Estruturas de Dados:

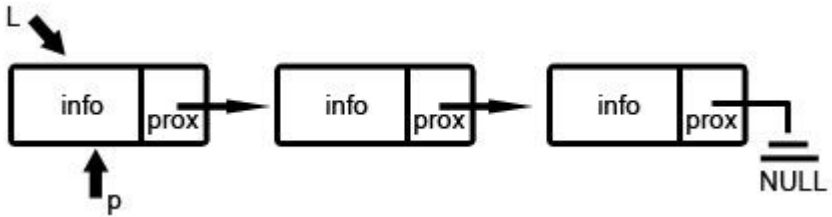
# Listas duplamente encadeadas

Helena Graziottin Ribeiro
hgrib@ucs.br

UCS
UNIVERSIDADE
DE CAXIAS DO SUL

# Listas duplamente encadeadas: por quê?

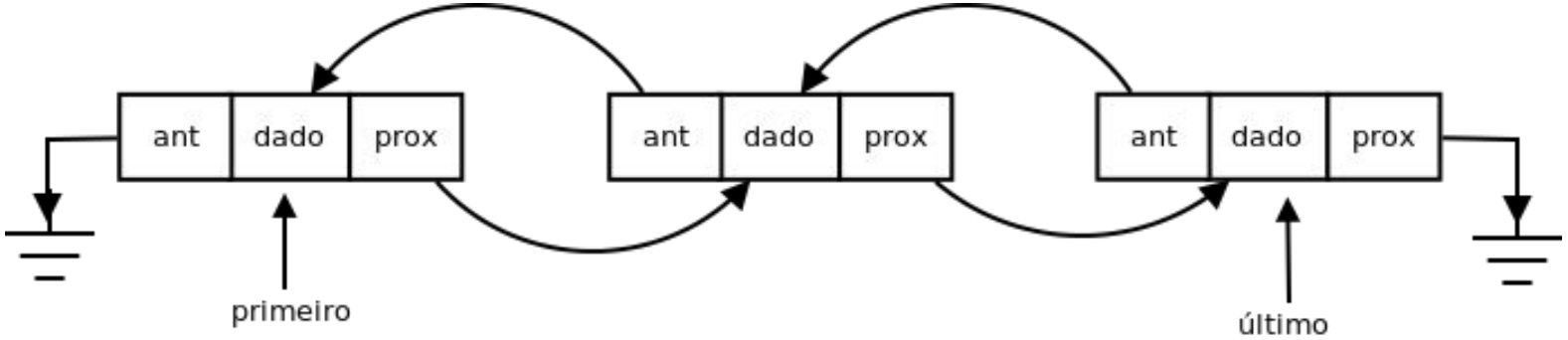
- na lista encadeada simples, cada elemento tem a referência (endereço de memória) do próximo elemento




- só é possível acessar o próximo elemento, não o anterior

# Listas duplamente encadeadas

- ou **listas encadeadas duplas**
- estruturas que usam **alocação dinâmica de memória**
- acesso aos elementos de forma sequencial:
  - bidirecional
  - cada nodo tem a referência do próximo e do anterior

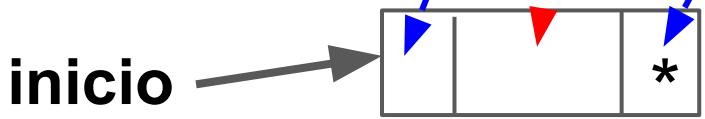


# Listas - como implementar? Listas encadeadas

- Lista encadeada simples:
  - seqüência encadeada de elementos, chamados de nós (ou nodos) da lista
  - cada nó da lista é representado por três partes:
    - a informação armazenada e
    - o ponteiro para o próximo elemento da lista
    - o ponteiro para o elemento anterior da lista

