

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA/
PIAUÍ
Campus Teresina - Central

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E
TECNOLOGIA DO PIAUÍ

CAMPUS TERESINA-CENTRAL

DIRETORIA DE ENSINO

Estrutura de Dados

Listas Encadeadas

(Baseado no Cap. 9 do Livro “ESTRUTURAS DE DADOS EM C”
– autor: Silvio do Lago Pereira)

Professora: Elanne Cristina O. dos Santos

elannecristina.santos@gmail.com

elannecristina.santos@ifpi.edu.br

Fundamentos

- Uma lista encadeada é uma sequencia de nós, em que cada nó guarda um ***item*** e um ***ponteiro para o próximo*** elemento da sequencia. Ex. do livro:

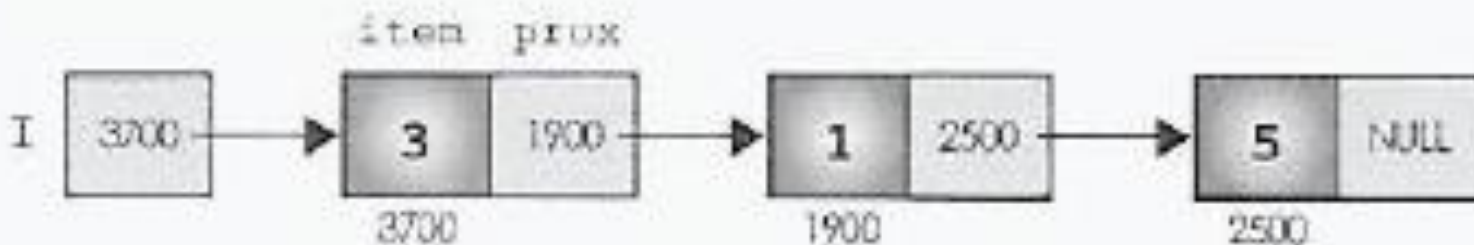


Figura 9.1 | Uma lista encadeada composta por três nós.

Fundamentos

- Elas são úteis em programas que precisam lidar com ***coleções dinâmicas***, cujas ***quantidades podem variar em tempo de execução***.
- Podem ser usadas para implementar ***pilhas*** e ***filas***, por exemplo.

Operações em Listas Encadeadas

- Para criar uma lista encadeada, é preciso definir **a estrutura dos nós** que serão usados em sua composição, bem como o **tipo de ponteiro** que será usado para **apontar seu nó inicial. Veja:**

```
#define fmt "%d"
using namespace std;
typedef int Item;
typedef struct no {
    Item item;
    struct no *prox;
}*Lista;
```

A estrutura **Lista** é um **ponteiro para struct no**, é usada para declarar **um ponteiro para a lista encadeada**. Assim, Suponha “l” é um ponteiro que aponta para o primeiro nó da lista. “l->item” representa o item guardado e “l->prox” é um ponteiro para o segundo elemento

Criação da Lista

- No exemplo vamos usar a função “inclue()”:

```
#include <stdio.h>
#include <iostream>
#define fmt "%d"
using namespace std;
typedef int Item;
typedef struct no {
    Item item;
    struct no *prox;
}*Lista;
Lista inclue(Item x, Lista p){
    Lista n = (Lista)malloc(sizeof(Lista));
    n->item = x;
    n->prox = p;
    return n;
}
```

Criação da Lista

INSERE UM ELEMENTO DE CADA VEZ

```
main(){  
    Lista l;  
    l = inclue(1,NULL);  
    l = inclue(2,l);  
    l = inclue(3,l);  
}
```

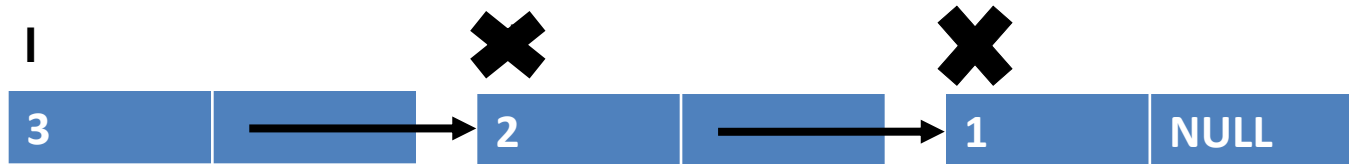
INSERE TRÊS ELEMENTOS NA LISTA

```
main(){  
    Lista l = inclue(3,inclue(2,inclue(1,NULL)));  
}
```

Como ficarão dispostos os elementos da lista?

```
main(){  
    Lista l;  
    l = inclue(1,NULL);  
    l = inclue(2,l);  
    l = inclue(3,l);  
}
```

```
main(){  
  
    Lista l = inclue(3,inclue(2,inclue(1,NULL)));  
}
```



- Como percorrer esta lista e imprimir todos os itens?

