Listas Lineares

Jacson Luiz Matte, Prof.

Estrutura de Dados I

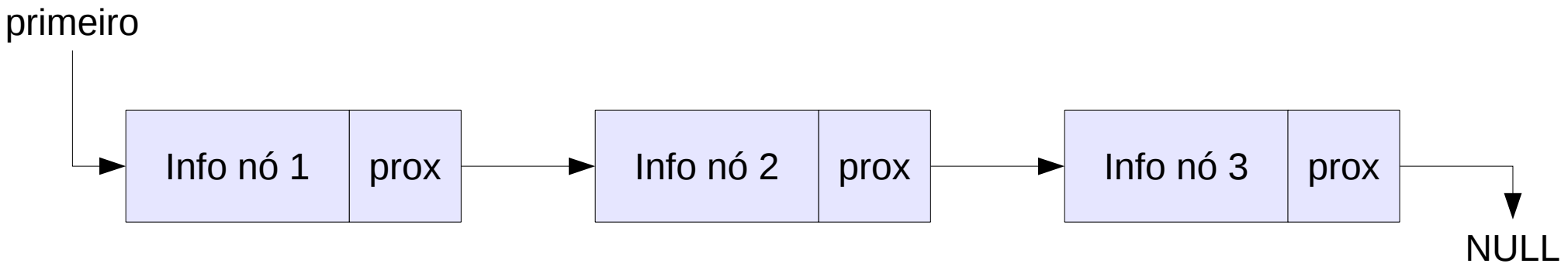
Listas Lineares Encadeadas

- Listas implementadas com arrays podem não ser eficientes:
  - Adicionar/remover um elemento na primeira posição de um vetor requer o deslocamento de todos os outros elementos;
  - A performance dessa operação piora com o aumento do tamanho do vetor;
  - Listas encadeadas são mais eficientes para estes casos.

Listas Lineares Encadeadas

- Numa lista encadeada não podemos garantir que os elementos serão armazenados em posições contíguas de memória (como nos arrays).
  - Não temos acesso direto aos elementos da lista;
  - Devemos guardar o encadeamento dos elementos;
    - Ponteiro para o próximo elemento da lista;

Listas Lineares Encadeadas



- A lista é representada por um ponteiro para o primeiro elemento;
- O segundo e demais elementos são alcançados seguindo o encadeamento;
- O último elemento aponta para NULL.

Listas Lineares Encadeadas

- Exemplo: armazenar valores inteiros numa lista encadeada.
  - Representação de cada nó da lista:

```
typedef struct _lista {
    int info;
    struct _lista * prox;
}TpLista;
```

  - O tipo Lista representa um nó da lista;
  - A estrutura de lista encadeada é representada pelo ponteiro para seu primeiro elemento.

