UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Câmpus Campo Mourão

Ciência da Computação

Algoritmos e Estrutura de Dados 1

Lista Duplamente Encadeada

Simulação

Prof. Rafael Liberato
liberato@utfpr.edu.br

Objetivos

- Prover simulações de **possíveis** algoritmos para as principais funções da Lista Duplamente Encadeada

É importante ressaltar que as simulações não representam a **única** e/ou a **melhor** forma de desenvolver o algoritmo. É apenas uma sugestão para orientar no desenvolvimento.

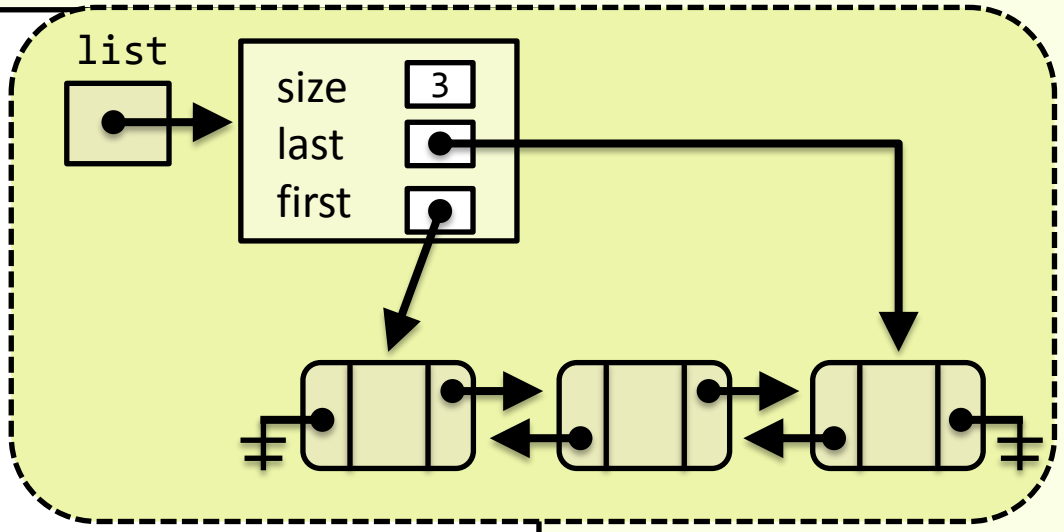
TAD Lista

Lista Duplamente Encadeada

```
#define ItemType int

typedef struct{
    Node *first;
    Node *last;
    int size;
}List;


List *createList ();
void initializeList(List *l);
int addList(List *l, ItemType e);
int addList(List* q, ItemType e, int index);
int removeList(List* q, int index, ItemType *e);
int removeList(List* q, ItemType* e);
int getList(List* q, int index, ItemType* e);
int setList(List* q, int index, ItemType* e);
int indexOfList(List* q, ItemType* e);
int containsList(List* q, ItemType *e);
int sizeList(List* q);
int isEmptyList(List* q);
void printList(List* q);
```




```
typedef struct node{
    ItemType     data;
    struct node  *prev
    struct node  *next;
}Node;
```

Simulações

Simulações

Clique na seta para ver a simulação

Simulação passo a passo das principais funções

```
List *createList ();
```

```
void initializeList(List *l);
```

```
int addLastList(List *l, ItemType e);
```

```
int addList(List* l, ItemType e, int index);
```

```
int removeList(List* l, int index, ItemType *e);
```

```
int removeElementList(List* l, ItemType* e);
```

```
List *createList ();
void initializeList(List *l);
int addLastList(List *l, ItemType e);
int addList(List* l, ItemType e, int index);
int removeList(List* l, int index, ItemType *e);
int removeElementList(List* l, ItemType* e);
```

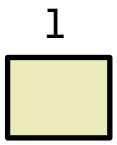

createList

createList

```
List *createList();
```

1

Representação

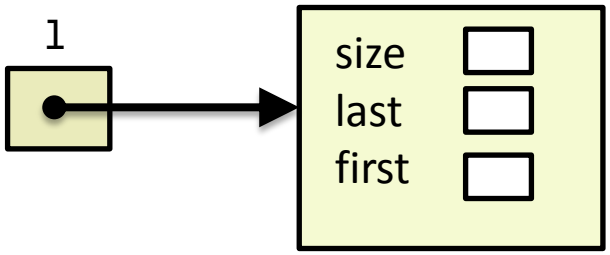


createList

```
List *createList();
```

1

Representação

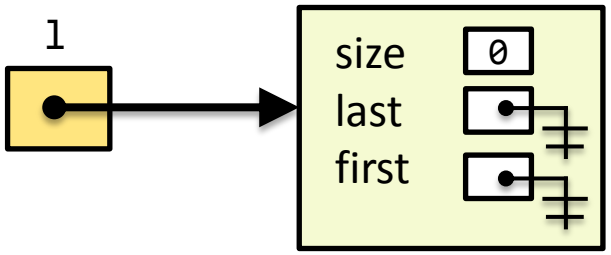


createList

```
List *createList();
```

1

Representação



Devolve o l (endereço da lista criada)

```
List *createList ();
void initializeList(List *l);
int addLastList(List *l, ItemType e);
int addList(List* l, ItemType e, int index);
int removeList(List* l, int index, ItemType *e);
int removeElementList(List* l, ItemType* e);
```


initializeList

initializeList

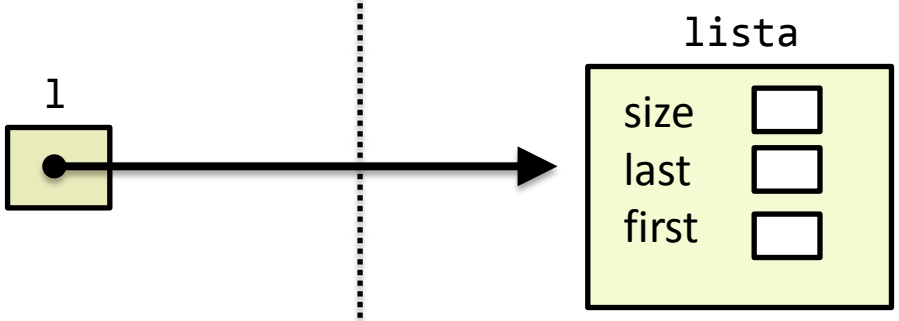
```
void initializeList(List *l);
```

1

Representação

Escopo da função **initializeList**

Escopo da função **main**



initializeList

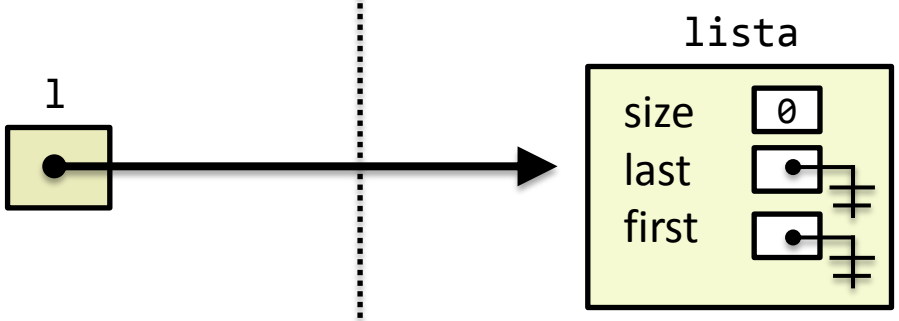
```
void initializeList(List *l);
```

1

Representação

Escopo da função **initializeList**

Escopo da função **main**



initializeList

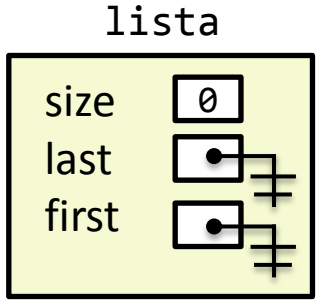
```
void initializeList(List *l);
```

1

Representação

Escopo da função **initializeList**

Escopo da função **main**



```
List *createList ();
void initializeList(List *l);
int addLastList(List *l, ItemType e);
int addList(List* l, ItemType e, int index);
int removeList(List* l, int index, ItemType *e);
int removeElementList(List* l, ItemType* e);
```

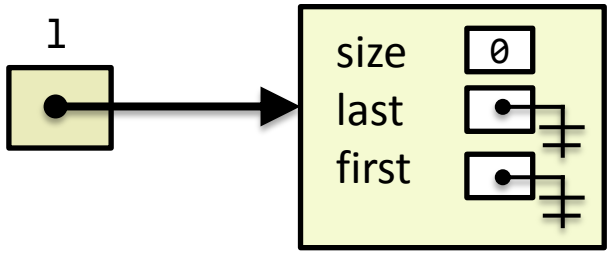

addLastList

addLastList

```
int addLastList(List *l, ItemType e);
```

1 2 Inserção quando a lista está vazia

Representação

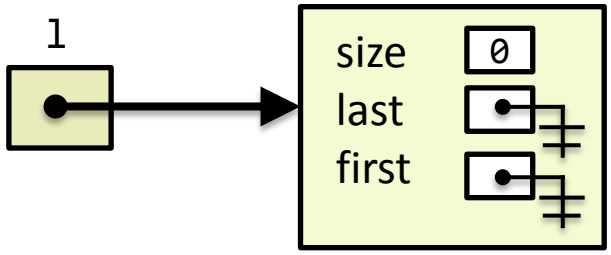


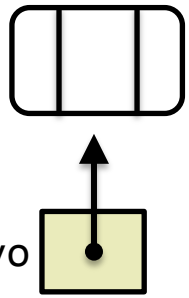
addLastList

```
int addLastList(List *l, ItemType e);
```

1 2 Inserção quando a lista está vazia

Representação



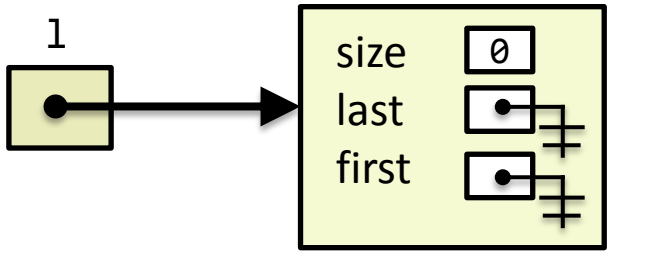


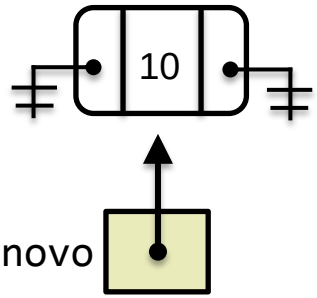
addLastList

```
int addLastList(List *l, ItemType e);
```

1 2 Inserção quando a lista está vazia

Representação



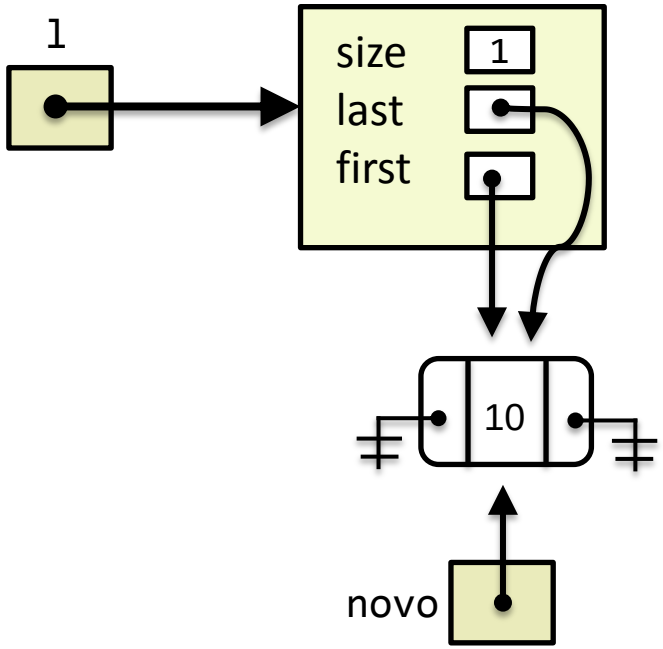


addLastList

```
int addLastList(List *l, ItemType e);
```

1 2 Inserção quando a lista está vazia

Representação

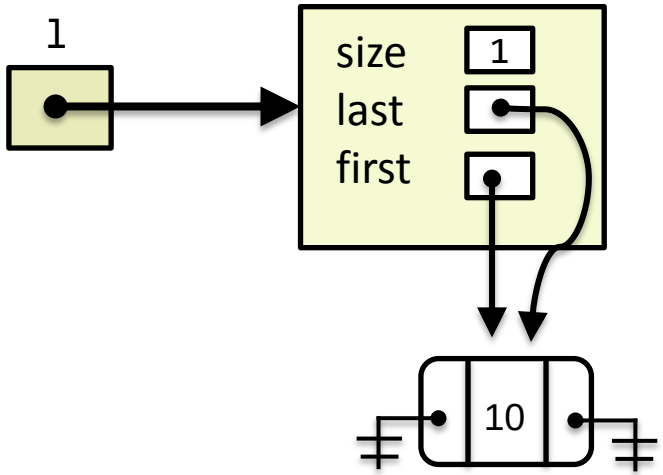


addLastList

```
int addLastList(List *l, ItemType e);
```

1 2 Inserção quando a lista está vazia

Representação

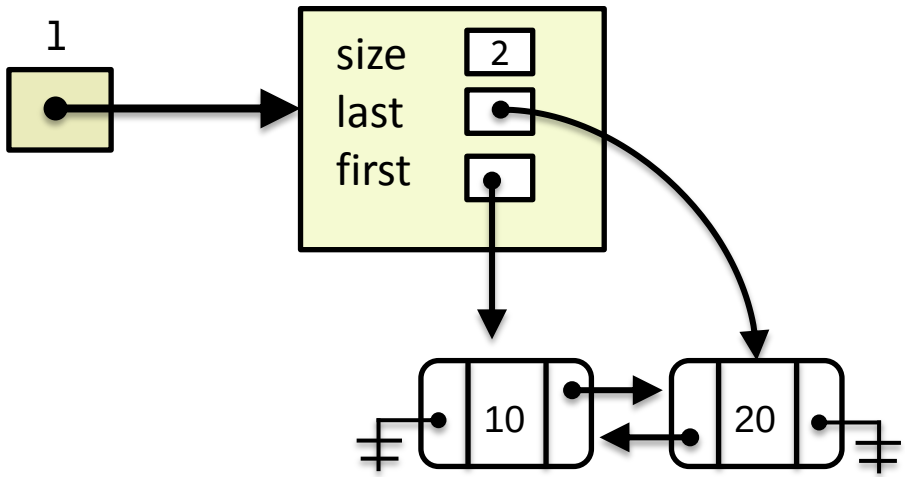


addLastList

```
int addLastList(List *l, ItemType e);
```

1 2 Inserção quando a lista NÃO está vazia

Representação

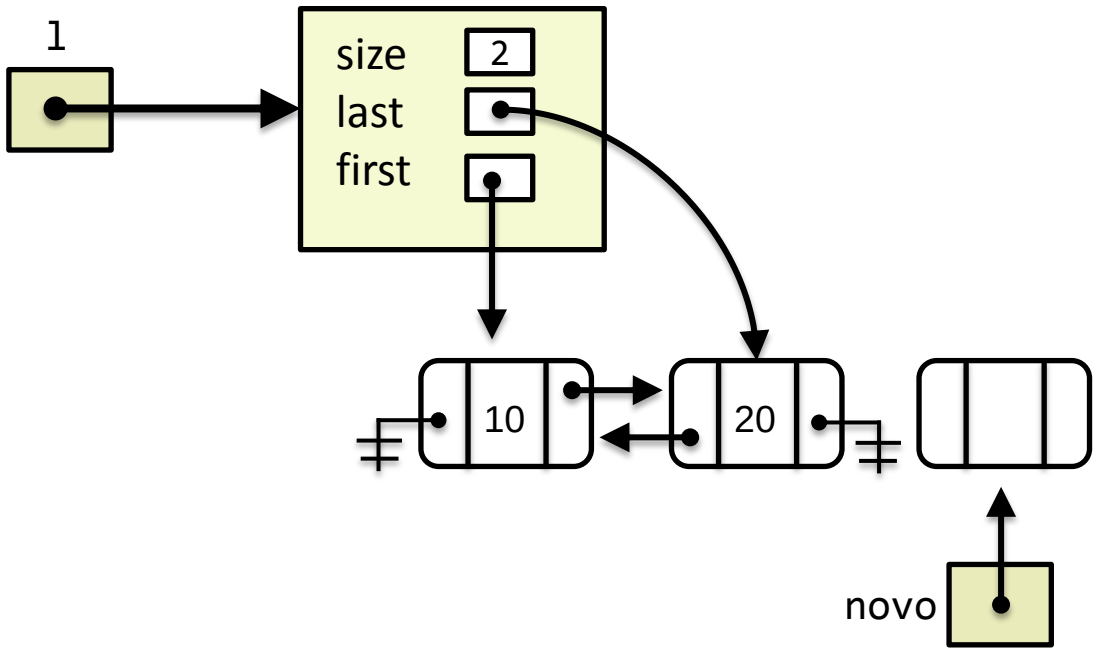


addLastList

```
int addLastList(List *l, ItemType e);
```

1 2 Inserção quando a lista NÃO está vazia

Representação

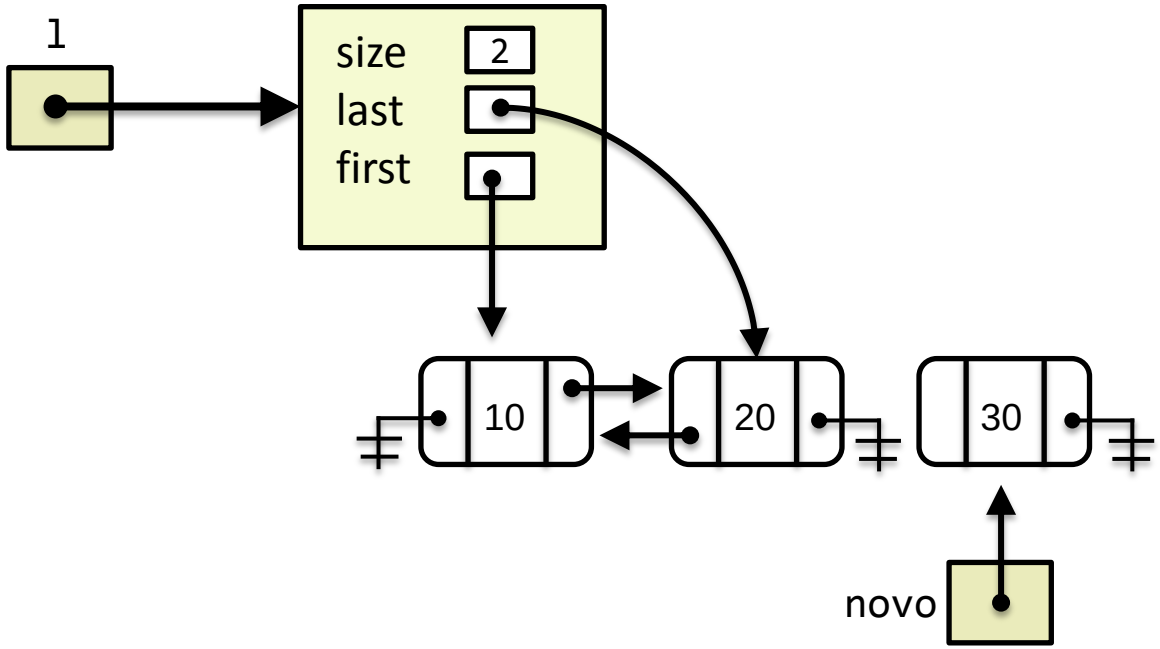


addLastList

```
int addLastList(List *l, ItemType e);
```

1 2 Inserção quando a lista NÃO está vazia

Representação

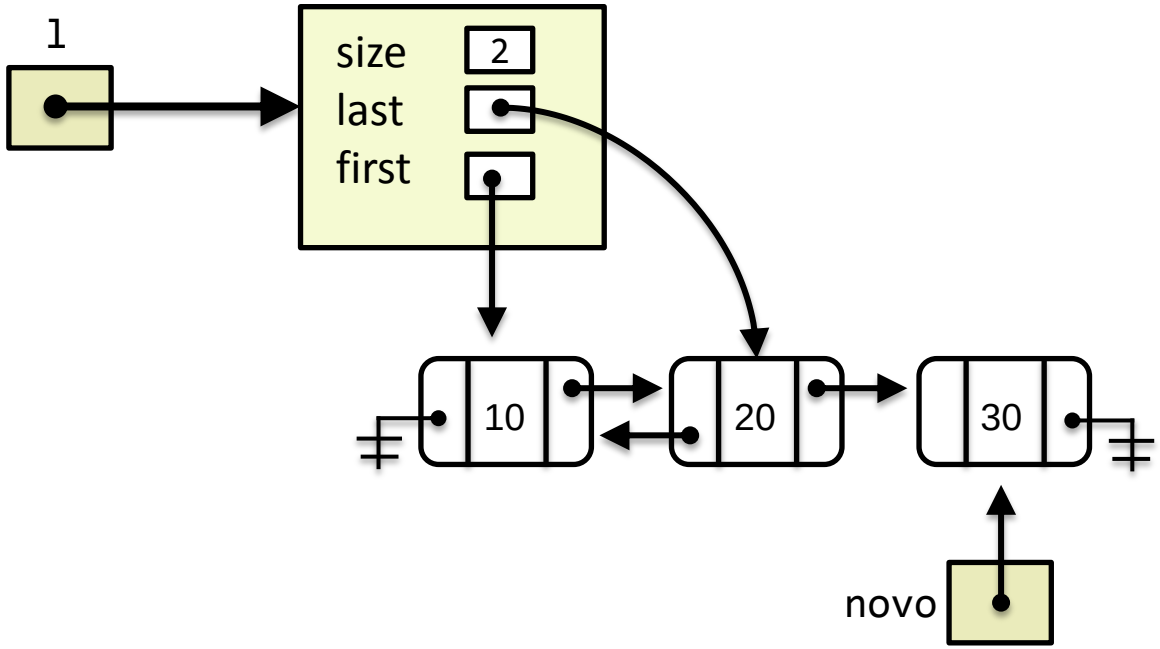


addLastList

```
int addLastList(List *l, ItemType e);
```

1 2 Inserção quando a lista NÃO está vazia

Representação

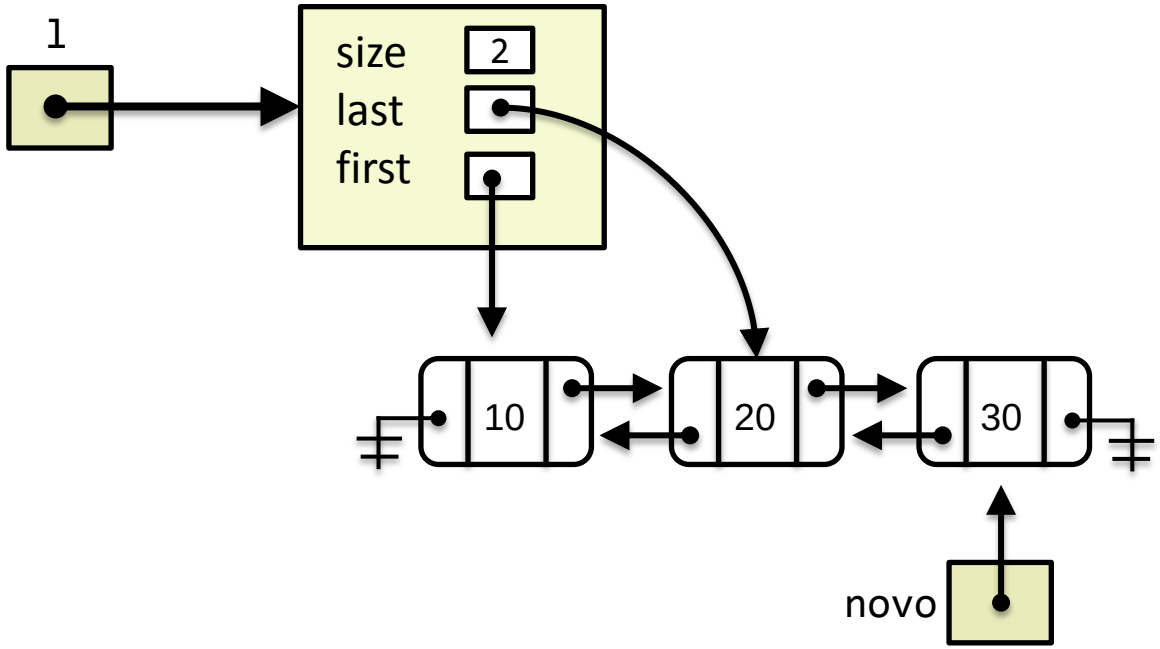


addLastList

```
int addLastList(List *l, ItemType e);
```

1 2 Inserção quando a lista NÃO está vazia

Representação

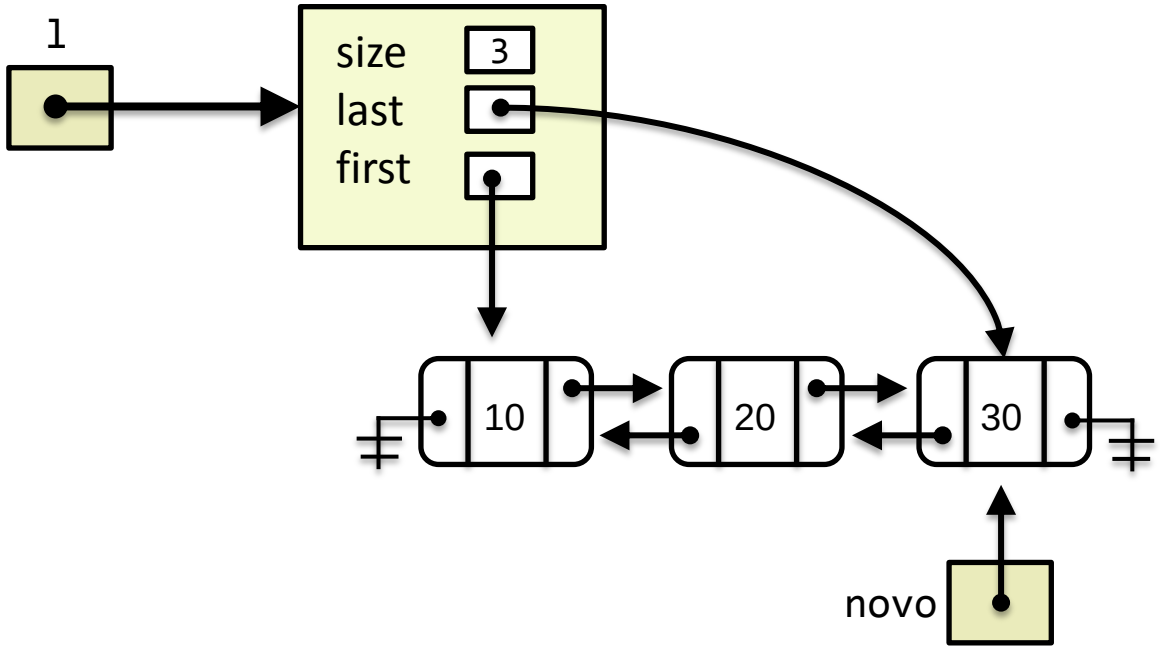


addLastList

```
int addLastList(List *l, ItemType e);
```

1 2 Inserção quando a lista NÃO está vazia

Representação

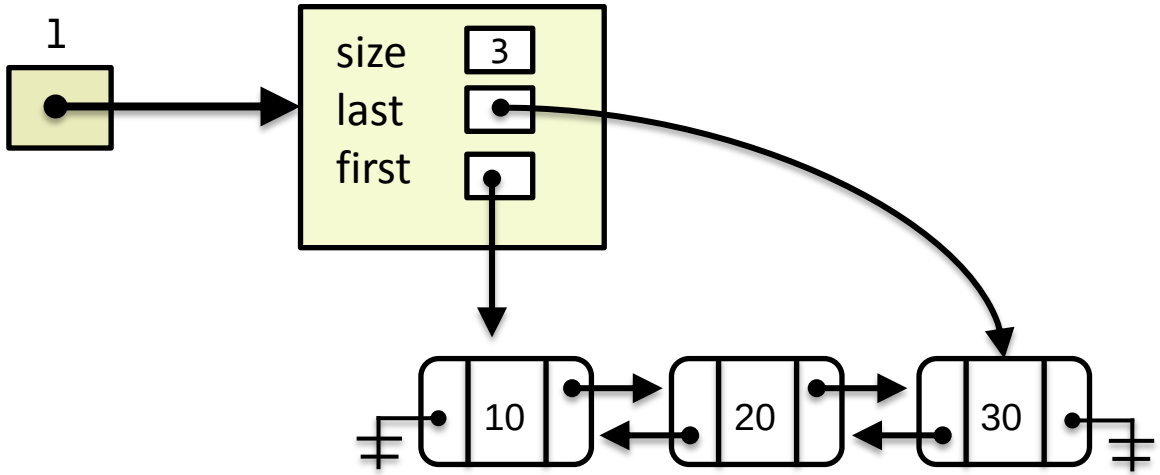


addLastList

```
int addLastList(List *l, ItemType e);
```

1 2 Inserção quando a lista NÃO está vazia

Representação



```
List *createList ();
void initializeList(List *l);
int addLastList(List *l, ItemType e);
int addList(List* l, ItemType e, int index);
int removeList(List* l, int index, ItemType *e);
int removeElementList(List* l, ItemType* e);
```