Como percorrer esta lista e imprimir todos os itens?

```
main(){
    Lista l;
    l = inclue(1, NULL);
    l = inclue(2, l);
    l = inclue(3, l);
    Lista atual=l;;
    while (atual!=NULL){
        cout<<atual->item<<endl;
        atual=atual->prox;
    }
}
```


Como percorrer esta lista e imprimir todos os itens?

```
void exhibe(Lista L){  
    while (L!=NULL){  
        cout<<L->item<<endl;  
        L=L->prox;  
    }  
}
```

```
main(){  
  
    Lista I = inclue(3,inclue(2,inclue(1,NULL)));  
    exhibe(I);  
}
```

Anexação de Listas

- Anexação ou concatenação é uma operação que anexa uma lista ao final de outra.
- Supondo o ponteiro H apontando para a lista [4,2] e o ponteiro I apontando para a lista [3,1,5]. Então após a anexação de I, H apontará para a lista [4,2,3,1,5].

Anexação de Listas

```
void anexa(Lista *A, Lista B){
```

```
    if (B==NULL) return;
```

```
    while (*A !=NULL)
```

```
        A = &(*A)->prox;
```

```
    *A=B;
```

```
}
```

```
main(){
```

```
    Lista H = inclue(4,inclue(2,NULL));
```

```
    Lista I = inclue(3,inclue(1,inclue(5,NULL)));
```

```
    anexa(&H,I);
```

```
    exhibe(H);
```

```
}
```

```
void anexar(Lista a,List b){
```

```
    if (b==0) return;
```

```
    while (a->prox!=0){
```

```
        a = a->prox;
```

```
    a->prox=b;
```

Anexação de Listas

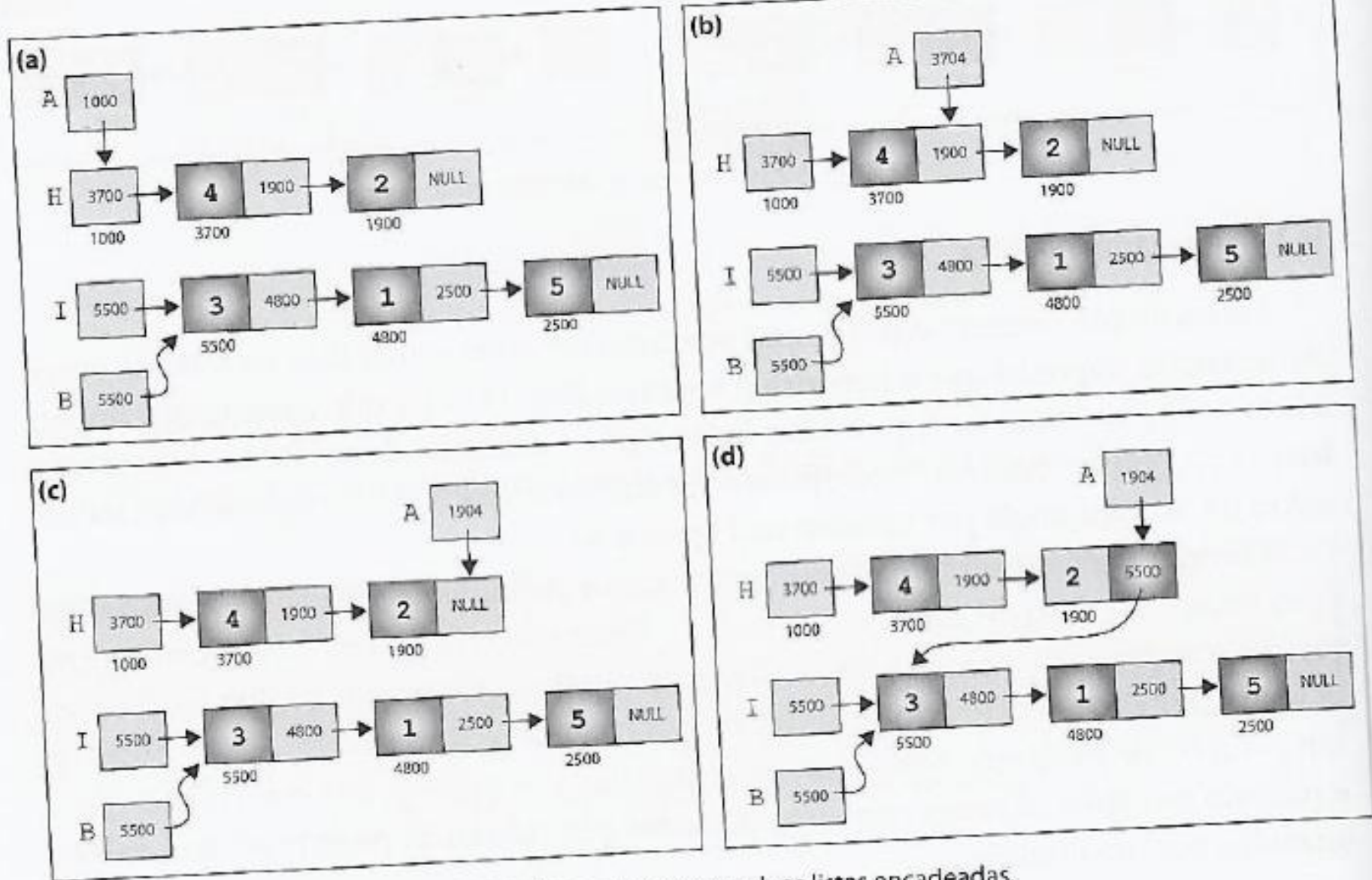
```
void anexa(Lista *A, Lista B){  
    if (B==NULL) return;  
    while (*A !=NULL)  
        A = &(*A)->prox;  
    *A=B;  
}
```

