



**INSTITUTO FEDERAL**  
Sul-rio-grandense

Câmpus  
Pelotas

EDUCAÇÃO  
**PÚBLICA**  
**100%**  
GRATUITA

# Estrutura de Dados

Aula 12

## Lista Duplamente Encadeadas

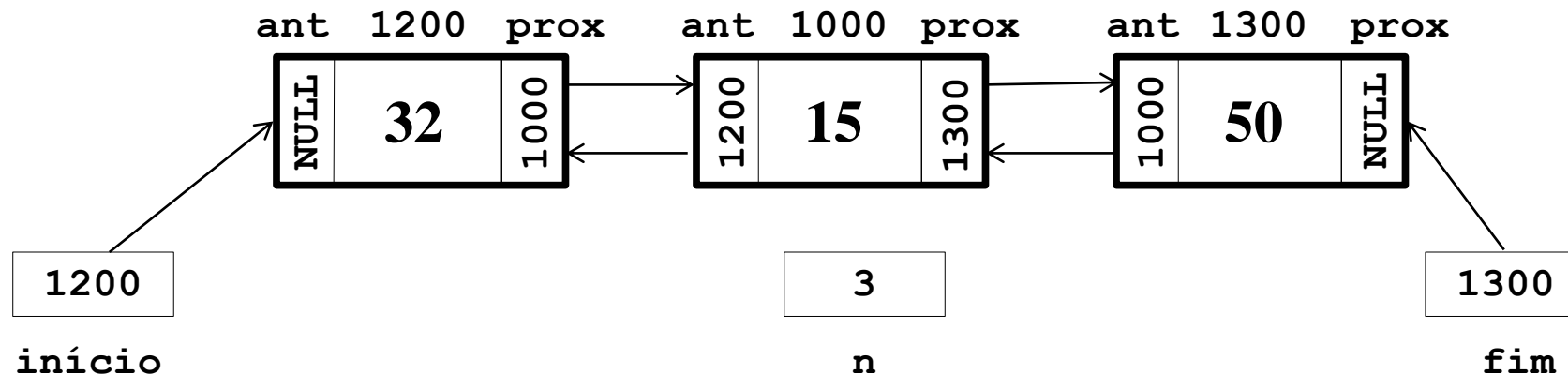
Alocação Dinâmica de Memória

**ListaDE**

# Lista Duplamente Encadeada

## Representação por duplo encadeamento

- Torna a inclusão e exclusão do último  $O(1)$
- Permite o percurso do fim para o início.
- O valor contido em um campo **prox** é o endereço do próximo nodo.
- O valor contido em um campo **ant** é o endereço do nodo anterior.
- O valor em **início** representa o endereço do primeiro nodo da lista (**NULL** se a lista está vazia).
- O valor em **fim** representa o endereço do último nodo da lista (**NULL** se a lista está vazia).
- O valor contido em **n** representa a quantidade de nodos armazenados na lista.



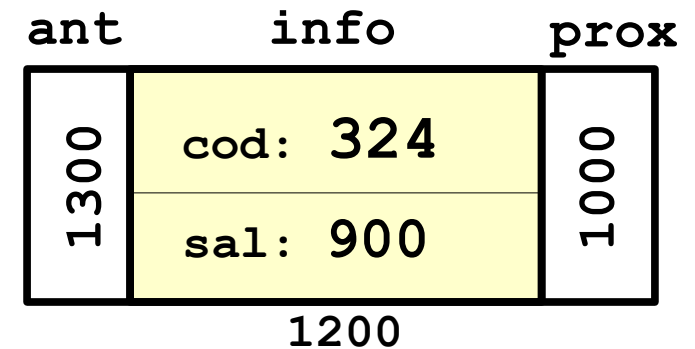
# Lista Duplamente Encadeada

```
typedef struct {  
    int cod;  
    float sal;  
} Dado;
```

```
typedef struct nodo Nodo;
```

```
struct nodo {  
    Dado info; /* Informação armazenada */  
    Nodo *ant; /* Endereço do anterior */  
    Nodo *prox; /* Endereço do próximo */  
};
```

```
typedef struct {  
    Nodo *inicio;  
    Nodo *fim;  
    int n;  
} ListaDE;
```



# Lista Duplamente Encadeada

```
...
int main() {
    ListaDE lista;

    criaLista(&lista);
    ...
}
```

```
void criaLista(ListaDE *lt) {
    lt->inicio = NULL;
    lt->fim = NULL;
    lt->n = 0;
}
```

lista      Lista vazia

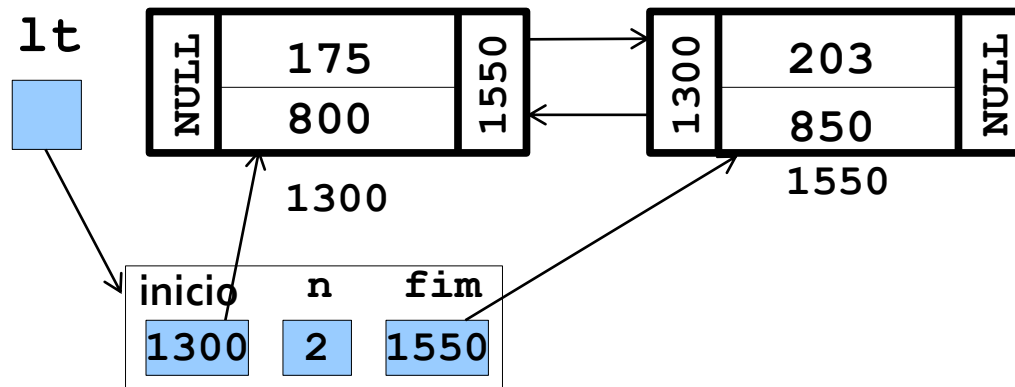
inicio	n	fim
NULL	0	NULL

# ListaDE - incluiNoInicio

```
int incluiNoInicio(ListaDE *lt, Dado d) {  
    Nodo *pNodo;  
  
    ...  
}
```

**pNodo**

???



**d**

cod: 123
sal: 900

**pNodo**

???

