Lista Encadeada

Prof. Renato Novais

INF029 – Laboratório de Programação

renato@ifba.edu.br

Agenda

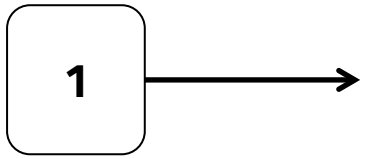
- Definições
- Exemplos

Lista Encadeada: o que é?

- Estrutura de dados linear, onde cada elemento aponta para o próximo
- Estrutura Dinâmica:
  - Não se prende a um tamanho fixo
- Quando eu devo usá-la?
  - Quando não sabemos a priori (em tempo de compilação) o tamanho do meu dado. Será criada/ incrementada/decrementada em tempo de execução

Lista Encadeada: Como implementar?

- Uso de ponteiros, para fazer a alocação dinâmica
- Alocação dinâmica?
  - Definição em tempo de execução de um variável. Alocando um espaço de memória para uso.
- Uso de struct, para representar um nó/Elemento da lista.
  - Cada nó armazena o dado de interesse e o ponteiro para o elemento seguinte.

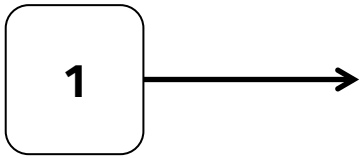


- Necessário ter um ponteiro para o início da lista

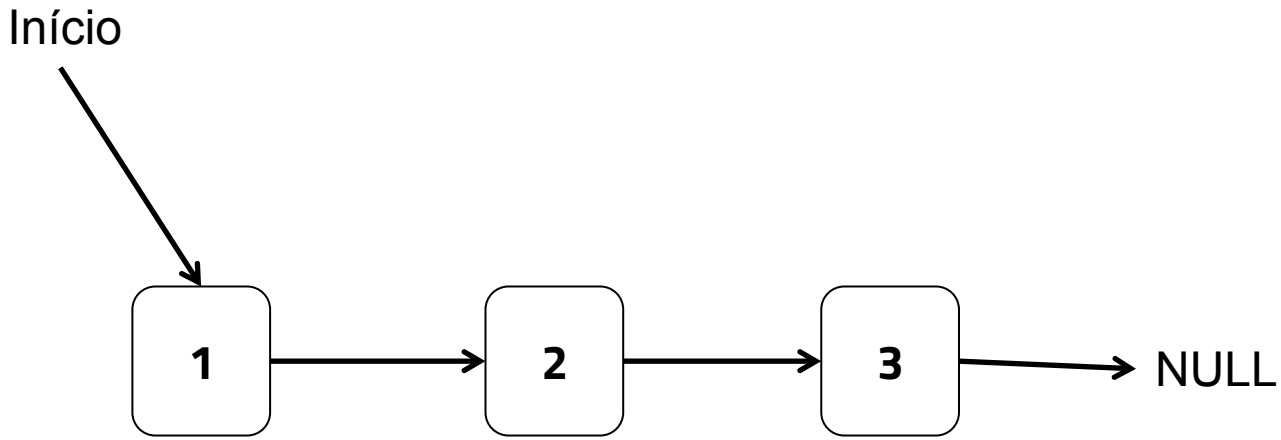
Lista Encadeada: a struct

- Nó da estrutura

```
typedef struct reg {
    int conteudo;
    struct reg *prox;
} No;
```



Lista Encadeada: representação visual

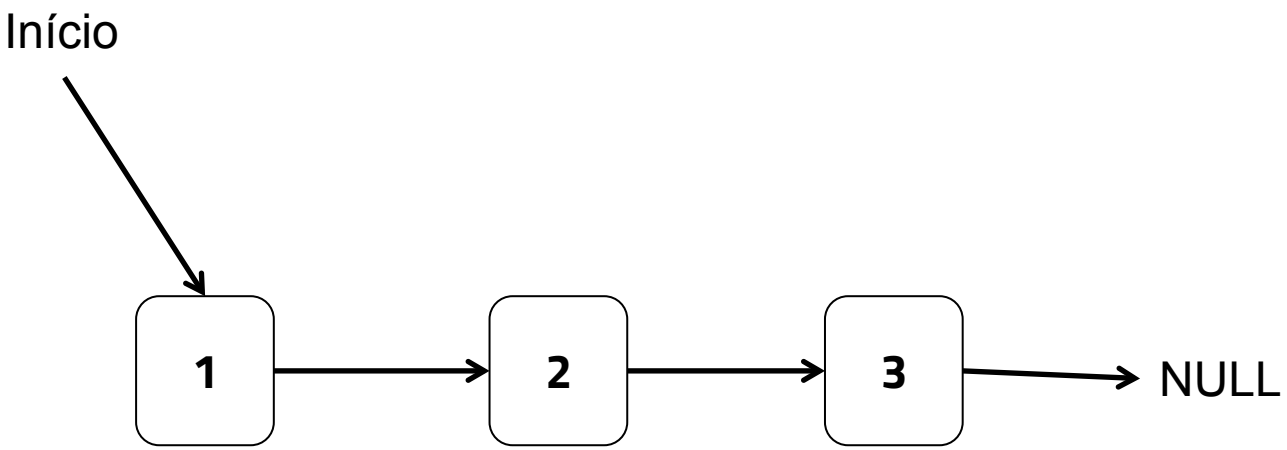


Lista Encadeada: tipos

- Lista encadeada simples
- Lista duplamente encadeada
- Lista encadeada circular

