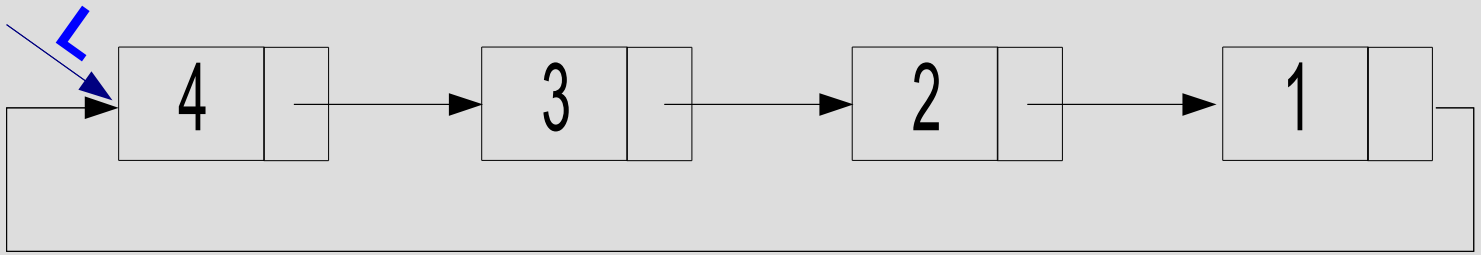
Estruturas de Dados

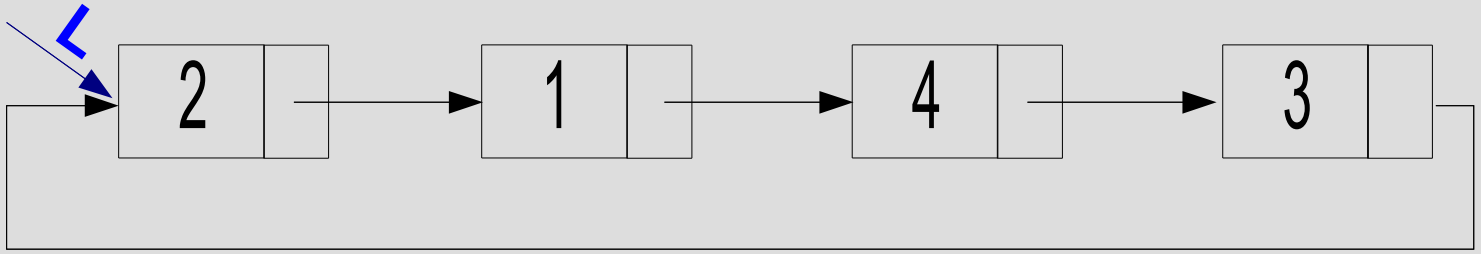
Lista Circular e
Duplamente Encadeada

Lista Circular

- Algumas aplicações necessitam representar conjuntos cíclicos, a exemplo de figuras geométricas.
- Em uma lista circular o último elemento tem o primeiro elemento como próximo