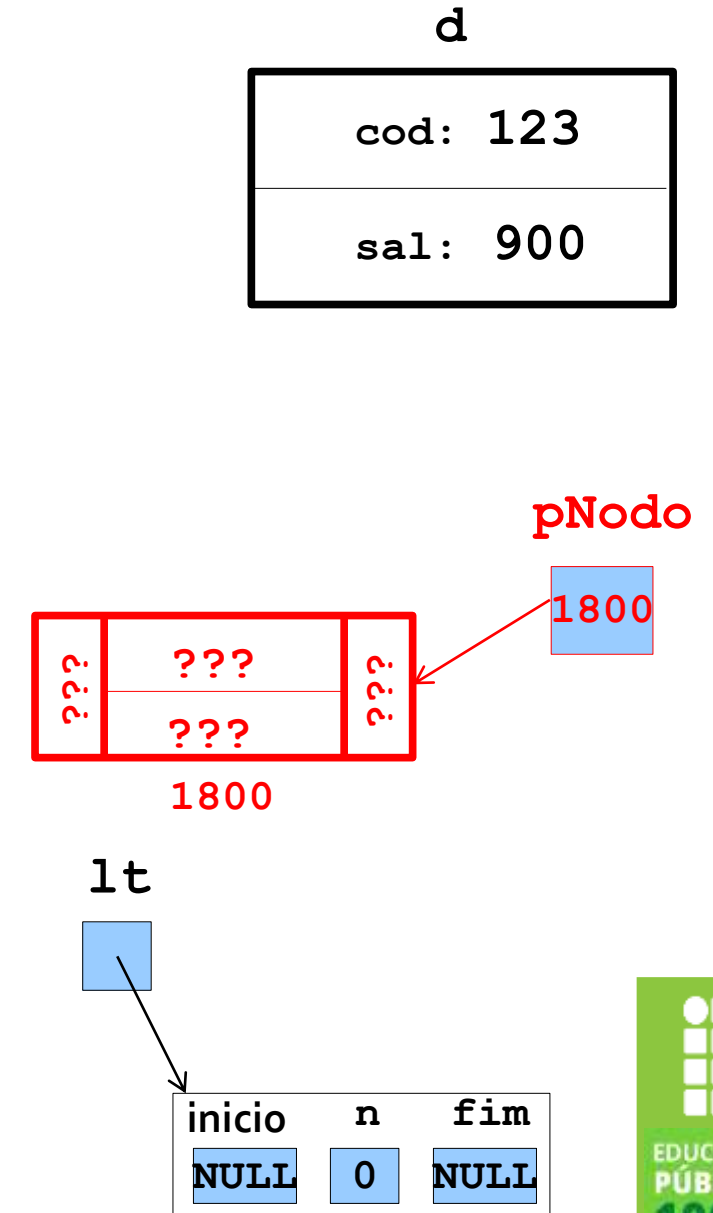
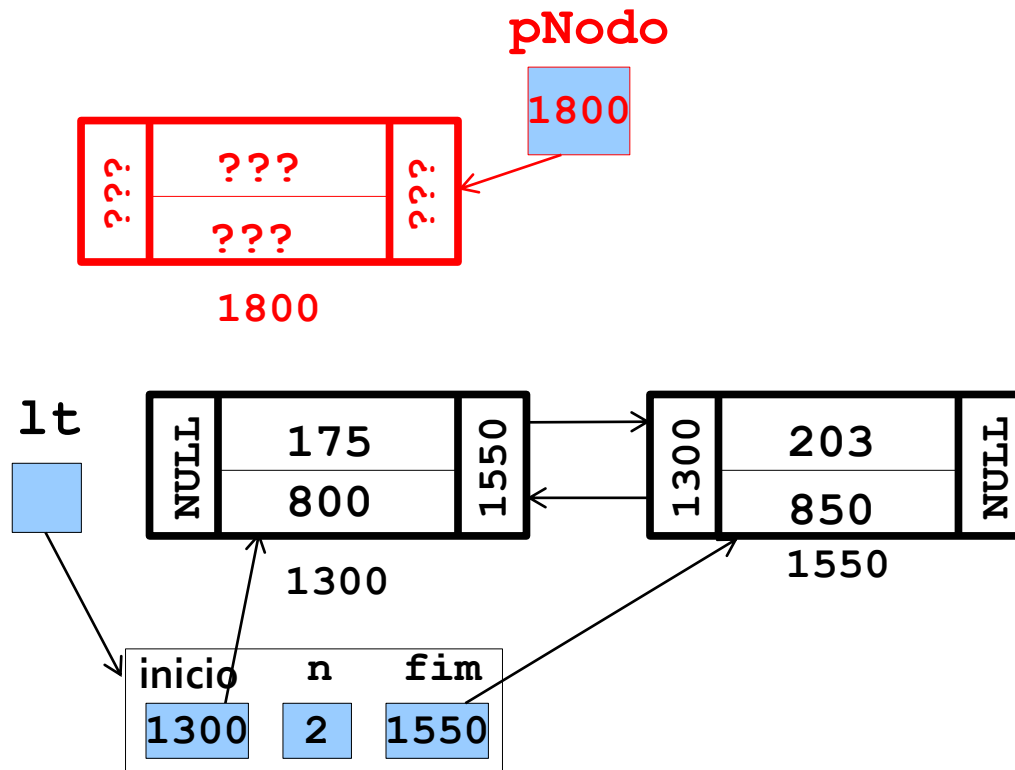
**lt**



inicio	n	fim
NULL	0	NULL

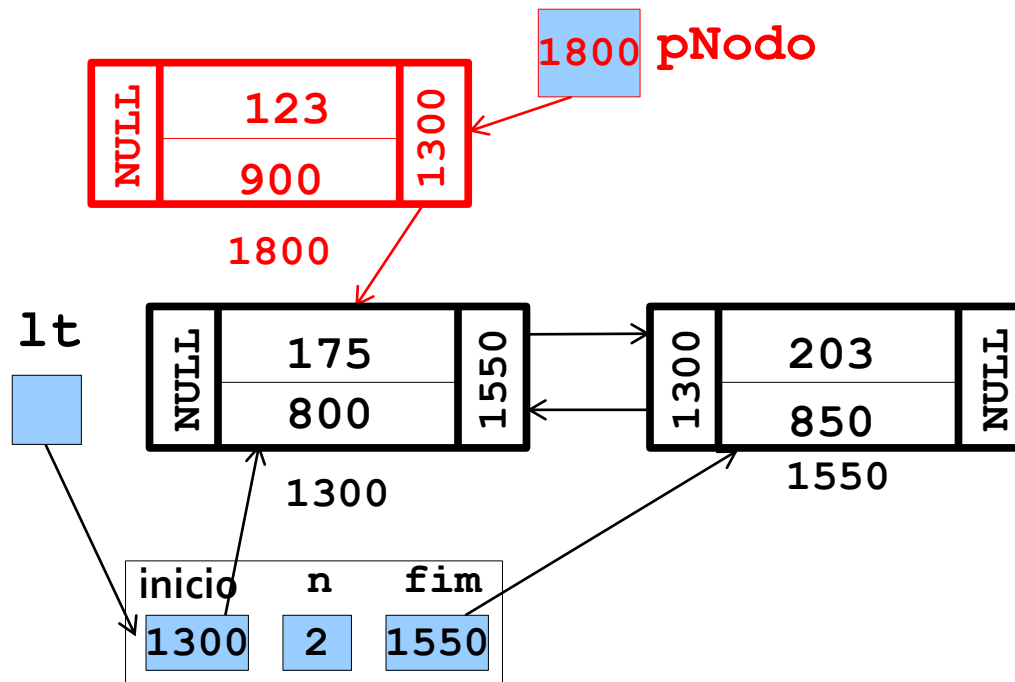
# ListaDE - incluiNoInicio

```
int incluiNoInicio(ListaDE *lt, Dado d) {  
    Nodo *pNodo;  
  
    pNodo = (Nodo *) malloc (sizeof (Nodo));  
    ...  
}
```



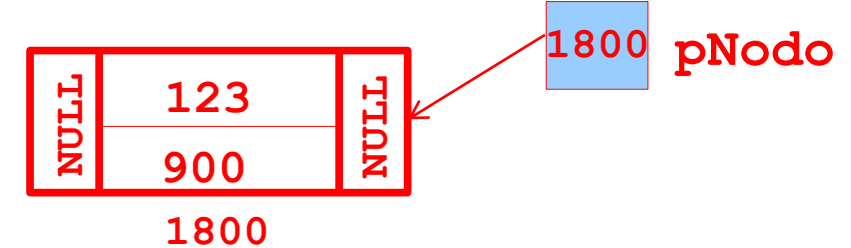
# ListaDE - incluiNoInicio

```
int incluiNoInicio(ListaDE *lt, Dado d) {  
    Nodo *pNodo;  
    pNodo = (Nodo *) malloc (sizeof (Nodo));  
    if (pNodo==NULL)  
        return FALTOU_MEMORIA;  
    else {  
        pNodo->info = d;  
        pNodo->ant = NULL;  
        pNodo->prox = lt->inicio;  
        ...  
    }  
}
```