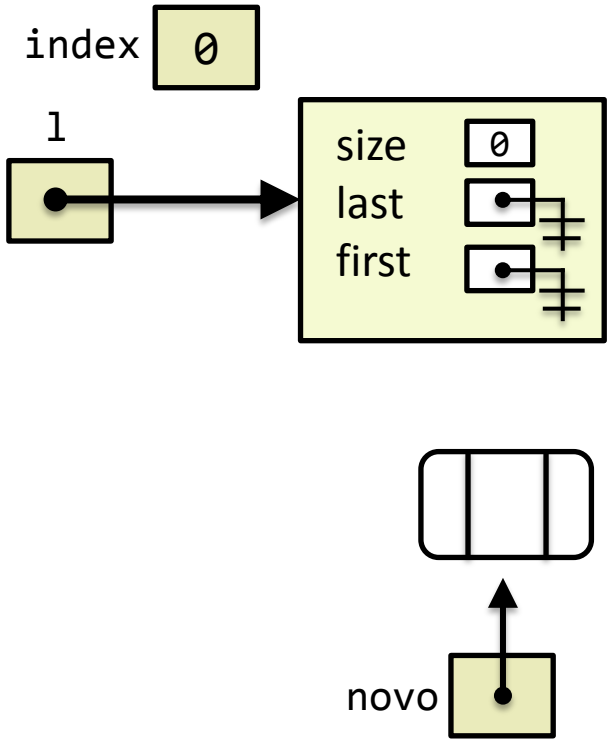

addList

addList

```
int addList(List* l, ItemType e, int index);
```

1 2 3 4 Inserção na primeira posição quando a lista está vazia

Representação

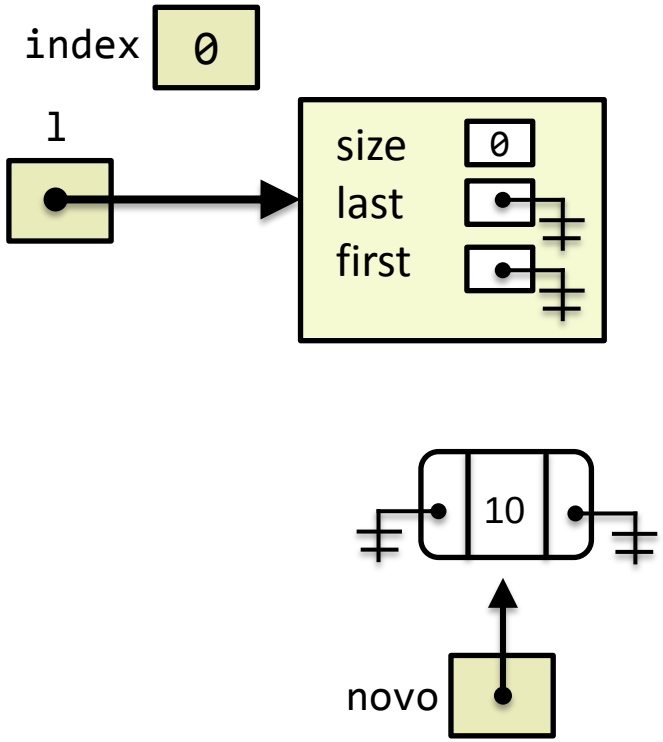


addList

```
int addList(List* l, ItemType e, int index);
```

1 2 3 4 Inserção na primeira posição quando a lista está vazia

Representação

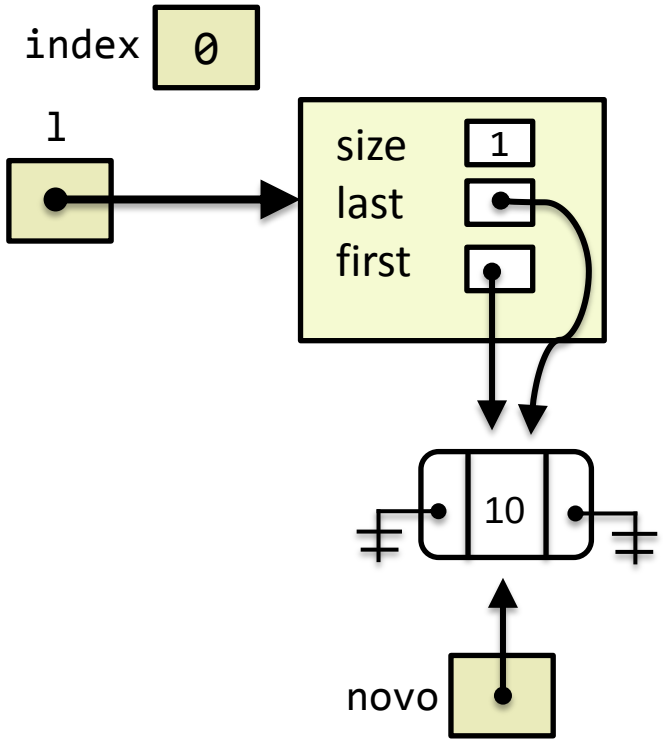


addList

```
int addList(List* l, ItemType e, int index);
```

1 2 3 4 Inserção na primeira posição quando a lista está vazia

Representação

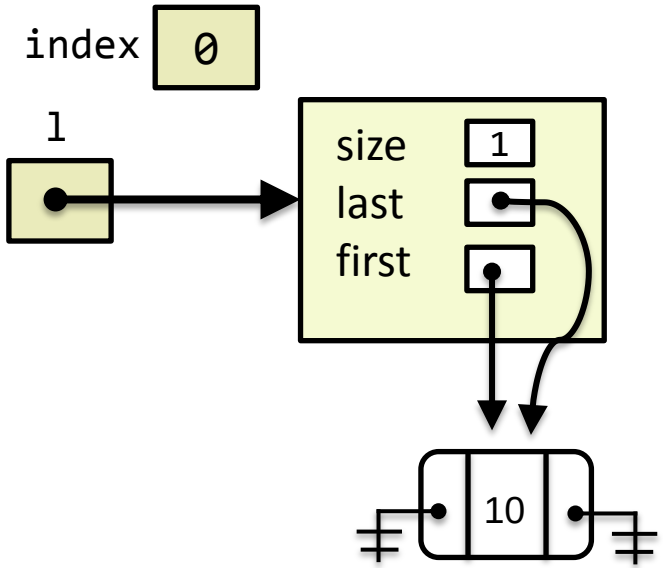


addList

```
int addList(List* l, ItemType e, int index);
```

1 2 3 4 Inserção na primeira posição quando a lista está vazia

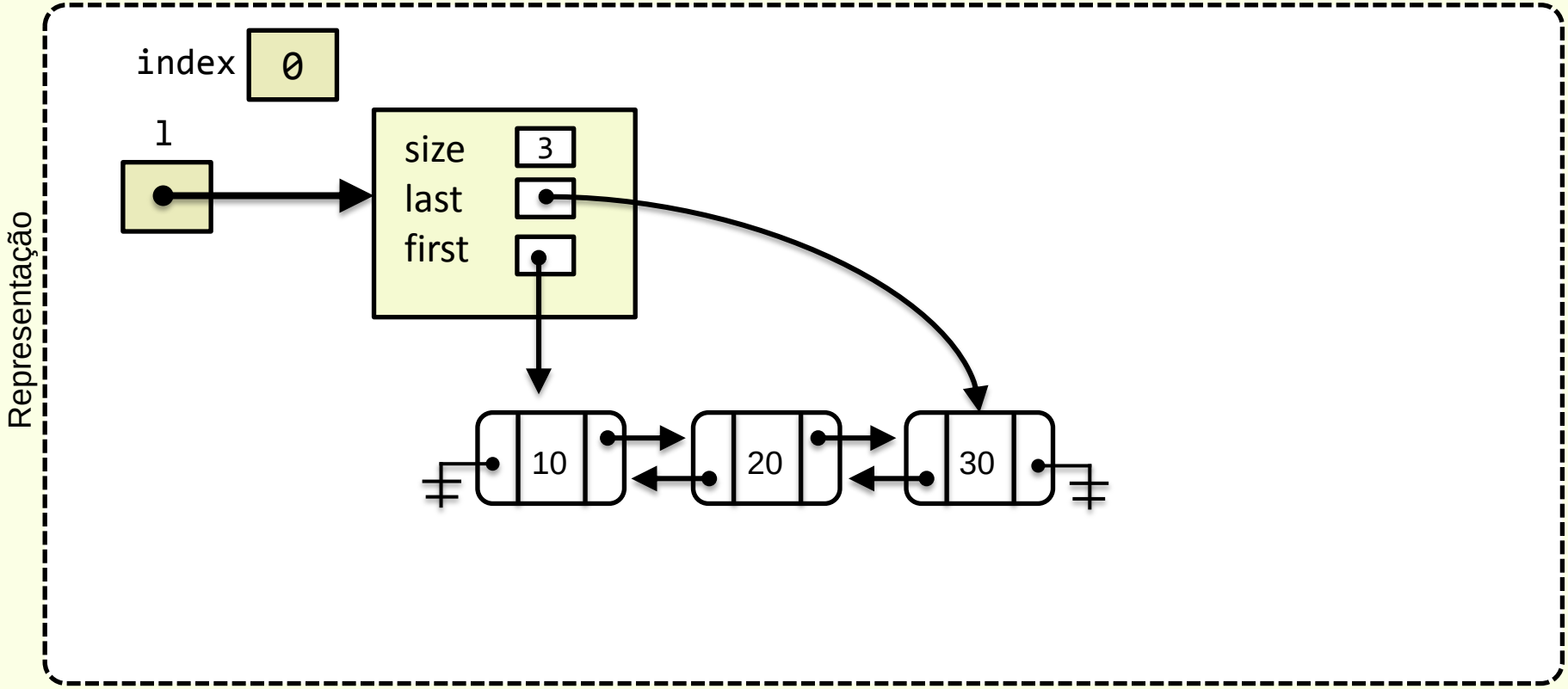
Representação



addList

```
int addList(List* l, ItemType e, int index);
```

1 2 3 4 Inserção na primeira posição quando a lista NÃO está vazia

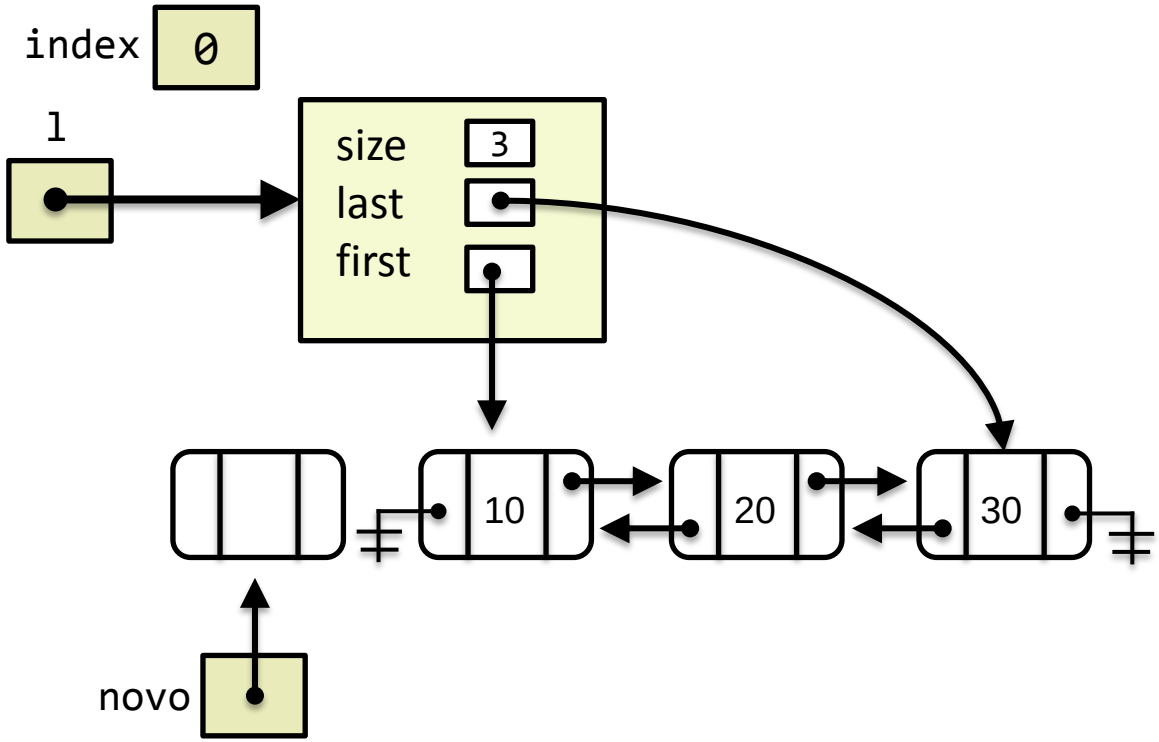


addList

```
int addList(List* l, ItemType e, int index);
```

1 2 3 4 Inserção na primeira posição quando a lista NÃO está vazia

Representação

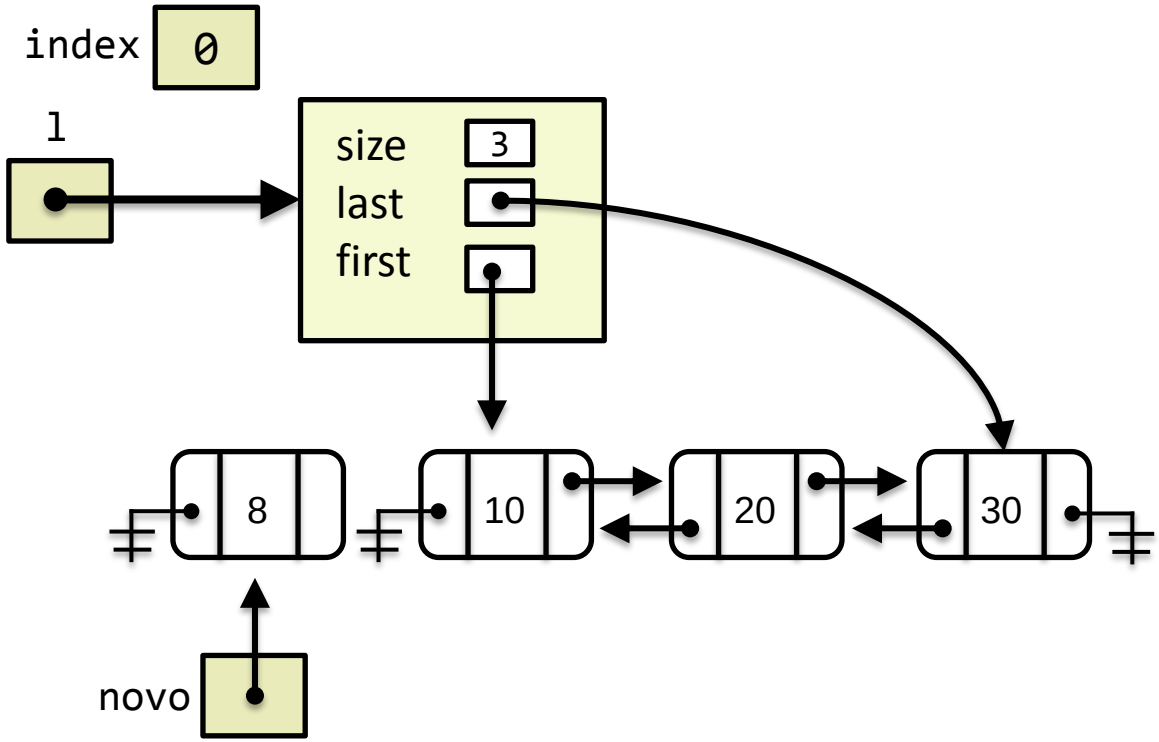


addList

```
int addList(List* l, ItemType e, int index);
```

1 2 3 4 Inserção na primeira posição quando a lista NÃO está vazia

Representação

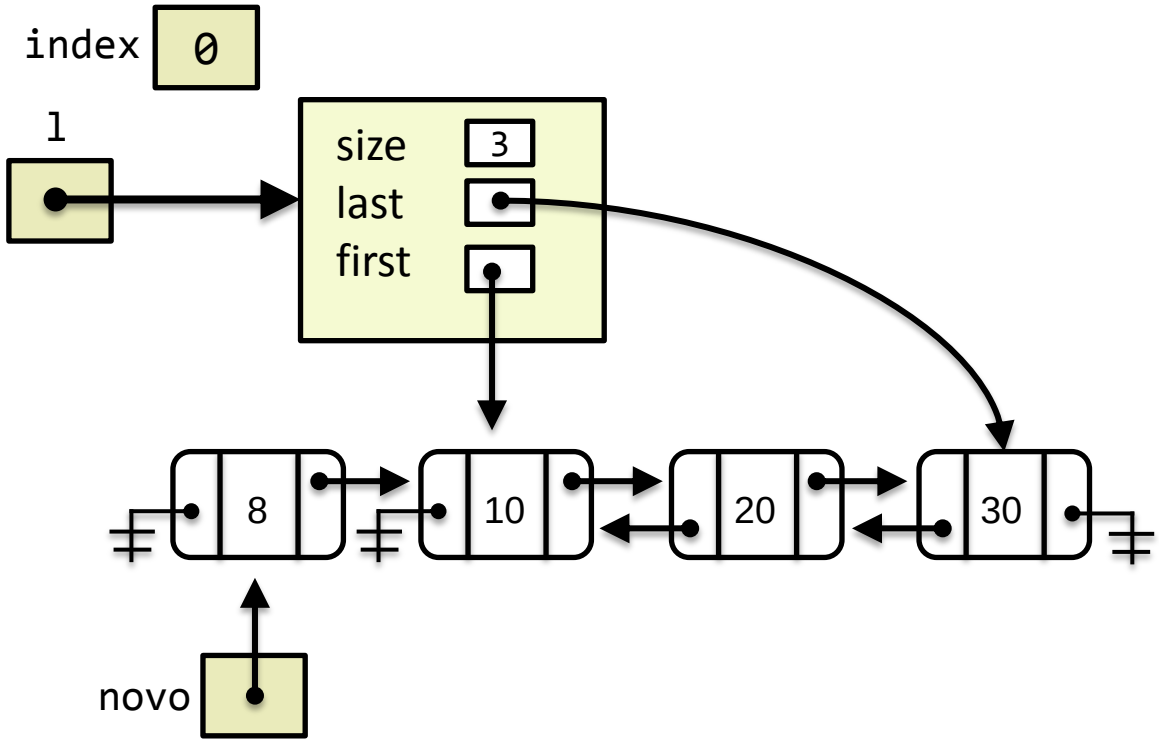


addList

```
int addList(List* l, ItemType e, int index);
```

1 2 3 4 Inserção na primeira posição quando a lista NÃO está vazia

Representação

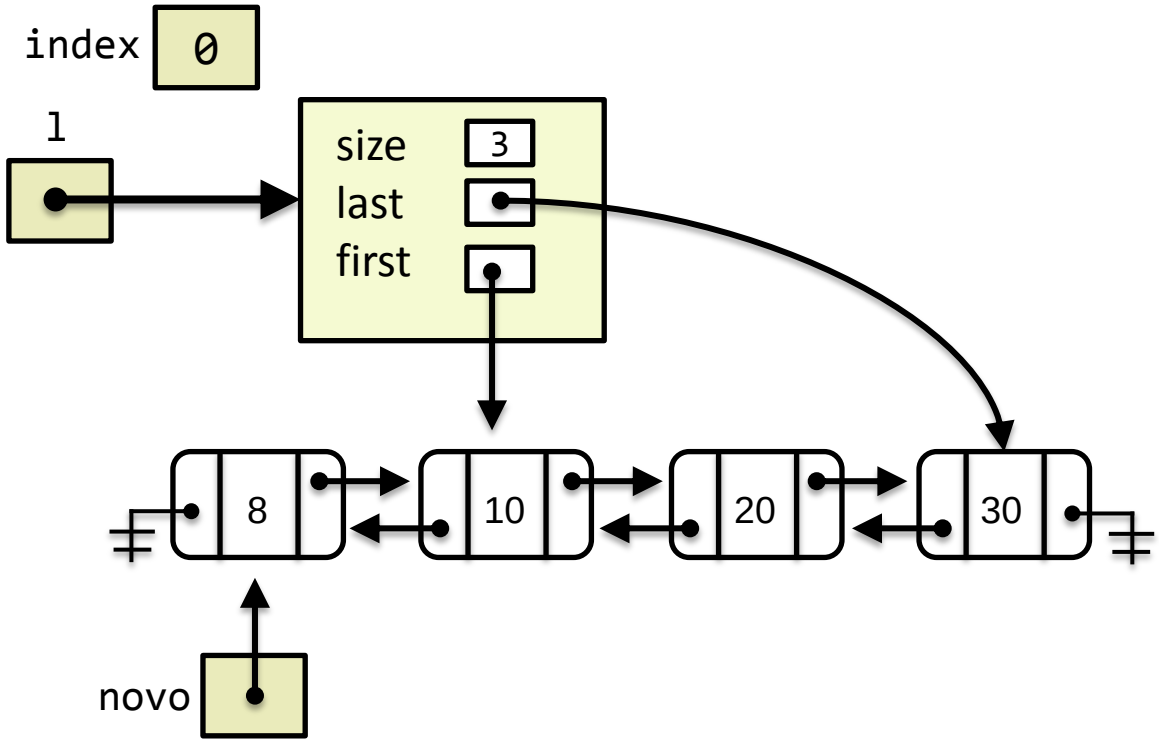


addList

```
int addList(List* l, ItemType e, int index);
```

1 2 3 4 Inserção na primeira posição quando a lista NÃO está vazia

Representação

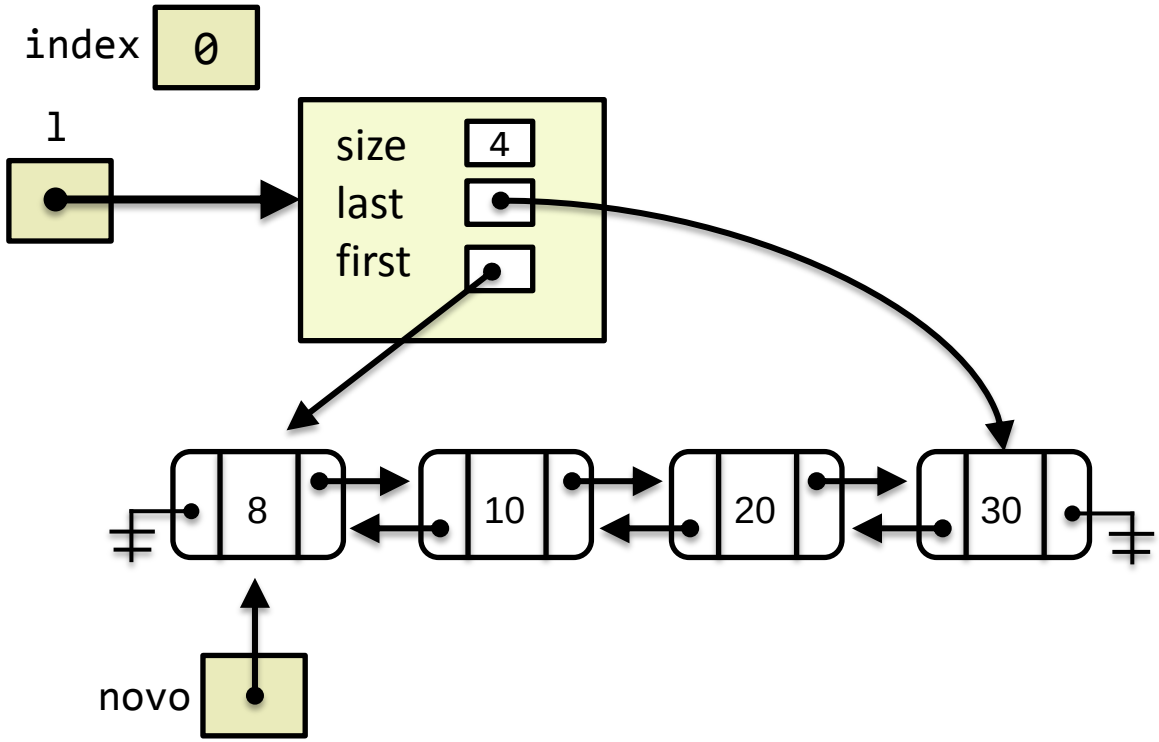


addList

```
int addList(List* l, ItemType e, int index);
```

1 2 3 4 Inserção na primeira posição quando a lista NÃO está vazia

Representação

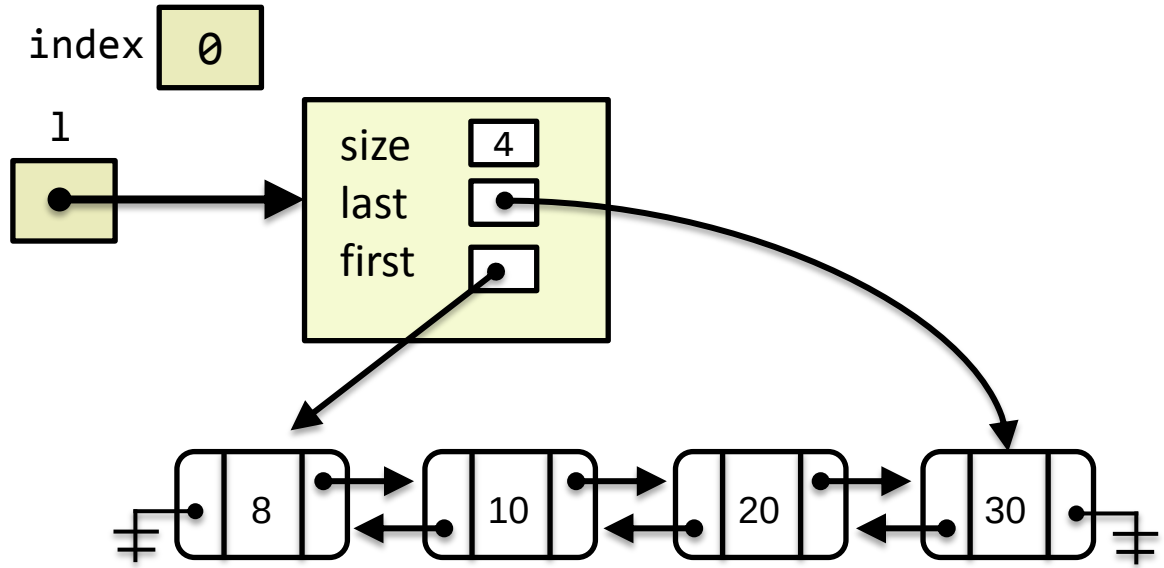


addList

```
int addList(List* l, ItemType e, int index);
```

1 2 3 4 Inserção na primeira posição quando a lista NÃO está vazia

Representação

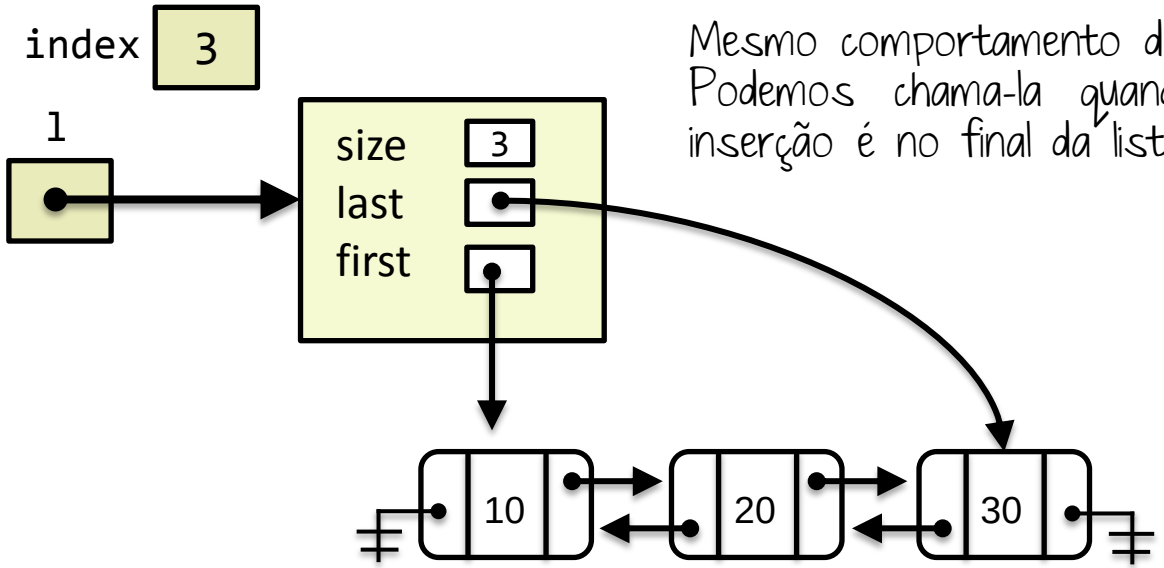


addList

```
int addList(List* l, ItemType e, int index);
```

1 2 3 4 Inserção na última posição da lista

Representação



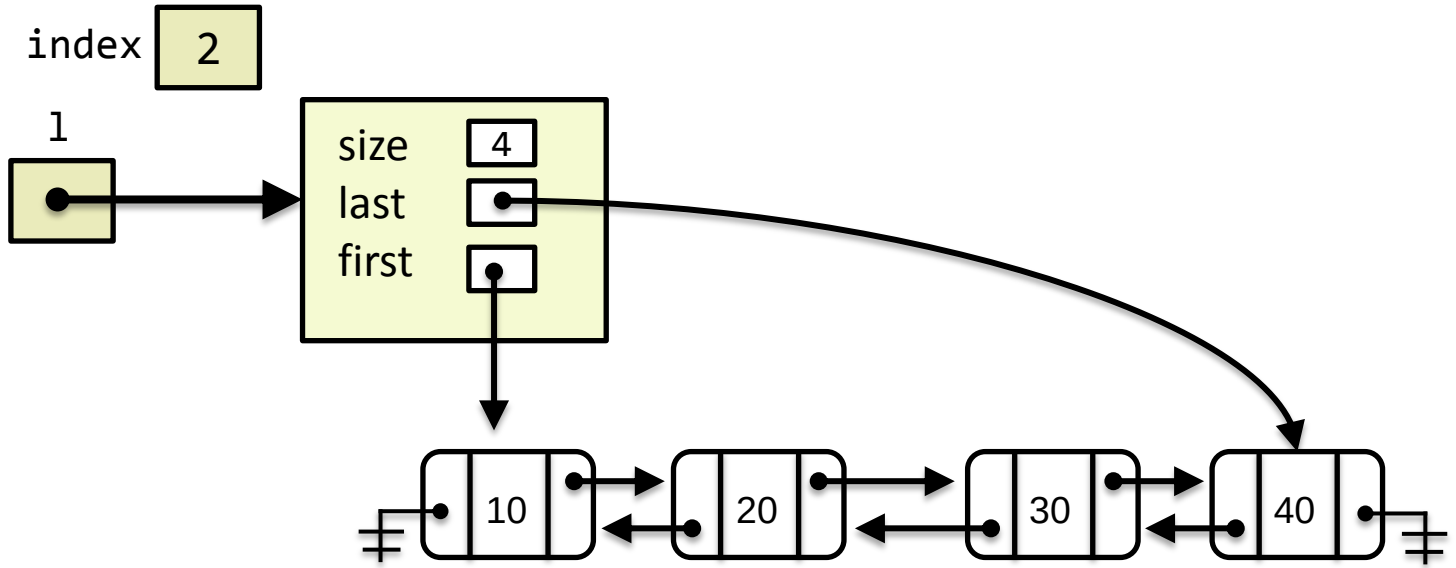
Mesmo comportamento da função **addLastList**
Podemos chama-la quando identificarmos que a inserção é no final da lista