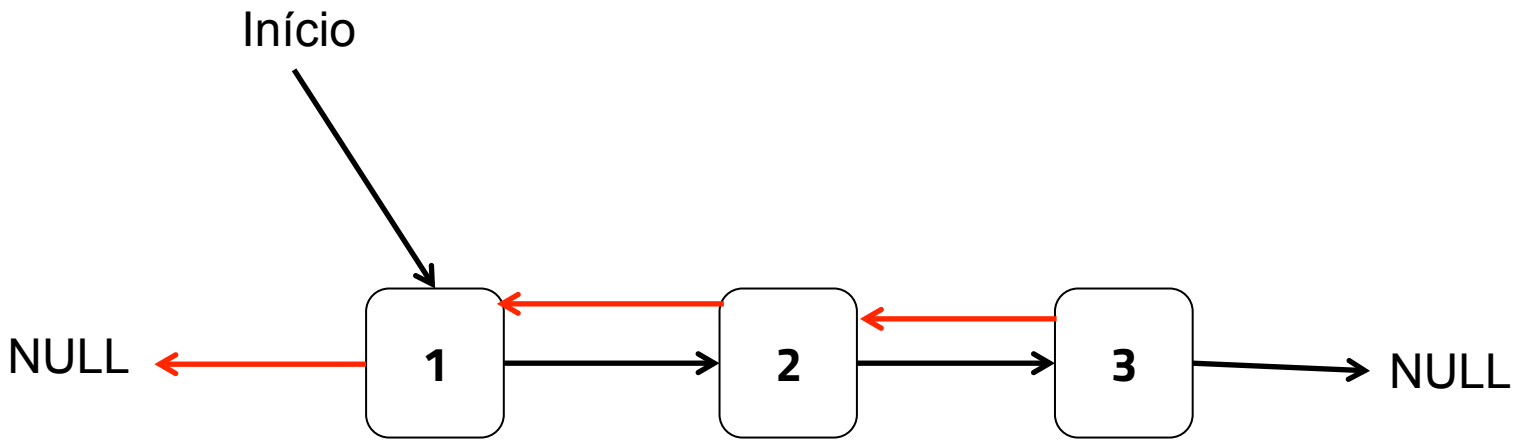
Lista Encadeada: tipos

- Lista encadeada simples



Lista Encadeada: tipos

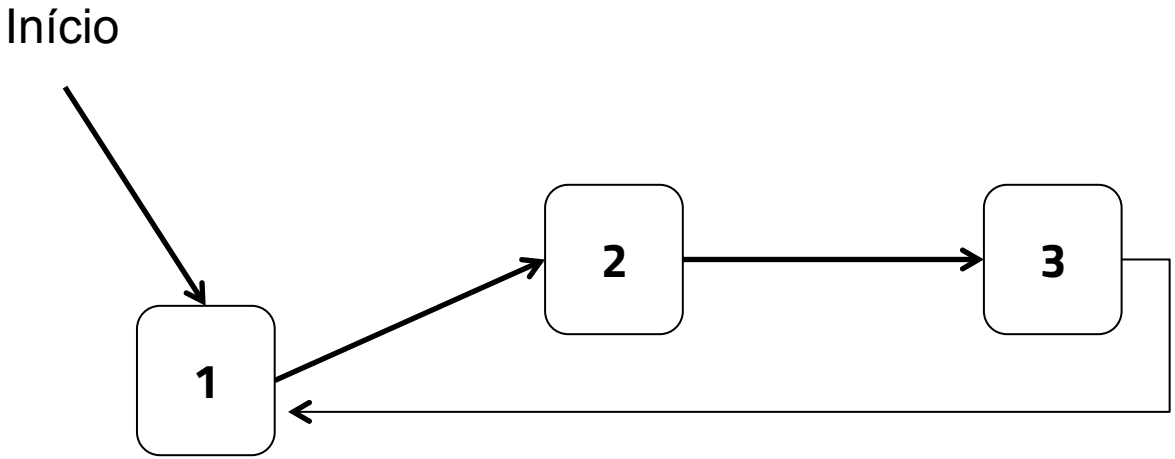
- Lista duplamente encadeada

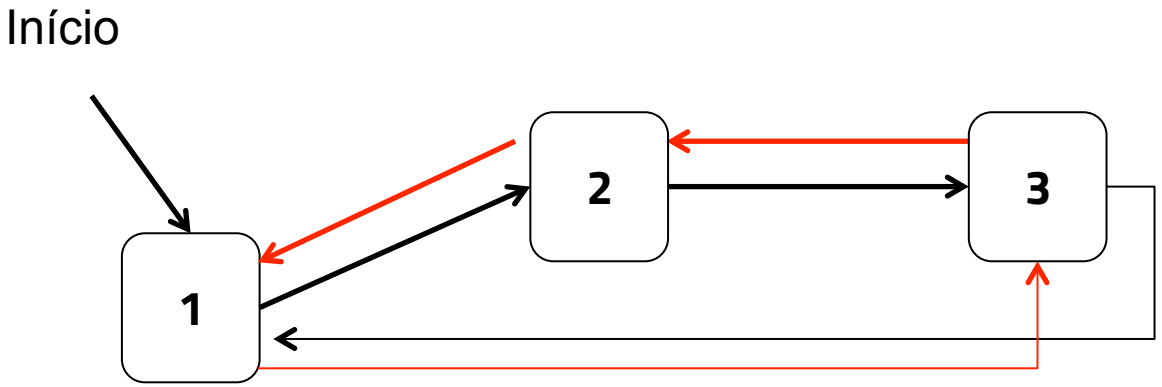



```
typedef struct reg {
    int conteudo;
    struct reg *prox;
    struct reg *anterior;
} No;
```

Lista Encadeada: tipos

- Lista encadeada circular

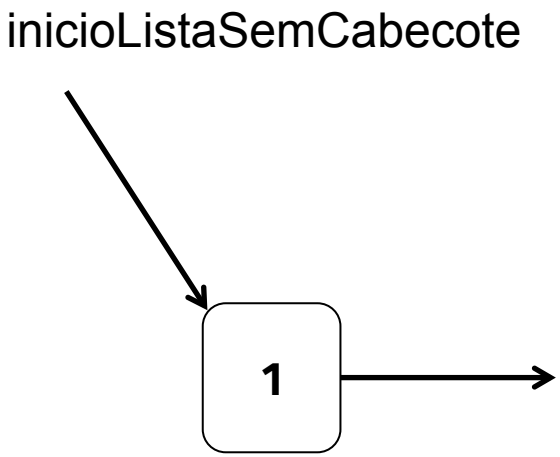




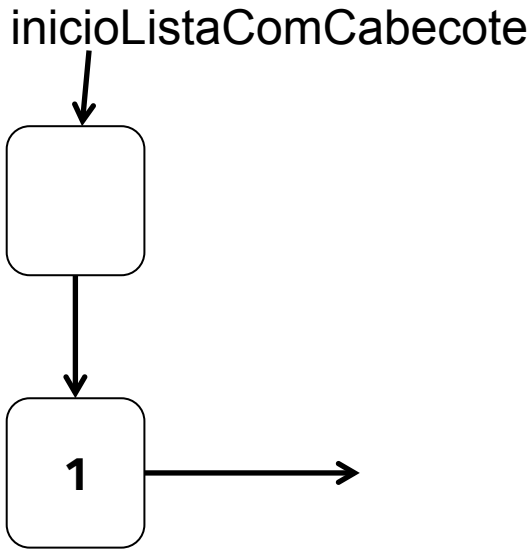
Lista Encadeada: o início

- Variável que controla o começo da lista, permitindo o acesso à mesma

Sem Cabeçote

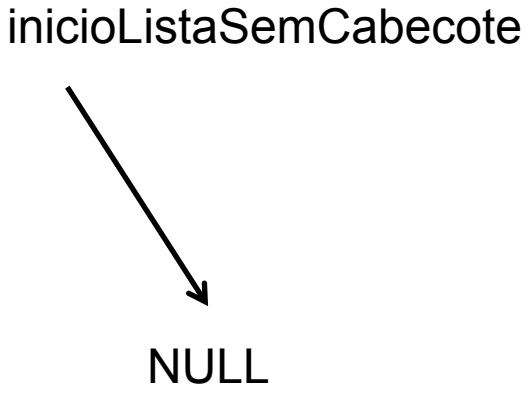


Com Cabeçote

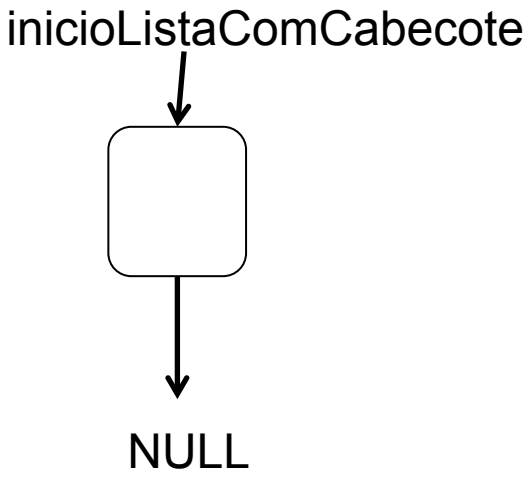


Lista Encadeada: o inicio, iniciando o programa, lista vazia

Sem Cabeçote



Com Cabeçote




```
No *inicioListaSemCabecote = NULL;

No *inicioListaComCabecote = (No*) malloc(sizeof(No));
inicioListaComCabecote->prox = NULL;
```

Lista Encadeada: operações

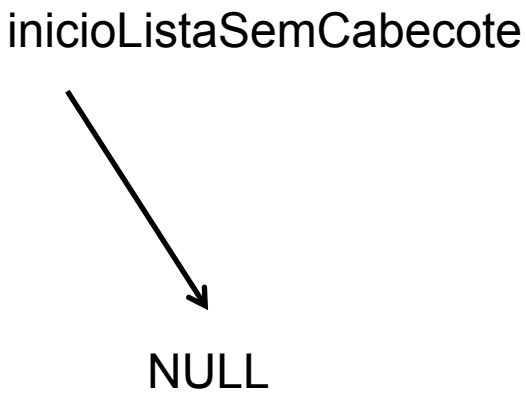
- Inserir elemento
- Excluir elemento
- Listar elementos
- Buscar elemento
- Etc, etc, etc.

Lista Encadeada: operações

- Inserir elemento
- Listar elementos
- Excluir elemento
- Buscar elemento
- Etc, etc, etc.