d

cod: 123
sal: 900



lt



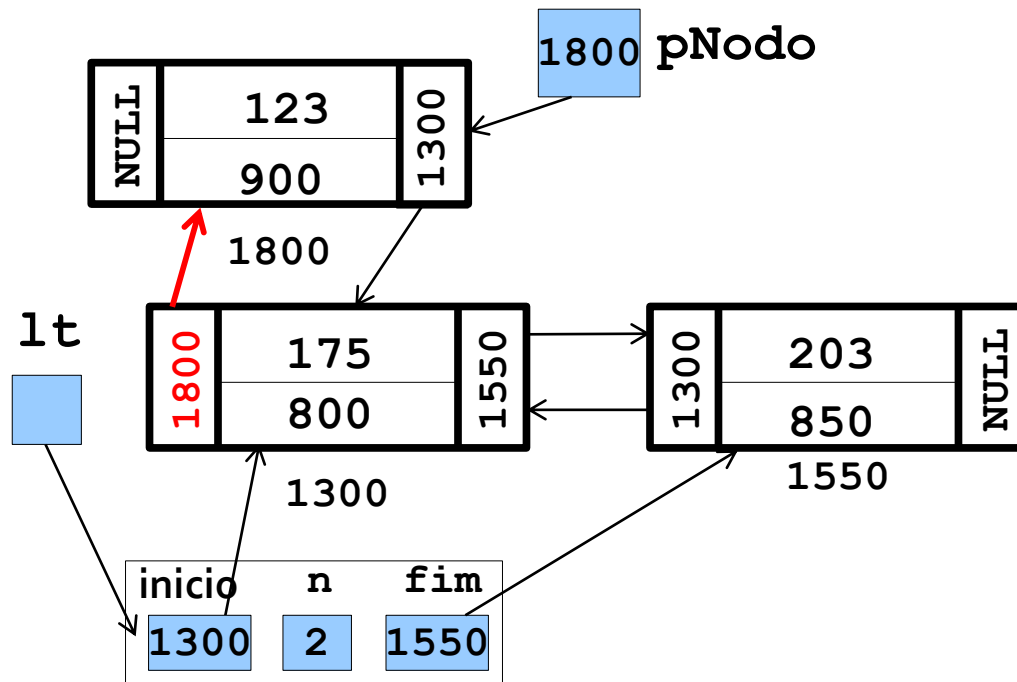
inicio	n	fim
NULL	0	NULL

```

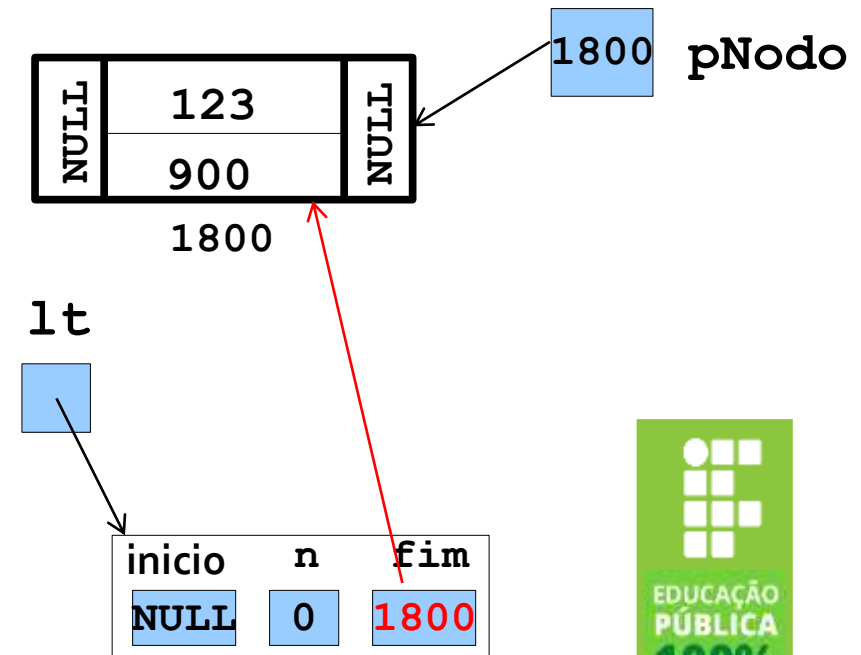
int incluiNoInicio(ListaDE *lt, Dado d) {
    Nodo *pNodo;

    pNodo = (Nodo *) malloc (sizeof (Nodo));
    if (pNodo==NULL)
        return FALTOU_MEMORIA;
    else {
        pNodo->info=d; pNodo->ant=NULL; pNodo->prox=lt->inicio;
        if (lt->n == 0)
            lt->fim = pNodo;
        else
            lt->inicio->ant = pNodo;
        ...
    }
}

```



## ListaDE - incluiNoInicio



```
int incluiNoInicio(ListaDE *lt, Dado d) {
```

```
    Nodo *pNodo;
```

```
    pNodo = (Nodo *) malloc (sizeof (Nodo));
```

```
    if (pNodo==NULL)
```

```
        return FALTOU_MEMORIA;
```

```
    else {
```

```
        pNodo->info=d; pNodo->ant=NULL; pNodo->prox=lt->inicio;
```

```
        if (lt->n == 0)
```

```
            lt->fim = pNodo;
```

```
        else
```

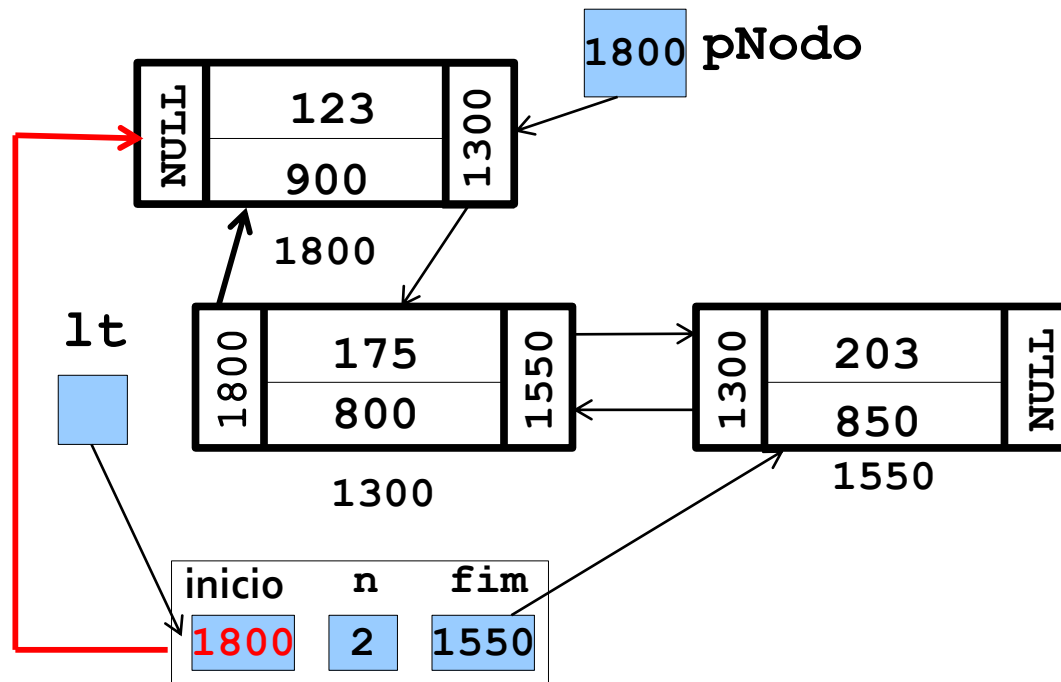
```
            lt->inicio->ant = pNodo;
```

```
        lt->inicio = pNodo;
```

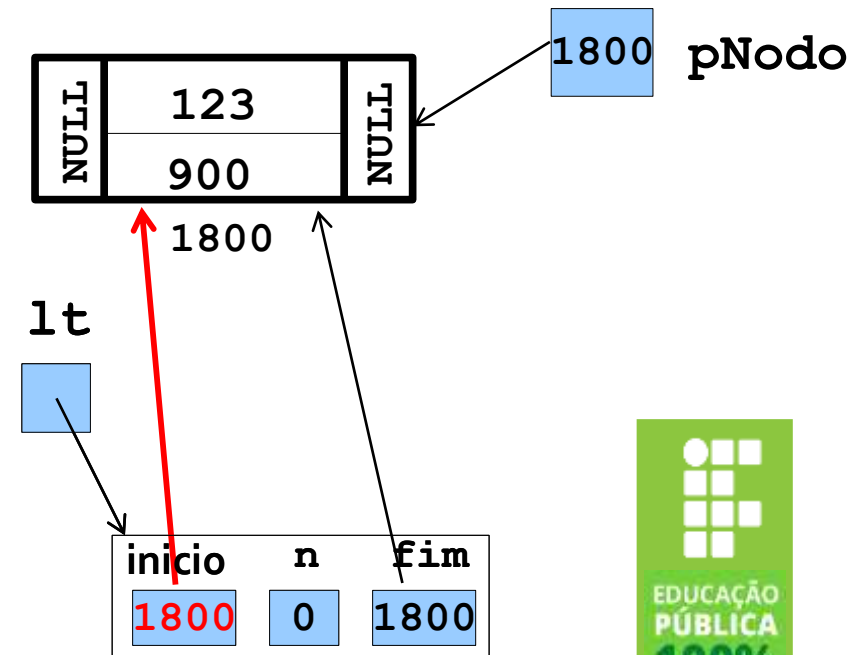
```
        ...
```

```
    }
```

```
}
```



