

DevTitans

Listas Encadeadas

Lista Encadeada – Outros

```
typedef struct tNo{
    int dado;
    struct tNo *prox;
} tNo;

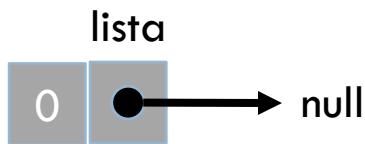
typedef struct tLista{
    int tam;
    tNo *prim;
} tLista;

void criaLista(tLista *l) {
    l->tam = 0;
    l->prim = NULL;
}

int listaVazia(tLista l) {
    return l.prim == NULL;
}
```

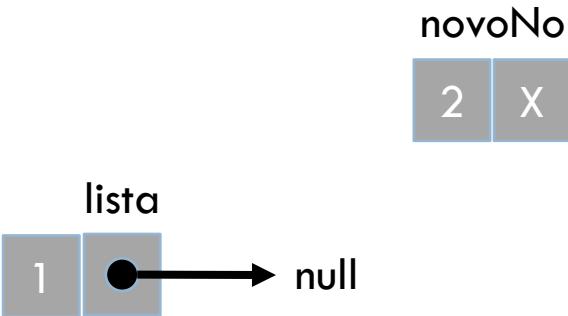
Inserção

Lista Encadeada – Inserção no Início



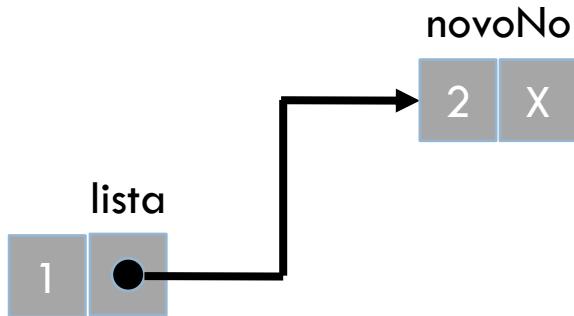
```
void insereInicio(tipoLista *lista, int novoDado) {
    tipoNo *novoNo = (tipoNo*) malloc(sizeof(tipoNo));
    novoNo->dado = novoDado;
    novoNo->prox = NULL;
    novoNo->prox = lista->prim;
    lista->prim = novoNo;
    lista->tamanho = lista->tamanho + 1;
}
```

Lista Encadeada – Inserção no Início



```
void insereInicio(tipoLista *lista, int novoDado){  
    tipoNo *novoNo = (tipoNo*) malloc(sizeof(tipoNo));  
    novoNo-> dado = novoDado;  
    novoNo-> prox = NULL;  
    novoNo-> prox = lista-> prim;  
    lista-> prim = novoNo;  
    lista-> tamanho = lista-> tamanho + 1;  
}
```

Lista Encadeada – Inserção no Início



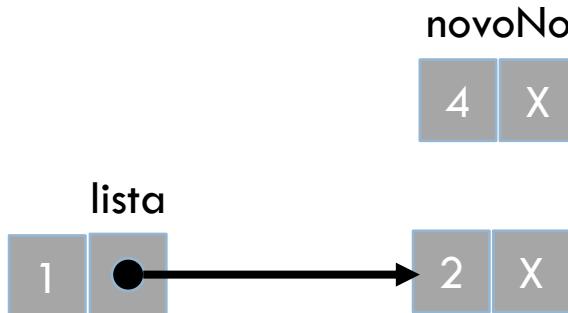
```
void insereInicio(tipoLista *lista, int novoDado){  
    tipoNo *novoNo = (tipoNo*) malloc(sizeof(tipoNo));  
    novoNo-> dado = novoDado;  
    novoNo-> prox = NULL;  
    novoNo-> prox = lista-> prim;  
    lista-> prim = novoNo;  
    lista-> tamanho = lista-> tamanho + 1;  
}
```

