Alocação Encadeada – Nós de cabeçalho

Ocasionalmente, é desejável manter um nó adicional no início de uma lista. Esse nó não representa um item (elemento) na lista e é chamado ***nó de cabeçalho*** ou ***cabeçalho de lista***. A parte ***inf*** desse nó cabeçalho poderia ficar sem uso. Mas, freqüentemente, a parte ***inf*** deste nó pode ser usada para manter informações globais sobre a lista. Por exemplo, a parte ***inf*** do nó de cabeçalho pode ser usada para armazenar o número de elementos na lista.

Alocação Encadeada – Nós de cabeçalho

A existência do nó cabeçalho, com a mesma estrutura de um elemento da lista, elimina a ocorrência de duas situações nas operações de inserção e remoção de elementos, como veremos.

Definiremos agora, um TAD LISTA_ENC_NC, o qual representa uma lista linear encadeada com a presença de um nó de cabeçalho contendo no campo inf o número de elementos contido na lista.

```
typedef struct nodo
{
   int inf;
   struct nodo * next;
}NODO;
typedef NODO * LISTA_ENC_NC;
void cria_lista (LISTA_ENC_NC *);
int eh_vazia (LISTA_ENC_NC);
int tam (LISTA_ENC_NC);
void ins (LISTA_ENC_NC *, int, int);
int recup (LISTA_ENC_NC, int);
void ret (LISTA_ENC_NC *, int);
```

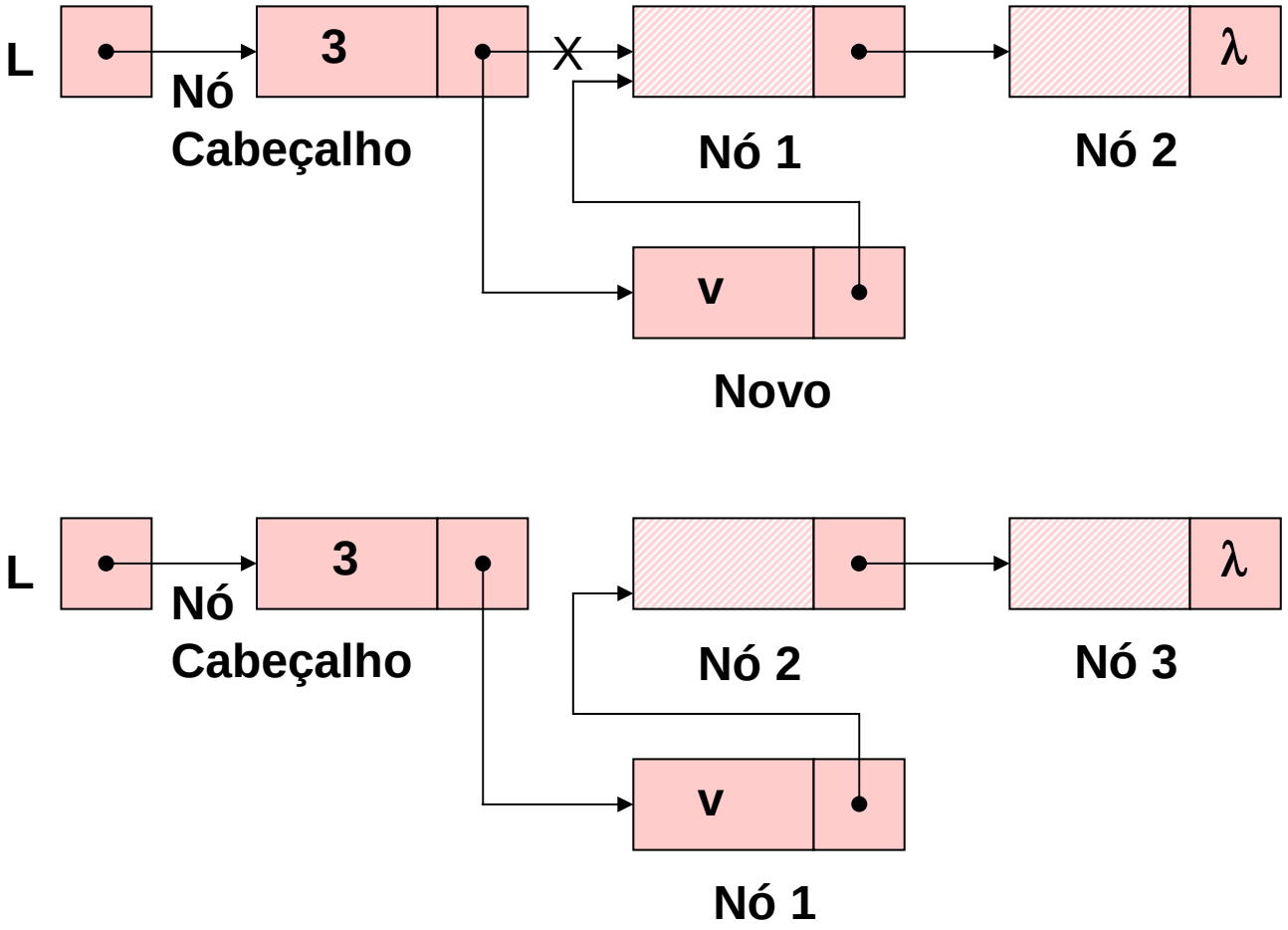
```
void cria_lista (LISTA_ENC_NC *pl)
{
   *pl = (NODO *) malloc (sizeof(NODO));
   if (!*pl)
   {
      printf ("\nERRO! Memoria
    insuficiente!\n");
      exit (2);
   }
   (*pl)->inf = 0;
   (*pl)->next = NULL;
}
```

```
int eh_vazia (LISTA_ENC_NC l)
{
   return (l->inf==0);
}


int tam (LISTA_ENC_NC l)
{
   return (l->inf);
}
```

