

# Computação 2

## Aula 10.1

### Listas Duplamente Encadeadas

Prof<sup>a</sup>. Fabiany  
fabiany1@utfpr.edu.br

# ListaEncadeada.h

```
#ifndef _LISTAENCADEADA_H_
#define _LISTAENCADEADA_H_

#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>

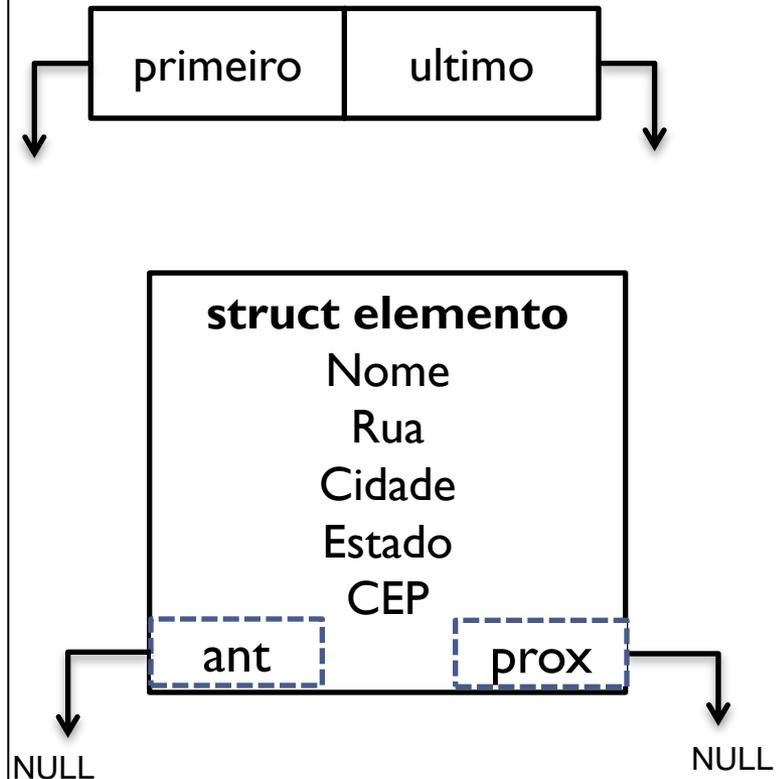
struct Elemento
{
    char nome [100];
    char rua [100];
    char cidade [100];
    char estado [2];
    char cep [10];
    struct Elemento* proximo;
    struct Elemento* anterior;
};

struct Lista
{
    struct Elemento* primeiro;
    struct Elemento* ultimo;
};

char menu ();
void inicia_lista ();
void cadastra ();
void mostra ();
void mostraReverso ();
void limpaLista ();

#endif
```

## Minha Lista



## ListaEncadeada.c

```
#include "ListaEncadeada.h"

struct Lista MinhaLista;

void inicia_lista ()
{
    MinhaLista.primeiro = NULL;

    MinhaLista.ultimo = NULL;
}
```

```
char menu ()
{
    printf ("\n \n \n");
    char opcao;

    printf ( "(C)adastrar. \n" );
    printf ( "(M)ostrat. \n" );
    printf ( " (M)ostrat (R)eversamente. \n" );
    printf ( "(T)erminar. \n" );

    fflush ( stdin );
    scanf ( "%c", &opcao );

    return opcao;
}
```

```
void cadastra()
{
    system ( "cls" ); printf ("\n \n \n");
    register int i;

    struct Elemento* novo;
    novo = malloc ( 1 * sizeof (struct Elemento) );
    novo->proximo = NULL;
    novo->anterior = NULL;

    printf ( "Nome: \n" );
    fflush ( stdin ); gets ( novo->nome );

    printf ( "Rua: \n" );
    fflush ( stdin ); gets ( novo->rua );

    printf ( "Cidade: \n" );
    fflush ( stdin ); gets ( novo->cidade );

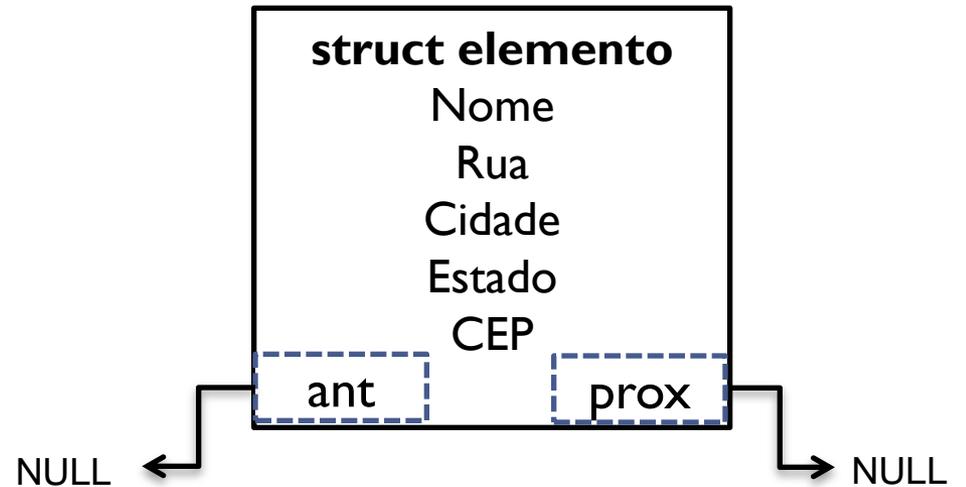
    printf ( "Estado: \n" );
    fflush ( stdin ); gets ( novo->estado );

    printf ( "CEP: \n" );
    fflush ( stdin ); gets ( novo->cep );

    if ( NULL == MinhaLista.primeiro )
    {
        MinhaLista.primeiro = novo;
        MinhaLista.ultimo = MinhaLista.primeiro;
    }
    else
    {
        MinhaLista.ultimo->proximo = novo;
        novo->anterior = MinhaLista.ultimo;
        MinhaLista.ultimo = novo;
    }
}
```