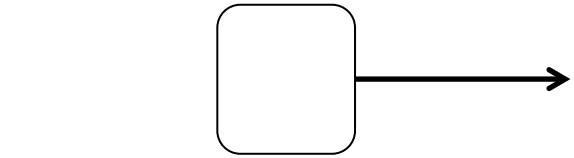
Inserindo Elemento

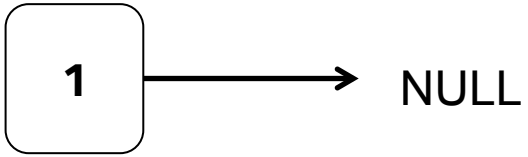
- O que você tem quando tudo começa?
  - A variável inicio apontando para NULL
  - Lista vazia



Inserindo Elemento: 1o Elemento

- Criar um novo elemento (fazer a alocação)
  - Preenche o conteúdo
  - O 'proximo' deste elemento = NULL
- Apontar inicio para o novo elemento



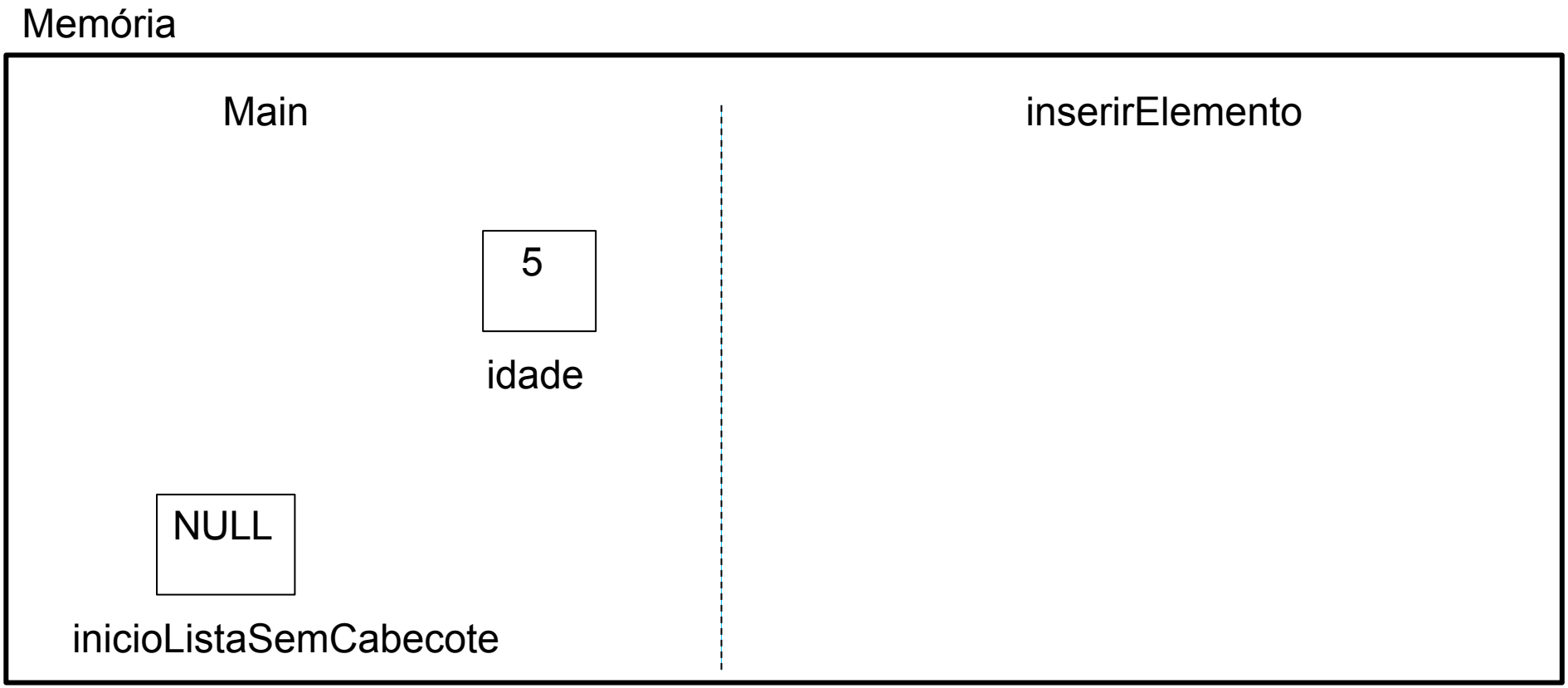


Inserindo Elemento: 1o Elemento

```c
//funcao para inserir um elemento na lista

void inserirElemento(No *inicioSemCabecote, int valor){

    //criar o elemento
    No *novo = (No *)malloc(sizeof(No));
    //preencher o conteudo
    novo->conteudo = valor;
    //apontar para null
    novo->prox = NULL;

    //apontar inicio para esse novo elemento (Com problemas)
    inicioSemCabecote = novo;
}
```

