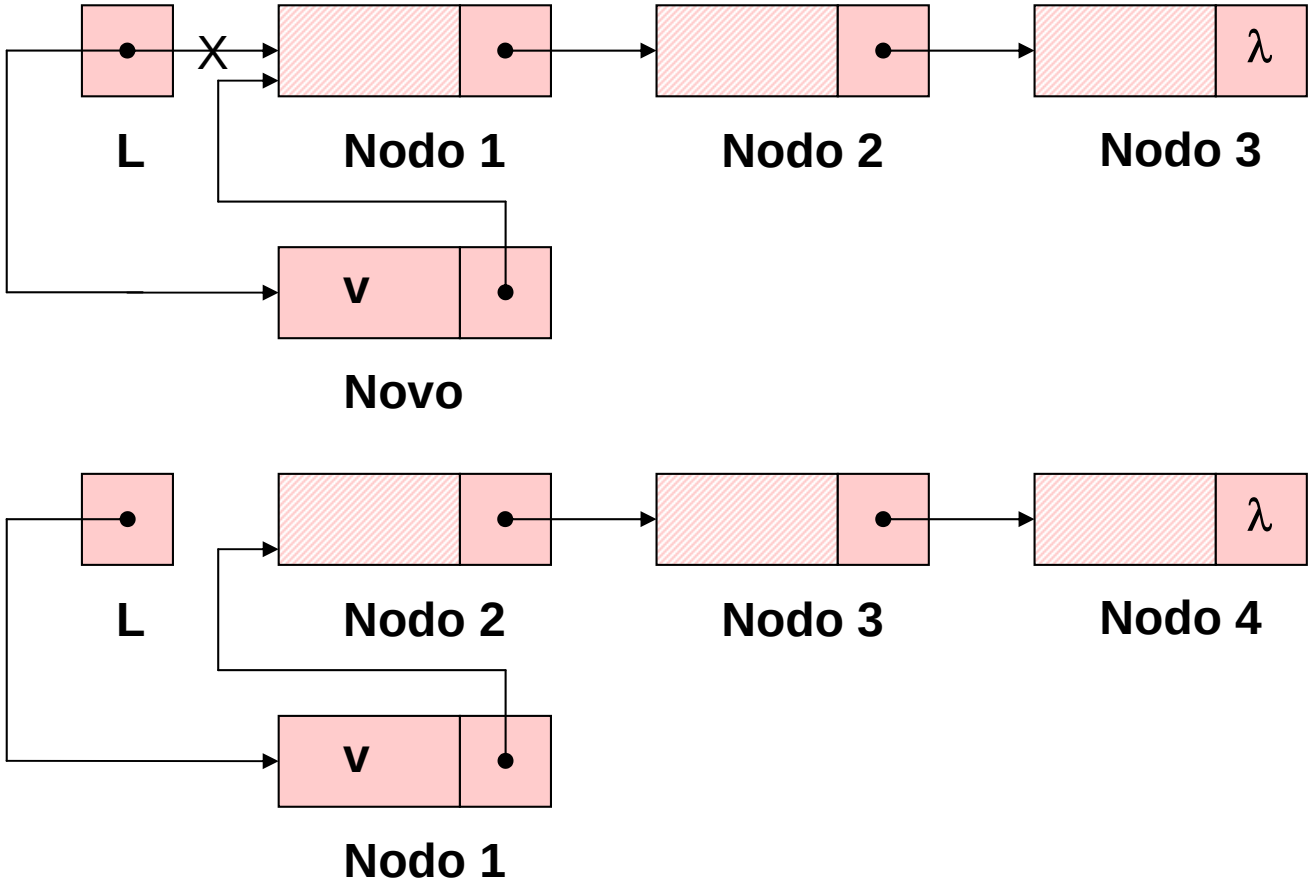
```
int tam (LISTA_ENC l)
{
   int cont;
   for (cont=0; l!= NULL; cont++)
      l = l->next;
   return (cont);
}
```