# ListaEncadeada.c

```
#include "ListaEncadeada.h"  
  
struct Lista MinhaLista;  
  
void inicia_lista ()  
{  
    MinhaLista.primeiro = NULL;  
    MinhaLista.ultimo = NULL;  
}
```



```

void cadastra()
{
    system ( "cls" );  printf ( "\n \n \n" );
    register int i;

    struct Elemento* novo;
    novo = malloc ( 1 * sizeof ( struct Elemento ) );
    novo->proximo = NULL;
    novo->anterior = NULL;

    printf ( "Nome: \n" );
    fflush ( stdin );  gets ( novo->nome );

    printf ( "Rua: \n" );
    fflush ( stdin );  gets ( novo->rua );

    printf ( "Cidade: \n" );
    fflush ( stdin );  gets ( novo->cidade );

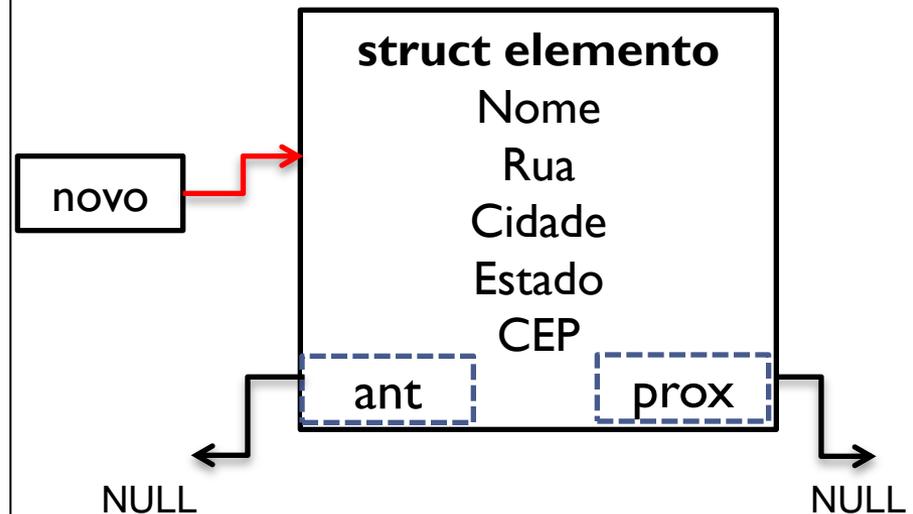
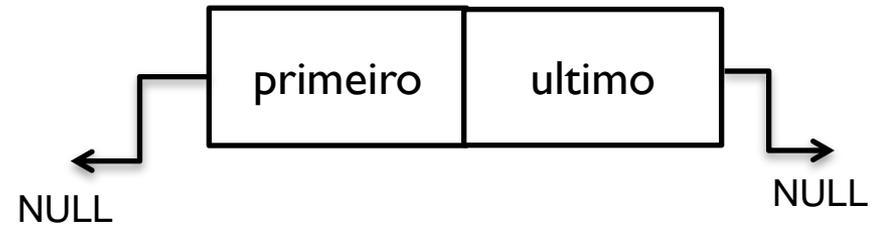
    printf ( "Estado: \n" );
    fflush ( stdin );  gets ( novo->estado );

    printf ( "CEP: \n" );
    fflush ( stdin );  gets ( novo->cep );

    if ( NULL == MinhaLista.primeiro )
    {
        MinhaLista.primeiro = novo;
        MinhaLista.ultimo = MinhaLista.primeiro;
    }
    else
    {
        MinhaLista.ultimo->proximo = novo;
        novo->anterior = MinhaLista.ultimo;
        MinhaLista.ultimo = novo;
    }
}