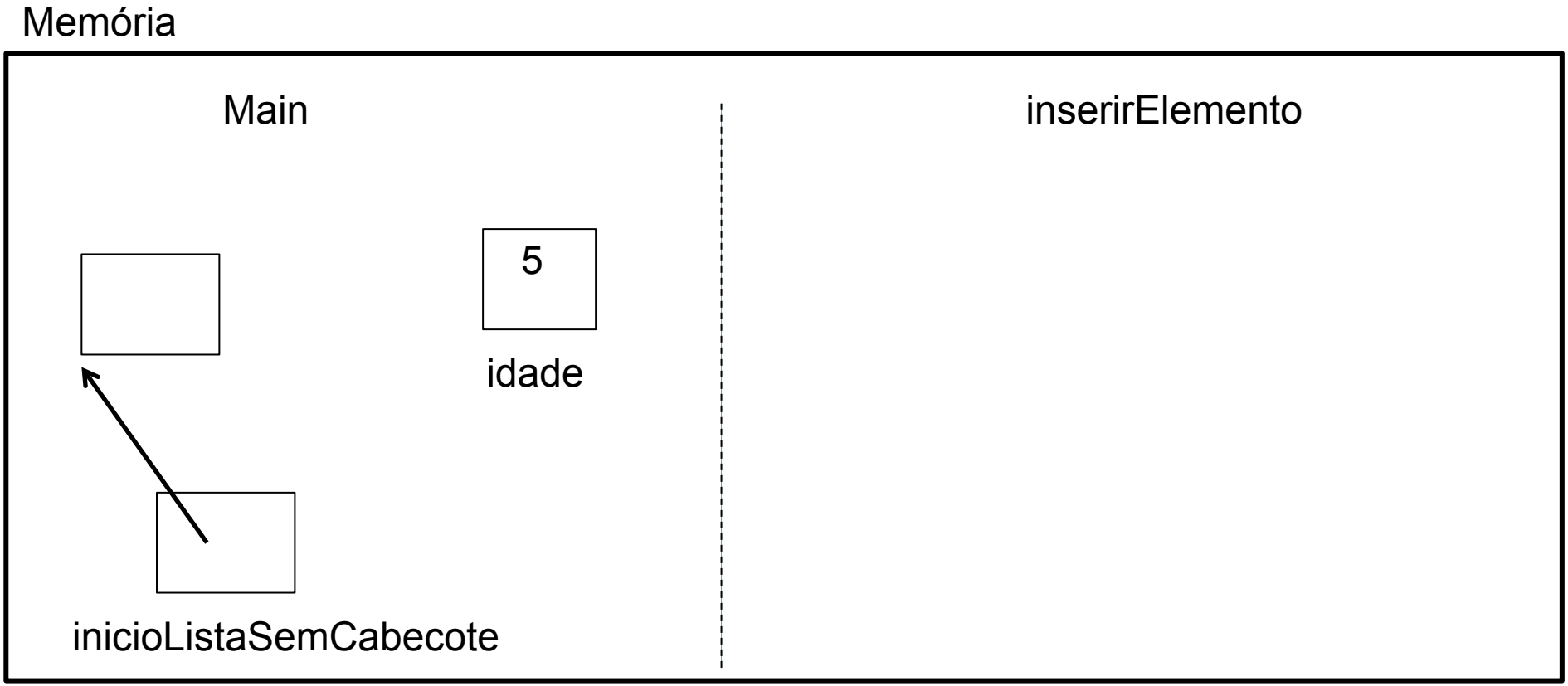
Há um problema no inicio

Sem cabeçote – Ponteiro p/ ponteiro



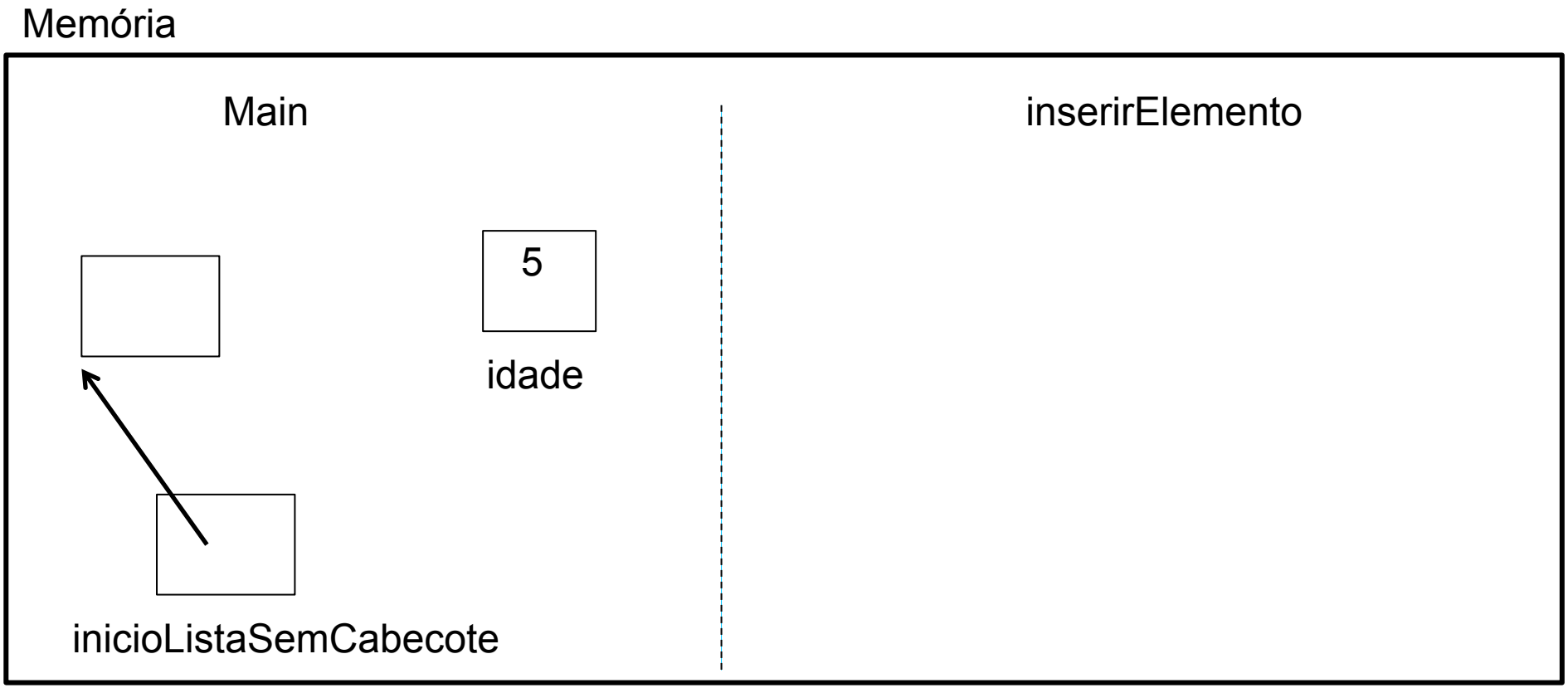

```
int idade = 5;
No *inicioListaSemCabecote = NULL;
```

Sem cabeçote – Ponteiro p/ ponteiro




```
int idade = 5;
No *inicioListaSemCabecote = NULL;
inicioListaSemCabecote = (No *)malloc(sizeof(No));
```

O Que eu quero?



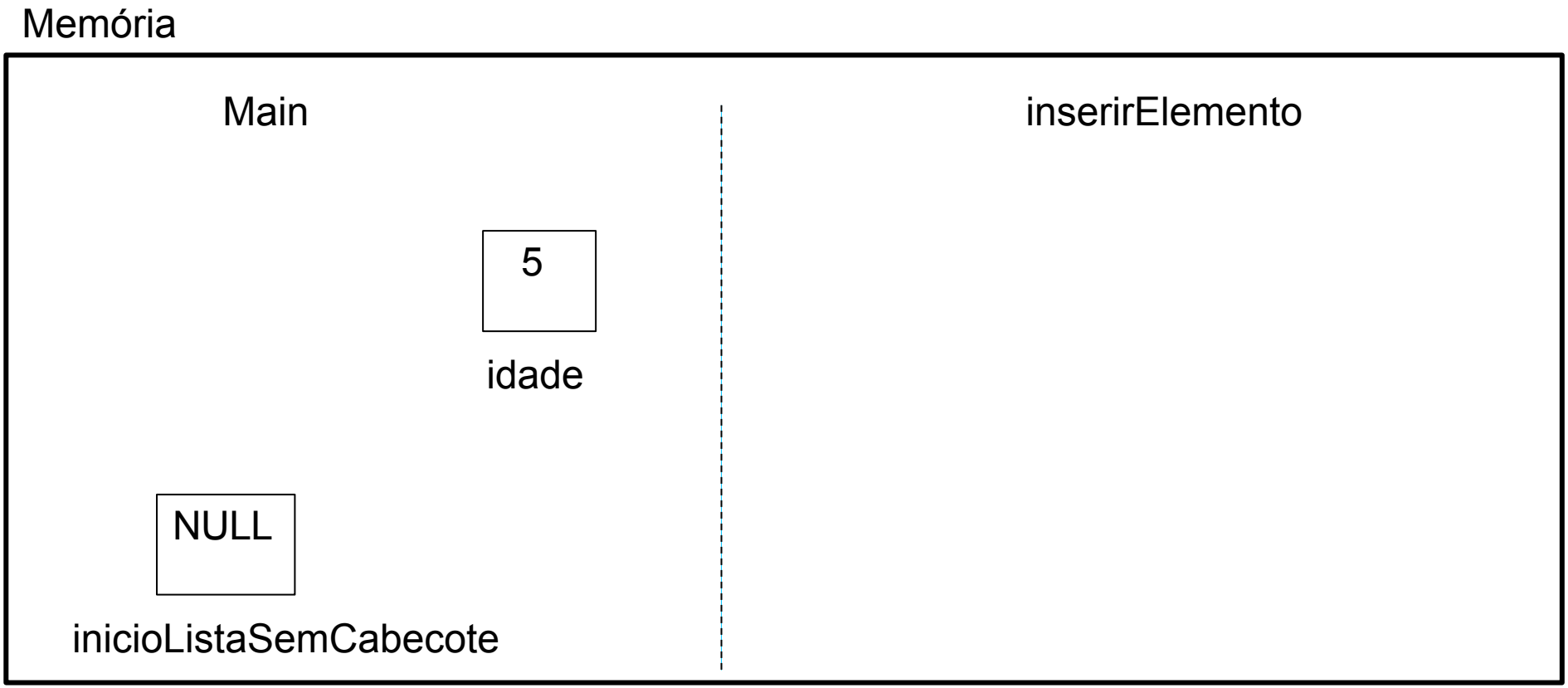
- Eu quero que o conteúdo de inicioListaSemCabecote passe a ter o endereço retornado pelo malloc dentro da funcao inserirElemento

Revendo o código da função

```
inserirElemento(inicioListaSemCabecote, idade);
```

