Estrutura de Dados – 2º semestre de 2020

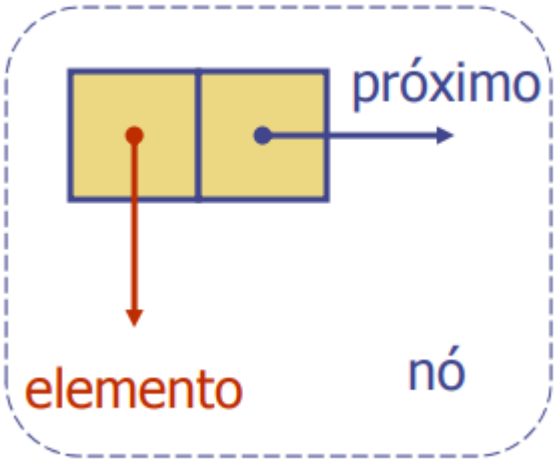
Professor Mestre Fabio Pereira da Silva

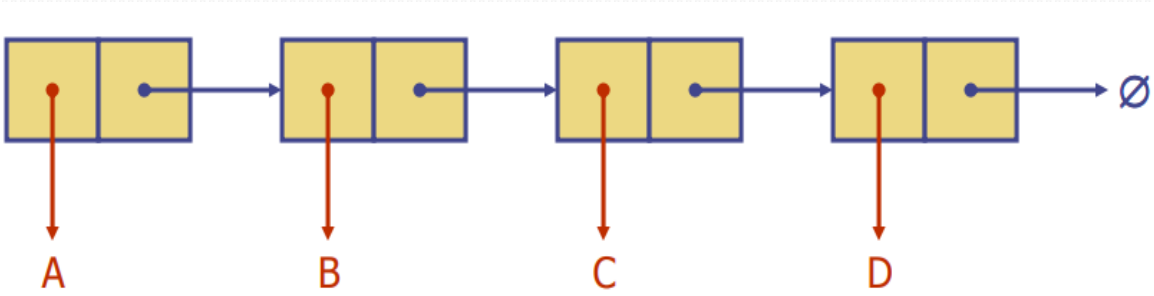
Lista Simplesmente Encadeada - Limitações

- Não se pode atravessar uma lista desse tipo no sentido contrário
- Para eliminar um nó de uma lista simplesmente encadeada, é preciso usar pelo menos uma variável auxiliar do tipo Nó
  - A solução é usar Lista Duplamente Encadeada

Lista Simplesmente Encadeada

- Uma lista encadeada é uma estrutura de dados concreta consistindo de uma sequência de nós
- Cada nó armazena
- Um elemento
- Uma ligação com o próximo nó





Lista Simplesmente Encadeada

- As listas encadeadas permitem fácil inserção e remoção de elementos sem um impacto global na estrutura.
- A lista encadeada é vantajosa quando há elementos que apresentam prioridade de acesso. S **Desvantagem da lista encadeada:**
- O acesso sequencial.
  - Para acessar um nó no meio da lista, todos os nós anteriores (ou posteriores) devem ser visitados.
- A necessidade de armazenar informações adicionais
  - os ponteiros para outros nós.

Vetores x Listas Dinâmicas

- Vetores (arrays):
  - Ocupa um espaço contíguo de memória
  - Permite acesso randômico
  - Requer pré-dimensionamento de espaço de memória
- Estruturas Dinâmicas
  - Crescem (ou decrescem) à medida que elementos são inseridos (ou removidos) –
  - Exemplo: Listas Encadeadas
  - Outras estruturas:
  - Pilhas
  - Filas

Alocação de memória

- Reservar na memória (principal), o espaço para guardar a informação através da declaração de uma variável.
- Estática: É a alocação do espaço de memória antes da execução de um programa em tempo de compilação:
  - int x; flot vet[10]; Produto vProd[500];
- Dinâmica: É a alocação do espaço de memória durante a execução do programa.
  - em tempo de execução.

Ponteiro

- Um ponteiro é uma variável que **aponta** para outra variável. Isto significa que um **ponteiro mantém o endereço de memória de outra variável**.
- Em outras palavras, **o ponteiro não contém um valor no sentido tradicional, mas sim o endereço de outra variável.** Um ponteiro "aponta para" esta outra variável mantendo uma cópia de seu endereço.
- Como um ponteiro contém um endereço, e não um valor, terá duas partes. O ponteiro contém um endereço e o endereço aponta para um valor.

Criando um objeto

- Objeto é uma instância de uma classe;
- Usamos o operador *new* para criar um objeto.

Variável que conterá uma referência a um objeto

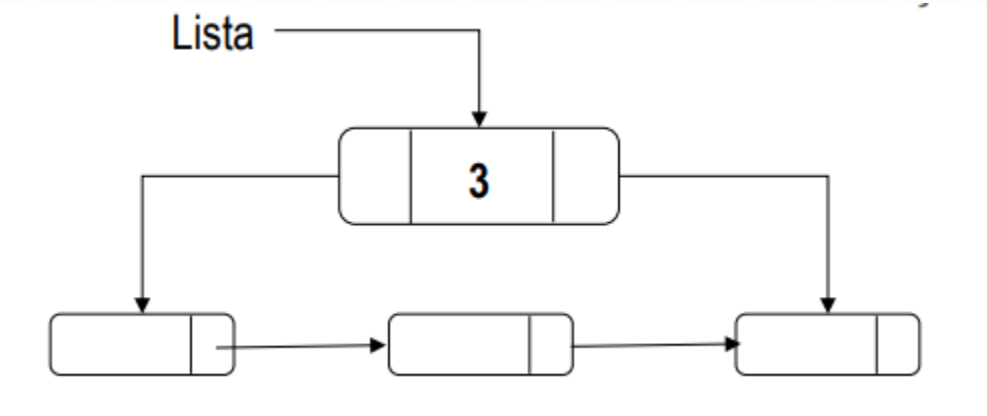
```
ContaCorrente minhaConta;
minhaConta = new ContaCorrente ( );
```

Criação do objeto

ContaCorrente minhaConta = new ContaCorrente ( );

Lista Duplamente Encadeada

- Para facilitar a gerência de informações de início e fim da lista pode-se reunir as referências em uma única estrutura chamada descritor, líder ou header da lista.
- O acesso aos elementos da lista é sempre realizado por meio do header.
- O header pode conter informações como: início e fim da lista, quantidade de nós da lista e outras informações que se deseje



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.

Lista Duplamente Encadeada

- A Lista Encadeada e a Fila Encadeada possuem a desvantagem de somente podermos caminhar em uma direção:
  - vimos que para olhar um elemento pelo qual "acabamos de passar" precisamos de uma variável auxiliar "anterior";
  - para olhar outros elementos ainda anteriores não temos nenhum meio, a não ser começar de novo.
- A Lista Duplamente Encadeada é uma estrutura de lista que permite deslocamento em ambos os sentidos:
  - útil para representar conjuntos de eventos ou objetos a serem percorridos em dois sentidos;
  - ex.: itinerários de ônibus, trem ou avião;
  - útil também quando realizamos uma busca aproximada e nos movemos para a frente e para trás.

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

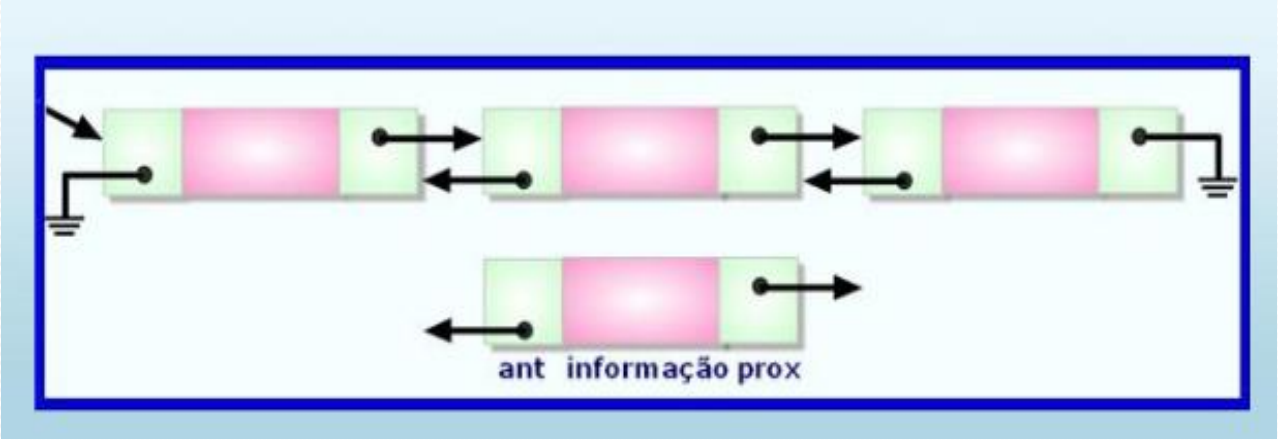
- Facilita a inserção/eliminação no interior de uma lista.
- Quando a busca ocorre segundo o valor do campo correspondente, ao encontrá-lo podemos inserir/eliminar sem a necessidade de ponteiro auxiliar.
- No encadeamento duplo pode ocorrer todo tipo de variação em sua implementação.