addList

```
int addList(List* l, ItemType e, int index);
```

1 2 3 4 Inserção no meio da lista

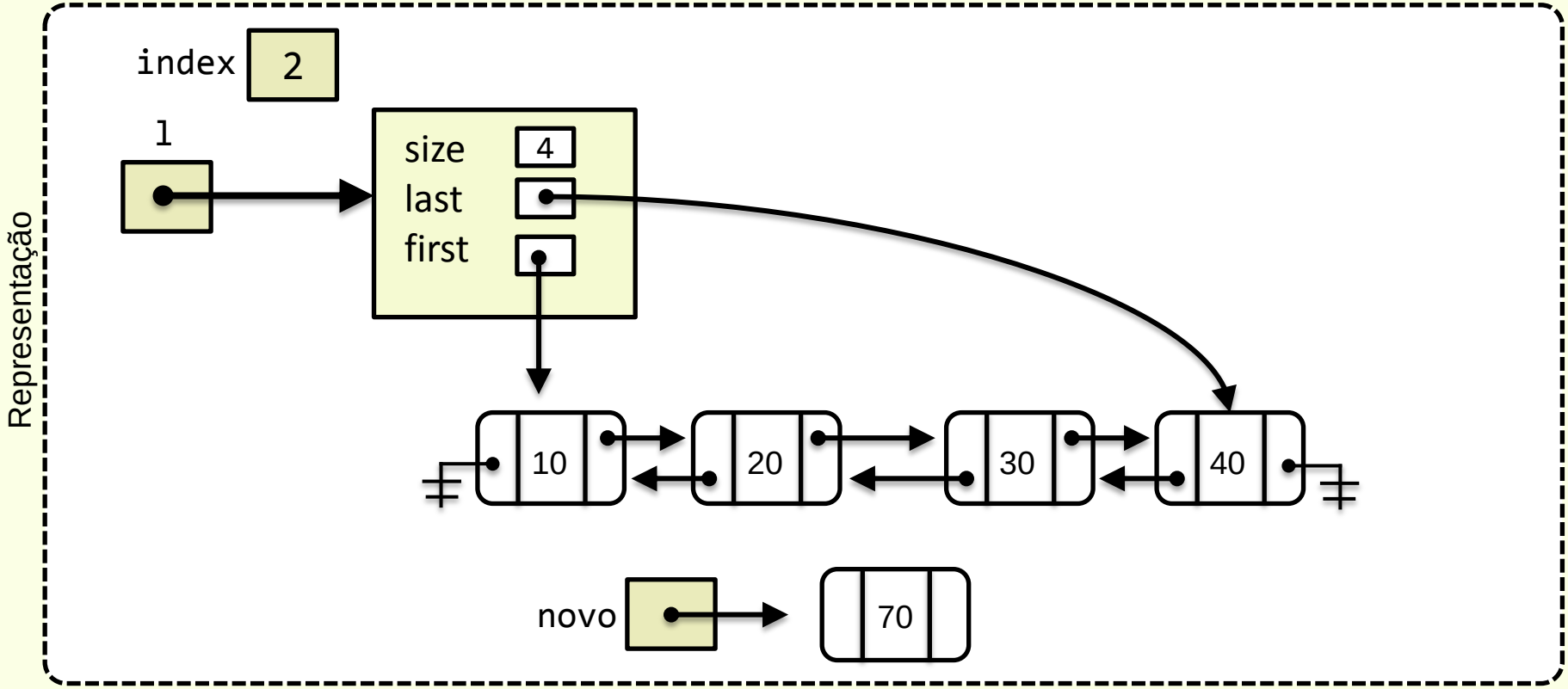
Representação



addList

```
int addList(List* l, ItemType e, int index);
```

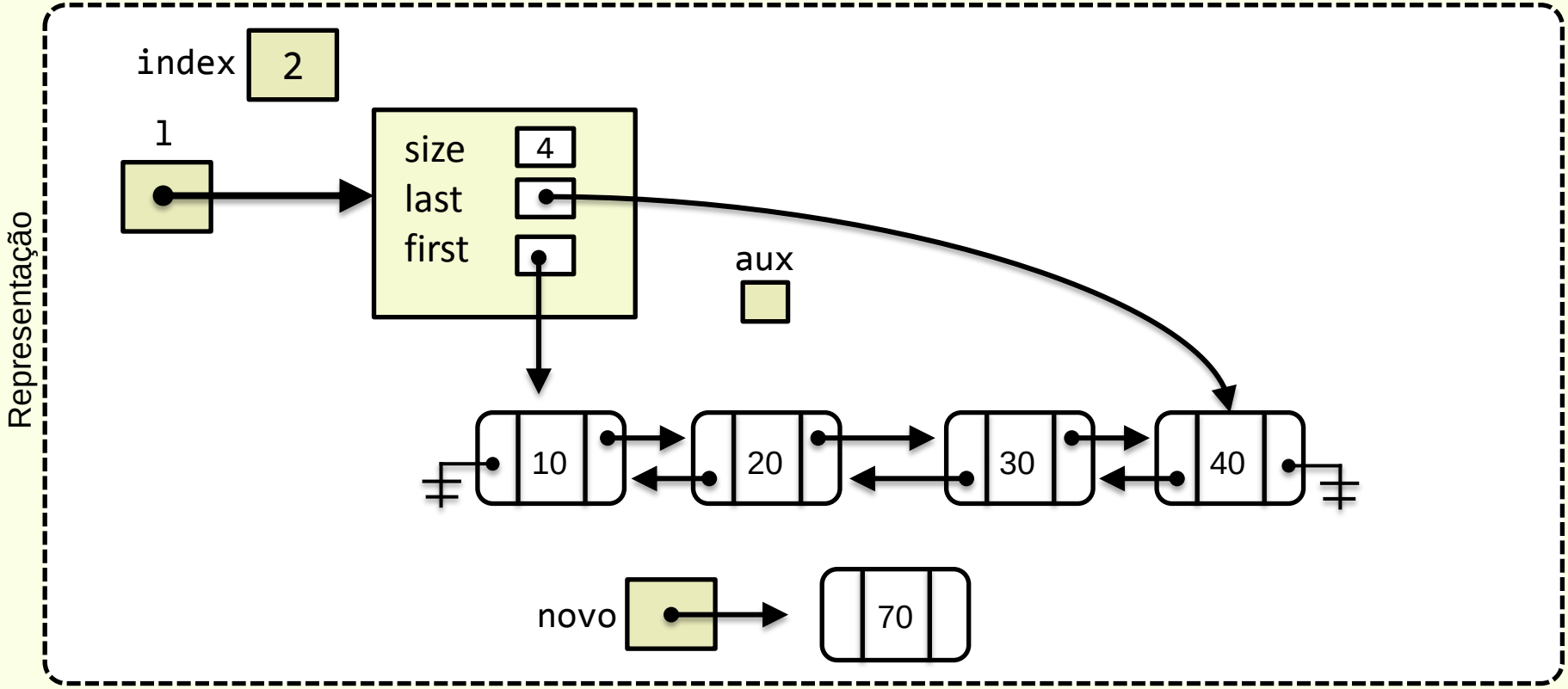
1 2 3 4 Inserção no meio da lista



addList

```
int addList(List* l, ItemType e, int index);
```

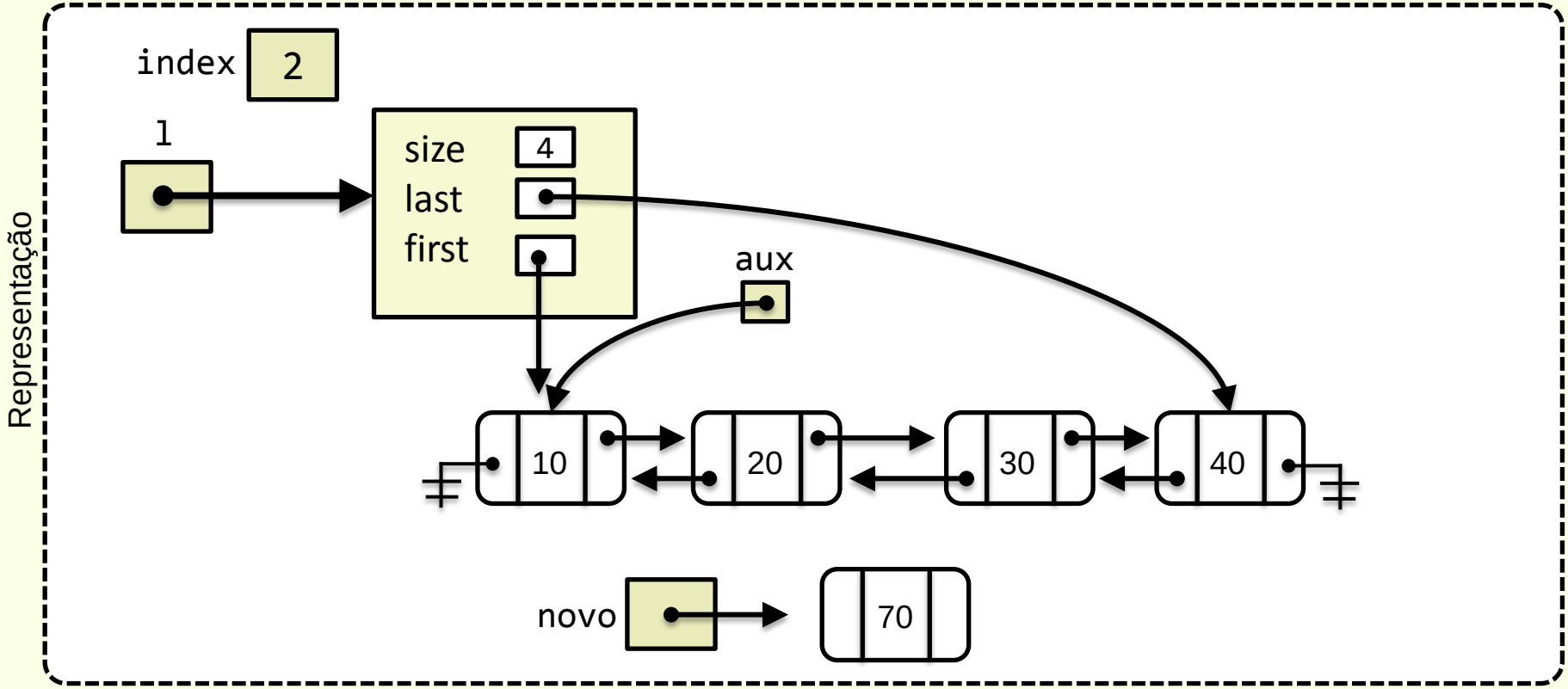
1 2 3 4 Inserção no meio da lista



addList

```
int addList(List* l, ItemType e, int index);
```

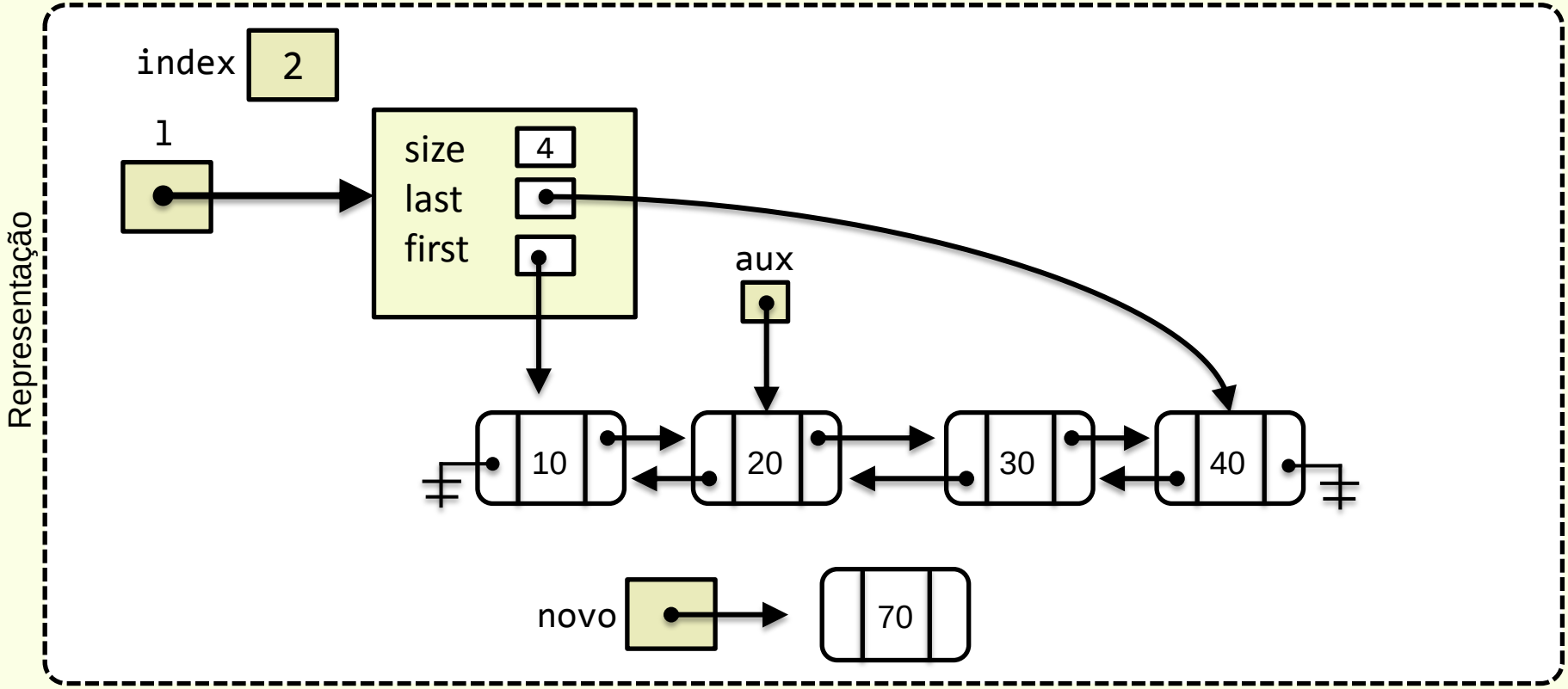
1 2 3 4 Inserção no meio da lista



addList

```
int addList(List* l, ItemType e, int index);
```

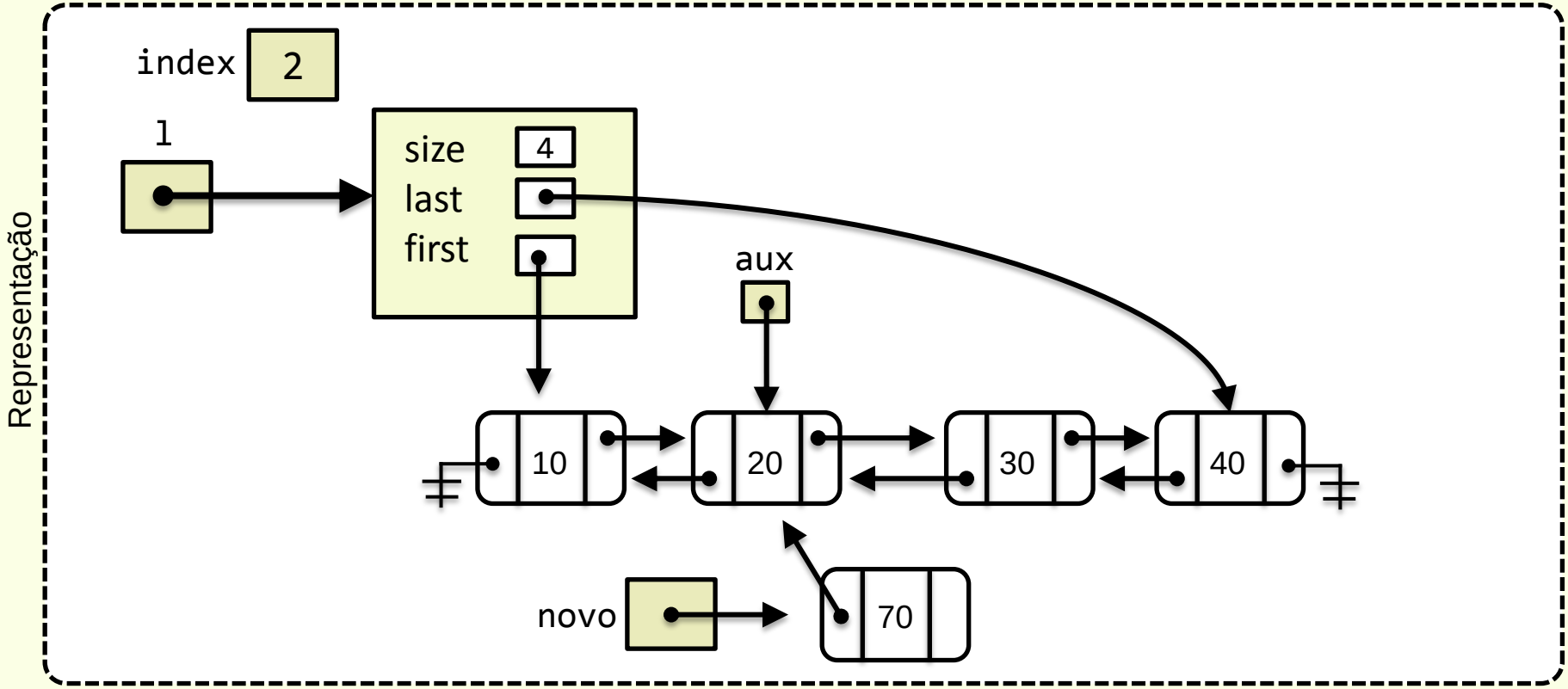
1 2 3 4 Inserção no meio da lista



addList

```
int addList(List* l, ItemType e, int index);
```

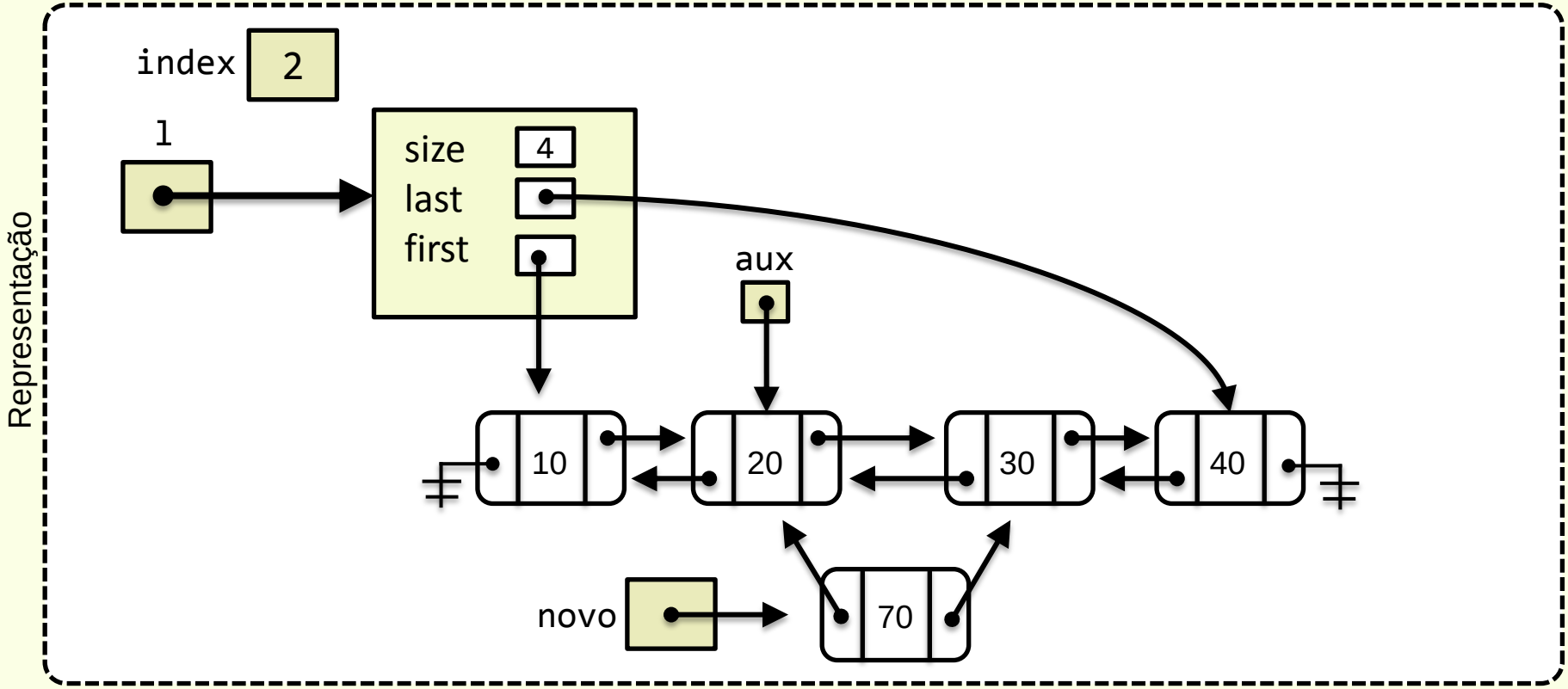
1 2 3 4 Inserção no meio da lista



addList

```
int addList(List* l, ItemType e, int index);
```

1 2 3 4 Inserção no meio da lista

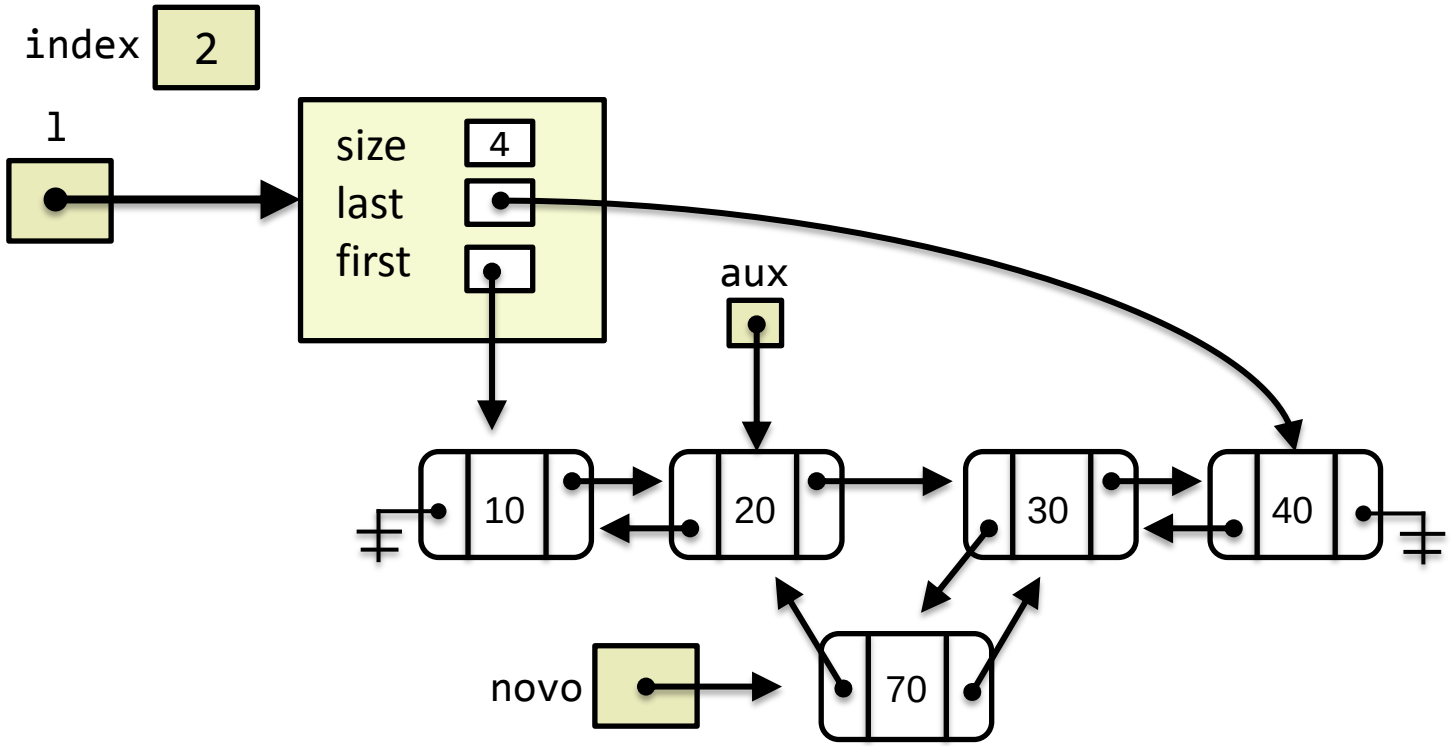


addList

```
int addList(List* l, ItemType e, int index);
```

1 2 3 4 Inserção no meio da lista

Representação

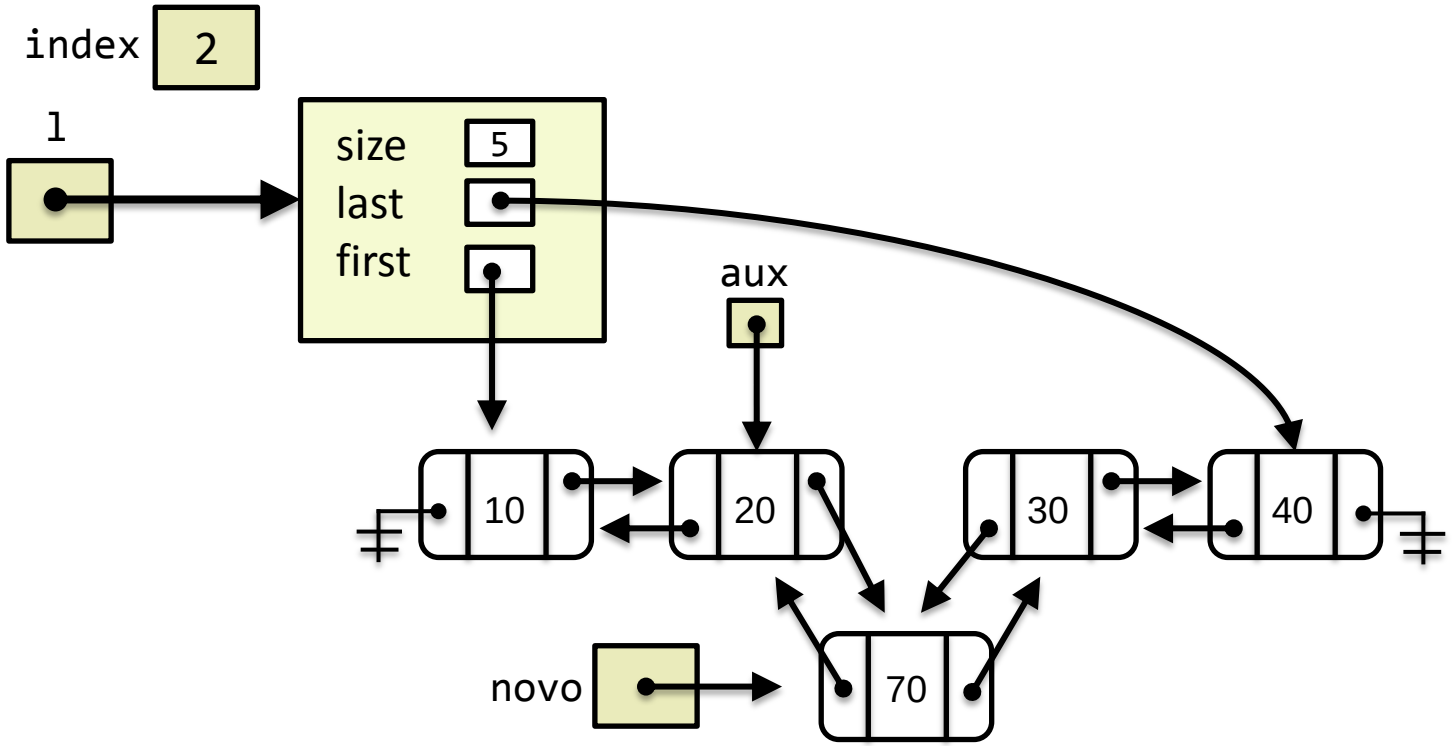


addList

```
int addList(List* l, ItemType e, int index);
```

1 2 3 4 Inserção no meio da lista

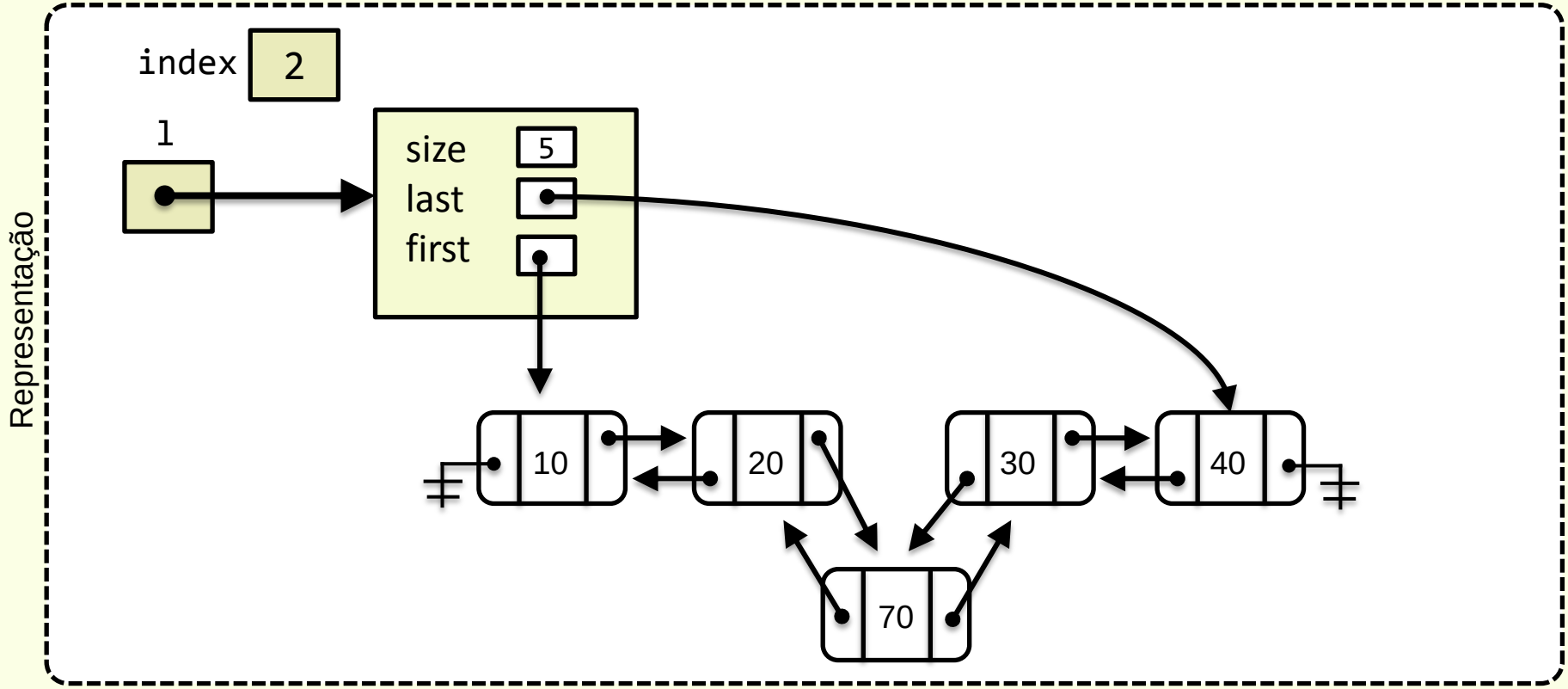
Representação



addList

```
int addList(List* l, ItemType e, int index);
```

1 2 3 4 Inserção no meio da lista



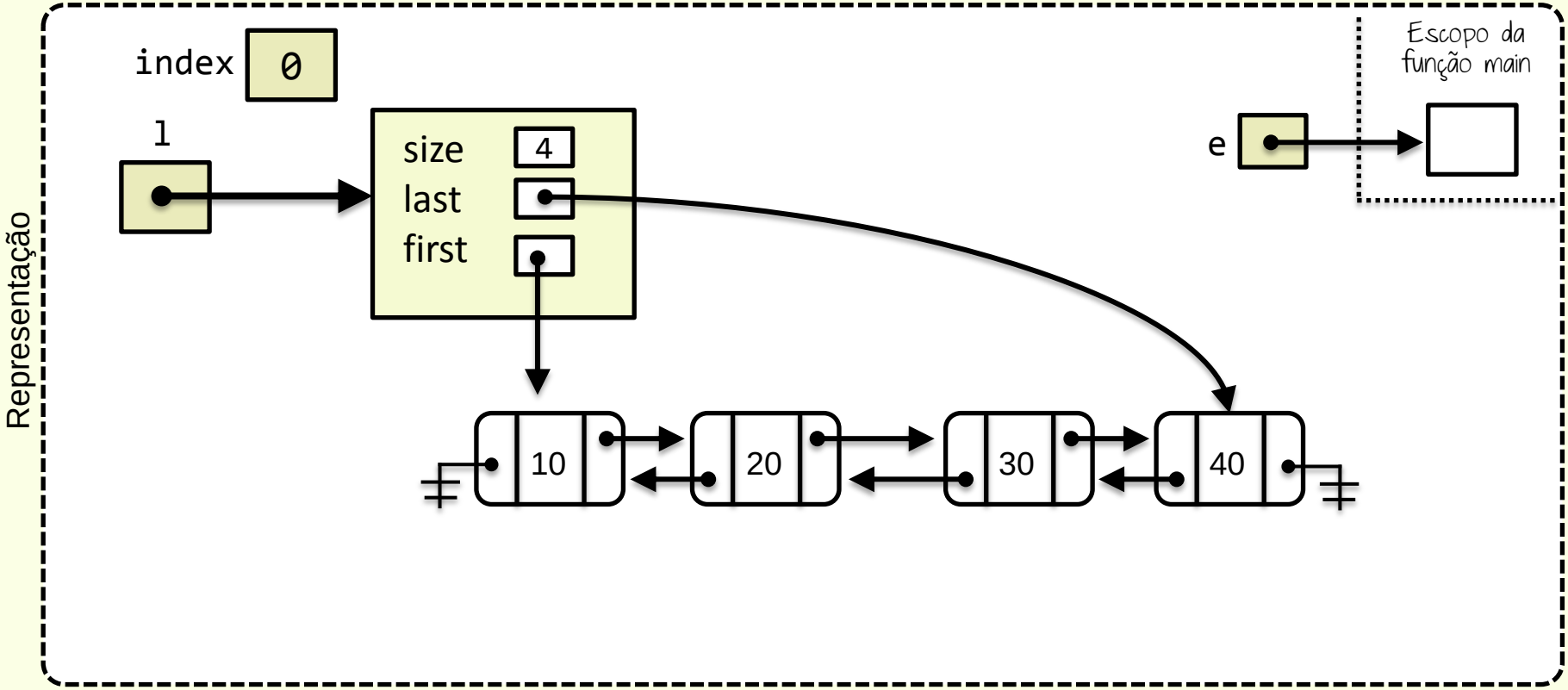
```
List *createList ();
void initializeList(List *l);
int addLastList(List *l, ItemType e);
int addList(List* l, ItemType e, int index);
int removeList(List* l, int index, ItemType *e);
int removeElementList(List* l, ItemType* e);
```


removeList

removeList

```
int removeList(List* l, int index, ItemType *e);
```

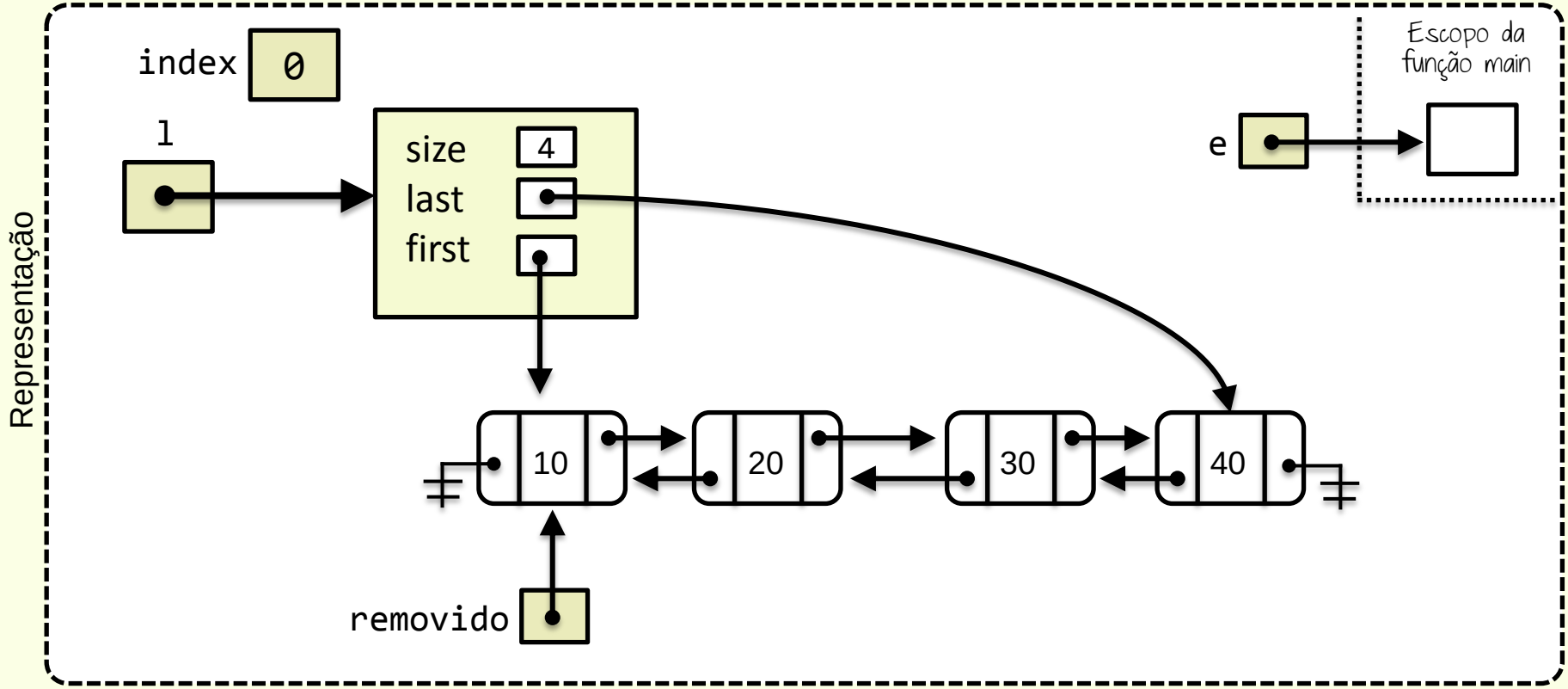
1 2 3 Remoção do primeiro elemento da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

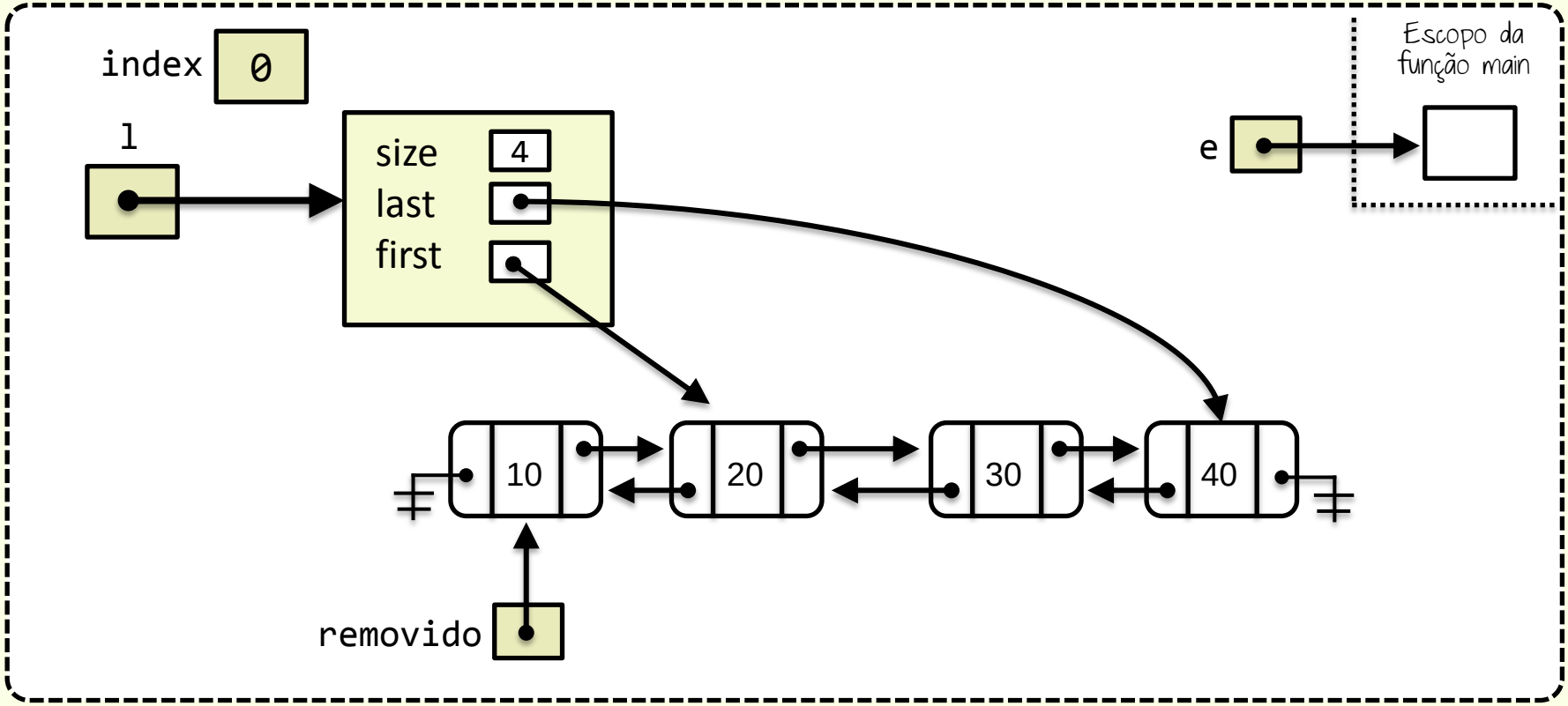
1 2 3 Remoção do primeiro elemento da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

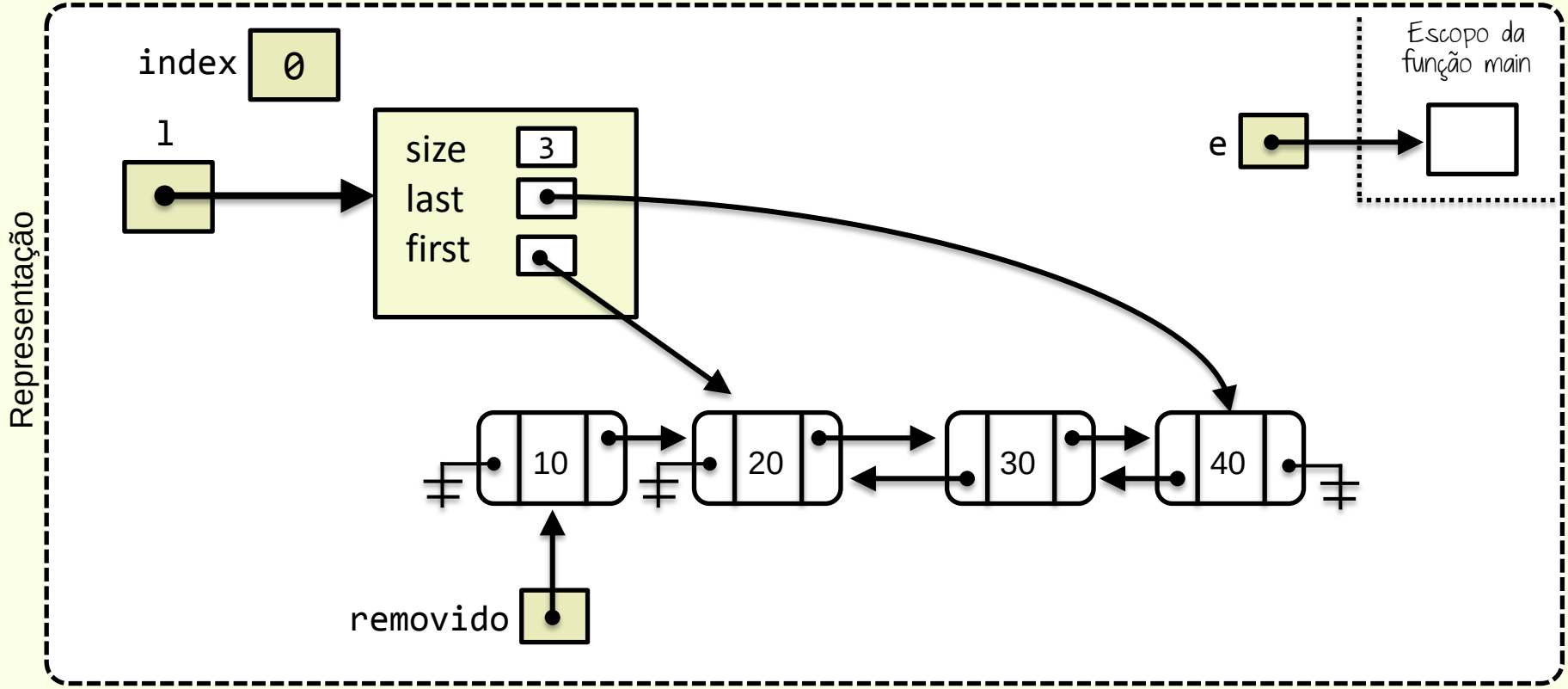
1 2 3 Remoção do primeiro elemento da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

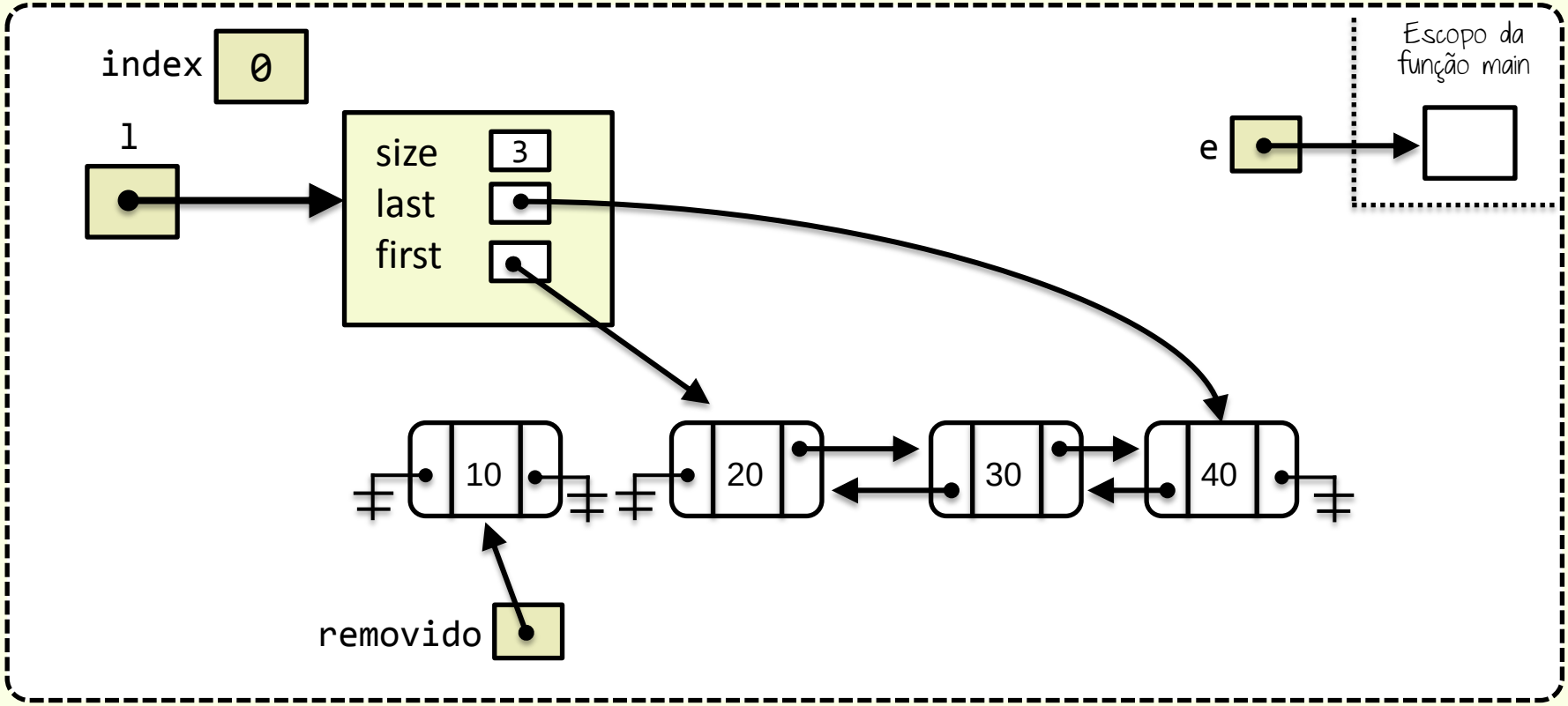
1 2 3 Remoção do primeiro elemento da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

