ESTRUTURA DE DADOS – EDAS2

Tecnologia em Análise e Desenvolvimento de Sistemas

2º. Semestre – 2019

Aula 5 – Lista Duplamente Encadeada

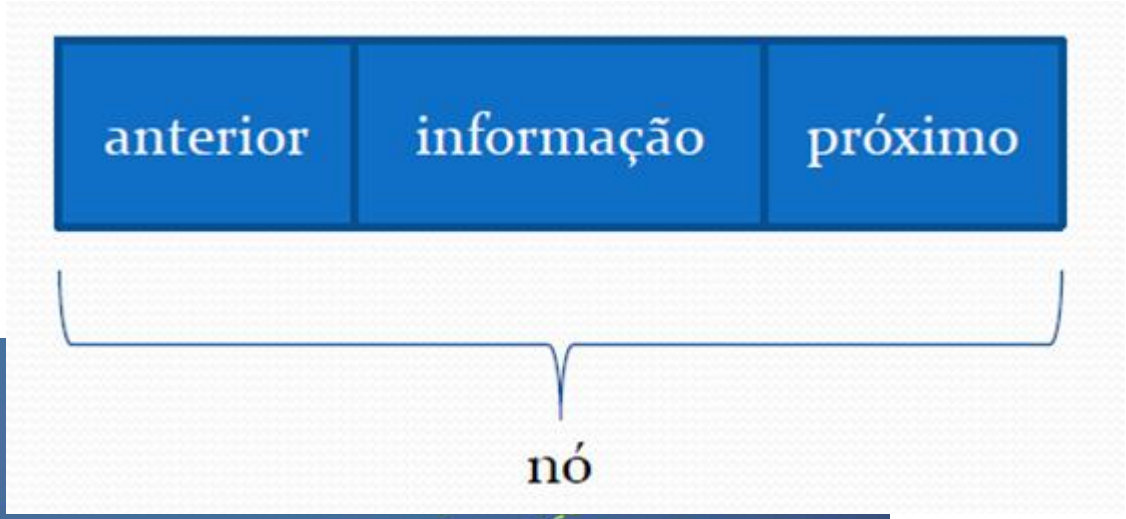
13/09/2019 e 20/09/2019

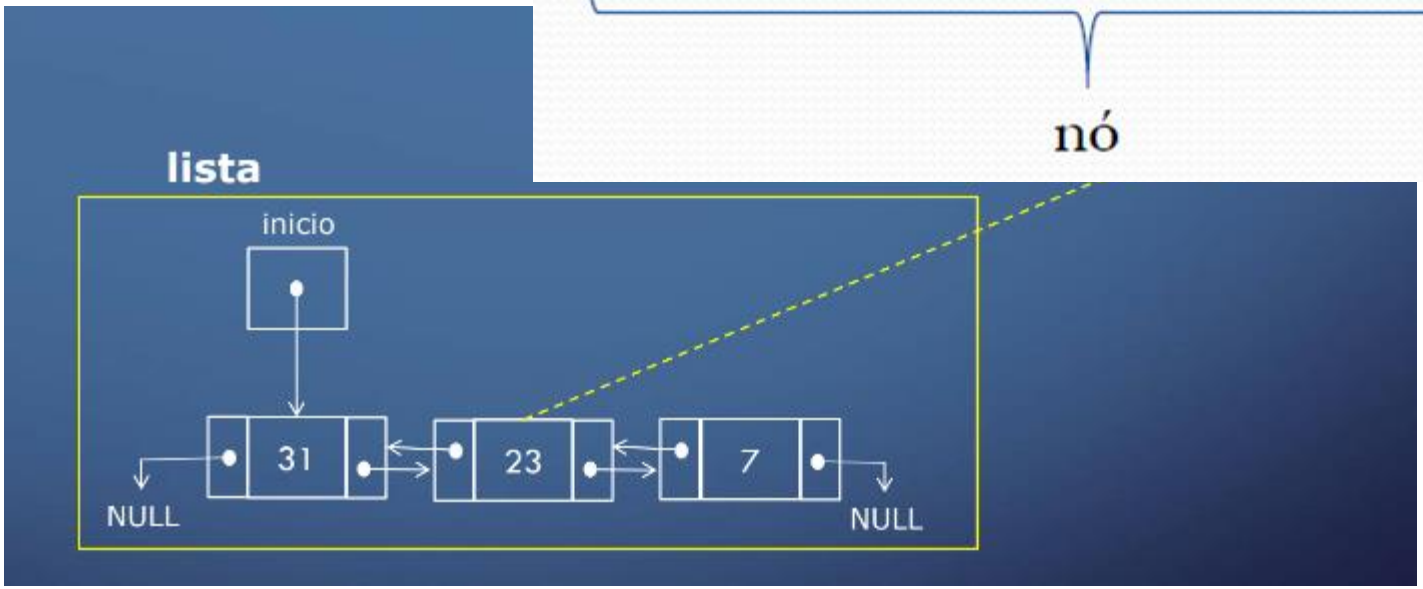
Profa. Dra. Janaina Cintra Abib

Tipos de Listas (Ordenadas ou Não)

