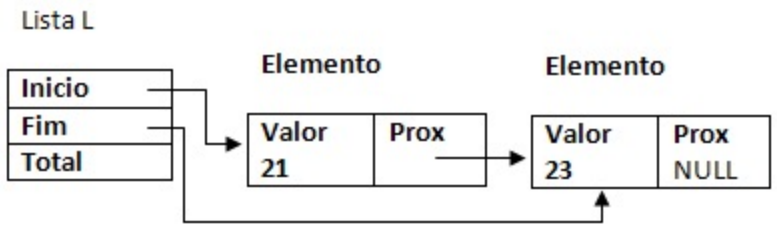
Estrutura de Dados

Listas Lineares

Listas Lineares

- Uma estrutura de dados dinâmica composta por elementos encadeados.
- Os elementos são encadeados por meio de “ponteiros”.
- As listas lineares ou listas encadeadas podem ser de dois tipos:
  - Listas Lineares **simplesmente** encadeadas;
  - Listas Lineares **duplamente** encadeadas.

Lista Linear Simplesmente Encadeada



Em uma Lista Linear Simplesmente Encadeada os ponteiros são empregados Apenas para “apontar” o próximo elemento.

Note que um elemento aponta apenas para o seguinte e nunca para o anterior:

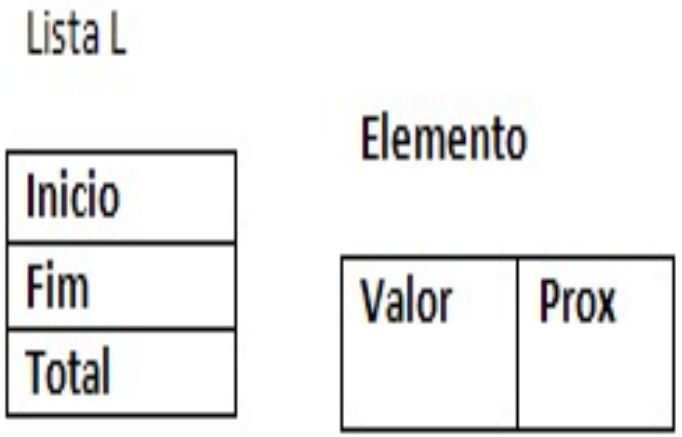
Existem apenas 2 elementos na Lista.

A Lista começa com 21 e termina com 23.

21 aponta para 23, mas 23 não aponta para 21.

Lista Linear Simplesmente Encadeada

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

typedef struct tipoElemento {
    int valor;
    struct tipoElemento *prox;
}TElemento;

typedef struct tipoLista {
    TElemento *inicio;
    TElemento *fim;
    int total;
}TLista;
```



Lista Linear Simplesmente Encadeada

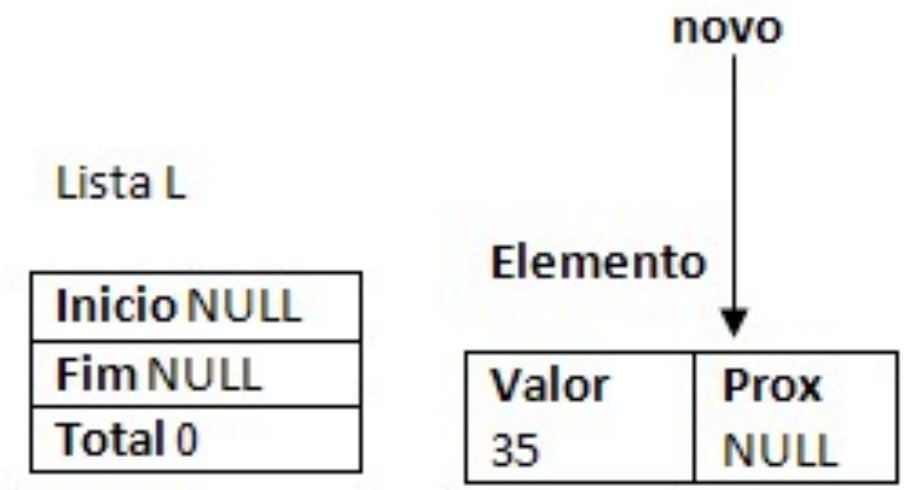
```
void inicializaLista(TLista *lista){
     lista->inicio = NULL;
     lista->fim = NULL;
     lista->total = 0;
}

int main(){
     inicializaLista(&L);