```

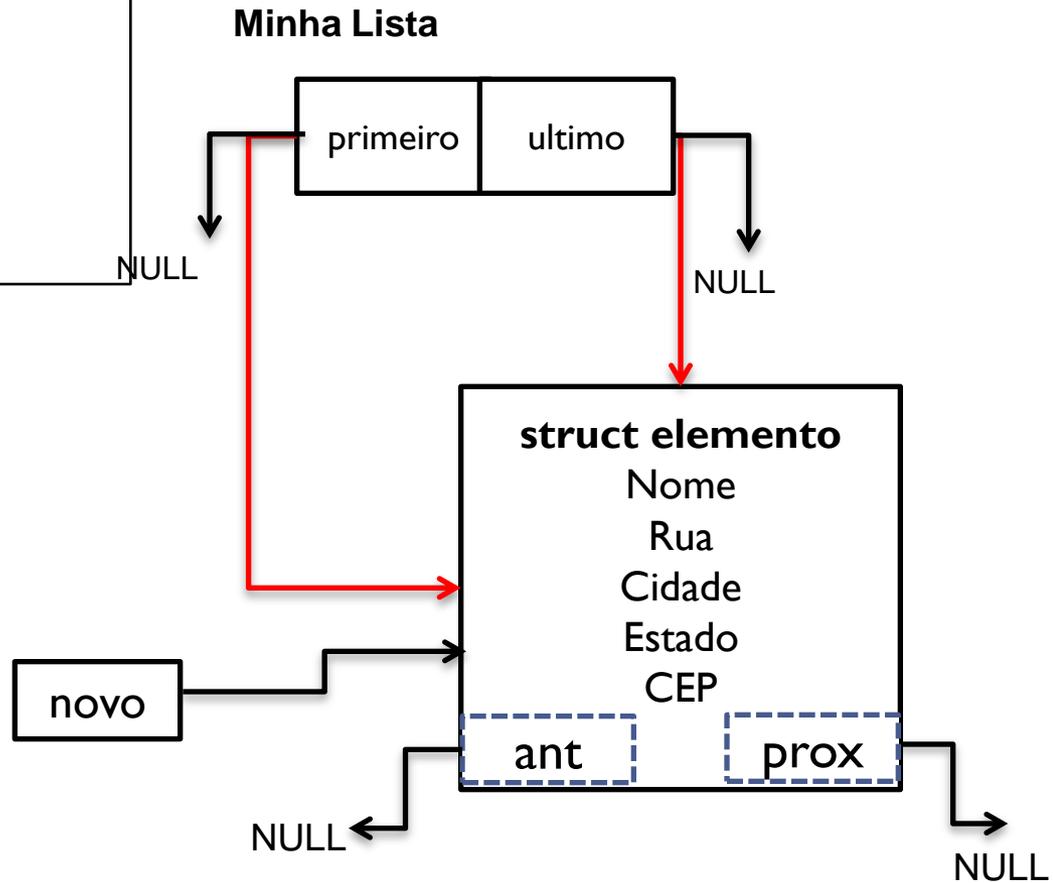
## Minha Lista



```

void cadastra()
{
    if ( MinhaLista.primeiro == NULL)
    {
        MinhaLista.primeiro = novo;
        MinhaLista.ultimo  = MinhaLista.primeiro;
    }
    else
    {
        MinhaLista.ultimo->proximo = novo;
        novo->anterior = MinhaLista.ultimo;
        MinhaLista.ultimo = novo;
    }
}

