 3 partes:
  - Dados, ponteiro para o próximo, ponteiro para o anterior
  - Último elemento tem NULL como próximo
  - Primeiro elemento tem NULL como anterior É possível caminhar pela lista nos dois sentidos

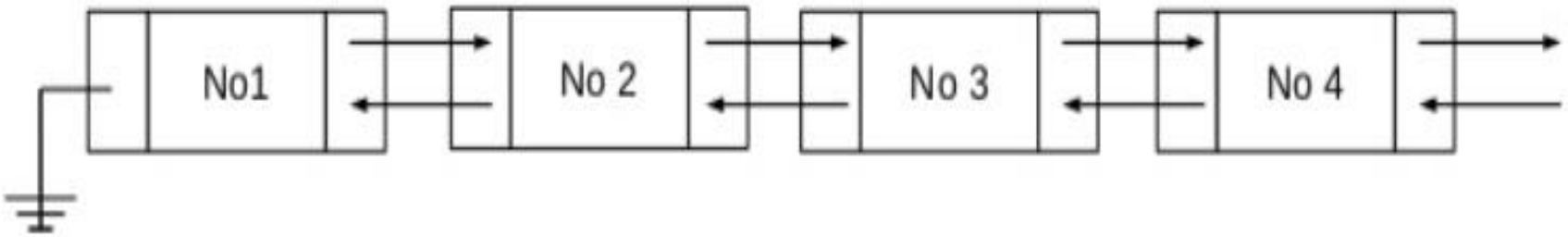


Lista Duplamente Encadeada

- Quando for preciso seguir a sequência de elementos em ambos os sentidos.
- Possui um conjunto maior de ligações a serem atualizadas
- Cada nó possui dois ponteiros: **ant e prox.**




anterior
informação
próximo
nó



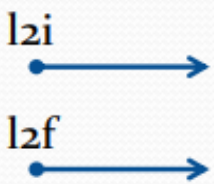
Em uma lista duplamente encadeada é possível percorrer a lista em ambas as direções.

Também apresenta uma maior segurança do que uma lista simplesmente encadeada uma vez que, existe sempre dois ponteiros apontando para cada registro.

Criação e Inicialização da Lista

- Declara-se o ponteiro para o primeiro nó da lista e o ponteiro para o último nó da lista. Aqui chamado de **l2i** e de **l2f** respectivamente.

```
// criação
noDupla *l2i; // ponteiro para o primeiro elemento da lista
noDupla *l2f; // ponteiro para o último elemento da lista

l2i = l2f = NULL;      // inicialização
```

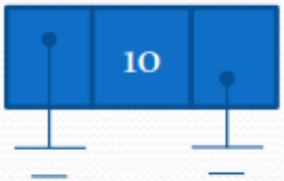


Exemplo

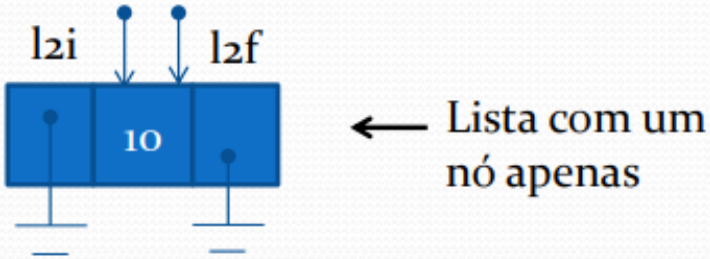
- Construir uma lista com um nó apenas com o valor 10:

novo = new noDupla;

novo->dado = valor;
novo->prox = NULL;
novo->ant = NULL;



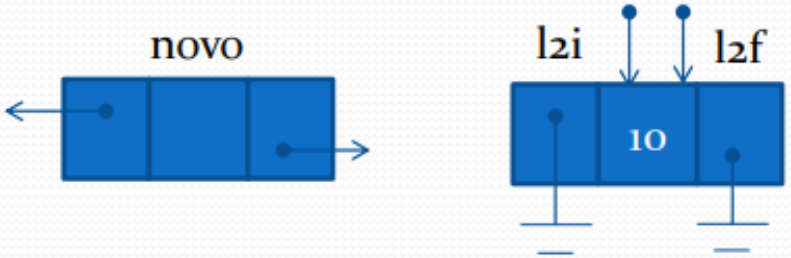
l2i = novo;
l2f = novo;