Lista Encadeada – Inserção no Início



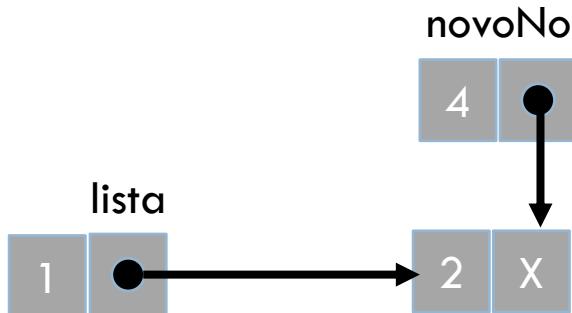
```
void insereInicio(tipoLista *lista, int novoDado){  
    tipoNo *novoNo = (tipoNo*) malloc(sizeof(tipoNo));  
    novoNo->dado = novoDado;  
    novoNo->prox = NULL;  
    novoNo->prox = lista->prim;  
    lista->prim = novoNo;  
    lista->tamanho = lista->tamanho + 1;  
}
```

Lista Encadeada – Inserção no Início



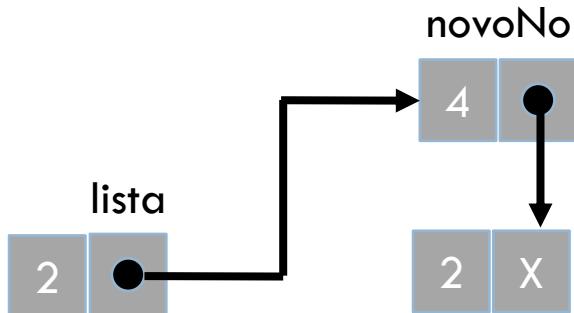
```
void insereInicio(tipoLista *lista, int novoDado){  
    tipoNo *novoNo = (tipoNo*) malloc(sizeof(tipoNo));  
    novoNo->dado = novoDado;  
    novoNo->prox = NULL;  
    novoNo->prox = lista->prim;  
    lista->prim = novoNo;  
    lista->tamanho = lista->tamanho + 1;  
}
```

Lista Encadeada – Inserção no Início



```
void insereInicio(tipoLista *lista, int novoDado){  
    tipoNo *novoNo = (tipoNo*) malloc(sizeof(tipoNo));  
    novoNo-> dado = novoDado;  
    novoNo-> prox = NULL;  
    novoNo-> prox = lista-> prim;  
    lista-> prim = novoNo;  
    lista-> tamanho = lista-> tamanho + 1;  
}
```

Lista Encadeada – Inserção no Início



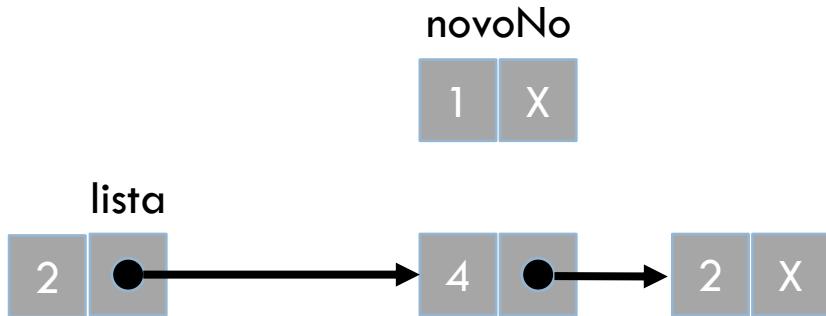
```
void insereInicio(tipoLista *lista, int novoDado){  
    tipoNo *novoNo = (tipoNo*) malloc(sizeof(tipoNo));  
    novoNo-> dado = novoDado;  
    novoNo-> prox = NULL;  
    novoNo-> prox = lista-> prim;  
    lista-> prim = novoNo;  
    lista-> tamanho = lista-> tamanho + 1;  
}
```