```



```

void cadastra()
{ system ( "cls" ); printf ("\n \n \n");
  register int i;

  struct Elemento* novo;
  novo = malloc ( 1 * sizeof (struct Elemento) );
  novo->proximo = NULL;
  novo->anterior = NULL;

  printf ( "Nome: \n" );
  fflush ( stdin ); gets ( novo->nome );

  printf ( "Rua: \n" );
  fflush ( stdin ); gets ( novo->rua );

  printf ( "Cidade: \n" );
  fflush ( stdin ); gets ( novo->cidade );

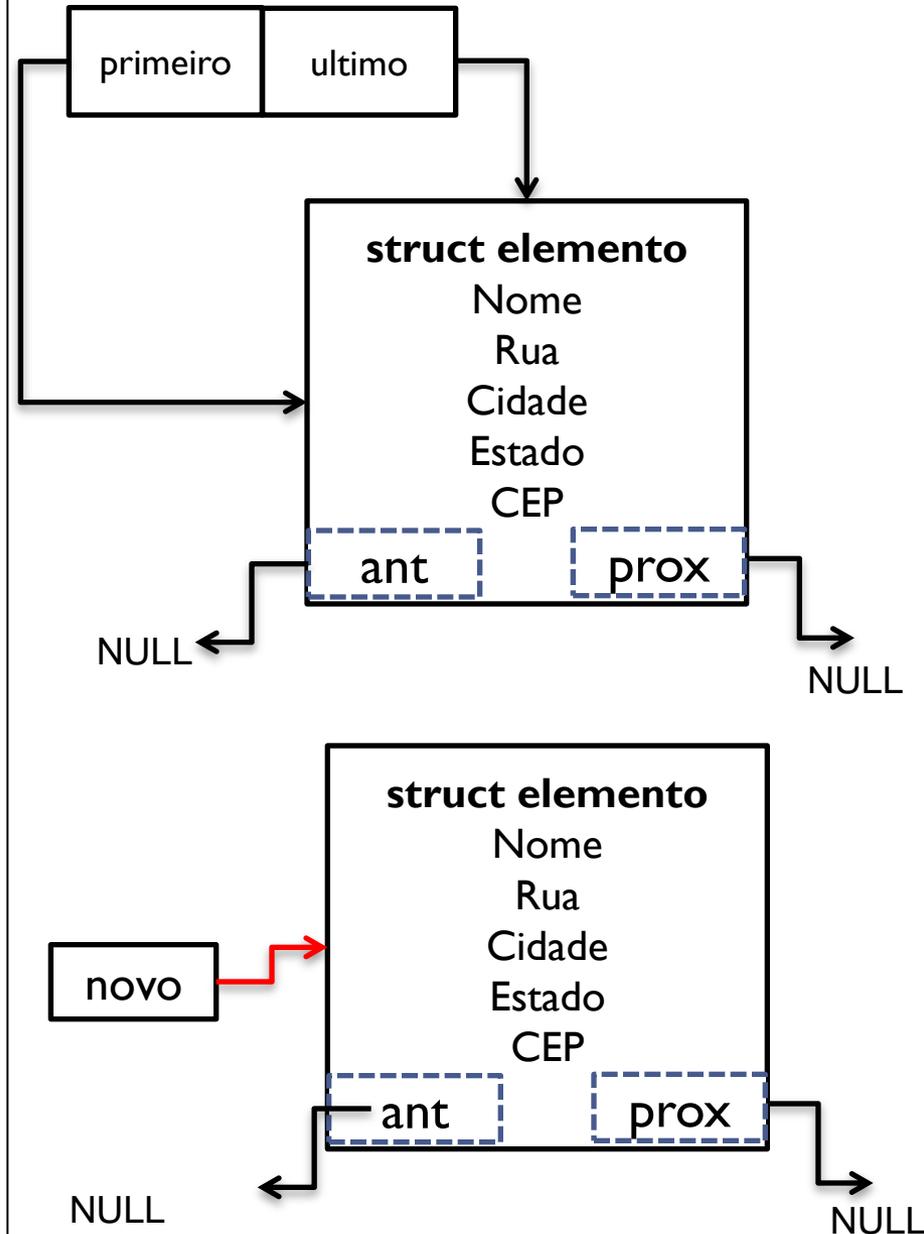
  printf ( "Estado: \n" );
  fflush ( stdin ); gets ( novo->estado );

  printf ( "CEP: \n" );
  fflush ( stdin ); gets ( novo->cep );

  if ( NULL == MinhaLista.primeiro )
  {
    MinhaLista.primeiro = novo;
    MinhaLista.ultimo = MinhaLista.primeiro;
  }
  else
  {
    MinhaLista.ultimo->proximo = novo;
    novo->anterior = MinhaLista.ultimo;
    MinhaLista.ultimo = novo;
  }
}

