Fundamentos de Programação 1

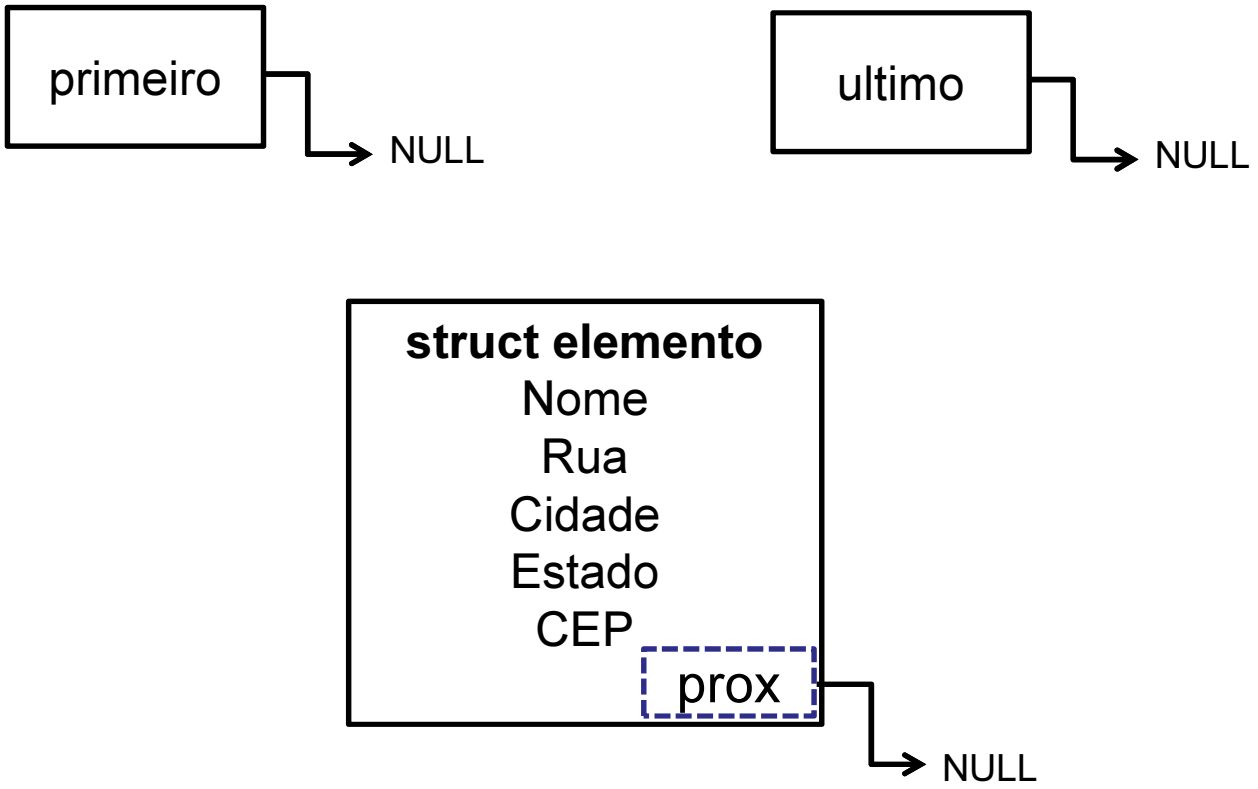
Linguagem C
“Lista Encadeada”.