- A lista pode então ser representada por qualquer elemento da lista

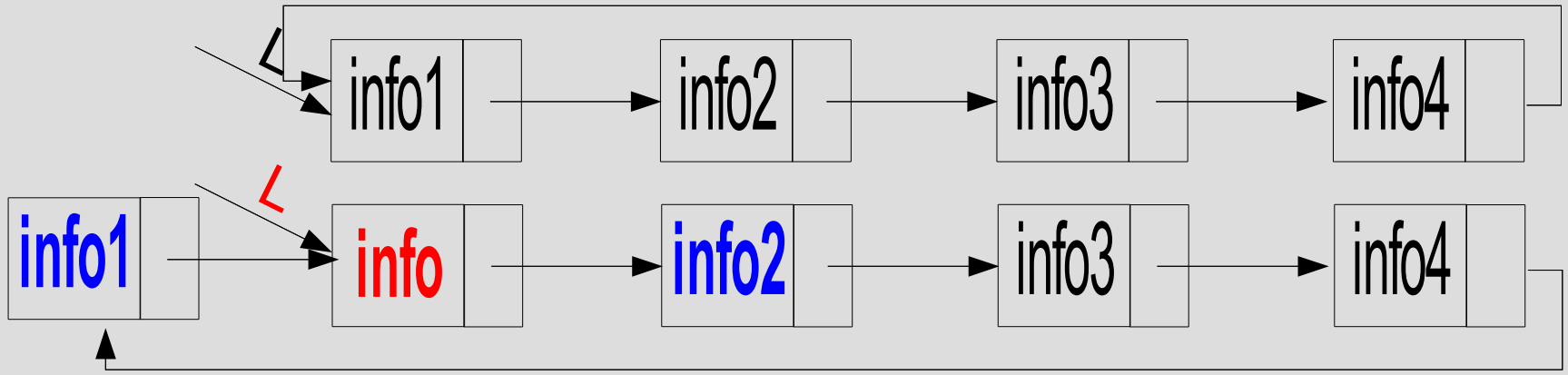


Tipo Abstrato de Dado Lista Circular

Podemos criar um TAD Lista Circular de Inteiros. Para tanto, devemos criar o arquivo lista_circular.h com o nome do tipo e os protótipos.

```
typedef struct lista_circ ListaCirc;

/* Cria uma lista circular vazia.*/
ListaCirc* lst_circ_cria();
/* Testa se uma lista circular é vazia.*/
int lst_circ_vazia(ListaCirc *l);
/* Insere um elemento em uma lista circular.*/
ListaCirc* lst_circ_insere(ListaCirc *l, int info);
/* Busca um elemento em uma lista circular.*/
ListaCirc* lst_circ_busca(ListaCirc *l, int info);
/* Imprime uma lista circular.*/
void lst_circ_imprime(ListaCirc *l);
/* Remove um elemento de uma lista circular.*/
ListaCirc* lst_circ_remove(ListaCirc *l, int info);
/* Libera o espaço alocado por uma lista circular .*/
void lst_circ_libera(ListaCirc *l);
```

TAD Lista Circular
Função de Inserção




```
ListaCirc* lst_circ_insere(ListaCirc *l, int info){
    ListaCirc* ln = (ListaCirc*)malloc(sizeof(ListaCirc));
    ln->info = info;
    if(l==NULL)
        ln->prox = ln;
    else{
        ln->prox = l->prox;
        l->prox = ln;
   }
    return ln;
}
```

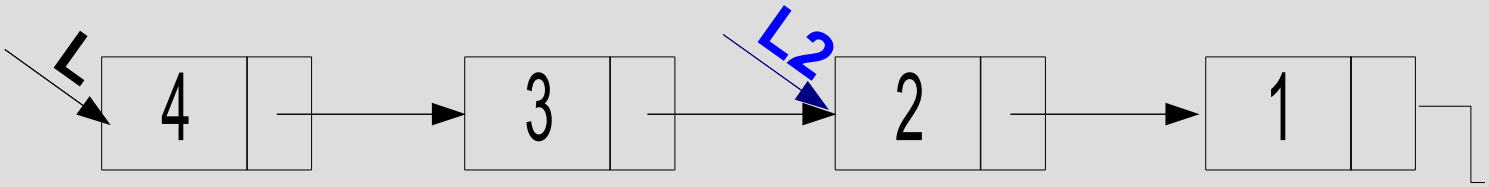
TAD Lista Circular
Função que imprime

A partir de um elemento l, percorre-se elemento a elemento, imprimindo os elementos, até alcançar o elemento l