CHAMADA DA FUNÇÃO:

```
anexa(&H, I);
```

- PAG. 98: Note que a função “anexa” recebe como parâmetro o endereço do ponteiro H e o valor do ponteiro I. Como H é do tipo Lista(que já é um ponteiro), o primeiro parâmetro da função deve ser do tipo Lista *(isto é, um ponteiro para ponteiro). Nesse caso dizemos que H é passado como referência e que I é passado como valor.
- **PARA ALTERAR VALORES EM UMA FUNÇÃO É NECESSÁRIO PASSAR UM PONTEIRO, ISTO CARACTERIZA UMA PASSAGEM POR REFERÊNCIA EM C.**

Anexação de Listas – Exemplo do livro:



Anexação de Listas

- Observe que na função “anexa1” passamos como parâmetro o valor de A e não um endereço que aponta para outro endereço.
- Verifique se na função “anexa1()” o valor de L é alterado ao final da execução da função.

```
void anexa1(Lista A, Lista B){  
    if (B==NULL) return;  
    while (A->prox!=NULL)  
        A=A->prox;  
    A->prox=B;  
}
```

CHAMADA DA FUNÇÃO:

```
main(){  
    Lista L = inclue(40,inclue(20,NULL));  
    exibe(L);  
    Lista M = inclue(30,inclue(10,inclue(50,NULL)));  
    exibe(M);  
    anexa1(L,M);  
    exibe(L);  
}
```

Destruição da Lista

- Uma variável dinâmica só é destruída automaticamente quando a execução do programa termina.
- Para destruir uma variável dinâmica em tempo de execução podemos utilizar a função “free()”.

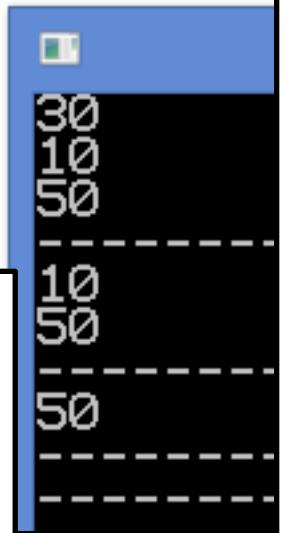
```
void destroi(Lista *L){  
    while (*L != NULL){  
        Lista n = *L;  
        *L = n->prox;  
        free(n);  
    }  
}
```

