

Tecnologia em Análise e Desenvolvimento de Sistemas – TADS

Estrutura de Dados

Prof. Luciano Vargas Gonçalves
E-mail: luciano.goncalves@riogrande.ifrs.edu.br



Estrutura da Dados

Aula 4 – Estruturas Dinâmicas

- Listas
- Pilhas
- Filas

Sumário

- **Estrutura de Dados**

- **Estáticas:**

- Tamanho Fixo
 - Vetores e Matrizes

- **Dinâmicas:**

- O tamanho se altera com a necessidade (Alocação Dinâmica);
 - Cresce ou Decresce
 - Listas
 - Simplesmente Encadeadas
 - Duplamente Encadeadas
 - Pilhas
 - Filas

Exemplos de Aplicações Dinâmicas

- **Exemplos**

- Agenda de Celular;
- Lista de Chamadas de alunos;
- Pessoas para o atendimento médico;
- Fila de Caixa (Banco);

- ***Sistemas dinâmicos***

- Os sistema cresce e decresce sob demanda
- Necessita alocação dinâmica de recursos
- Alocação e liberação de memória em tempo de execução.
 - Malloc e Free

Dados e Interface de dados

- **Duas partes distintas:**

- **DADOS:**

- Informação a ser armazenada (carros, clientes, livros);
 - Necessário uma estrutura para armazenar as informações propriamente dita.
 - Lista de contatos:
 - São os contatos (Nome, telefone, endereço);

- **INTERFACE:**

- Estrutura que gerencia os dados;
 - Controla as ações sobre os dados;
 - Insere, Remove, Consulta e Atualiza;
 - Ex: inserir, remover, editar contatos.



Interface do Usuário

Dados e Interface

- Vantagens da divisão:
 - Aproveitamento dos dados em diferentes aplicações;
 - Alterações na interface não alteram a estrutura de armazenamento;
 - A interface deve ser a mais genérica possível para aproveitamento em outras aplicações;

Estrutura da Dados

- **Listas**

Estrutura de Dados - Listas

- **LISTA :**

- É uma coleção de **elementos** (Nós) do mesmo tipo, dispostos linearmente, que podem ou não seguir determinada organização.
 - Por exemplo: lista de telefone, lista carros, lista filmes, etc;
- A lista dinâmica é composta por um número infinito de elementos (E0,E1,E2,...En);
 - Onde “n” é a quantidade de elementos;
- O fim da lista é marcado pela terminador (“NULL”);
- Os ponteiros (exemplo próximo) determinam o encadeamento dos elementos.



Lista de Elementos

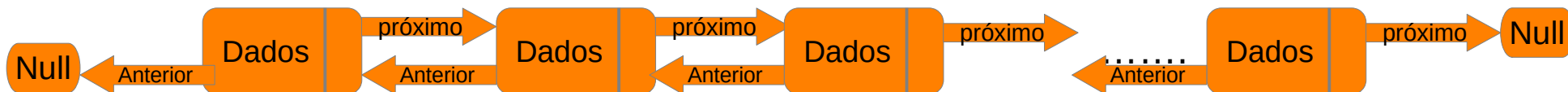
Estrutura de dados - Listas

- **Tipos de Listas Encadeadas:**

- **Simples** (Deslocamento em apenas *um sentido* – *Um ponteiro “Próximo”*)



- **Duplamente** (Deslocamento em *ambos sentidos* – *dois ponteiros “Anterior e Próximo”*)