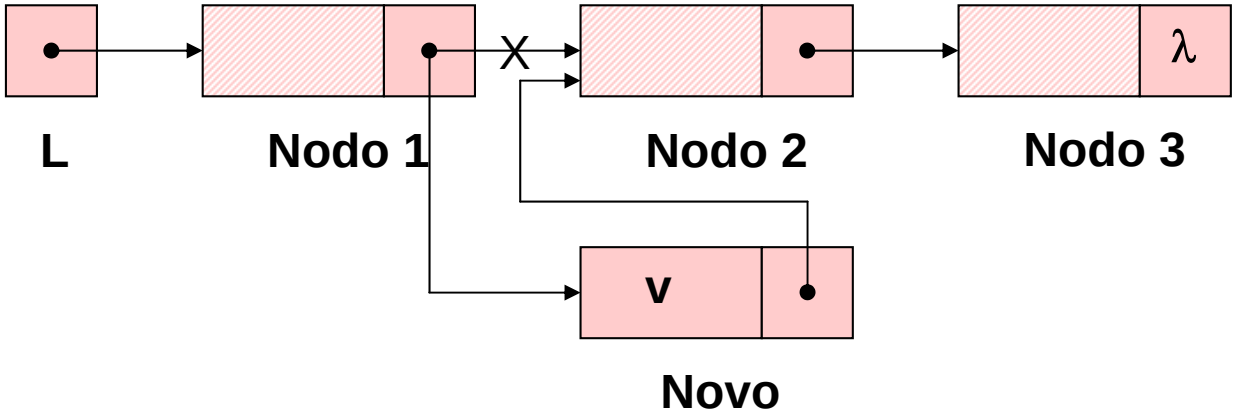
Alocação Encadeada

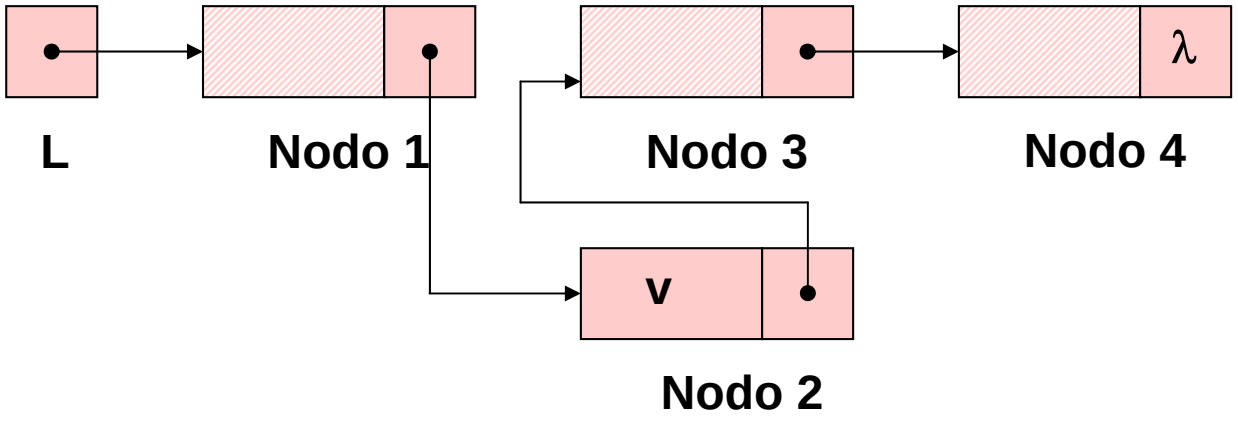
Esquema do processo da inserção de um novo nó na lista. (situação um)



Alocação Encadeada

Esquema do processo da inserção de um novo nó na lista. (situação dois)