Slides 21

Prof.ª Fabiany e Prof. SIMÃO

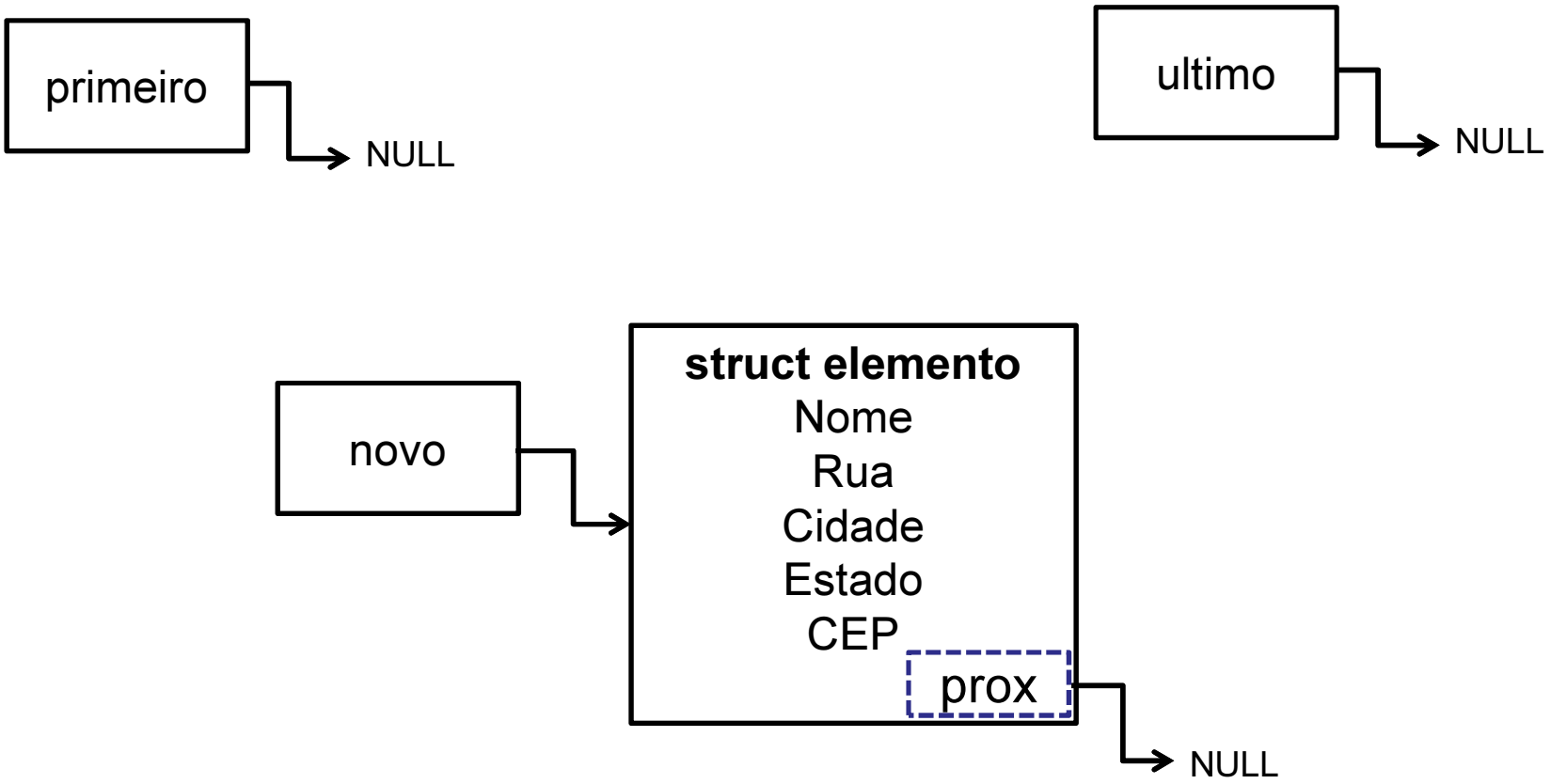
```c
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>

#define TAM 2

struct Elemento
{

      char nome [100];
      char rua [100];
      char cidade [100];
      char estado [2];
      char cep [10];

      struct Elemento  *proximo;

};

struct Elemento  *primeiro;
struct Elemento  *ultimo;

char menu ();

void inicia_lista ();

void cadastra ();

void mostra ();

void salva ();

void carrega ();

void limpaLista ();
```

```c
int main()
{
   char escolha;
   inicia_lista ();

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

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

         case 't' :
         case 'T' : {
                        limpaLista ();
                        exit ( 0 );
                    } break;
         default : {  printf ( "Opcao invalida. \n" );  }
      }

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

   system ( "Pause" );
   return 0;
}

void inicia_lista ()
{
    primeiro = NULL;
    ultimo   = NULL;
}
```