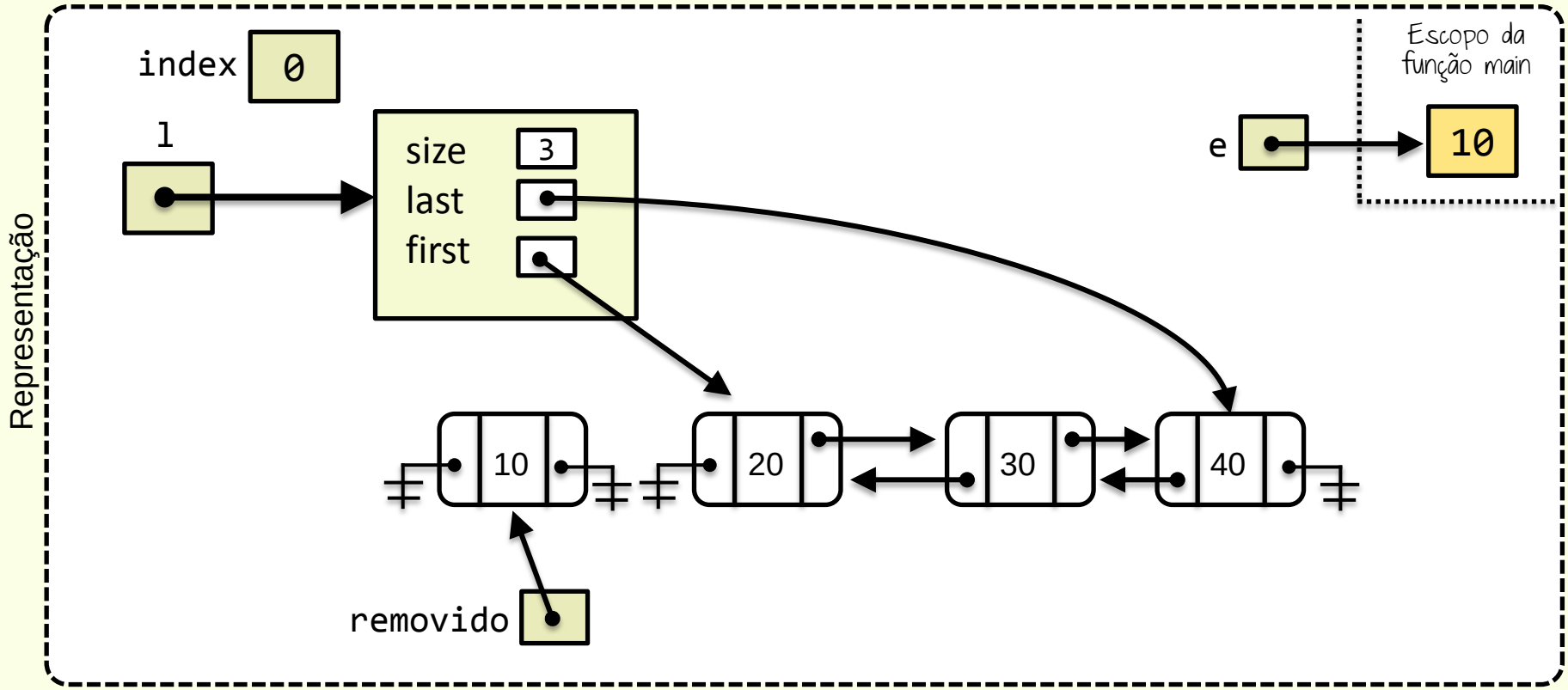
1 2 3 Remoção do primeiro elemento da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

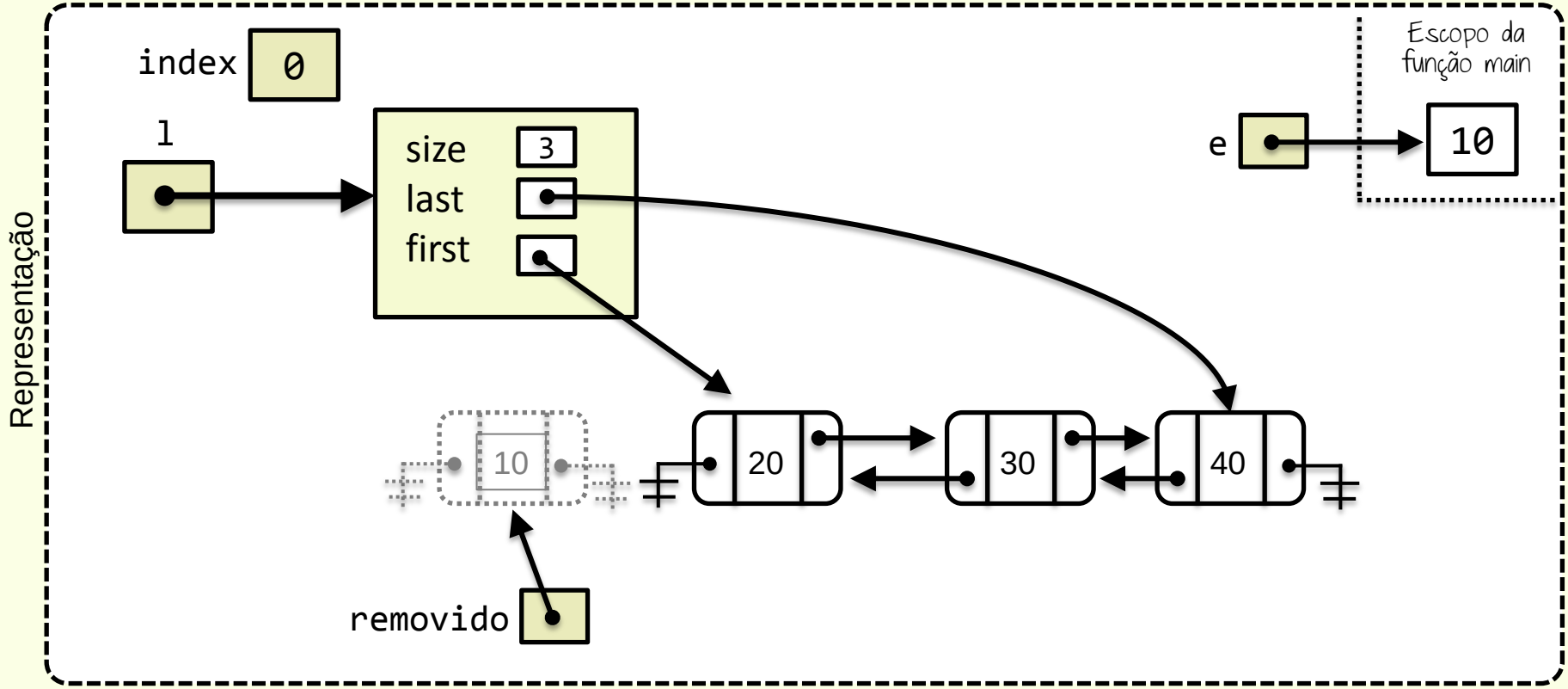
1 2 3 Remoção do primeiro elemento da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

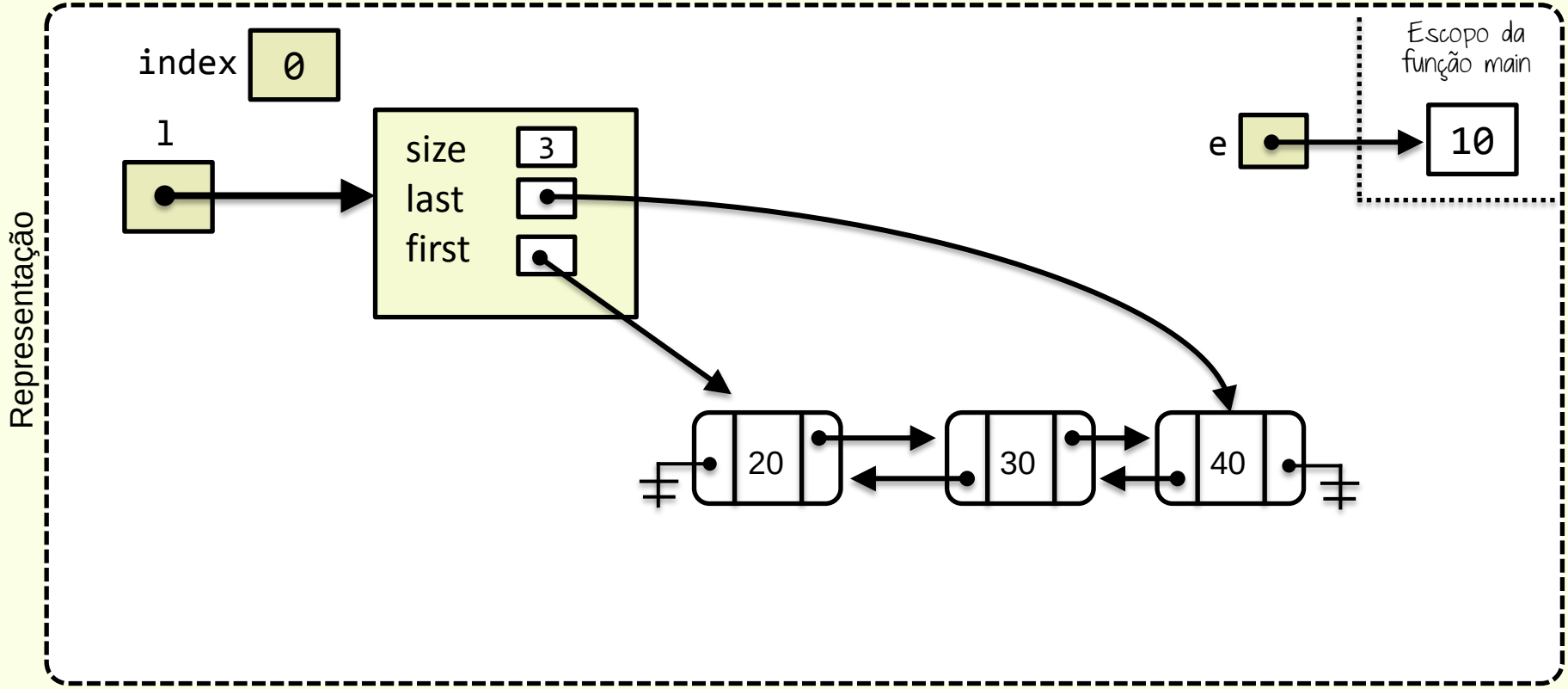
1 2 3 Remoção do primeiro elemento da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

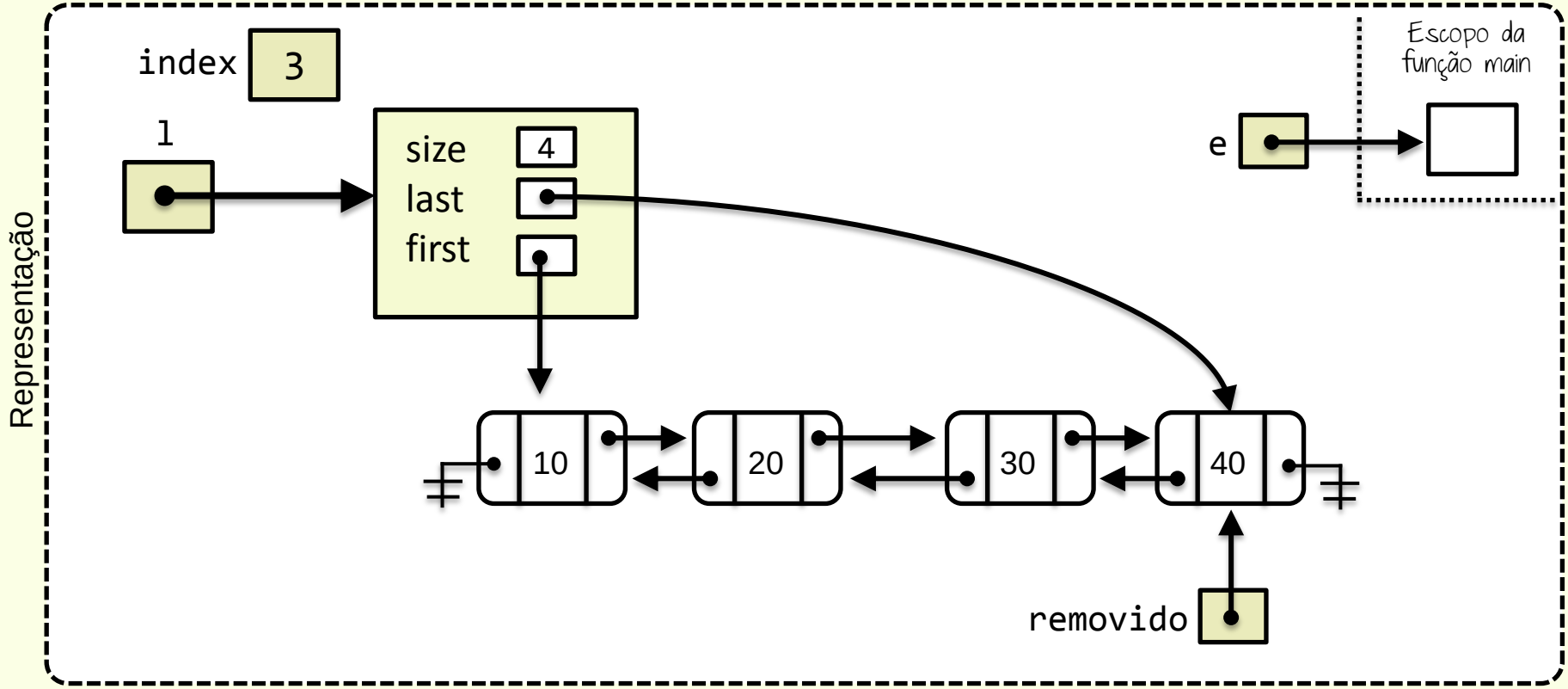
1 2 3 Remoção do primeiro elemento da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

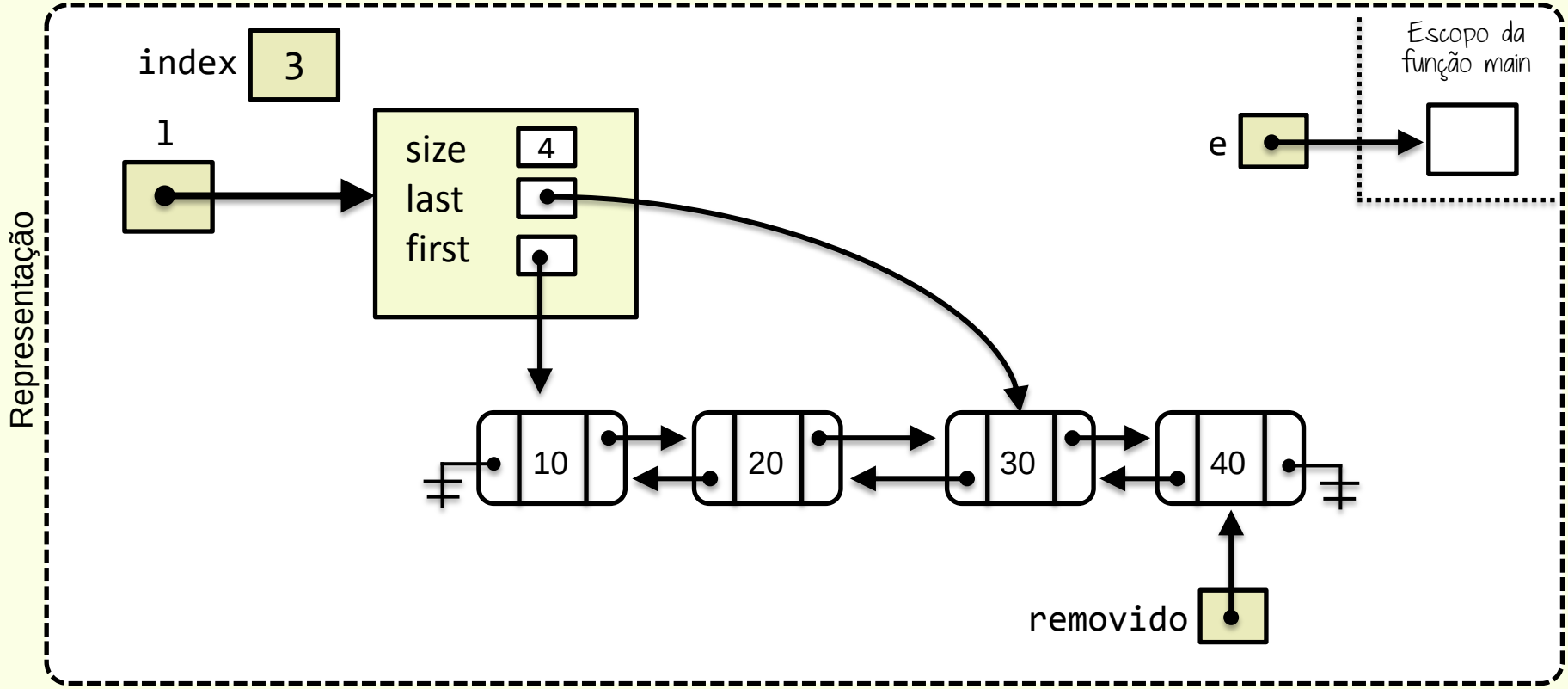
1 2 3 Remoção do último elemento da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

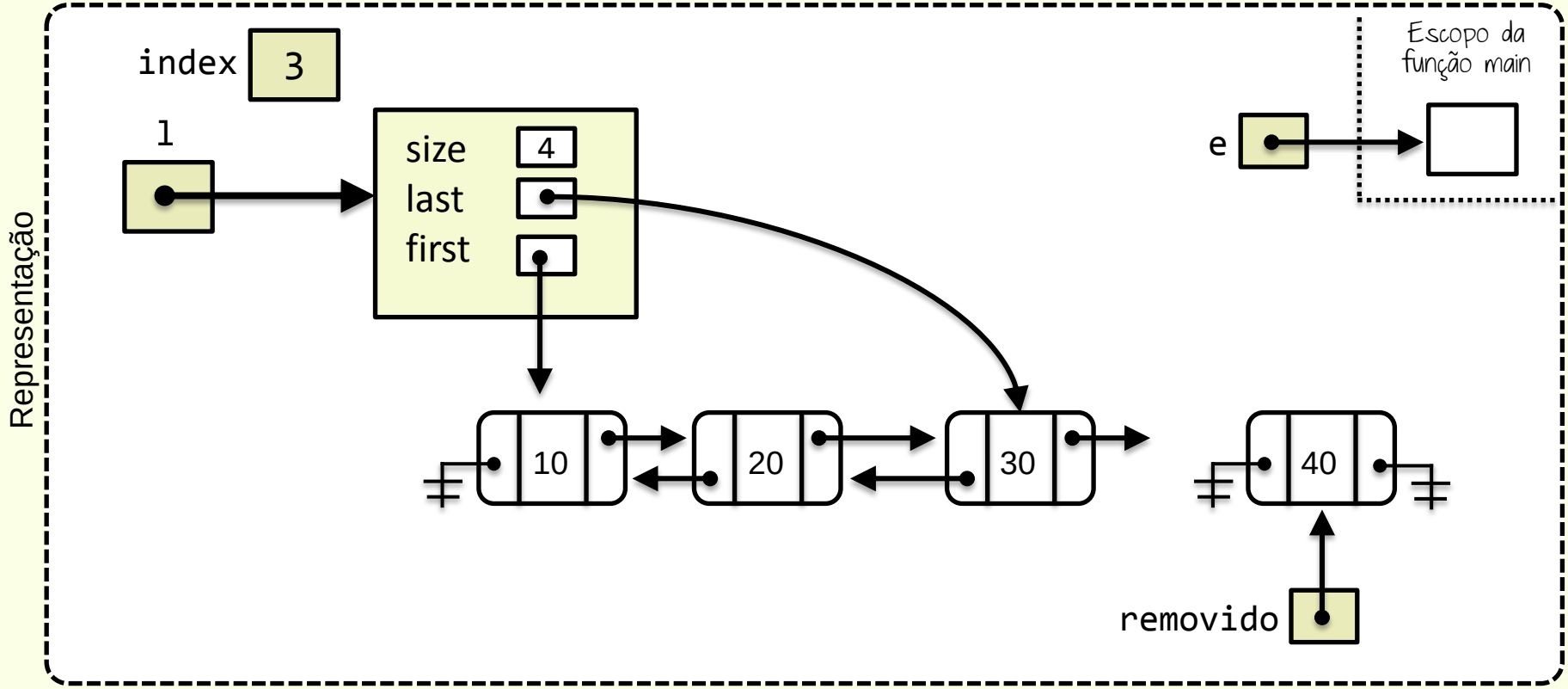
1 2 3 Remoção do último elemento da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

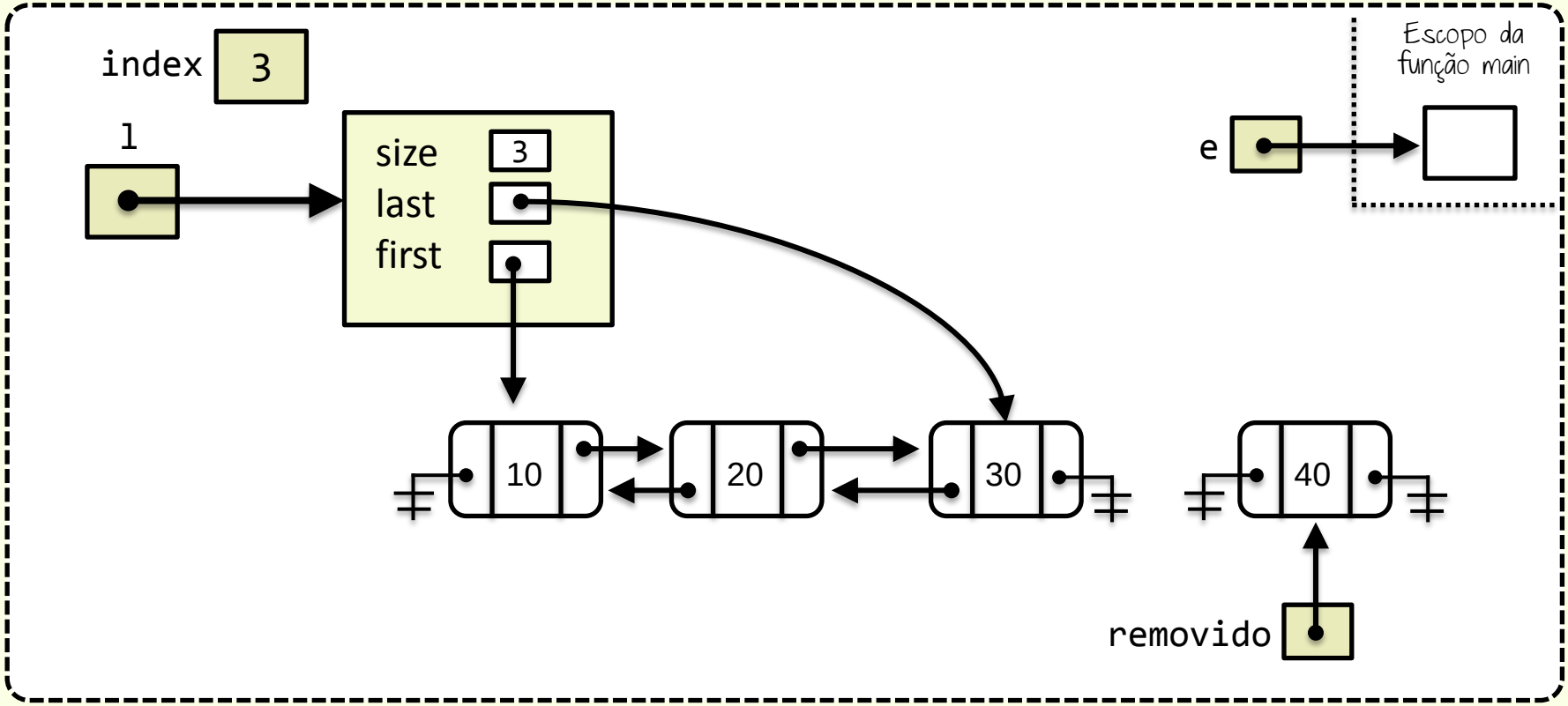
1 2 3 Remoção do último elemento da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

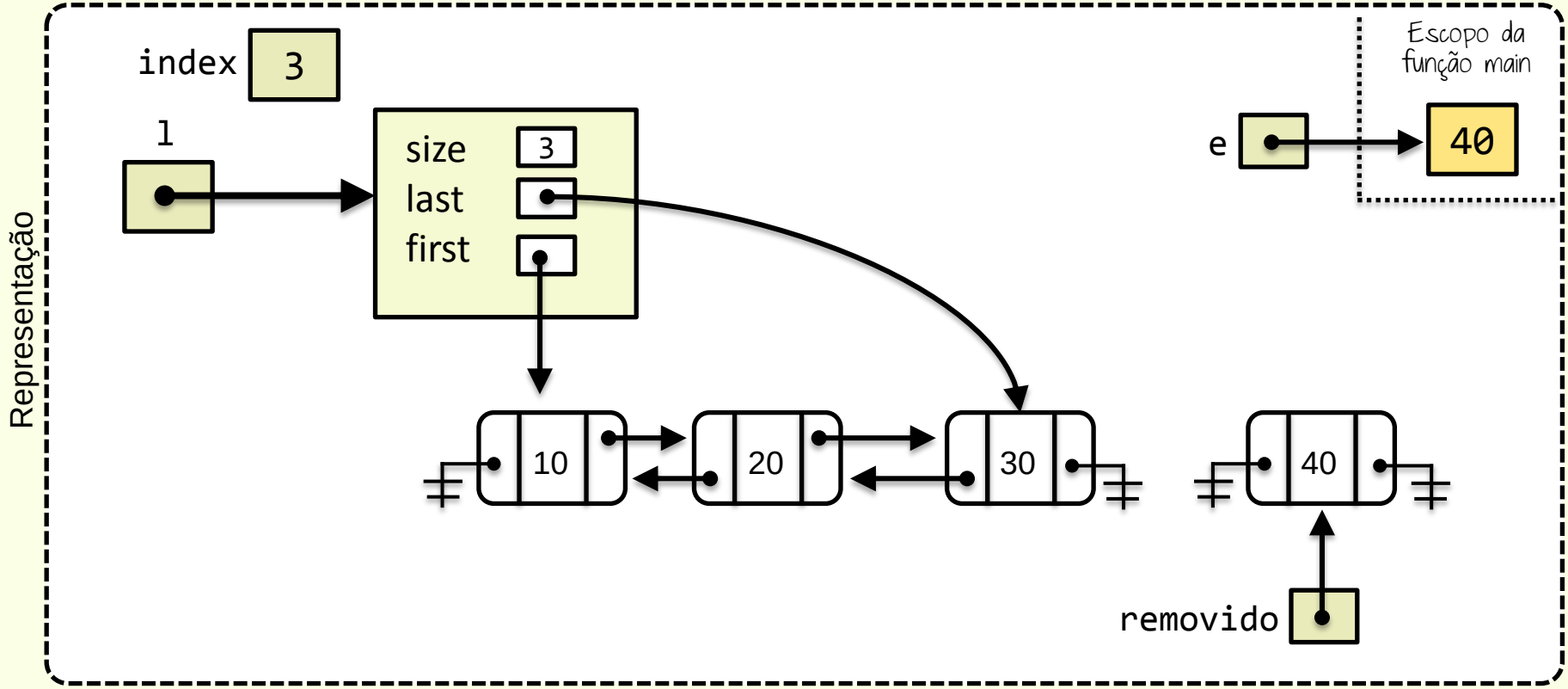
1 2 3 Remoção do último elemento da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

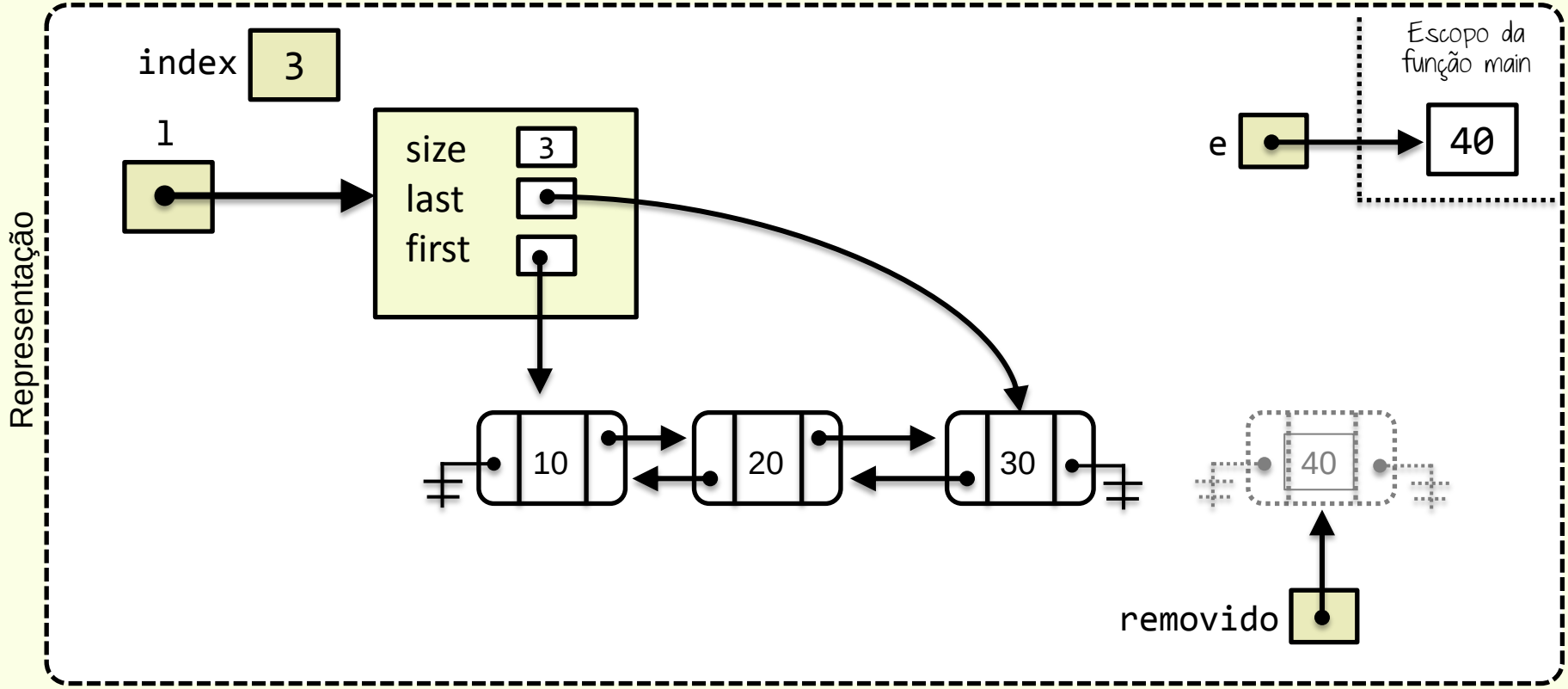
1 2 3 Remoção do último elemento da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

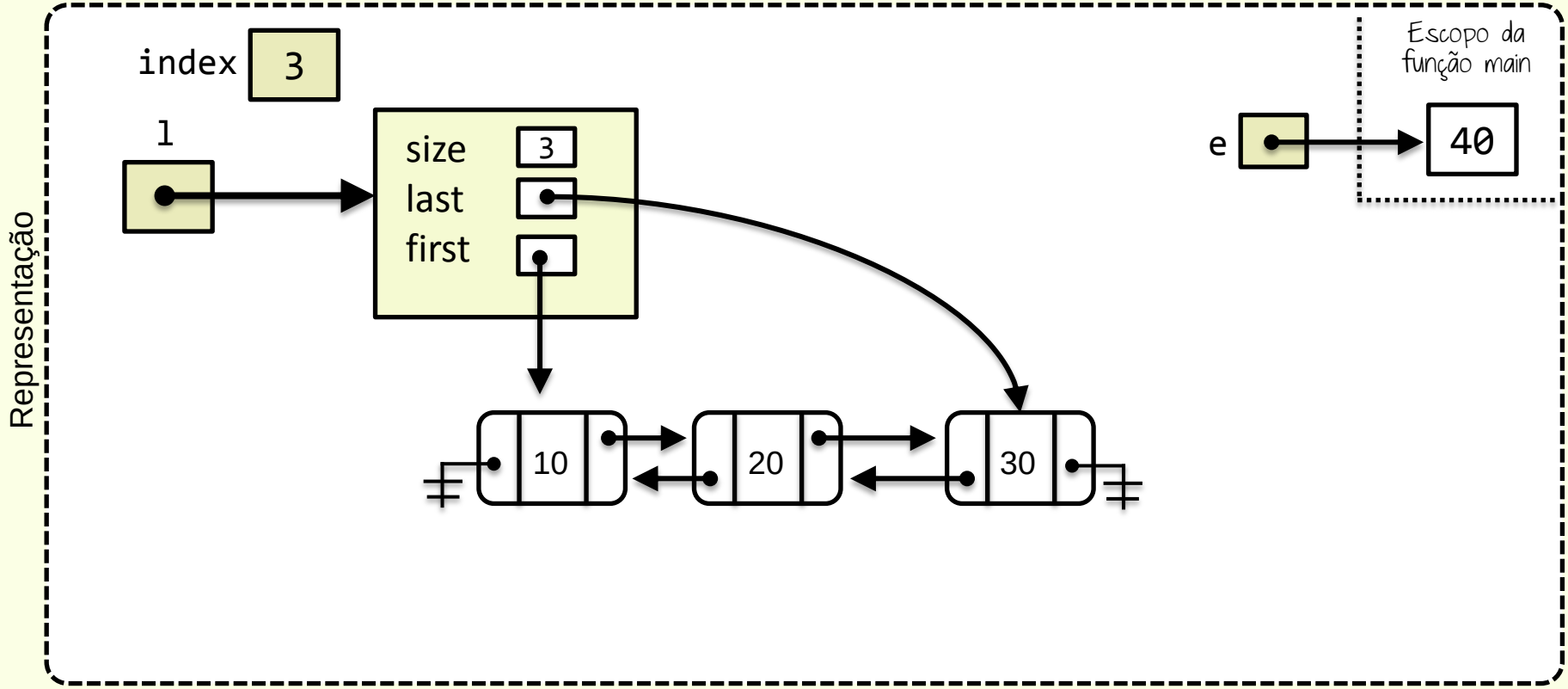
1 2 3 Remoção do último elemento da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

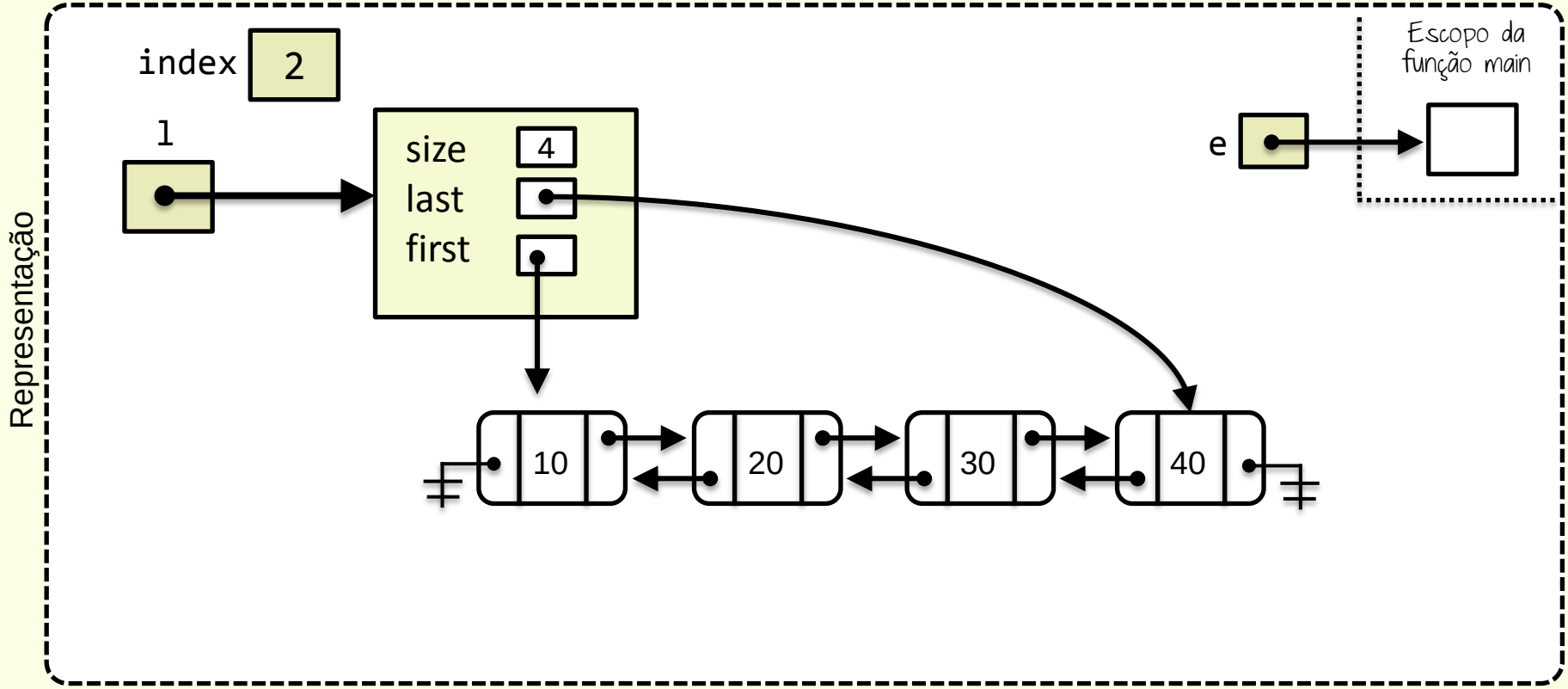
1 2 3 Remoção do último elemento da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