## ListaDE - incluiNoInicio

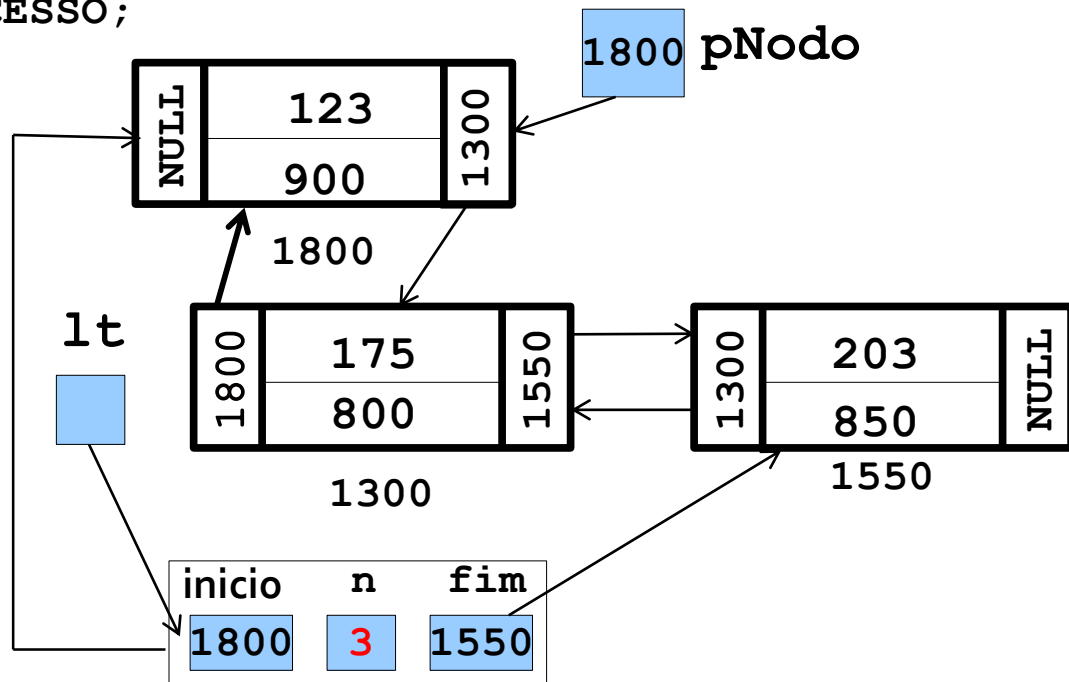


```

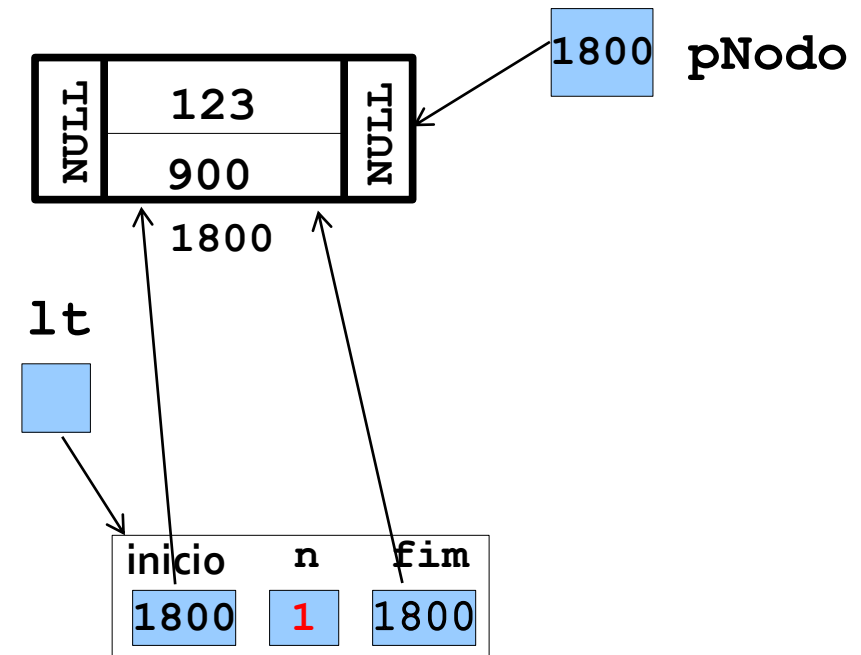
int incluiNoInicio(ListaDE *lt, Dado d) {
    Nodo *pNodo;

    pNodo = (Nodo *) malloc (sizeof (Nodo));
    if (pNodo==NULL)
        return FALTOU_MEMORIA;
    else {
        pNodo->info=d; pNodo->ant=NULL; pNodo->prox=lt->inicio;
        if (lt->n == 0)
            lt->fim = pNodo;
        else
            lt->inicio->ant = pNodo;
        lt->inicio = pNodo;
        lt->n++;
        return SUCESSO;
    }
}

```

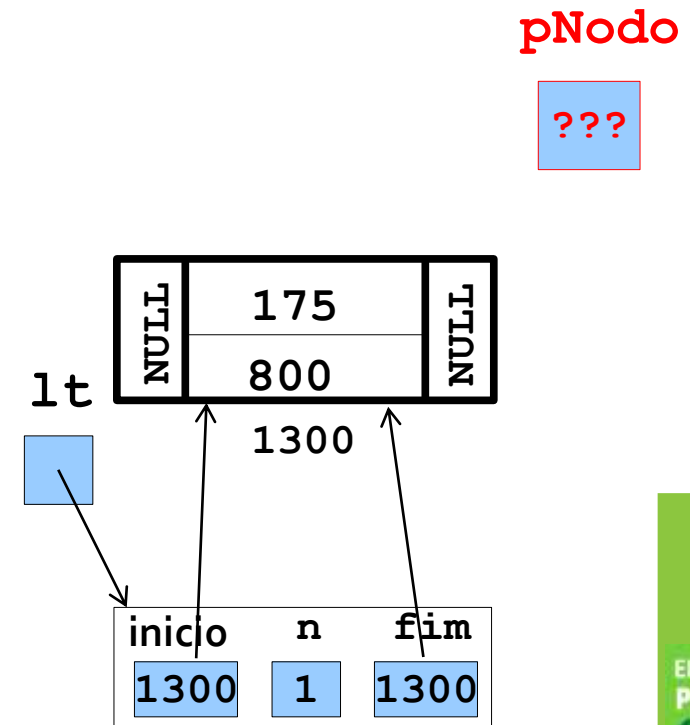
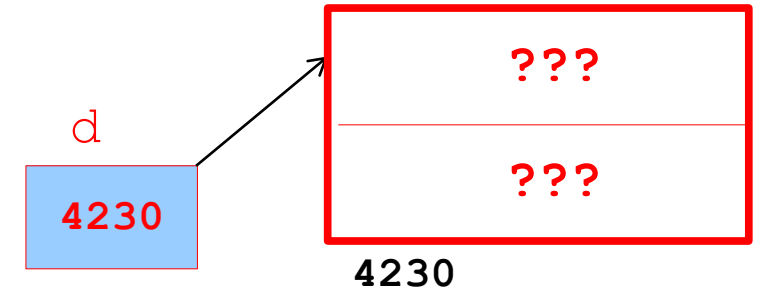
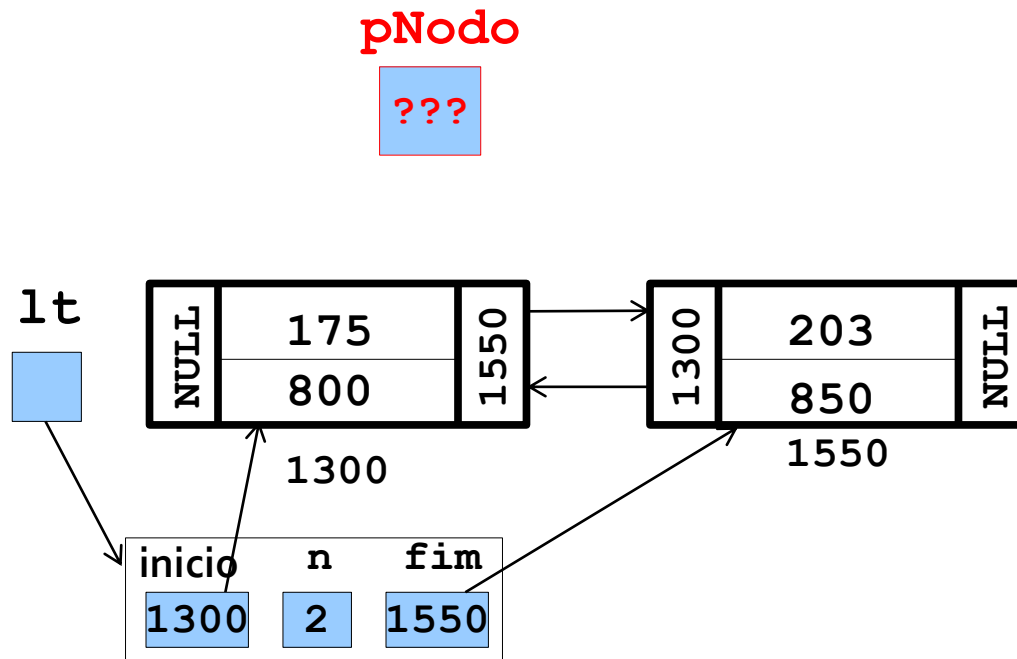