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

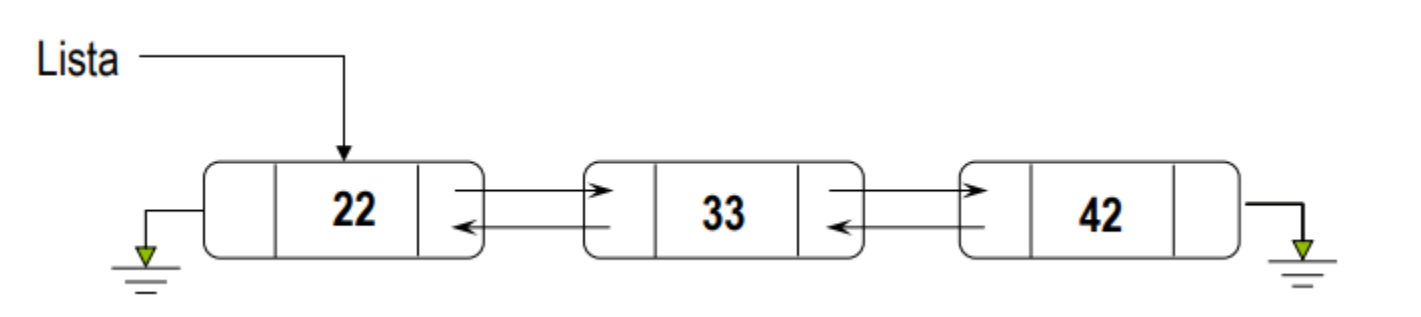
Lista Duplamente Encadeada

- 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.



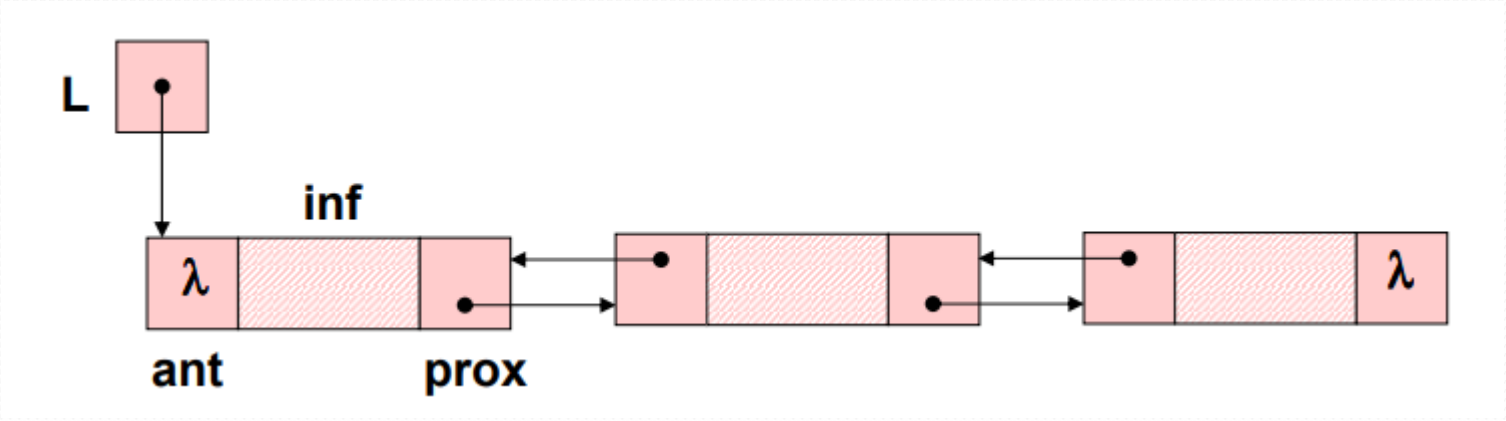
Lista Duplamente Encadeada

- Nas listas duplamente encadeadas, os nós contêm dois ponteiros
- Um para o sucessor e
- Outro para o predecessor.



Lista Duplamente Encadeada

- Em uma lista duplamente encadeada os elementos possuem três campos: o campo inf o qual contém a informação, o campo ant que possui um ponteiro para o elemento antecessor e o campo prox que é uma referência para o elemento que sucede.



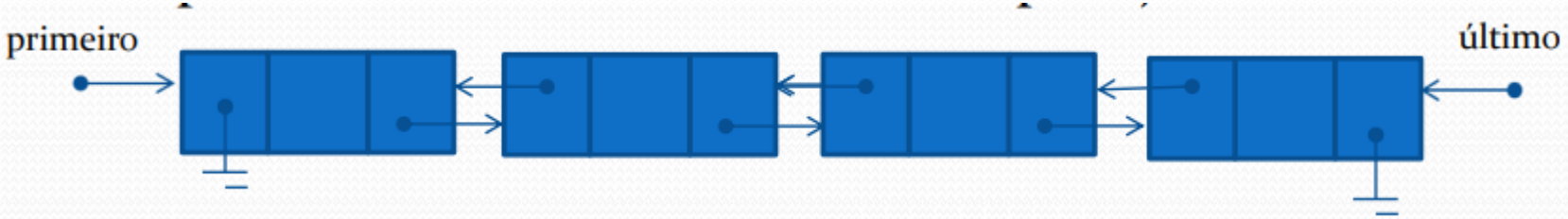
Lista Duplamente Encadeada

- 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

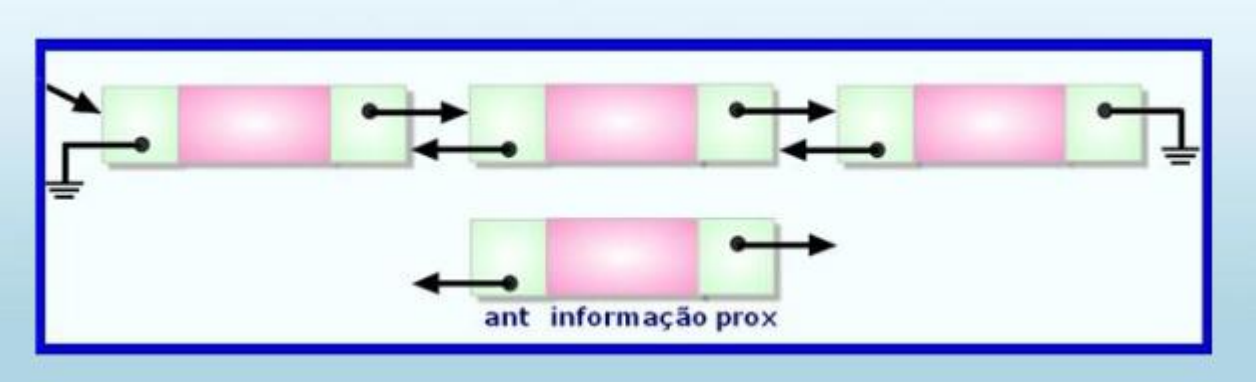
- Um nó da lista é representado por uma estrutura (struct), que deverá conter, no mínimo, três campos : um campo com a informação ou dado a ser armazenado e dois ponteiros: um para o próximo nó da lista e outro para o nó anterior.

- É preciso que o primeiro nó seja apontado por um ponteiro, assim como o último, para que assim, a lista possa ser manipulada através de suas diversas operações.



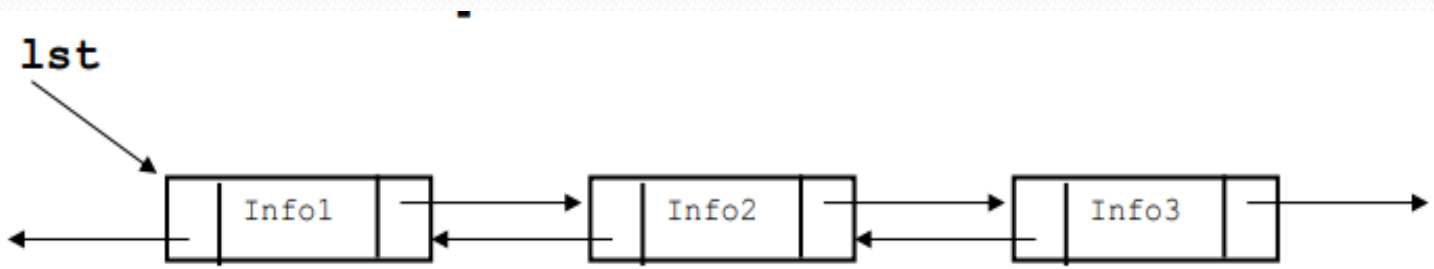
Lista Duplamente Encadeada

- 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.