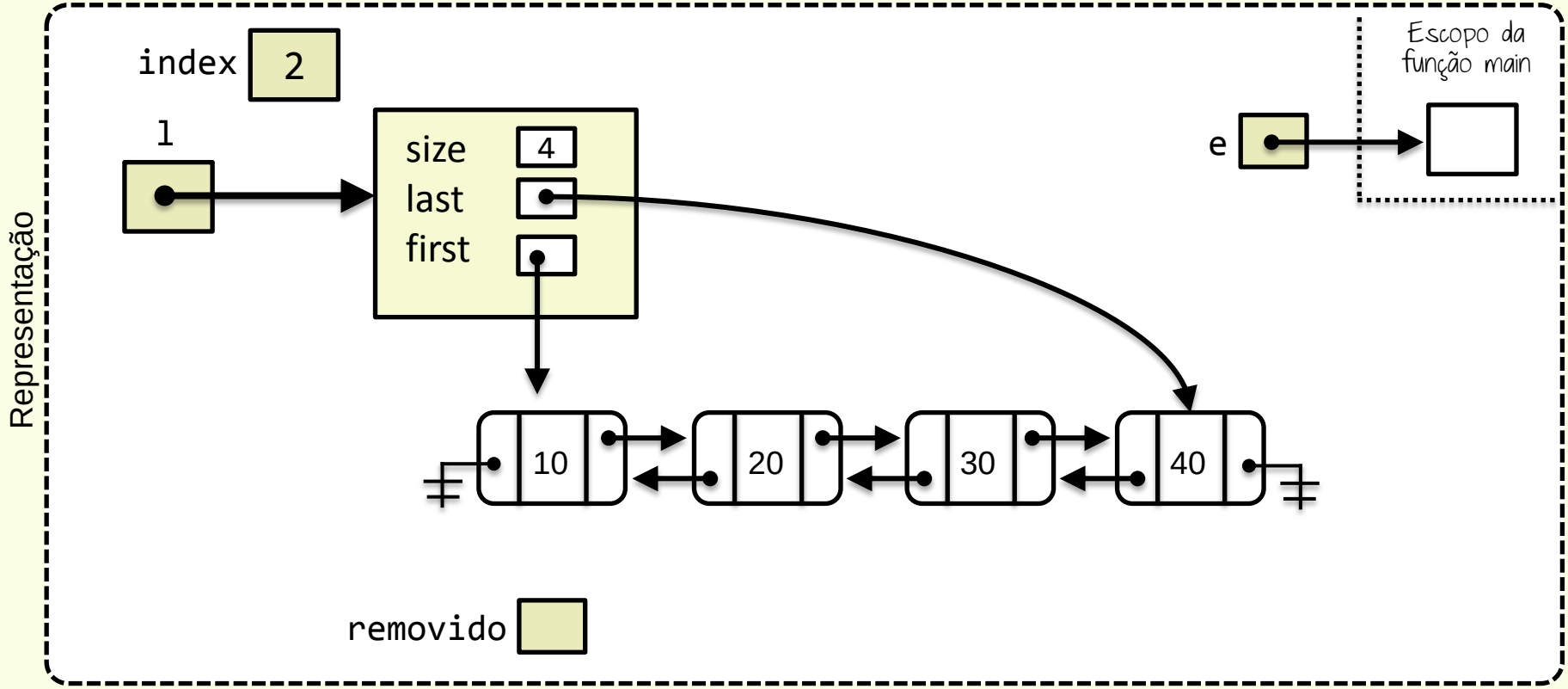
1 2 3 Remoção no meio da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

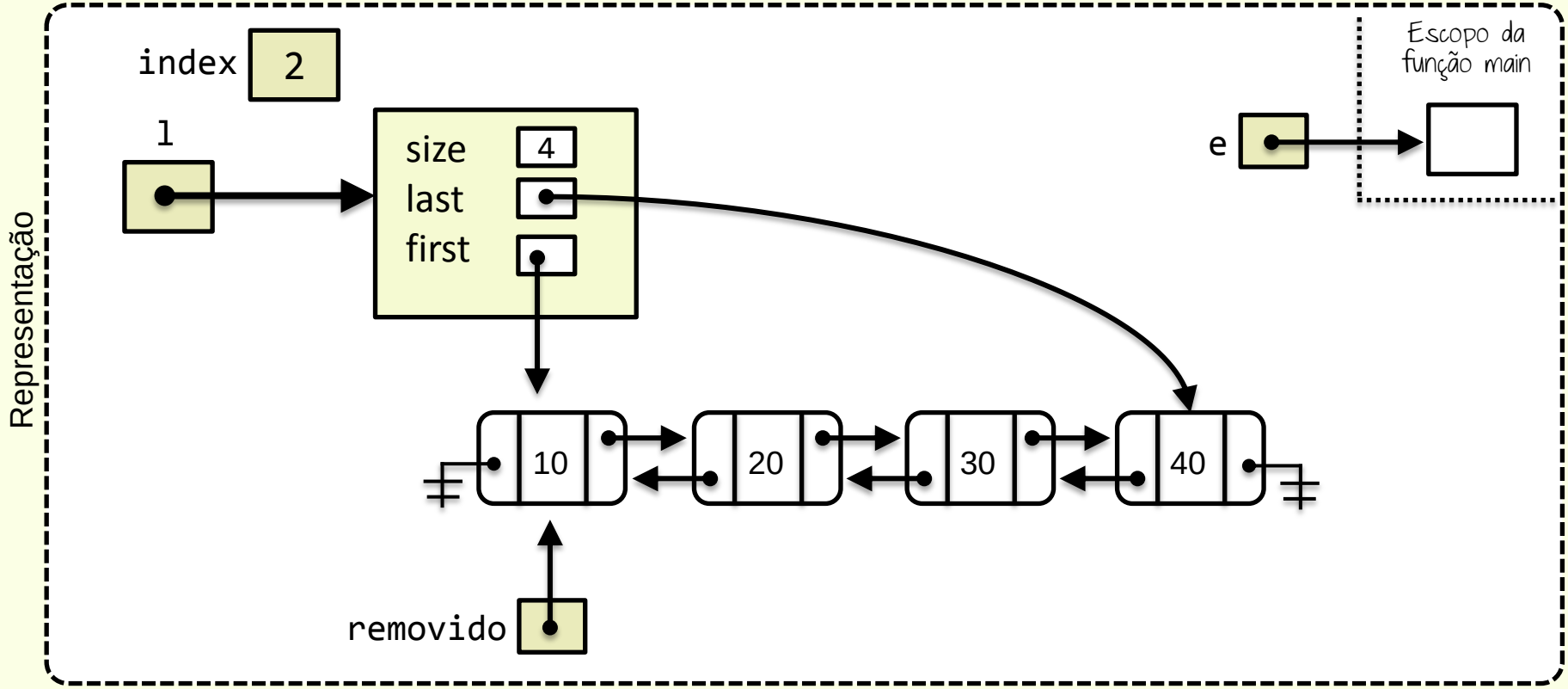
1 2 3 Remoção no meio da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

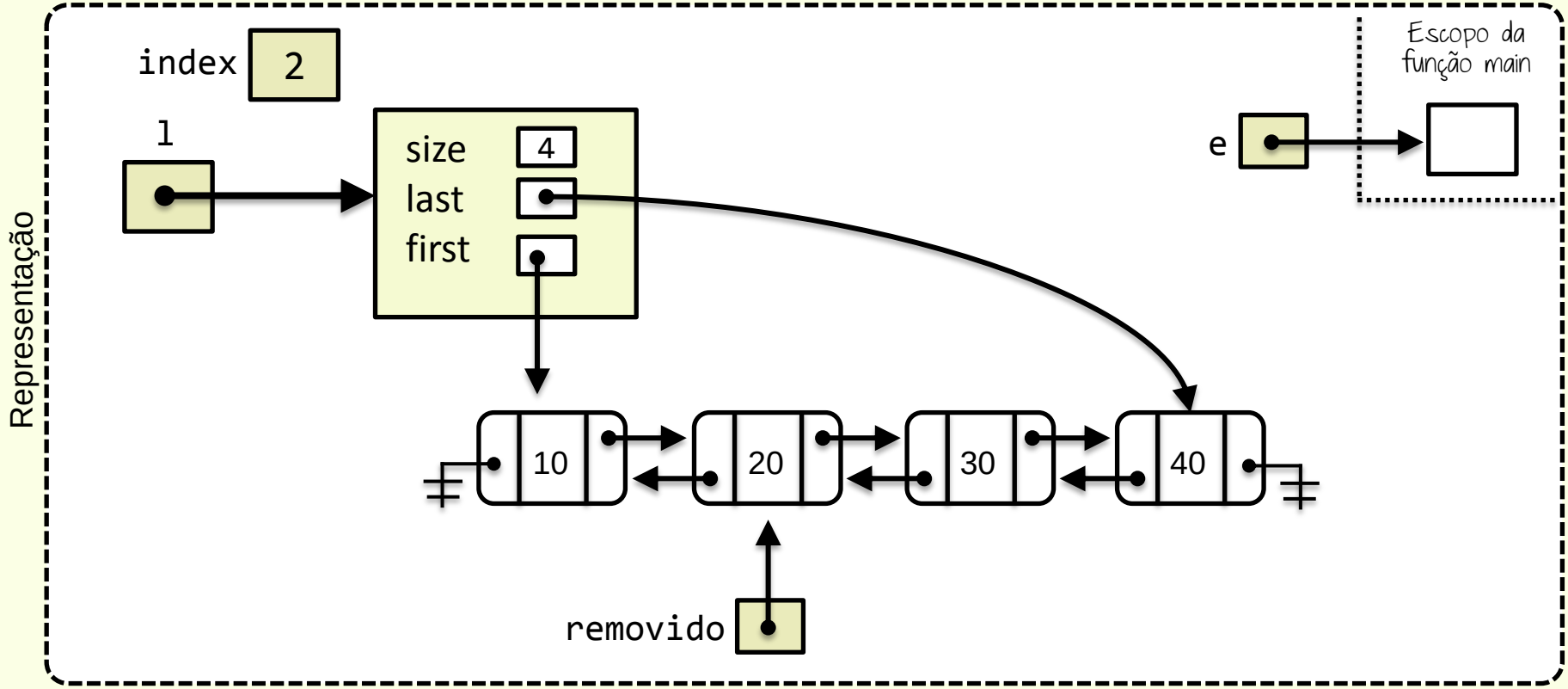
1 2 3 Remoção no meio da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

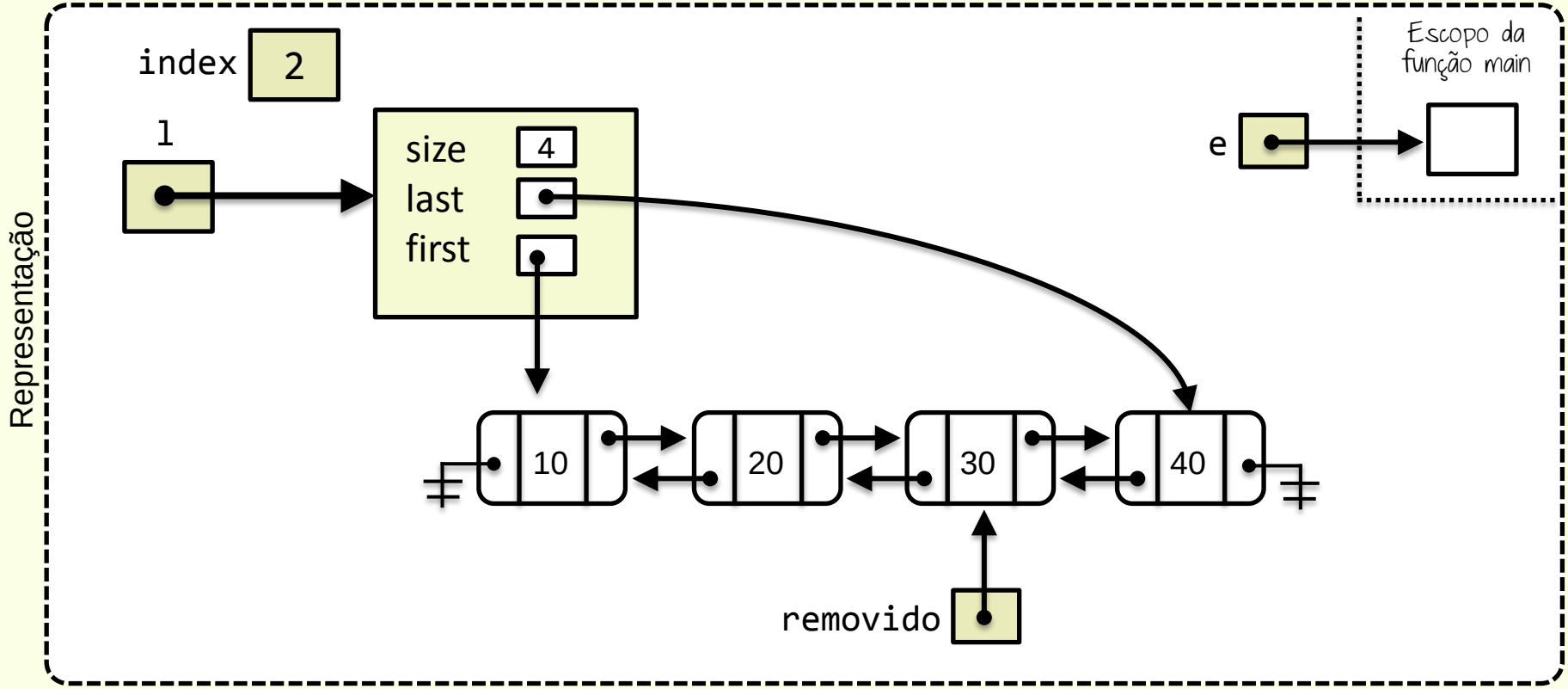
1 2 3 Remoção no meio da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

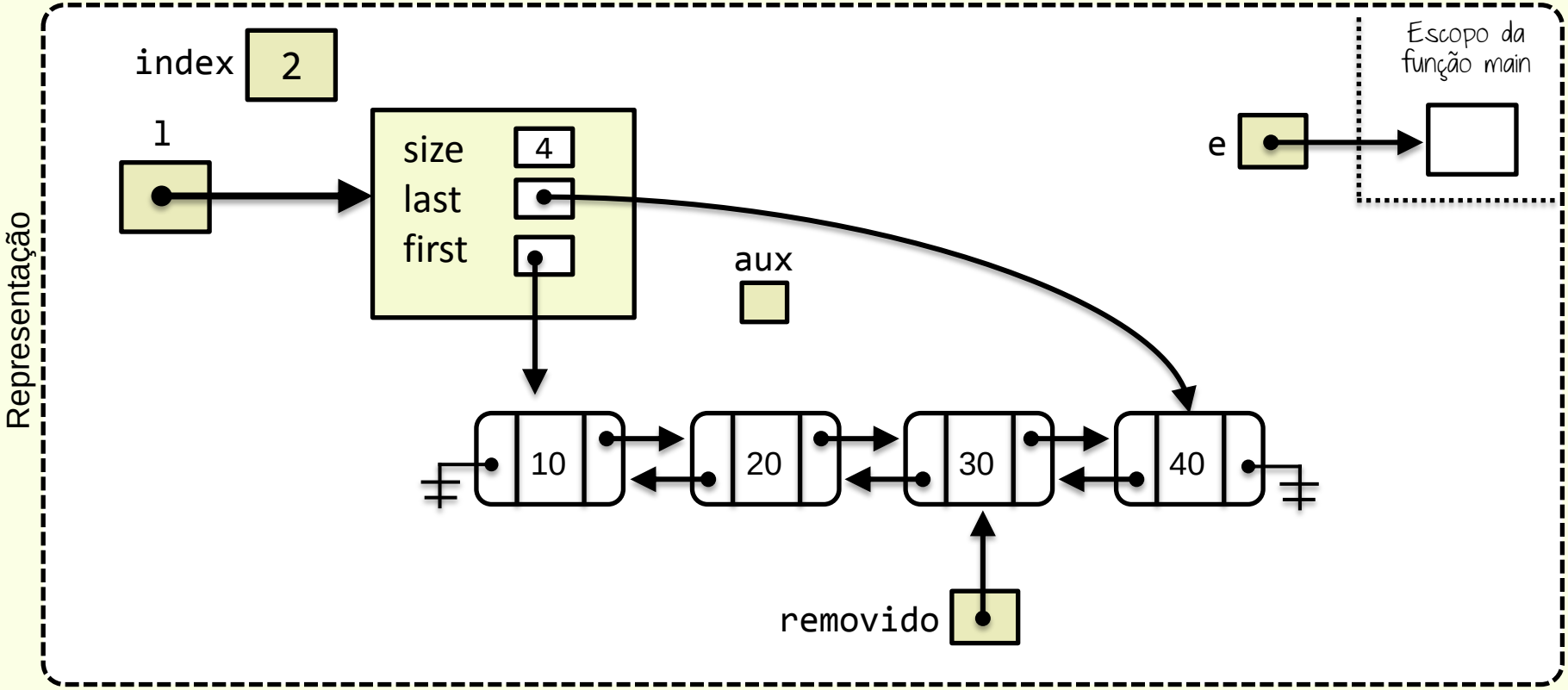
1 2 3 Remoção no meio da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

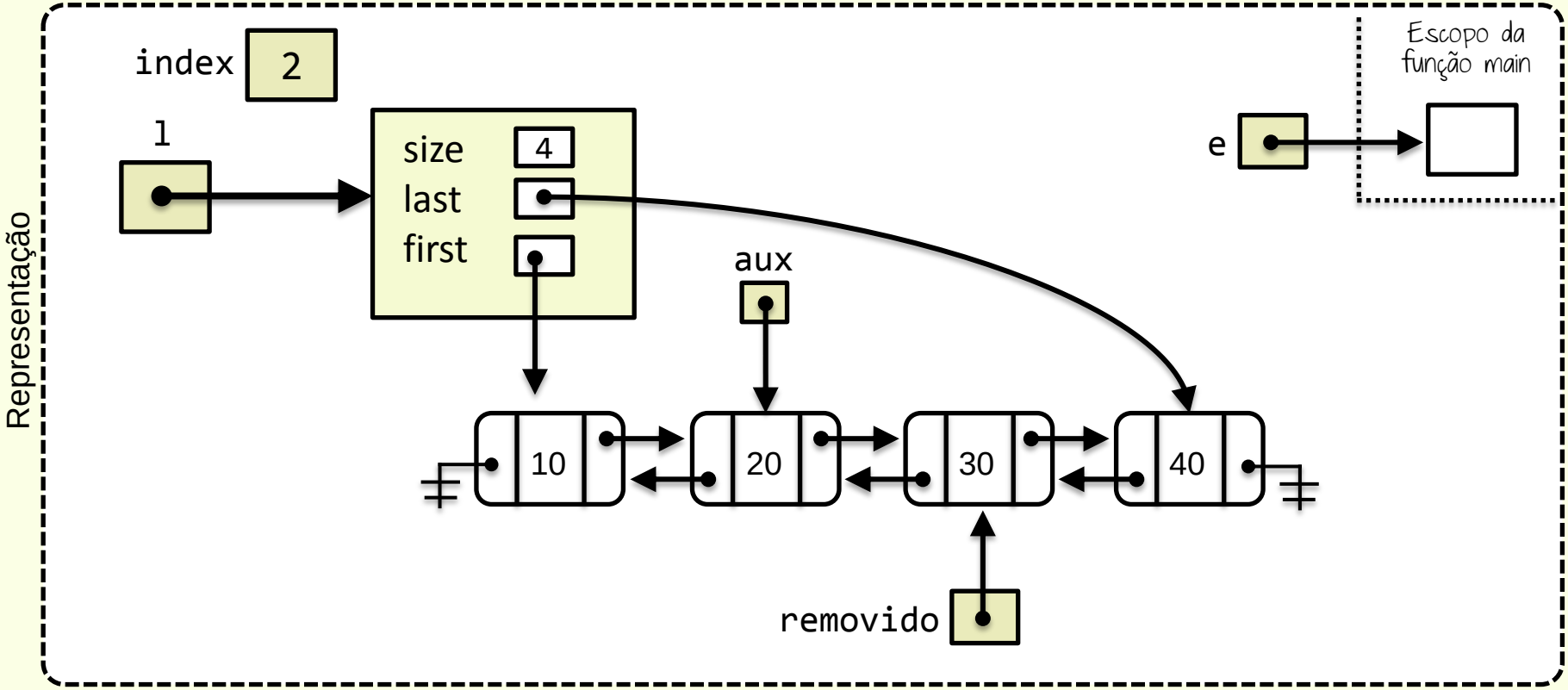
1 2 3 Remoção no meio da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

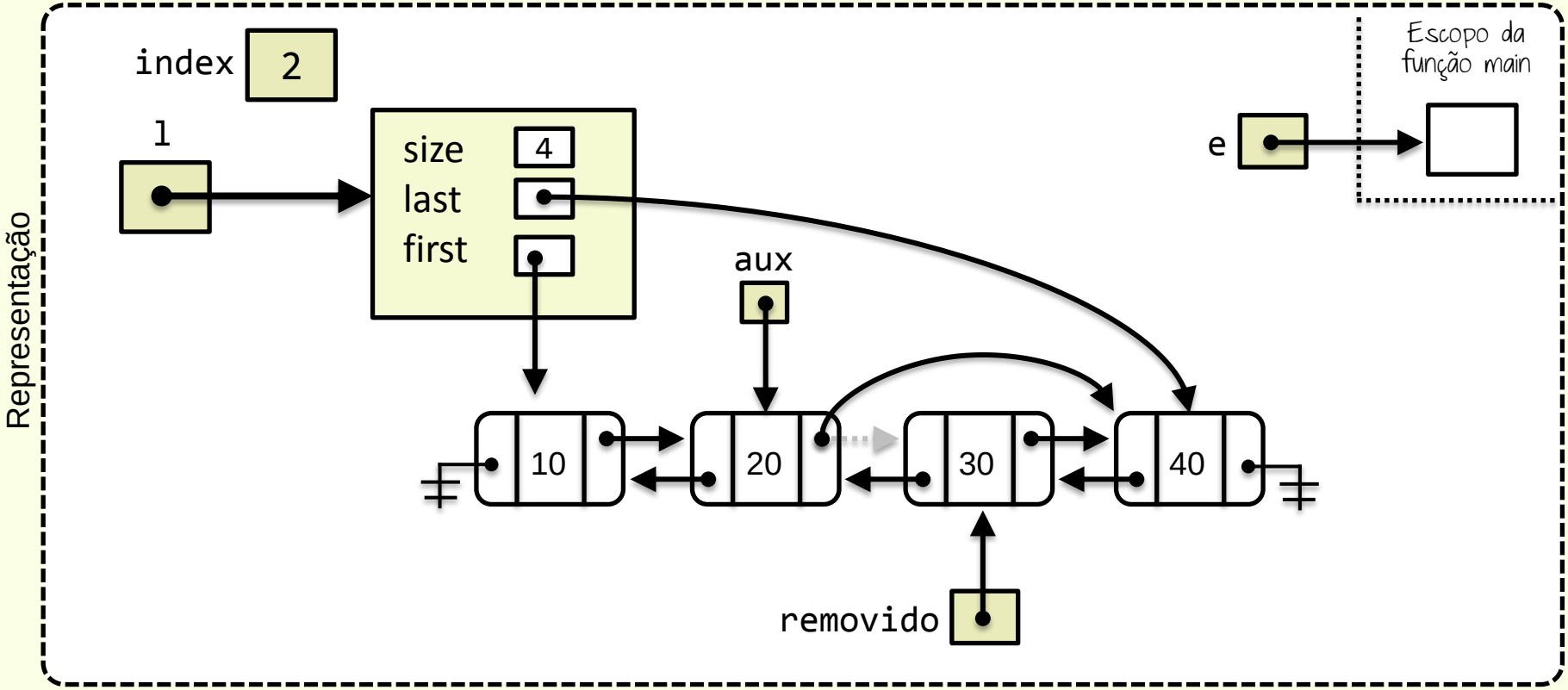
1 2 3 Remoção no meio da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

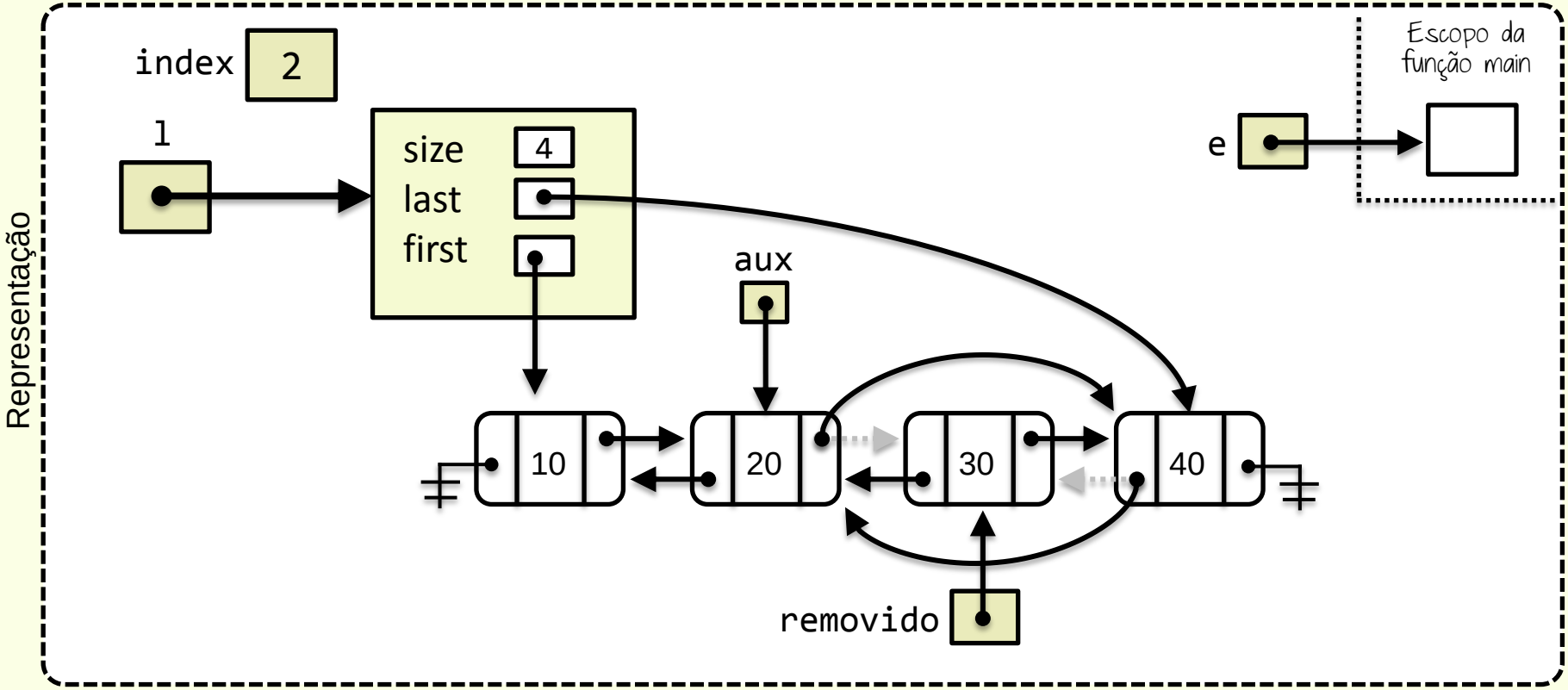
1 2 3 Remoção no meio da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

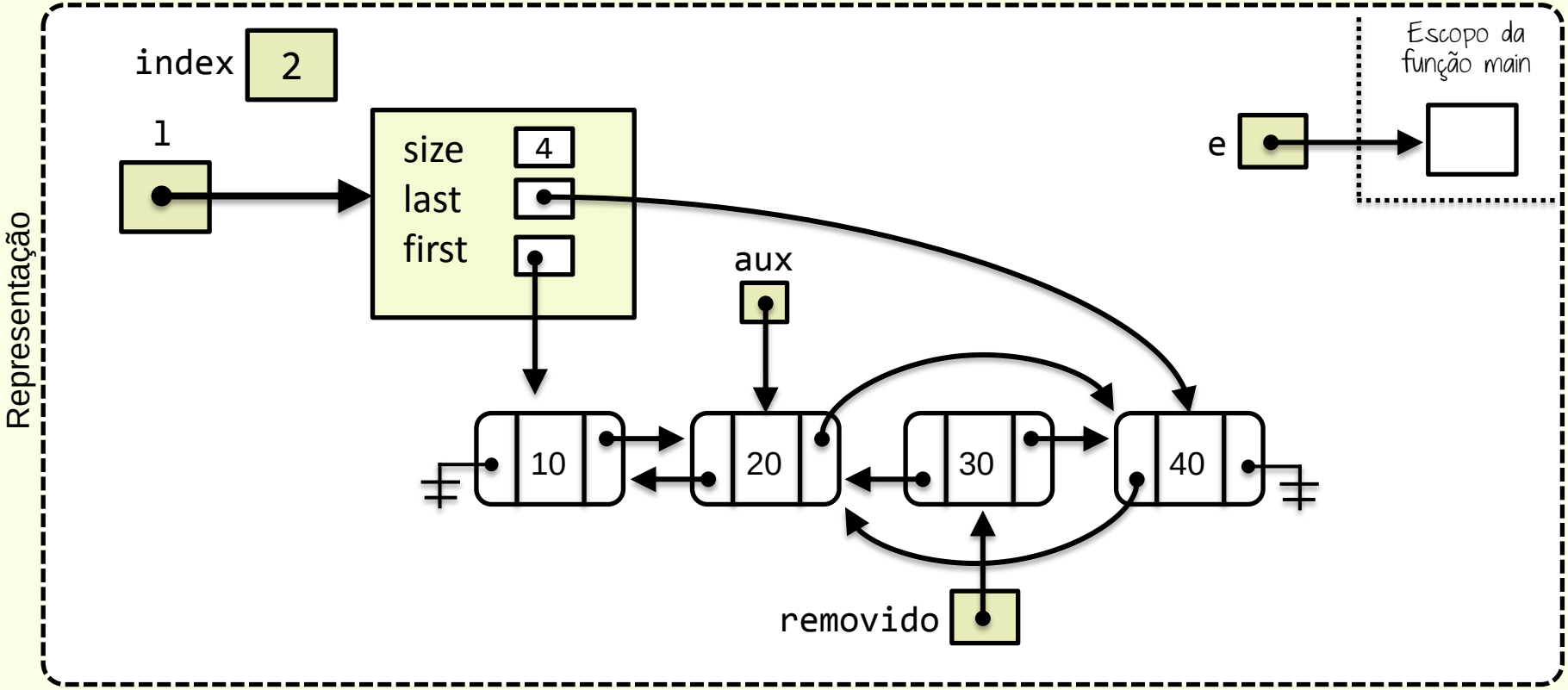
1 2 3 Remoção no meio da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

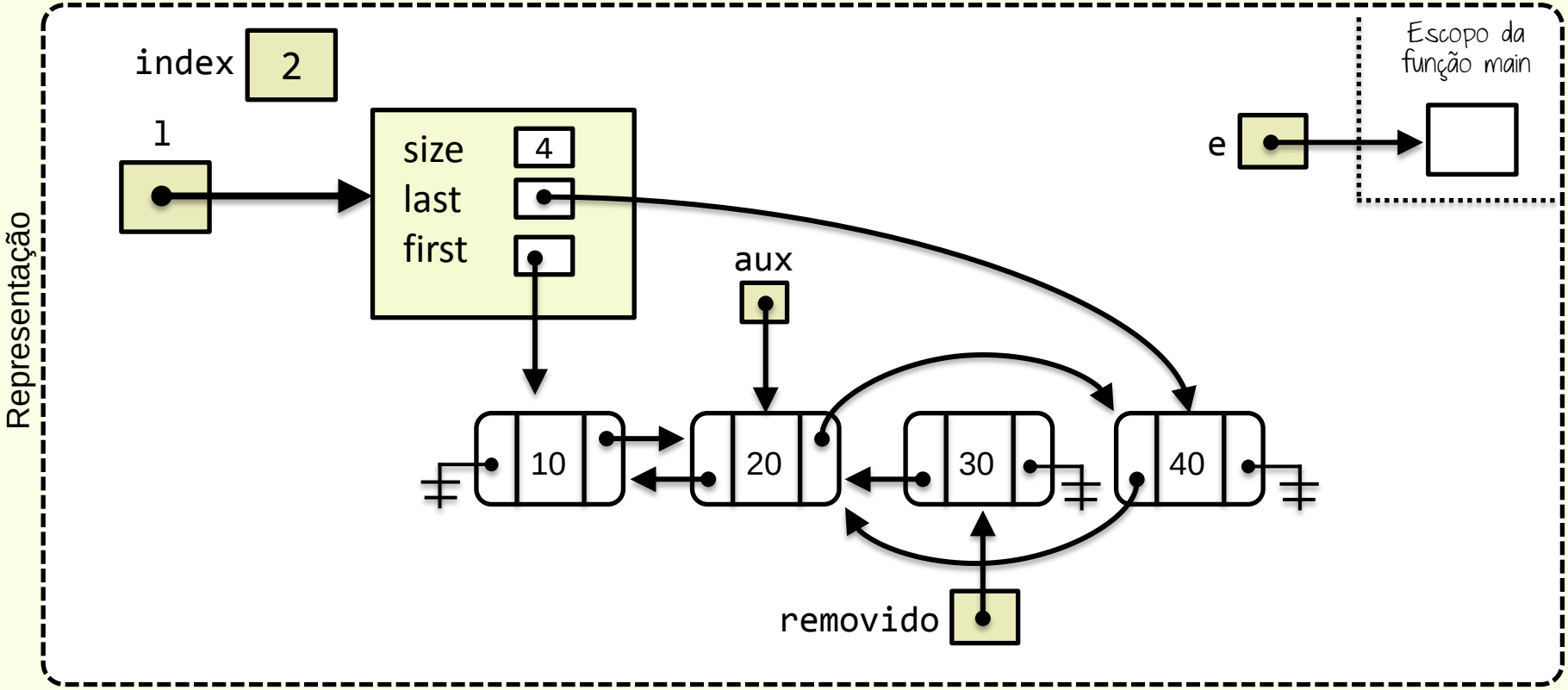
1 2 3 Remoção no meio da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

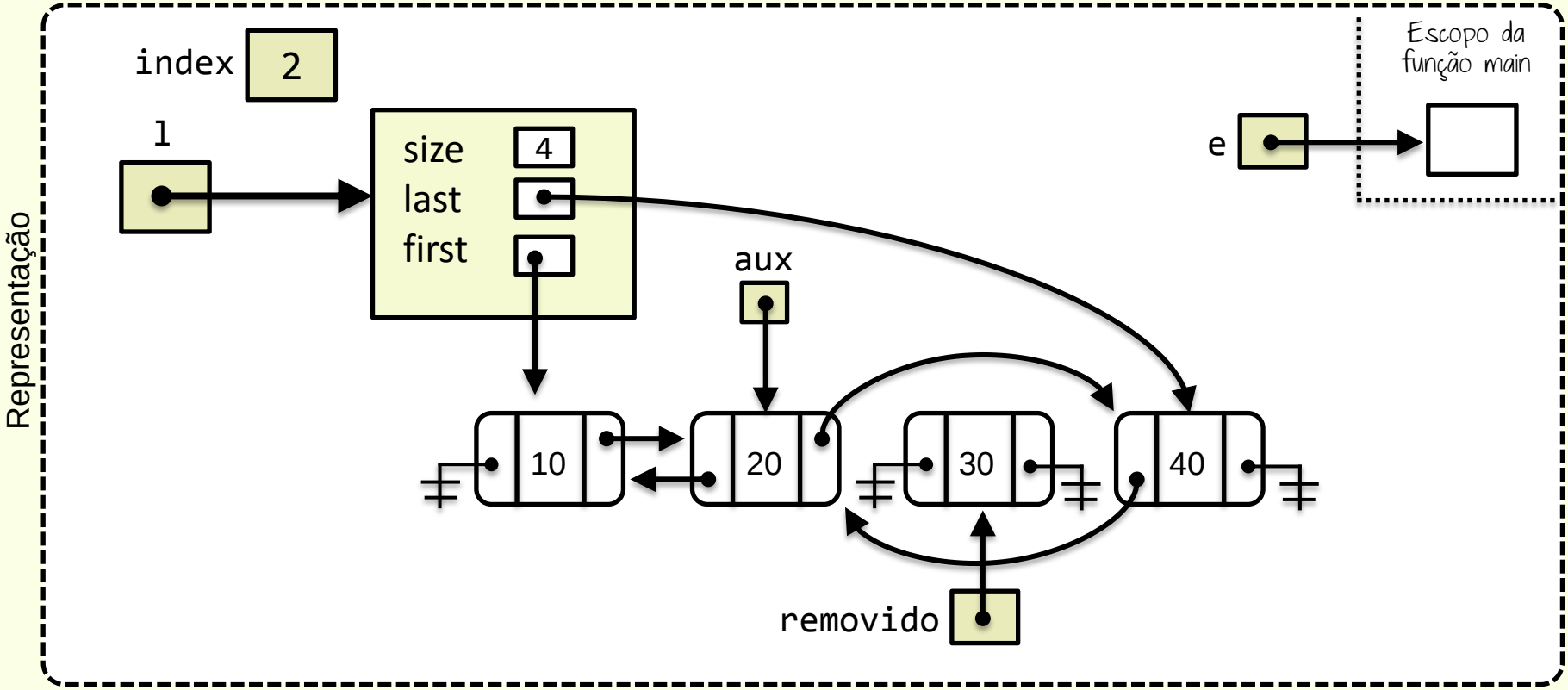
1 2 3 Remoção no meio da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