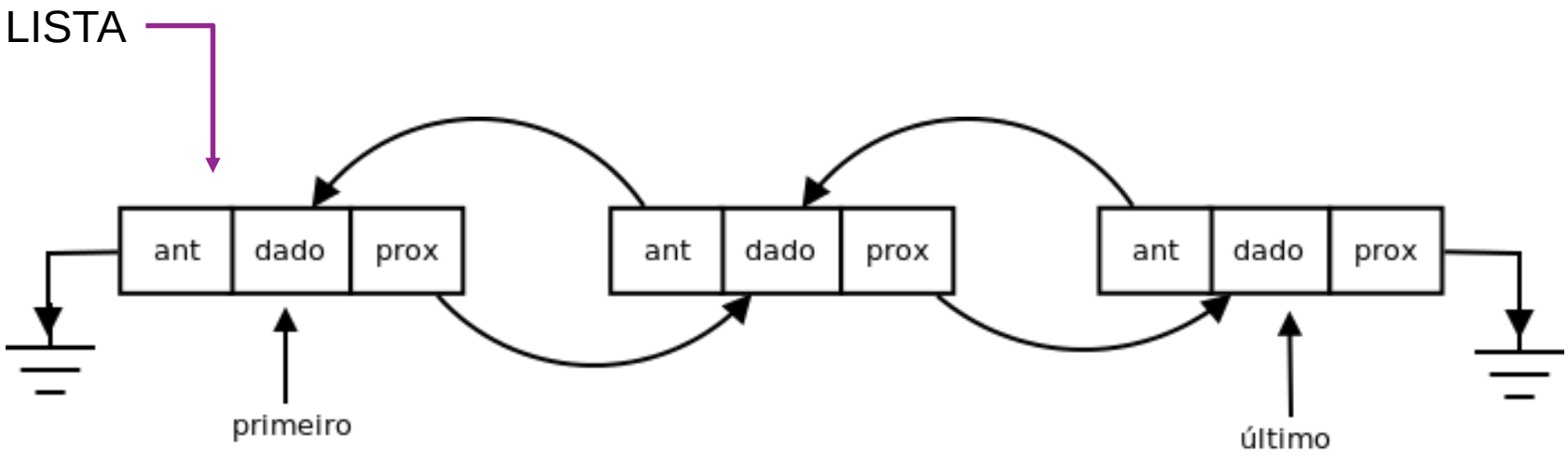
- **Simplesmente Encadeada**
  - Estamos em sequência.
- **Duplamente Encadeada**
  - Sabemos de onde viemos e para onde vamos.
- **Circular**
  - Ela nunca acaba, e seguimos na mesma direção.
- **Circular Duplamente Encadeada**
  - Podemos ir e voltar no anel lógico.
- **Com Descritor**
  - Quem sou, quantos nós tenho, etc.

Duplamente Encadeada - Representação





Representação



PRIMEIRO NÓ
O anterior aponta para o NULL

ÚLTIMO NÓ
O próximo aponta para o NULL

Lista Simples X Lista Dupla

- Vantagem da utilização de lista duplamente encadeada sobre a lista simplesmente encadeada é a maior facilidade para navegação:
  - Lista duplamente encadeada pode ser feita nos dois sentidos
  - Facilita a realização de operações tais como inclusão e remoção de nós, pois diminui a quantidade de variáveis auxiliares necessárias.

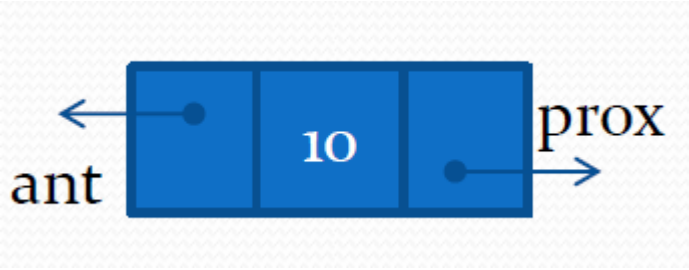
- Se não existe a necessidade de se percorrer a lista de trás para frente, a lista simplesmente encadeada é a mais interessante, pois é mais simples.

- Quando o tamanho da lista é desconhecido (alocação dinâmica de elementos);
- Quando a inserção e a remoção ordenadas ocorrerem com frequência.

Representação da Estrutura

```
struct no
{
        int info;
        struct no *ant;
        struct no *prox;
};

typedef struct no DUPLA;
```



Operações - Lista Duplamente Encadeada

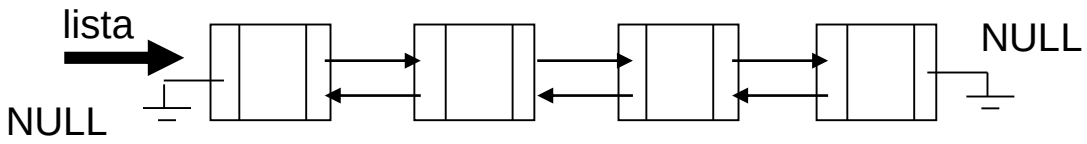
- LDE não ordenada:
  - inserção no início da lista;
  - inserção no final da lista;
  - exclusão;
  - consulta;

- LDE ordenada:
  - inserção;
  - exclusão;
  - consulta;

© Can Stock Photo - csp4137294

Para simplificar nosso entendimento, neste primeiro momento vamos definir que nossa informação (INFO) será um valor inteiro, mas isso mudará em breve.

Representação da Lista Duplamente Encadeada



- Possui os nós interligados, sendo que um ponteiro do último nó da lista não aponta para nenhum outro nó e um ponteiro do primeiro nó não aponta para nenhum.

- Existe um **ponteiro Externo** a lista que aponta para seu primeiro nó e todo nó da Lista só é acessível através do ponteiro externo.

- **Para facilitar vamos chamar os elementos do nó:**
- Campo informação: info
- Campo apontador (ponteiro): ant e prox
- Ponteiro Externo: lista
- O ant do primeiro nó aponta para NULL
- O prox do último nó aponta para NULL

Lista Vazia
É a lista sem nós. Seu ponteiro externo (lista) aponta para NULL.