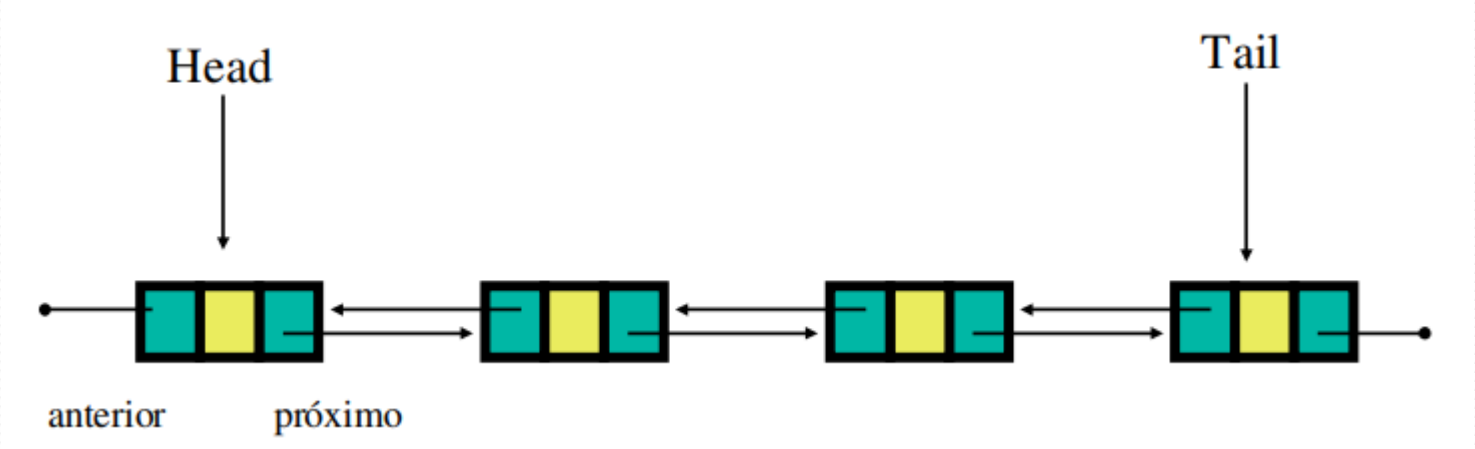
Lista Duplamente Encadeada

- 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



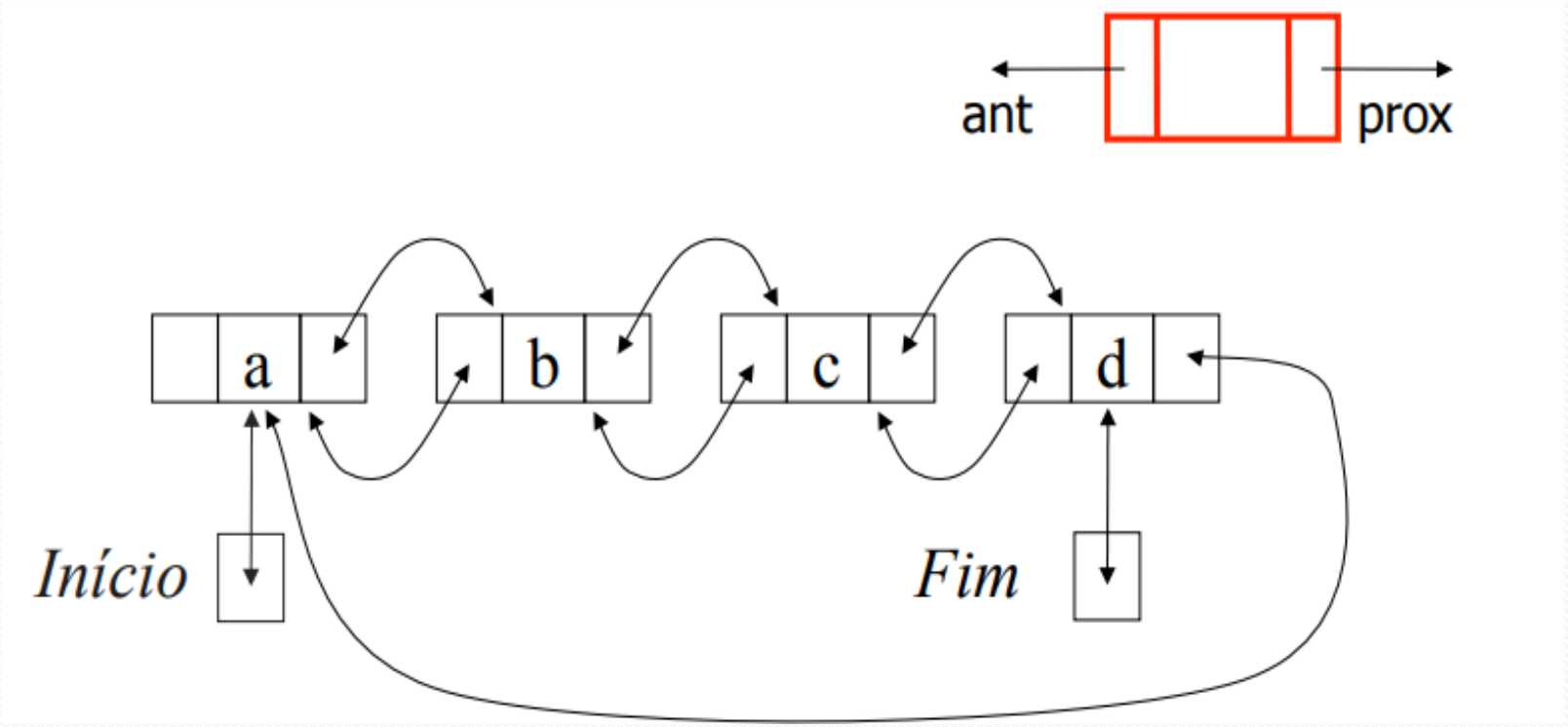
Lista Duplamente Encadeada

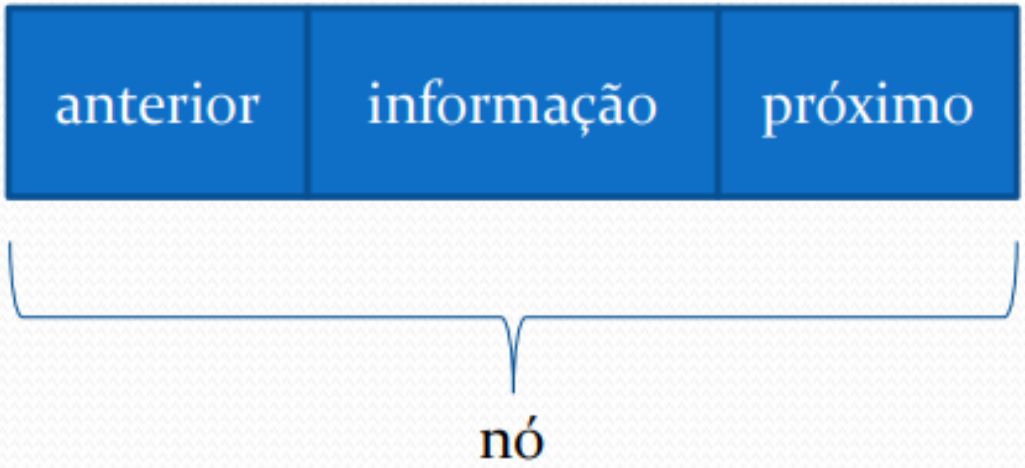
- Cada elemento possui 3 partes:
  - Dados, ponteiro para o próximo, ponteiro para o anterior
  - Último elemento tem NULL como próximo
  - Primeiro elemento tem NULL como anterior É possível caminhar pela lista nos dois sentidos

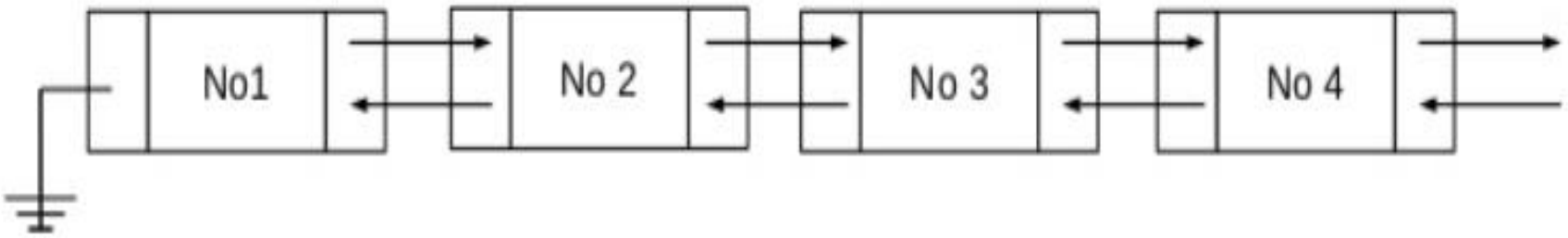


Lista Duplamente Encadeada

- 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

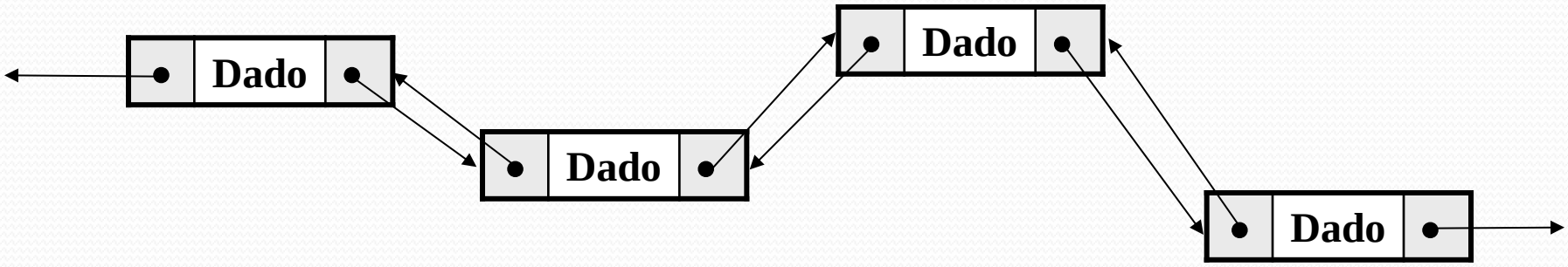


Em uma lista duplamente encadeada é possível percorrer a lista em ambas as direções.