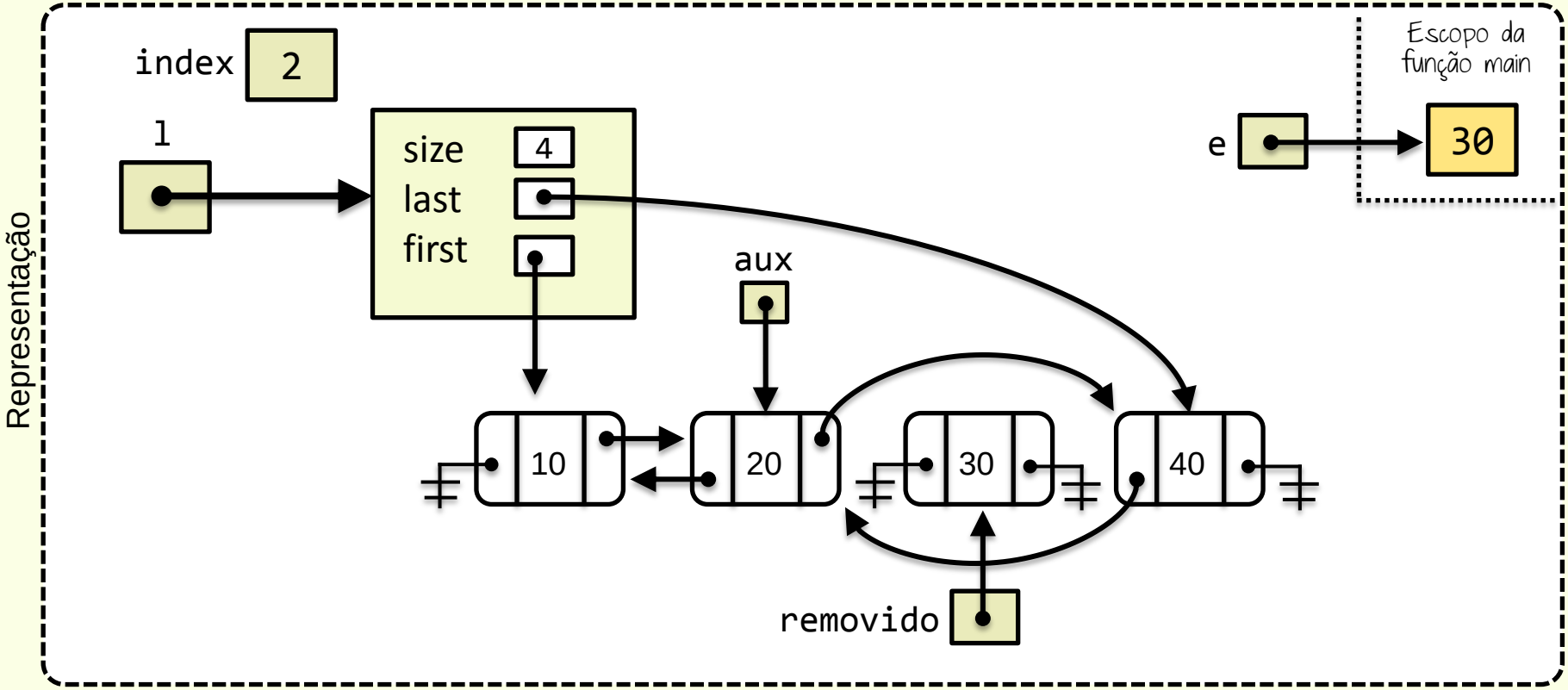
1 2 3 Remoção no meio da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

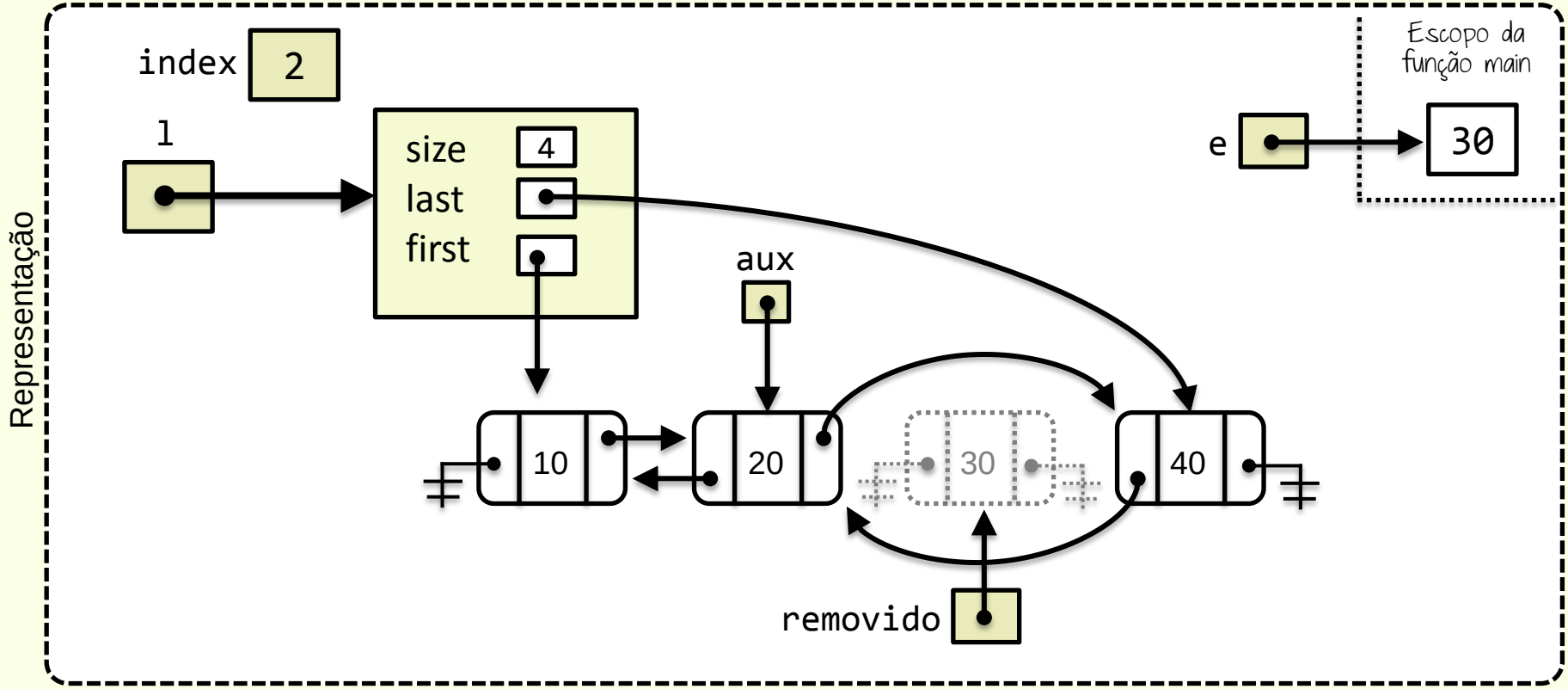
1 2 3 Remoção no meio da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

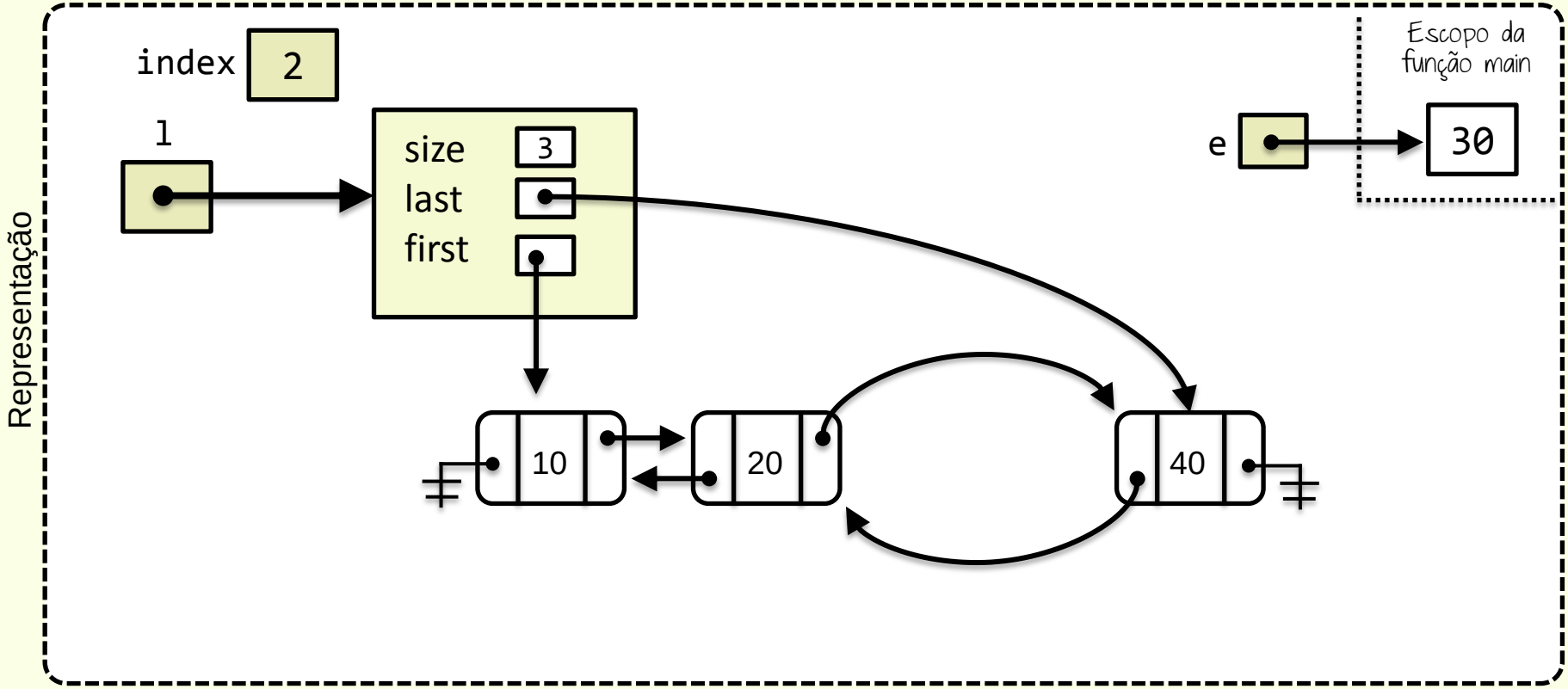
1 2 3 Remoção no meio da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

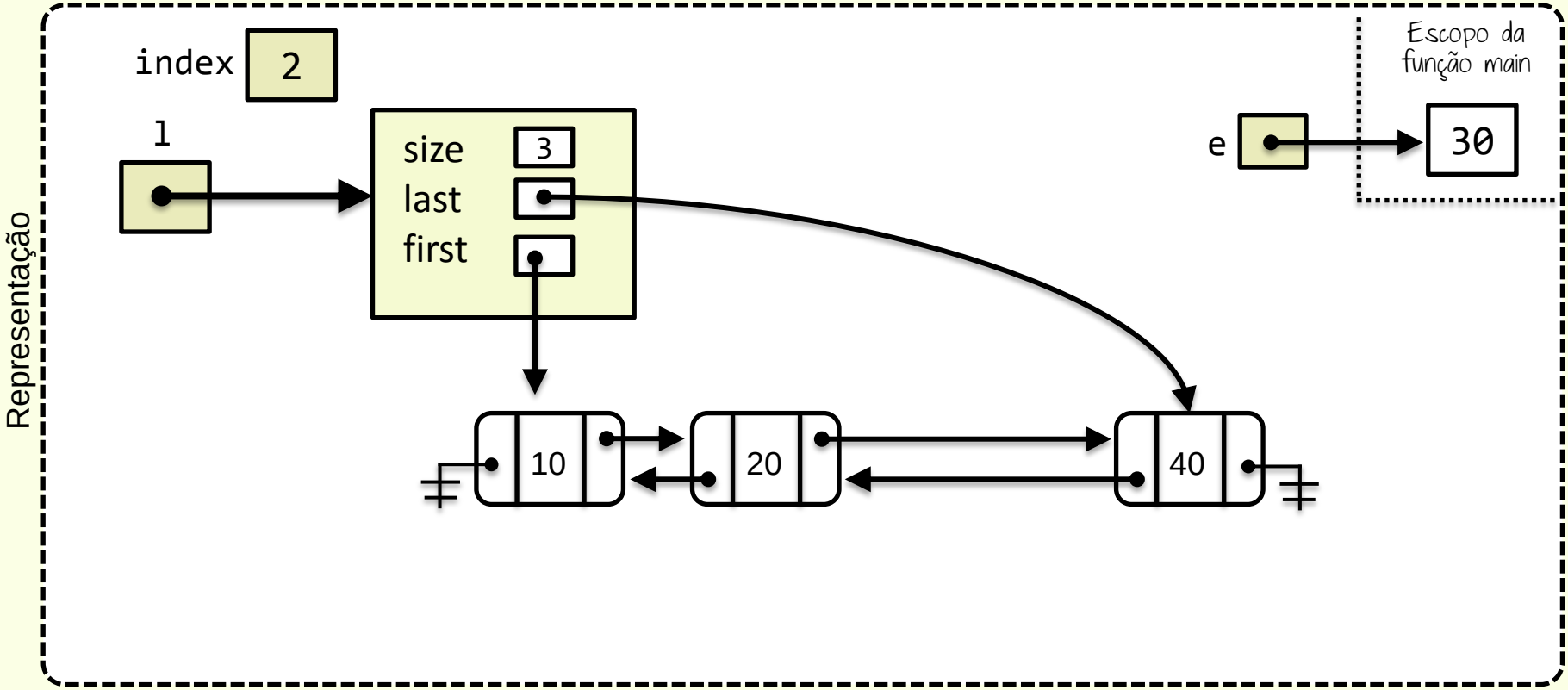
1 2 3 Remoção no meio da lista



removeList

```
int removeList(List* l, int index, ItemType *e);
```

1 2 3 Remoção no meio da lista



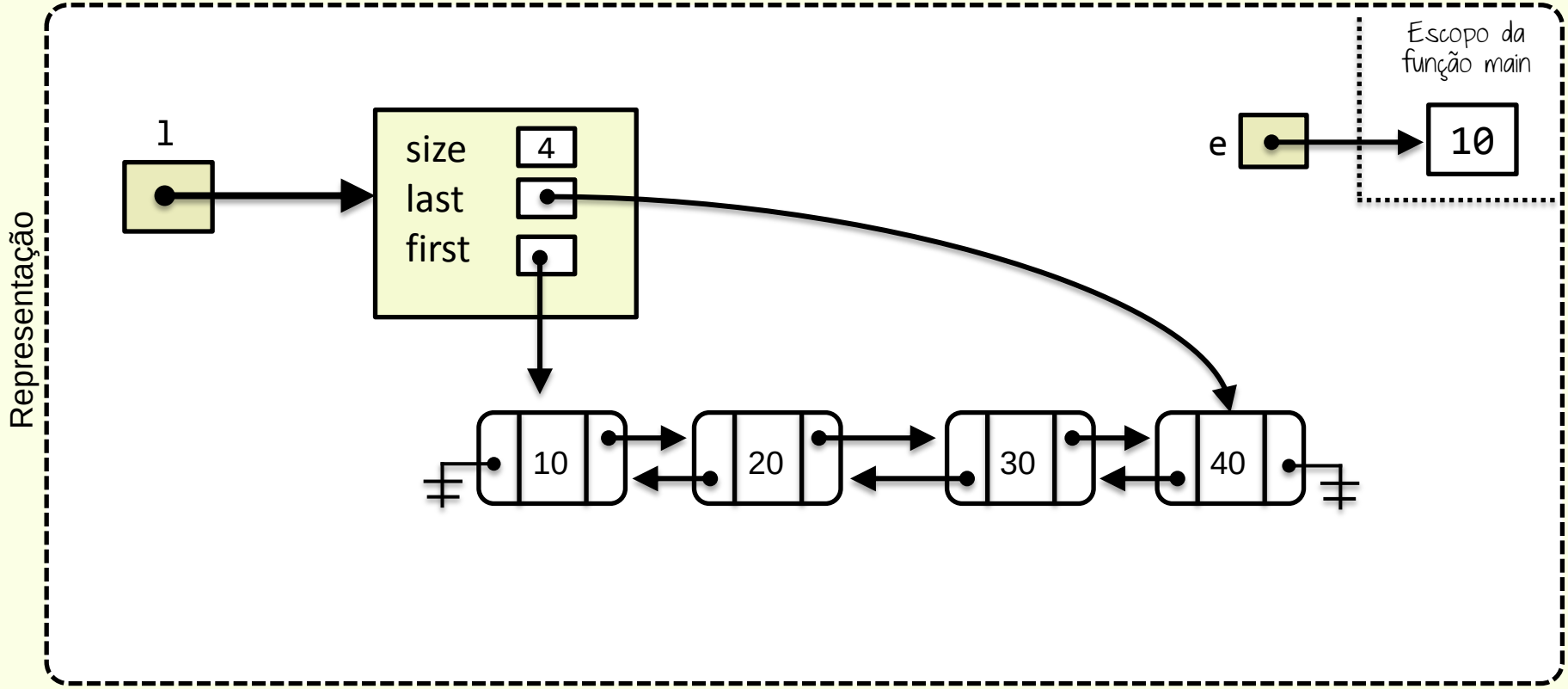
```
List *createList ();
void initializeList(List *l);
int addLastList(List *l, ItemType e);
int addList(List* l, ItemType e, int index);
int removeList(List* l, int index, ItemType *e);
int removeElementList(List* l, ItemType* e);
```


removeElementList

removeElementList

```
int removeElementList(List* l, ItemType* e);
```

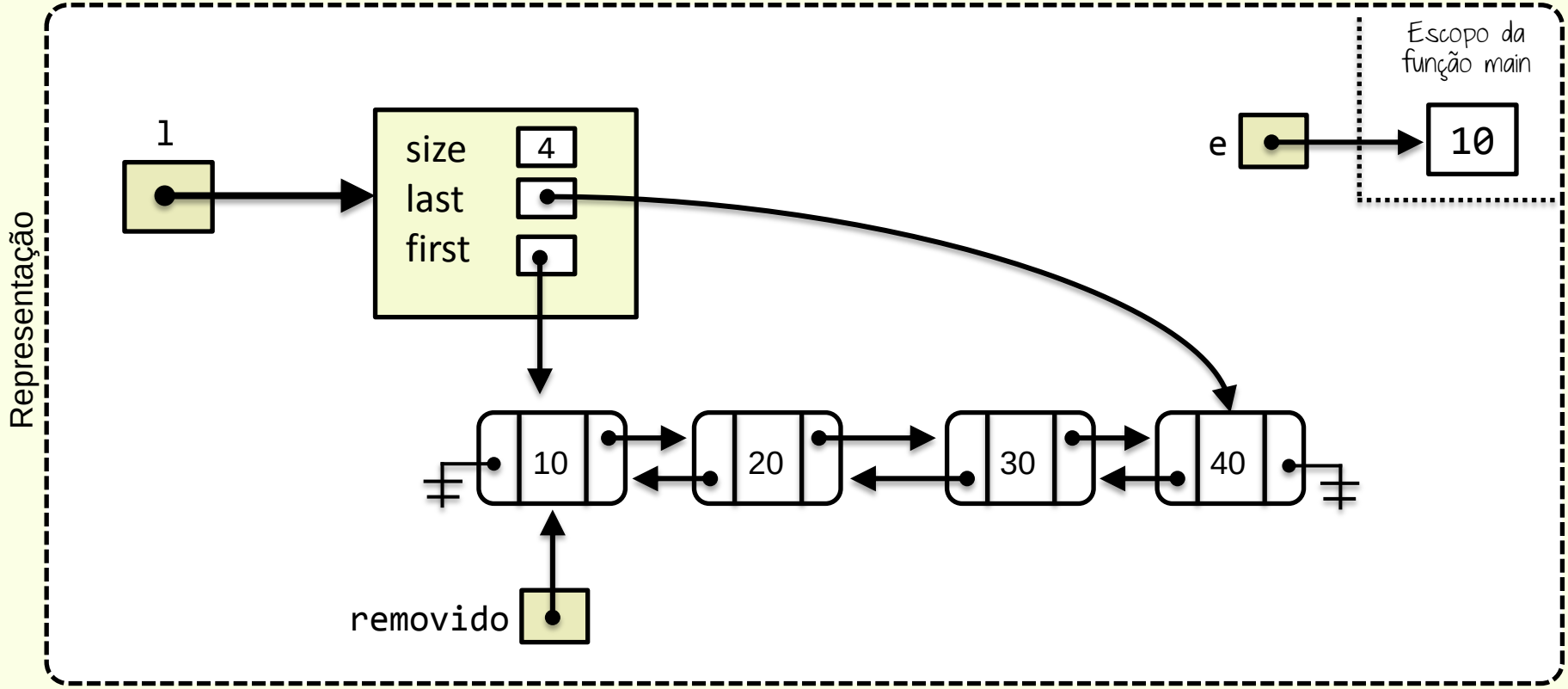
1 2 3 Remoção do elemento que é o primeiro da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

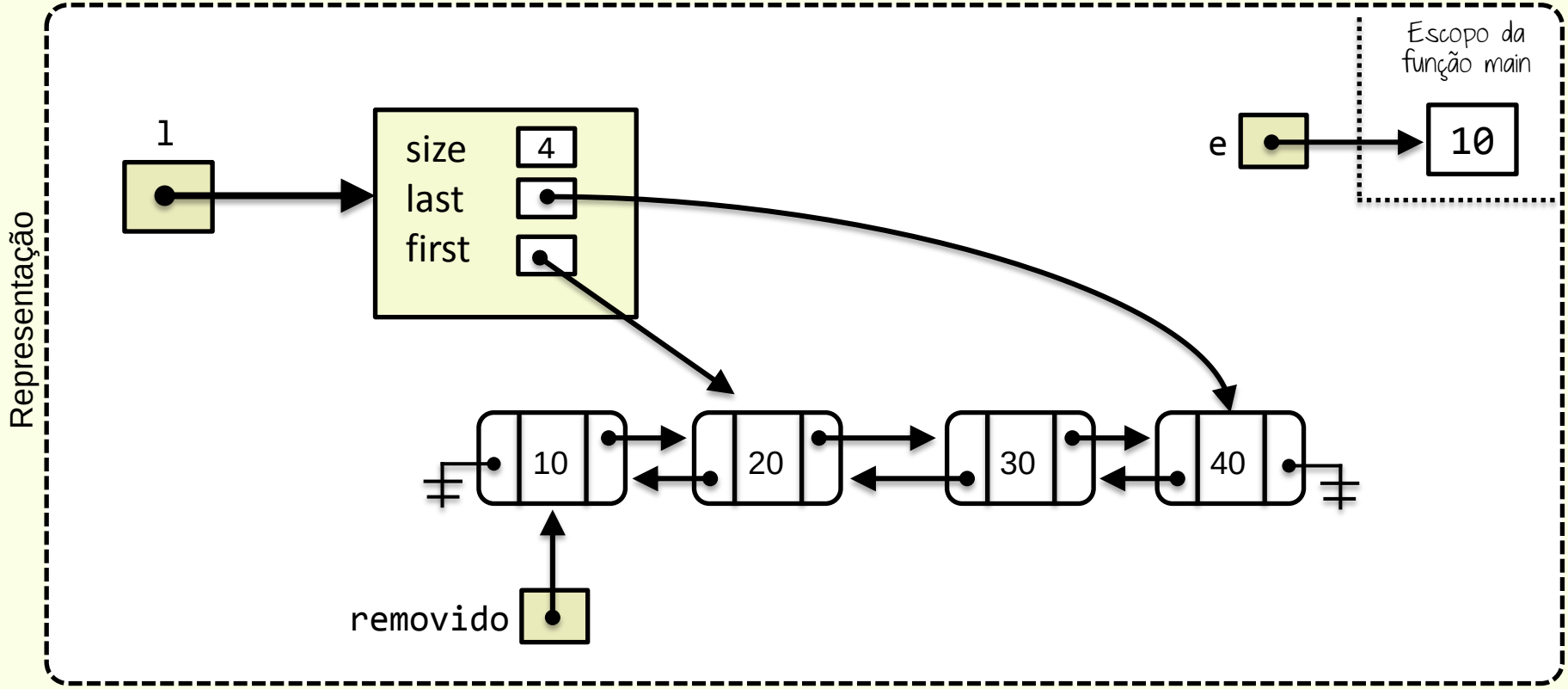
1 2 3 Remoção do elemento que é o primeiro da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

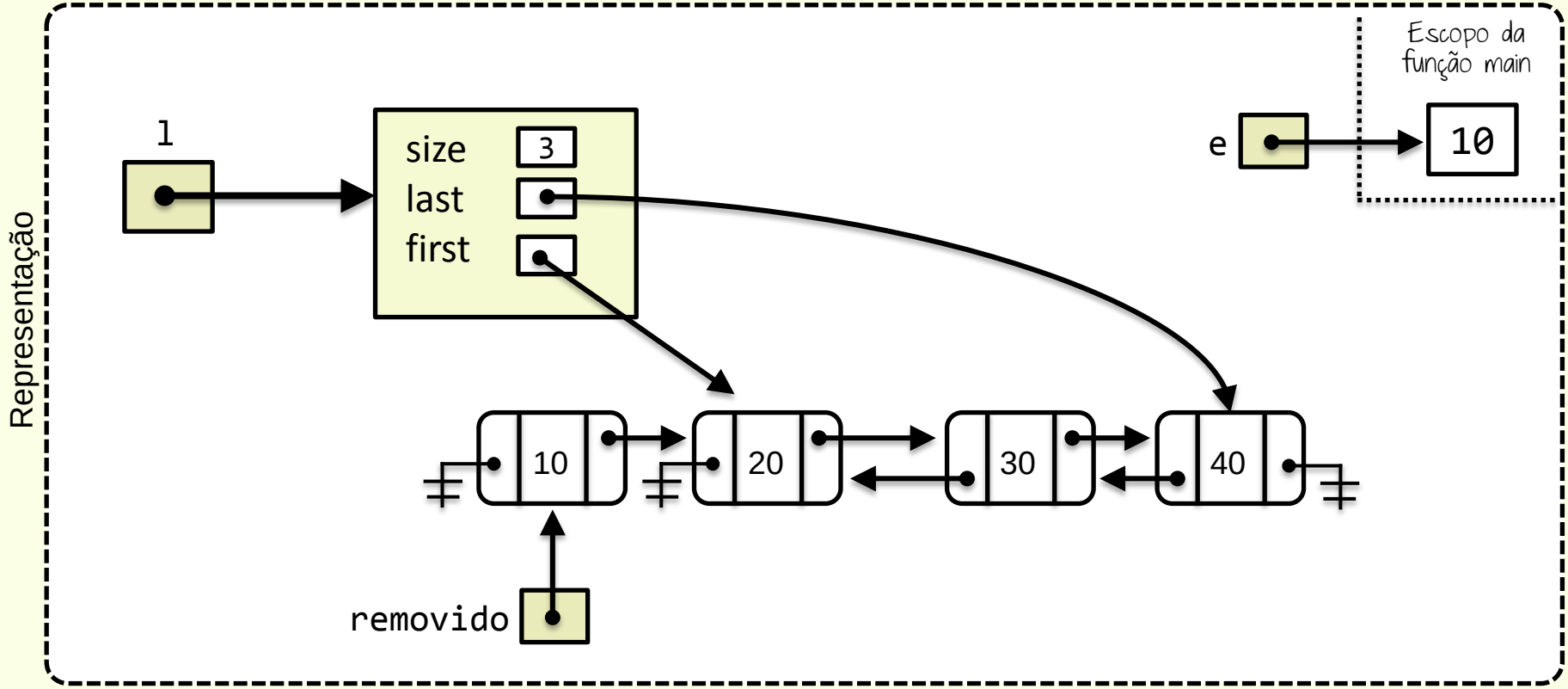
1 2 3 Remoção do elemento que é o primeiro da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

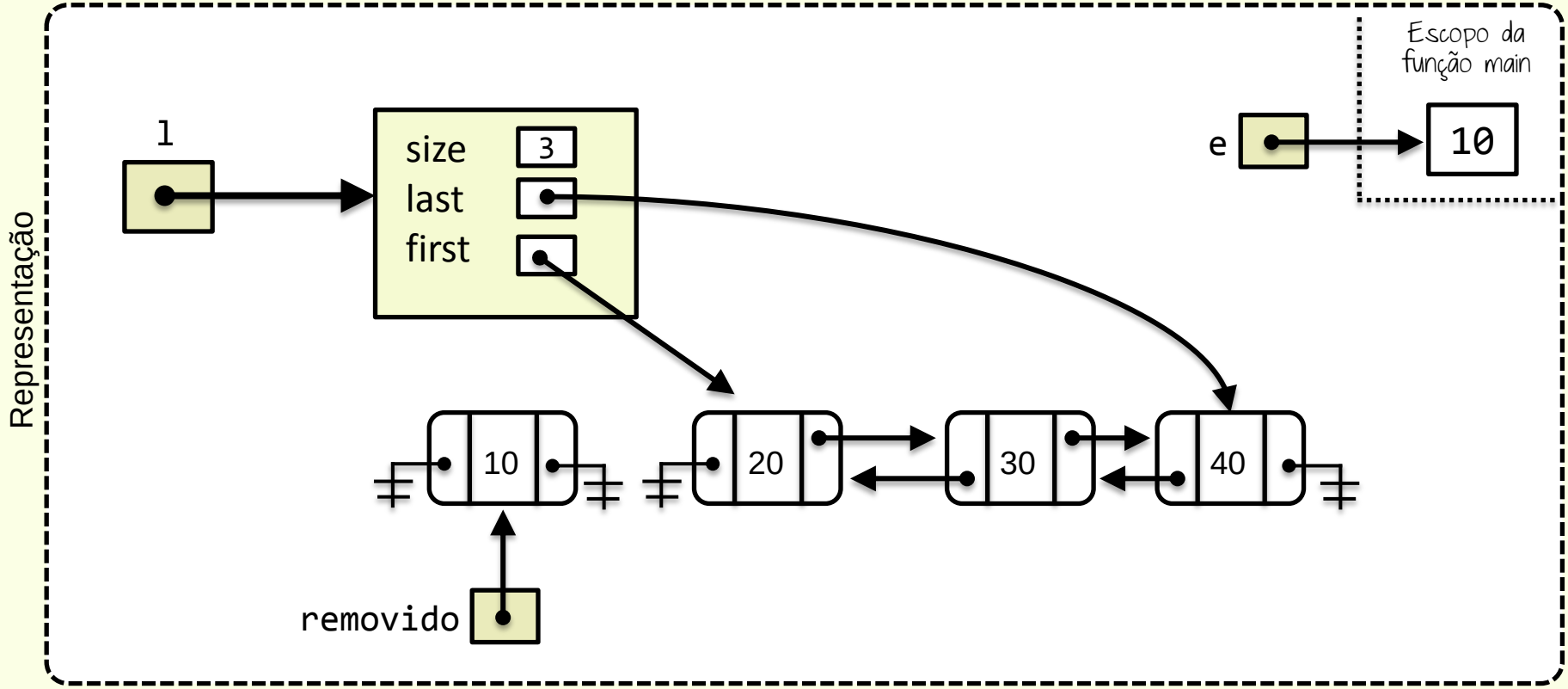
1 2 3 Remoção do elemento que é o primeiro da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

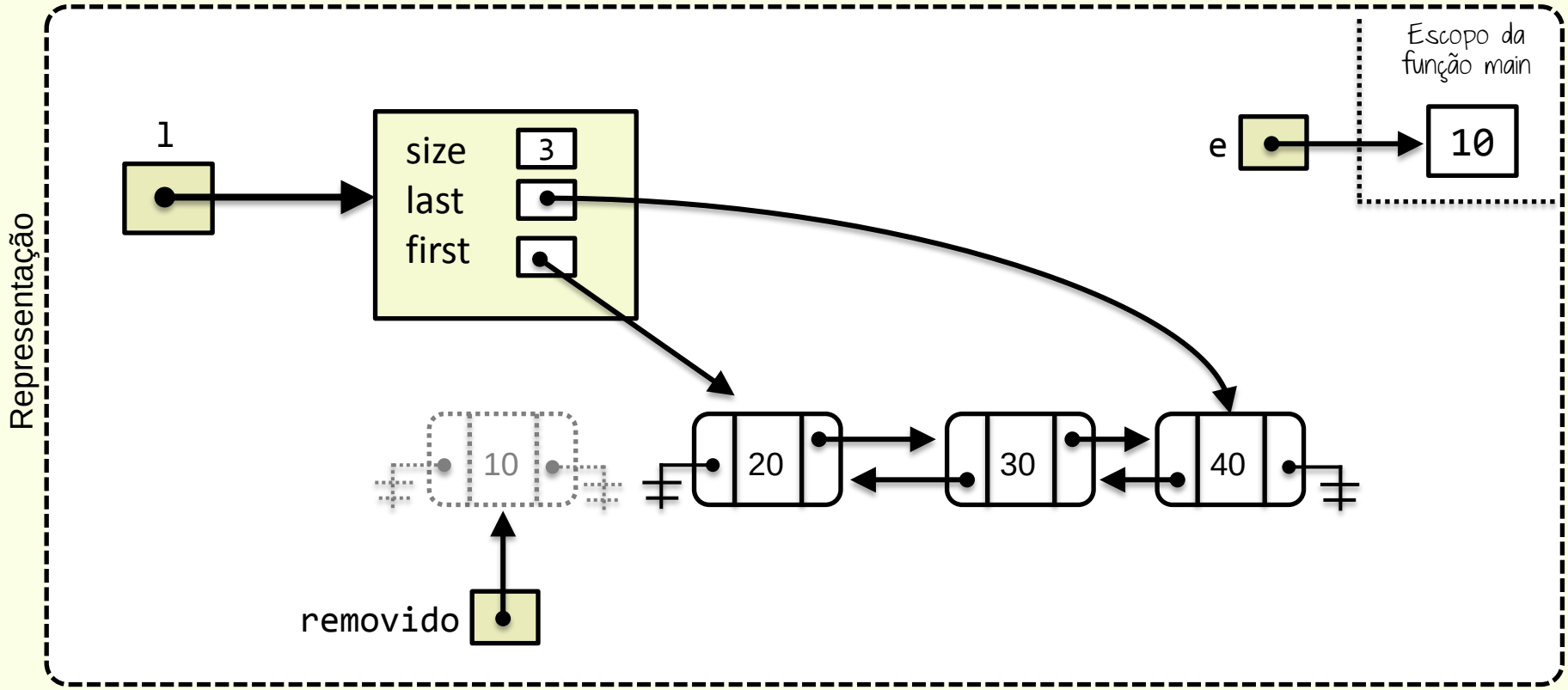
1 2 3 Remoção do elemento que é o primeiro da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

