

Tecnologia em Análise e Desenvolvimento de Sistemas - TADS

Estrutura de Dados

Prof. Luciano Vargas Gonçalves

E-mail: luciano.goncalves@riogrande.ifrs.edu.br



Sumário

Estrutura de Dados

- Listas Encadeadas
 - Simplesmente Encadeada – LSE
 - **Duplamente Encadeadas – LDE**

Sumário

- **Estrutura de Dados**
 - **Dinâmicas:**
 - O tamanho se altera com a necessidade;
 - Cresce ou Decresce
 - Listas
 - Simplesmente Encadeadas
 - **Duplamente Encadeadas**

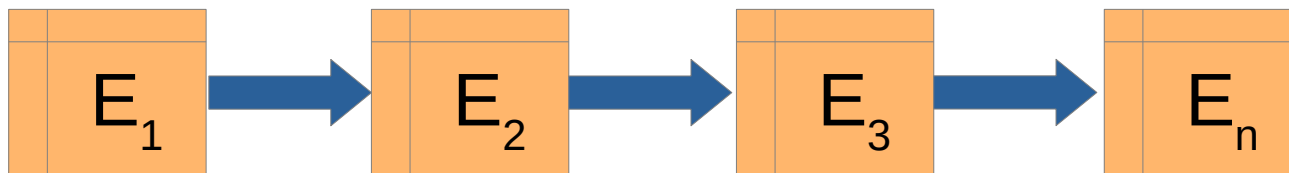
Listas

- **Listas**

- É uma coleção de elementos (Nó) do mesmo tipo, dispostos linearmente, que podem ou não seguir determinada organização, por exemplo:

- $[E_1, E_2, E_3, E_4, E_5, \dots, E_n]$

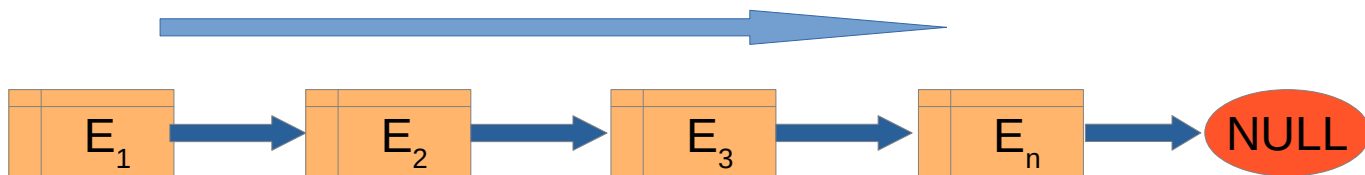
- Onde n seja $n \geq 0$;



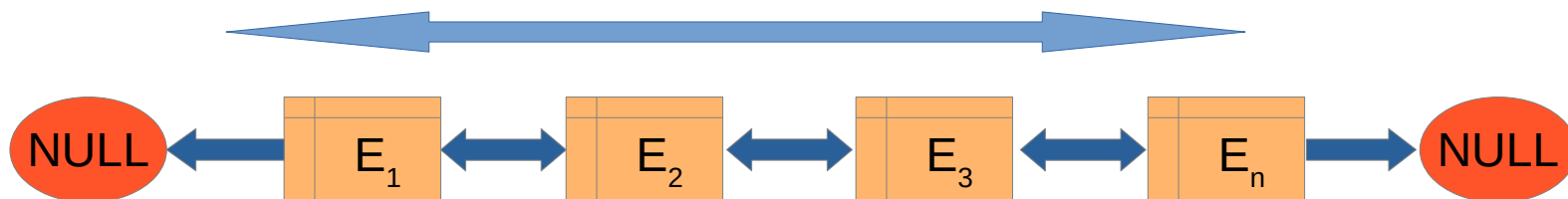
Listas

- **Listas Encadeadas:**

- Simples (Deslocamento (acesso) em um sentido)