Alocação Encadeada – Nós de cabeçalho

Esquema do processo da inserção de um novo nó na lista com nó cabeçalho.



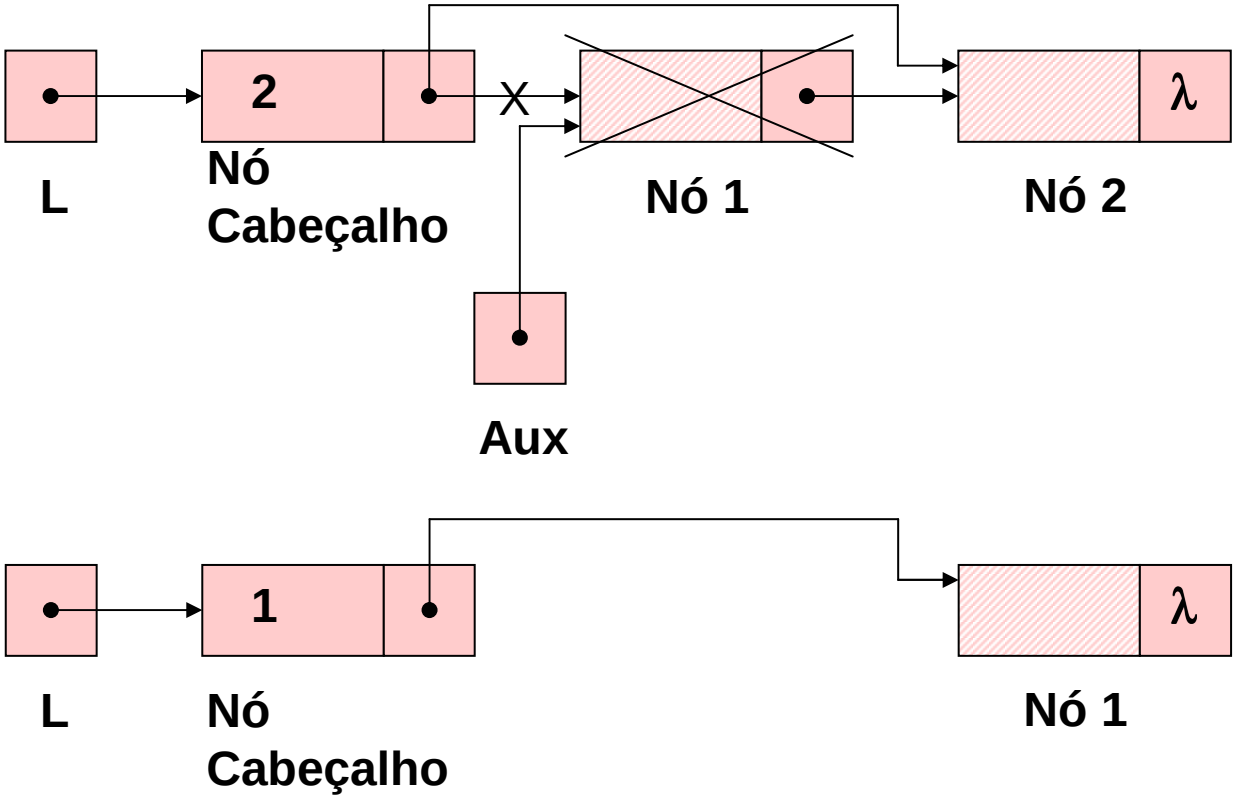
```
void ins (LISTA_ENC_NC *pl, int v, int k)
{
   NODO *novo, *aux;
   if (k < 1 || k > (*pl)->inf+1)
   {
      printf ("\nERRO! Posição invalida para insercao.\n");
      exit (1);
   }
   novo = (NODO *) malloc (sizeof(NODO));
   if (!novo)
   {
      printf ("\nERRO! Memoria insuficiente!\n");
      exit (2);
   }
```

```
    for (aux=*pl; k>1; aux=aux->next, k--);
    novo->inf = v;
    novo->next = aux->next;
    aux->next = novo;
    (*pl)->inf++;
}
```

```
int recup (LISTA_ENC_NC l, int k)
{
   if (k < 1 || k > l->inf)
   {
      printf ("\nERRO! Consulta invalida.\n");
      exit (3);
   }
   for (;k>0;k--)
      l=l->next;
   return (l->inf);
}
```

Alocação Encadeada – Nós de cabeçalho

Esquema do processo da retirada de um nó da lista com nó cabeçalho.



```
void ret (LISTA_ENC_NC *pl, int k)
{
   NODO *aux, *aux2;
   if (k < 1 || k > (*pl)->inf)
   {
      printf ("\nERRO! Posição invalida para retirada.\n");
      exit (4);