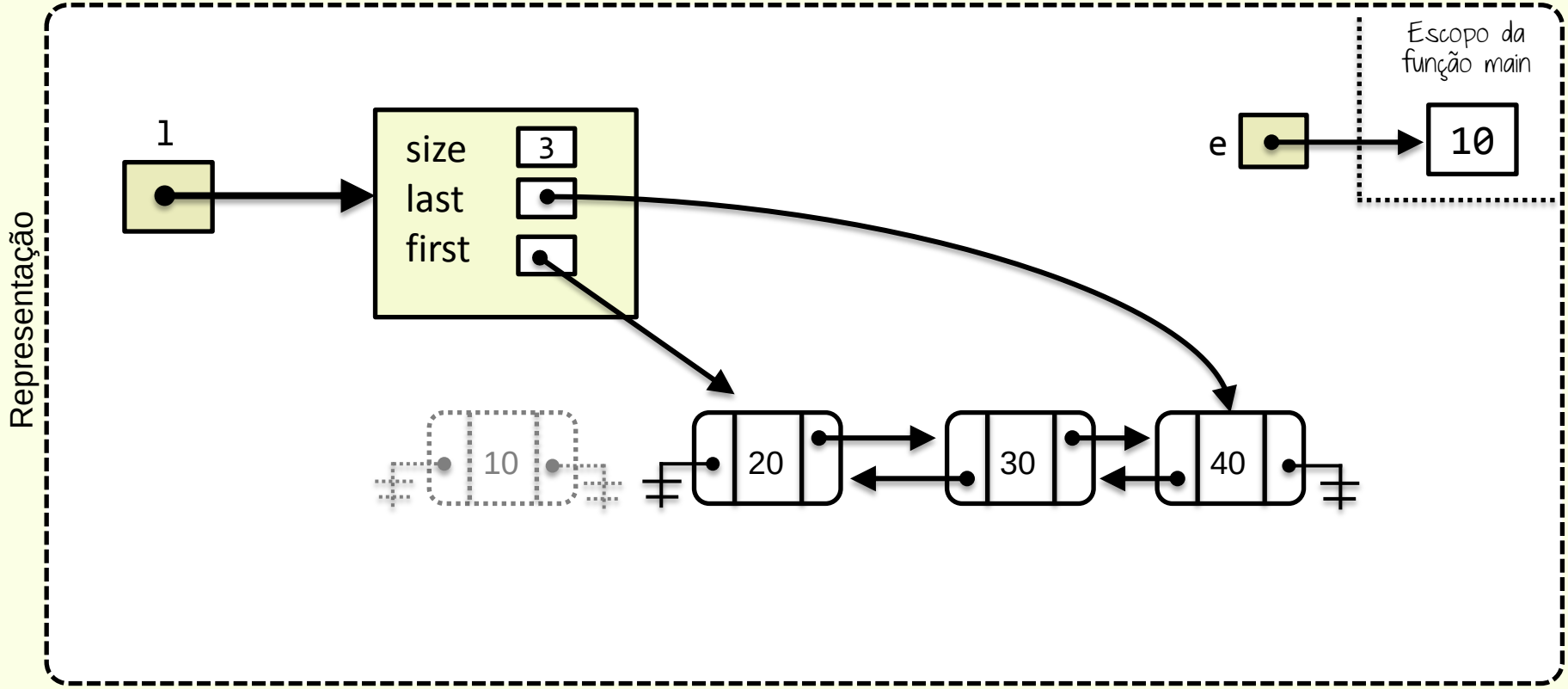
1 2 3 Remoção do elemento que é o primeiro da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

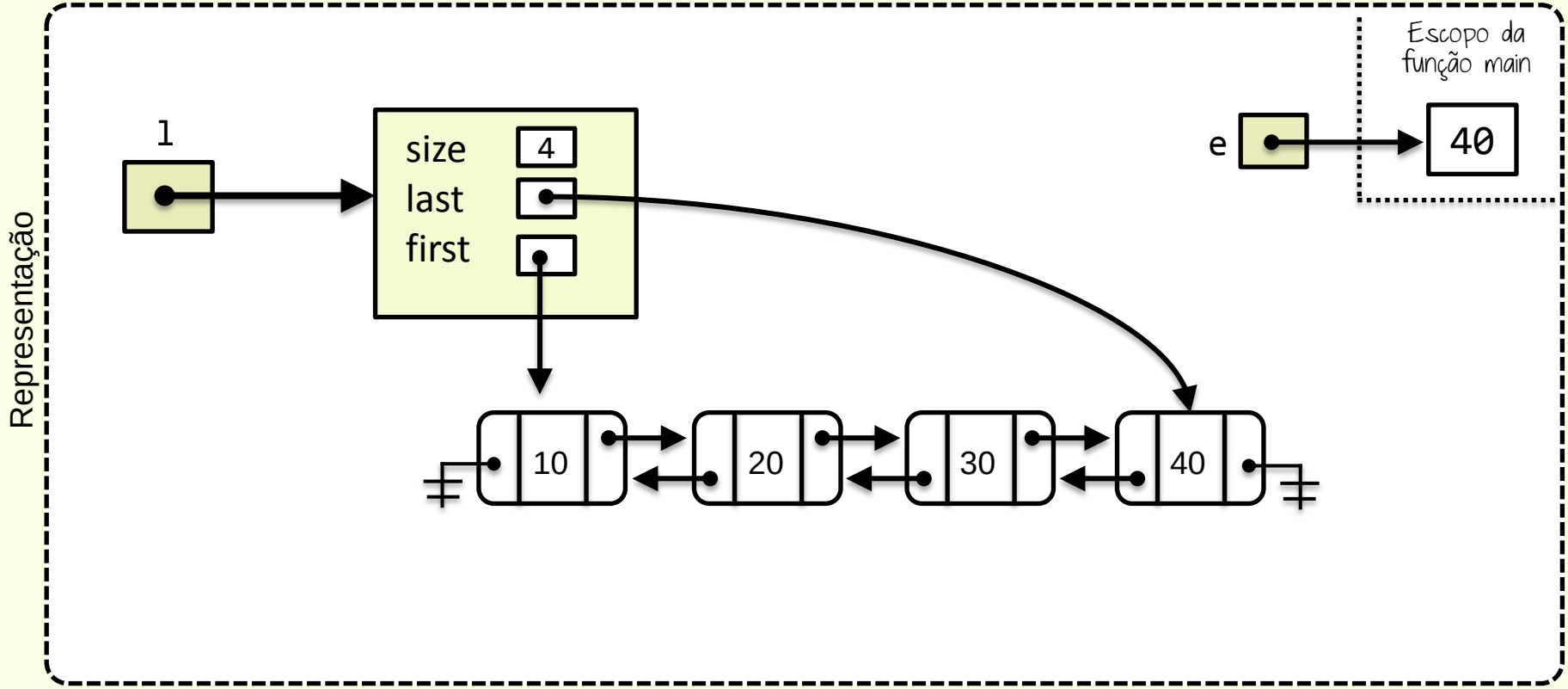
1 2 3 Remoção do elemento que é o primeiro da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

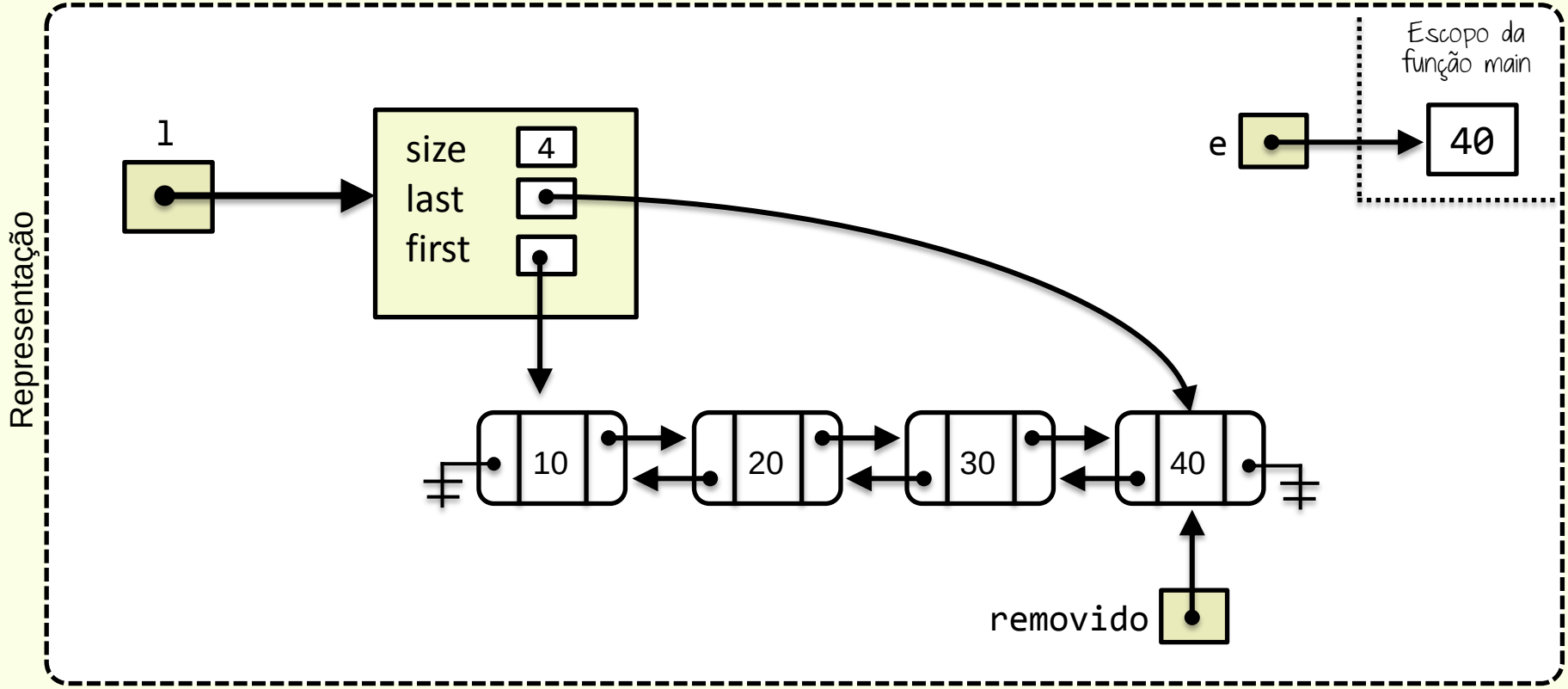
1 2 3 Remoção do elemento que é o último da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

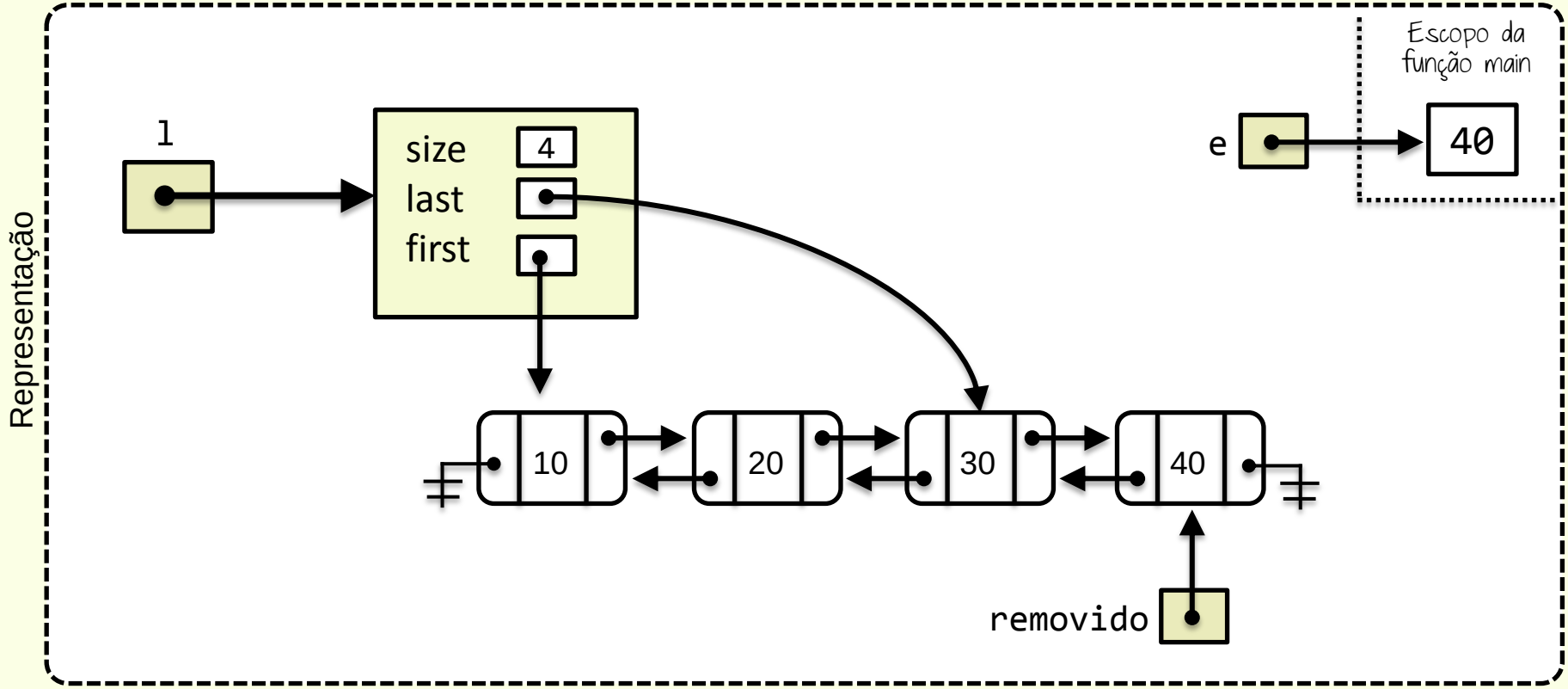
1 2 3 Remoção do elemento que é o último da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

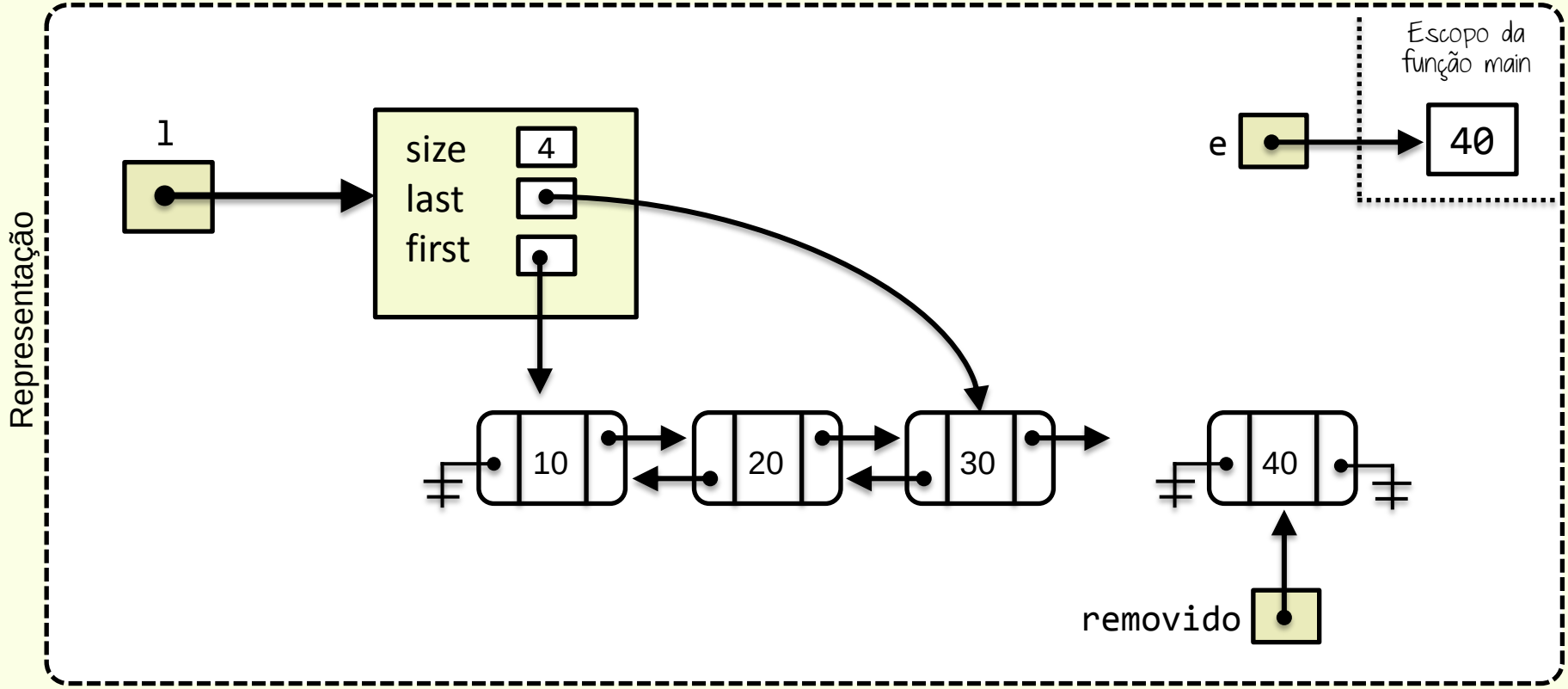
1 2 3 Remoção do elemento que é o último da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

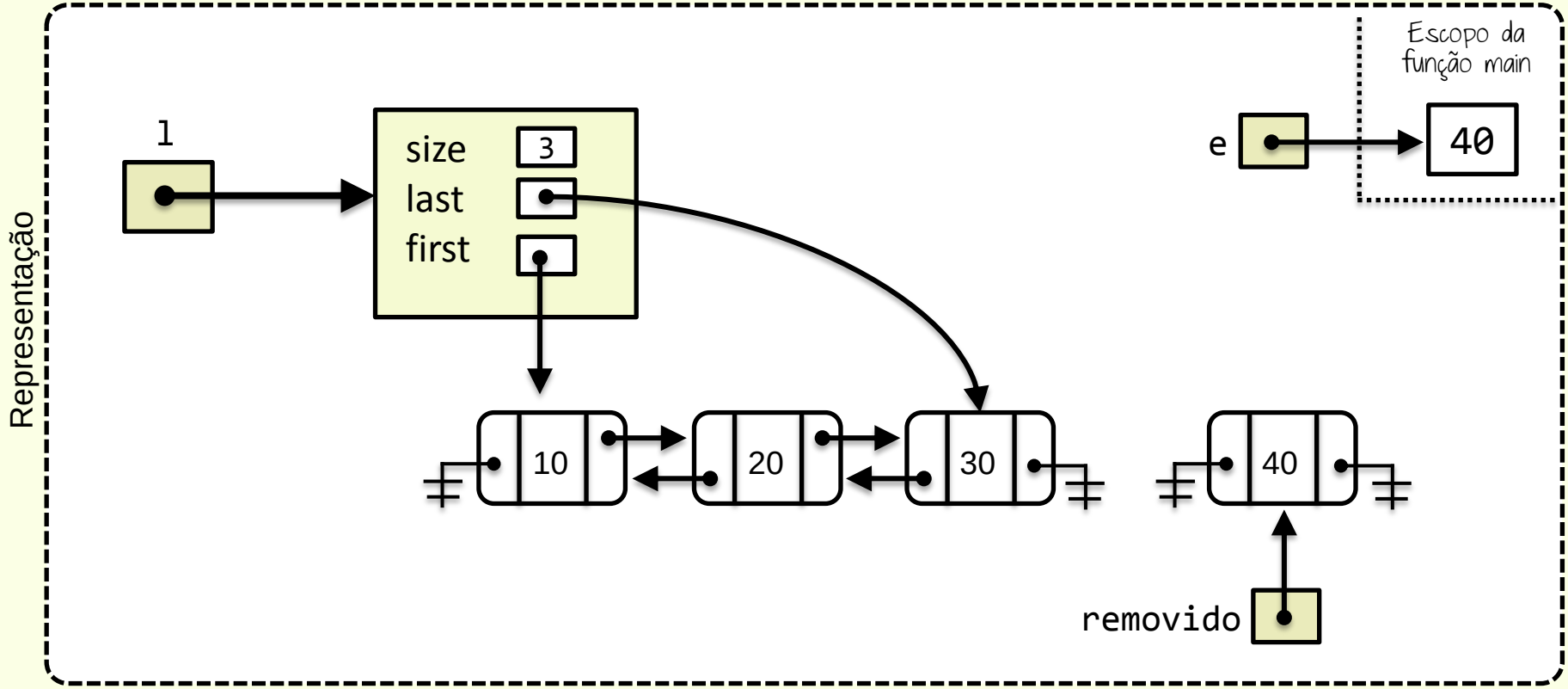
1 2 3 Remoção do elemento que é o último da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

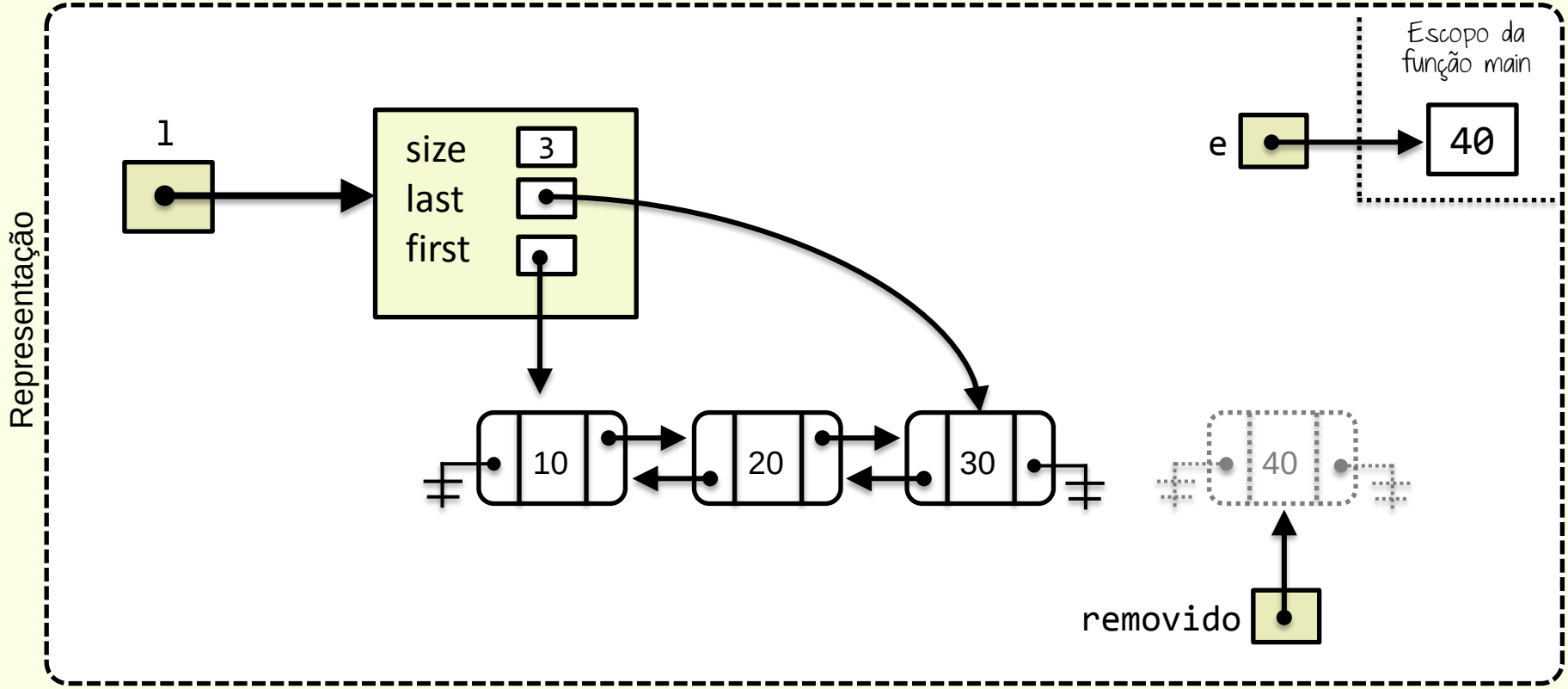
1 2 3 Remoção do elemento que é o último da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

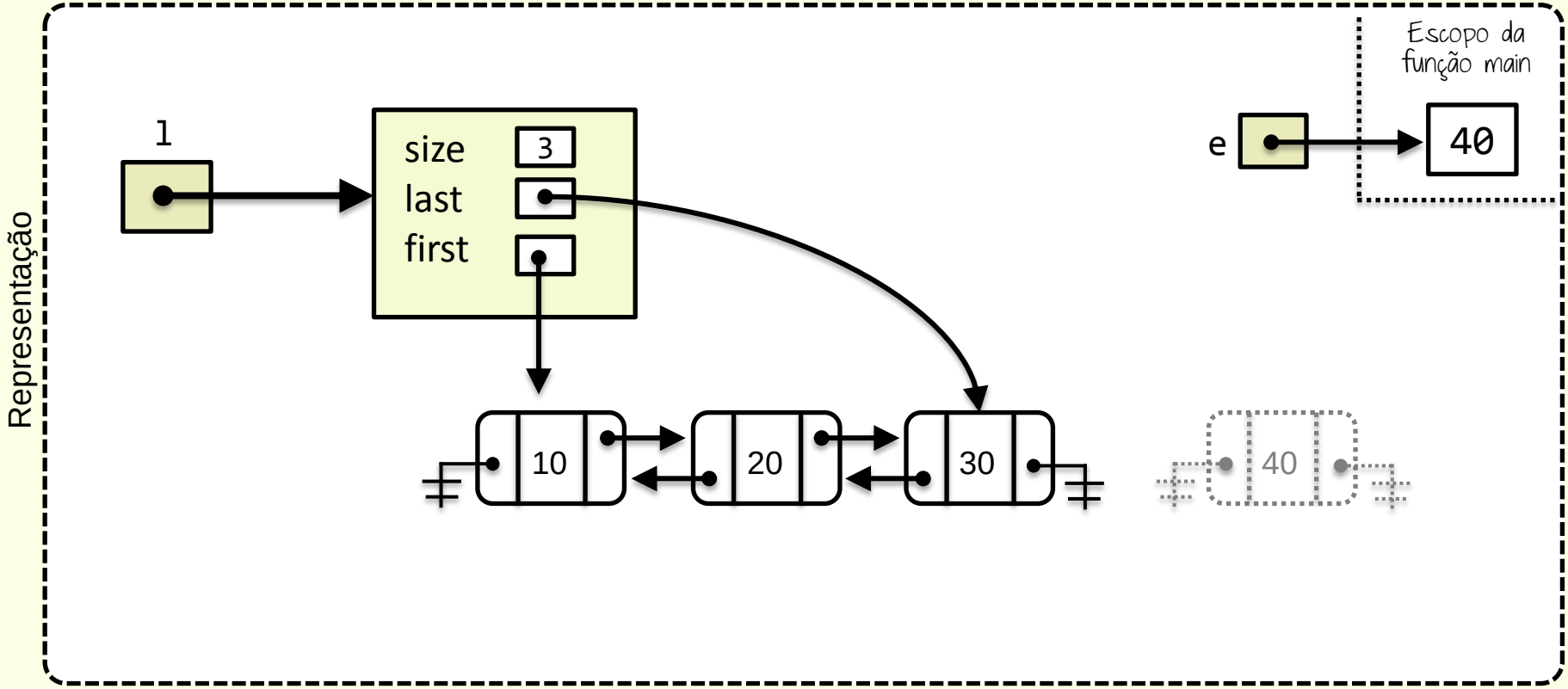
1 2 3 Remoção do elemento que é o último da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

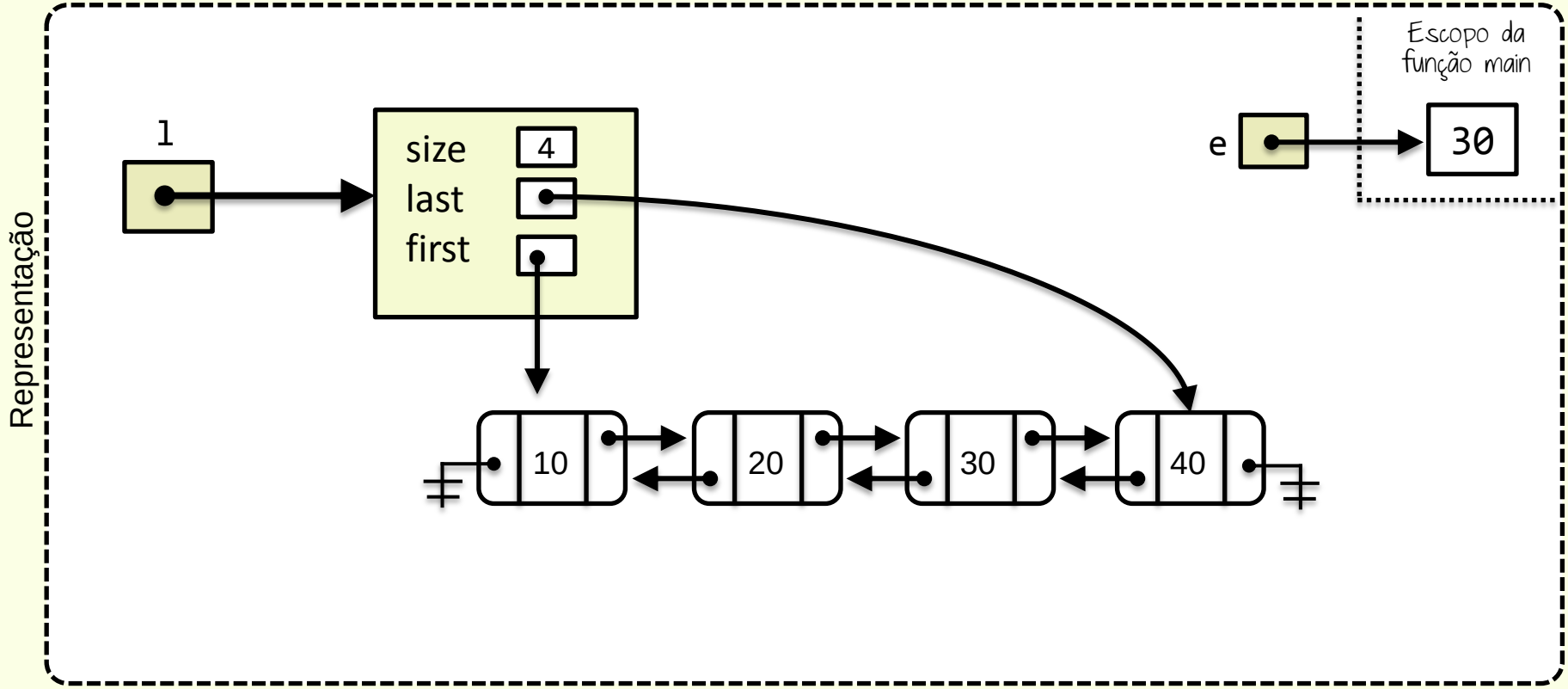
1 2 3 Remoção do elemento que é o último da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

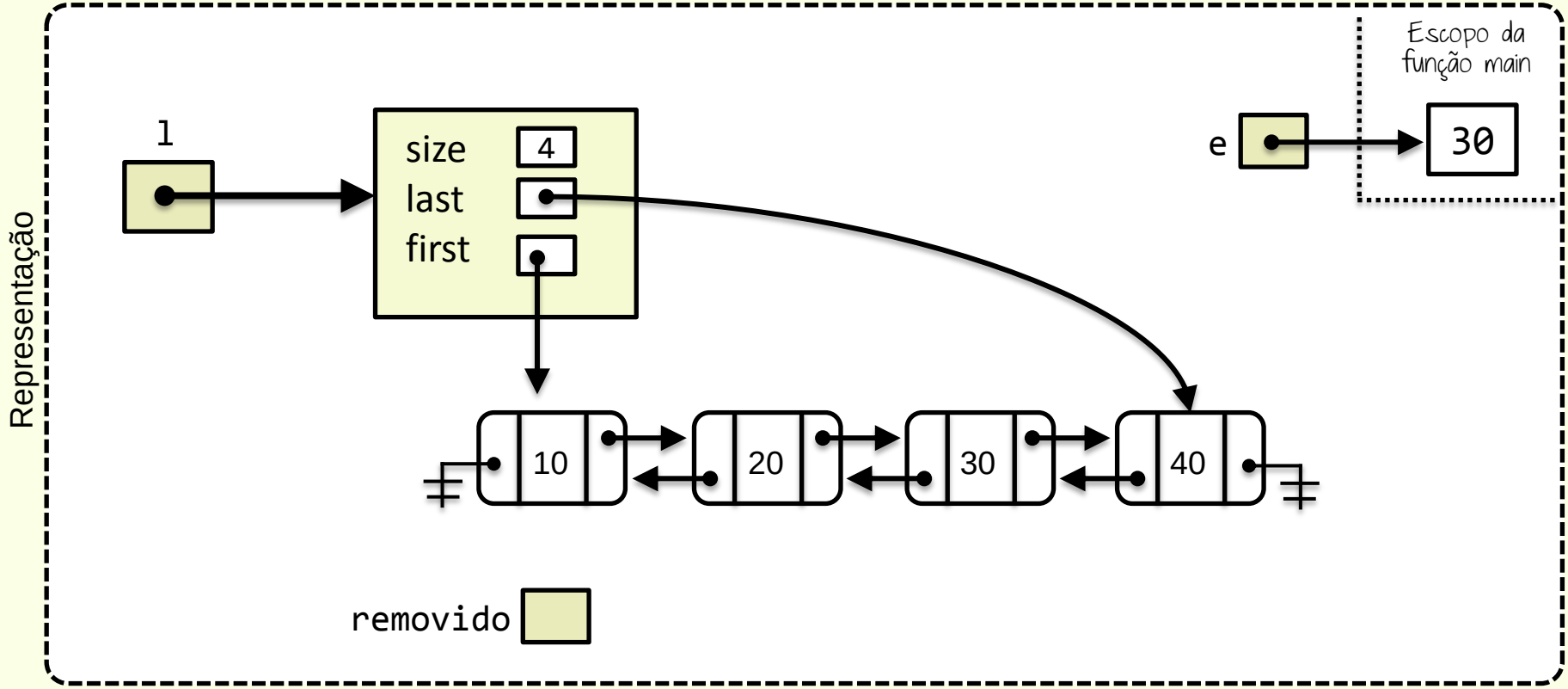
1 2 3 Remoção do elemento que é está no meio da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

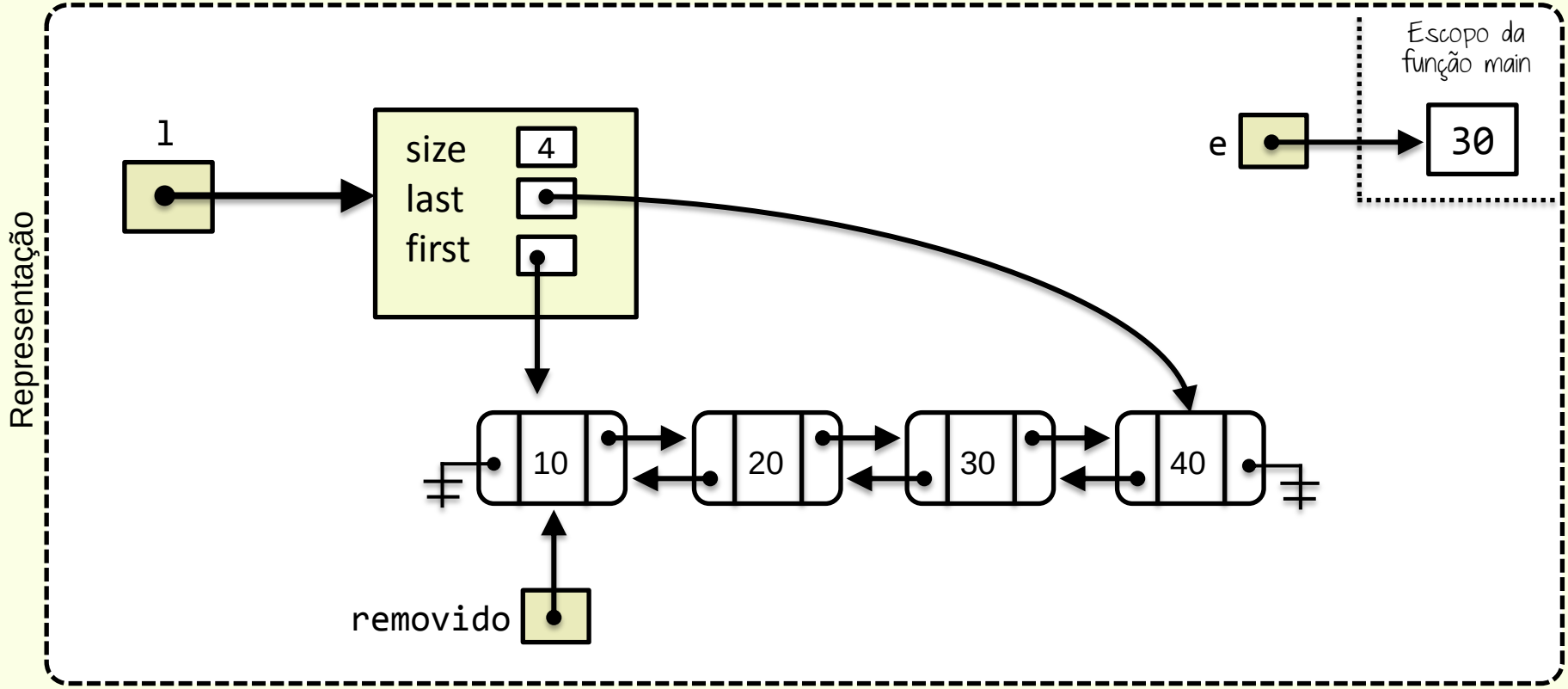
1 2 3 Remoção do elemento que é está no meio da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