```

## Minha Lista

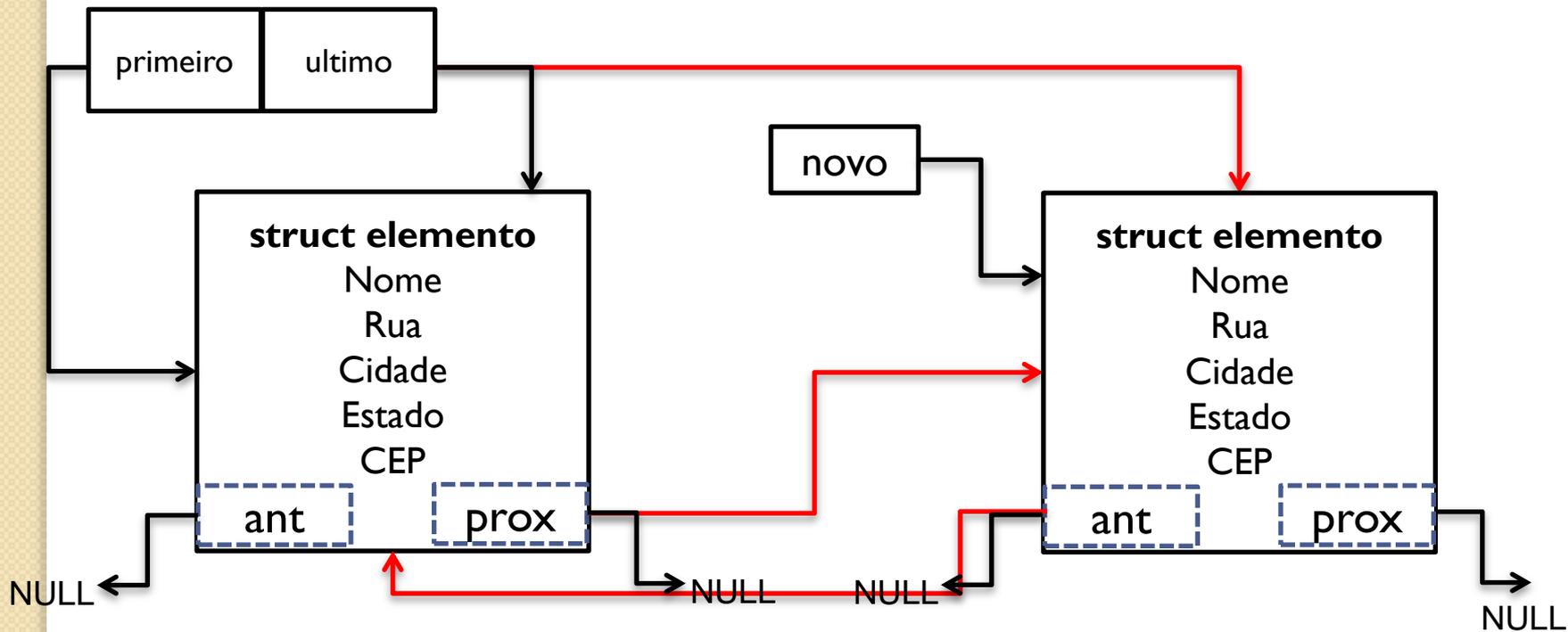


```

void cadastra()
{
    if ( NULL == MinhaLista.primeiro )
    {
        MinhaLista.primeiro = novo;
        MinhaLista.ultimo  = MinhaLista.primeiro;
    }
    else
    {
        MinhaLista.ultimo->proximo = novo;
        novo->anterior = MinhaLista.ultimo;
        MinhaLista.ultimo = novo;
    }
}