## ListaDE - incluiNoInicio



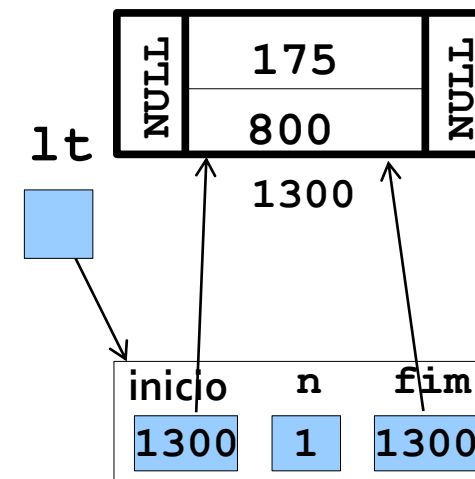
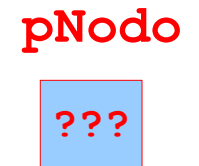
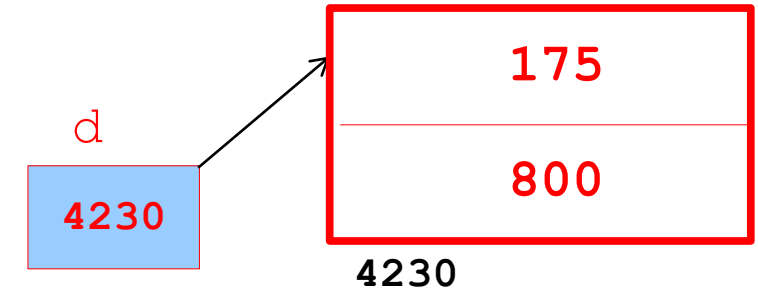
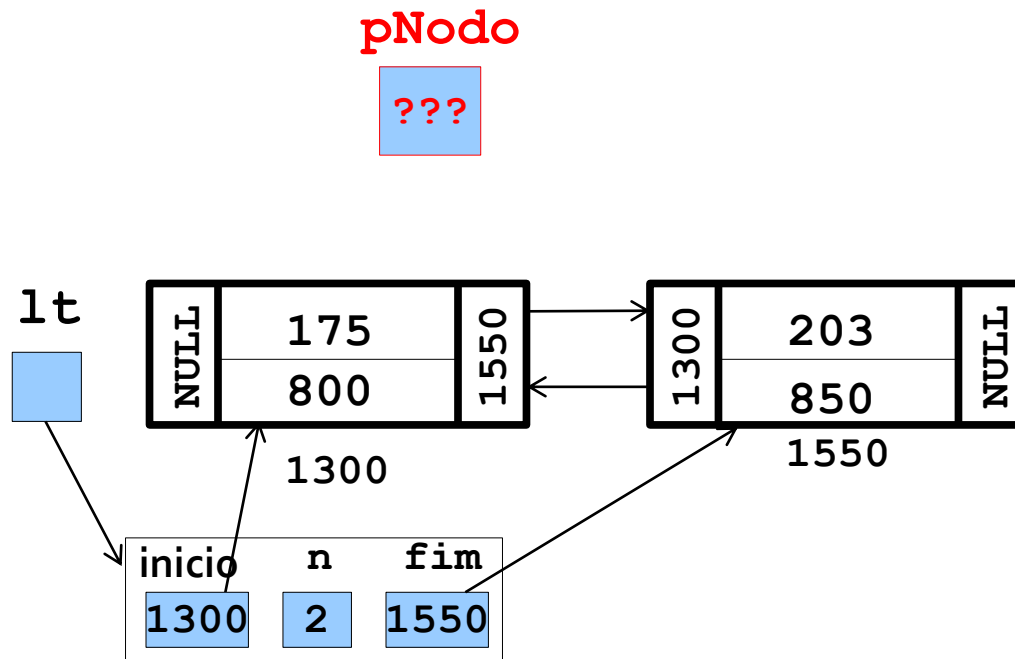
# ListaDE - excluiDoInicio

```
int excluiDoInicio(ListaDE *lt, Dado *d) {  
    Nodo *pNodo;  
  
    ...  
}
```