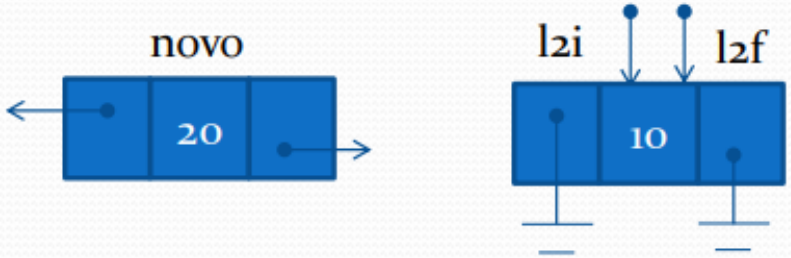
Exemplo

- Inserindo mais um nó na lista (antes do primeiro nó – inserir na frente)

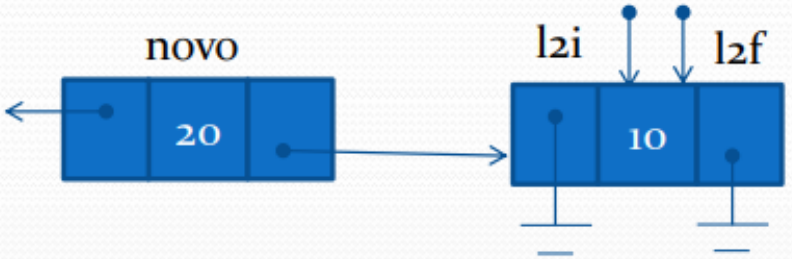
novo = new noDupla;



novo-> dado = 20;



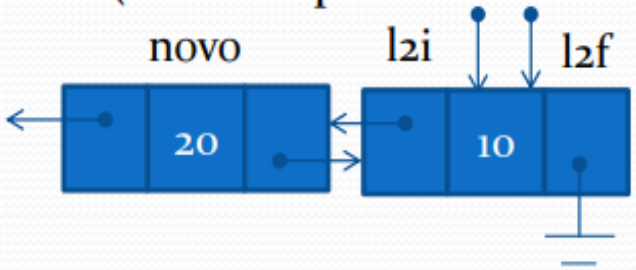
novo-> prox = l2i;



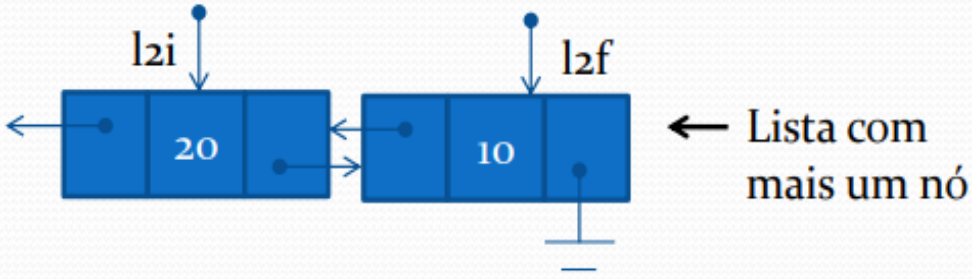
Exemplo

- Inserindo mais um nó na lista (antes do primeiro nó – inserir na frente)

l2i->ant = novo;



l2i = novo;