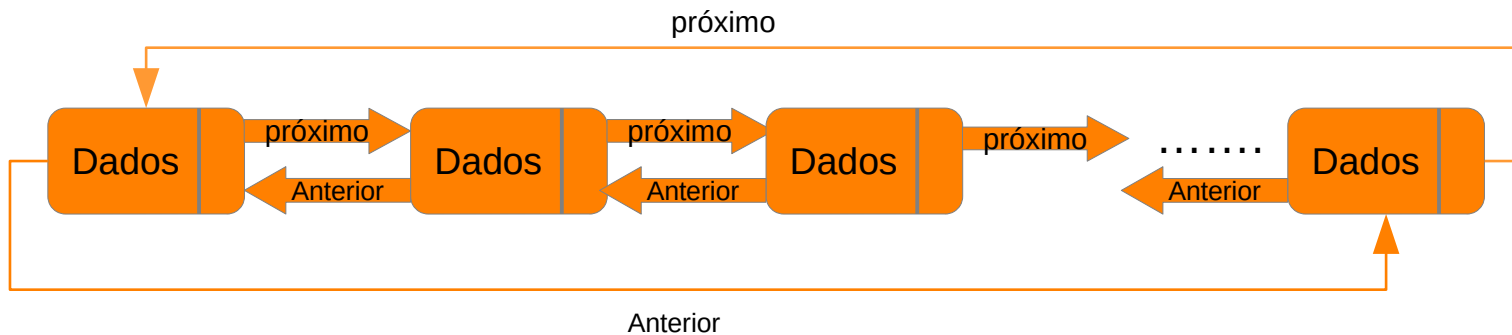


Estrutura de dados - Listas

- **Tipos de Listas Simplesmente Encadeadas:**

- **Circulares**

- **Ponteiros para Anterior e Próximo não apontam para NULL nos extremos, apontam para o primeiro e último.**
 - **Não possuem terminadores (NULL)**



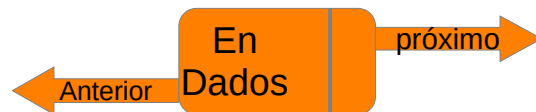
Estrutura de dados - Listas

- **Quando falamos de listas dinâmicas:**
 - Precisamos pensar nos **elementos da estrutura**:
 - Ex:
 - Lista de Chamada: os elementos são os alunos matriculados;
 - Lista de Carros: os elementos são os carros;
 - Precisamos pensar **na interface da lista**:
 - Ex:
 - Lista de Chamada: a estrutura é definida pelas marcações, que determinam **início e fim** da lista, e os pontos de inserção, consulta e remoção na lista.

Elemento de Lista - Nó

- **Elemento (Nó)**

- É a estrutura de dados que irá armazenar as informações do elemento da lista (dados) e os encadeamentos com o **Anterior** e/ou **Próximo** elemento (ponteiro);
 - A Lista de chamada armazena Alunos;
 - A Lista de Filmes armazena títulos de filmes;
 - A Lista de tarefas armazena as atividades pendentes;



Elemento de Lista

Elementos de Lista

- Exemplo: Lista de Chamada de Alunos
 - O elemento será um Aluno;
 - Onde iremos armazenar as informações de um Aluno:
 - Nome, idade, matricula;

Aluno
Nome Idade Matrícula

Elemento Aluno

Elementos de Lista

- Elementos na memória RAM;
 - Os elementos ficam dispersos na memória quando alocados dinamicamente;
 - Cada elemento é alocado no espaço disponível no momento da alocação;
 - Exemplo: Alunos
 - João, Maria e Paula

Endereço	Dados
xxx01	Aluno João
xxx02
xxx03
....	
xxx50
xxx51	Aluna Maria
.....	
.....	...
xxx81	Aluna Paula
xxx82	

Ex: Memória RAM

Estrutura da Dados

Formas de encadear os elementos alocados em memória:

- Listas**
 - Simplesmente Encadeadas - LSE**
 - Duplamente Encadeadas - LDE**
 - Pilhas**
 - Filas**
 - Árvores e Grafos**

Listas Simplesmente Encadeada - LSE

- **Lista Simplesmente encadeada – LSE**

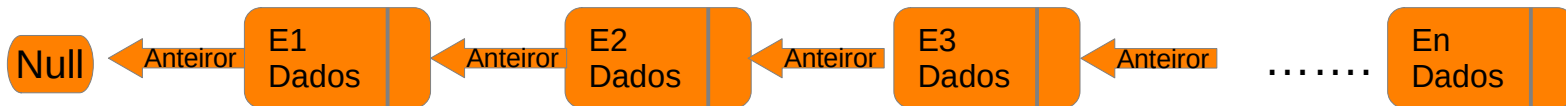
- Navegação em apenas **um sentido**;
- Apenas um ponteiro para o encadeamento (Anterior ou Próximo)

