Fundamentos de Programação 1

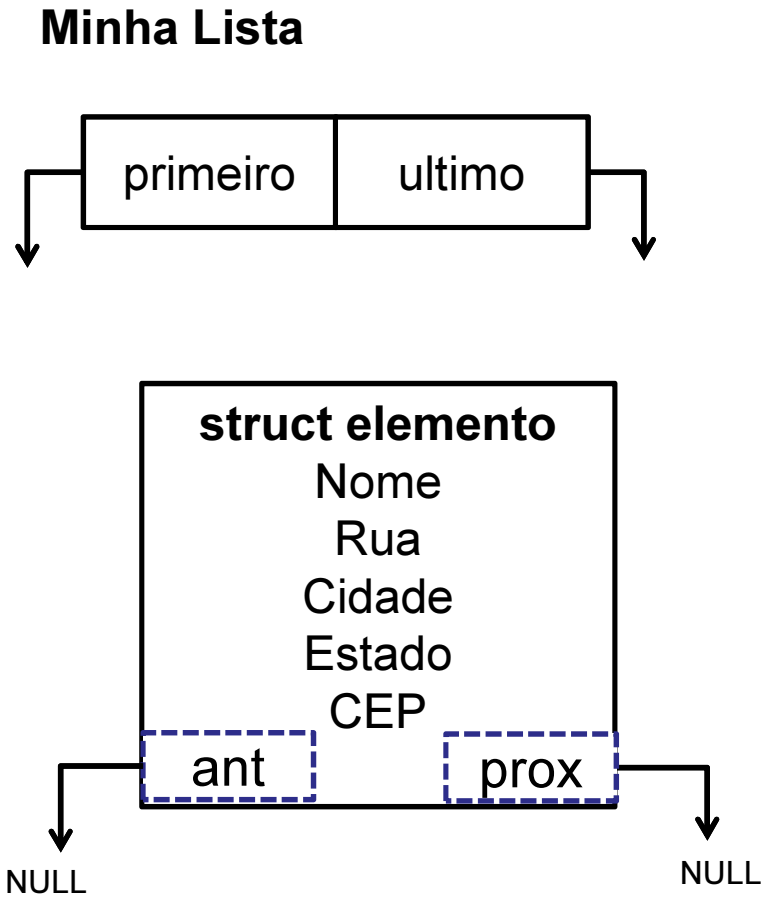
Linguagem C
“Lista Duplamente Encadeada
-
Projeto com vários Arquivos”.

Slides 22

Prof.ª Fabiany e Prof. SIMÃO

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
```



Projeto em Dev C++

ListaEncadeada.c

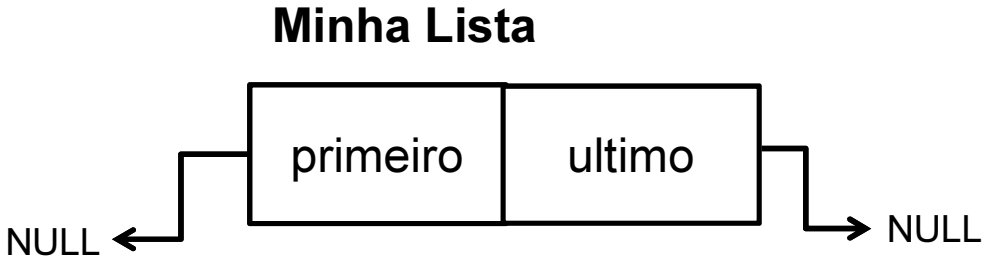
```
#include "ListaEncadeada.h"

struct Lista  MinhaLista;

void inicia_lista ()
{

    MinhaLista.primeiro = NULL;

    MinhaLista.ultimo   = NULL;