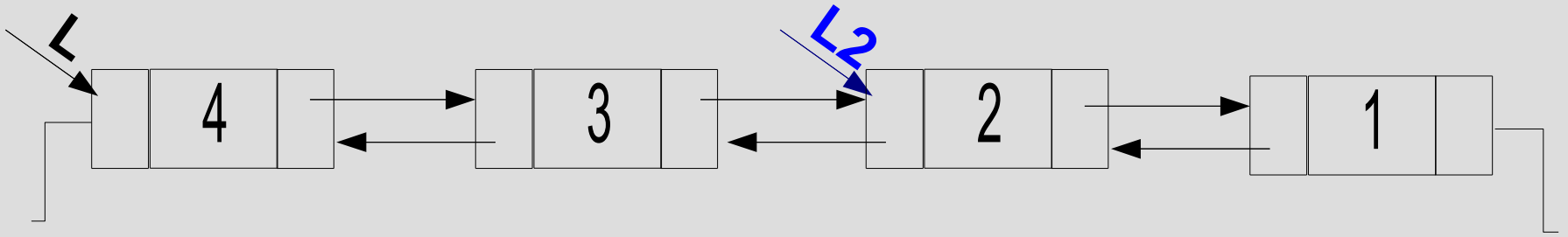
```
void lst_circ_imprime(ListaCirc *l){
 if(l!=NULL){
   ListaCirc* lAux = l;
   printf("Lista de Elementos \n");
   do{
      printf("Info = %d\n",lAux->info);
      lAux = lAux->prox;
   }while(l!=lAux);
 }
}
```

Lista Duplamente Encadeada

- Alguns problemas com listas encadeadas simples:
  - Não temos como percorrer a lista na ordem inversa
  - Dificulta a retirada de um elemento da lista



- Para solucionarmos, podemos considerar uma lista que tenha um ponteiro para o próximo e para o anterior

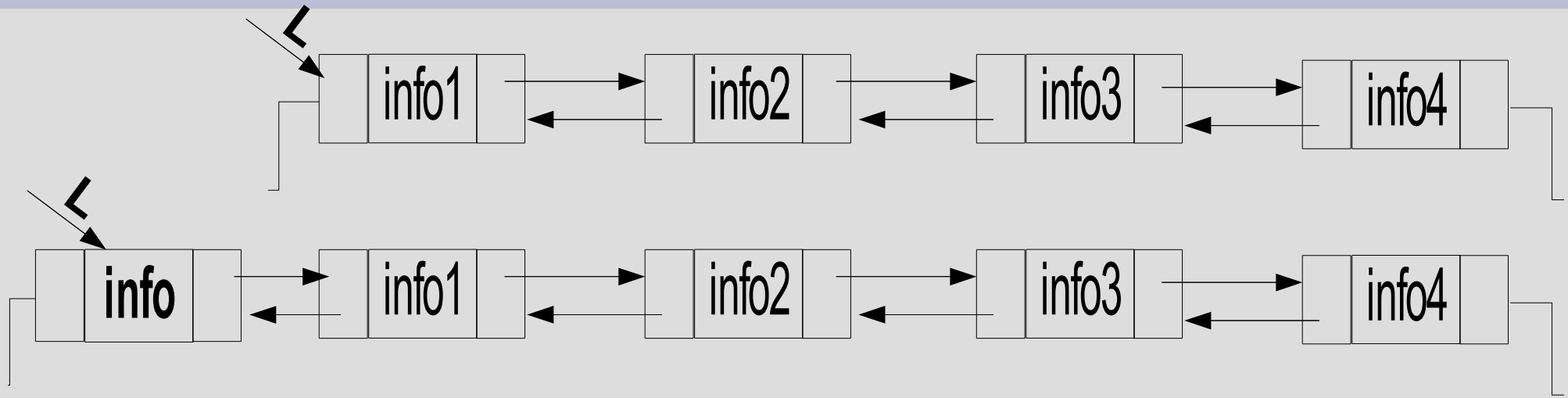


Tipo Abstrato de Dado Lista Duplamente Encadeada

Podemos criar um TAD Lista Duplamente Encadeada de Inteiros. Para tanto, devemos criar o arquivo lista_dupl.h com o nome do tipo e os protótipos.

```
typedef struct lista_dupl ListaDupl;

/* Cria uma lista dupl. encadeada vazia.*/
ListaDupl* lst_dupl_cria();
/* Testa se uma lista dupl. encadeada é vazia.*/
int lst_dupl_vazia(ListaDupl *l);
/* Insere um elemento no início da lista dupl. encadeada.*/
ListaDupl* lst_dupl_insere(ListaDupl *l, int info);
/* Imprime uma lista dupl. encadeada.*/
void lst_dupl_imprime(ListaDupl *l);
/* Busca um elemento em uma lista dupl. encadeada.*/
ListaDupl* lst_dupl_busca(ListaDupl *l, int info);
/* Remove um elemento de uma lista dupl. encadeada.*/
ListaDupl* lst_dupl_remove(ListaDupl *l, int info);
/* Libera o espaço alocado por uma lista dupl. encadeada.*/
void lst_dupl_libera(ListaDupl *l);
```