primeiro
NULL
ultimo
NULL
struct elemento
Nome
Rua
Cidade
Estado
CEP
prox
NULL

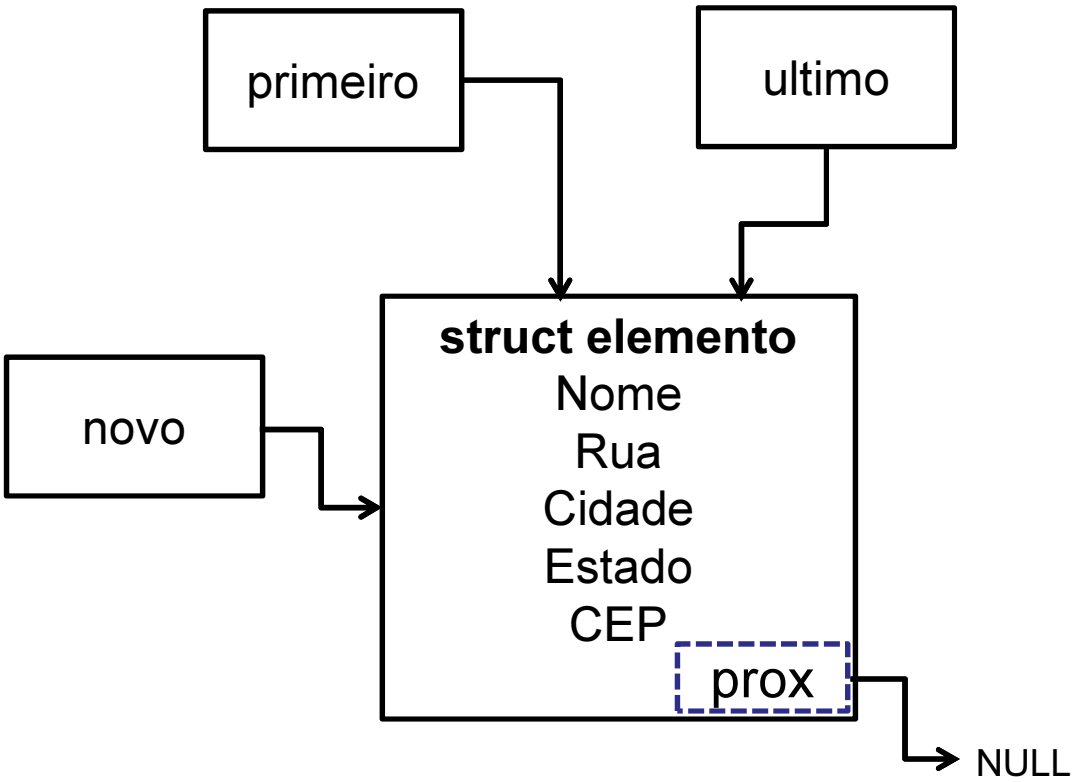
```
void cadastra ()
{
    system ( "cls" );
    printf ( "\n \n \n" );

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

    novo->proximo = 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 == primeiro )
    {
      primeiro  = novo;
      ultimo    = primeiro;
    }
    else
    {
      ultimo->proximo = novo;
      ultimo = novo;
    }
}
```

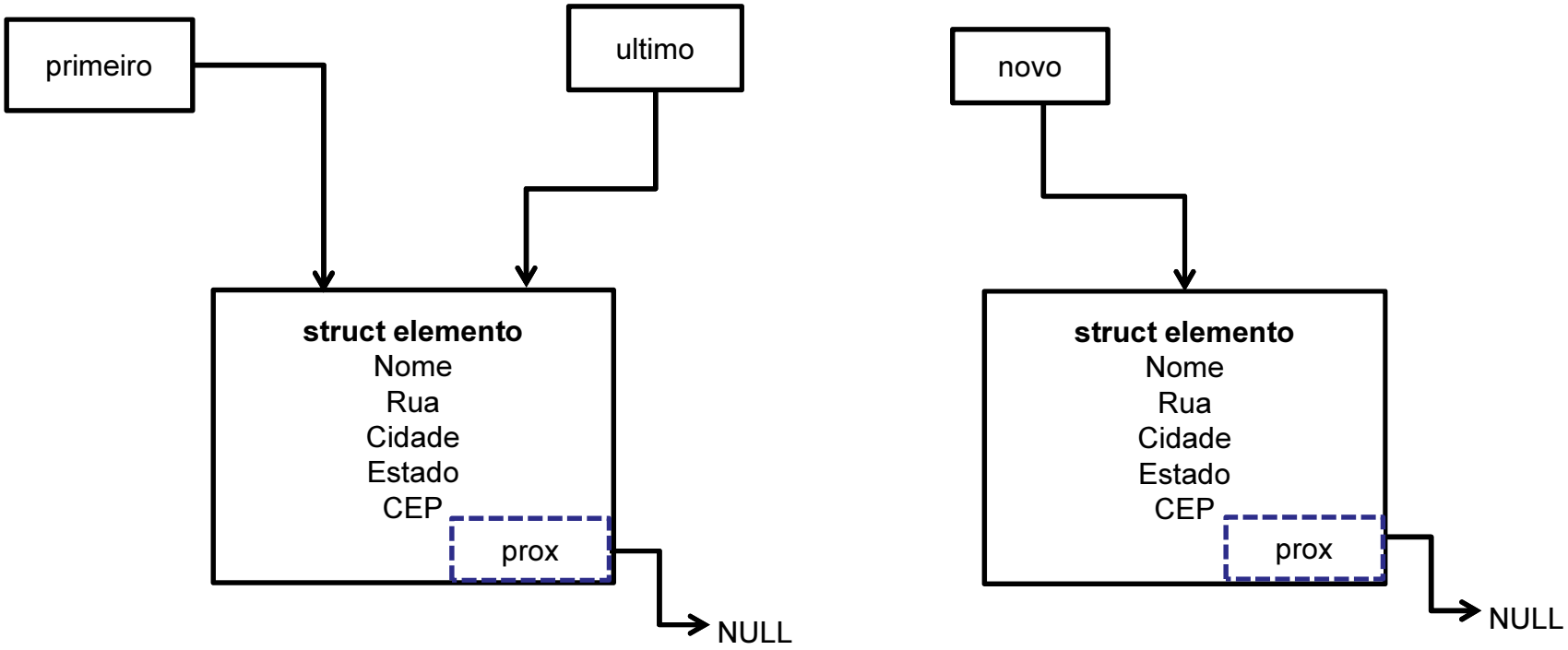
```
void mostra ()
{

    system ( "cls" );

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

    struct Elemento* aux;

    aux = 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;

    }

}
```