```

### Minha Lista



# ListaEncadeada.c

```
#include "ListaEncadeada.h"
```

```
struct Lista MinhaLista;
```

```
void inicia_lista ()
```

```
{
```

```
    MinhaLista.primeiro = NULL;
```

```
    MinhaLista.ultimo = NULL;
```

```
}
```

```
char menu ()
```

```
{
```

```
    printf ("\n\n\n");
```

```
    char opcao;
```

```
    printf ( "(C)adastrar. \n" );
```

```
    printf ( "(M)ostr. \n" );
```

```
    printf ( "Mostrar (R)eversamente. \n" );
```

```
    printf ( "(T)erminar. \n" );
```

```
    fflush ( stdin );
```

```
    scanf ( "%c", &opcao );
```

```
    return opcao;
```

```
}
```

```
void cadastra()
```

```
{ system ( "cls" ); printf ("\n\n\n");
```

```
register int i;
```

```
struct Elemento* novo;
```

```
novo = malloc ( 1 * sizeof (struct Elemento) );
```

```
novo->proximo = NULL;
```

```
novo->anterior = NULL;
```

```
printf ( "Nome: \n" );
```

```
fflush ( stdin ); gets ( novo->nome );
```

```
printf ( "Rua: \n" );
```

```
fflush ( stdin ); gets ( novo->rua );
```

```
printf ( "Cidade: \n" );
```

```
fflush ( stdin ); gets ( novo->cidade );
```

```
printf ( "Estado: \n" );
```

```
fflush ( stdin ); gets ( novo->estado );
```

```
printf ( "CEP: \n" );
```

```
fflush ( stdin ); gets ( novo->cep );
```

```
if ( NULL == MinhaLista.primeiro )
```

```
{
```

```
    MinhaLista.primeiro = novo;
```

```
    MinhaLista.ultimo = MinhaLista.primeiro;
```

```
}
```

```
else
```

```
{
```

```
    MinhaLista.ultimo->proximo = novo;
```

```
    novo->anterior = MinhaLista.ultimo;
```

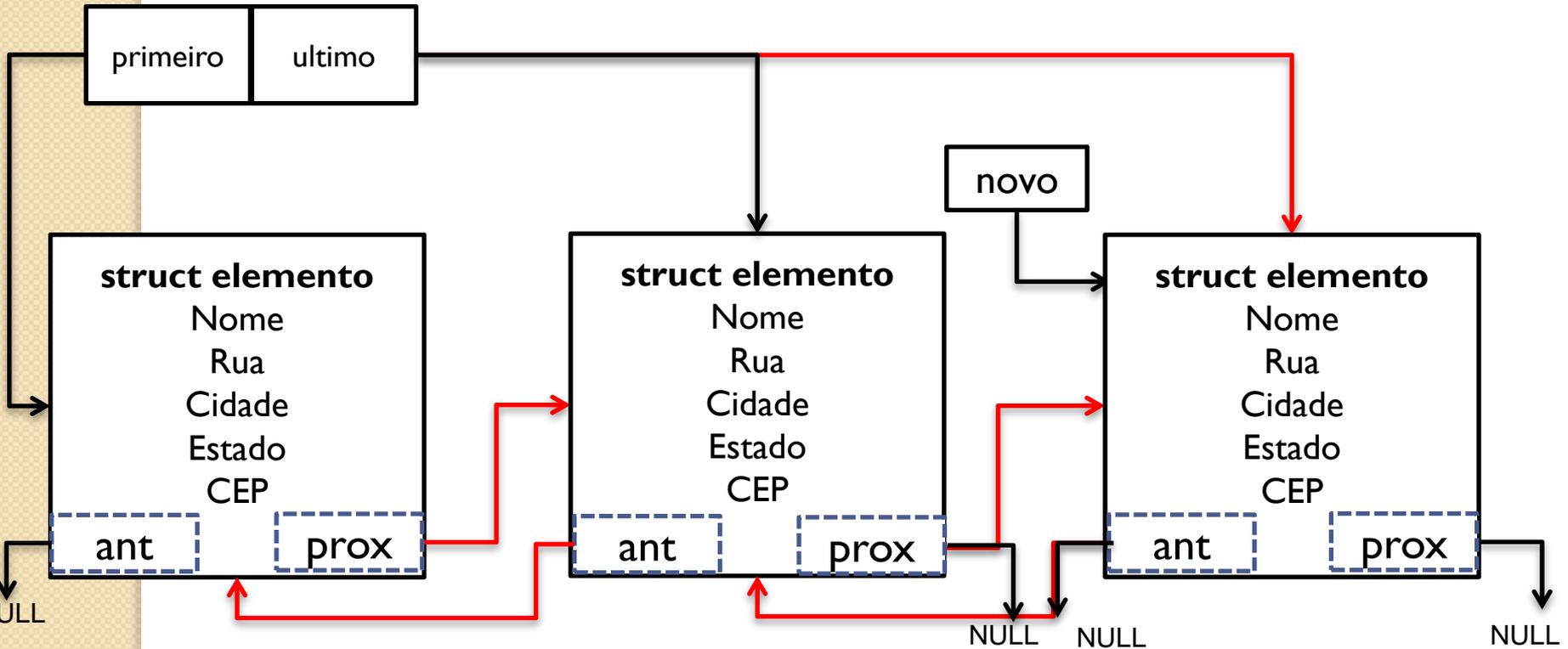
```
    MinhaLista.ultimo = novo;
```

```
}
```

```
}
```