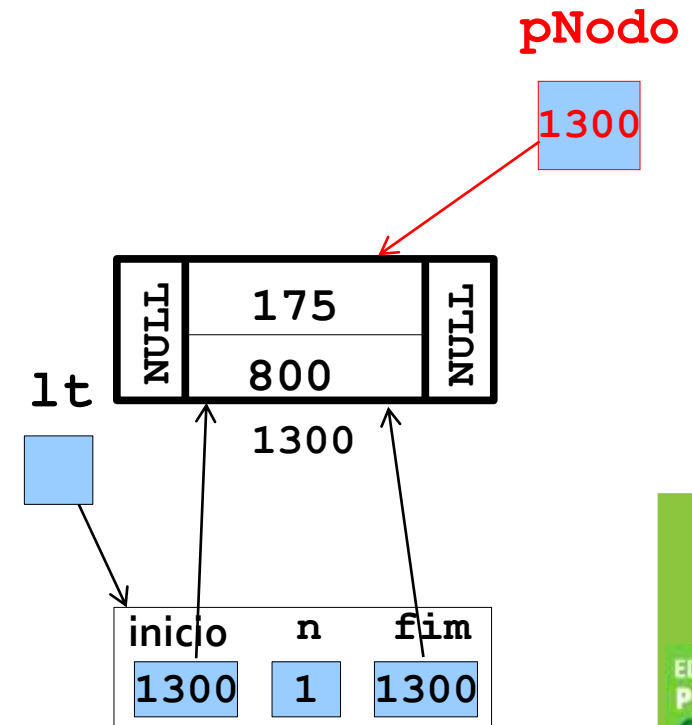
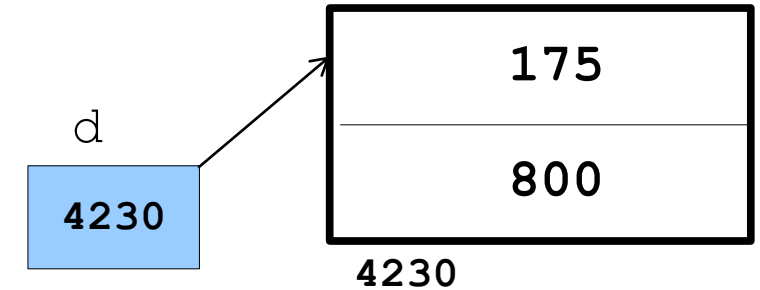
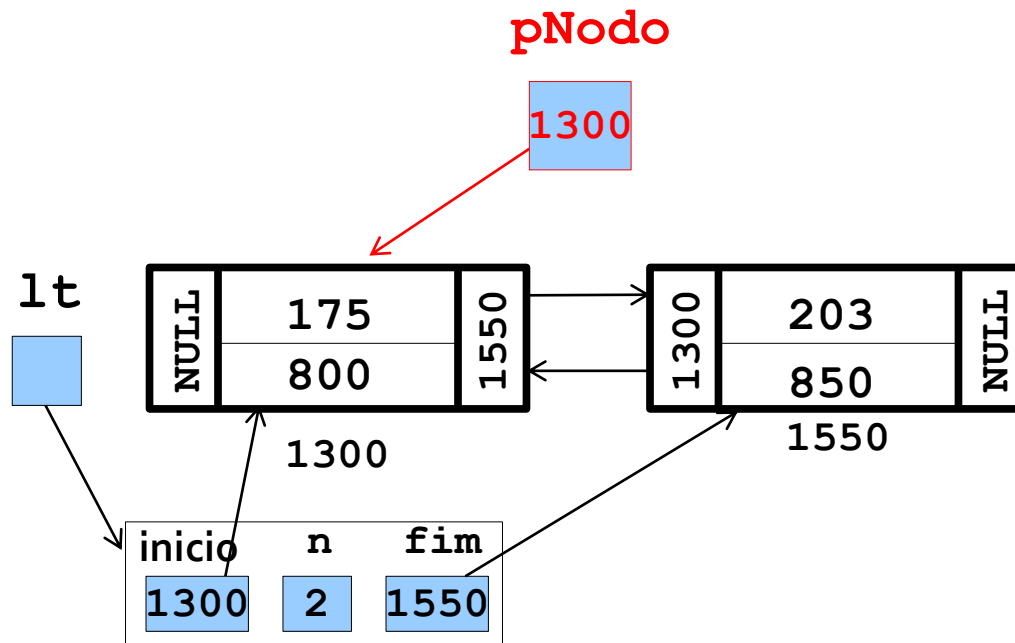
# ListaDE - excluiDoInicio

```
int excluiDoInicio(ListaDE *lt, Dado *d) {
    Nodo *pNodo;
    if (lt->n==0)
        return LISTA_VAZIA;
    else {
        *d = lt->inicio->info;
        ...
    }
}
```



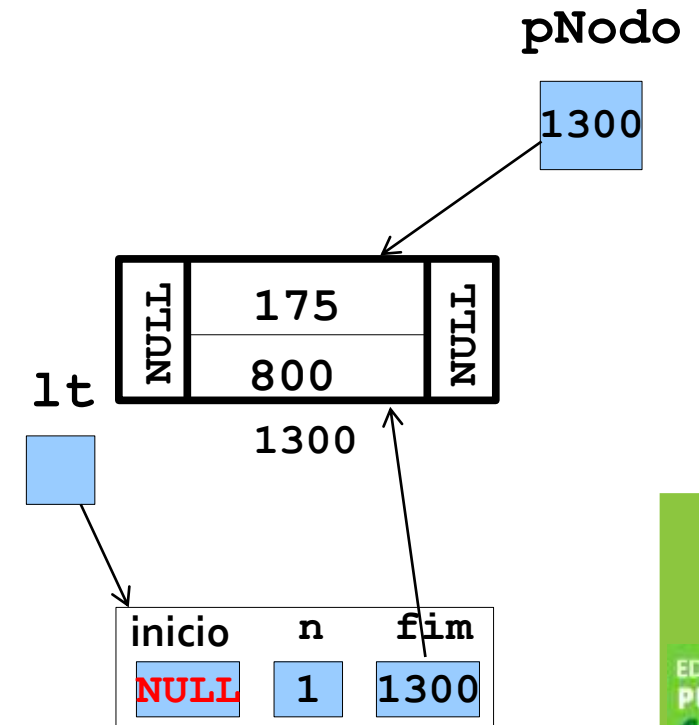
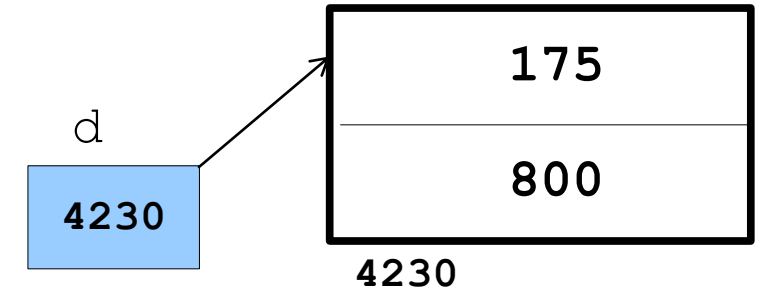
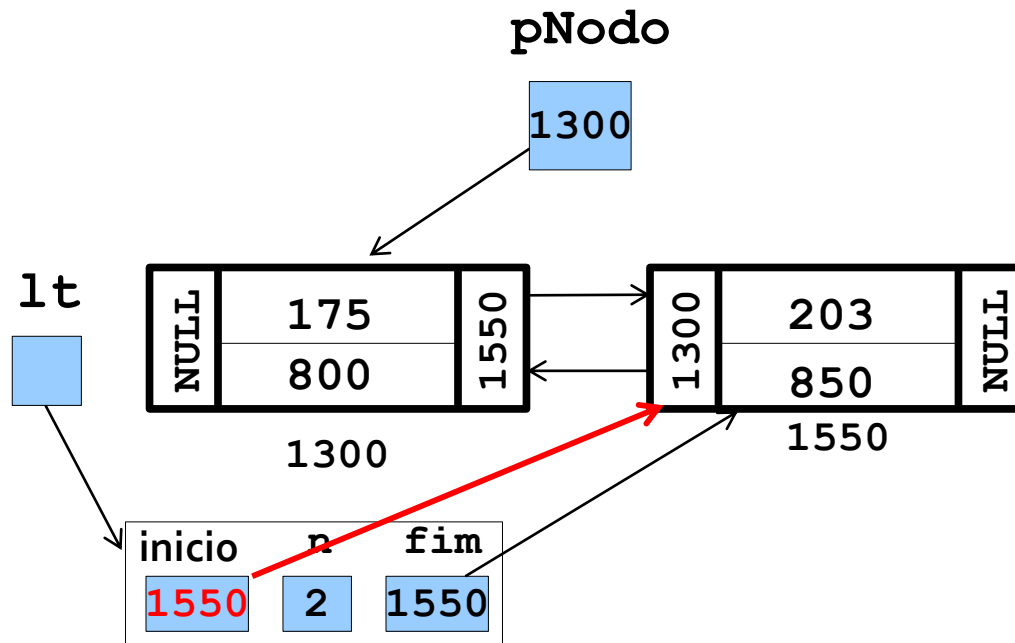
# ListaDE - excluiDoInicio

```
int  excluiDoInicio(ListaDE *lt, Dado *d) {
    Nodo *pNodo;
    if (lt->n==0)
        return LISTA_VAZIA;
    else {
        *d = lt->inicio->info;
        pNodo = lt->inicio;
        ...
    }
}
```