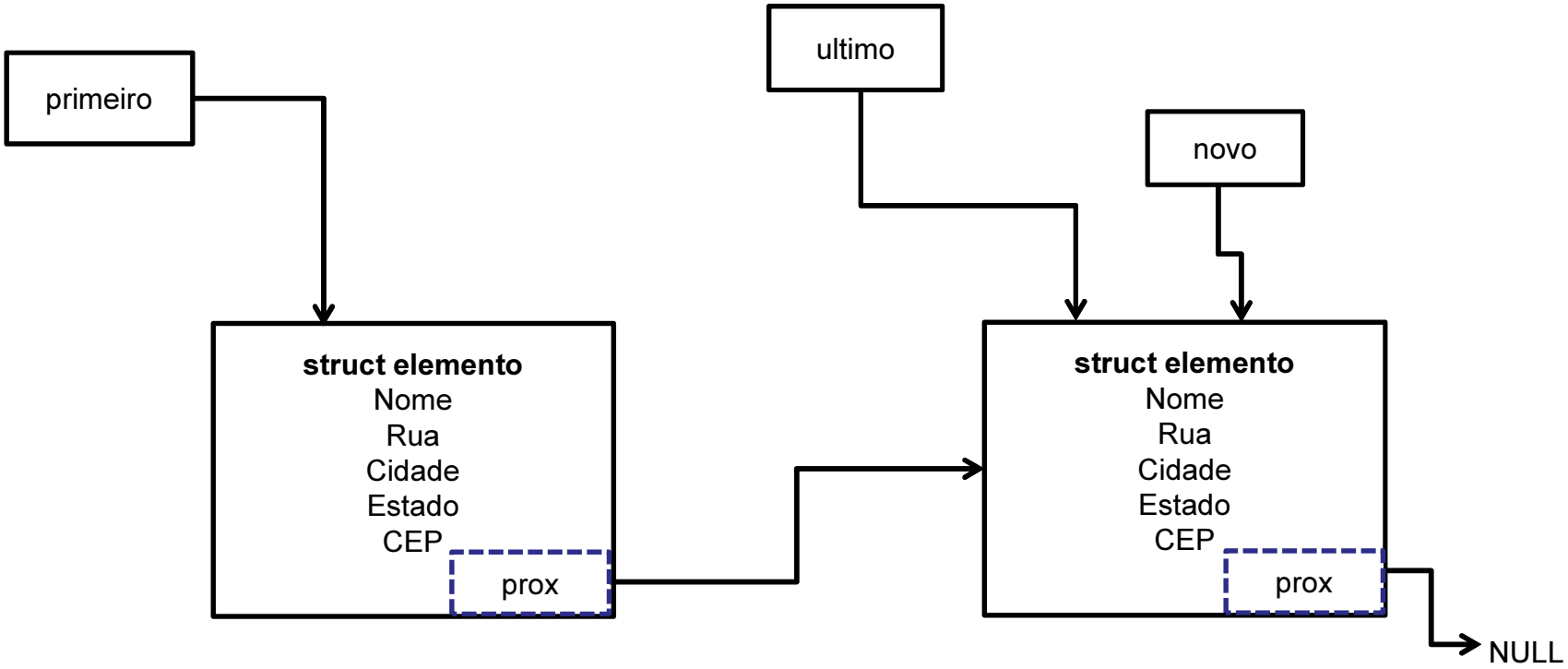

primeiro
NULL
ultimo
NULL
novo
struct elemento
Nome
Rua
Cidade
Estado
CEP
prox
NULL