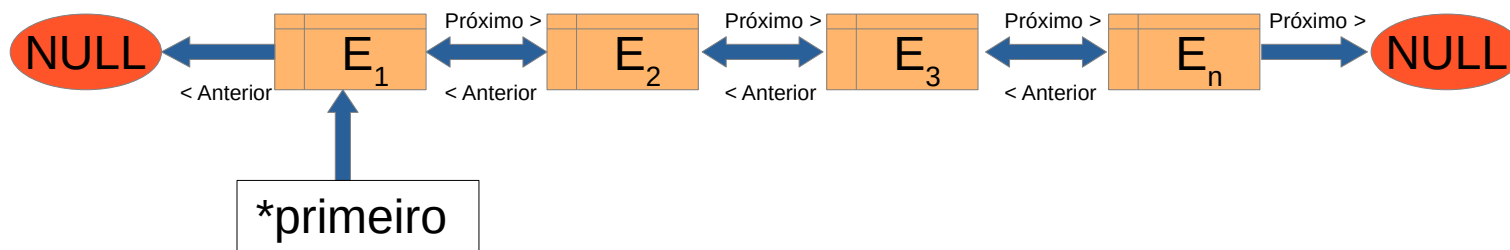


- Duplamente (Deslocamento em ambos sentidos)



Listas Duplamente Encadeada

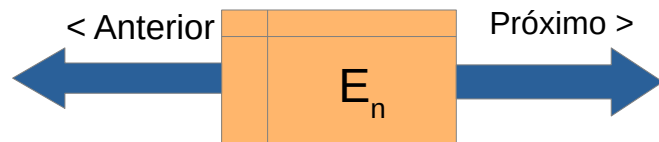
- Cada elemento (nó) armazenar uma referência para o **anterior** e o **próximo** elemento, sendo os elos de ligação com demais elementos;
- Navegação em duplo sentido
- Restrições:
 - Acesso pelo Extremo (Primeiro);
 - Inserção e remoção em qualquer parte da Lista



Lista Duplamente Encadeada

Listas Duplamente Encadeada

- O elemento (E_n) é uma estrutura de dados que irá armazenar as informações:
 - Informação do Elemento (dados);
 - Apontadores para os elementos **Anterior e Próximo** (Ponteiros *)

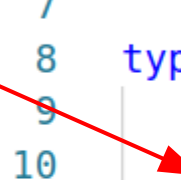


Elemento de armazenamento

Listas Duplamente Encadeada

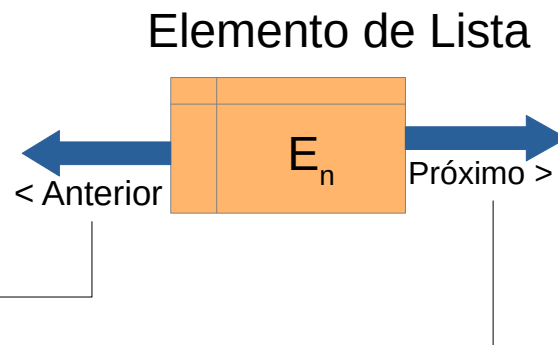
- Elemento de Lista:
 - Contará com os dados;
 - Ponteiros para **Anterior e Próximo**;
 - Exemplo:
 - Dados de uma lista de tarefas;
 - Descrição;
 - Prioridade;
 - Id.

```
3  typedef struct tarefa{
4      char descricao[30];
5      int id, prioridade;
6  }Tarefa;
7
8  typedef struct elemento{
9      Tarefa tf;
10     struct elemento *anterior;
11     struct elemento *proximo;
12 }Elemento;
13
14
15 typedef struct LDE {
16     Elemento *primeiro;
17     int n;
18 }LDE;
```