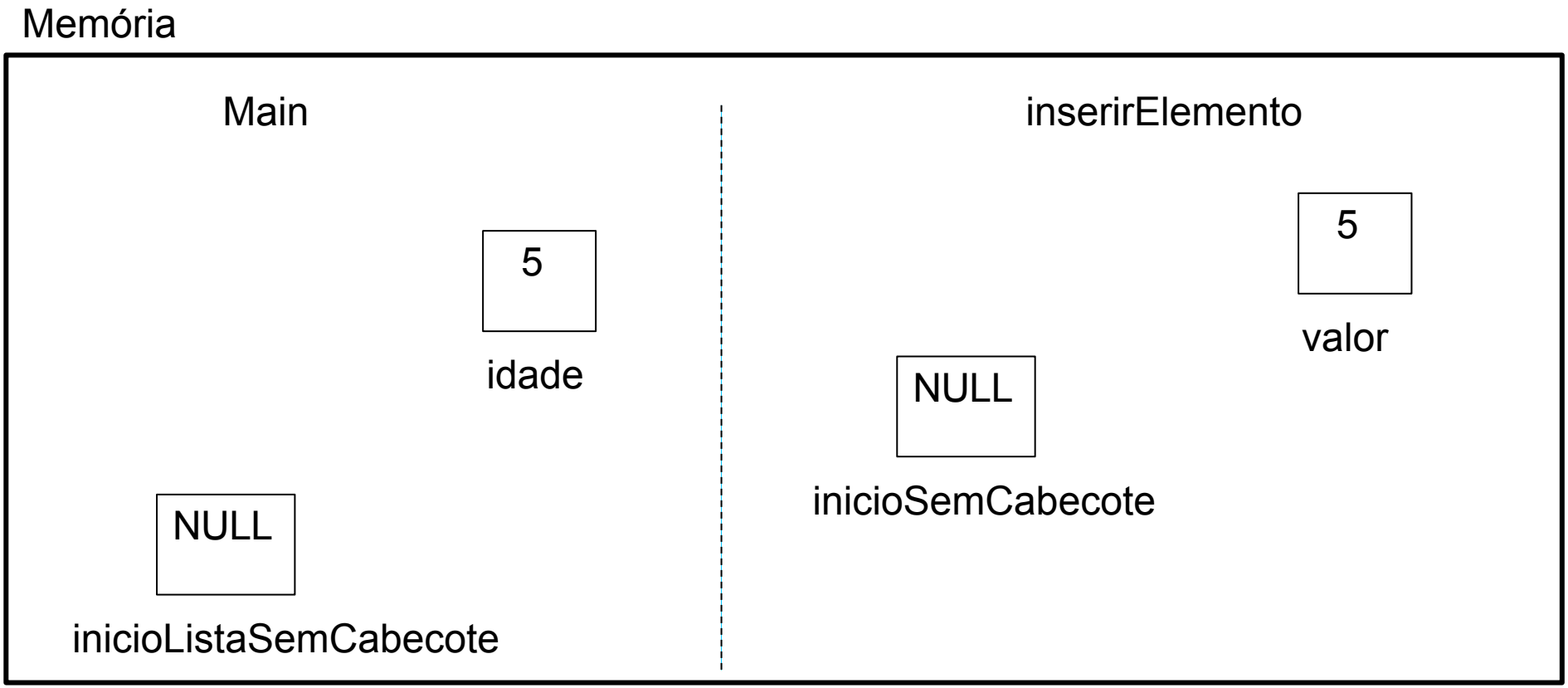
```
//funcao para inserir um elemento na lista

void inserirElemento(No *inicioSemCabecote, int valor){

    //criar o elemento
    No *novo = (No *)malloc(sizeof(No));
    //preencher o conteudo
    novo->conteudo = valor;
    //apontar para null
    novo->prox = NULL;

    //apontar inicio para esse novo elemento (Com problemas)
    inicioSemCabecote = novo;
}
```

Sem cabeçote – Ponteiro p/ ponteiro



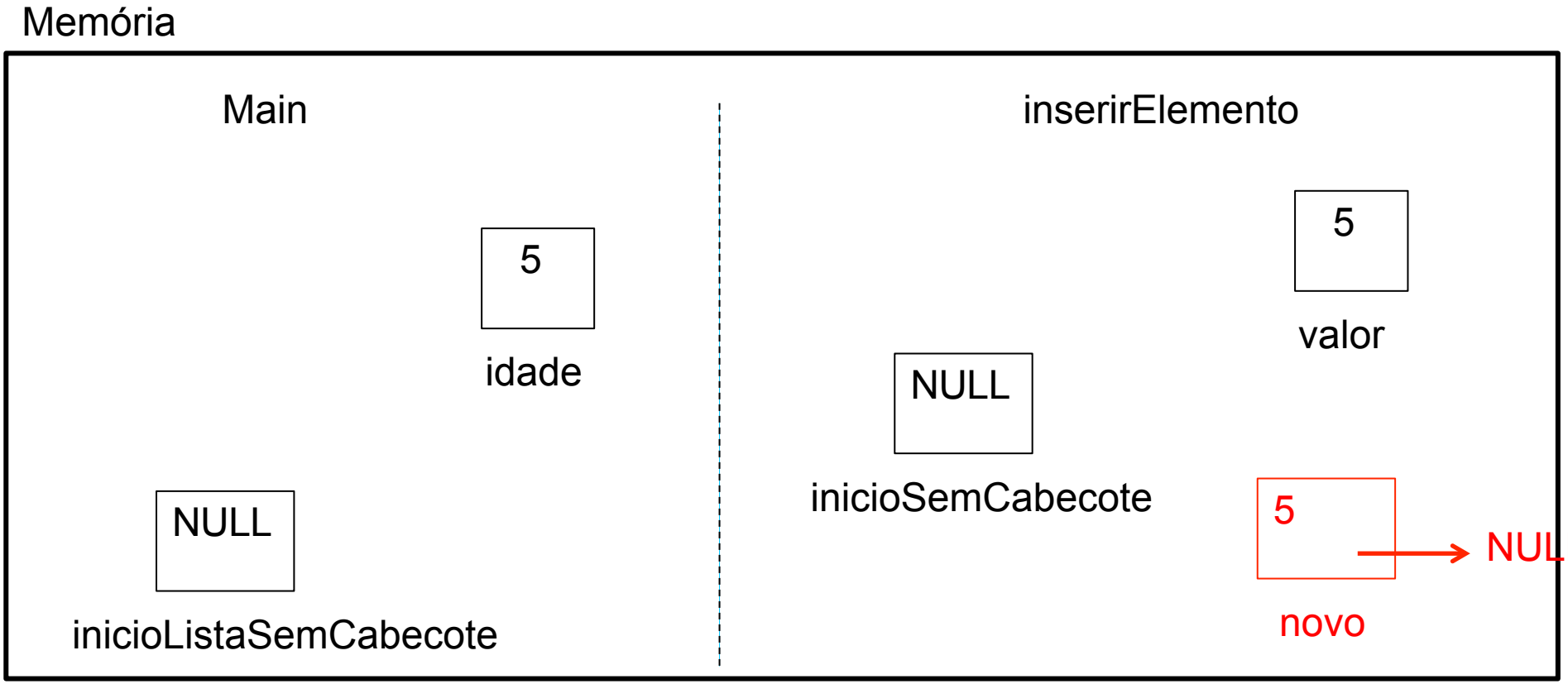

```
inserirElemento(inicioListaSemCabecote, idade);
```