Lista Encadeada – Inserção no Início



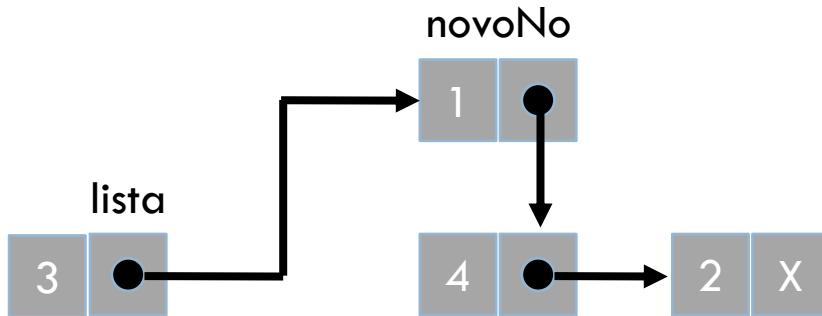
```
void insereInicio(tipoLista *lista, int novoDado){  
    tipoNo *novoNo = (tipoNo*) malloc(sizeof(tipoNo));  
    novoNo->dado = novoDado;  
    novoNo->prox = NULL;  
    novoNo->prox = lista->prim;  
    lista->prim = novoNo;  
    lista->tamanho = lista->tamanho + 1;  
}
```

Lista Encadeada – Inserção no Início



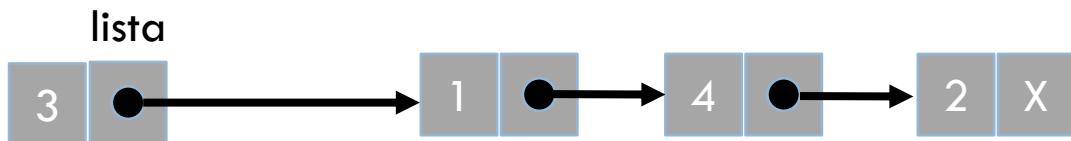
```
void insereInicio(tipoLista *lista, int novoDado){  
    tipoNo *novoNo = (tipoNo*) malloc(sizeof(tipoNo));  
    novoNo->dado = novoDado;  
    novoNo->prox = NULL;  
    novoNo->prox = lista->prim;  
    lista->prim = novoNo;  
    lista->tamanho = lista->tamanho + 1;  
}
```

Lista Encadeada – Inserção no Início



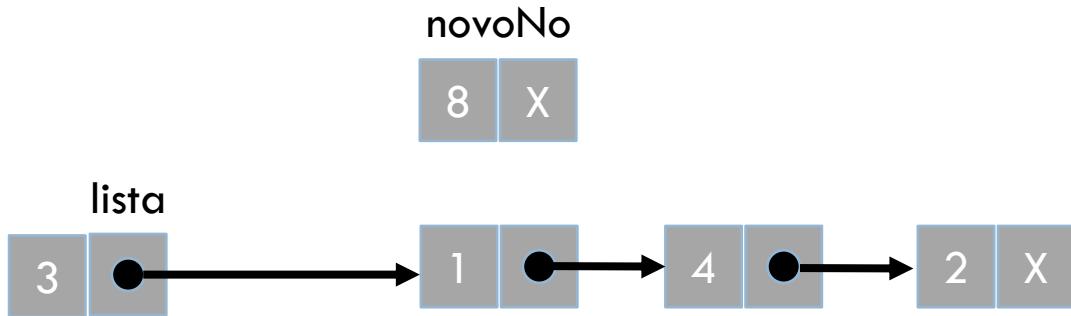
```
void insereInicio(tipoLista *lista, int novoDado){  
    tipoNo *novoNo = (tipoNo*) malloc(sizeof(tipoNo));  
    novoNo-> dado = novoDado;  
    novoNo-> prox = NULL;  
    novoNo-> prox = lista-> prim;  
    lista-> prim = novoNo;  
    lista-> tamanho = lista-> tamanho + 1;  
}
```