- Exemplo:

- Lista Crescente “*Próximo”

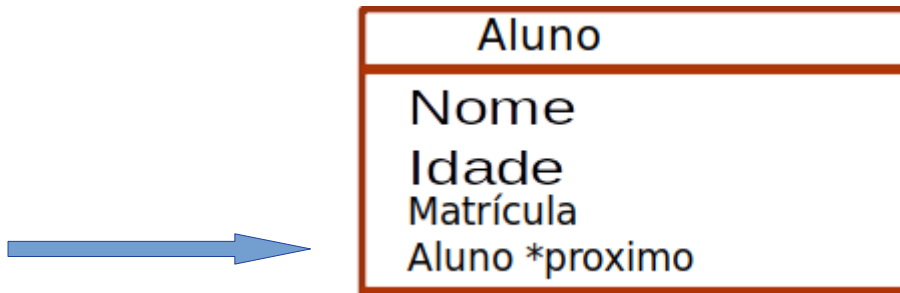


- Lista Decrescente “*Anterior”



Listas Simplesmente Encadeada - LSE

- **LSE - Ponteiro próximo:**
 - Para apontar para o próximo elemento iremos criar um apontador que irá armazenar uma referência para o “próximo” elemento (Aluno). Por exemplo, no caso, outro Aluno (*próximo).
 - Próximo é um ponteiro do tipo Aluno;

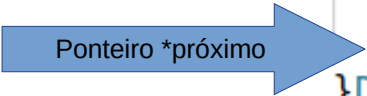


Listas Simplesmente Encadeada - LSE

- **Estrutura de Dados para armazenar Alunos**

- Elemento DadosAluno armazena os dados de um aluno e o ponteiro para o próximo DadosAluno (**proximo*);

```
typedef struct DadosAluno
{
    char nome[100];
    int idade;
    int matricula;
    struct DadosAluno *proximo;
}DadosAluno;
```

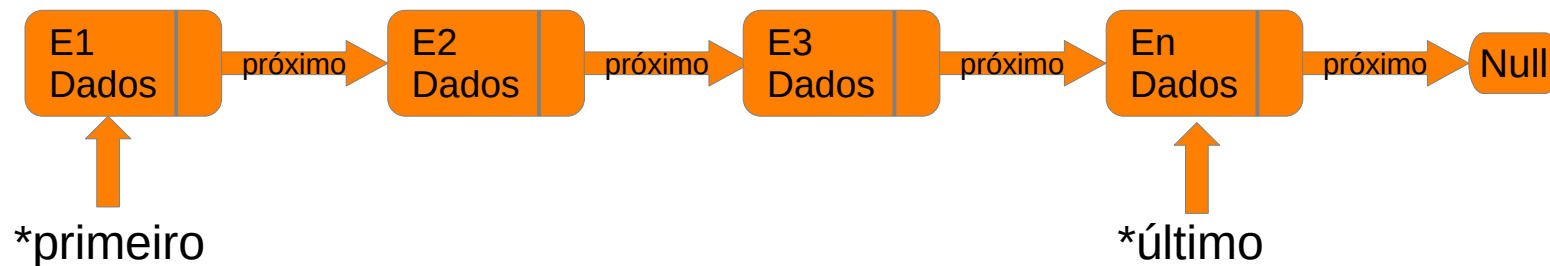


Exemplo de um Elemento DadosAluno

Listas Simplesmente Encadeada - LSE

- **Listas Simplesmente Encadeada (Crescente) – LSE**

- Lista pode ter de 0 ou mais elementos encadeados (n);
- *O acesso se dá pelas extremidades e a navegação ocorre do **Primeiro** elemento em direção **Último** elemento;*
- **Primeiro e Último também são ponteiros;**




Estrutura de Dados para LSE

- **Definição da Estrutura da LSE**

- Estruturas de dados (LSE) para gerenciar uma Lista Simplesmente Encadeada.