# Listas - como implementar? Listas encadeadas

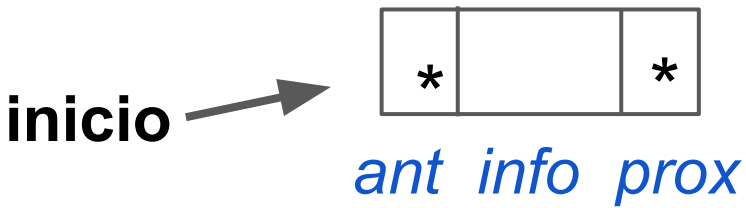
- cada nó da lista é representado por três partes:
  - a informação armazenada e
  - o ponteiro para o próximo elemento da lista
  - o ponteiro para o elemento anterior da lista




- a lista é representada por um **ponteiro para o primeiro nó (inicio)**

# Listas encadeadas duplas

```
struct elemento2 {
     int info;
     struct elemento2 *prox, *ant;
};
typedef struct elemento2 Elemento2;
```



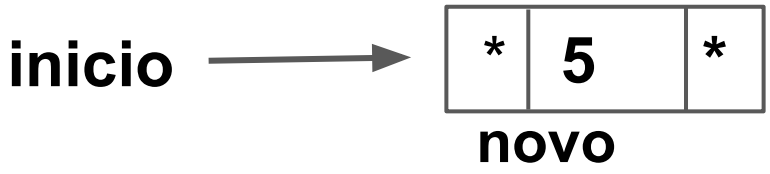
# Listas encadeadas duplas

## Operações em listas:

- inserção:
  - do primeiro e único
  - do primeiro
  - do último
  - no “meio”

# Listas encadeadas duplas: inserção (1º e único)

```
Elemento2 *inicio, *novo;
inicio = NULL; /* inicialização da lista */
...
novo = (Elemento2*) malloc(sizeof(Elemento2));
novo->info = 5;
novo->prox = NULL;
novo->ant = NULL;
inicio = novo;
```



# Listas encadeadas duplas: inserção (primeiro)

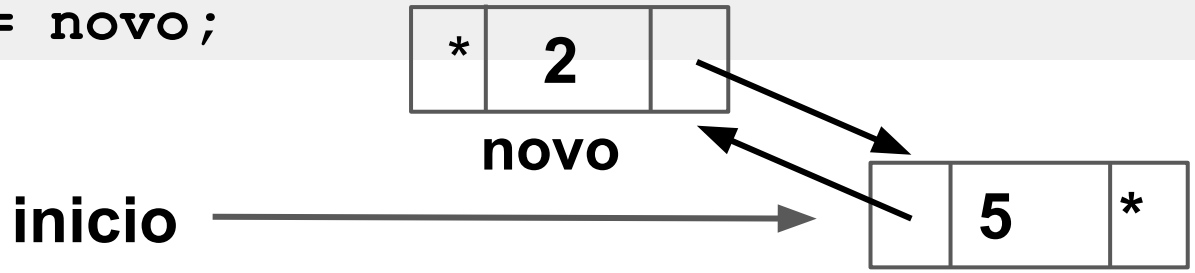
```
Elemento2 *novo;
novo = (Elemento2*) malloc(sizeof(Elemento2));
novo->info = 2;
novo->ant = NULL;
novo->prox = inicio;
inicio->ant = novo;
inicio = novo;
```

# Listas encadeadas duplas: inserção (primeiro)

```
Elemento2 *novo;
novo = (Elemento2*) malloc(sizeof(Elemento2));
novo->info = 2;
novo->ant = NULL;
novo->prox = inicio;
inicio->ant = novo;
inicio = novo;
```

# Listas encadeadas duplas: inserção (primeiro)

```
Elemento2 *novo;
novo = (Elemento2*) malloc(sizeof(Elemento2));
novo->info = 2;
novo->ant = NULL;
novo->prox = inicio;
inicio->ant = novo;
inicio = novo;
```

