

## Alocação Encadeada – Nós de cabeçalho

Este caso particular, da existência do nó cabeçalho com a mesma estrutura de um elemento da lista, elimina a ocorrência de duas situações na operação de inserção.

```

void ins (LISTA_ENC_NC l, int v, int k)
{
    NODO *novo, *aux;
    if (k < 1 || k > l->inf/*ou tam()*/+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;  
for (aux=l; k>1; aux=aux->next, k--);  
novo->next = aux->next;  
aux->next = novo;  
l->inf++;  
}
```

## Alocação Encadeada

Com base no que foi visto implemente a operação `recup()` que compõem o TAD `LISTA_ENC_NC`.

```
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);
```

```
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

Com base no que foi visto implemente a operação `ret()` que compõem o TAD `LISTA_ENC_NC`.

```
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);
```

## Alocação Encadeada – Nós de cabeçalho

Como ocorreu na operação de inserção, este caso particular, da existência do nó cabeçalho com a mesma estrutura de um elemento da lista, também elimina a ocorrência de duas situações na operação de remoção.

```

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

Com base na operação de inserção implementada no TAD LISTA\_ENC\_ORD implemente a operação de inserção de um elemento no TAD LISTA\_ENC\_NC\_ORD.

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

```

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