Lista Encadeada – Inserção no Início



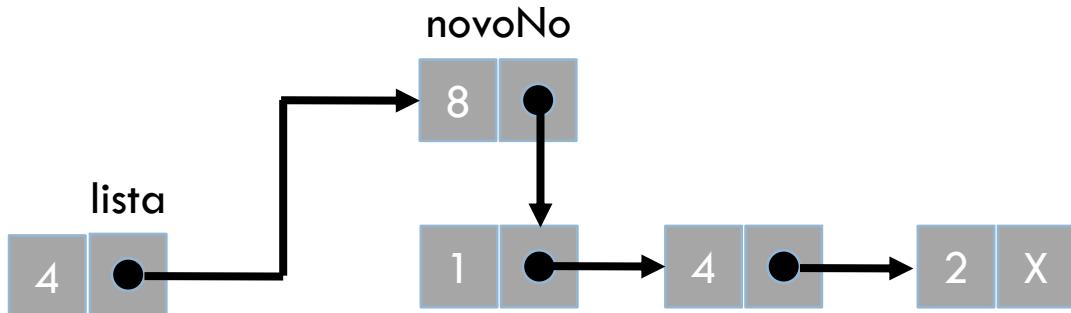
```
void insereInicio(tipoLista *lista, int novoDado){  
    tipoNo *novoNo = (tipoNo*) malloc(sizeof(tipoNo));  
    novoNo-> dado = novoDado;  
    novoNo-> prox = NULL;  
    novoNo-> prox = lista-> prim;  
    lista-> prim = novoNo;  
    lista-> tamanho = lista-> tamanho + 1;  
}
```

Lista Encadeada – Inserção no Início



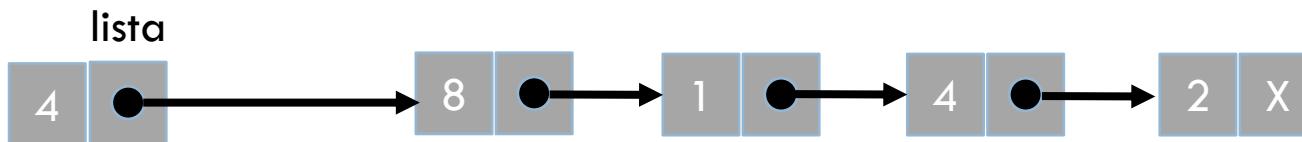
```
void insereInicio(tipoLista *lista, int novoDado){  
    tipoNo *novoNo = (tipoNo*) malloc(sizeof(tipoNo));  
    novoNo-> dado = novoDado;  
    novoNo-> prox = NULL;  
    novoNo-> prox = lista-> prim;  
    lista-> prim = novoNo;  
    lista-> tamanho = lista-> tamanho + 1;  
}
```

Lista Encadeada – Inserção no Início



```
void insereInicio(tipoLista *lista, int novoDado){  
    tipoNo *novoNo = (tipoNo*) malloc(sizeof(tipoNo));  
    novoNo->dado = novoDado;  
    novoNo->prox = NULL;  
    novoNo->prox = lista->prim;  
    lista->prim = novoNo;  
    lista->tamanho = lista->tamanho + 1;  
}
```

Lista Encadeada – Inserção no Início



```
void insereInicio(tipoLista *lista, int novoDado){  
    tipoNo *novoNo = (tipoNo*) malloc(sizeof(tipoNo));  
    novoNo-> dado = novoDado;  
    novoNo-> prox = NULL;  
    novoNo-> prox = lista-> prim;  
    lista-> prim = novoNo;  
    lista-> tamanho = lista-> tamanho + 1;  
}
```