Sem cabeçote – Ponteiro p/ ponteiro



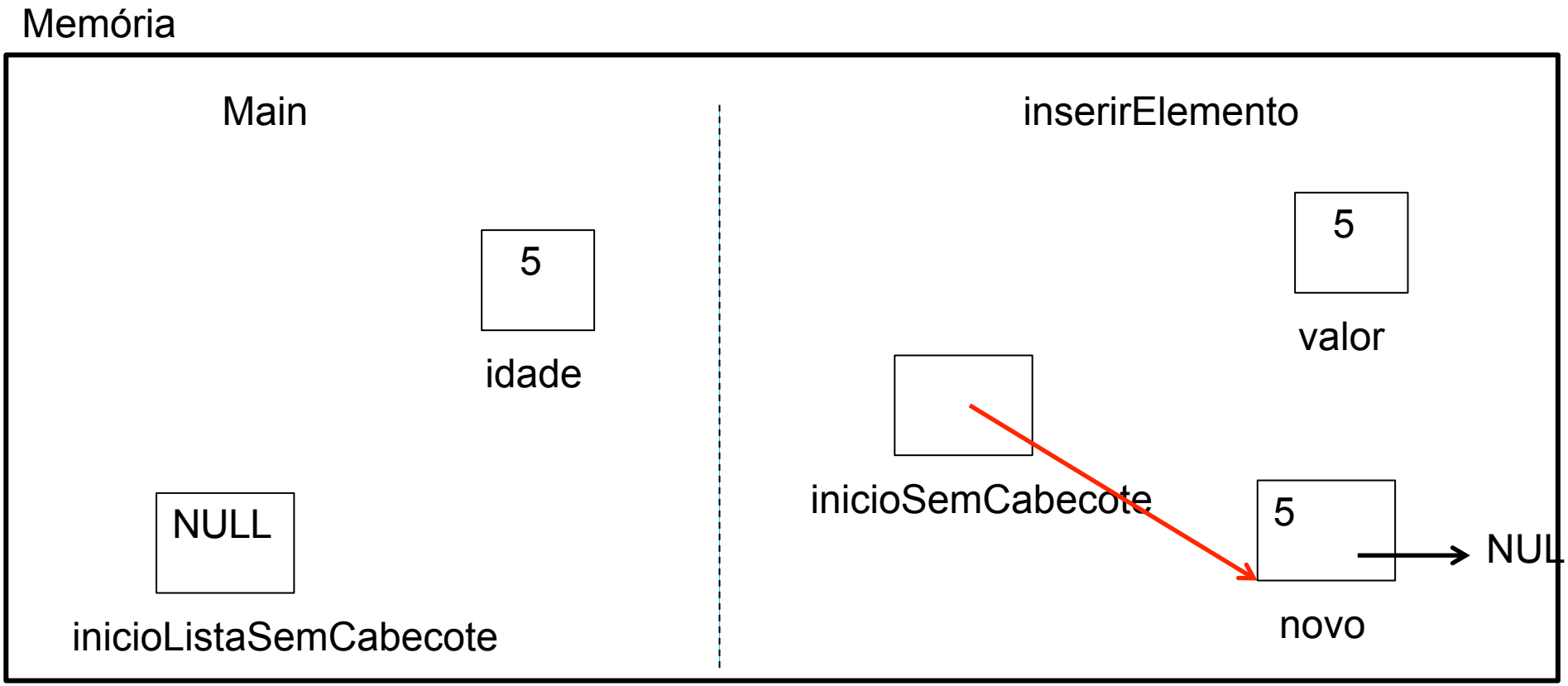

```
void inserirElemento(No *inicioSemCabecote, int valor){
```

Sem cabeçote – Ponteiro p/ ponteiro



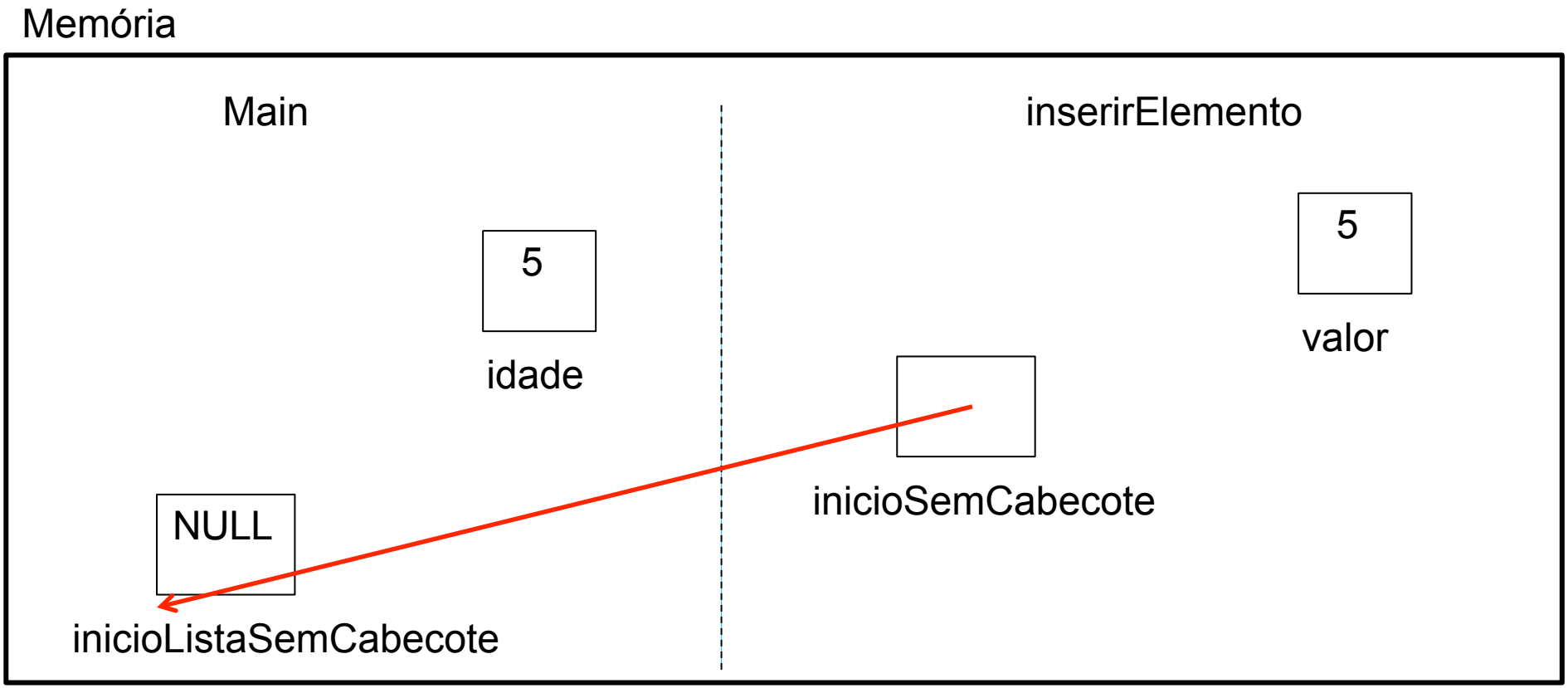

```
//criar o elemento
No *novo = (No *)malloc(sizeof(No));
//preencher o conteudo
novo->conteudo = valor;
//apontar para null
novo->prox = NULL;
```

Sem cabeçote – Ponteiro p/ ponteiro




```
//apontar inicio para esse novo elemento (Com problemas)
inicioSemCabecote = novo;
```

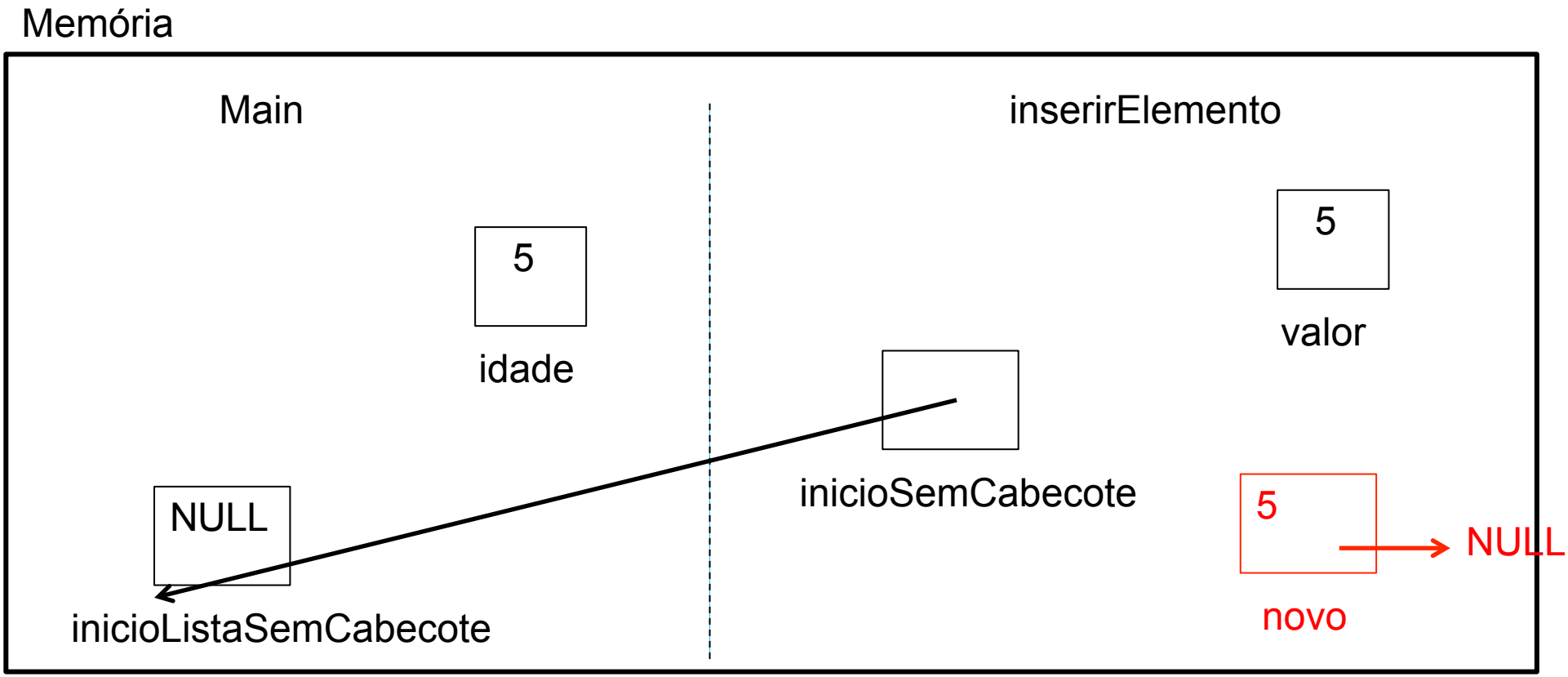
Sem cabeçote – Ponteiro p/ ponteiro




```
inserirElemento(&inicioSemCabecote, idade);
```