Função de Inicialização

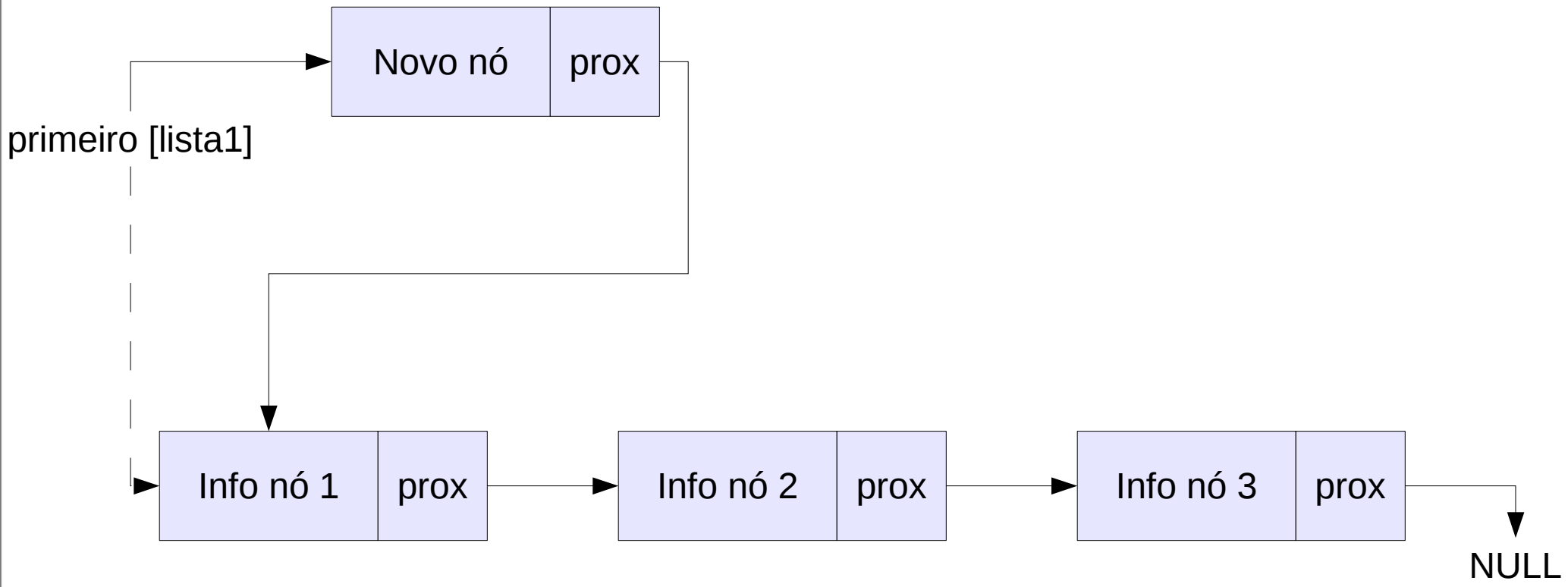
Listas Lineares Encadeadas

- Cria uma lista vazia (sem elementos);
- Uma lista vazia é representada pelo ponteiro NULL;
- O valor de retorno da função é o valor NULL (a lista inicializada);

```
/* função de inicialização */
TpLista* inicializa(){
   return NULL;
}
```

Função de Inserção no Início

Listas Lineares Encadeadas



Função de Inserção no Início

Listas Lineares Encadeadas

- A memória para cada novo elemento da lista deve ser alocada dinamicamente (suficiente para o elemento e o ponteiro);
- O ponteiro que representa a lista deve ser atualizado;
- O valor de retorno é uma nova lista, representada pelo ponteiro para o novo elemento.

```
/* função de inserção no iníco da lista */
TpLista* insereini(TpLista* l, int e){
   TpLista* novo = (TpLista *)
malloc(sizeof(TpLista));
   novo->info = e;
   novo->prox = l;
   return novo;
}
```

Programa exemplo

Inicializa e insere no início: Listas Lineares Encadeadas

```
/* Inicializa e insere elementos no início da lista */
#include <stdio.h>

/* Protótipo das Funções */

TpLista* insereini(TpLista* l, int i);
TpLista* inicializa();


/* Função principal */
int main(){
    TpLista* l;  /* declara uma lista não inicializada */
    l = inicializa(); /* inicializa lista como vazia */
    l = insereini(l, 10); /* insere o elemento 10 */
    l = insereini(l, 25); /* insere o elemento 25 */
return 0;
}
//... implementação das funções...
```

Função para Percorrer elementos

Listas Lineares Encadeadas

- Considerando a impressão dos valores dos elementos:

```
/* função para percorrer lista. Imprime os valores
dos elementos */

void imprime(TpLista* l){
    TpLista* p;
    for(p = l; p != NULL; p = p->prox){
        printf("Informação = %d\n", p->info);
    }
}
```

Função para Verificar lista (vazia?)

Listas Lineares Encadeadas

- Uma lista está vazia quando seu valor é NULL.

```
/* função evazia: retorna 1 se vazia ou 0 se não vazia */

int evazia(TpLista* l){
    if(l == NULL){
        return 1;
    }else{
        return 0;
    }
}
```

Função para Buscar Elementos

Listas Lineares Encadeadas

- Dado um elemento para busca, a função retorna o ponteiro do nó da lista que representa o elemento;
- Retorna NULL se não encontrar o elemento.

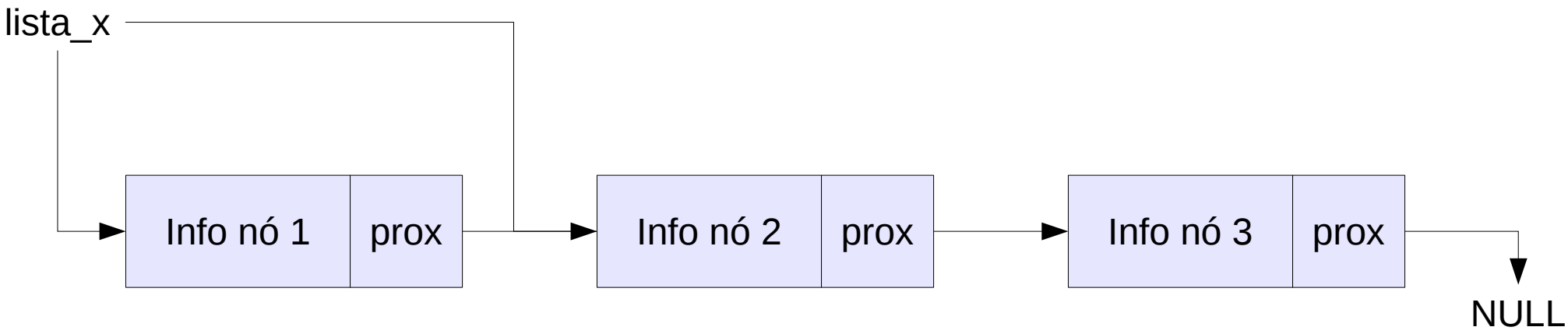
```
/* função busca: retorna o ponteiro para o elemento ou NULL */

TpLista* busca(TpLista* l, int e){
    TpLista* p;
    for(p = l; p != NULL; p=p->prox){
        if(p->info == e){
            return p;
        }
    }
    return NULL; //O elemento não está na lista
}
```