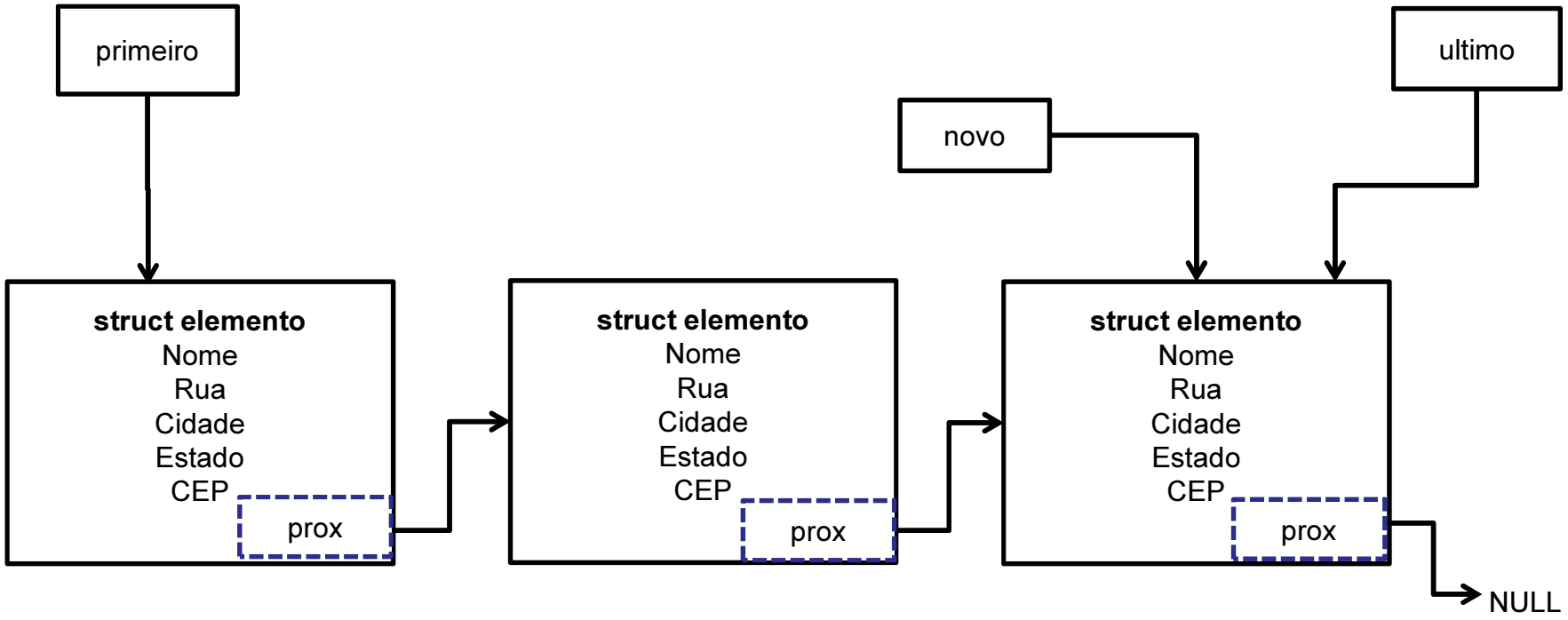
```
void cadastra ()
{
    system ( "cls" );
    printf ( "\n \n \n" );

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

    novo->proximo = 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 == primeiro )
    {
      primeiro  = novo;
      ultimo    = primeiro;
    }
    else
    {
      ultimo->proximo = novo;
      ultimo = novo;
    }
}
```

```
void mostra ()
{

    system ( "cls" );

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

    struct Elemento* aux;

    aux = 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;

    }

}
```