# ListaDE - excluiDoInicio

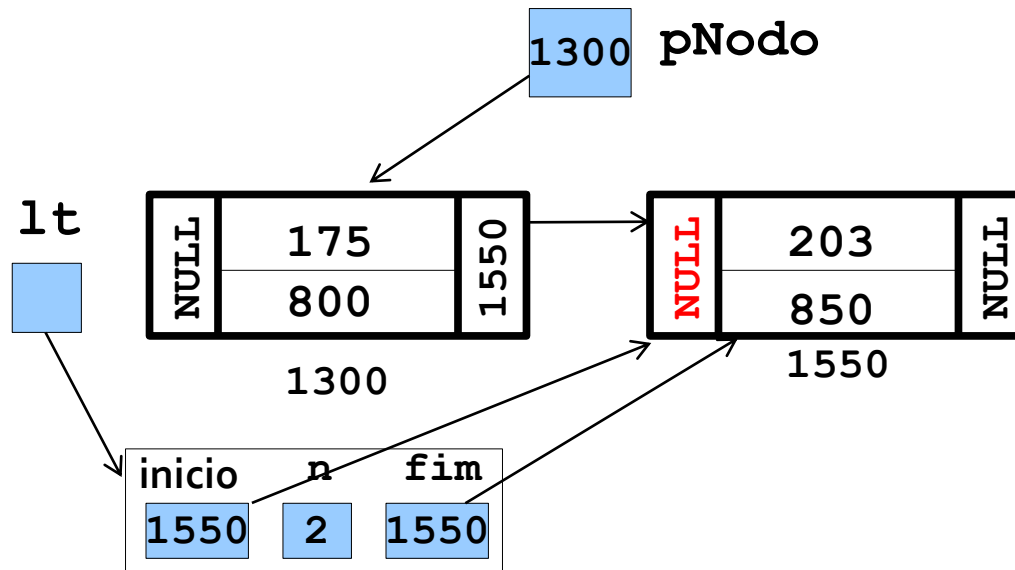
```
int excluiDoInicio(ListaDE *lt, Dado *d) {
    Nodo *pNodo;
    if (lt->n==0)
        return LISTA_VAZIA;
    else {
        *d = lt->inicio->info;
        pNodo = lt->inicio;
        lt->inicio = lt->inicio->prox;
        ...
    }
}
```



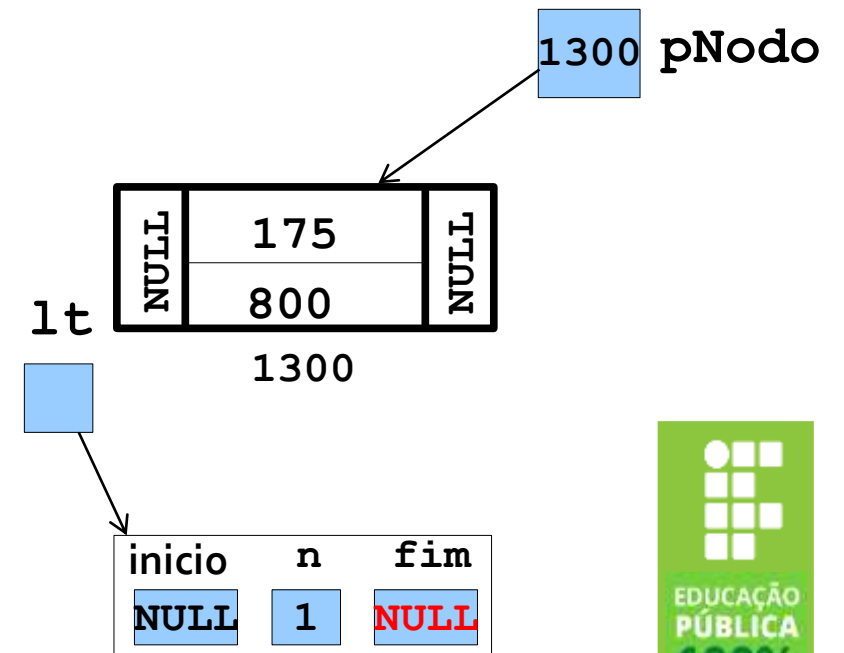
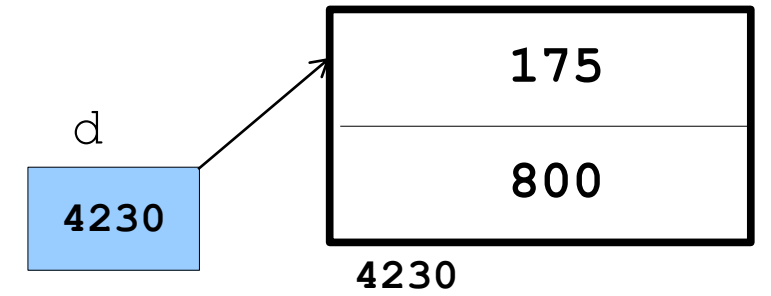
```

int  excluiDoInicio(ListaDE *lt, Dado *d) {
    Nodo *pNodo;
    if (lt->n==0)
        return LISTA_VAZIA;
    else {
        *d = lt->inicio->info;
        pNodo = lt->inicio;
        lt->inicio = lt->inicio->prox;
        if (lt->n == 1)
            lt->fim = NULL;
        else
            lt->inicio->ant = NULL;
        ...
    }
}