}
```

Lista L

Inicio NULL
Fim NULL
Total 0

Lista Linear Simplesmente Encadeada

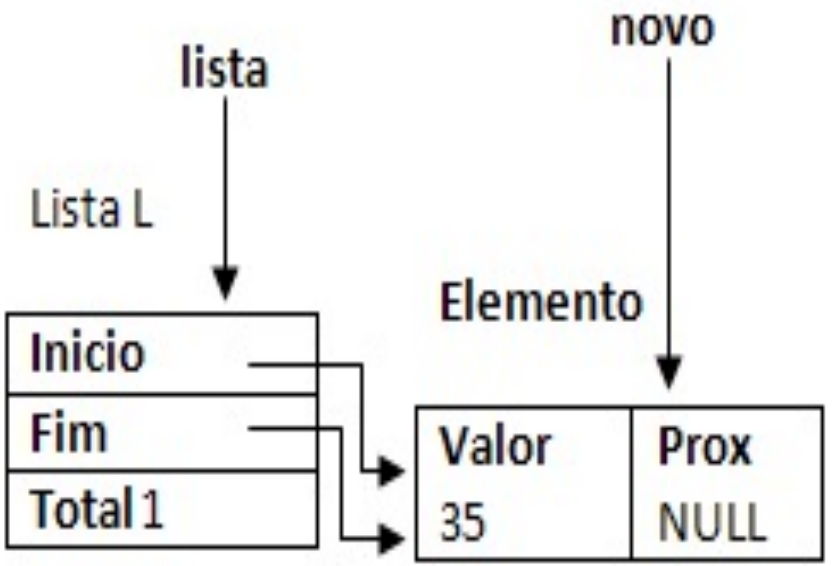
```
void insereElemento(TLista *lista, int numero){
    TElemento *novo = (TElemento *)malloc(sizeof(TElemento));

    novo->valor = numero;
    novo->prox = NULL;

}

int main(){
    inicializaLista(&L);

    insereElemento(&L, 35);

}
```



Falta ainda inserir o Elemento 35 na Lista!!!

Lista Linear Simplesmente Encadeada

```
void insereElemento(TLista *lista, int numero){
    TElemento *novo = (TElemento *)malloc(sizeof(TElemento));

    novo->valor = numero;
    novo->prox = NULL;

    if(lista->inicio == NULL){
       //A Lista está VAZIA !!!
       lista->inicio = novo;
       lista->fim = novo;
       lista->total = 1;
    }//if

}//insereElemento( )
```



Lista Linear Simplesmente Encadeada

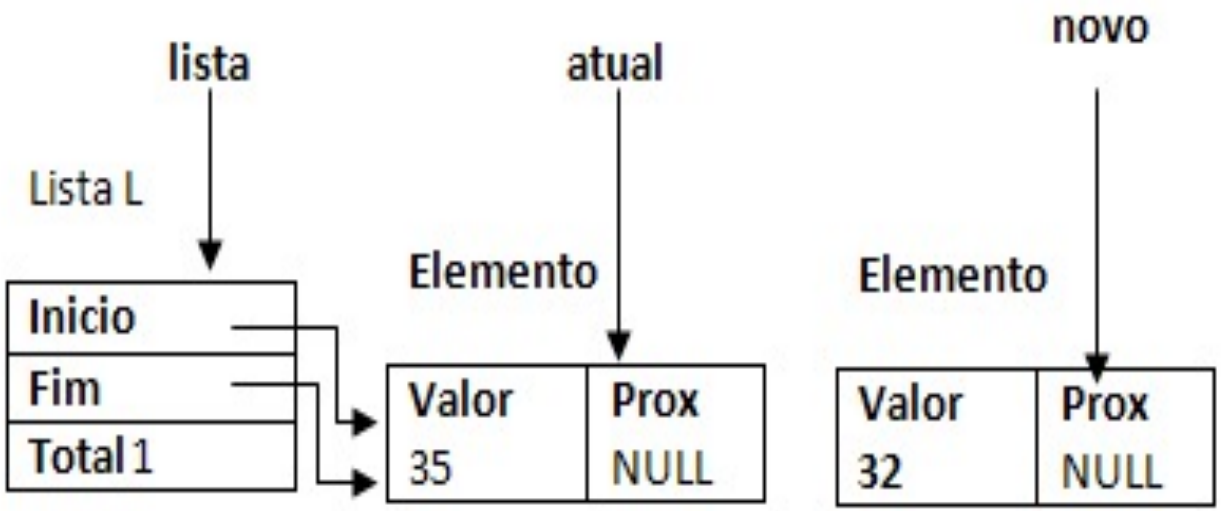
- Por enquanto, nossa Lista encontra-se preparada para **inserir** apenas o primeiro e único **Elemento**.
- Suponha que agora desejemos inserir um Elemento de valor **32**.
- Esse Elemento deverá ser inserido antes do Elemento de valor **35** (para manter a ordem crescente).

Lista Linear Simplesmente Encadeada