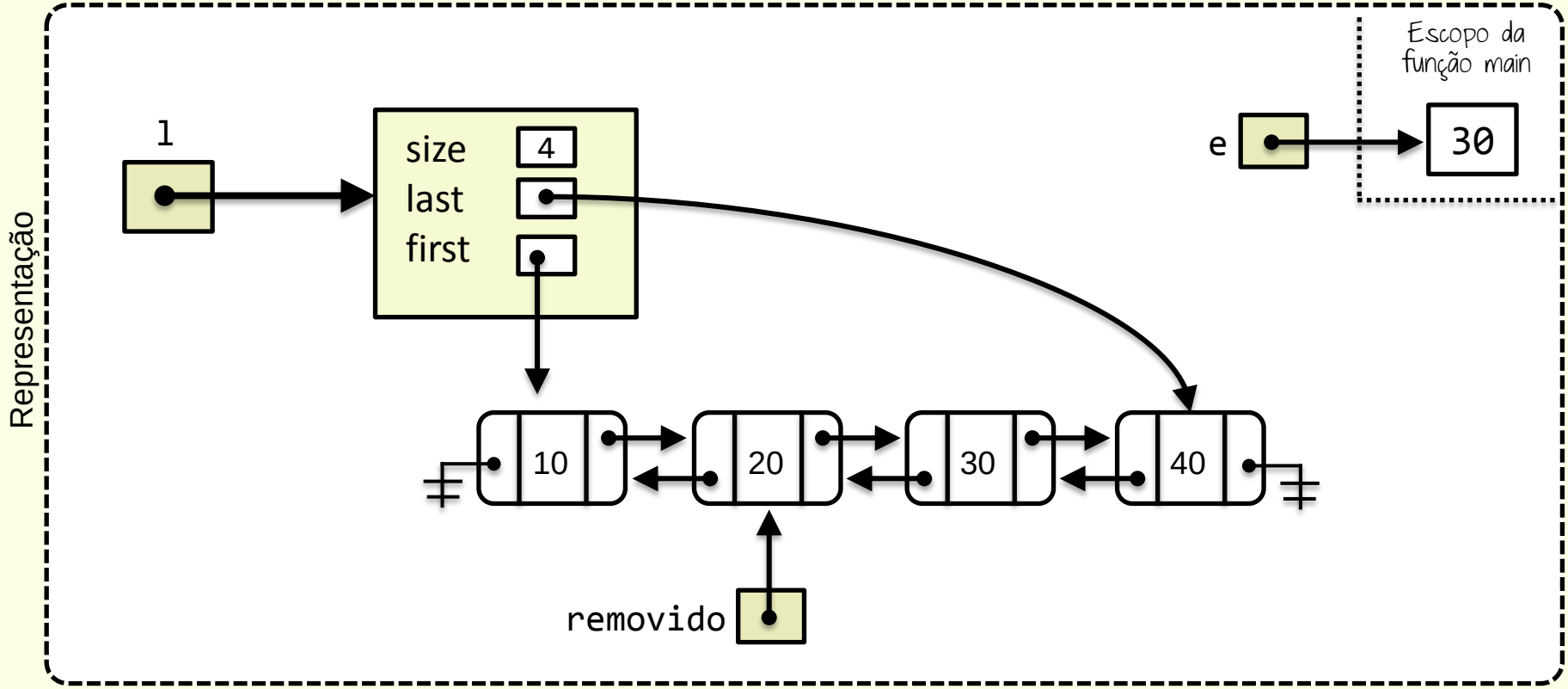
1 2 3 Remoção do elemento que é está no meio da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

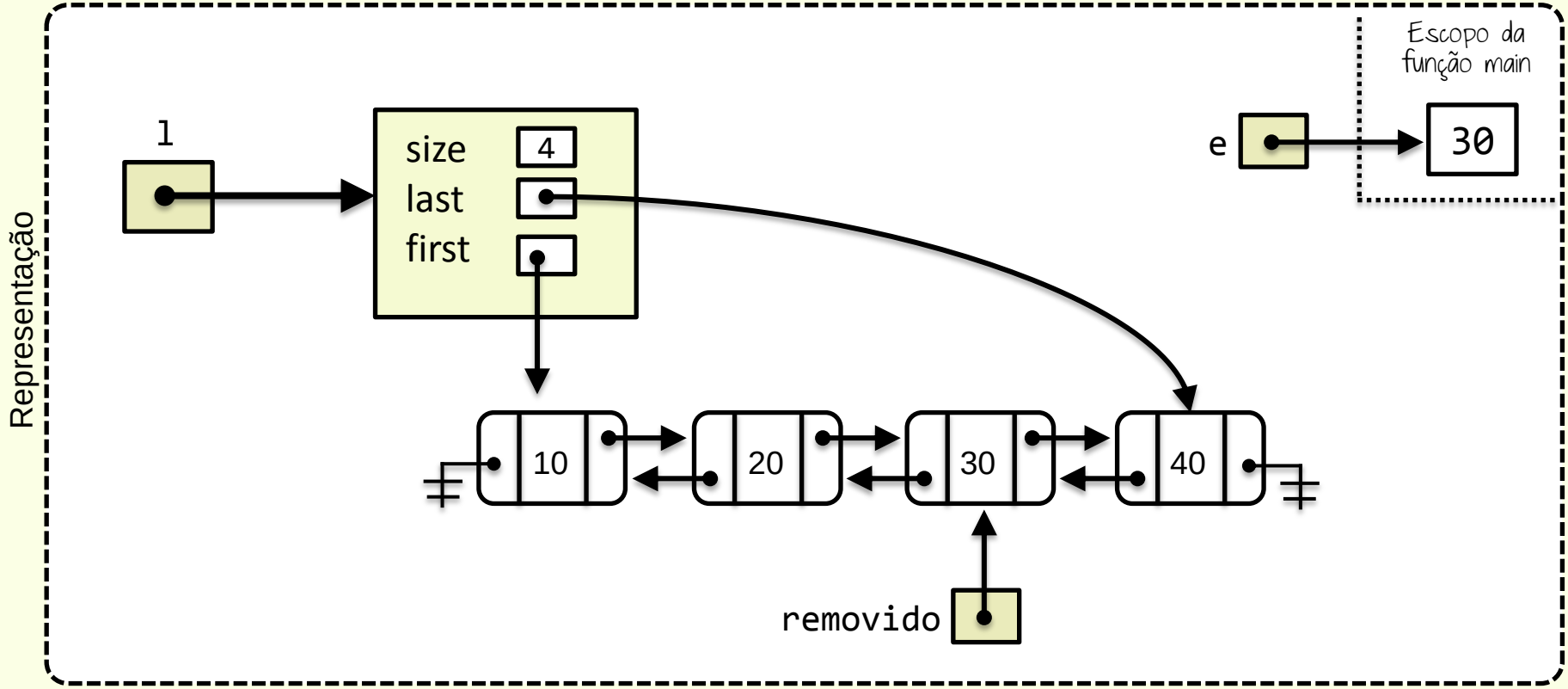
1 2 3 Remoção do elemento que é está no meio da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

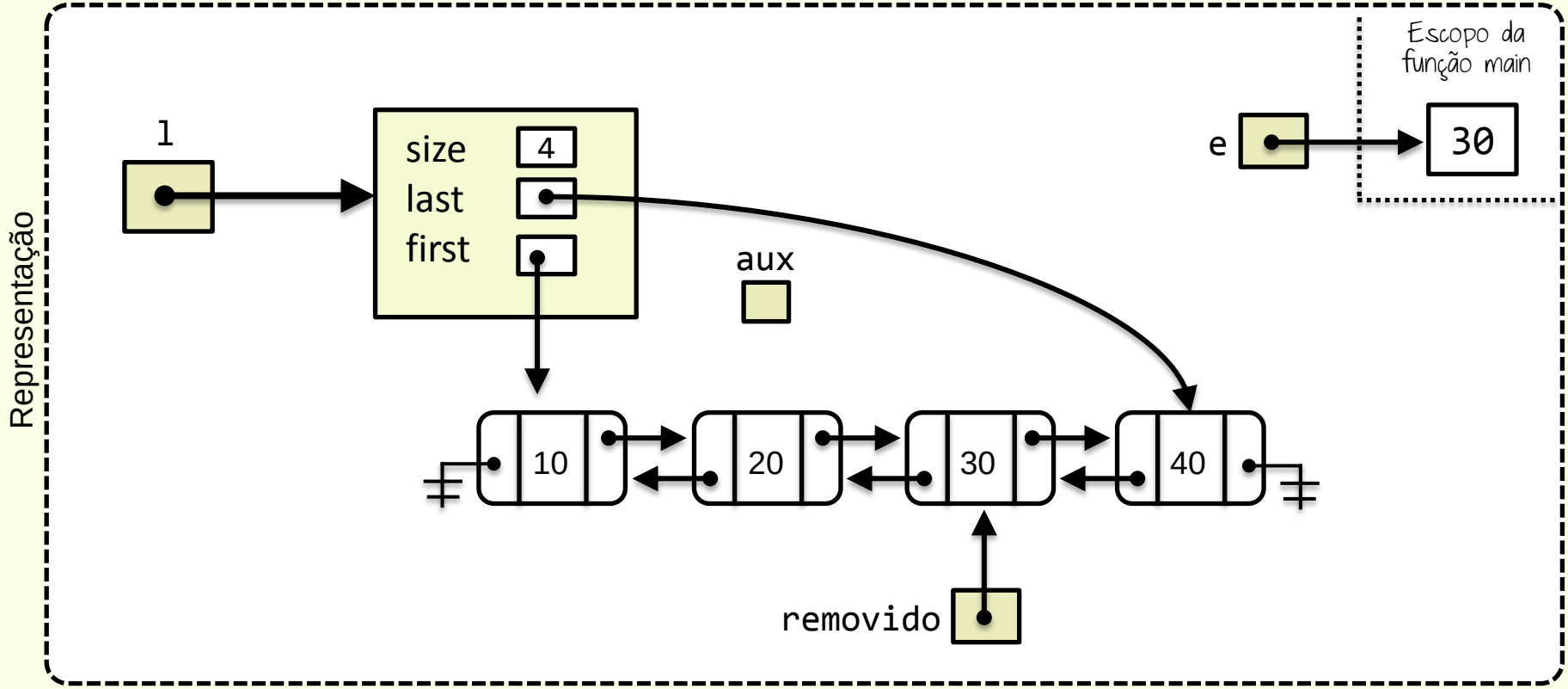
1 2 3 Remoção do elemento que é está no meio da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

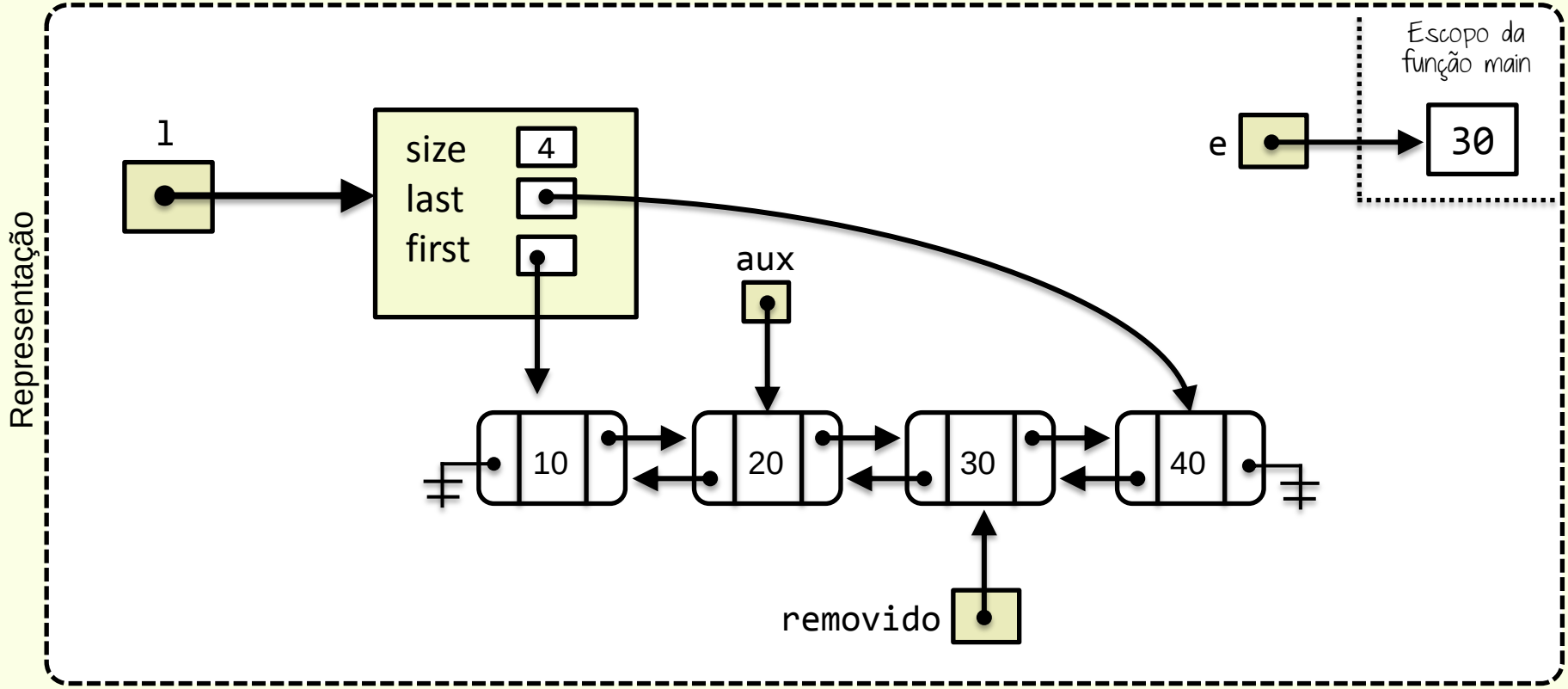
1 2 3 Remoção do elemento que é está no meio da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

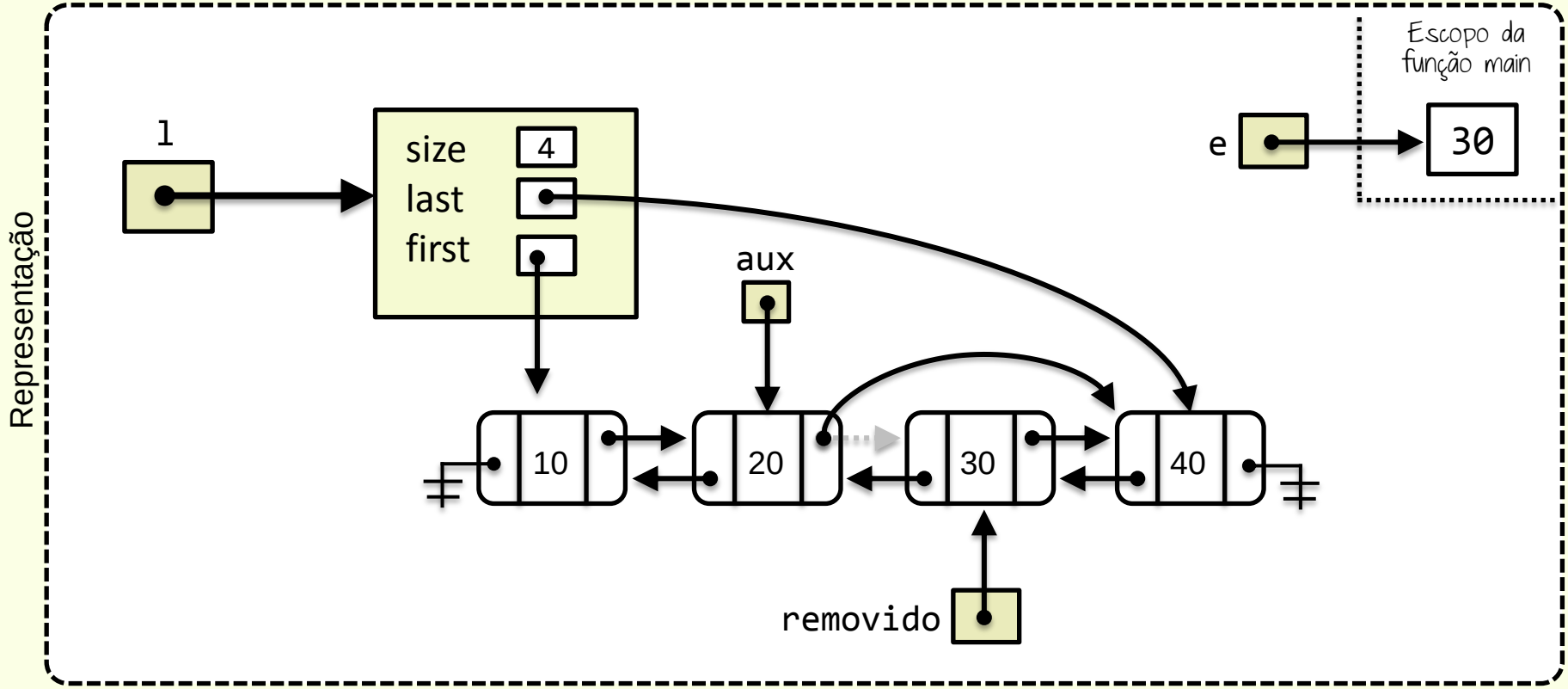
1 2 3 Remoção do elemento que é está no meio da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

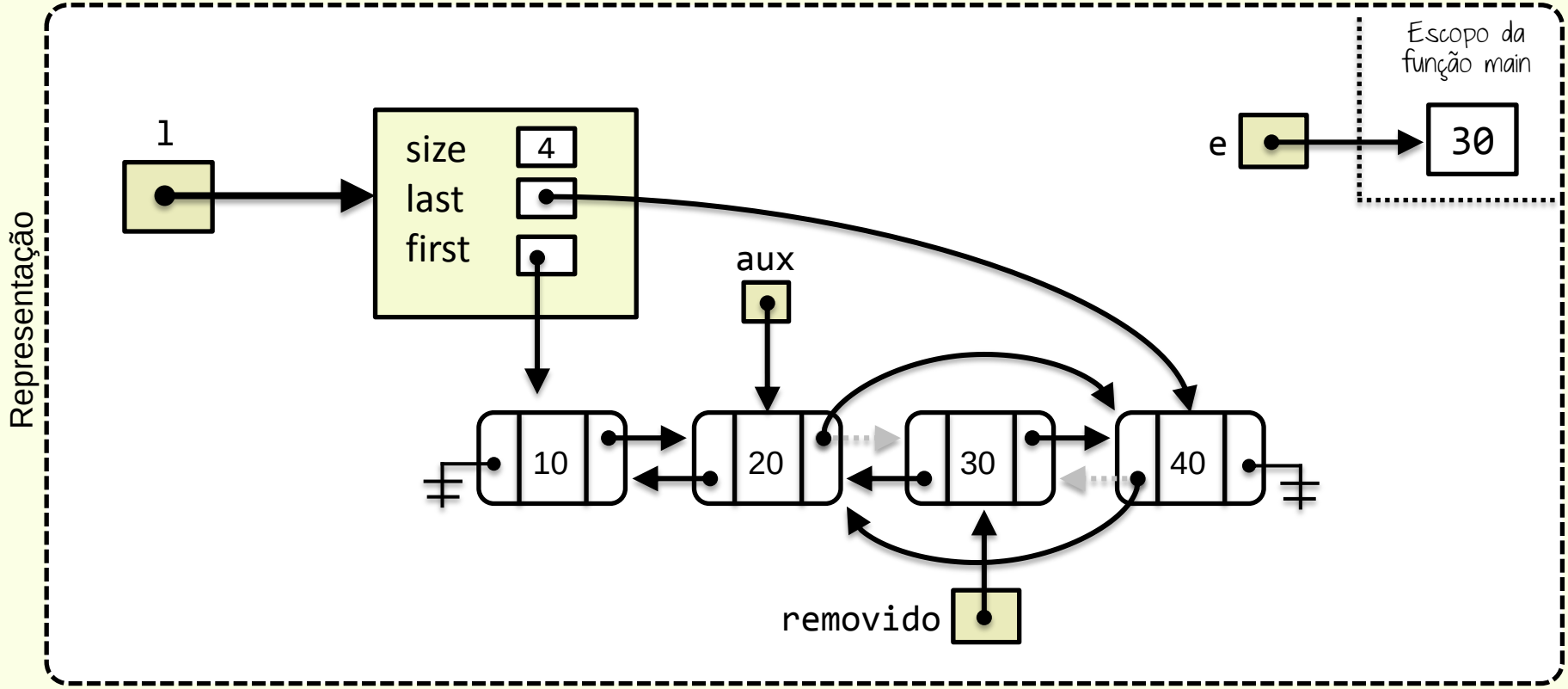
1 2 3 Remoção do elemento que é está no meio da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

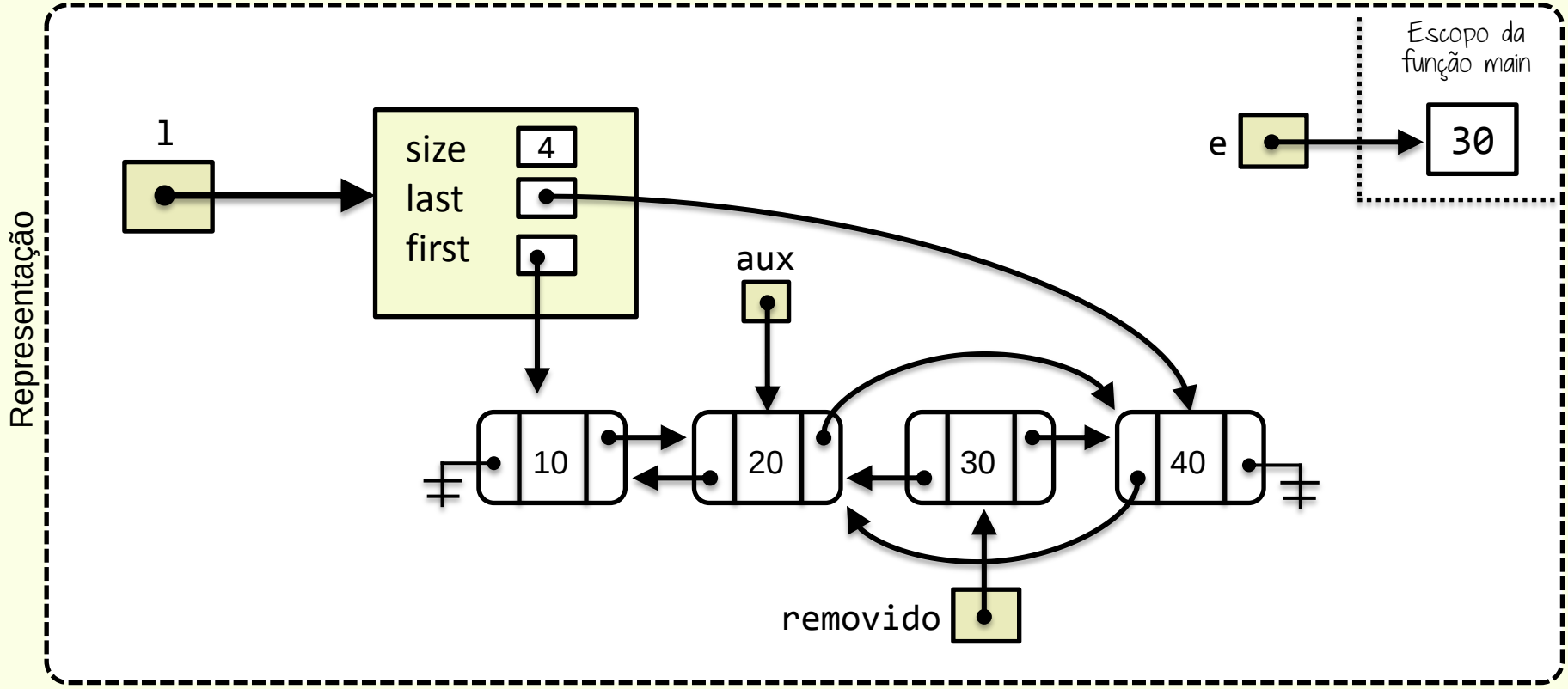
1 2 3 Remoção do elemento que é está no meio da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

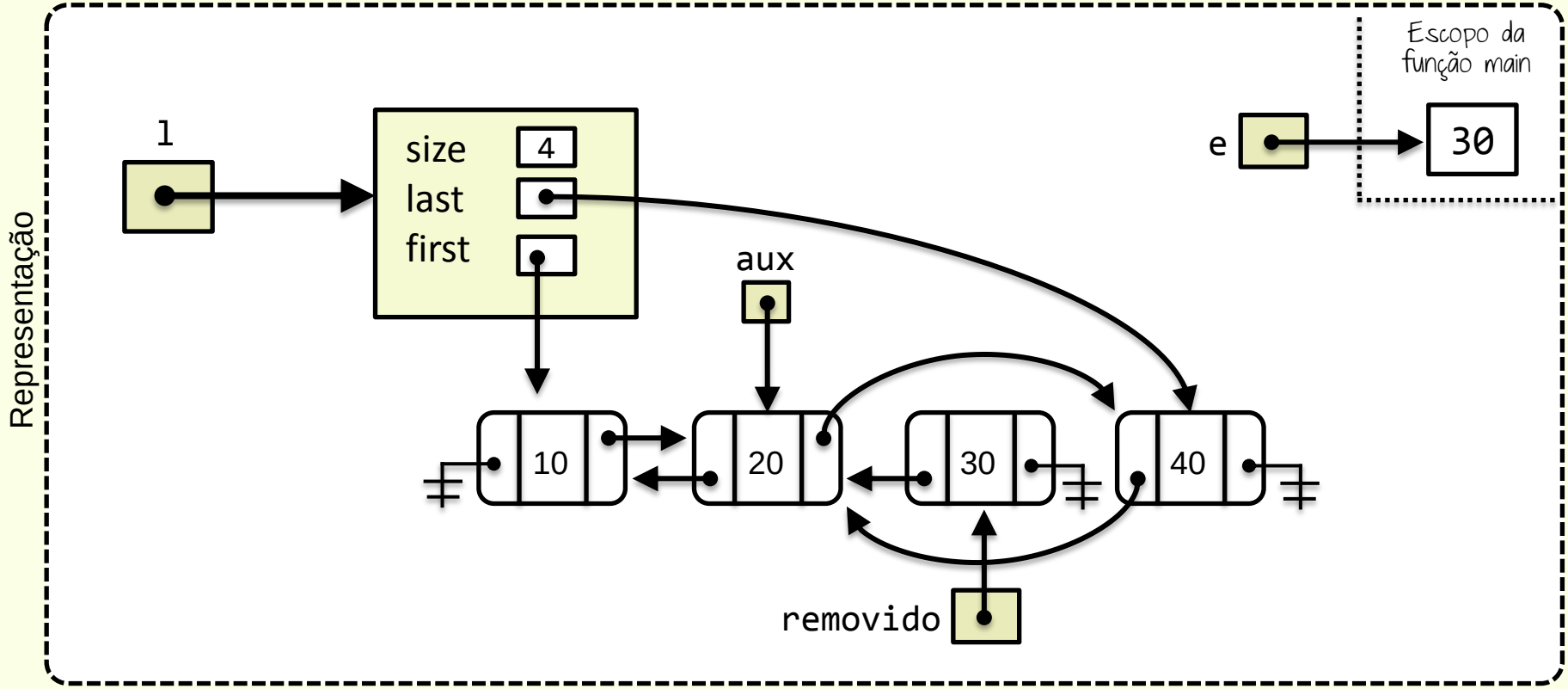
1 2 3 Remoção do elemento que é está no meio da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

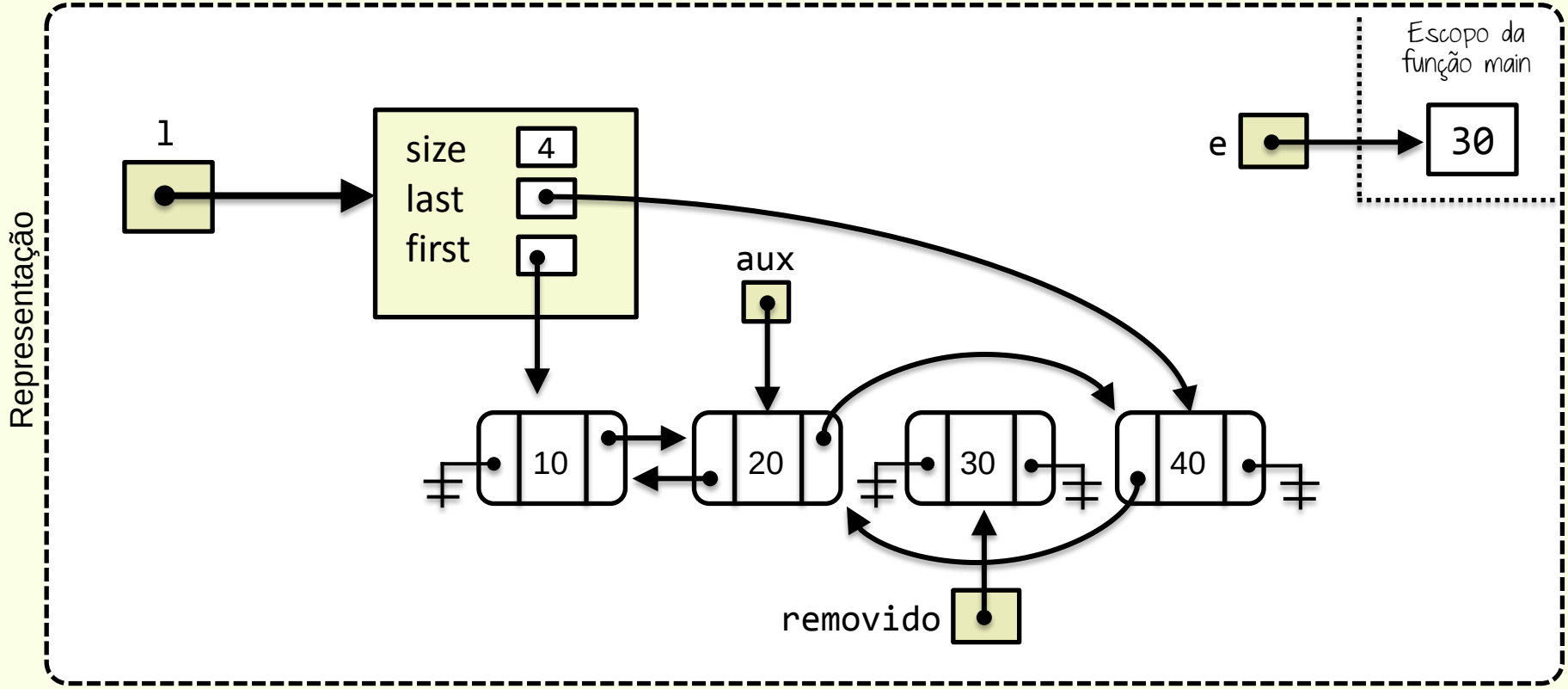
1 2 3 Remoção do elemento que é está no meio da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

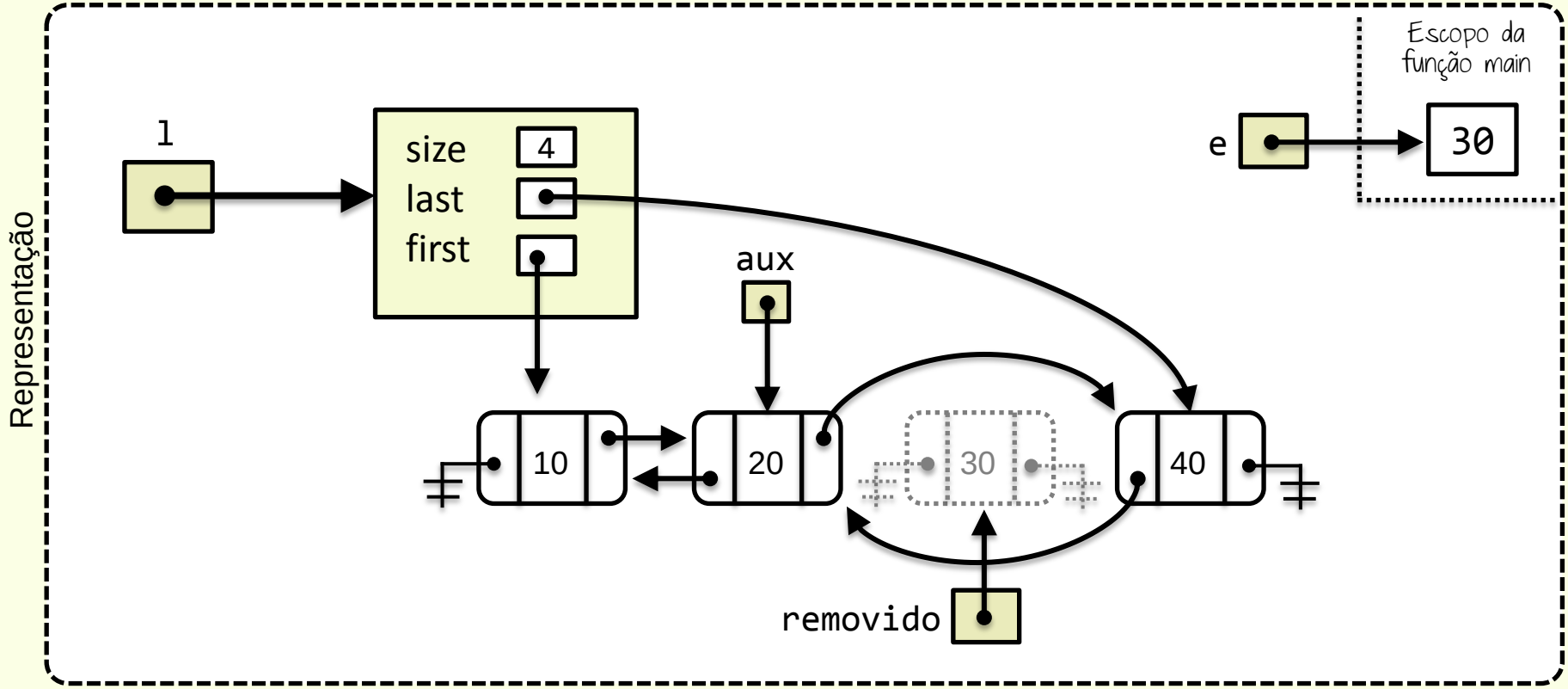
1 2 3 Remoção do elemento que é está no meio da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

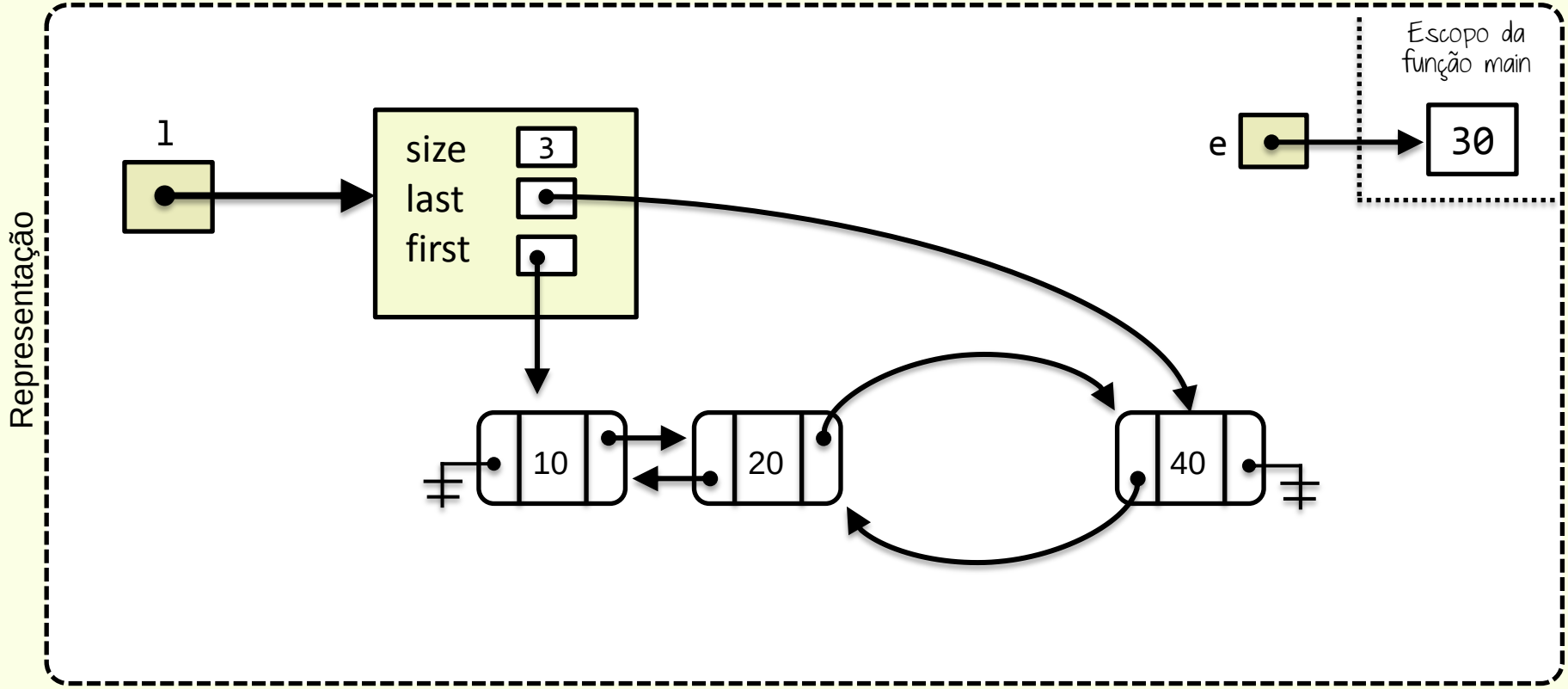
1 2 3 Remoção do elemento que é está no meio da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

1 2 3 Remoção do elemento que é está no meio da lista



removeElementList

```
int removeElementList(List* l, ItemType* e);
```

1 2 3 Remoção do elemento que é está no meio da lista