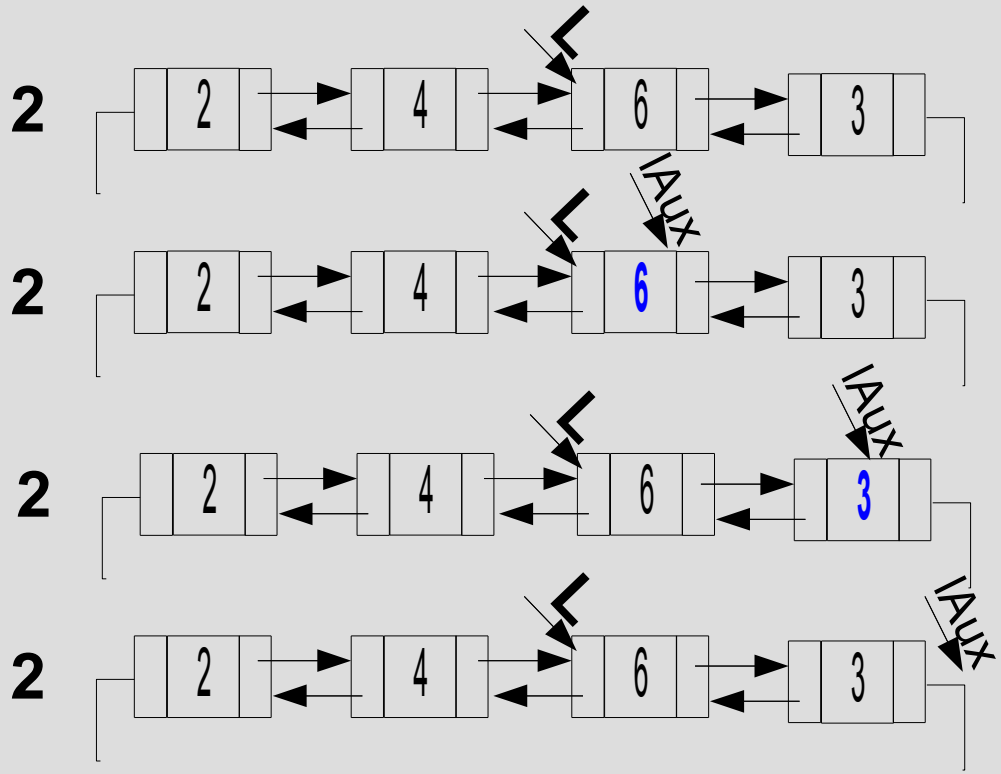
TAD Lista Dupl. Encadeada
Função de Inserção