}
```

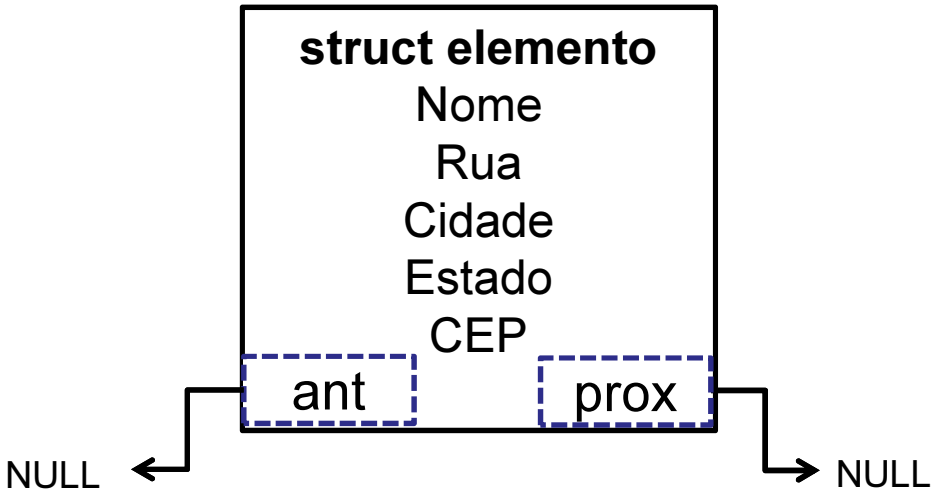
```
char menu ()
{
   printf ("\n \n \n");
   char opcao;

   printf ( "(C)adastrar. \n" );
   printf ( "(M)ostrar.   \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;
    }
}
```

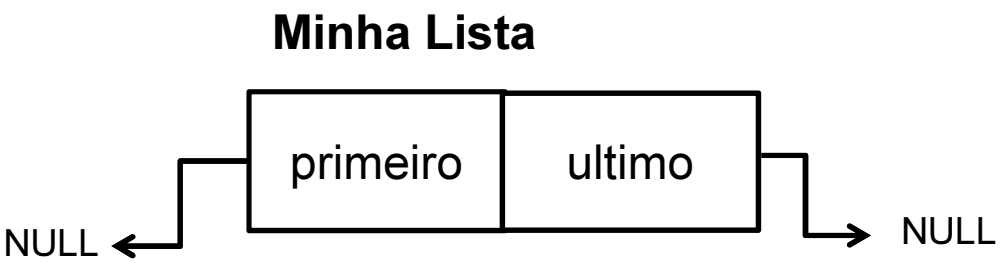
ListaEncadeada.c

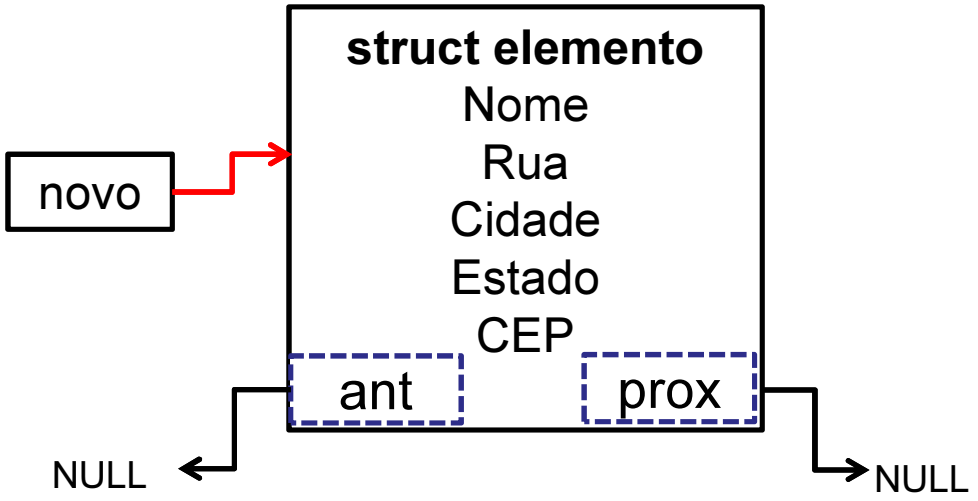
```
#include "ListaEncadeada.h"

struct Lista  MinhaLista;

void inicia_lista ()
{

    MinhaLista.primeiro = NULL;