- **Exemplo 1:**

```
typedef struct lse{  
    DadosAluno *primeiro;  
    int n_elementos;  
}LSE;
```



Início da lista (ponteiro *primeiro Elemento da Lista)

Quantidade de elementos

- **Exemplo2:**

```
typedef struct lse{  
    DadosAluno *primeiro;  
    DadosAluno *ultimo;  
    int n_elementos;  
}LSE;
```

Ponteiro último é facultativo

Alocar memória para Lista(LSE) e para Elementos

- **Alocar memória para:**

- Uma nova lista LSE e retornar um ponteiro da Lista criada (*nova);

```
""Aloca memória para uma nova lista"";  
LSE *nova = (LSE*)malloc(sizeof(LSE));
```

- Um novo elemento de lista (DadosAluno) e retornar um ponteiro (*novoAluno);

```
""Aloca memória para um novo Elemento (aluno)"";  
DadosAluno *novoAluno = (DadosAluno*)malloc(sizeof(DadosAluno));
```

Interface da LSE

- A interface da lista implementa as funções, operações e as consultas sobre a lista LSE:
 - **Funções de Inserção:**
 - InsereNoInicio(), InsereNoFim(), InsereNaPosição()
 - **Funções de Remoção:**
 - RemoveNoInicio(), RemoveNoFim(), RemoveNaPosição();
 - **Funções de Consulta:**
 - MostraElemento(), MostraLista()
 - **Funções de Exclusão:**
 - ApagaElemento(), ApagaLista()
- ***OBS: Lista Simplesmente encadeada LSE, não possui restrição para inserção e remoção.***

Exemplo LSE – Chamada

- **Primeira função da Interface da Lista - CriaLista()**

- Aloca espaço para estrutura LSE e faz a inicialização da lista, atribui valores ao ponteiro *primeiro e a quantidade de elementos;
- Retorna um ponteiro de lista LSE (*nova);

```
LSE* criaListaLSE(){  
    LSE *nova = (LSE*)malloc(sizeof(LSE));  
    nova->primeiro = NULL;  
    nova->n_elementos = 0;  
    return nova;  
}
```

← Alocação de memória
para armazenar a lista;

← Retorna o endereço da estrutura lista LSE

Exemplo LSE – Chamada

- **Função cadastraAluno()**

- Recebe dados de um novo aluno (nome, idade, matricula)
- Aloca memória para um elemento DadosAluno e preenche os dados;
- Retorna um ponteiro DadosAluno (*novoAluno);

```
DadosAluno* cadastraAluno(char nome[20], int idade, int matricula){  
    DadosAluno *novoAluno = (DadosAluno*)malloc(sizeof(DadosAluno));  
    strcpy(novoAluno->nome, nome);  
    novoAluno->idade = idade;  
    novoAluno->matricula = matricula;  
    novoAluno->proximo = NULL;  
    return novoAluno;  
}
```



Reserva de Memória

Programa Principal – Main ()

- Main ()
 - Vamos criar um programa para gerenciar um lista ;
 - Exemplo: Lista de alunos da turma de matemática

```
//cria a lista de alunos de matemática
```

```
LSE *ListaMatematica = criaListaLSE();
```

```
//cria o elemento de LSE chamado pedro
```

```
DadosAluno *pedro = cadastraAluno("Pedro",44,1123301);
```

Funções de Inserção

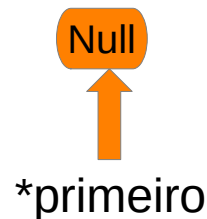
- Como Inserir um novo elemento na Lista, Funções:
 - **InserNoInicio():**
 - Insere novo elemento na posição E0;
 - **InserNoFim()**
 - Insere novo elemento na posição En;
 - **InserNaPosição(p)**
 - Insere novo elemento na posição Ep;
- **Condições a serem avaliadas:**
 - Lista vazia ($n_elementos = 0$);
 - Lista com elementos ($n_elementos > 0$);

Interface da LSE – Insere no Início

- Método para inserir um **novο** elemento no **Início** na Lista:
 - Duas situações precisam ser avaliadas;