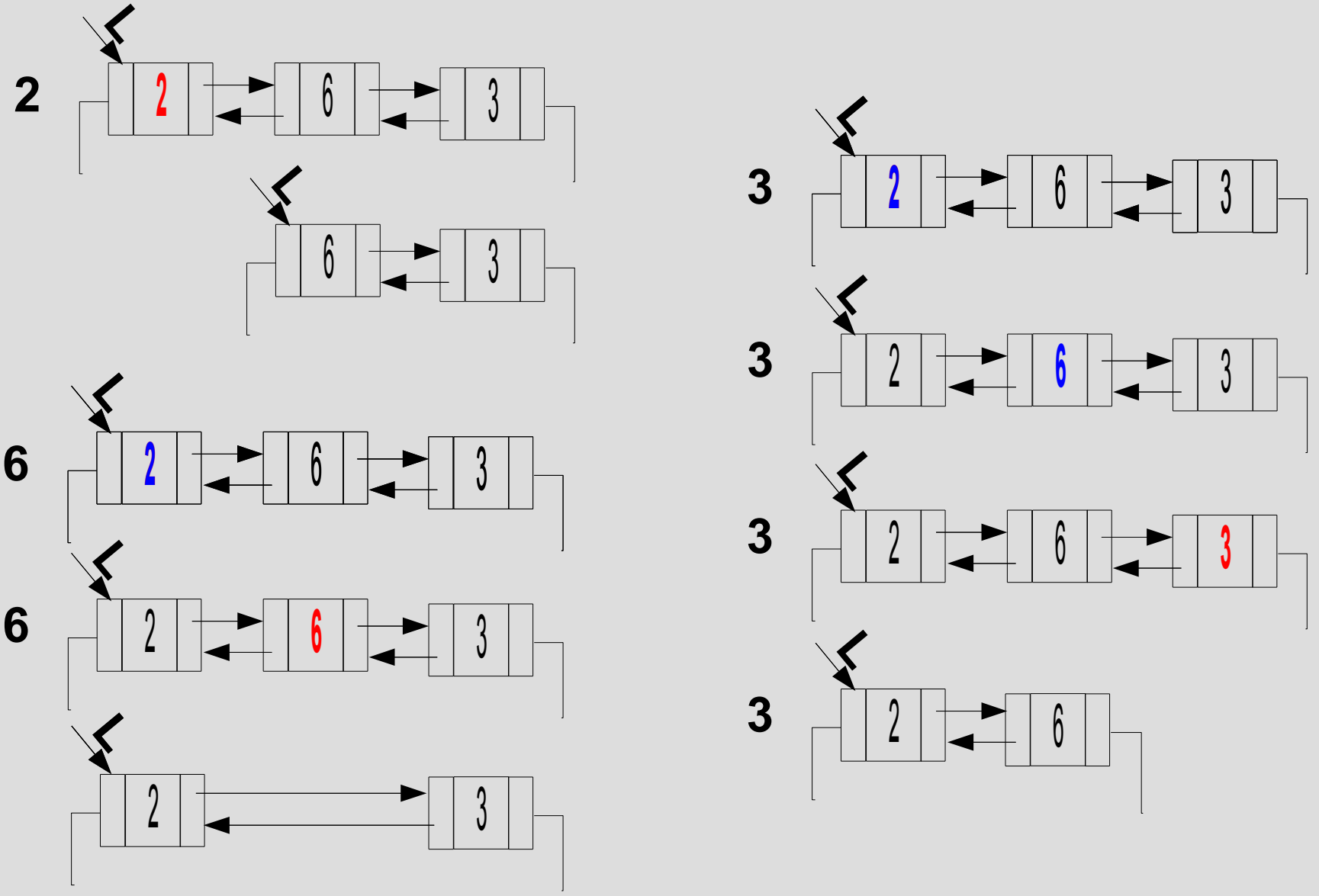

```
ListaDupl* lst_dupl_insere(ListaDupl *l, int info){
    ListaDupl* ln = (ListaDupl*)malloc(sizeof(ListaDupl));
    ln->info = info;
    ln->prox = l;
    ln->ant = NULL;
    if(l!=NULL)
       l->ant = ln;
    return ln;
}
```

TAD Lista Dupl. Encadeada
Função Busca

```
ListaDupl* lst_dupl_busca(ListaDupl *l, int info){
    ListaDupl* lAux = l;
    while(lAux!=NULL){
        if(lAux->info == info)
            return lAux;
        lAux = lAux->prox;
    }
    return NULL;
}
```