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

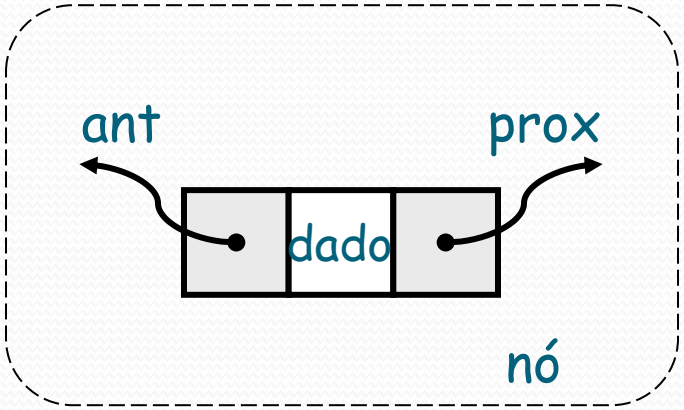
Lista Duplamente Encadeada

- O encadeamento simples também dificulta a retirada de um elemento da lista
  - Temos que percorrer a lista, elemento por elemento, para encontrarmos o elemento anterior àquele que queremos eliminar.
- A solução é usar uma lista duplamente encadeada

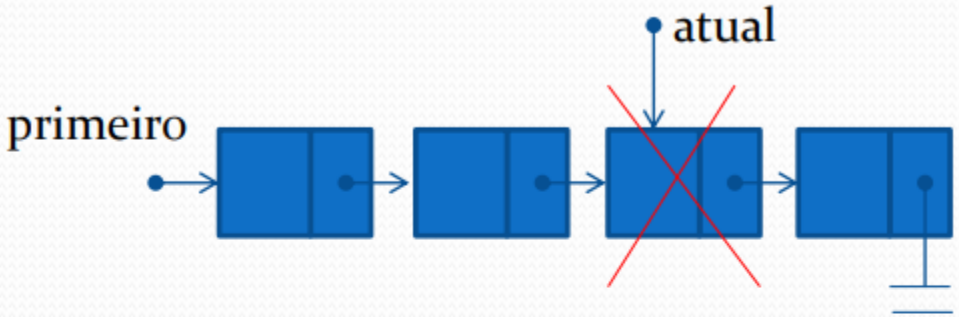


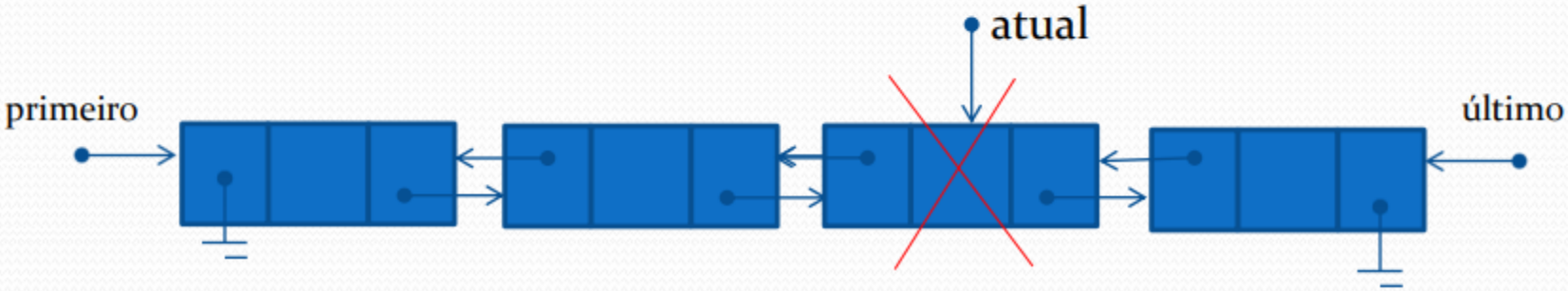
Lista Duplamente Encadeada

- Cada nó armazena :
  - O conteúdo do elemento
  - A ligação para o próximo nó
  - A ligação para o nó anterior



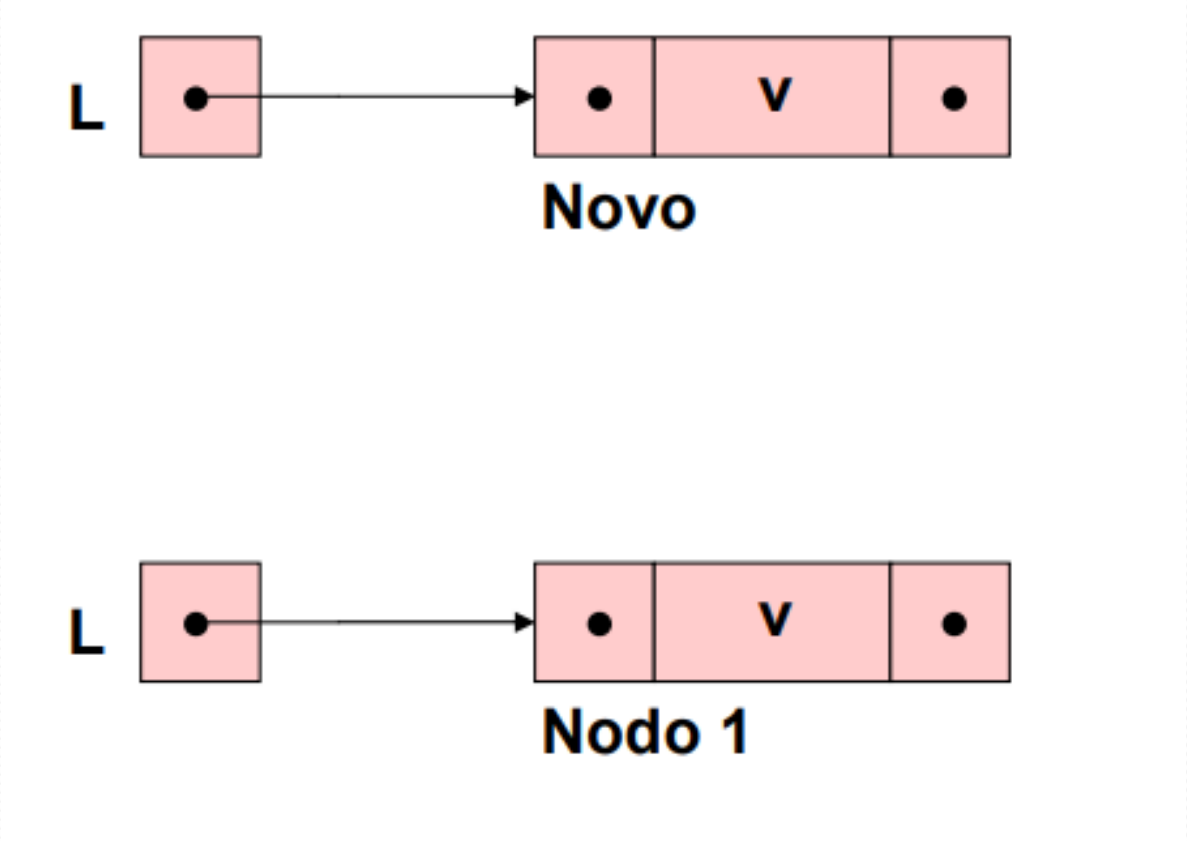
Lista Encadeada x Lista Duplamente Encadeada