Listas Duplamente Encadeada

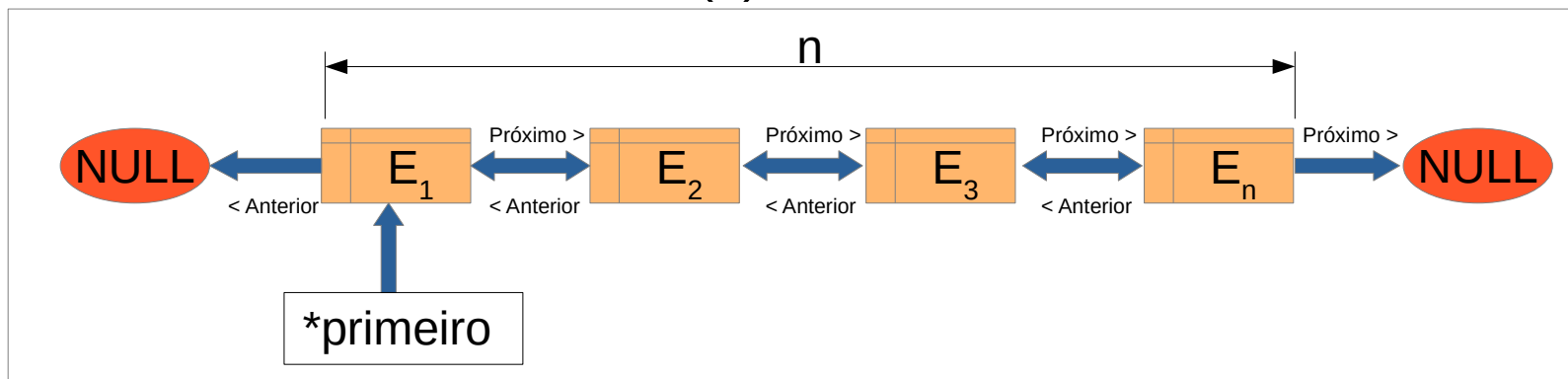
```
typedef struct elemento{  
    Tarefa tf;  
    struct elemento *anterior;  
    struct elemento *proximo;  
}Elemento;
```



Listas Duplamente Encadeada

- Estrutura da Lista:
 - Precisamos armazenar:
 - O apontador para “PRIMEIRO”;
 - Contador de Elementos (n);

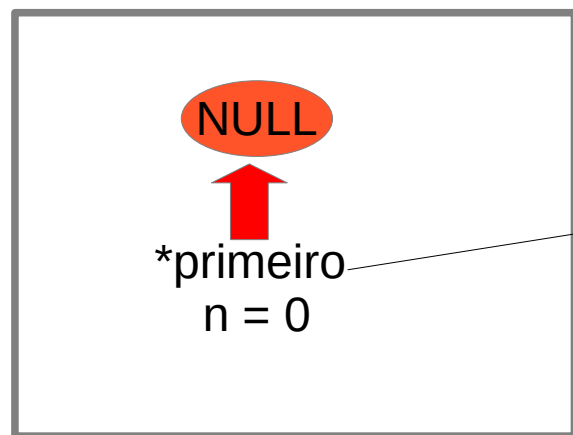
```
typedef struct LDE {  
    Elemento *primeiro;  
    int n;  
}LDE;
```



Lista Duplamente Encadeada

Listas Duplamente Encadeada

- **Criar uma Estrutura de Lista Vazia:**
 - Precisamos armazenar os apontador para “PRIMEIRO”;
 - Contador de Elementos (n);



Lista Vazia

Definir a Lista

```
LDE* criaListaLDE(){  
    LDE *nova = (LDE *)malloc(sizeof(LDE));  
    nova->primeiro = NULL;  
    nova->n = 0;  
    return nova;  
}
```

Criar e Inicializar a Lista Duplamente Encadeada

Listas Duplamente Encadeada

- Função para criar um elemento de lista do tipo nova tarefa

```
Elemento * criaElemento(int id){
    //aloca memória para um novo elemento, e preenche os campos
    Elemento *novo = (Elemento*)malloc(sizeof(Elemento));
    novo->tf.id = id;
    fflush(stdin);
    printf("Informe sua Prioridade:");
    scanf("%d",&novo->tf.prioridade);
    printf("Informe a Descricao:");
    scanf("%s",novo->tf.descricao);
    printf("Elemento Criada com Sucesso\n ");
    return novo;
}
```

Funções para controle da Lista

```
LDE* criaListaLDE();
Elemento* criaElemento(int id);

void insereNoInicio(LDE *l, Elemento *novo);
void insereNoFim(LDE *l, Elemento *novo);
int insereNaPosicao(LDE *l, Elemento *novo, int p);

Elemento * removeNoInicio(LDE *l);
Elemento * removeNoFim(LDE *l);
Elemento * removeNaPosicao(LDE *l, int p);

void mostraDados(Elemento d);

void mostraListaED(LDE *l); //mostra da Esquerda para Direita
void mostraListaDE(LDE *l); //mostra da Direita para Esquerda

void mostraElementoPosicao(LDE *l, int p);

void apagaLDE(LDE *l);

void menu(LDE *l);
```