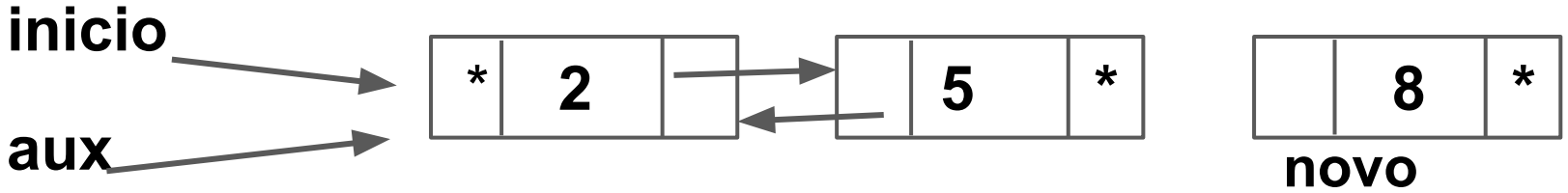


# Listas encadeadas duplas: inserção (primeiro)

```
Elemento2 *novo;
novo = (Elemento2*) malloc(sizeof(Elemento2));
novo->info = 2;
novo->ant = NULL;
novo->prox = inicio;
inicio->ant = novo;
inicio = novo;
```

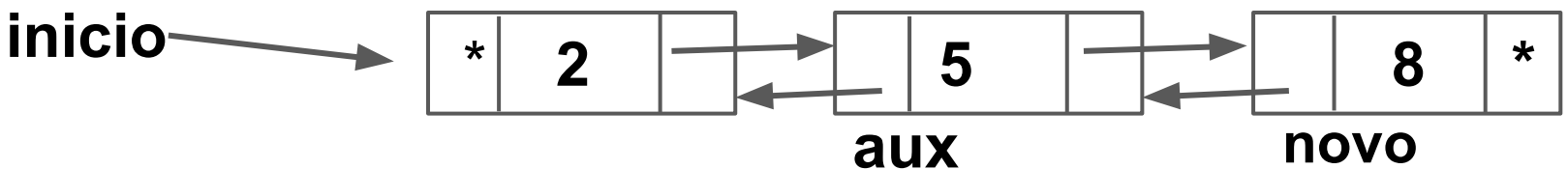
# Listas encadeadas duplas: inserção (último)

```
Elemento2 *novo, *aux=inicio;
novo = (Elemento2*) malloc(sizeof(Elemento2));
novo->info = 8;
novo->prox = NULL;
while (aux->prox != NULL)
      aux = aux->prox;
aux->prox = novo;
novo->ant = aux;
```



# Listas encadeadas duplas: inserção (último)

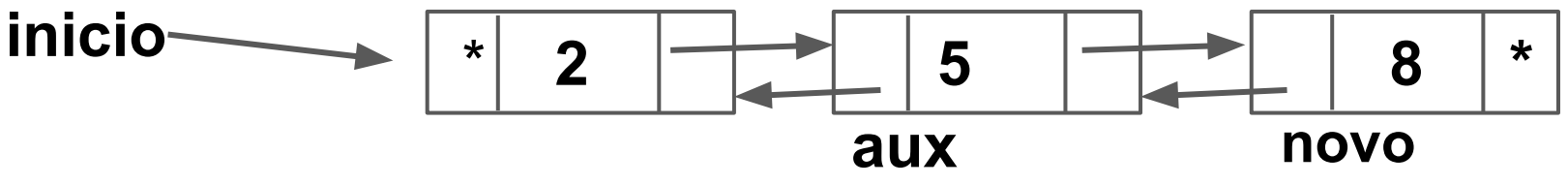
```
Elemento2 *novo, *aux=inicio;
novo = (Elemento2*) malloc(sizeof(Elemento2));
novo->info = 8;
novo->prox = NULL;
while (aux->prox != NULL)
      aux = aux->prox;
aux->prox = novo;
novo->ant = aux;
```