Adicionar mais um elemento na lista:

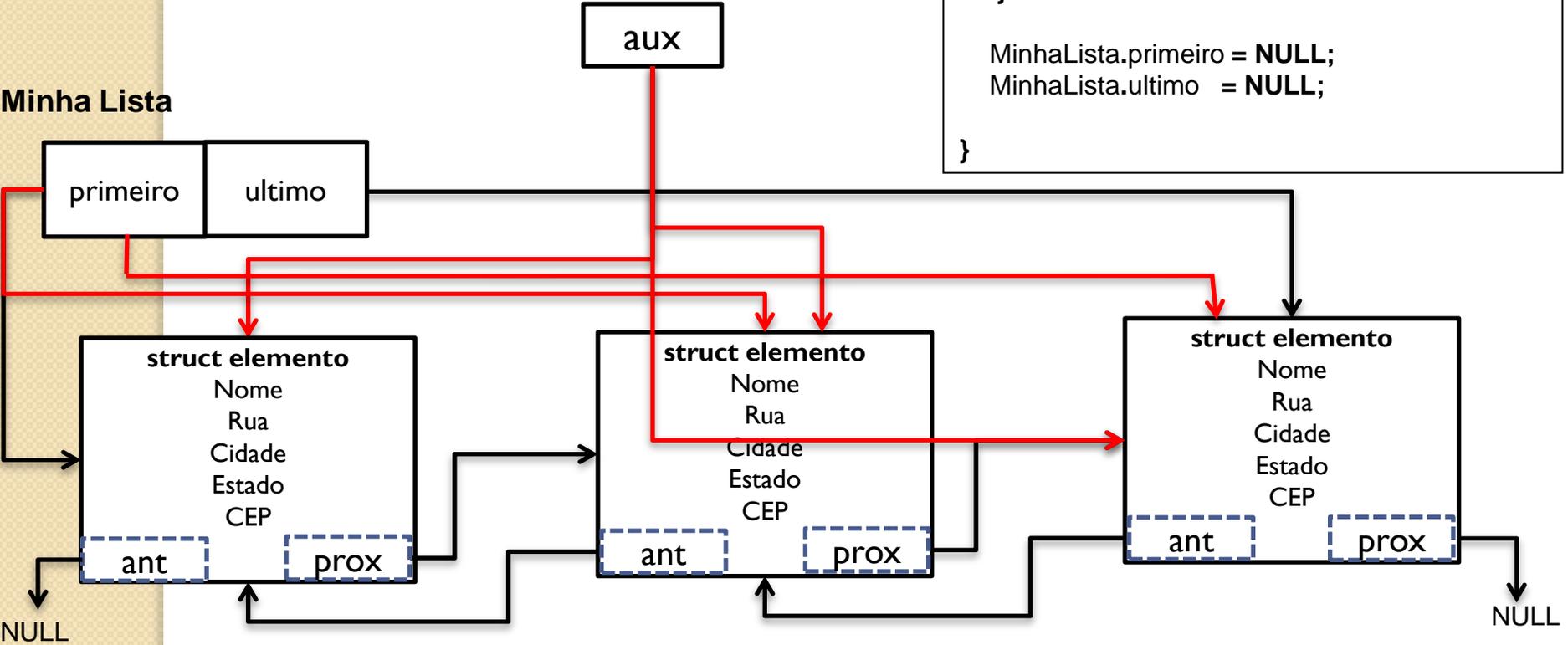
Minha Lista



# ListaEncadeada.c

```
void limpaLista ()  
{  
    struct Elemento* aux;  
  
    aux = primeiro;  
  
    while ( aux != NULL )  
    {  
  
        primeiro = primeiro->proximo;  
  
        free ( aux ) ;  
  
        aux = primeiro;  
  
    }  
  
    MinhaLista.primeiro = NULL;  
    MinhaLista.ultimo = NULL;  
  
}
```