Destruição da Lista

- Exibindo os valores e chamando a função “destrói”:

```
main(){
    Lista M = inclue(30,inclue(10,inclue(50,NULL)));
    destroi(&M);
}
```

```
void destroi(Lista *a){
    Lista n = *a;
    cout<<"---percorre---"<<endl;
    mostra(n);
    while (*a!=0){
        n = *a;
        *a=n->prox;
        mostra(n);
        cout<<"-----retirando... "<<n->item<<endl;
        free(n);
    }
}
```



```
30
10
50
-----
10
50
-----
50
-----
```


Destruição da Lista

- Exemplo do livro:

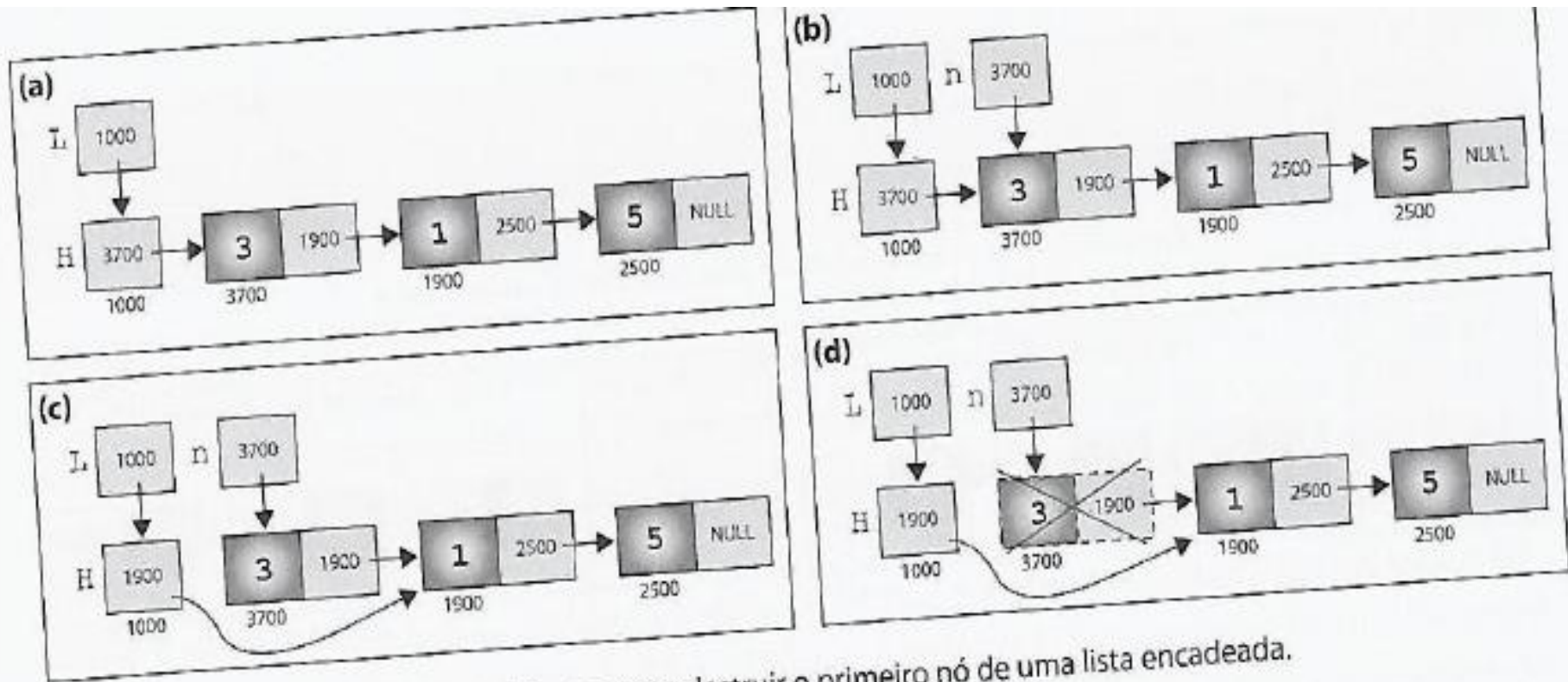


Figura 9.12 | Passos para destruir o primeiro nó de uma lista encadeada.

Obrigada pela atenção!!!
Boa semana de estudos pra vcs!!