Função Insere no Início

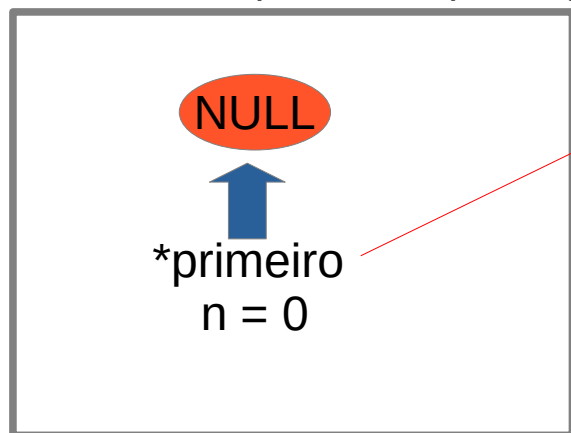
- **Método para inserir um elemento no *Início* da Lista**
 - Possibilidades:
 - Lista vazia
 - Lista com elementos
 - Protótipo da Função
 - Recebe um ponteiro para lista (*l) e um ponteiro para o novo elemento (*novo) ;
 - void insereNoInicio(LDE *l, Elemento *novo);

Função Insere no Início

- Método para inserir um novo elemento no **Início** da Lista

- Primeiro Passo: Lista Vazia

- *Ponteiro primeiro* aponta para NULL;



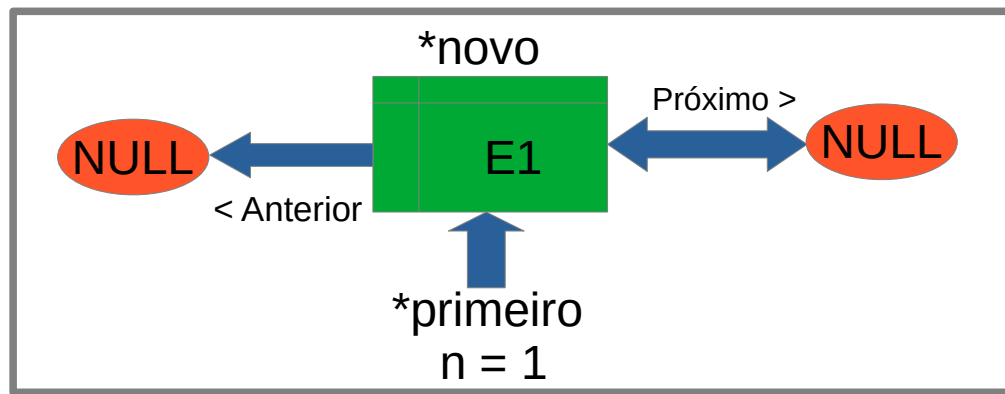
Lista Vazia

```
void insereNoInicio(LDE *l, Elemento *novo){  
    //insere um elemento no início da lista  
    printf("\nInsere no INICIO\n");  
    novo->anterior = NULL;  
    if(l->primeiro == NULL){  
        novo->proximo = NULL;  
    }else{  
        novo->proximo = l->primeiro;  
        l->primeiro->anterior = novo;  
    }  
    l->primeiro = novo;  
    l->n++;  
}
```