```



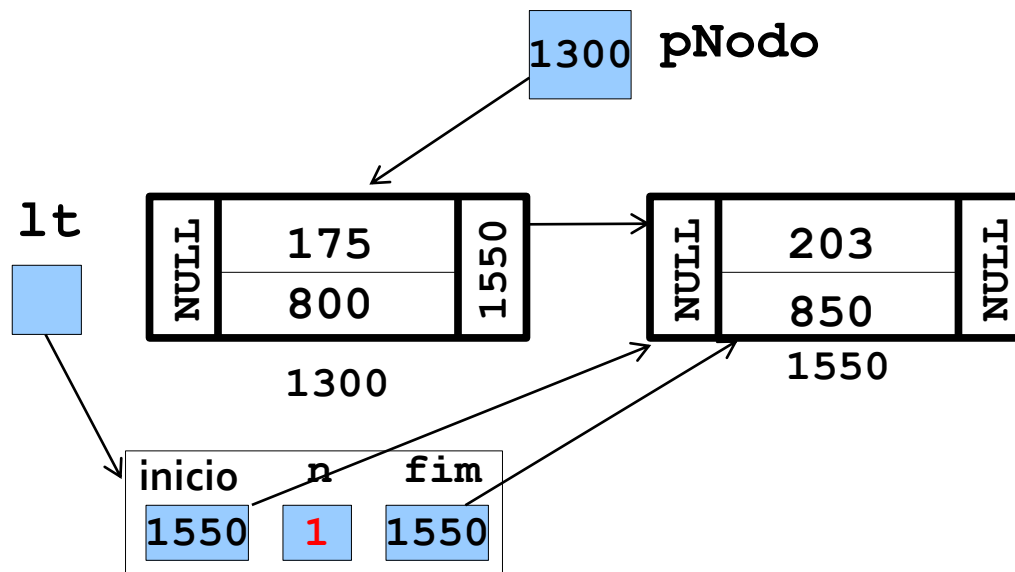
## ListaDE - excluiDoInicio



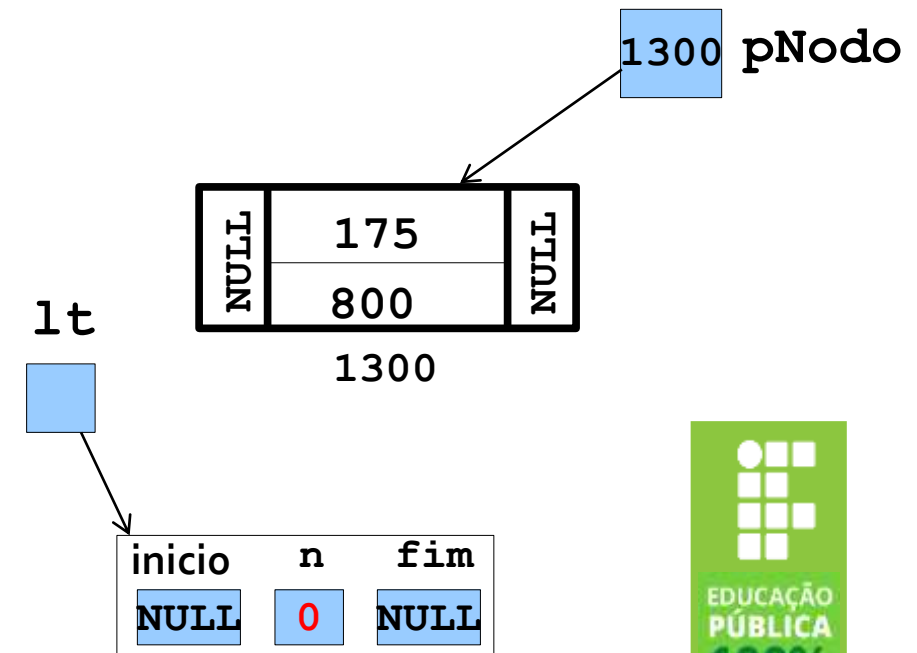
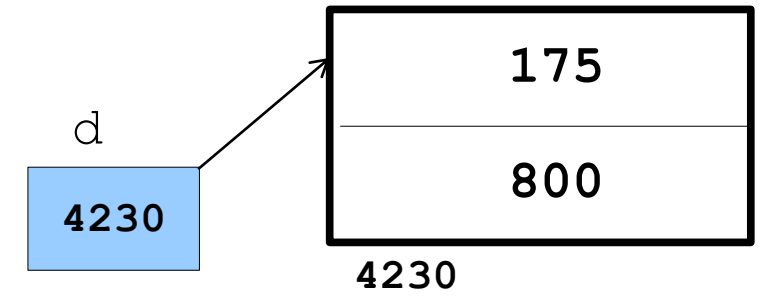
```

int  excluiDoInicio(ListaDE *lt, Dado *d) {
    Nodo *pNodo;
    if (lt->n==0)
        return LISTA_VAZIA;
    else {
        *d = lt->inicio->info;
        pNodo = lt->inicio;
        lt->inicio = lt->inicio->prox;
        if (lt->n == 1)
            lt->fim = NULL;
        else
            lt->inicio->ant = NULL;
        lt->n--;
        ...
    }
}

```



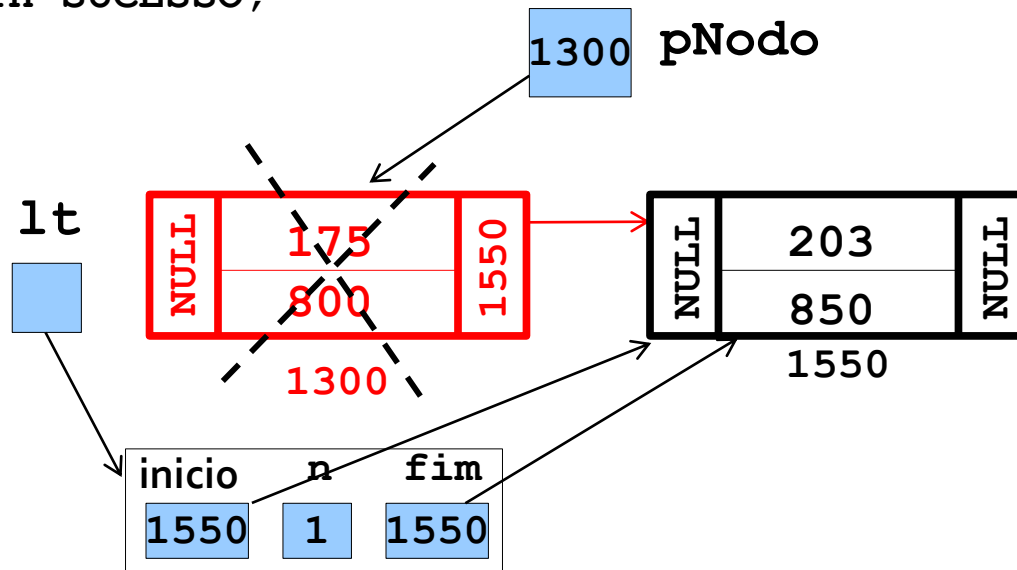
## ListaDE - excluiDoInicio



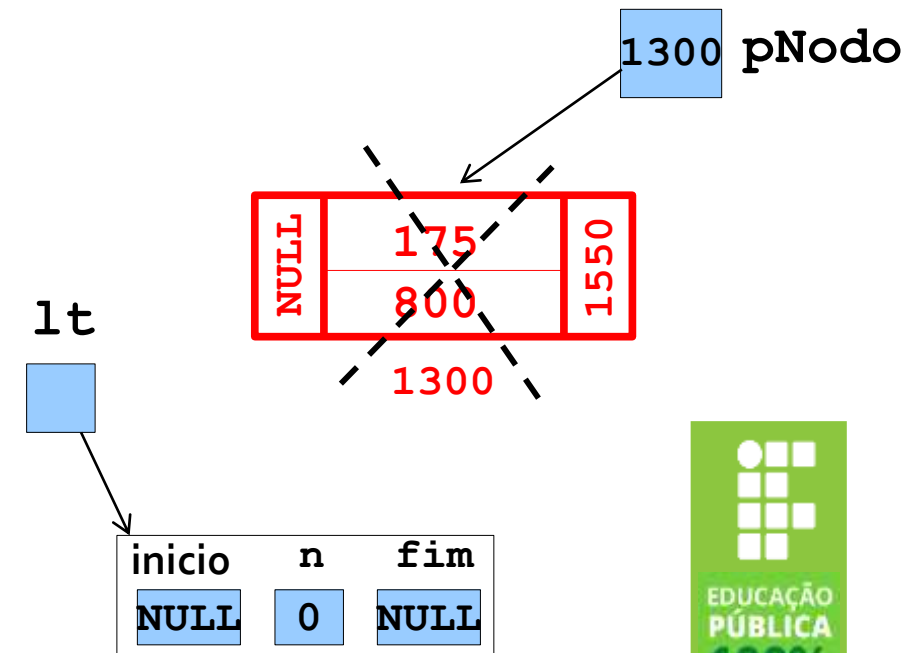
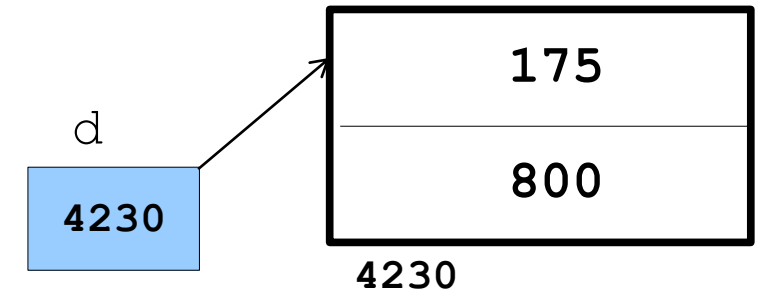
```

int  excluiDoInicio(ListaDE *lt, Dado *d) {
    Nodo *pNodo;
    if (lt->n==0)
        return LISTA_VAZIA;
    else {
        *d = lt->inicio->info;
        pNodo = lt->inicio;
        lt->inicio = lt->inicio->prox;
        if (lt->n == 1)
            lt->fim = NULL;
        else
            lt->inicio->ant = NULL;
        lt->n--;
        free (pNodo);
        return SUCESSO;
    }
}

```



## ListaDE - excluiDoInicio



# ListaDE - Operações

1. **Inclui no inicio**
2. **Exibe lista**
3. Quantidade de nodos
4. Exibe situação da lista
5. **Exclui do início**
6. Inclui no fim
7. Exclui do fim
8. Consulta por código
9. Inclui Depois
10. Exclui nodo
11. Ler Arquivo texto
12. Gravar Arquivo texto



**INSTITUTO FEDERAL**  
Sul-rio-grandense

Câmpus  
Pelotas

EDUCAÇÃO  
**PÚBLICA**  
**100%**  
GRATUITA

# Estrutura de Dados

Aula 12

## Lista Duplamente Encadeadas

Alocação Dinâmica de Memória

**ListaDE**