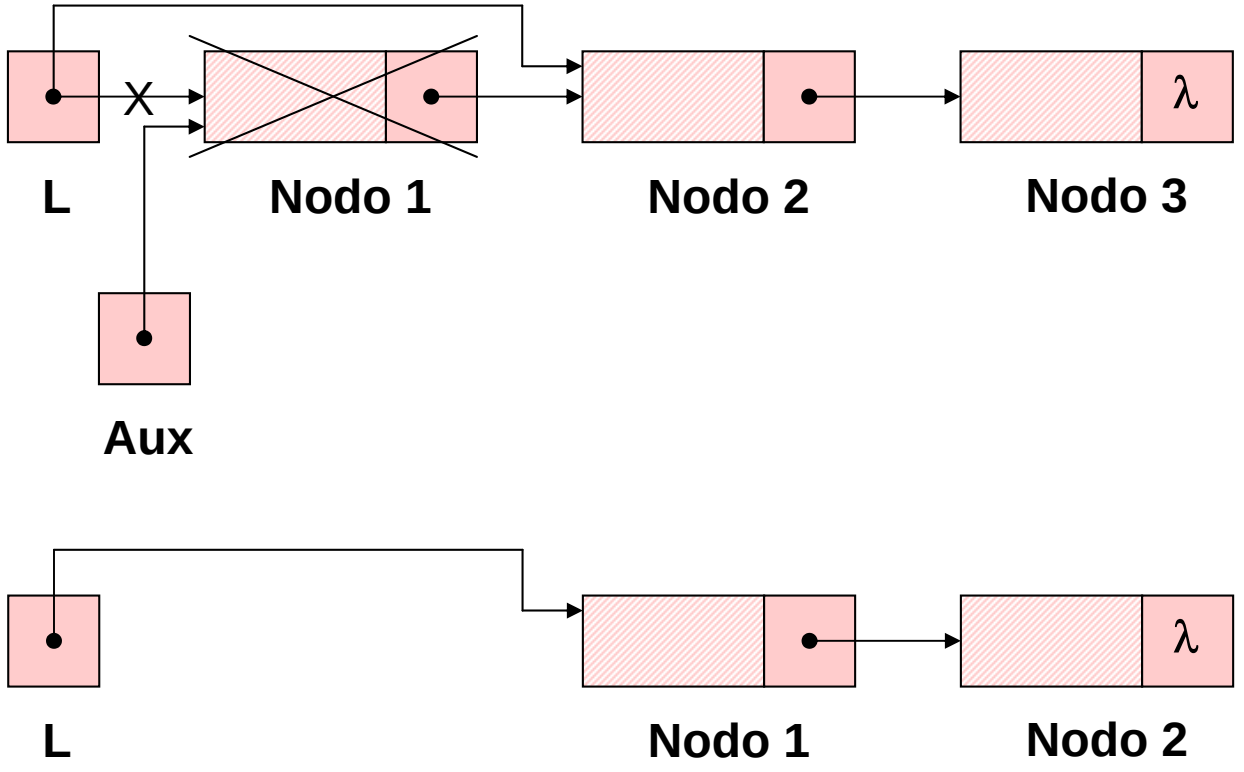
```
void ins (LISTA_ENC *pl, int v, int k)
{
   NODO *novo;
   if (k < 1 || k > tam(*pl)+1)
   {
      printf ("\nERRO! Posição invalida para
    insercao.\n");
      exit (1);
   }
   novo = (NODO *) malloc (sizeof(NODO));
   if (!novo) { printf ("\nERRO! Memoria insuficiente!\n");
                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