Podemos inicializar uma lista vazia pela operação:

lista = null

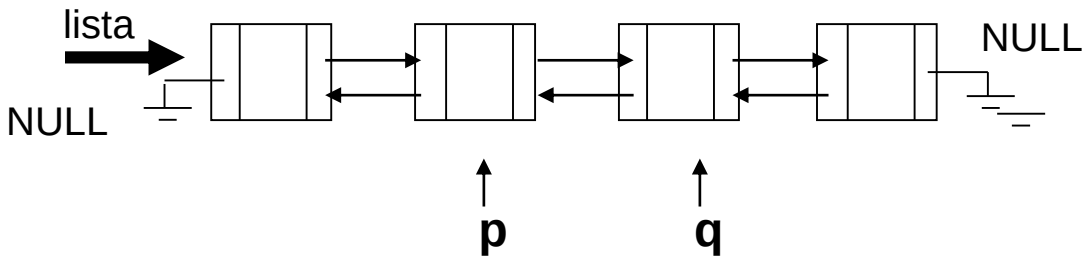
```
void criarLista (DUPLA **l)
{
     *l = NULL;
}
```

Notação a ser utilizada:
Se p é um ponteiro para um nó:
p: refere-se ao nó
p.info: refere-se à parte da informação desse nó
p.ant: endereço seguinte (também é um ponteiro)
p.prox: endereço seguinte (também é um ponteiro)

```
DUPLA* criarLista2()
{
     DUPLA *l;
     l = (DUPLA*)malloc(sizeof(DUPLA));
     if(l != NULL)
     {
          l = NULL;
     }
     return(l);
}
```

Assim, dada a seguinte lista, onde:

Temos que:
p.prox é igual a q
q.ant é igual a p

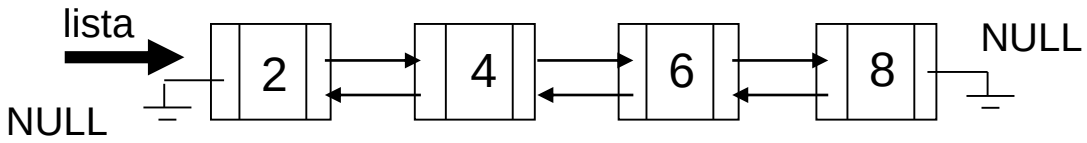


Operações na Lista – **NÃO ORDENADA**

INSERIR NO INÍCIO

Inserindo nó na lista:

Supondo que temos a seguinte lista de inteiros:



E queremos incluir o elemento 5 na **PRIMEIRA** posição dessa lista (no início).

1º passo:
Obter um nó novo p/ armazenar o valor inteiro adicional.

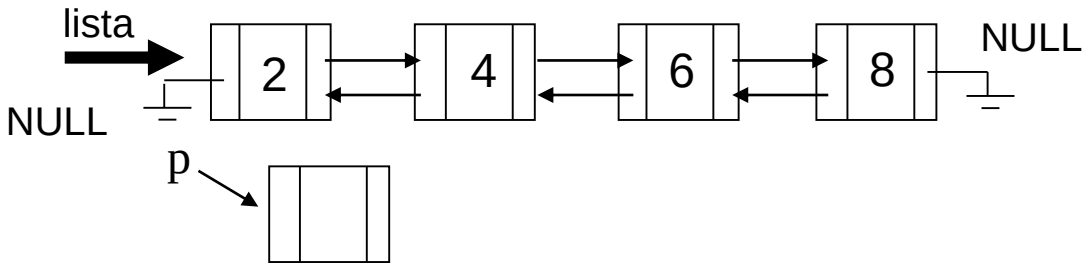
Para tanto é necessário um mecanismo para obter nós vazios novos a ser incluídos na lista. Admite-se a existência de um mecanismo para obter novos nós (criar) vazios.

A operação :
p = CriaNo

Obterá um novo nó vazio e definirá o conteúdo de uma variável chamada p com o endereço desse nó.

Dessa forma o valor de p é um ponteiro para esse nó recém-alocado.

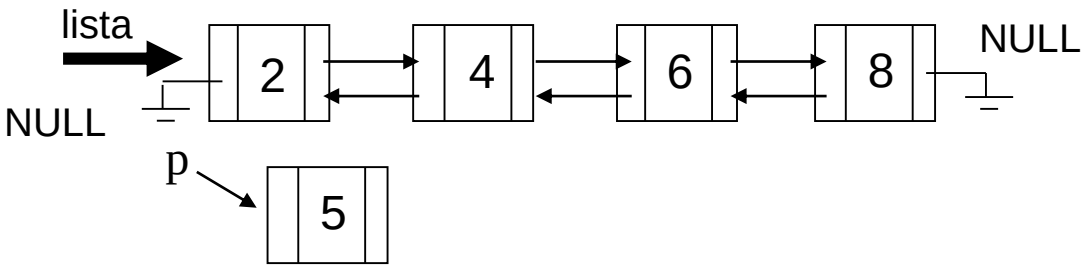
Depois de executar o primeiro passo, a nova configuração será:



2º passo:
Inserir o valor em questão na parte info do nó p.

Isso será feito através da operação:
p.info = 5

Depois de executar o segundo passo, a nova configuração será:

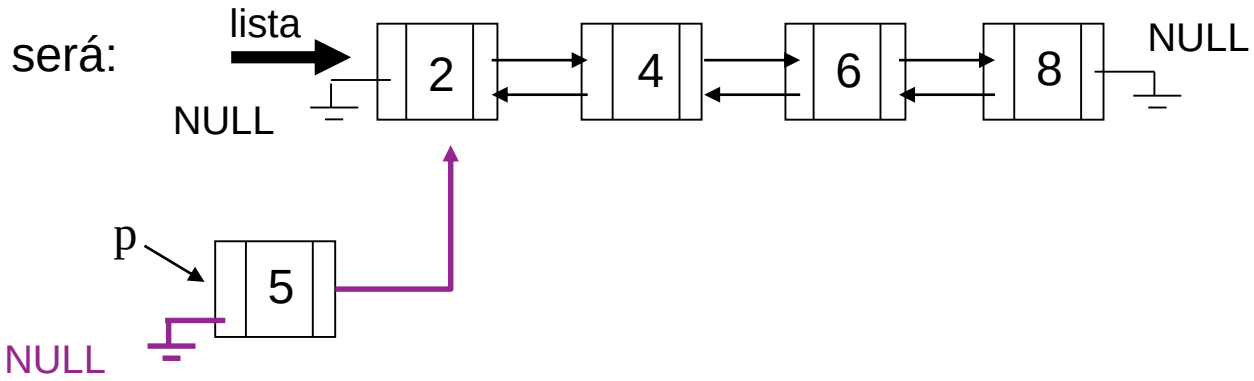


3º passo:
Fazer a parte **ant** e **prox** do nó p apontar para os locais adequados.