```
int main(){
     inicializaLista(&L);

     insereElemento(&L, 35);
     insereElemento(&L, 32);

}//main( )
```

Nossa **Lista** já possui um Elemento de valor 35. E agora inserimos outro de Valor 32.

Lista Linear Simplesmente Encadeada

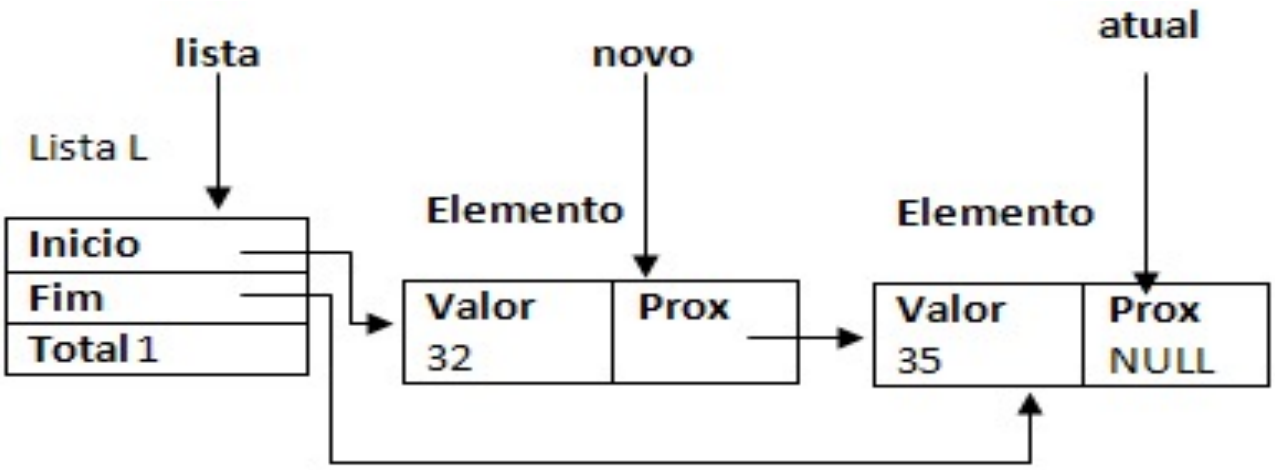


Note que **if(lista->inicio == NULL){** resulta em FALSO – A Lista Não está vazia!

O ponteiro atual “aponta” para Elemento 35 e novo “aponta” para 32.

A condicional **if (atual->valor >= novo->valor){** resulta em verdadeiro, pois O valor apontado por atual (= 35) é MAIOR que o valor apontado por novo (= 32).

Lista Linear Simplesmente Encadeada



Agora o Elemento de valor 32 é o primeiro da Lista.

O Elemento de valor 35 continua sendo o último da Lista.

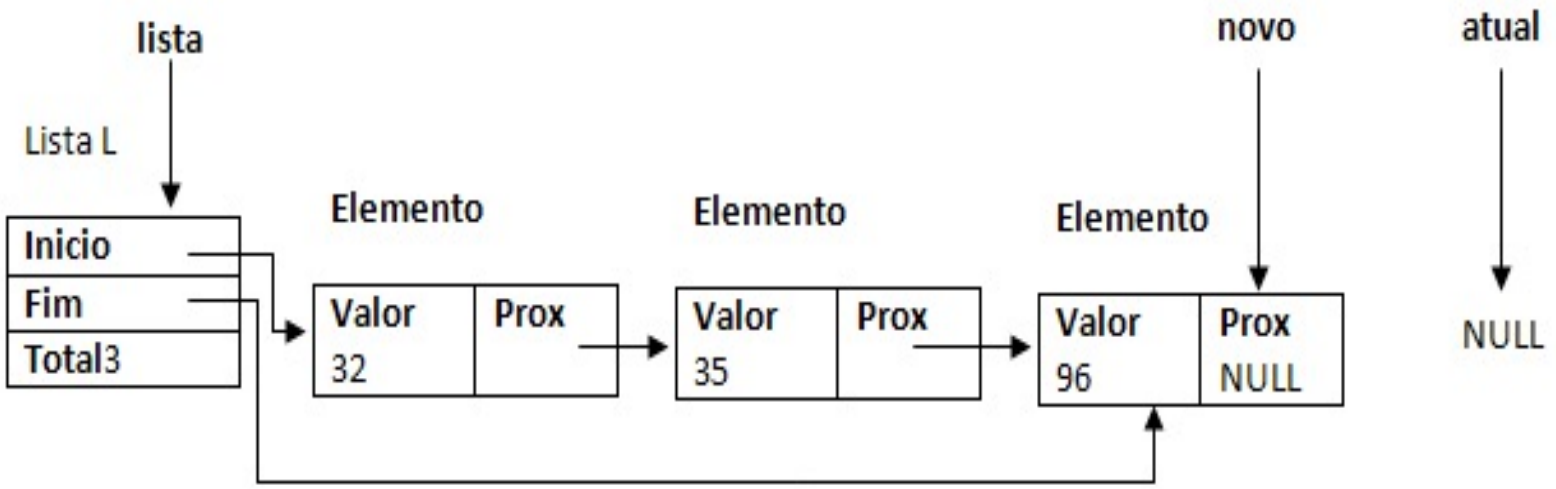
O Elemento 32 “aponta” para o Elemento 35 por meio de seu PROX.

O PROX de 35 tem valor NULL (não “aponta” para ninguém).

Lista Linear Simplesmente Encadeada