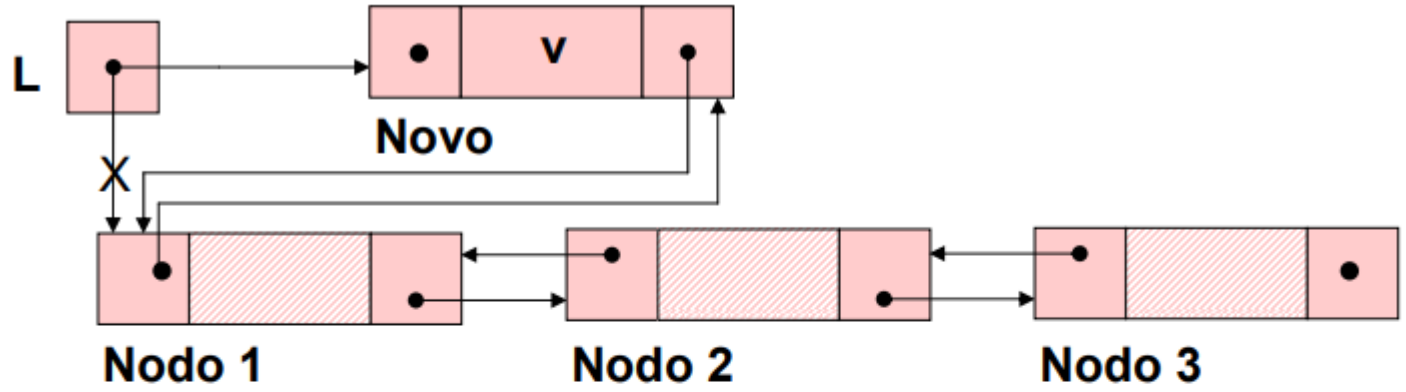
Lista Duplamente Encadeada – Inserção

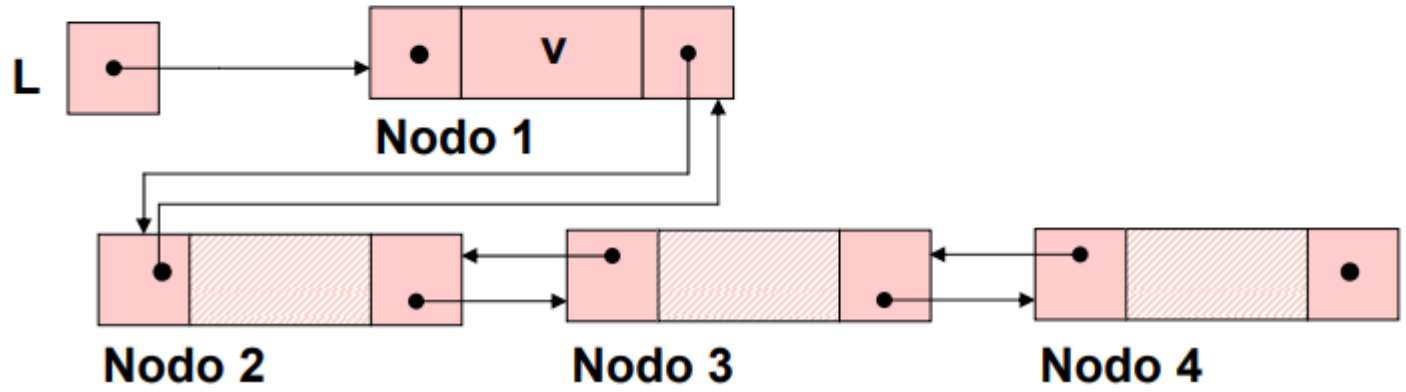
- Esquema do processo da inserção de um nó da lista duplamente encadeada. (Nó único)



Lista Duplamente Encadeada - Inserção

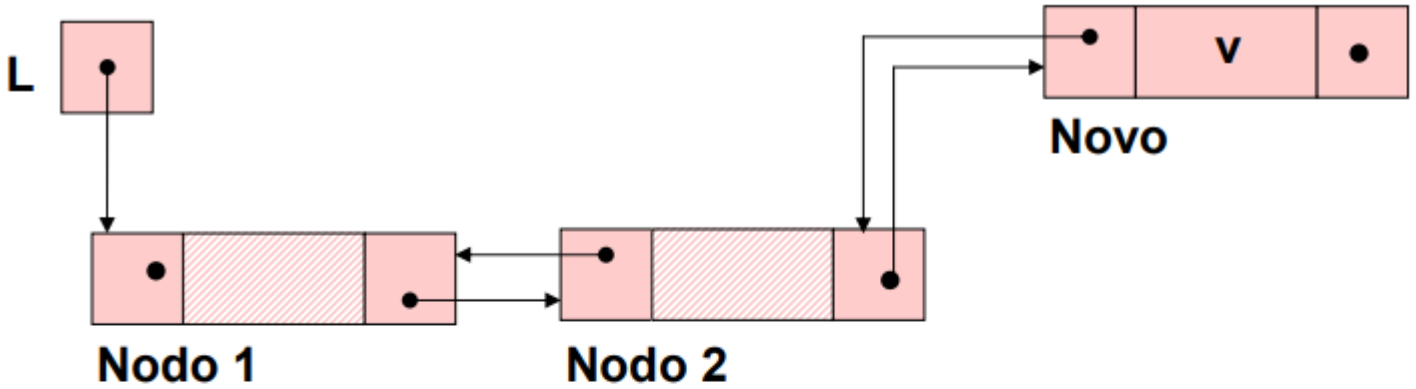
- Esquema do processo da inserção de um nó da lista duplamente encadeada. (Inserção no início)

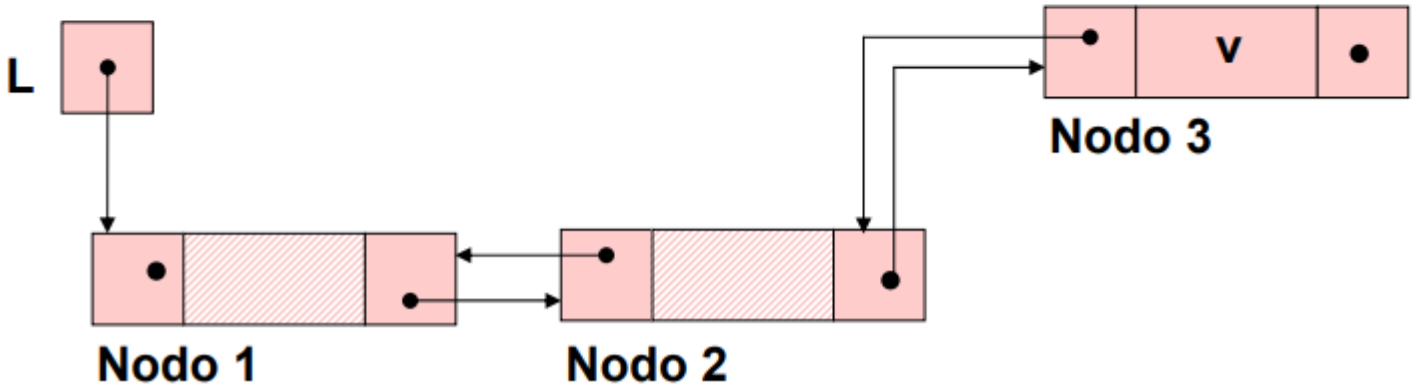




Lista Duplamente Encadeada - Inserção

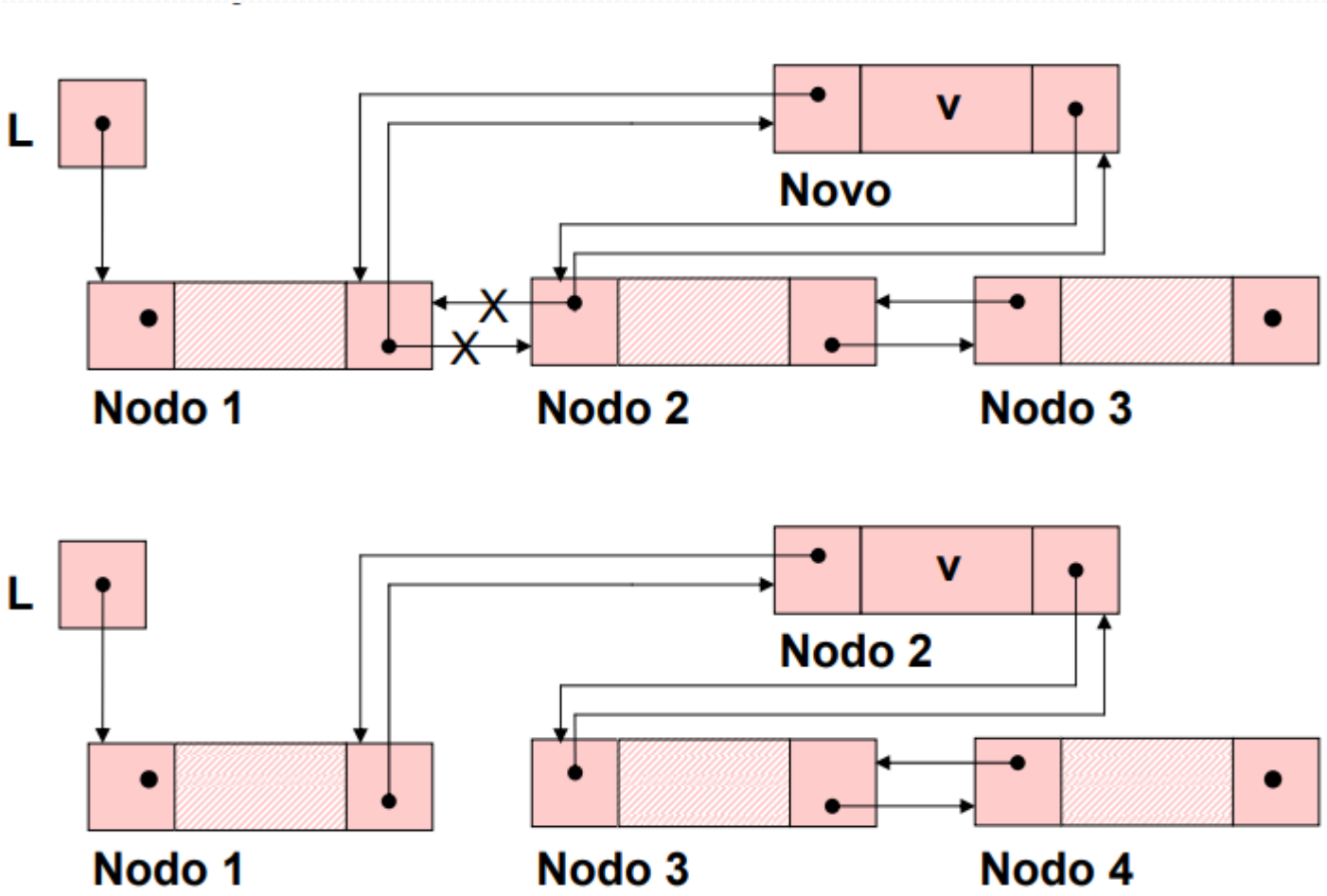
- Esquema do processo da inserção de um nó da lista duplamente encadeada. (Inserção no final)