primeiro
ultimo
struct elemento
Nome
Rua
Cidade
Estado
CEP
prox
novo
NULL

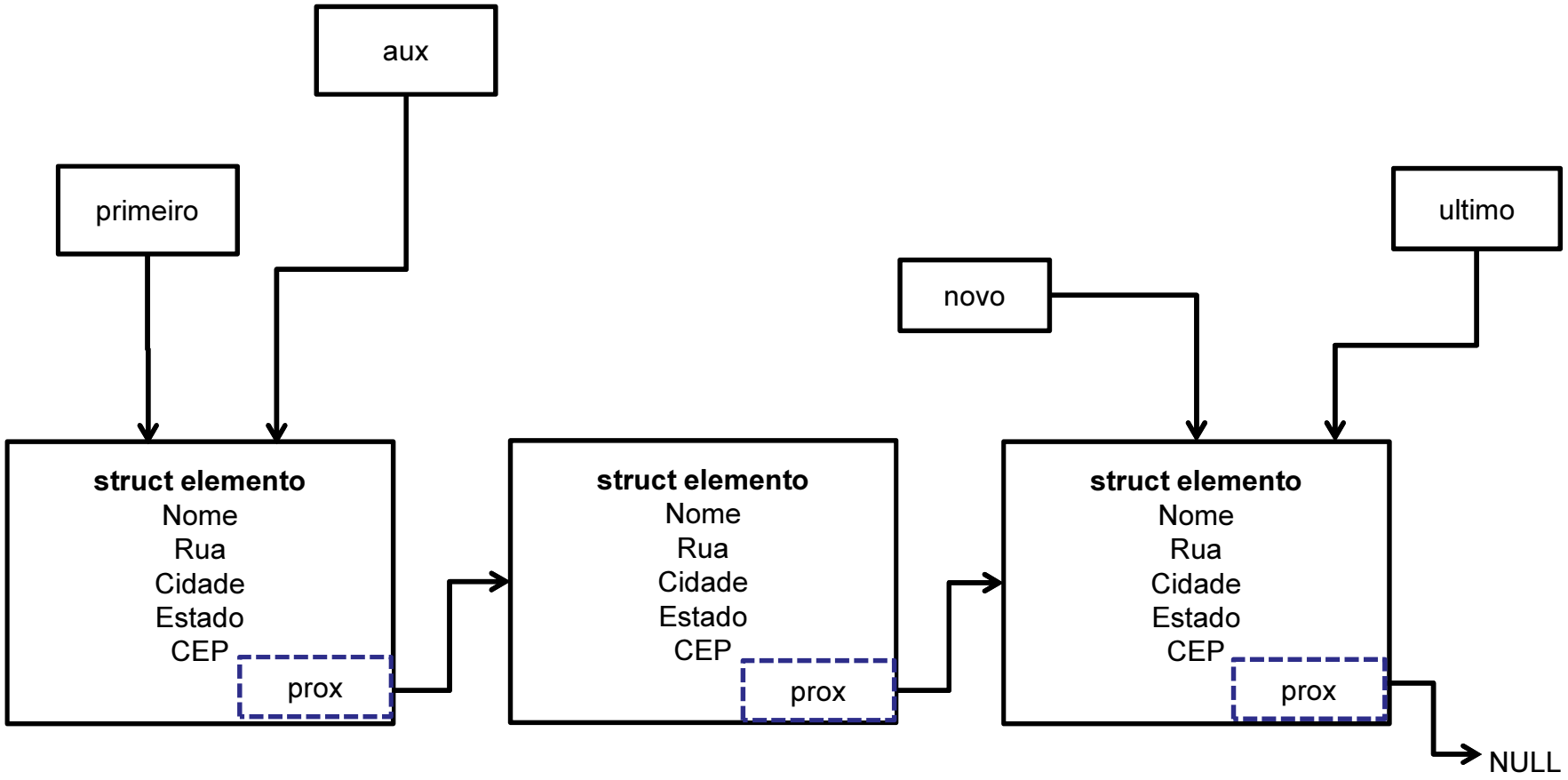

primeiro
ultimo
struct elemento
Nome
Rua
Cidade
Estado
CEP
prox
NULL
novo
struct elemento
Nome
Rua
Cidade
Estado
CEP
prox
NULL