```
    novo->inf = v;
    if (k==1){
       novo->next = *pl;
       *pl = novo;
    }
    else {
       LISTA_ENC aux;
       for (aux=*pl; k>2; aux=aux->next, k--);
       novo->next = aux->next;
       aux->next = novo;
    }
}
```

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

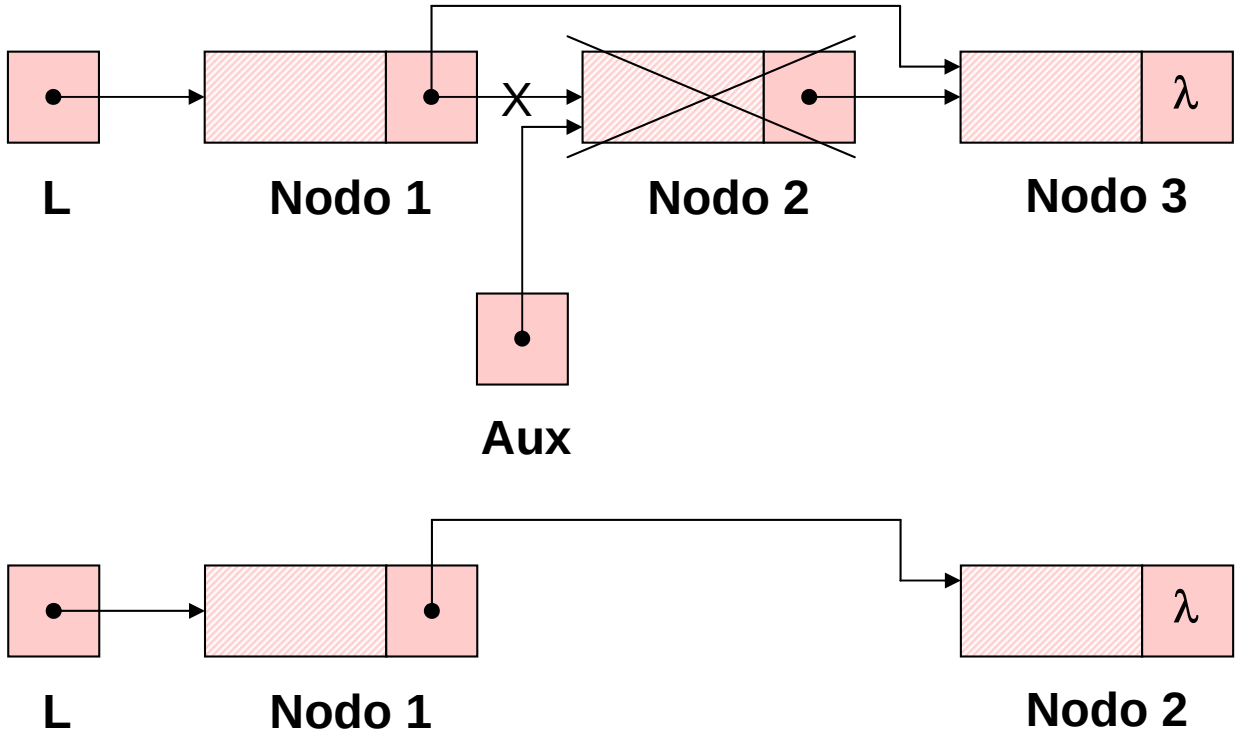
Alocação Encadeada

Esquema do processo da retirada de um nó da lista. (situação um)



Alocação Encadeada

Esquema do processo da retirada de um nó da lista. (situação dois)



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

```
    else
    {
        NODO *aux2;
        for (aux=*pl; k>2; k--, aux=aux->next);
        aux2 = aux->next;
        aux->next = aux2->next;
        free (aux2);
    }
}
```

Alocação Encadeada - Exercício

Implemente, no TAD LISTA_ENC, a seguinte operação:

int pertence (LISTA_ENC l, int v)

a qual retorna 1 (um) se v pertence a lista l e 0 (zero) caso contrário.

Alocação Encadeada - Exercício

Implemente, no TAD LISTA_ENC, a seguinte operação:

int eh_ord (LISTA_ENC l)

a qual retorna 1 (um) se a lista l está em ordem crescente e 0 (zero) caso contrário.

UNIVASF

Alocação Encadeada - Exercício

Implemente, no TAD LISTA_ENC, utilizando recursividade, a seguinte operação:

void gera_lista (LISTA_ENC *pl,int m,int n)

a qual utilizando-se das operações do TAD LISTA produz uma lista de inteiros correspondente a [m..n].

Alocação Encadeada

Com base em nossos novos conhecimentos adquiridos, podemos agora definir um novo TAD LISTA_ENC_ORD, na qual os elementos encontram-se ordenados de forma crescente ou decrescente, ou seja, no caso da ordenação crescente, o primeiro elemento é menor que o segundo, que por sua vez é menor que o terceiro e assim sucessivamente. (Para exemplificar, consideraremos a ordem crescente)

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

Alocação Encadeada

Com uma pequena análise, percebe-se que a única operação que requer alteração do TAD LISTA_ENC para o TAD LISTA_ENC_ORD é a operação de inserção.

Implementaremos agora esta operação.