Insere no Início da Lista Vazia

Função Insere no Início

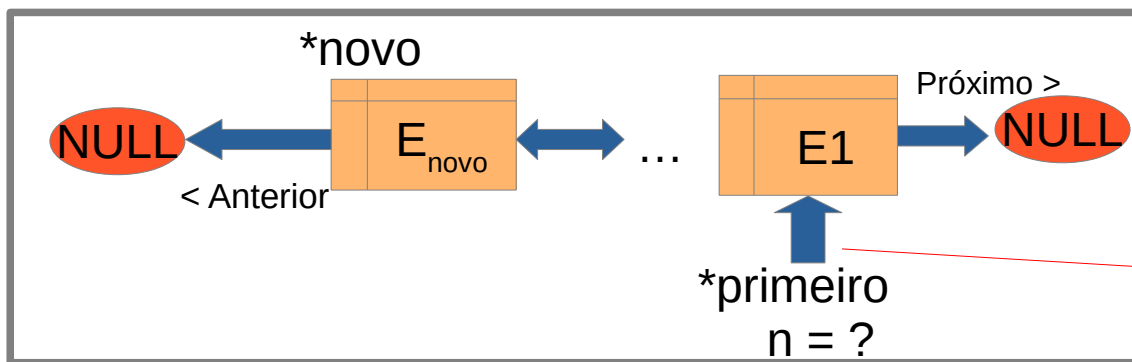
- Método para inserir um elemento no **Início** na Lista
 - Após a Inserção a lista ficará com a seguinte configuração



Lista com 1 elemento após inserção

Função Insere no Início

- Método para inserir um elemento no *Início* da Lista
 - Segundo Passo: lista com 1 ou mais elementos
 - Ponteiro para primeiro aponta para um elemento;

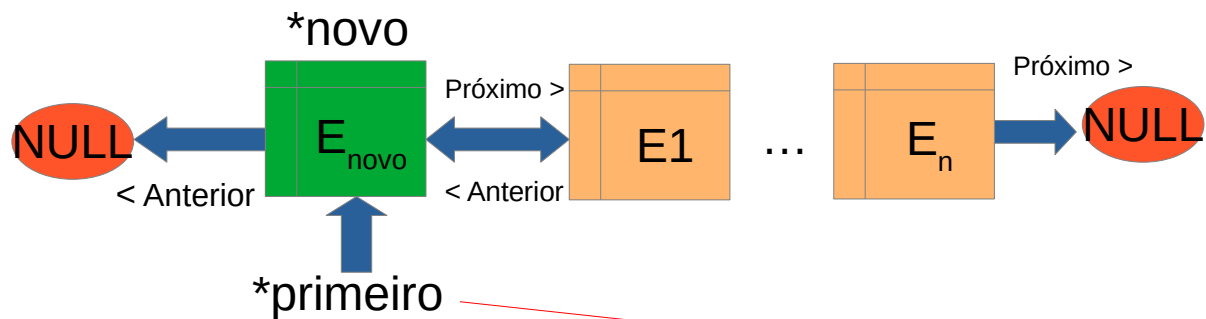


Lista com 1 ou mais elementos

```
void insereNoInicio(LDE *l, Elemento *novo){
    //insere um elemento no início da lista
    printf("\nInsere no INICIO\n");
    novo->anterior = NULL;
    if(l->primeiro == NULL){
        novo->proximo = NULL;
    }else{
        novo->proximo = l->primeiro;
        l->primeiro->anterior = novo;
    }
    l->primeiro = novo;
    l->n++;
}
```

Função Insere no Início

- Método para inserir um elemento no **Início** da Lista
 - Após a Inserção a lista ficará com a seguinte configuração



Lista com 1 ou mais elementos

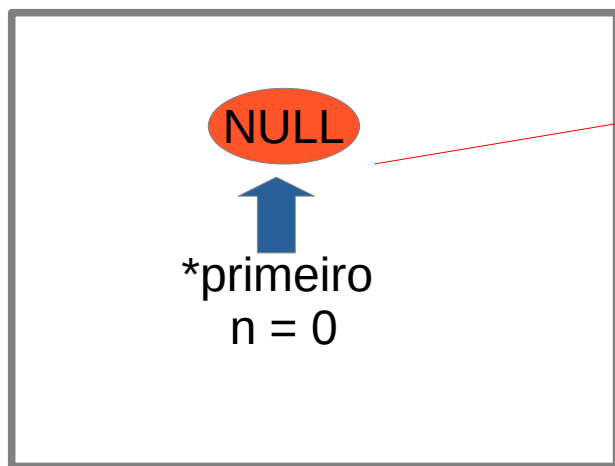
```
void insereNoInicio(LDE *l, Elemento *novo){  
    //insere um elemento no início da lista  
    printf("\nInsere no INICIO\n");  
    novo->anterior = NULL;  
    if(l->primeiro == NULL){  
        novo->proximo = NULL;  
    }else{  
        novo->proximo = l->primeiro;  
        l->primeiro->anterior = novo;  
    }  
    l->primeiro = novo;  
    l->n++;  
}
```