# Listas encadeadas duplas: inserção (no meio)

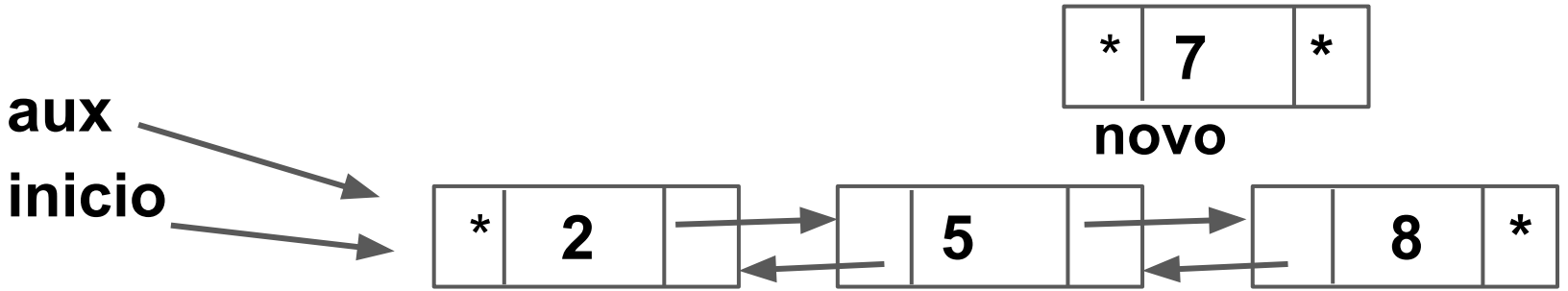
- Deve haver algum critério para caracterizar a inserção no meio, por exemplo:
  - inserção ordenada
  - inserção em posição determinada

**Exemplo: inserir o 7, em ordem crescente de valores**



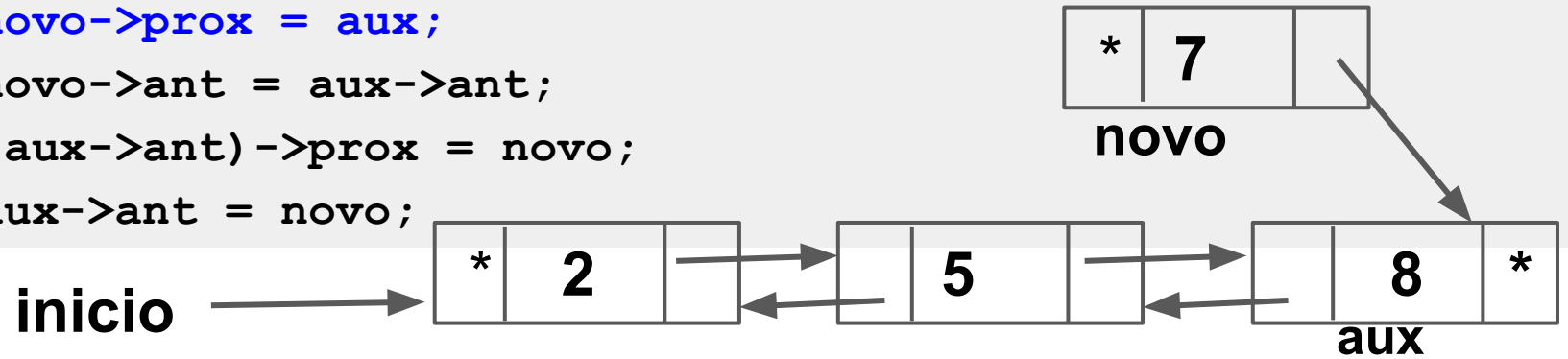
# Listas encadeadas duplas: inserção (no meio)

```
Elemento2 *novo, *aux=inicio;
novo = (Elemento2*) malloc(sizeof(Elemento2));
novo->info = 7;
novo->prox = NULL;
novo->ant = NULL;
...
```