    MinhaLista.ultimo   = NULL;

}
```



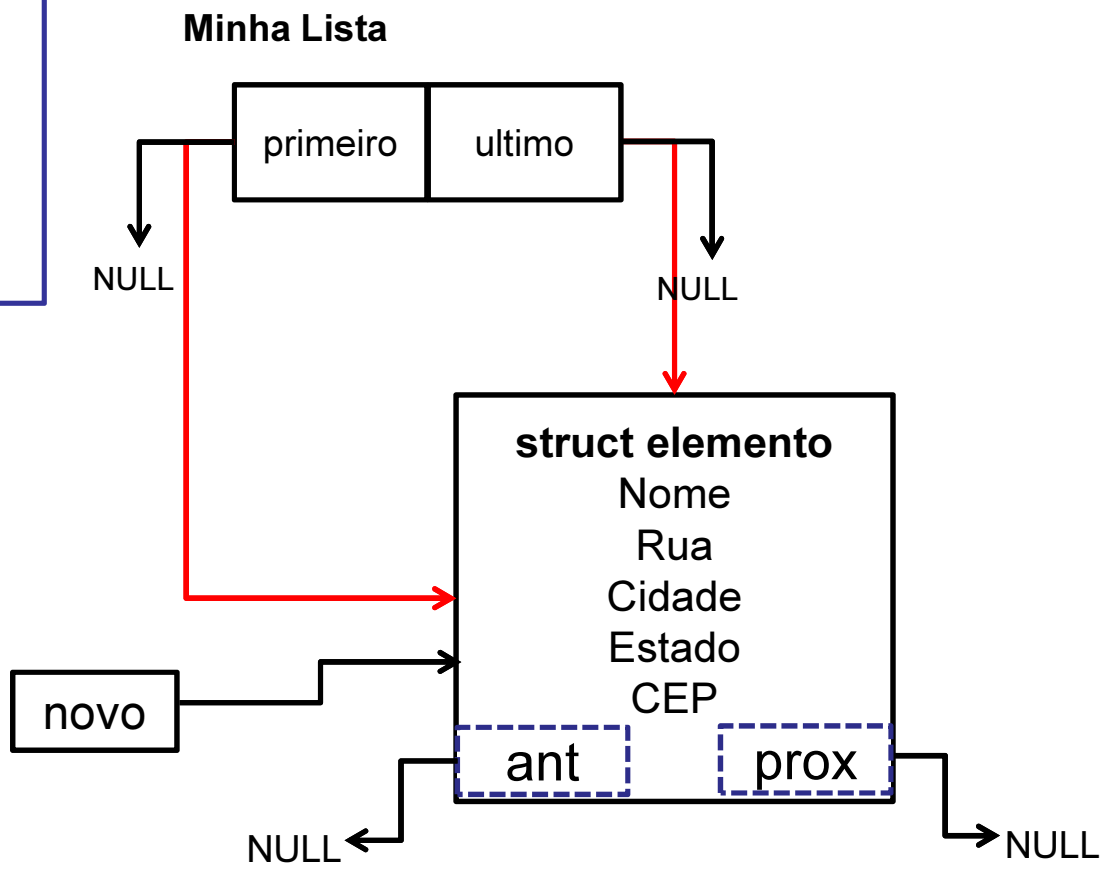


```
void cadastra()
{   system ( "cls" );    printf ("\n \n \n");
    register int i;