Função Insere no FIM

- Método para inserir um elemento no **Fim** da Lista

- Primeiro Passo: Lista Vazia

- Processo idêntico ao `insereNoInicio`;

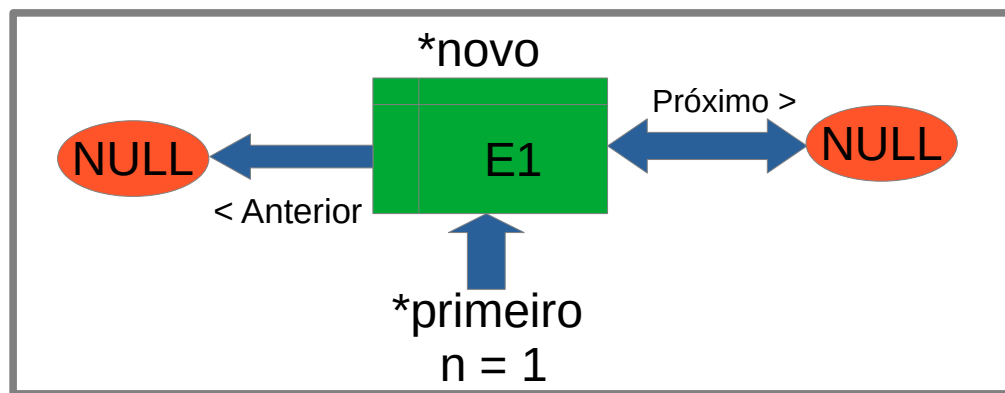


Lista estado atual

```
void insereNoFim(LDE *l, Elemento *novo){  
    //insere um elemento no Fim da lista  
    printf("\nInsere no FIM\n");  
    Elemento *aux = l->primeiro;  
    if(l->primeiro == NULL)  
        insereNoInicio(l,novo);  
    else{  
        while(aux->proximo != NULL){  
            aux = aux->proximo;  
        }  
        aux->proximo = novo;  
        novo->anterior = aux;  
        novo->proximo = NULL;  
        l->n++;  
    }  
}
```

Função Insere no FIM

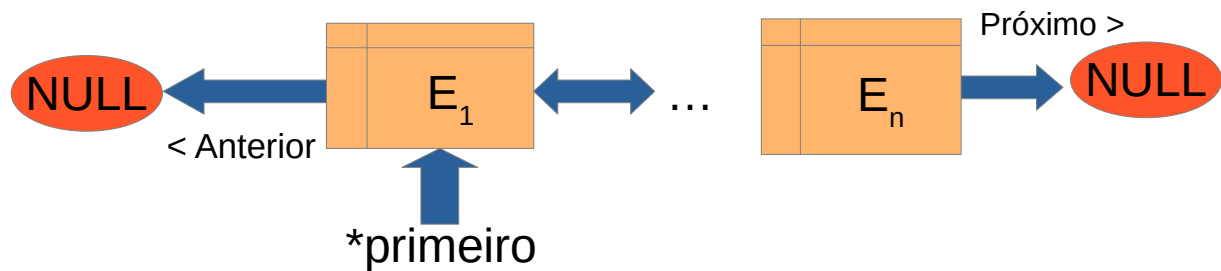
- Método para inserir um elemento no **Fim** na Lista
 - Após a Inserção a lista ficará com a seguinte configuração