Remoção

Lista Encadeada – Remoção

d = 8

noAt ● → null

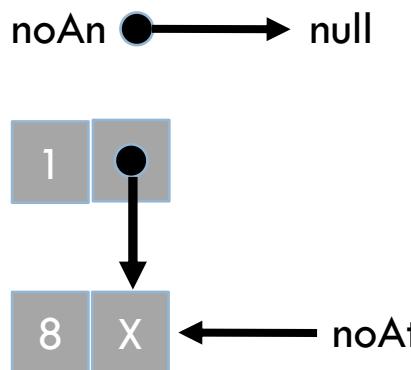
noAn ● → null

| 0 | ● → null

```
tLista *remNo(tLista *l, int d) {
    tNo *noAt = l->prim;
    tNo *noAn = NULL;
    while(noAt != NULL &&
          noAt->dado != d) {
        noAn = noAt;
        noAt = noAt->prox;
    }
    if(noAtual == NULL) return NULL;
    if(noAt == l->prim)
        l->prim = l->prim->prox;
    else
        noAn->prox = noAt->prox;
    noAt->prox = NULL;
    free(noAt);
    l->tam = l->tam - 1;
    return l;
}
```

Lista Encadeada – Remoção

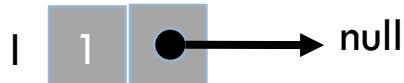
d = 8



```
tLista *remNo(tLista *l, int d) {
    tNo *noAt = l->prim;
    tNo *noAn = NULL;
    while(noAt != NULL &&
          noAt->dado== d) {
        noAn = noAt;
        noAt = noAt->prox;
    }
    if(noAt == NULL) return NULL;
    if(noAt == l->prim)
        l->prim = l->prim->prox;
    else
        noAn->prox = noAt->prox;
    noAt->prox = NULL;
    free(noAt);
    l->tam = l->tam - 1;
    return l;
}
```

Lista Encadeada – Remoção

d = 8



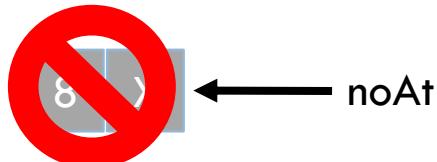
```
tLista *remNo(tLista *l, int d) {
    tNo *noAt = l->prim;
    tNo *noAn = NULL;
    while(noAt != NULL &&
          noAt->dado != d) {
        noAn = noAt;
        noAt = noAt->prox;
    }
    if(noAt == NULL) return NULL;
    if(noAt == l->prim)
        l->prim = l->prim->prox;
    else
        noAn->prox = noAt->prox;
    noAt->prox = NULL;
    free(noAt);
    l->tam = l->tam - 1;
    return 1;
}
```

Lista Encadeada – Remoção

d = 8

noAn ● → null

| 1 ● → null



```
tLista *remNo(tLista *l, int d) {
    tNo *noAt = l->prim;
    tNo *noAn = NULL;
    while(noAt != NULL &&
          noAt->dado != d) {
        noAn = noAt;
        noAt = noAt->prox;
    }
    if(noAt == NULL) return NULL;
    if(noAt == l->prim)
        l->prim = l->prim->prox;
    else
        noAn->prox = noAt->prox;
    noAt->prox = NULL;
    free(noAt);
    l->tam = l->tam - 1;
    return 1;
}
```