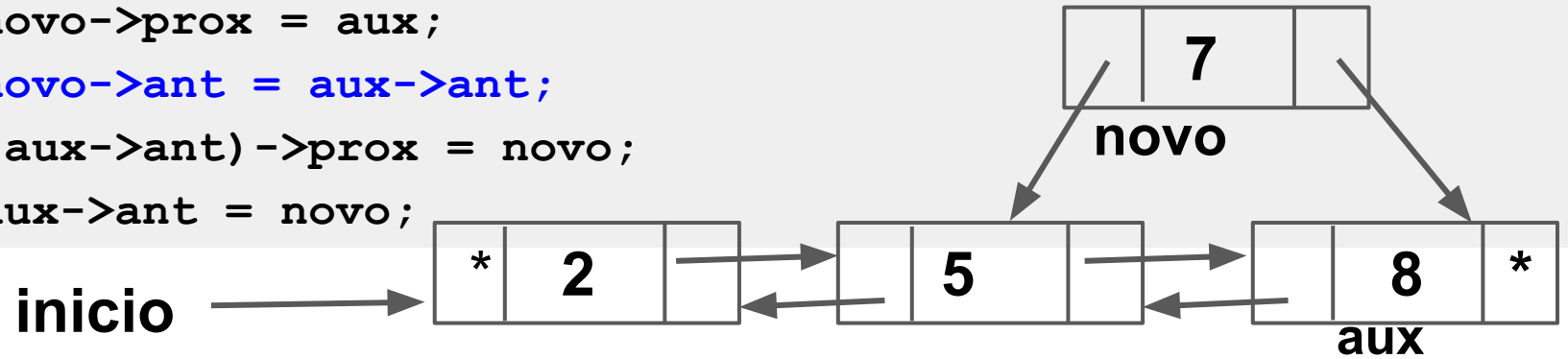
# Listas encadeadas duplas: inserção (no meio)

```
Elemento2 *novo, *aux=inicio;
novo = (Elemento2*) malloc(sizeof(Elemento2));
novo->info = 7;
novo->prox = NULL;
novo->ant = NULL;
while (aux != NULL && aux->info < novo->info ){
       aux = aux->prox;}
novo->prox = aux;
novo->ant = aux->ant;
(aux->ant)->prox = novo;
aux->ant = novo;
```



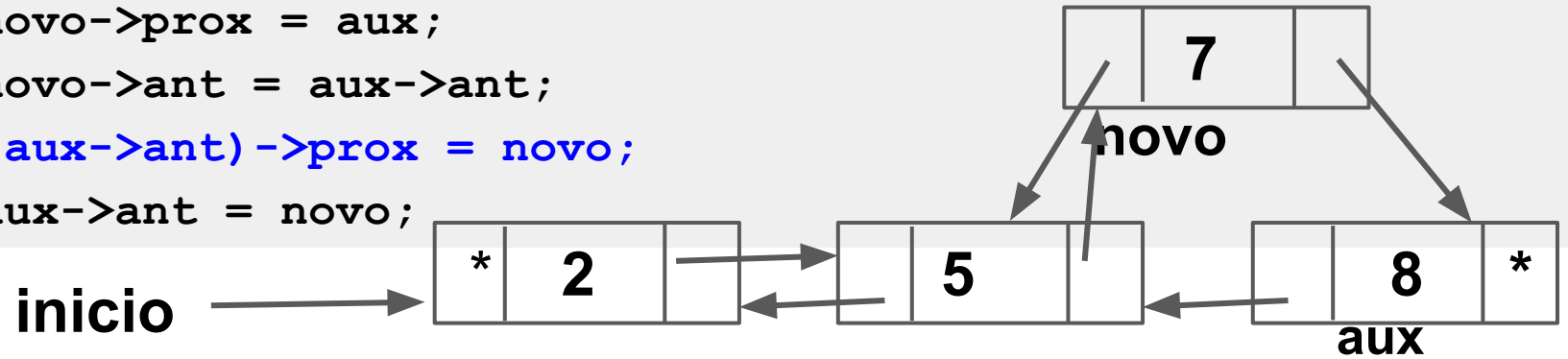
# Listas encadeadas duplas: inserção (no meio)

```
Elemento2 *novo, *aux=inicio;
novo = (Elemento2*) malloc(sizeof(Elemento2));
novo->info = 7;
novo->prox = NULL;
novo->ant = NULL;
while (aux != NULL && aux->info < novo->info ){
       aux = aux->prox;}
novo->prox = aux;
novo->ant = aux->ant;
(aux->ant)->prox = novo;
aux->ant = novo;
```