```
int main(){
    inicializaLista(&L);

    insereElemento(&L, 35);
    insereElemento(&L, 32);
    insereElemento(&L, 96);

}//main( )
```

Nosso próximo passo será **inserir** Um novo Elemento de valor 96.



Lista Linear Simplesmente Encadeada

```
void insereElemento(TLista *lista, int numero){
    TElemento *novo = (TElemento *)malloc(sizeof(TElemento));
    TElemento *atual = NULL;
    TElemento *anterior;
    bool inserido = 0;

    novo->valor = numero;
    novo->prox = NULL;

    if(lista->inicio == NULL){
       //A Lista está VAZIA !!!
       lista->inicio = novo;
       lista->fim = novo;
       lista->total = 1;
```

Lista Linear Simplesmente Encadeada

```
} else {
    //ponteiros atual e anterior percorrem a LISTA
    atual = lista->inicio;
    anterior = NULL;

    while (atual != NULL){
        if (atual->valor >= novo->valor){
            novo->prox = atual;
            if (anterior == NULL) lista->inicio = novo;
            else anterior->prox = novo;
            inserido = 1; lista->total++;
            break; //interrompe o WHILE
        }//if
        anterior = atual;
        atual = atual->prox;
    }//while
```

Lista Linear Simplesmente Encadeada

```
        if (!inserido){
            //SE mesmo após percorrer toda a LISTA o NOVO não foi inserido
            lista->fim->prox = novo; //insere NOVO no FIM da LISTA
            lista->fim = novo;
        }//if
    }//if
}//insereElemento( )

int main(){
    inicializaLista(&L);

    insereElemento(&L, 35);
    insereElemento(&L, 32);

}//main( )
```

Lista Linear Simplesmente Encadeada

```
void exibeElementosLista(TLista lista){
    TElemento *atual = lista.inicio;

    printf("\n\n\nElementos Existentes na LISTA: \n\n");

    while (atual != NULL){
         printf("%d\t", atual->valor);
         atual = atual->prox;
    }//while

    printf("\n\n");
    system("PAUSE");

}//exibeElementosLista( )

int main(){
     inicializaLista(&L);

     insereElemento(&L, 35);
     insereElemento(&L, 32);
     insereElemento(&L, 96);

     exibeElementosLista(L);

