

## Listas Duplamente Encadeadas

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

```
1  typedef struct nodo
2  {
3      int inf;
4      struct nodo * ant;
5      struct nodo * prox;
6  }NODO;
7  typedef NODO * LISTA_DUP_ENC;
8  void cria_lista (LISTA_DUP_ENC *);
9  int eh_vazia (LISTA_DUP_ENC);
10 int tam (LISTA_DUP_ENC);
11 void ins (LISTA_DUP_ENC *, int, int);
12 int recup (LISTA_DUP_ENC, int);
13 void ret (LISTA_DUP_ENC *, int);
14 void destruir (LISTA_DUP_ENC);
```

```
1  int recup (LISTA_DUP_ENC l, int k)
2  {
3      if (k < 1 || k > tam(l))
4      {
5          printf ("\nERRO! Consulta invalida.\n");
6          exit (3);
7      }
8      for (;k>1;l=l->prox,k--) ;
9      return (l->inf);
10 }
```

## Listas Duplamente Encadeadas

Com base no que foi visto implemente a operação ret() que compõem o T A D LISTA\_DUP\_ENC.

```
1  typedef struct nodo
2  {
3      int inf;
4      struct nodo * ant;
5      struct nodo * prox;
6  }NODO;
7  typedef NODO * LISTA_DUP_ENC;
8  void cria_lista (LISTA_DUP_ENC *);
9  int eh_vazia (LISTA_DUP_ENC);
10 int tam (LISTA_DUP_ENC);
11 void ins (LISTA_DUP_ENC *, int, int);
12 int recup (LISTA_DUP_ENC, int);
13 void ret (LISTA_DUP_ENC *, int);
14 void destruir (LISTA_DUP_ENC);
```

## Listas Duplamente Encadeadas

### Dicas:

A posição é válida?

Todas as situações de remoção são tratadas da mesma forma?

```
1 void ret (LISTA_DUP_ENC *p1, int k) {
2     LISTA_DUP_ENC aux;
3     if (k < 1 || k > tam(*p1)) {
4         printf ("\nERRO! Posição invalida para retirada.\n");
5         exit (4);
6     }
7     if (k==1) { /*situações um e dois*/
8         aux = *p1;
9         *p1 = aux->prox;
10        if (*p1) /*situação dois*/
11            (*p1)->ant=NULL;
12        free (aux);
13    } else { /*situações três e quatro*/
14        for (aux=(*p1)->prox; k>2; k--, aux=aux->prox);
15        aux->ant->prox = aux->prox;
16        if (aux->prox) /*situação quatro*/
17            aux->prox->ant = aux->ant;
18        free (aux);
19    }
20 }
```

## Listas Duplamente Encadeadas

Com base no que foi visto implemente a operação destruir() que compõem o TAD LISTA\_DUP\_ENC.

```
1  typedef struct nodo
2  {
3      int inf;
4      struct nodo * ant;
5      struct nodo * prox;
6  }NODO;
7  typedef NODO * LISTA_DUP_ENC;
8  void cria_lista (LISTA_DUP_ENC *);
9  int eh_vazia (LISTA_DUP_ENC);
10 int tam (LISTA_DUP_ENC);
11 void ins (LISTA_DUP_ENC *, int, int);
12 int recup (LISTA_DUP_ENC, int);
13 void ret (LISTA_DUP_ENC *, int);
14 void destruir (LISTA_DUP_ENC);
```

```
1 void destruire (LISTA_DUP_ENC l)
2 {
3     LISTA_DUP_ENC aux;
4     while (l)
5     {
6         aux = l;
7         l = l->prox;
8         free (aux);
9     }
10 }
```

## Listas Duplamente Encadeadas

Implemente, no TAD LISTA\_DUP\_ENC, a seguinte operação:

```
void inverter_lista (LISTA_DUP_ENC *pl);
```

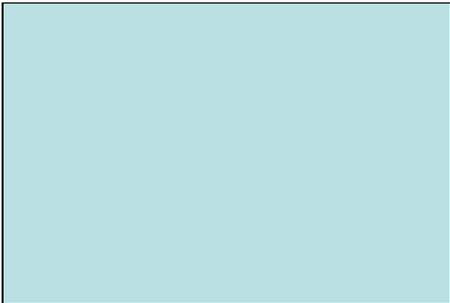
a qual recebe uma referência para uma lista duplamente encadeada e inverte a ordem de seus elementos.

```
1 void inverter_lista (LISTA_DUP_ENC *p1)
2 {
3     if (*p1) {
4         LISTA_DUP_ENC aux; /* NODO *aux; */
5         do {
6             aux=(*p1)->prox;
7             (*p1)->prox=(*p1)->ant;
8             (*p1)->ant=aux;
9             if (aux)
10                *p1=aux;
11        }while (aux);
12    }
13 }
```

## Listas Duplamente Encadeadas

Com base no que foi visto sobre o TAD LISTA\_DUP\_ENC, implemente, utilizando recursividade, as operações que julgar viáveis.