    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;
    }
}
```

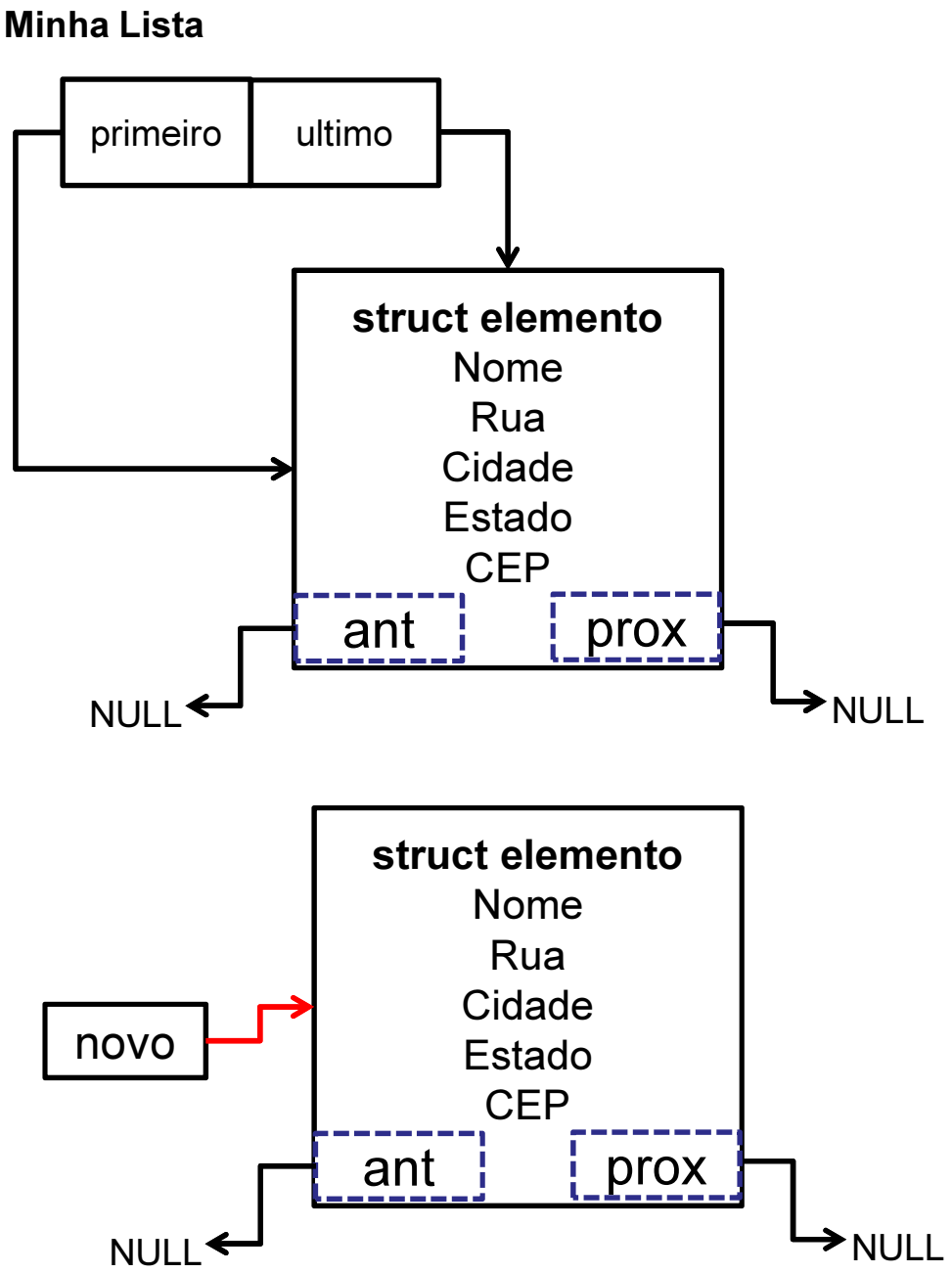




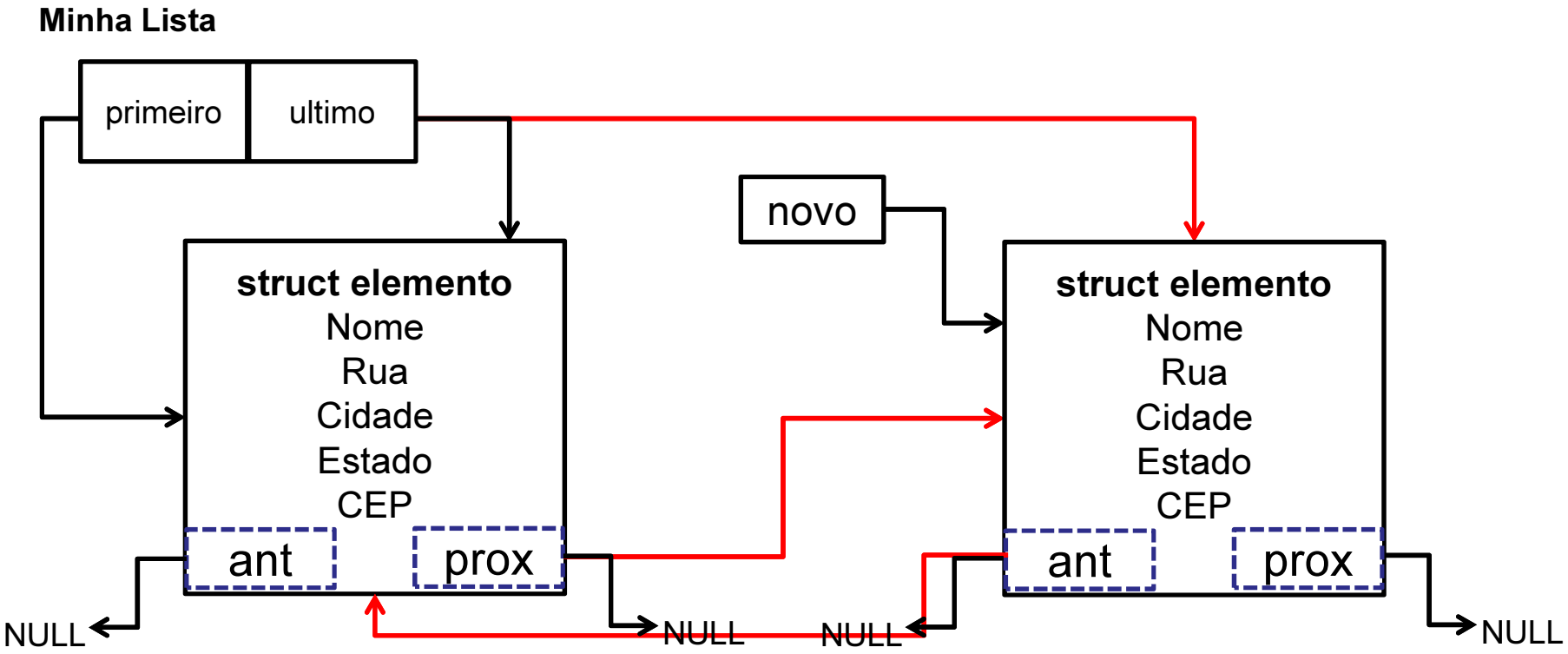
```
void cadastra()
{
    if ( MinhaLista.primeiro  ==  NULL)
    {
      MinhaLista.primeiro = novo;
      MinhaLista.ultimo   = MinhaLista.primeiro;
    }
    else
    {
      MinhaLista.ultimo->proximo = novo;
      novo->anterior  = MinhaLista.ultimo;
      MinhaLista.ultimo = novo;
    }
}
```