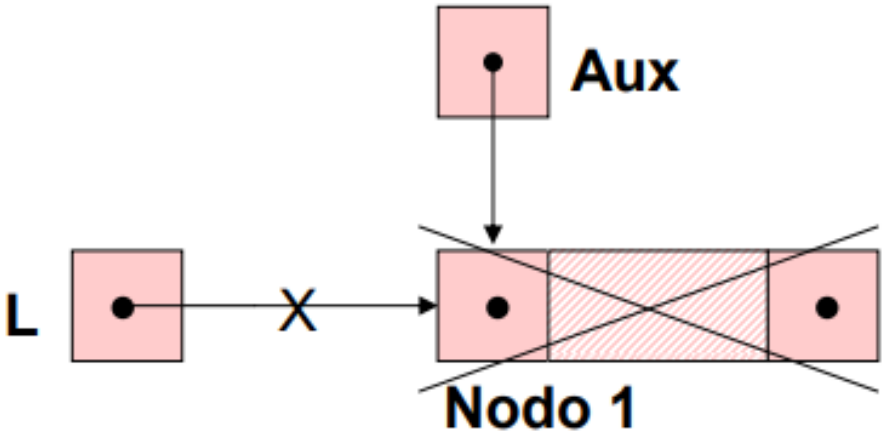
Lista Duplamente Encadeada - Inserção

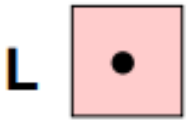
- Esquema do processo da inserção de um nó da lista duplamente encadeada. (inserção em um local especifico)



Lista Duplamente Encadeada - Remoção

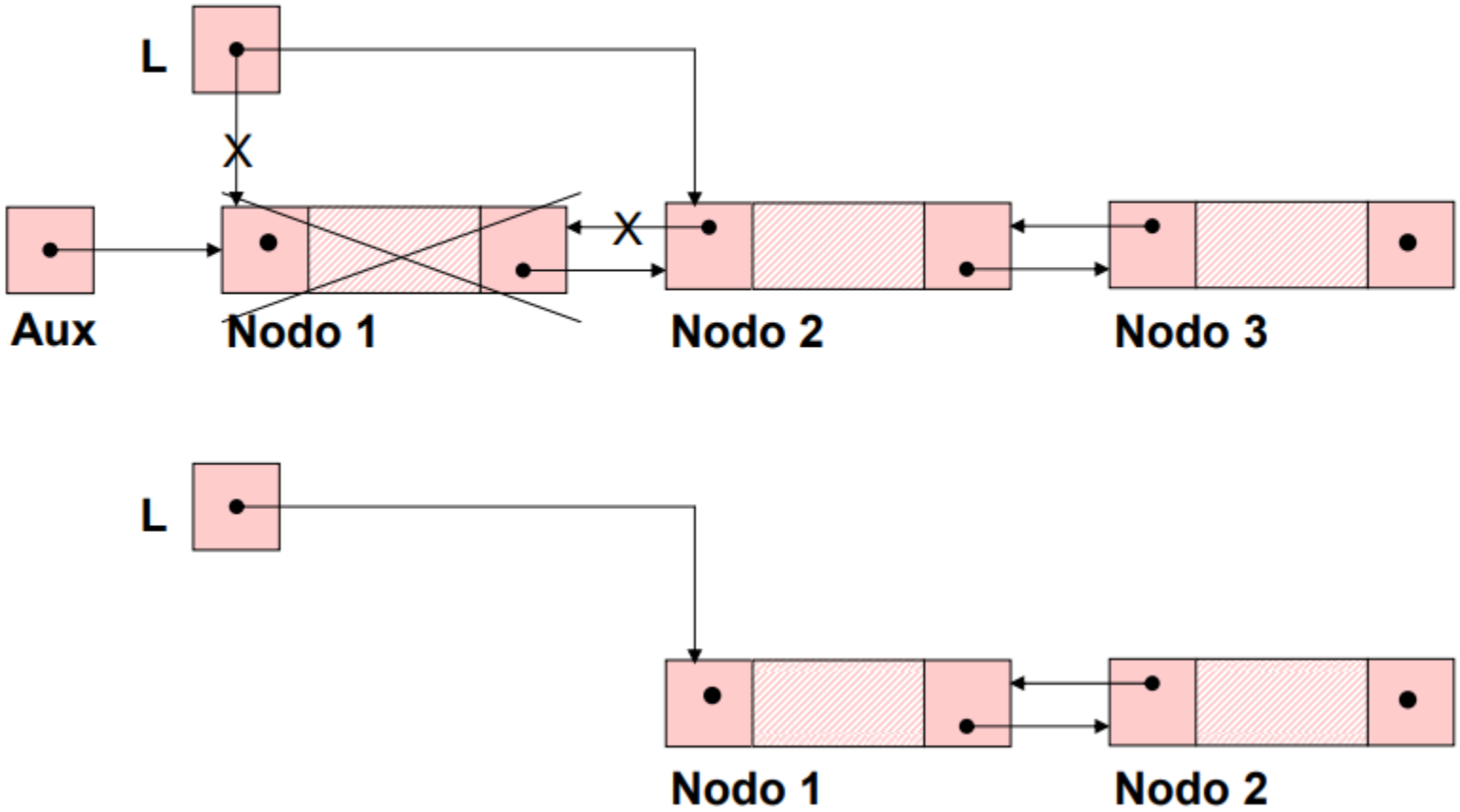
- Esquema do processo da retirada de um nó da lista duplamente encadeada. (remoção de um único Nó)





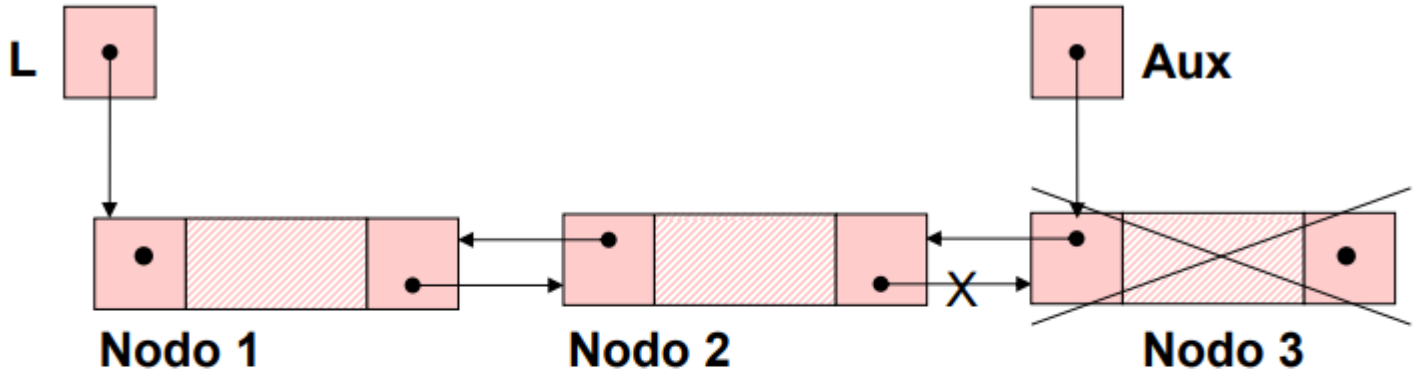
Lista Duplamente Encadeada - Remoção

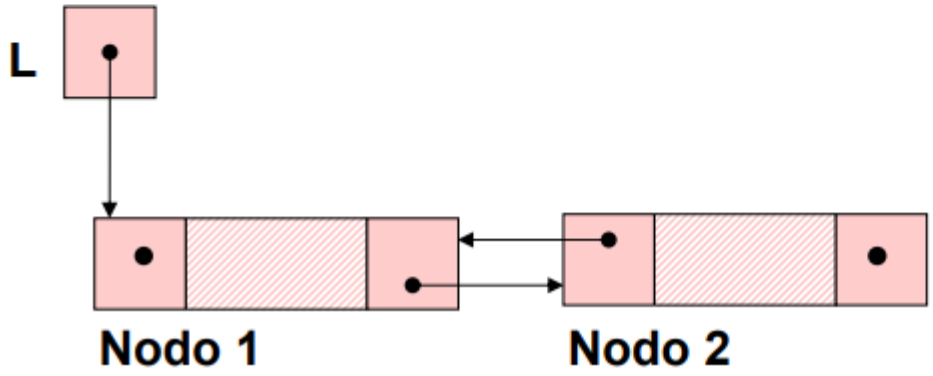
- Esquema do processo da retirada de um nó da lista duplamente encadeada. (remoção no início)