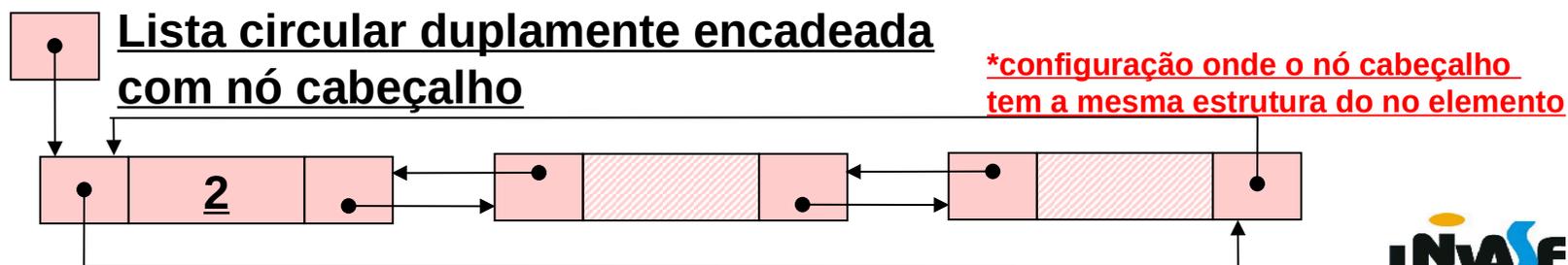
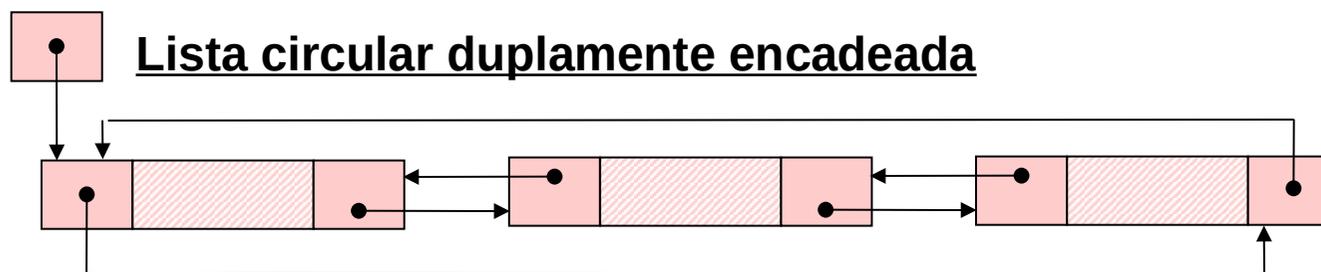
```
1  typedef struct nodo
2  {
3      int inf;
4      struct nodo * ant;
5      struct nodo * prox;
6  }NODO;
7  typedef NODO * LISTA_DUP_ENC;
8  void cria_lista (LISTA_DUP_ENC *);
9  int eh_vazia (LISTA_DUP_ENC);
10 int tam (LISTA_DUP_ENC);
11 void ins (LISTA_DUP_ENC *, int, int);
12 int recup (LISTA_DUP_ENC, int);
13 void ret (LISTA_DUP_ENC *, int);
14 void destruir (LISTA_DUP_ENC);
```



**Listas Circulares  
Duplamente Encadeadas  
com Nó Cabeçalho**

## Listas Dup. Encad. Circ. com NC

Também podemos construir *listas circulares duplamente encadeadas* ou *listas circulares duplamente encadeadas com nó cabeçalho*.



```
1  typedef struct nodo
2  {
3      int inf;
4      struct nodo * ant;
5      struct nodo * prox;
6  }NODO;
7  typedef NODO * LISTA_CIR_DUP_ENC_NC;
8  void cria_lista (LISTA_CIR_DUP_ENC_NC *);
9  int eh_vazia (LISTA_CIR_DUP_ENC_NC);
10 int tam (LISTA_CIR_DUP_ENC_NC);
11 void ins (LISTA_CIR_DUP_ENC_NC, int, int);
12 int recup (LISTA_CIR_DUP_ENC_NC, int);
13 void ret (LISTA_CIR_DUP_ENC_NC, int);
14 void destruir (LISTA_CIR_DUP_ENC_NC);
```