Lista com 1 (uma) tarefa

Função Insere no FIM

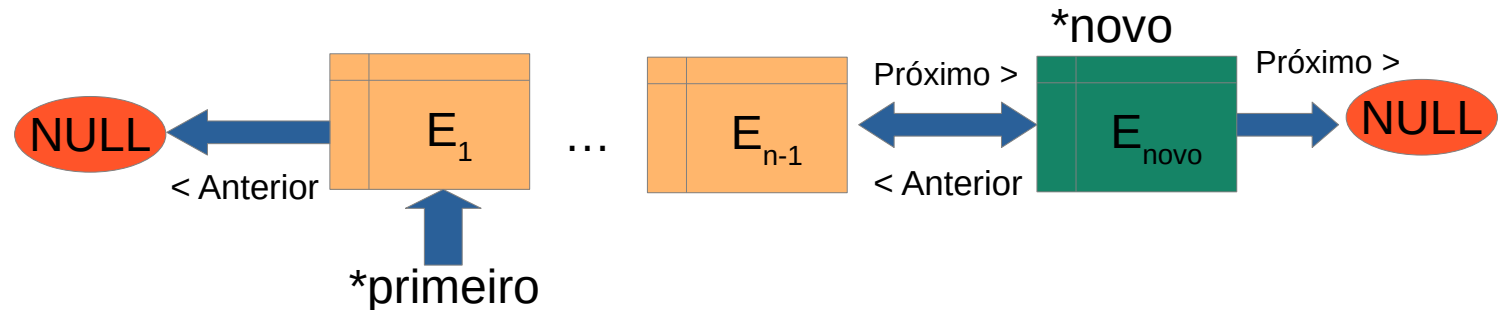
- Método para inserir um elemento no **Fim** da Lista
 - Segundo Passo: lista com 1 ou mais elementos
 - Ponteiro para primeiro aponta para um elemento;



Lista com 1 ou mais elementos

Função Insere no FIM

- Método para inserir um elemento no **Fim** da Lista
 - Processo de navegação para encontrar último elemento da lista



Lista com 1 ou mais elementos

Função Insere no FIM

- Método para inserir um elemento no **Fim** da Lista

- Busca pelo último elemento da lista

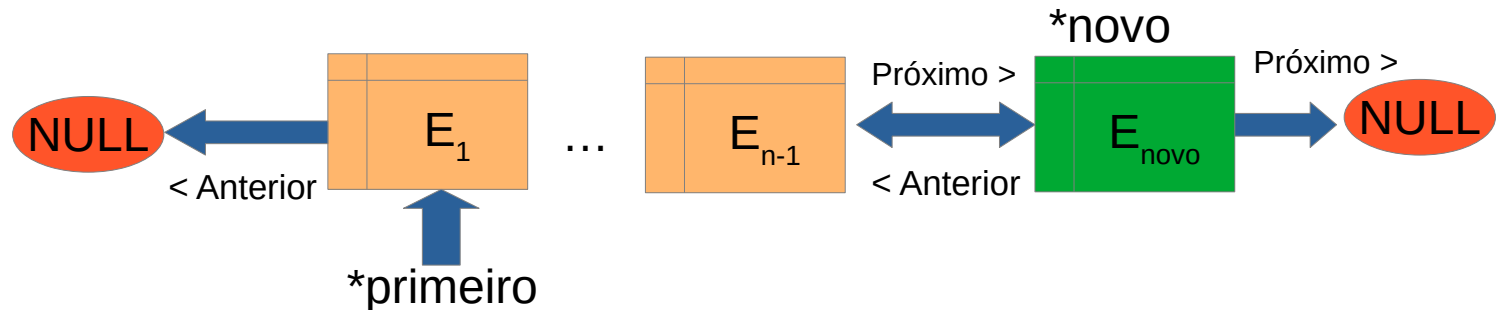
Alcançar o último elemento

Inserir o próximo elemento no fim

```
void insereNoFim(LDE *l, Elemento *novo){
    //insere um elemento no Fim da lista
    printf("\nInsere no FIM\n");
    Elemento *aux = l->primeiro;
    if(l->primeiro == NULL)
        insereNoInicio(l,novo);
    else{
        while(aux->proximo != NULL){
            aux = aux->proximo;
        }
        aux->proximo = novo;
        novo->anterior = aux;
        novo->proximo = NULL;
        l->n++;
    }
}
```

Função Insere no FIM

- Método para inserir um elemento no **Fim** da Lista
 - Processo de navegação para encontrar último elemento da lista