Como o nó p será inserido no início da lista a operação seguinte resolverá o problema:

p.ant = null
p.prox = lista

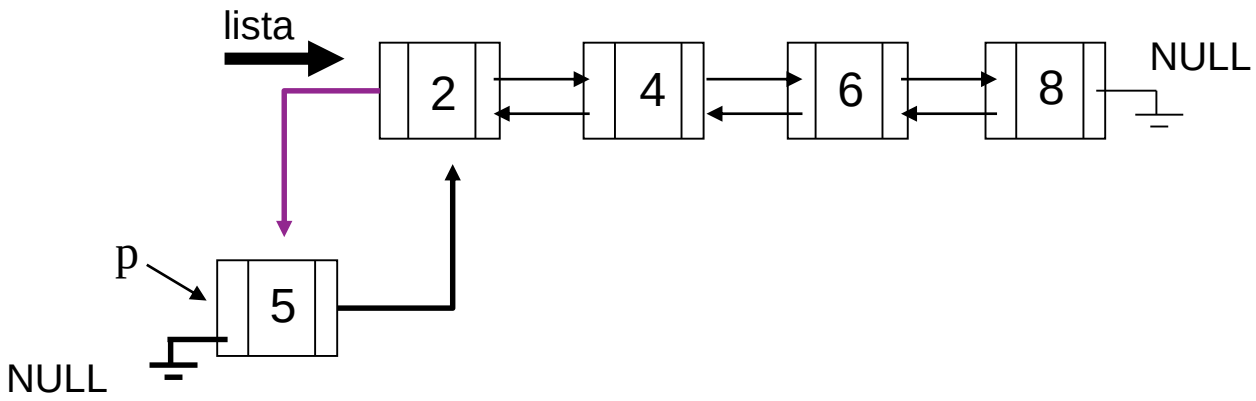
A próxima configuração será:



4º passo:
Fazer o ponteiro do primeiro nó, caso exista, apontar para o nó p. (ponteiro auxiliar para essa operação).

Caso não exista, nada precisa ser feito.

Dessa forma a nova configuração será:

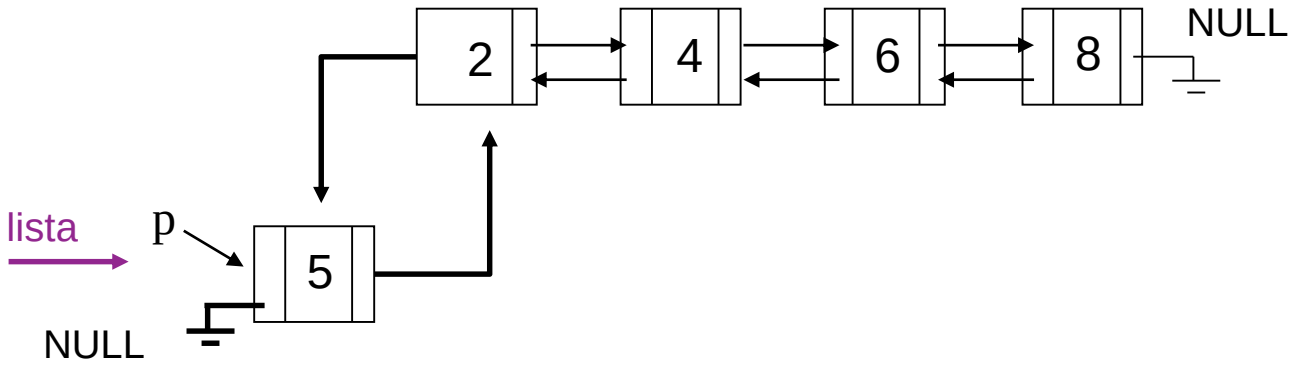


Neste momento o nó p ainda não pertence a lista pois, para pertencer a lista, um nó deve ser acessível pelo seu ponteiro externo, o que não se consegue fazer ainda com p.

5º passo:
Fazer o ponteiro externo apontar para o nó p.

lista = p;

Dessa forma a nova configuração será:

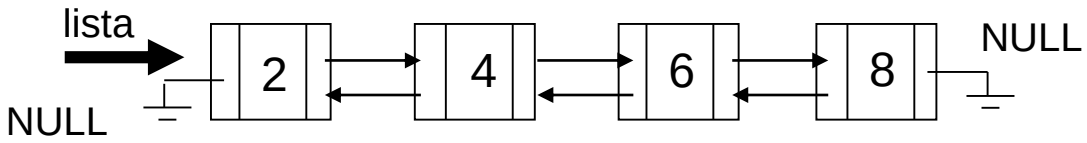


Operações na Lista – **NÃO ORDENADA**

INSERIR NO FIM

Inserindo nó na lista:

Supondo que temos a seguinte lista de inteiros:



E queremos incluir o elemento 9 na **ÚLTIMA** posição dessa lista (no início).

1º passo:
Obter um nó novo p/ armazenar o valor inteiro adicional.

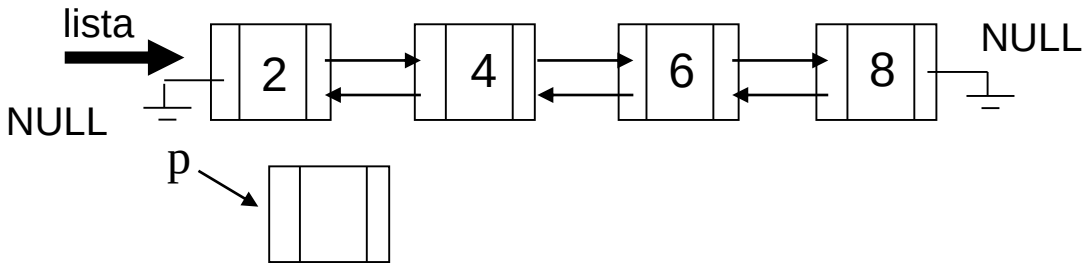
Para tanto é necessário um mecanismo para obter nós vazios novos a ser incluídos na lista. Admite-se a existência de um mecanismo para obter novos nós (criar) vazios.