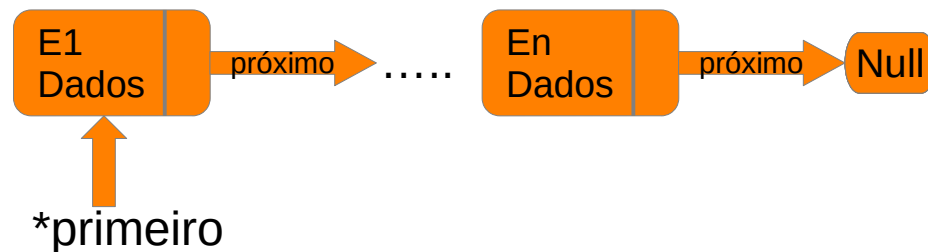
1) Lista vazia;

- Ponteiro ($*primeiro = NULL$);
- Número elementos ($n = 0$)



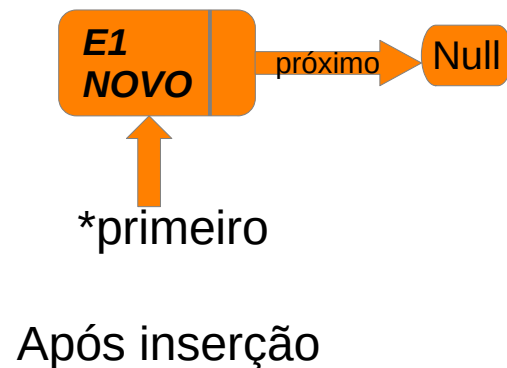
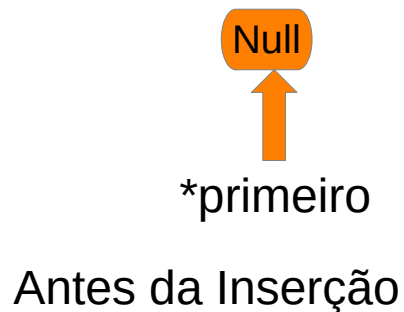
2) Lista com elementos;

- Ponteiro ($*primeiro \neq NULL$);
- Número de elementos ($n \neq 0$);



