

INSTITUTO FEDERAL DE  
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

SUL DE MINAS GERAIS  
Campus Muzambinho

# Algoritmo e Estruturas de Dados II

*Lista Linear Duplamente Encadeada por  
Alocação Dinâmica de Memória*

IFSULDEMINAS, *campus* Muzambinho

Curso de Bacharelado em Ciência da Computação

AED II – Algoritmo e Estruturas de Dados II

# Lista Duplamente Encadeada

- É uma variação de lista simplesmente encadeada;
- Permite percorrer a lista nos dois sentidos: “*do início para o final*” e “*do final para o início*”

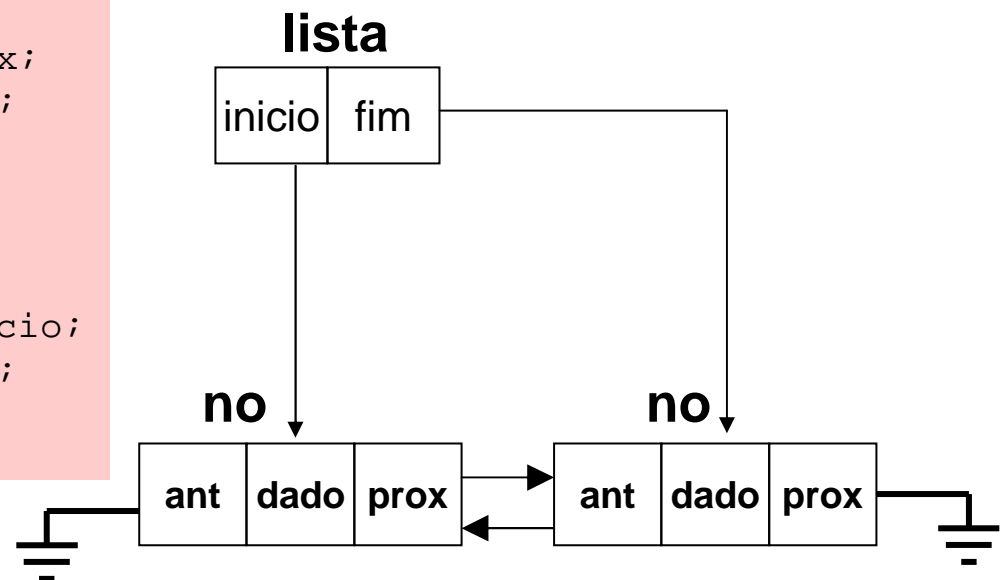


# TAD - Lista Duplamente Encadeada

```
#define TRUE 1
#define FALSE 0

struct no
{
    int dado;
    struct no *prox;
    struct no *ant;
};

typedef struct
{
    struct no *inicio;
    struct no *fim;
} listaDE;
```



```
void create(listaDE *q);
int isEmpty(listaDE q);
int insert(listaDE *q, int d);
int remover(listaDE *q, int d);
void imprime(listaDE q, char modo);
```

# Lista Duplamente Encadeada

```
main( )
{
    listaDE L;

    create(&L);


    insert(&L,12);
    insert(&L,320);
    insert(&L,413);
    insert(&L,197);
    insert(&L,26);
    imprime(L,'i');
    remover(&L,320);
    imprime(L,'f');
}
```

```
void create(listaDE *q)
{
    q->inicio=NULL;
}

int isEmpty(listaDE q)
{
    if (q.inicio==NULL)
        return (TRUE);
    else
        return (FALSE);
}
```



# Lista Duplamente Encadeada



```
main( )
{
    listaDE L;

    create(&L);

    insert(&L,12);
    insert(&L,320);
    insert(&L,413);
    insert(&L,197);
    insert(&L,26);
    imprime(L,'i');
    remover(&L,320);
    imprime(L,'f');
}
```

```
int insert(listaDE *q, int d)
{
    struct no *aux, *ant;
    if(q->inicio == NULL)
    {
        aux=(struct no*)malloc(sizeof(struct no));
        aux->dado = d;  q->inicio = aux;
        q->fim = aux;  aux->prox = NULL;
        aux->ant = NULL;  return(TRUE);
    }
    aux= (struct no*) malloc(sizeof(struct no));
    aux->dado = d;
    aux->prox = NULL;
    aux->ant = q->fim;
    q->fim->prox = aux;
    q->fim = aux;
    return(TRUE);
}
```

# Lista Duplamente Encadeada


```
main( )
{
    listaDE L;

    create(&L);

    insert(&L,12);
    insert(&L,320);
    insert(&L,413);
    insert(&L,197);
    insert(&L,26);
    imprime(L,'i');
    remover(&L,320);
    imprime(L,'f');
}
```

```
void imprime(listaDE q, char modo)
{
    struct no *aux;
    if (!isEmpty(q))
    {
        if (modo == 'i')
        {
            aux = q.inicio;
            while (aux != NULL)
            {
                printf("%d ", aux->dado);
                aux = aux->prox;
            }
        }
        else
        {
            aux = q.fim;
            while (aux != NULL)
            {
                printf("%d ", aux->dado);
                aux = aux->ant;
            }
        }
    }
}
```

## Lista Duplamente Encadeada

```
int remover(listaDE *q, int d)
{
    struct no *aux;
    struct no *atual;
    struct no *anterior;
    if(d == (q->inicio)->dado)
    {
        
    }
    else
    {
        anterior = q->inicio;
        atual = q->inicio->prox;
        while (atual != NULL && atual->dado != d)
        {
            anterior = atual; atual = atual->prox;
        }
        if (atual != NULL)
        {
            aux = atual; anterior->prox = atual->prox;
            atual->prox->ant = atual->ant;
            if (atual->prox == NULL)
            {
                anterior->prox = NULL; q->fim = anterior;
            }
            free(aux); return(TRUE);
        }
    }
    return(FALSE);
}
```

```
aux = q->inicio;
if (q->inicio->prox == NULL)
{
    q->inicio = NULL;
    q->fim = NULL;
}
else
{
    q->inicio = aux->prox;
    q->inicio->ant = aux->ant;
}
free(aux); return(TRUE);
```