A operação :
p = CriaNo

Obterá um novo nó vazio e definirá o conteúdo de uma variável chamada p com o endereço desse nó.

Dessa forma o valor de p é um ponteiro para esse nó recém-alocado.

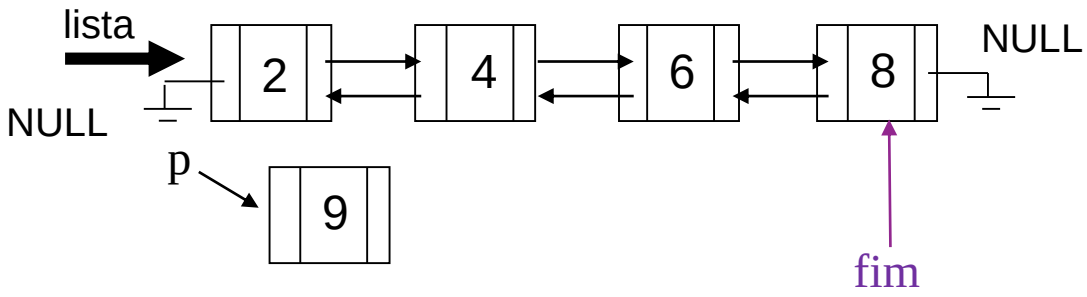
Depois de executar o primeiro passo, a nova configuração será:



2º passo:
Inserir o valor em questão na parte info do nó p.

Isso será feito através da operação:
p.info = 9

Depois de executar o segundo passo, a nova configuração será:



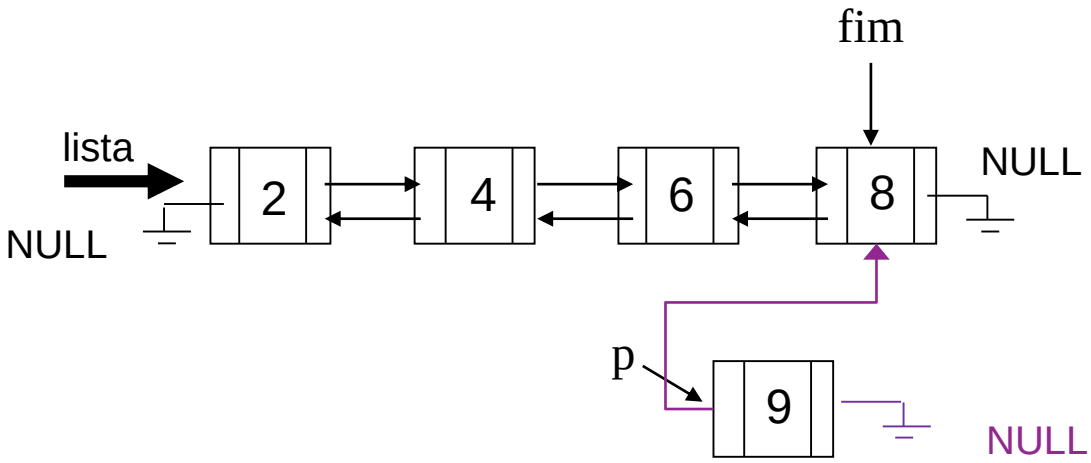
3º passo: encontrar o final da lista (ponteiro fim)

4º passo:
Fazer a parte **ant** e **prox** do nó p apontar para os locais adequados.

Como o nó p será inserido no final da lista a operação seguinte resolverá o problema:

p.ant = fim
p.prox = null

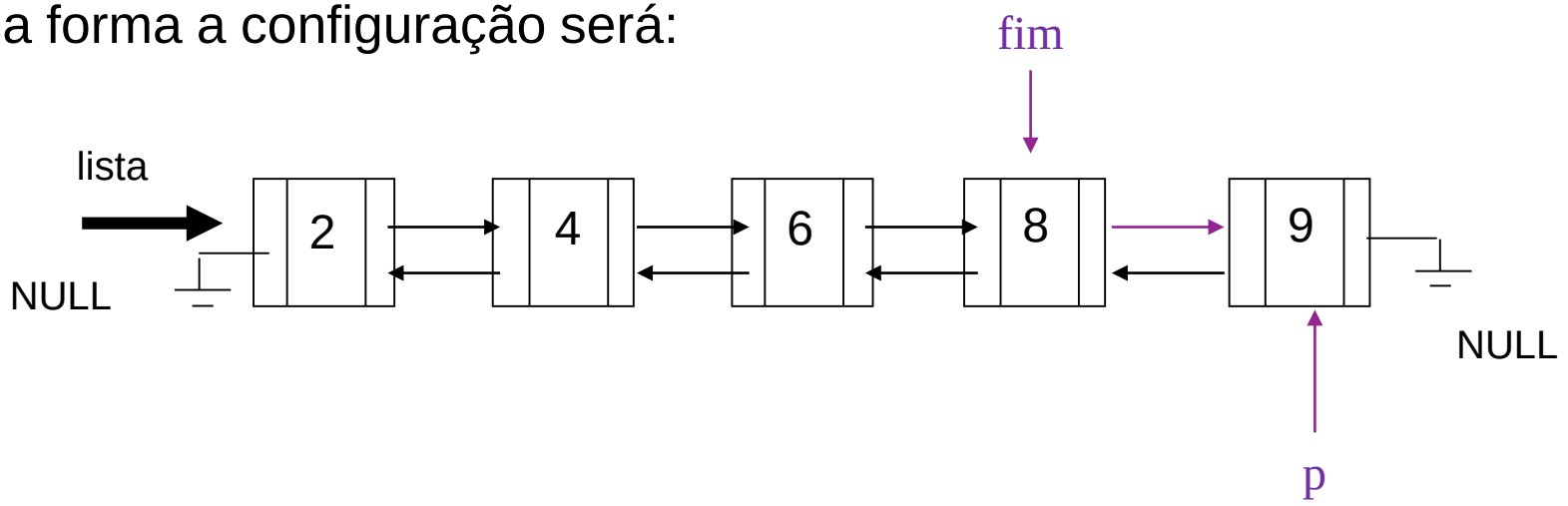
A próxima configuração será:



5º passo:
Fazer o ponteiro do último nó apontar para o nó p. (ponteiro auxiliar para essa operação - fim).

fim->prox = p

Dessa forma a configuração será:



Inserção Ordenada (1/4)

```
int inserirOrdenado2(DUPLA **l, int valor)
{
    int resultado = FALSE;
    DUPLA *novo;
    DUPLA *atual;

    novo = (DUPLA *) malloc(sizeof(DUPLA));

    if(novo != NULL)
    {
        novo->info = valor;
        novo->ant = NULL;
        novo->prox = NULL;

        if(*l == NULL)
        {
            *l = novo;
            resultado = TRUE;
        }
        else
        {
```

Se alocou memória corretamente

SE a lista está vazia

Inserção Ordenada (2/4)

```
else
{
    atual = *l;
    while((atual->prox != NULL) && (valor > atual->info))
    {
        atual = atual->prox;
    }

    if((*l == atual)&&(valor < atual->info))
    {
        if(inserirInicio(&(*l), valor) == TRUE)
        {
            resultado = TRUE;
        }
    }
    else
```

PROCURA O LOCAL A INSERIR

SE é a primeira posição...

Inserção Ordenada (3/4)

SE é a última posição...

```
else
{
    if((atual->prox == NULL)&&(valor > atual->info))
    {
        if(inserirFim(&(*l), valor) == TRUE)
        {
            resultado = TRUE;
        }
    }
    else
```

Inserção Ordenada (4/4)

RESTA uma posição no meio...

```
                else
                {
                    novo->prox = atual;
                    novo->ant = atual->ant;
                    (atual->ant)->prox = novo;
                    atual->ant = novo;
                    resultado = TRUE;
                }
            }
        }
    }
    return(resultado);
}
```

Inserção Ordenada – Outra Forma (1/2)

```
if(*l == NULL)
{
     *l = novo;
     resultado = TRUE;
}
else
{
     atual = *l;
     while((atual->prox != NULL) && (valor > atual->info))
     {
          anterior = atual;
          atual = atual->prox;
     }
     if((*l == atual)&&(valor < atual->info))
     {
          novo->prox = atual;
          atual->ant = novo;
          *l = novo;
          resultado = TRUE;
     }
     else
```

Lista vazia

Encontra posição para inserir

COM DOIS PONTEIROS...