```
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;
    }
}
```



```
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;
    }
}
```



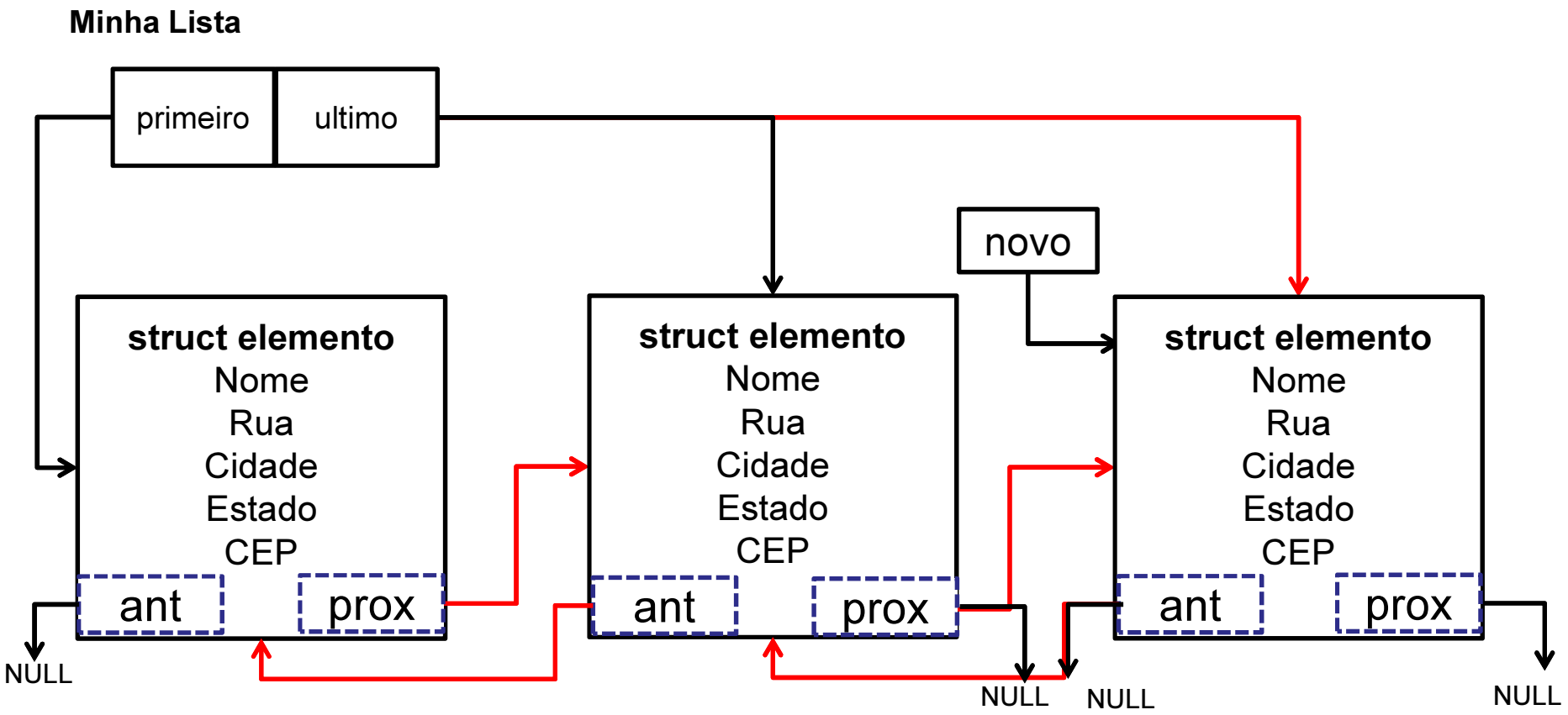
ListaEncadeada.c

```c
#include "ListaEncadeada.h"

struct Lista  MinhaLista;

void inicia_lista ()
{

    MinhaLista.primeiro = NULL;

    MinhaLista.ultimo   = NULL;

}
```

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

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

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

   return opcao;
}
```