Lista Encadeada – Remoção

d = 8

noAn ● → null

| 0 ● → null

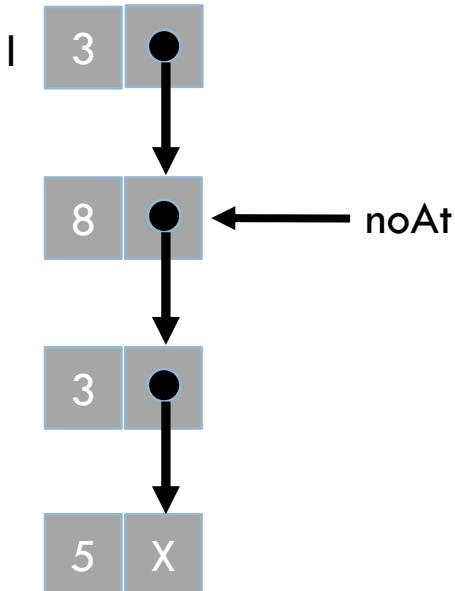
8  ← noAt

```
tLista *remNo(tLista *l, int d) {
    tNo *noAt = l->prim;
    tNo *noAn = NULL;
    while(noAt != NULL &&
          noAt->dado != d) {
        noAn = noAt;
        noAt = noAt->prox;
    }
    if(noAt == NULL) return NULL;
    if(noAt == l->prim)
        l->prim = l->prim->prox;
    else
        noAn->prox = noAt->prox;
    noAt->prox = NULL;
    free(noAt);
    l->tam = l->tam - 1;
    return 1;
}
```

Lista Encadeada – Remoção

d = 3

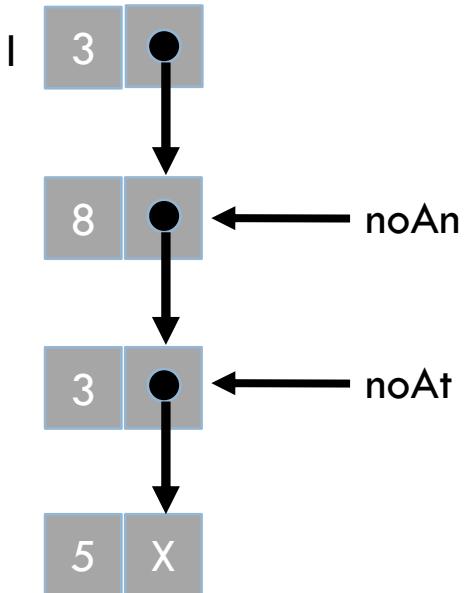
noAn → null



```
tLista *remNo(tLista *l, int d) {
    tNo *noAt = l->prim;
    tNo *noAn = NULL;
    while(noAt != NULL &&
          noAt->dado != d) {
        noAn = noAt;
        noAt = noAt->prox;
    }
    if(noAt == NULL) return NULL;
    if(noAt == l->prim)
        l->prim = l->prim->prox;
    else
        noAn->prox = noAt->prox;
    noAt->prox = NULL;
    free(noAt);
    l->tam = l->tam - 1;
    return l;
}
```

Lista Encadeada – Remoção

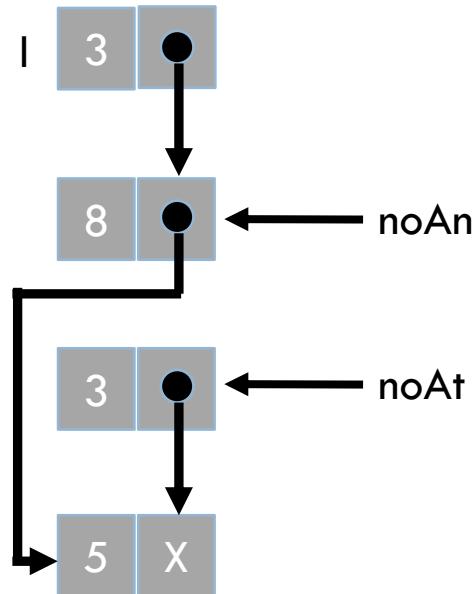
$d = 3$



```
tLista *remNo(tLista *l, int d) {
    tNo *noAt = l->prim;
    tNo *noAn = NULL;
    while(noAt != NULL &&
          noAt->dado != d) {
        noAn = noAt;
        noAt = noAt->prox;
    }
    if(noAt == NULL) return NULL;
    if(noAt == l->prim)
        l->prim = l->prim->prox;
    else
        noAn->prox = noAt->prox;
    free(noAt);
    l->tam = l->tam - 1;
    return l;
}
```