   }
   for (aux=*pl; k>1; k--, aux=aux->next);
   aux2 = aux->next;
   aux->next = aux2->next;
   free (aux2);
   (*pl)->inf--;
}
```

Alocação Encadeada – Nós de cabeçalho

Como vimos, a única operação que requer alteração para transformarmos o TAD LISTA_ENC no TAD LISTA_ENC_ORD é a operação de inserção, o mesmo ocorre com os TAD´s LISTA_ENC_NC e LISTA_ENC_NC_ORD.

Implementaremos agora esta operação.

```
void ins (LISTA_ENC_NC *pl, int v)
{
   NODO *novo, *aux;
   novo = (NODO *) malloc (sizeof(NODO));
   if (!novo)
   {
      printf ("\nERRO! Memoria insuficiente!\n");
      exit (2);
   }
   for (aux=*pl; aux->next!=NULL && v>(aux-
     >next)->inf; aux=aux->next);
   novo->inf = v;
   novo->next = aux->next;
   aux->next = novo;
   (*pl)->inf++; }
```

Alocação Encadeada- Nós de cabeçalho- Exercício

Implemente, no TAD LISTA_ENC_NC_ORD, a seguinte operação:

LISTA_ENC_NC_ORD concatenar (LISTA_ENC_NC_ORD, LISTA_ENC_NC_ORD);

a qual recebe duas listas e retorna uma lista resultante da concatenação das mesmas. ***OBS. A lista resultante não deve apresentar elementos com mesmo campo inf.***