Lista Duplamente Encadeada - Remoção

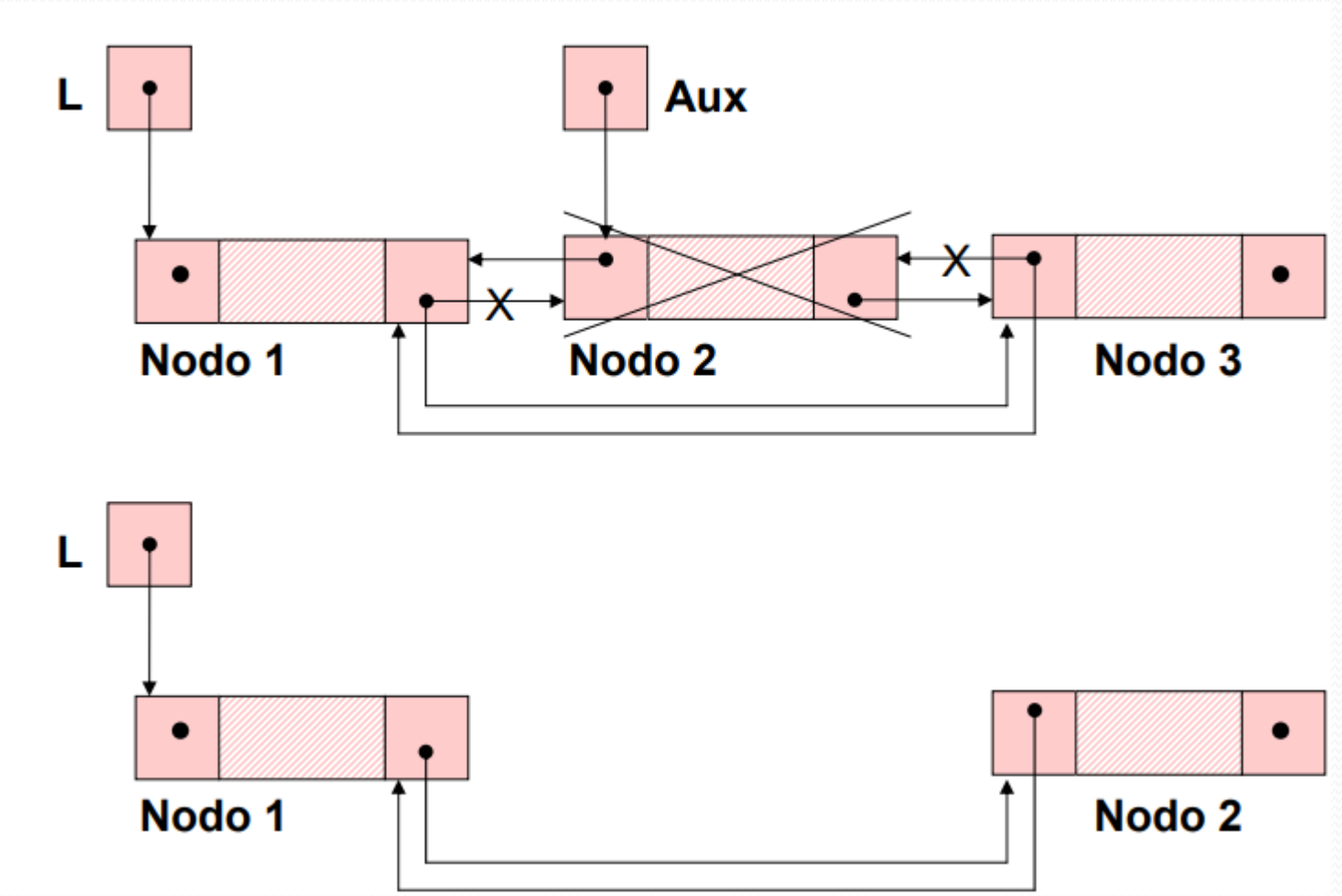
- Esquema do processo da retirada de um nó da lista duplamente encadeada. (remoção no final)



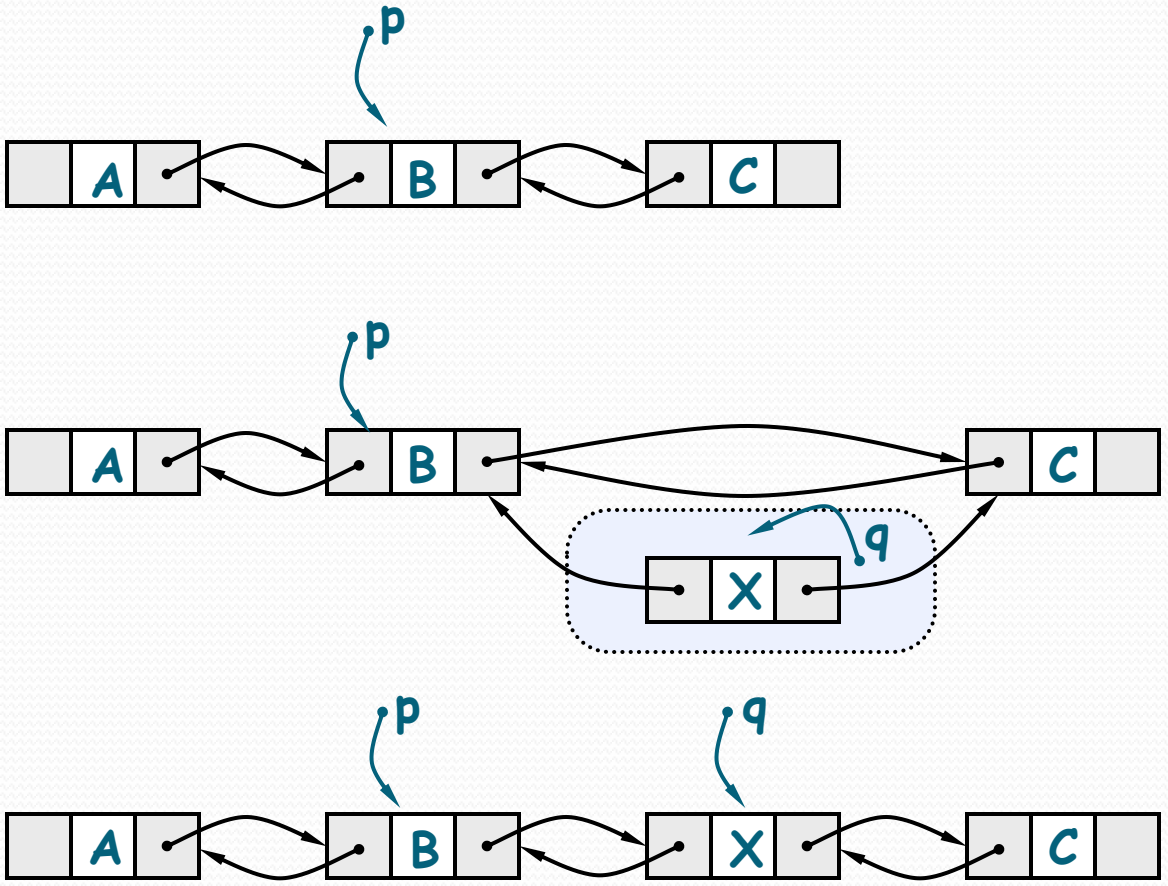


Lista Duplamente Encadeada - Remoção

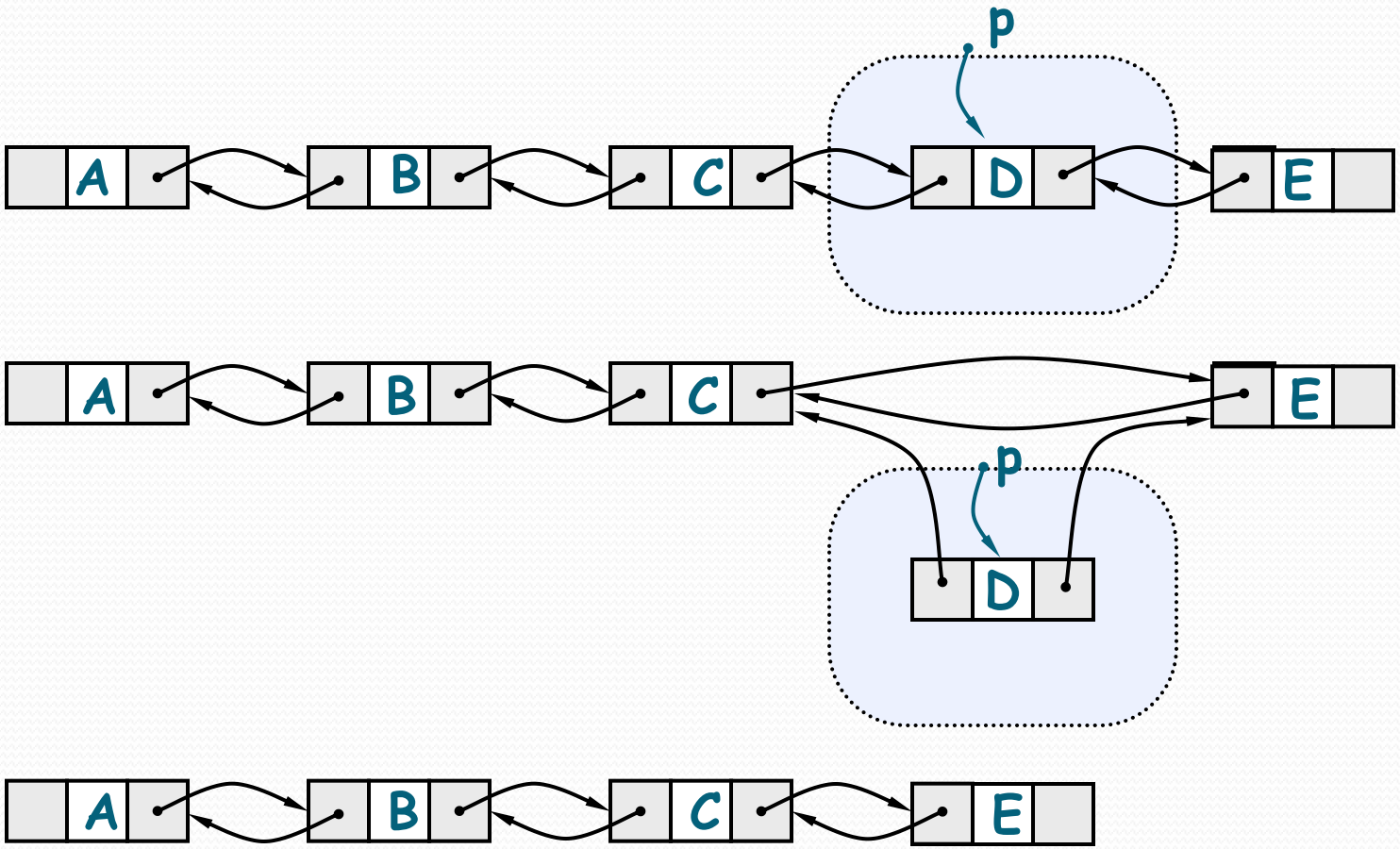
- Esquema do processo da retirada de um nó da lista duplamente encadeada. (remoção em um local específico)