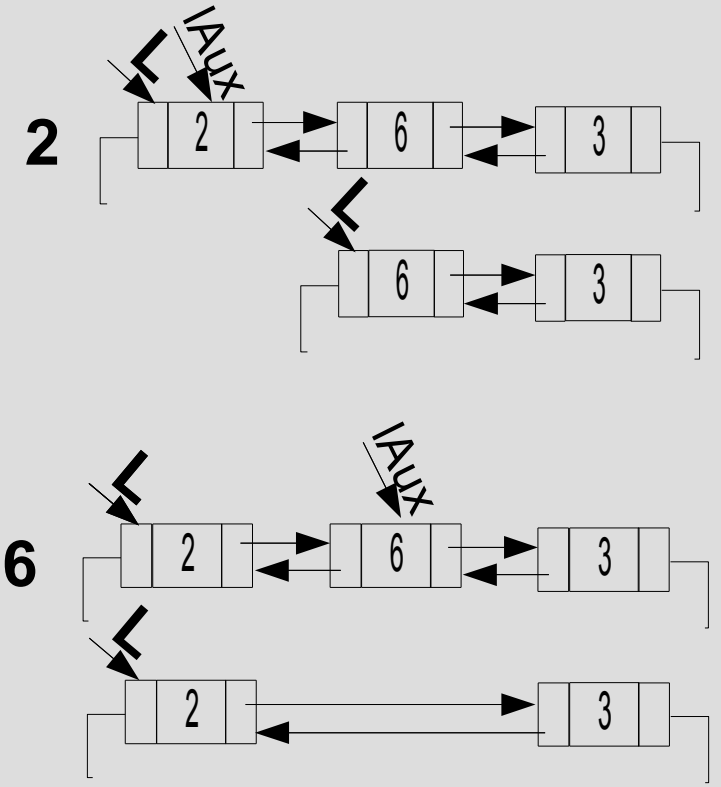


TAD Lista Dupl. Encadeada
Função Remove

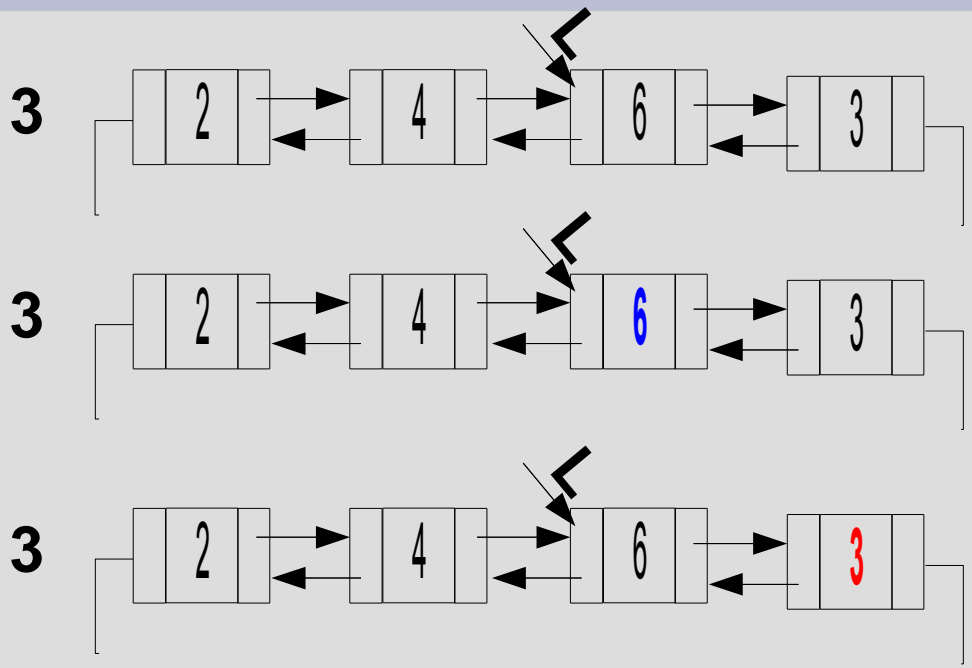


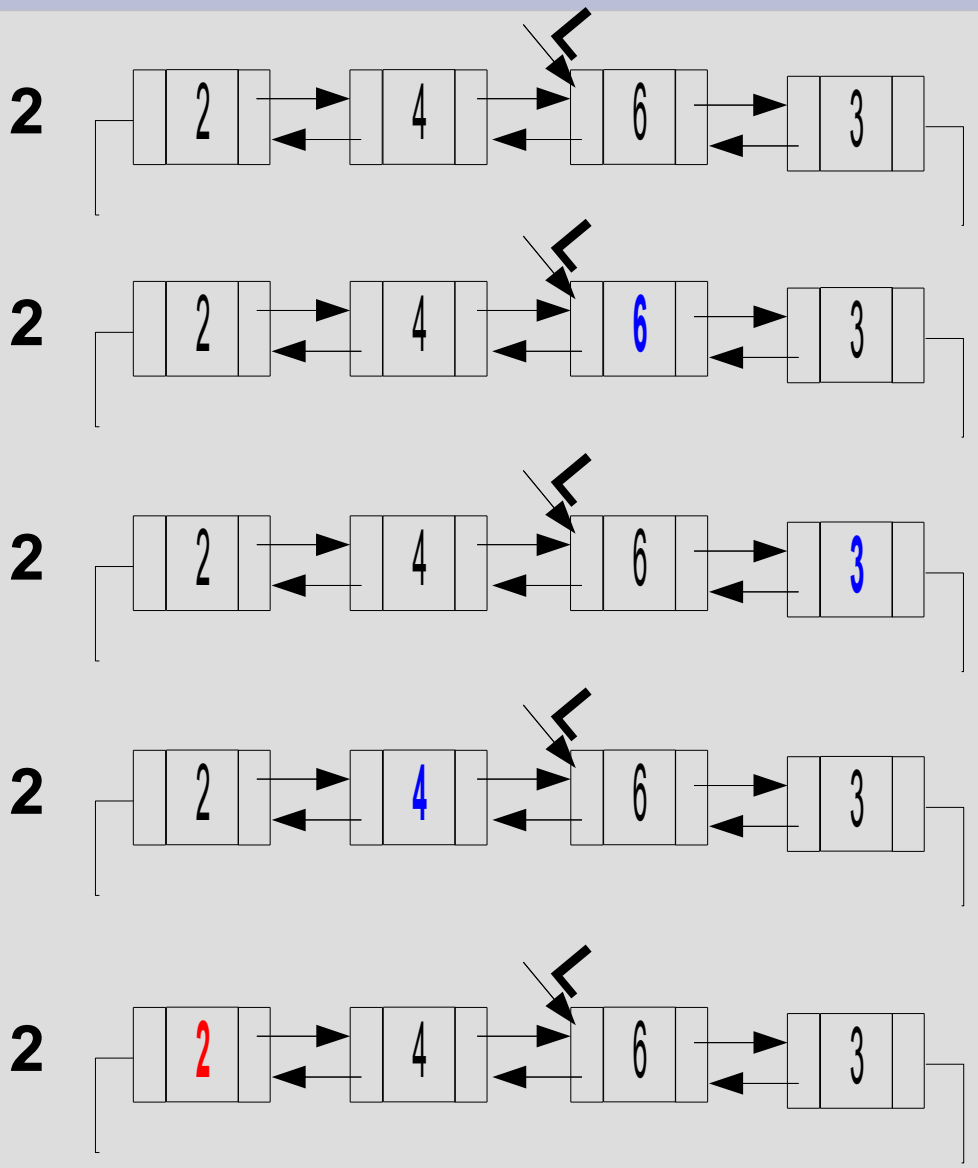
TAD Lista Dupl. Encadeada
Função Remove

```
ListaDupl* lst_dupl_remove(ListaDupl *l, int info){
    ListaDupl* lAux = busca(l,info);
    // Não achou o elemento
    if(lAux==NULL)
        return l;
    //Se é o primeiro
    if(lAux==l)
        l = lAux->prox;
    else
        lAux->ant->prox = lAux->prox;
    //Se o próximo não é vazio
    if(lAux->prox!=NULL)
        lAux->prox->ant = lAux->ant;

    free(lAux);

    return l;
}
```