Insere no início

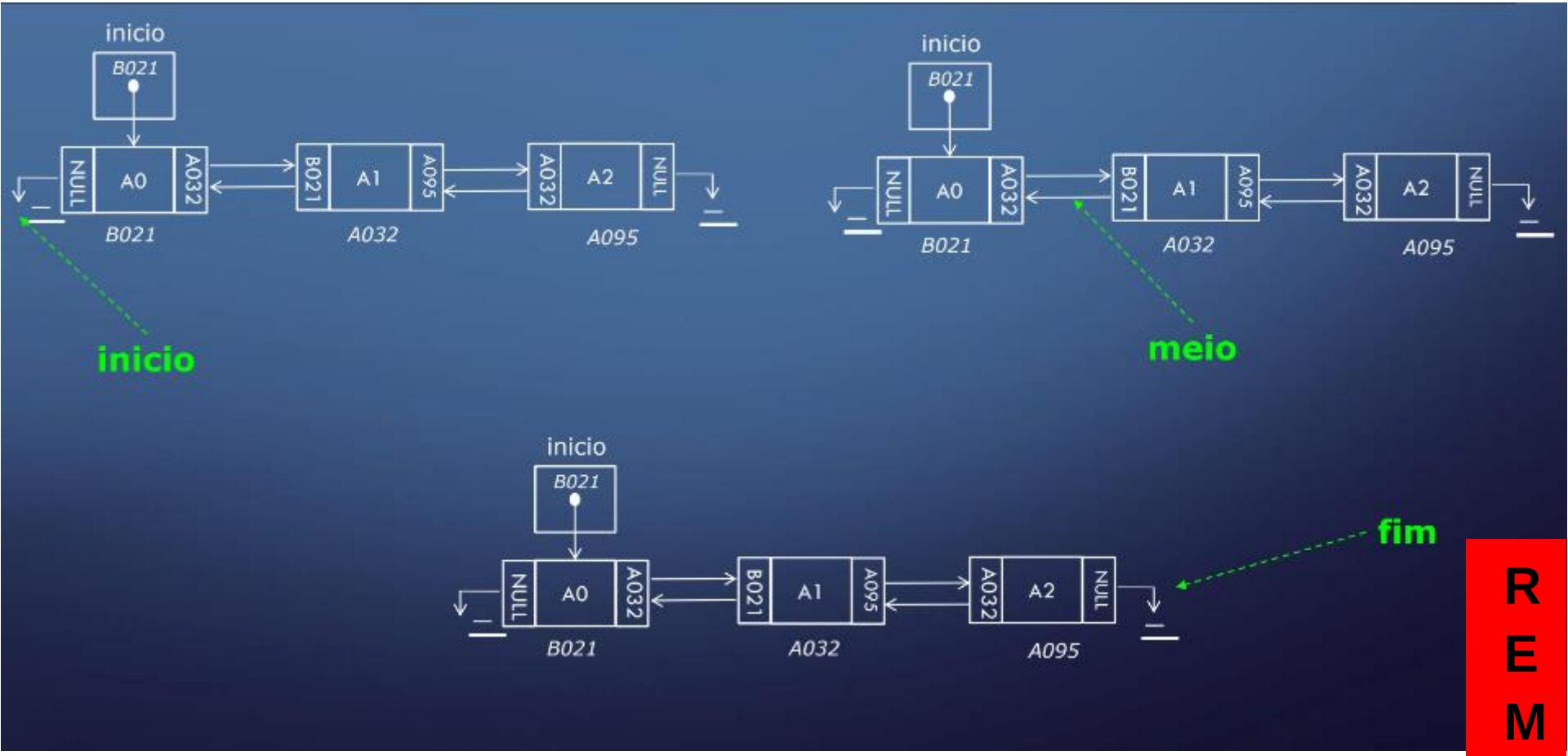
Exemplo sem chamar função

Inserção Ordenada – Outra Forma (2/2)

```
            else
            {
                novo->prox = atual;
                novo->ant = anterior;
                anterior->prox = novo;
                if(atual != NULL)
                {
                    atual->ant = novo;
                }
                resultado = TRUE;
                }
            }
        }
    }
    return(resultado);
}
```

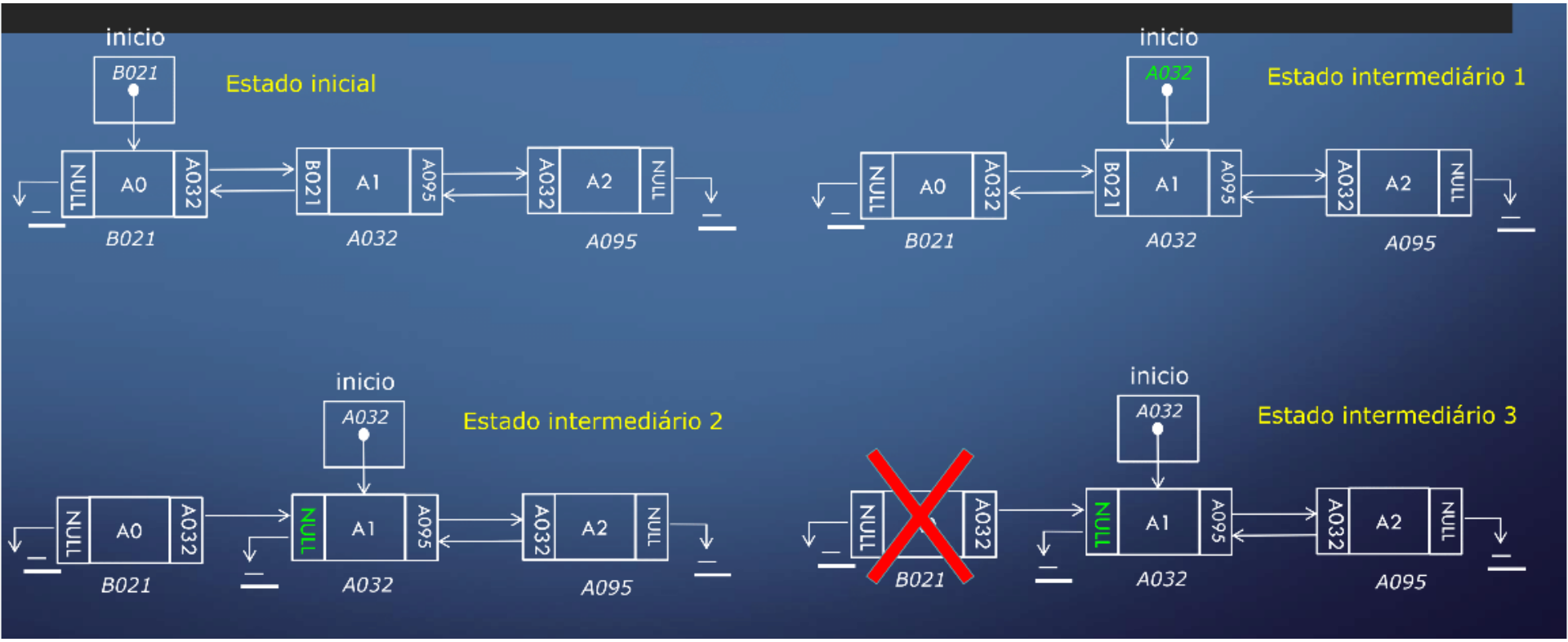
Insere em qualquer posição: meio ou fim

COM DOIS PONTEIROS...