## Minha Lista



# ListaEncadeada.c

```
void mostra()
{

    system ( "cls" );

    printf ("\n \n \n");

    struct Elemento* aux;

    aux = MinhaLista.primeiro;

    while ( aux != NULL )
    {

        printf ( "%s \n", aux->nome );
        printf ( "%s \n", aux->rua );
        printf ( "%s \n", aux->cidade );
        printf ( "%s \n", aux->estado );
        printf ( "%s \n", aux->cep );
        printf ( "\n");

        aux = aux->proximo;

    }
}
```

```
void mostraReverso()
{

    system ( "cls" );

    printf ("\n \n \n");

    struct Elemento* aux;

    aux = MinhaLista.ultimo;

    while ( aux != NULL )
    {

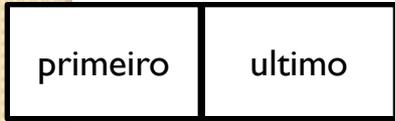
        printf ( "%s \n", aux->nome );
        printf ( "%s \n", aux->rua );
        printf ( "%s \n", aux->cidade );
        printf ( "%s \n", aux->estado );
        printf ( "%s \n", aux->cep );
        printf ( "\n");

        aux = aux->anterior;

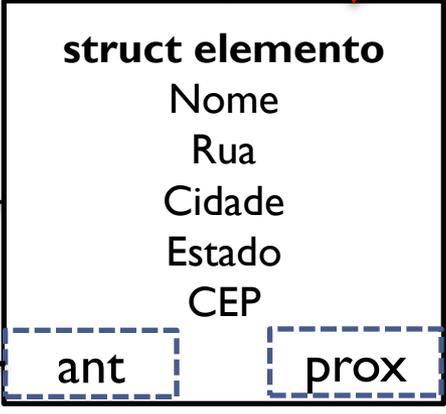
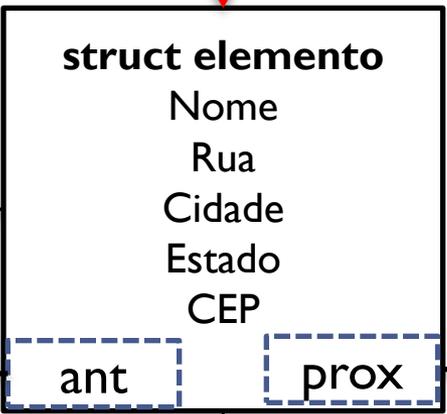
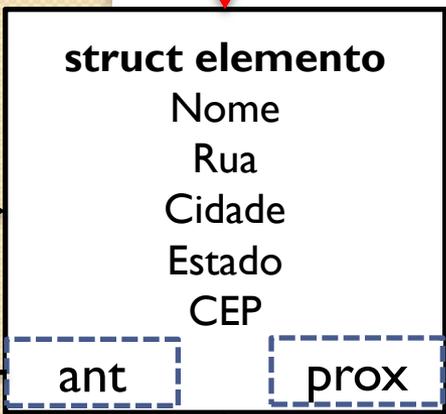
    }
}
```

mostraReverso();

Minha Lista



aux



aux = aux->anterior;

## main.c

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

#include "ListaEncadeada.h"

int main (int argc, char *argv[])
{
    char escolha;
    inicia_lista ();

    for (;;)
    {
        escolha = menu ();
        switch ( escolha )
        {
            case 'c':
            case 'C': { cadastra(); } break;

            case 'm':
            case 'M': { mostra(); } break;

            case 'r':
            case 'R': { mostraReverso(); } break;

            case 't':
            case 'T': { limpaLista(); system("Pause"); exit(0); } break;

            default : { printf("Opcao invalida. \n"); }
        }

        printf ("\n \n \n");
    }
    system("PAUSE");
    return 0;
}
```

# Referências Bibliográficas

- Baseado nos slides do Professor Jean Simão disponível em:  
<http://www.pessoal.utfpr.edu.br/jeansimao/FundamentosI/FundamentosI.htm>
- **SHILDT**, H. C, Completo e Total, 3a edição, rev. e atual. Ed. Makron. São Paulo, c1997.