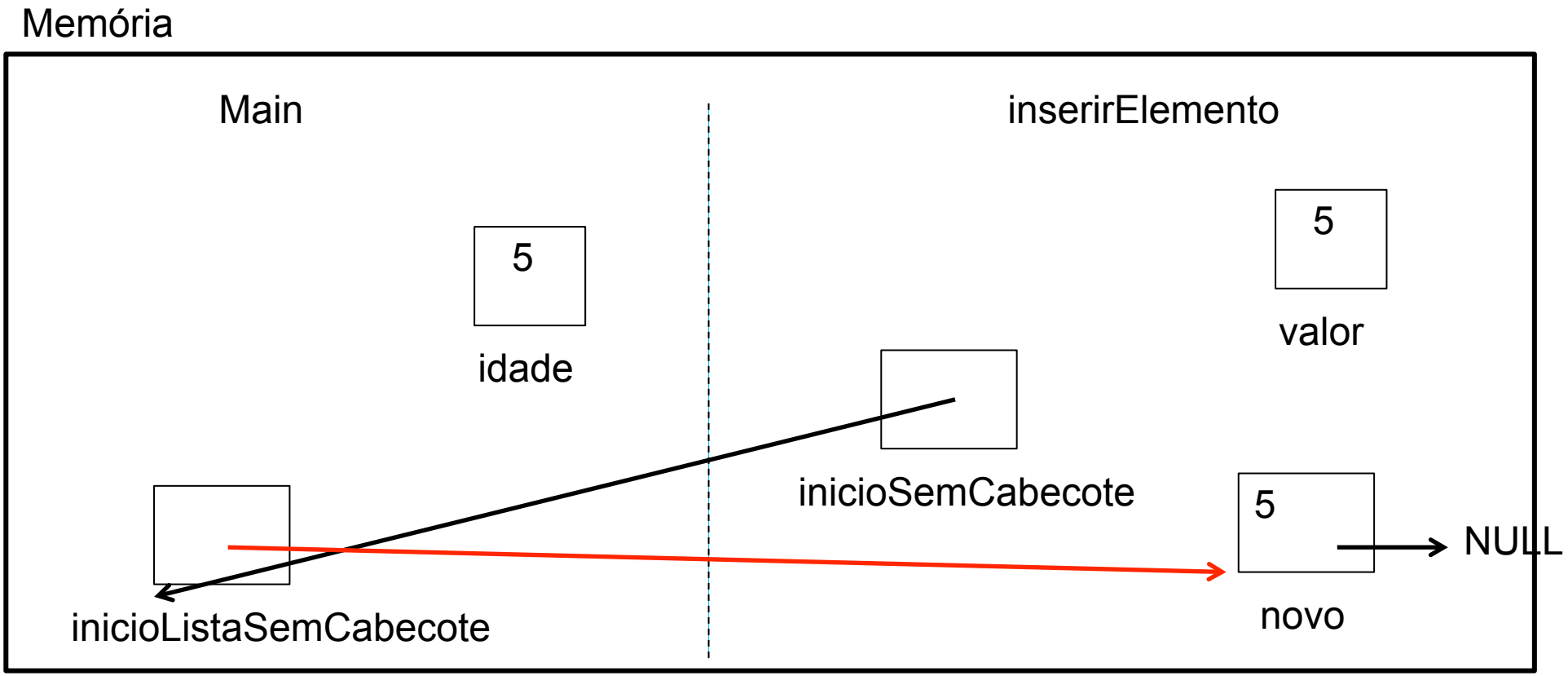
```
void inserirElemento(No **inicioSemCabecote, int valor){
```

Sem cabeçote – Ponteiro p/ ponteiro




```
//criar o elemento
No *novo = (No *)malloc(sizeof(No));
//preencher o conteudo
novo->conteudo = valor;
//apontar para null
novo->prox = NULL;
```

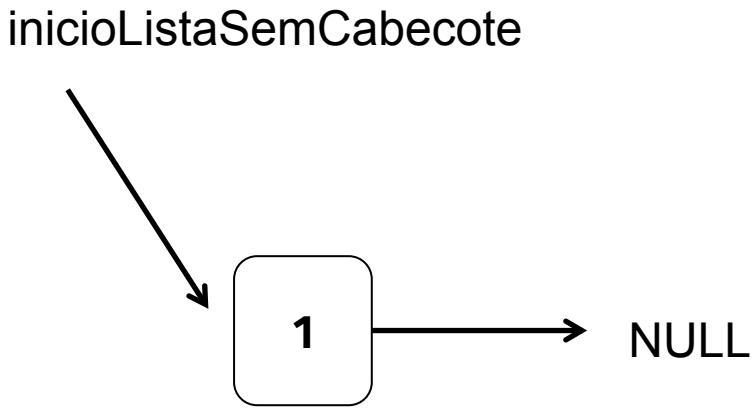
Sem cabeçote – Ponteiro p/ ponteiro




```
//apontar inicio para esse novo elemento
*inicioSemCabecote = novo;
```

Situação atual da lista

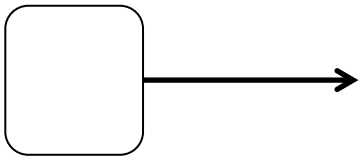
- Um elemento inserido

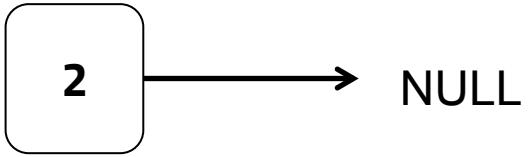


- Novas inserções ao final da lista.