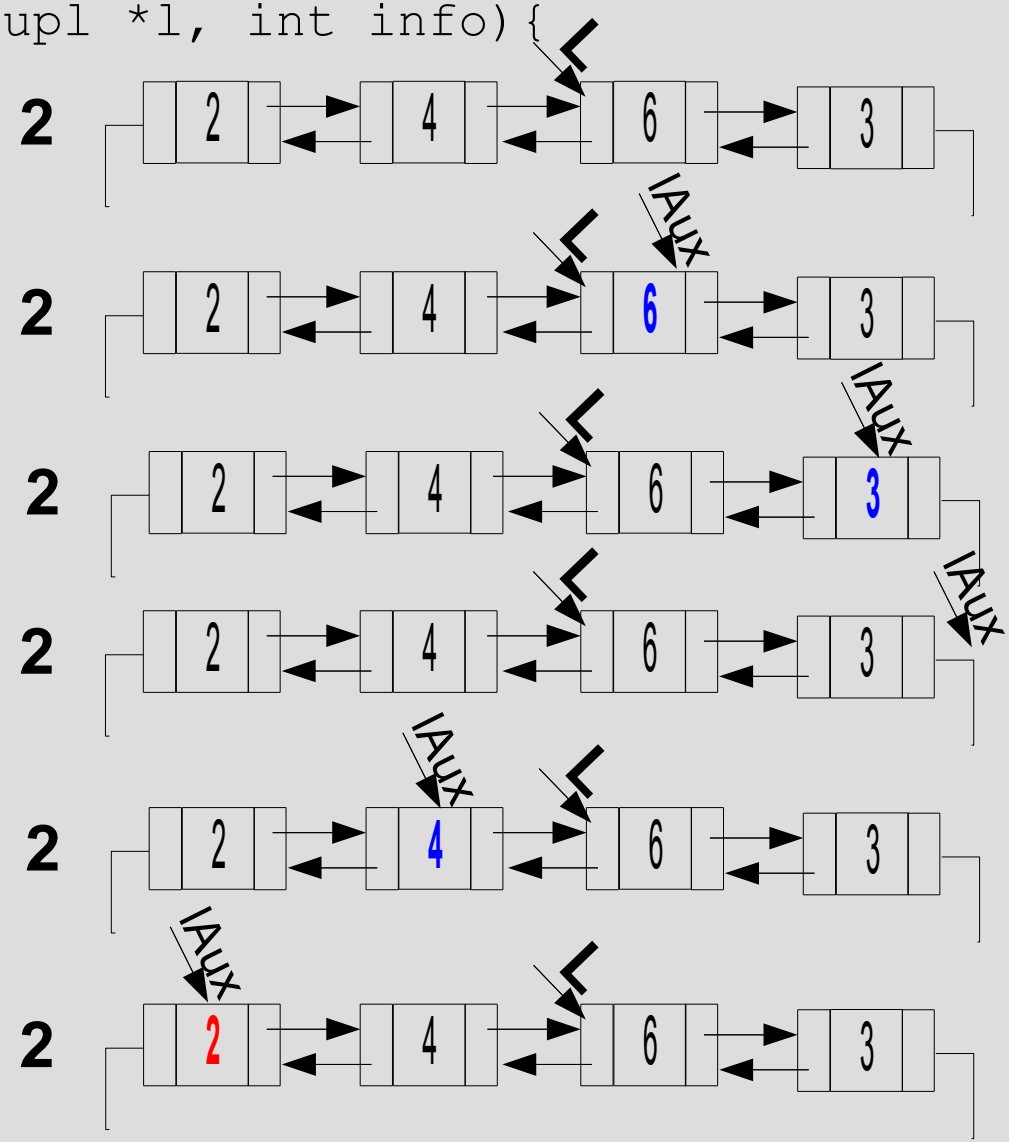
TAD Lista Dupl. Encadeada
Busca nas Duas Direções





TAD Lista Dupl. Encadeada
Função Busca Duas Direções

```
ListaDupl* lst_dupl_busca2(ListaDupl *l, int info){
    ListaDupl* lAux = l;
    if(l==null)
       return l;
    while(lAux!=NULL){
       if(lAux->info == info)
          return lAux;
       lAux = lAux->prox;
    }
    lAux = l->ant;
    while(lAux!=NULL){
       if(lAux->info == info)
          return lAux;
       lAux = lAux->ant;
    }
    return NULL;
}
```



TAD Lista Dupl. Encadeada
Função Remove