Lista Encadeada – Remoção

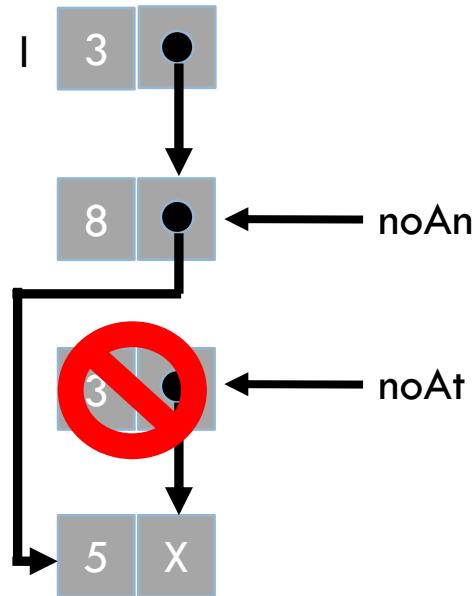
$d = 3$



```
tLista *remNo(tLista *l, int d) {
    tNo *noAt = l->prim;
    tNo *noAn = NULL;
    while(noAt != NULL &&
          noAt->dado != d) {
        noAn = noAt;
        noAt = noAt->prox;
    }
    if(noAt == NULL) return NULL;
    if(noAt == l->prim)
        l->prim = l->prim->prox;
    else
        noAn->prox = noAt->prox;
    free(noAt);
    l->tam = l->tam - 1;
    return 1;
}
```

Lista Encadeada – Remoção

$d = 3$

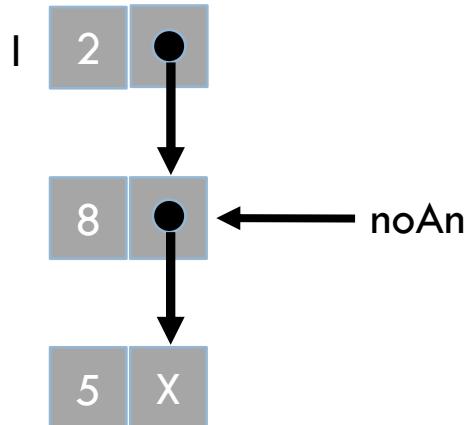


```
tLista *remNo(tLista *l, int d) {
    tNo *noAt = l->prim;
    tNo *noAn = NULL;
    while(noAt != NULL &&
          noAt->dado != d) {
        noAn = noAt;
        noAt = noAt->prox;
    }
    if(noAt == NULL) return NULL;
    if(noAt == l->prim)
        l->prim = l->prim->prox;
    else
        noAn->prox = noAt->prox;
    free(noAt);
    l->tam = l->tam - 1;
    return l;
}
```

Lista Encadeada – Remoção

$d = 3$

noAt  \rightarrow ???



```
tLista *remNo(tLista *l, int d) {
    tNo *noAt = l->prim;
    tNo *noAn = NULL;
    while(noAt != NULL &&
          noAt->dado != d) {
        noAn = noAt;
        noAt = noAt->prox;
    }
    if(noAt == NULL) return NULL;
    if(noAt == l->prim)
        l->prim = l->prim->prox;
    else
        noAn->prox = noAt->prox;
    free(noAt);
    l->tam = l->tam - 1;
    return 1;
}
```

Lista Encadeada – Busca

```
tNo *busca(tLista lista, int d){  
    tNo *noAtual = lista.prim;  
    while(noAtual != NULL){  
        if(noAtual->dado == d)  
            return noAtual;  
        noAtual = noAtual->prox;  
    }  
    return NULL;  
}
```

Lista Encadeada – Imprime

```
void imprLista(tLista l){
    printf("Lista: ");
    while(l.prim != NULL) {
        printf("%d\t", l.prim->dado);
        l.prim = l.prim->prox;
    }
    printf("\n");
}
```