Lista com 1 ou mais elementos, sendo o último elemento inserido no Fim

Listas Duplamente Encadeada

- **A Lista duplamente encadeada terá as funções:**
 - Insere um elemento no início da lista;
 - Insere um elemento em uma posição definida;
 - Conta Elementos;
 - Mostra Lista;
 - Remove e retorna o primeiro elemento da lista;
 - Remove e retorna o último elemento da lista;
 - Remove e retorna o elemento na posição definida;
 - Mostra Lista da Esquerda para Direita;
 - Mostra Lista da Direita para Esquerda;

Listas Duplamente Encadeada

Protótipo das funções a serem implementadas:

```
int insereNaPosicao(LDE *l, Elemento *novo, int p);
```

```
Elemento * removeNoInicio(LDE *l);
```

```
Elemento * removeNoFim(LDE *l);
```

```
Elemento * removeNaPosicao(LDE *l, int p);
```

```
void mostraDados(Elemento d);
```

```
void mostraListaED(LDE *l); //mostra da Esquerda para Direita
```

```
void mostraListaDE(LDE *l); //mostra da Direita para Esquerda
```

```
void mostraElementoPosicao(LDE *l, int p);
```

```
void apagaLDE(LDE *l);
```

Programa principal (main)

```
int main(){  
    printf("\n Exemplo de Lista Duplamente Encadeada!!");  
  
    printf("\n Criando a LDE - Vazia ");  
  
    LDE *agenda = criaListaLDE();  
  
    if(!agenda)  
        exit(0);  
    else  
        printf("\n LDE Criada com Sucesso- Vazia\n");  
  
    menu(agenda); //para gerenciar a aplicacao.  
}
```

Função Menu

Chamada das Funções

```
void menu(LDE *l){  
    int op, posicao, id=1;  
    Tarefa *tf;  
    char ch;  
    do{  
        //system("clear");  
        printf("\n\nInforme uma Opção:");  
        printf("\n -- 1 - para Inserir Tarefa no Início:");  
        printf("\n -- 2 - para Inserir Tarefa no Fim:");  
        printf("\n -- 3 - para Inserir Tarefa na Posição:");  
        printf("\n -- 4 - para Remove Tarefa no Início:");  
        printf("\n -- 5 - para Remove Tarefa no Fim:");  
        printf("\n -- 6 - para Remove Tarefa na Posição:");  
        printf("\n -- 7 - para Mostra um Tarefa da Posição:");  
        printf("\n -- 8 - para Mostrar a Lista ED:");  
        printf("\n -- 9 - para Mostrar a Lista DE:");  
        printf("\n -- 10 - Apagar Lista:");  
        printf("\n -- 0 - para Sair do Programa:\n");  
        printf("\nInforme sua Opcao:");  
        scanf("%d",&op);  
        fflush(stdin);
```

Execução do programa

- Opções para Lista de tarefas

```
-----  
Informe uma Opção:  
-- 1 - para Inserir Elemento no Início:  
-- 2 - para Inserir Elemento no Fim:  
-- 3 - para Inserir Elemento na Posição:  
-- 4 - para Remove Elemento no Início:  
-- 5 - para Remove Elemento no Fim:  
-- 6 - para Remove Elemento na Posição:  
-- 7 - para Mostra um Elemento da Posição:  
-- 8 - para Mostrar a Lista ED:  
-- 9 - para Mostrar a Lista DE:  
-- 10 - Apagar Lista:  
-- 0 - para Sair do Programa:
```

Informe sua Opção:

Execução do programa

Posição: 1

Elemento: 1

Descrição: Aula

Prioridade: 1

Posição: 2

Elemento: 2

Descrição: Trabalhar

Prioridade: 1

Posição: 3

Elemento: 3

Descrição: Sair

Prioridade: 2

Posição: 4

Elemento: 4

Descrição: descansar

Prioridade: 4

Fim da Lista de Elementos 1-p/continuar



Exercício

- Implementar as funções que estão em Branco;
- Testar a criação de uma lista com várias tarefas agendadas;
 - Realizar várias inserções e remoções.
- Mostrar as atividades cadastradas.
 - Encerrar o programa;

- 
- Dúvidas pelo Fórum ou por e-mail!!!