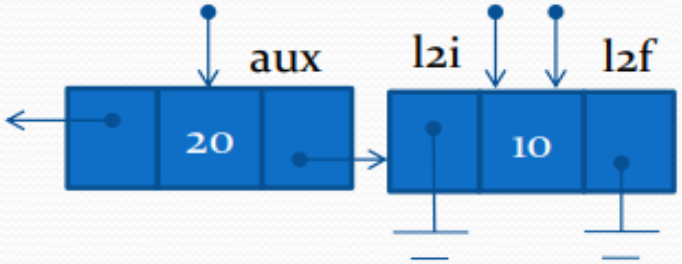


Código da Inserção no Início

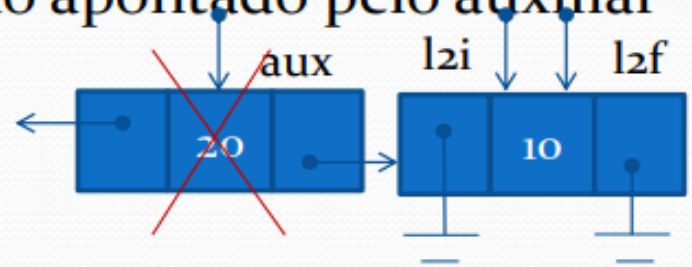
```
void inserirInic(int valor) {
    // cria um novo no
    noDupla *novo = new noDupla;

    novo->dado = valor;
    novo->prox = NULL;
    novo->ant = NULL;

    // inserindo no inicio da lista
    if (l2i != NULL) {
      novo->prox = l2i;
      l2i->ant = novo;
    }
    else
      l2f = novo;
    l2i = novo;
}
```

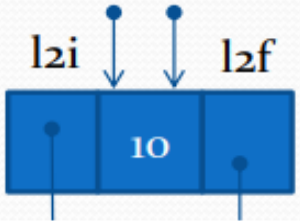
Removendo o primeiro da Lista

- Ponteiro *anterior* do *início da lista*



- Remove nó apontado pelo auxiliar



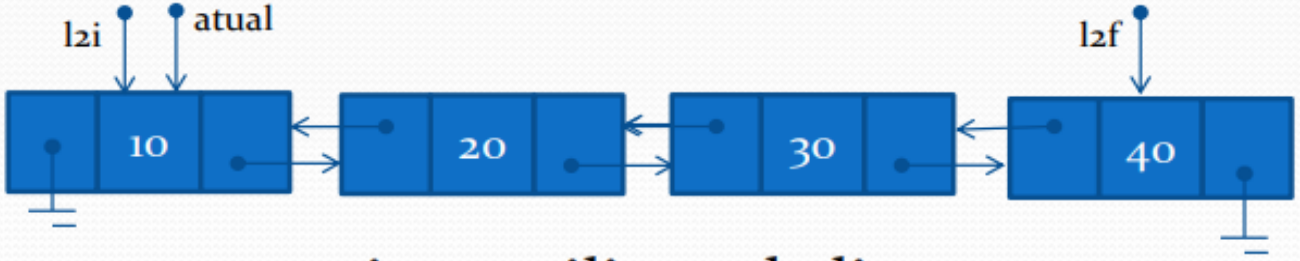


Código da Remoção de um nó do Início

```
noDupla* removerInic () {
   noDupla *aux = l2i;
   l2i = l2i->prox;
   if (l2i == NULL) // se foi removido o ultimo elemento que existia na lista
      l2f = NULL; // ultimo tb ficara apontando para nada
   else
      l2i->ant = NULL;
   return aux;
}
```

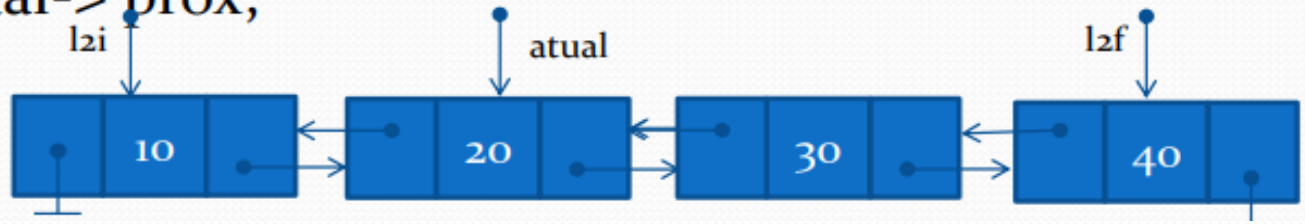
Percorrer a Lista (mostrar)

- Verifica-se se a lista não está vazia
- Usando um ponteiro auxiliar (aqui chamado de atual) percorre-se a lista a partir do primeiro (l2i)



- Move-se o ponteiro auxiliar pela lista:

atual= atual-> prox;



Contatos

- Email: fabio.silva321@fatec.sp.gov.br
- Linkedin: https://br.linkedin.com/in/b41a5269