Função para Excluir Elementos

Listas Lineares Encadeadas

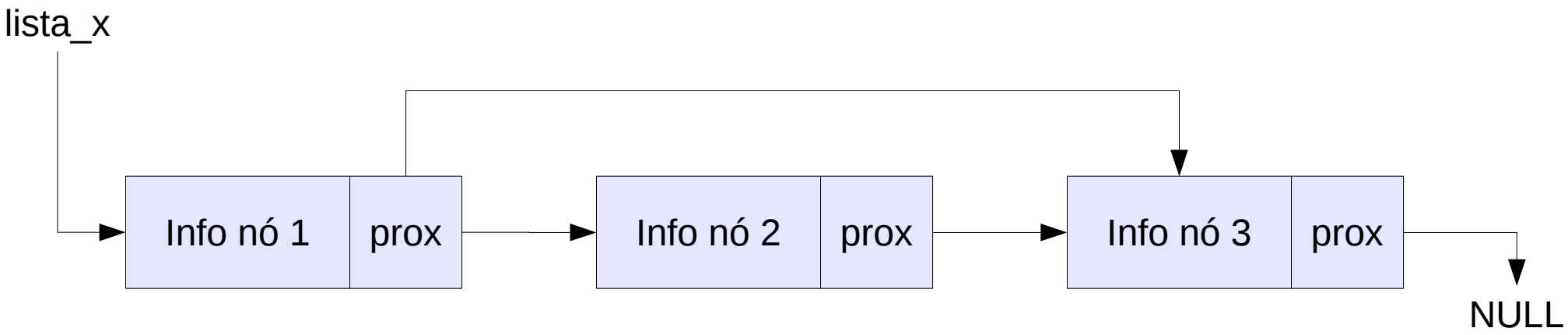
- Dado um elemento para exclusão, a função retorna um ponteiro para a lista atualizada;
- Retirar um elemento do início da lista requer um processo diferente de retirar no meio da lista;
  - Remoção no início da lista:



Função para Excluir Elementos

Listas Lineares Encadeadas

- Remoção no meio da lista:



Função para Excluir Elementos

Listas Lineares Encadeadas

```
/* função exclui: retorna o ponteiro para o elemento ou NULL */

Lista * exclui(TpLista * l, int e){
    TpLista * anterior = NULL; // ponteiro para o elemento anterior
    TpLista * p = l; // ponteiro para percorrer a lista

    /* procura elemento na lista e guarda o anterior */
    while(p != NULL && p->info != e){
        anterior = p;
        p = p->prox;
    }

    /* verifica se achou elemento */
    if(p == NULL){
        return l; // não achou o elemento, retorna a lista original
    }
```