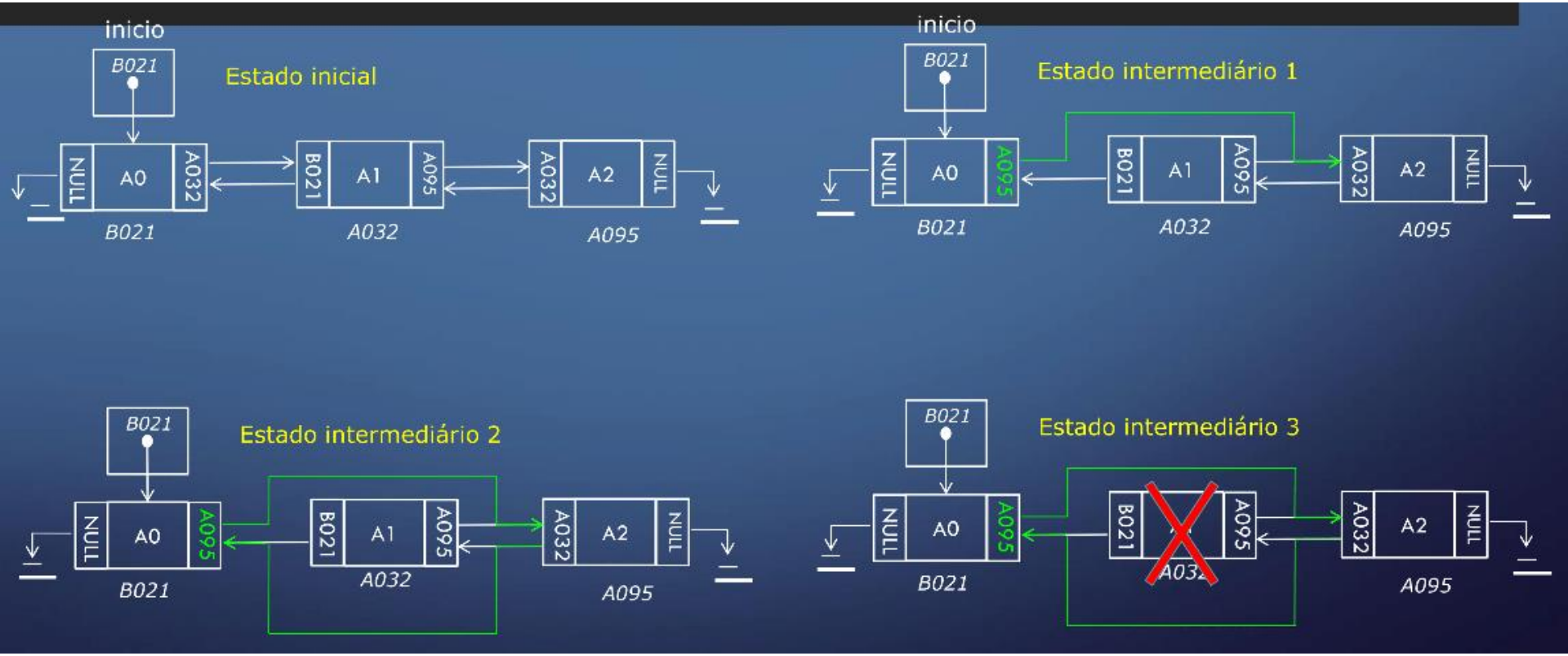

inicio
B021
NULL
A0
A032
B021
A1
A095
A032
A2
NULL
B021
A032
A095
inicio
inicio
B021
NULL
A0
A032
B021
A1
A095
A032
A2
NULL
B021
A032
A095
meio
inicio
B021
NULL
A0
A032
B021
A1
A095
A032
A2
NULL
B021
A032
A095
fim

REMOÇÃO

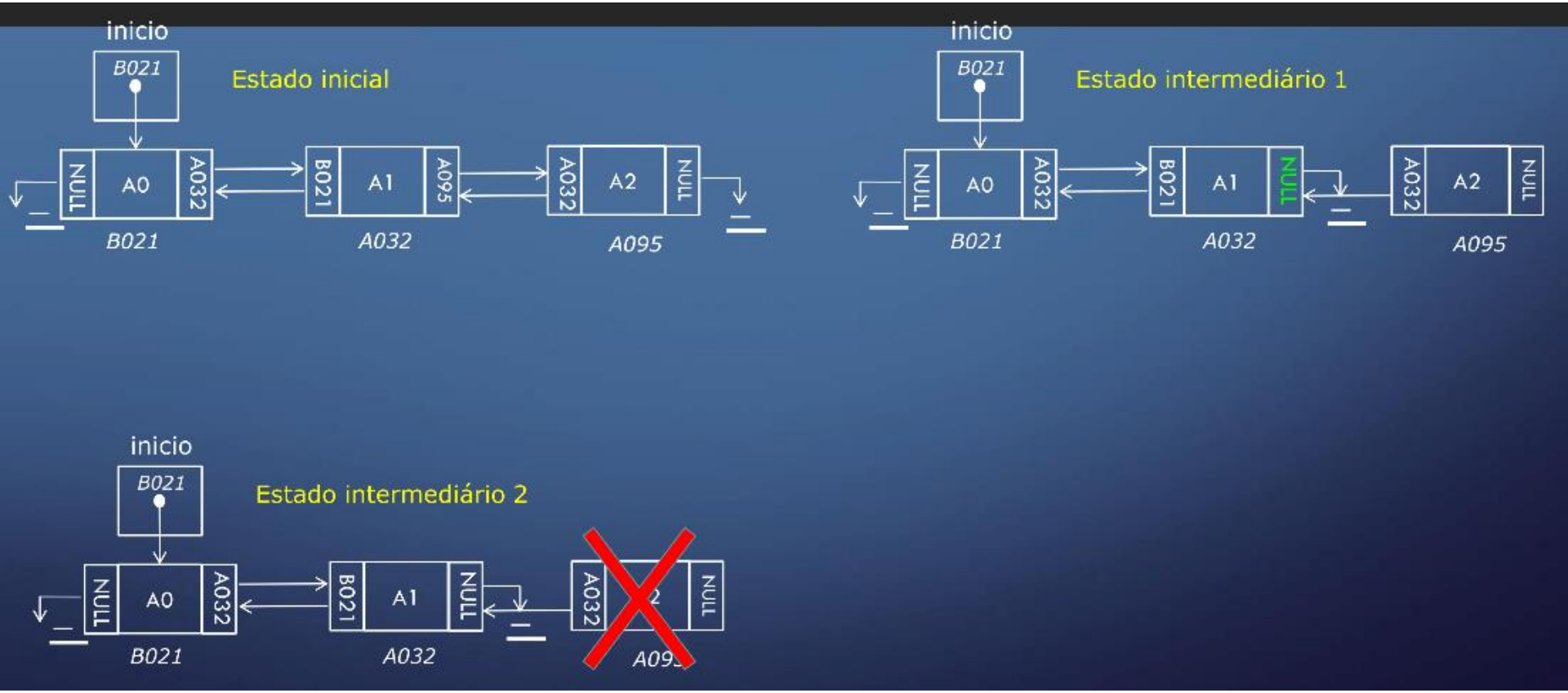
Remoção no Início



Remoção no Meio



Remoção no Fim



Agora é a sua vez!

- Considerando uma lista **duplamente** encadeada e não ordenada, faça funções para
  - Inserir uma informação no início da lista;
  - Inserir uma informação no final da lista;
  - Mostrar os valores da lista;
  - Consultar um valor da lista;
  - Remover um valor da lista;
  - Contar quantos nós tem a lista.
  - INSERÇÃO EM LISTA ORDENADA