}//main( )
```

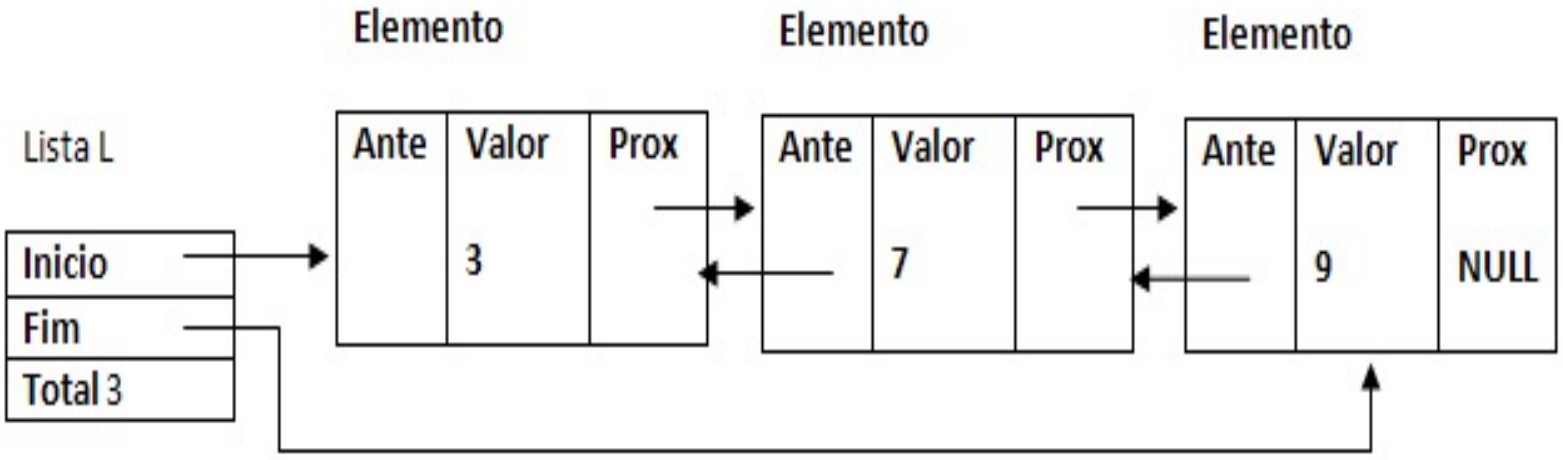
```
Elementos Existentes na LISTA:
32      35      96
Pressione qualquer tecla para continuar. . .
```

Lista Linear Simplesmente Encadeada

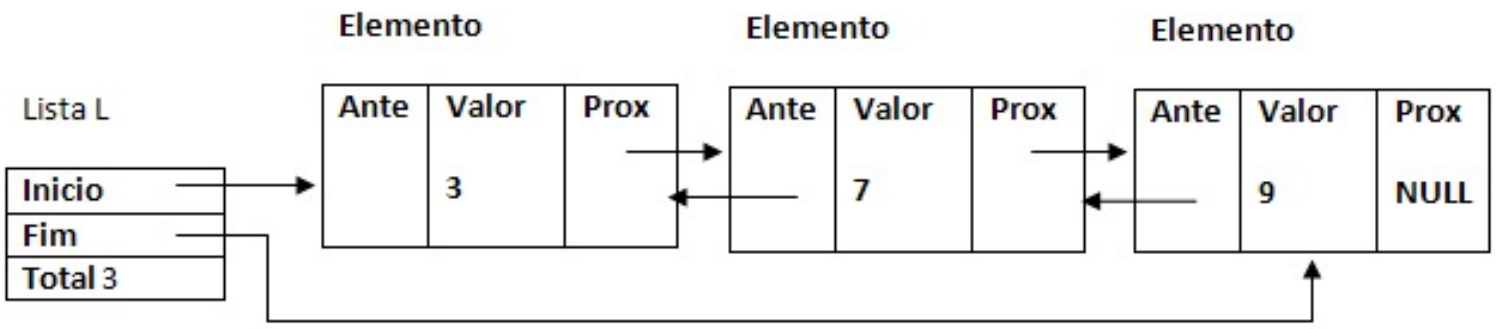
```
int excluiElemento(TLista *lista, int argumento){
      TElemento *atual, *anterior;
      bool excluido = 0;

      atual = lista->inicio;
      anterior = NULL;

      while (atual != NULL){
         if(atual->valor == argumento){
            excluido = 1;

            if (anterior == NULL) lista->inicio = atual->prox;
            else anterior->prox = atual->prox;

            if (atual == lista->fim) lista->fim = NULL;

            free(atual);

            break; //interrompe WHILE
         }//if

         anterior = atual;  //anterior sempre fica um passo atrás de atual
         atual = atual->prox; //move atual para o Elemento seguinte
      }//while

      return excluido;
}//excluiElemento( )
```

Listas Duplamente Encadeadas

- As **Listas duplamente encadeadas** apresentam ponteiros que apontam para o Elemento seguinte (**PROX**) e ponteiros que apontam para o Elemento anterior (**ANTE**).



Listas Duplamente Encadeadas



O fato de ter dois ponteiros em cada elemento aumenta o consumo de Memória, mas facilita a implementação dos procedimentos de inserção e exclusão, dispensando a necessidade de se ter o ponteiro para o “anterior”.