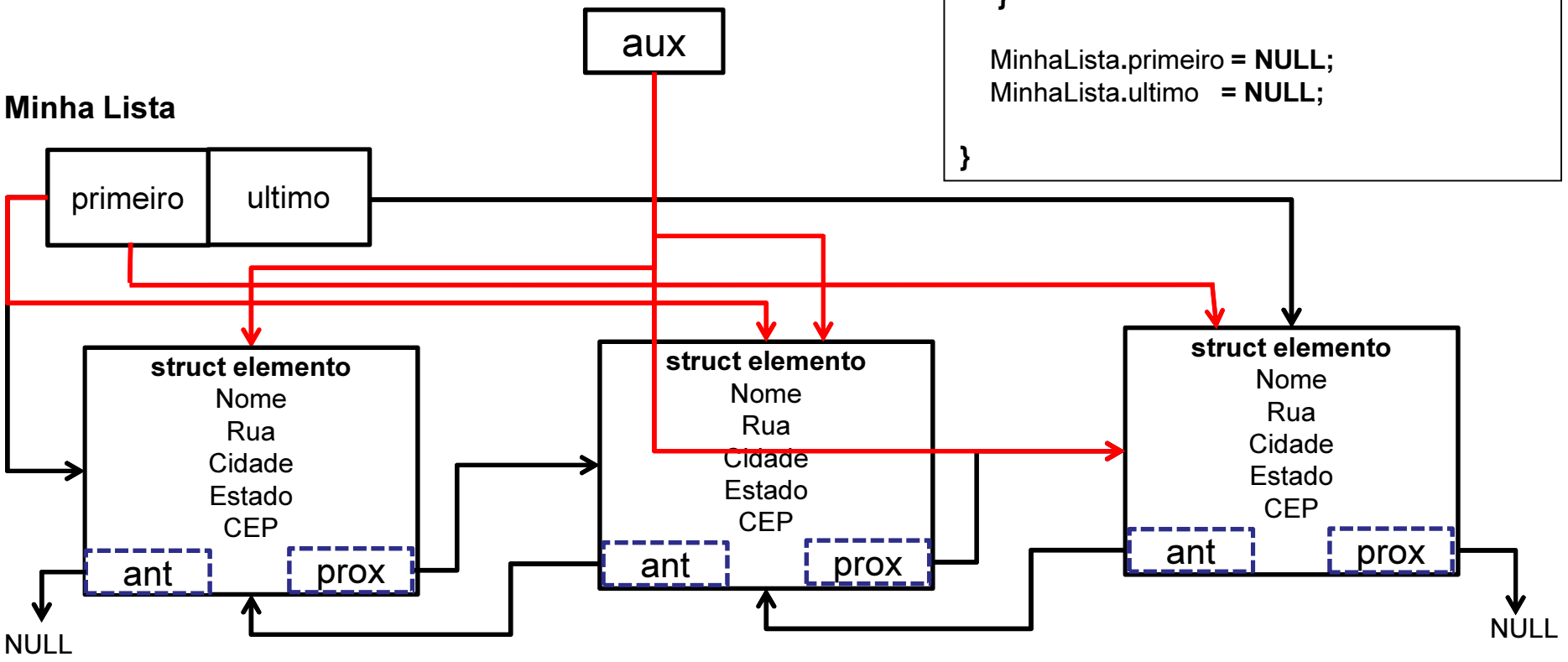
```c
void cadastra()
{   system ( "cls" );    printf ("\n \n \n");
    register int i;

    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:



ListaEncadeada.c

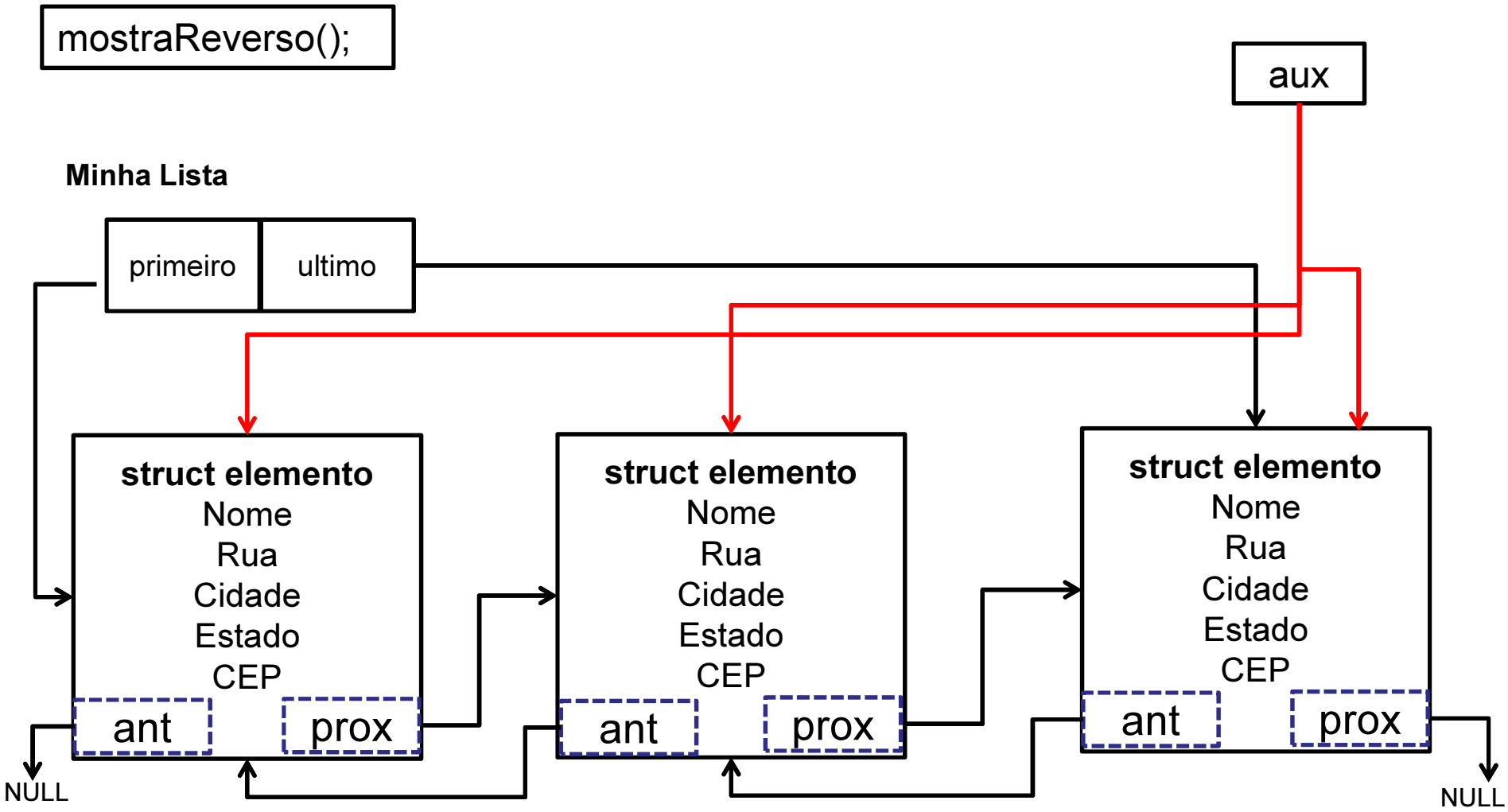
```
void limpaLista ()
{
    struct Elemento* aux;

    aux = primeiro;

    while ( aux != NULL )
    {

        primeiro = primeiro->proximo;

        free ( aux ) ;

        aux = primeiro;

    }

   MinhaLista.primeiro = NULL;
   MinhaLista.ultimo   = NULL;

}
```



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;

    }
}
```



aux = aux->anterior;

main.c

```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;
}
```

Exercícios

- Re-elaborar a solução anterior sem utilizar variáveis ou ponteiros globais.
- Elaborar uma função para encontrar os dados de um elemento da lista dado o *valor* do campo nome.
- Elaborar uma função que permita eliminar um elemento da lista dado o *valor* do campo nome.
- Elaborar um solução que permita gravar e recuperar as informações da lista em arquivo.