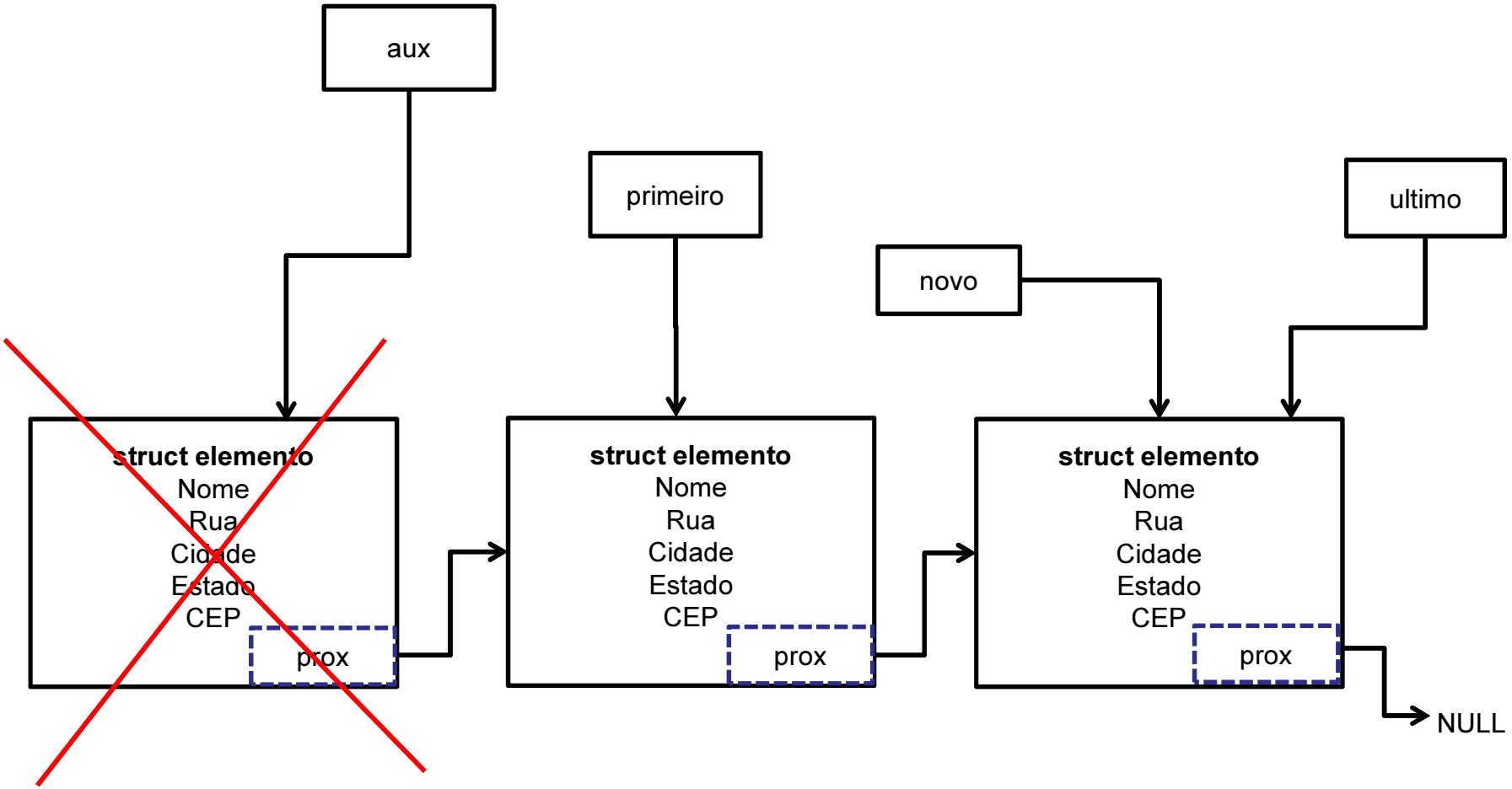

primeiro
ultimo
novo
struct elemento
Nome
Rua
Cidade
Estado
CEP
prox
struct elemento
Nome
Rua
Cidade
Estado
CEP
prox
NULL