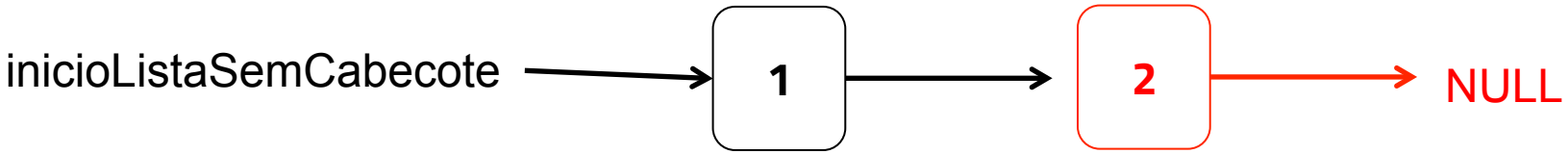
Inserindo Elemento: 2o Elemento

- Criar um novo elemento (fazer a alocação)
  - Preenche o conteúdo
  - O 'proximo' deste elemento = NULL



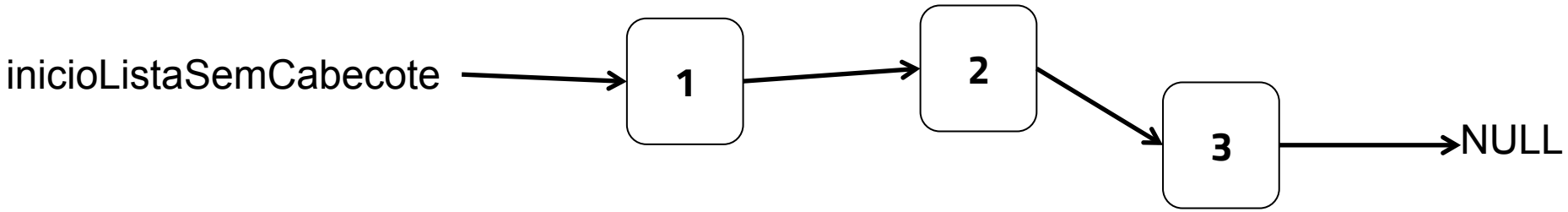


- Percorrer até o final da lista
- Fazer o ultimo elemento apontar para o novo



Inserindo Elemento: 2 ou + Elementos

- Percorrer até o final da lista
- Fazer o ultimo elemento apontar para o novo



- Criar uma variável auxiliar para percorrer a lista, perguntando se o 'proximo' dela == NULL
  - Se não for, passa pro próximo
  - Ser for, pára

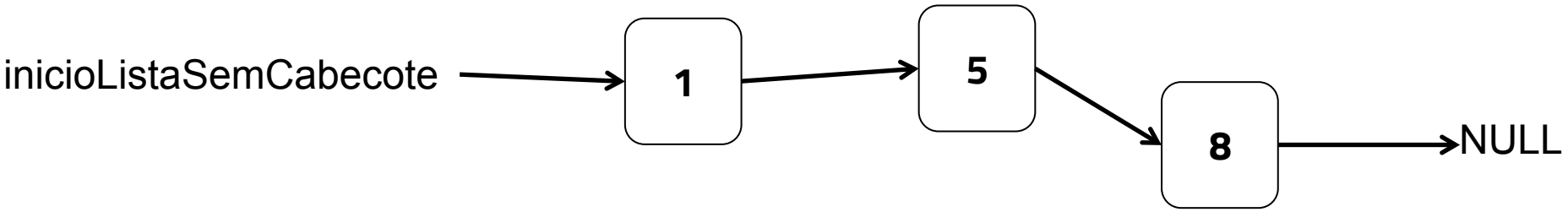
```
atual = *inicioSemCabecote;
while (atual->prox != NULL)
    atual = atual->prox;
```

Código da inserção

```
No *atual;

if (*inicioSemCabecote == NULL)
    //apontar inicio para esse novo elemento
    *inicioSemCabecote = novo;
else{

    //varrer a lista até o final
    atual = *inicioSemCabecote;
    while (atual->prox != NULL)
        atual = atual->prox;

    //neste momento, atual aponta para o último elemento

    //fazer atual apontar para novo
    atual->prox = novo;
}
```

Inserindo ordenado

- Varrer a lista até achar o elemento maior que ele ou chegar no final. Ex: Inserir elemento ‘6’



- Fazer o anterior apontar para novo e o novo para atual

Código da inserção ordenada

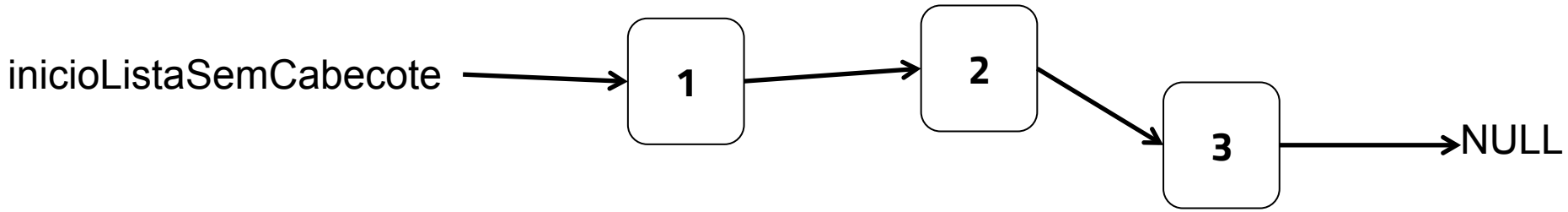
```
No *atual, *anterior;

if (*inicioSemCabecote == NULL)
    //apontar inicio para esse novo elemento
    *inicioSemCabecote = novo;
else{

    //varrer a lista até o achar o primeiro maior ou chegar no final
    atual = *inicioSemCabecote;
    anterior = *inicioSemCabecote;
    while (atual != NULL && atual->conteudo < valor){
        anterior = atual;
        atual = atual->prox;
    }

    //neste momento, atual aponta para
    //o primeiro elemento maior que o novo

    //fazer anterior apontar para novo
    anterior->prox = novo;

    //fazer novo->prox apontar para atual
    novo->prox = atual;
}
```

Lista Encadeada: operações

- Inserir elemento
- Listar elementos
- Excluir elemento
- Buscar elemento
- Etc, etc, etc.

Listagem

- Percorrer até o final da lista
- Imprimir o conteúdo de interesse

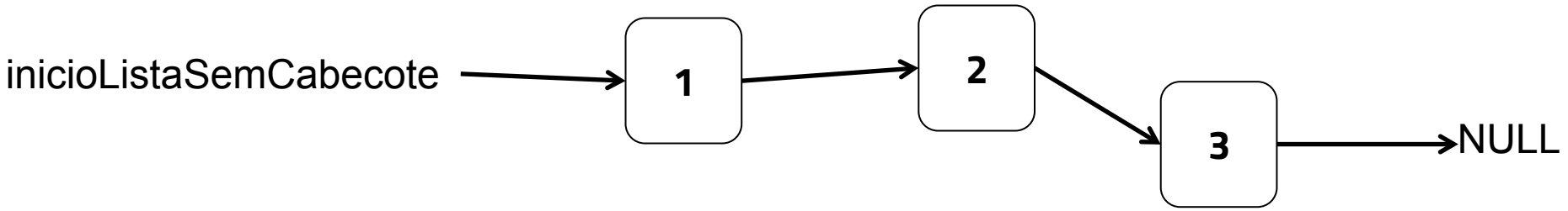


Código da listagem

```
void listarElementos(No *inicioSemCabecote){

    if (inicioSemCabecote == NULL)
        printf("Lista Vazia\n", );
    else {
        printf("Elementos da lista:\n", );

        No *atual = inicioSemCabecote;

        while (atual != NULL){
            printf("%d", atual->conteudo);
            atual = atual->prox;
        }
    }
}
```

Exclusão

- Percorrer até o achar o elemento ou chegar no final
- Quando encontrar o elemento, o que fazer?



- Suponha que seja o '2'
- Precisa fazer o anterior (1) apontar para o posterior (3)

Lista Encadeada: operações

- Inserir elemento
- Listar elementos
- Excluir elemento
- Buscar elemento
- Etc, etc, etc.

Exclusão

- Se for o primeiro elemento, o inicio tem que apontar para o próximo
- Para finalizar, fazer o free no elemento a ser excluído

Código da exclusão

```c
int excluirElemento(No** inicio, int valor){
    if (*inicio == NULL)
        return LISTA_VAZIA; // lista vazia

    No* anterior = *inicio;
    No* atual = *inicio;
    No* proximo = atual->prox;
    int achou = 0;

    while(atual != NULL){
        if (atual->conteudo == valor){
            achou = 1;
            break;
        }
        anterior = atual;
        atual = proximo;
        if (atual != NULL)
            proximo = atual->prox;
    }

    if (achou){
        if (atual == *inicio)
            *inicio = proximo;
        else
            anterior->prox = atual->prox;
        free(atual);
        return SUCESSO_EXCLUSAO;
    }else
        return NAO_ENCONTRADO;
}
```

Lista Encadeada

perguntas?

Prof. Renato Novais
renatonovais@gmail.com