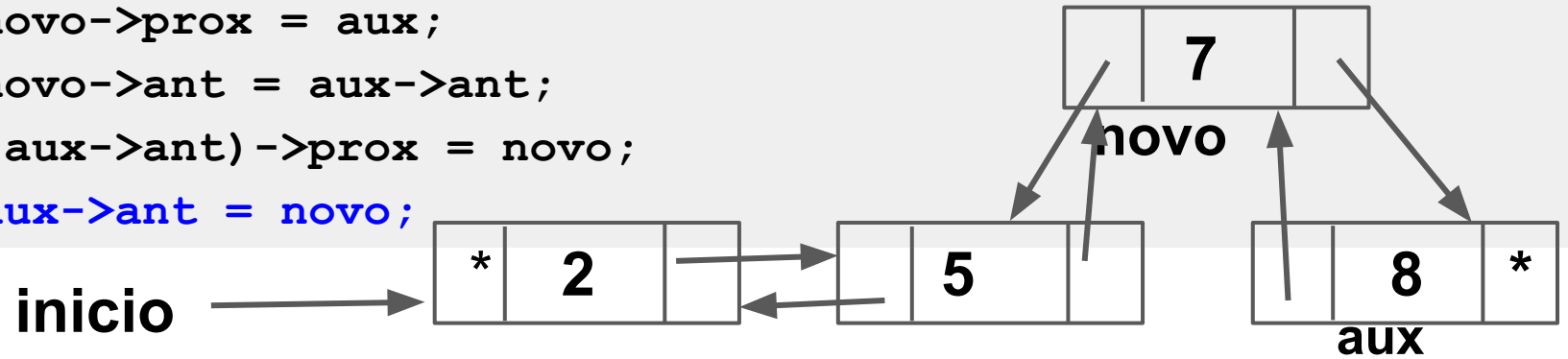
# Listas encadeadas duplas: inserção (no meio)

```
Elemento2 *novo, *aux=inicio;
novo = (Elemento2*) malloc(sizeof(Elemento2));
novo->info = 7;
novo->prox = NULL;
novo->ant = NULL;
while (aux != NULL && aux->info < novo->info ){
       aux = aux->prox;}
novo->prox = aux;
novo->ant = aux->ant;
(aux->ant)->prox = novo;
aux->ant = novo;
```



# Listas encadeadas duplas: inserção (no meio)

```
Elemento2 *novo, *aux=inicio;
novo = (Elemento2*) malloc(sizeof(Elemento2));
novo->info = 7;
novo->prox = NULL;
novo->ant = NULL;
while (aux != NULL && aux->info < novo->info ){
       aux = aux->prox;}
novo->prox = aux;
novo->ant = aux->ant;
(aux->ant)->prox = novo;
aux->ant = novo;
```



# Listas encadeadas duplas - inserção

```
/* inserção no início: retorna a lista atualizada */
Elemento2* lst_insere (Elemento2* lst, int val)
{
 Elemento2* novo = (Elemento2*) malloc(sizeof(Elemento2));
 novo->info = val;
 novo->prox = lst;
 novo->ant = NULL;
 return novo;
}
...
inicio = lst_insere (inicio, 9);
```