Função para Excluir Elementos

Listas Lineares Encadeadas

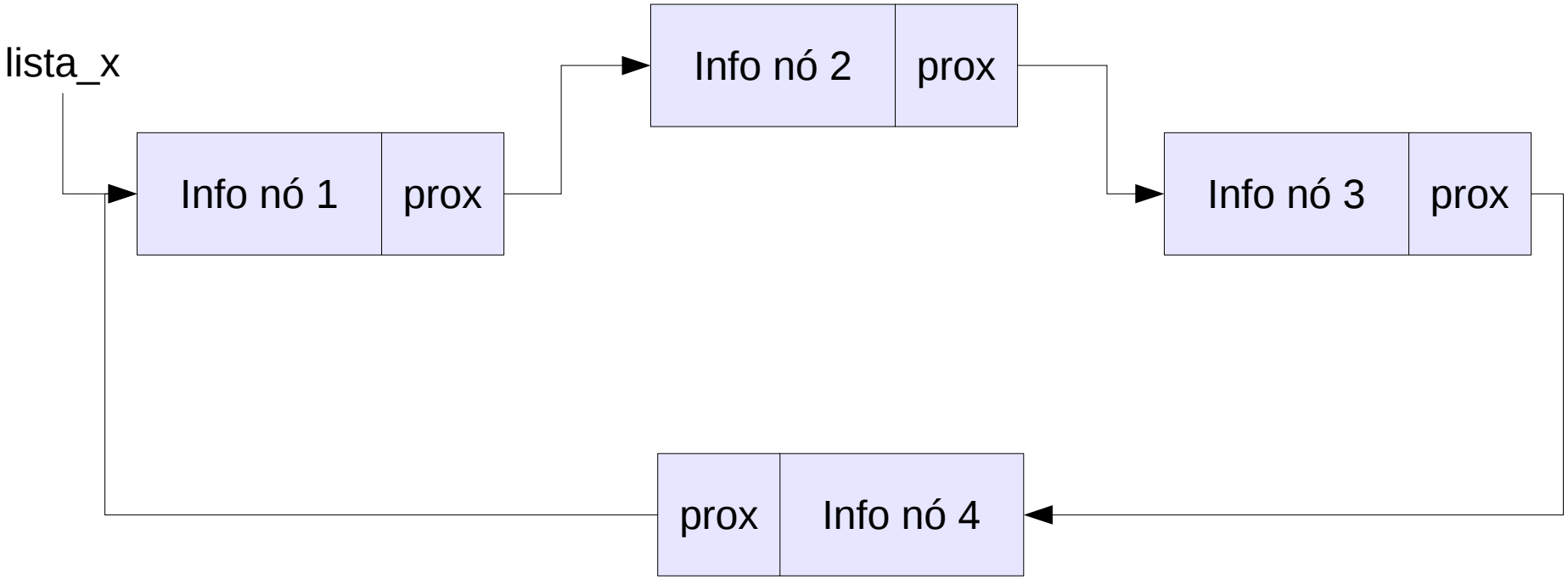
```
/* retira elemento */
    if(anterior == NULL){ // então retirar o primeiro elemento
        l = p->prox;
    }else{ // retira elemento do meio da lista
        anterior->prox = p->prox;
    }
    free(p);
    return l;
} // fim de Lista * exclui...
```

Exercício: implemente uma função para inserir no meio da lista (em ordem crescente).

Listas Circulares

Listas Lineares Encadeadas

- O último elemento aponta para o primeiro elemento;



Listas Circulares

Listas Lineares Encadeadas

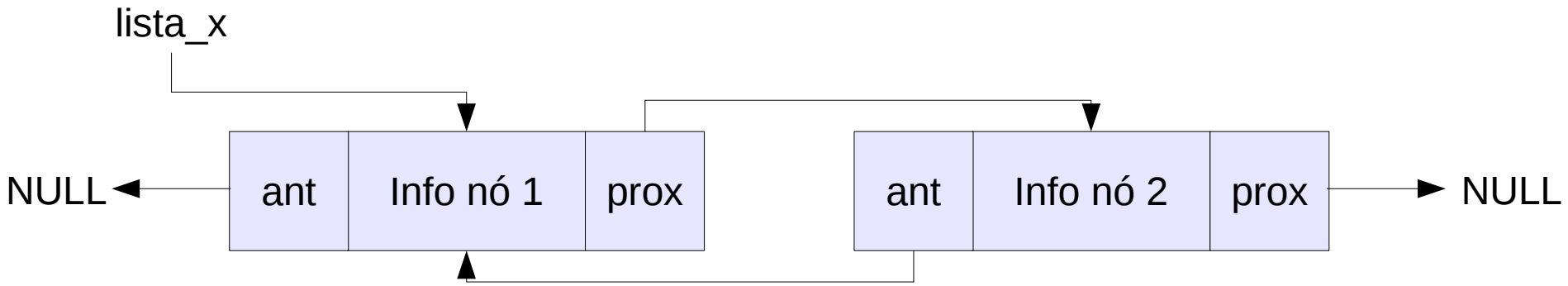
- Percorrer toda a lista requer que visitemos todos os elementos a partir do ponteiro para o elemento inicial, até alcançar novamente o mesmo elemento;

```
void imprime_circular (TpLista* l){
    TpLista* p = l; // p aponta para o elemento inicial

    if (p != NULL){ // lista não está vazia
        do {
            printf("%d\n", p->info); // imprime info do nó
            p = p->prox; // avança para o próximo nó
        }while(p != l);
    }else{
        printf(”\nA lista está vazia\n”);
    }
}
```

Listas Duplamente Encadeadas

- Cada nó tem um ponteiro para o próximo e para o nó anterior;
- Pode-se percorrer a lista em ordem inversa;



Listas Lineares Encadeadas

- Exemplo: armazenar valores inteiros numa lista duplamente encadeada.
  - Representação de cada nó da lista:

```
typedef struct
_listaDupla {
    int info;
    struct _listaDupla
* ant;
    struct _listaDupla
* prox;
}TpListaDupla;
```

  - O tipo ListaDupla representa um nó da lista;