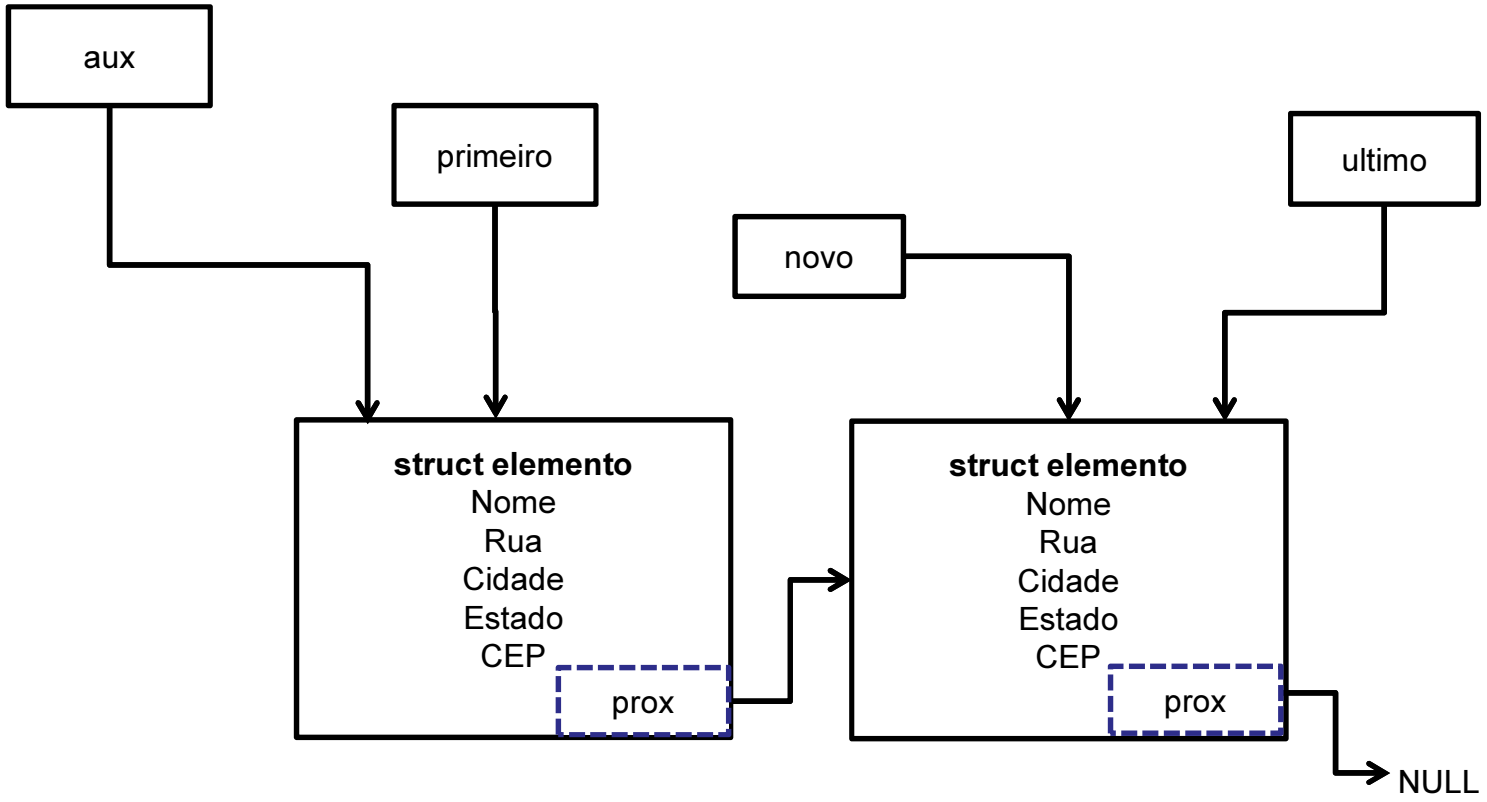

primeiro
ultimo
novo
struct elemento
Nome
Rua
Cidade
Estado
CEP
prox
struct elemento
Nome
Rua
Cidade
Estado
CEP
prox
struct elemento
Nome
Rua
Cidade
Estado
CEP
prox
NULL

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

    aux = primeiro;

    while ( aux != NULL )
    {

        primeiro = primeiro->proximo;

        free ( aux ) ;

        aux = primeiro;

    }

    primeiro = NULL;
    ultimo   = NULL;

}
```

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

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

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

     return opcao;
}
```


aux
primeiro
ultimo
novo
struct elemento
Nome
Rua
Cidade
Estado
CEP
prox
struct elemento
Nome
Rua
Cidade
Estado
CEP
prox
struct elemento
Nome
Rua
Cidade
Estado
CEP
prox
NULL


aux
primeiro
novo
ultimo
struct elemento
Nome
Rua
Cidade
Estado
CEP
prox
struct elemento
Nome
Rua
Cidade
Estado
CEP
prox
struct elemento
Nome
Rua
Cidade
Estado
CEP
prox
NULL


aux
primeiro
novo
ultimo
struct elemento
Nome
Rua
Cidade
Estado
CEP
prox
struct elemento
Nome
Rua
Cidade
Estado
CEP
prox
NULL

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 imprima a lista de elemento de trás para frente.