Inserir novo elemento na lista



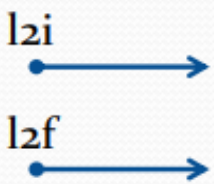
Remover elemento na lista



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
```

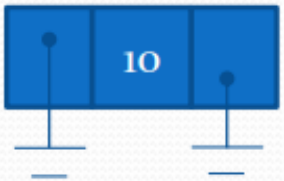


Exemplo

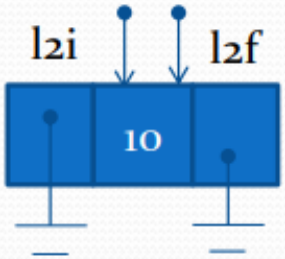
- Construir uma lista com um nó apenas com o valor 10:

```
novo = new noDupla;
```

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




```
l2i = novo;
l2f = novo;
```

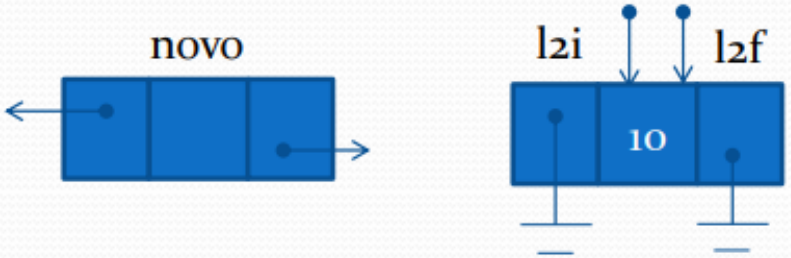


← Lista com um nó apenas

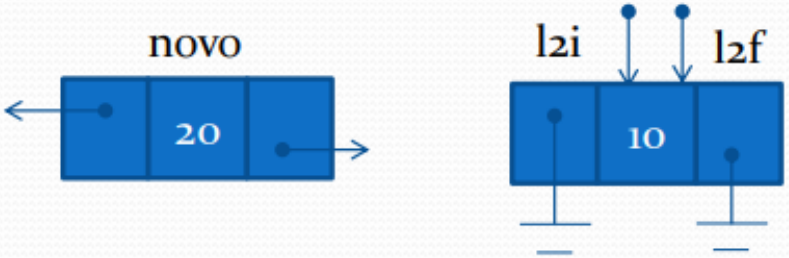
Exemplo

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

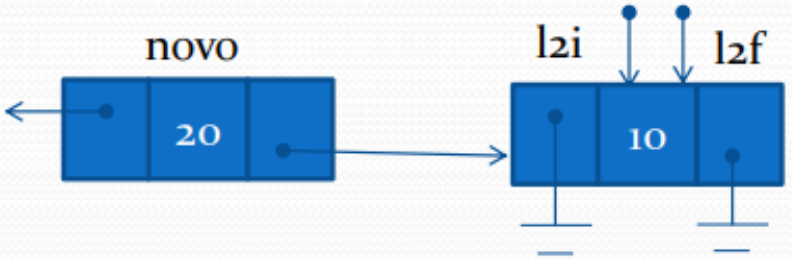
novo = new noDupla;



novo-> dado = 20;



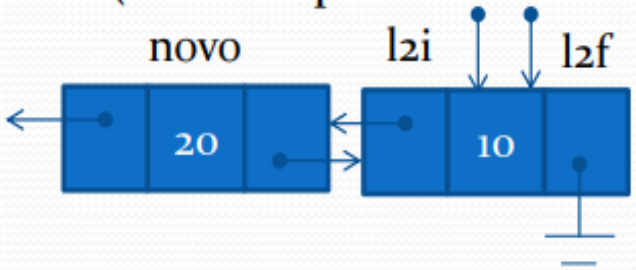
novo-> prox = l2i;



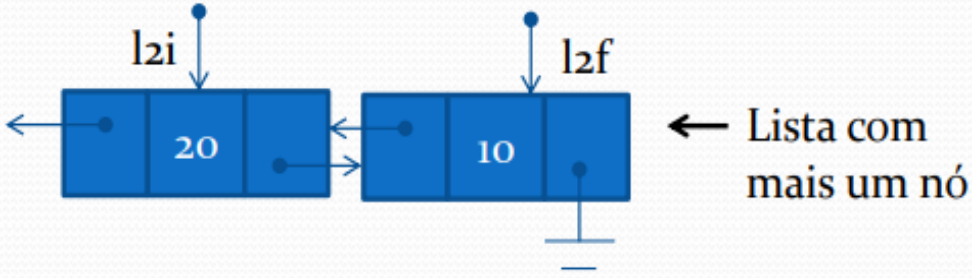
Exemplo

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

l2i->ant = novo;



l2i = novo;

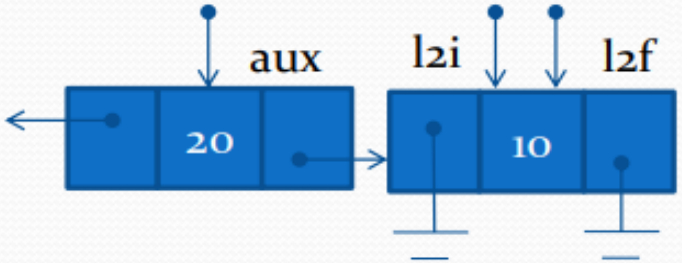


Código da Inserção no Início

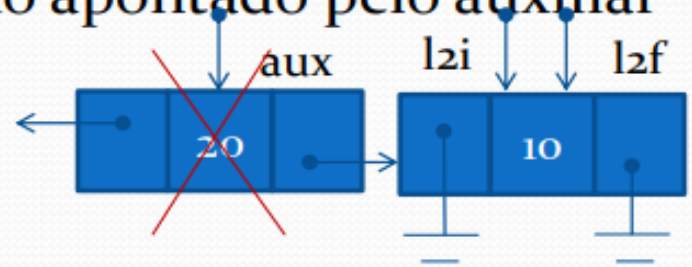
```
void inserirInic(int valor) {
    // cria um novo no
    noDupla *novo = new noDupla;

    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;
}
```

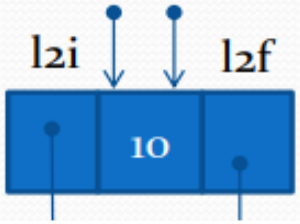
Removendo o primeiro da Lista

- Ponteiro *anterior* do *início da lista*



- Remove nó apontado pelo auxiliar



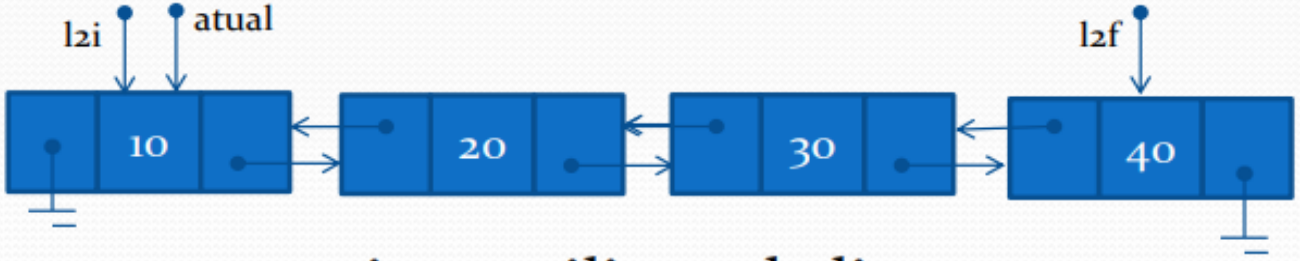


Código da Remoção de um nó do Início

```
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;
}
```

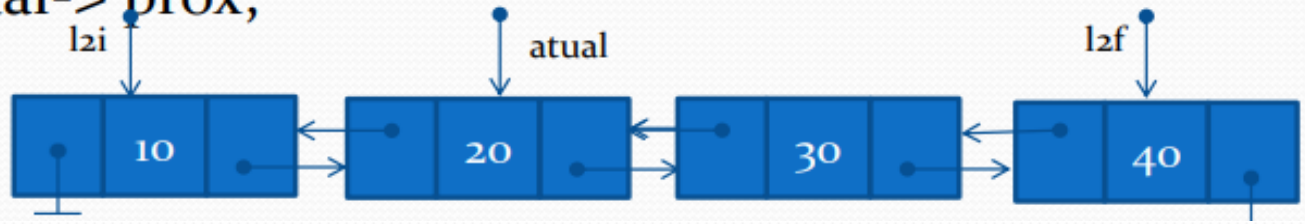
Percorrer a Lista (mostrar)

- 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
- Facebook: https://www.facebook.com/fabio.silva.56211