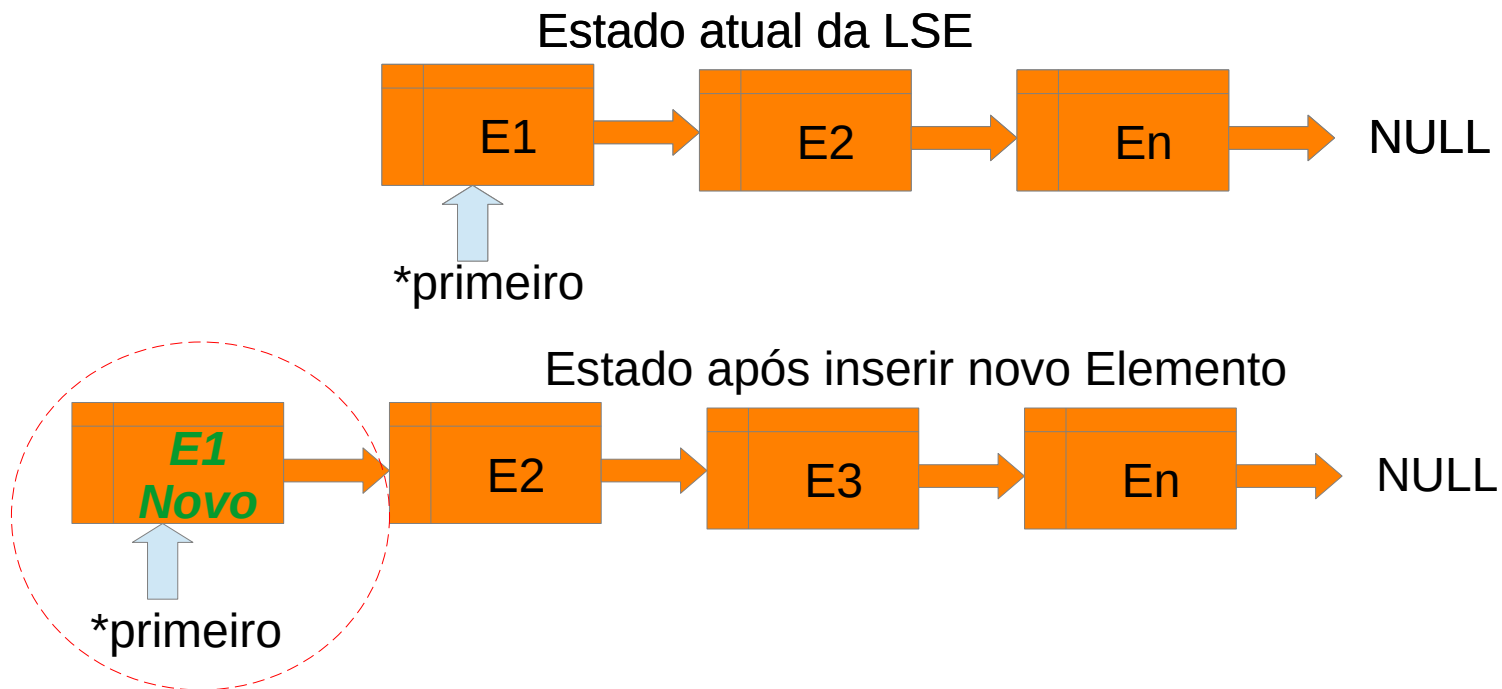
Interface da LSE – Insere no Início

- Método para inserir um **novο** elemento no **Início** na Lista
 - Estado Atual – Lista Vazia



Interface da LSE – Insere no Início

- Método para inserir um **novο** elemento no **Início** na Lista
 - Estado Atual – Lista COM ELEMENTOS



Interface da LSE – Insere no Início

- Método para inserir um **novο** elemento no **Início** na Lista:

- A função recebe um ponteiro de lista (*ls) e um ponteiro de elemento (*novo)
- Situações a serem avaliadas;

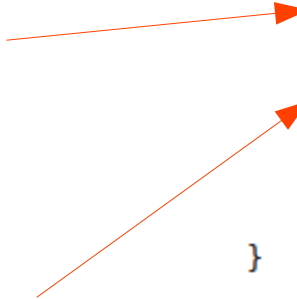
1) Lista vazia;

- Ponteiro (*primeiro = NULL);
- Número elementos (n = 0)

2) Lista com elementos;

- Ponteiro (*primeiro != NULL);
- Número de elementos (n != 0);

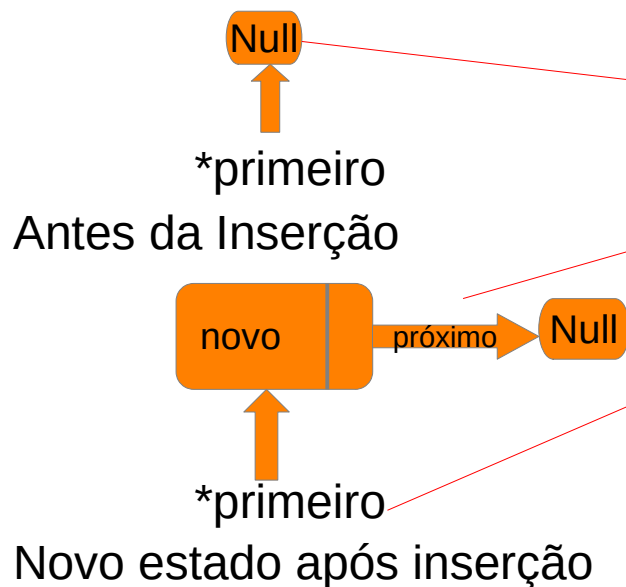
```
void insereNoInicio(LSE *ls, DadosAluno *novo){  
    ""Insero o Elemento no inicio da Lista"";  
    if(ls->primeiro == NULL) //Lista Vazia  
        novo->proximo = NULL;  
    else //Lista Com DadosAlunos  
        novo->proximo = ls->primeiro;  
    ls->primeiro = novo;  
    ls->n_elementos++;  
}
```



- A função recebe dois ponteiros, um para lista (*ls) e um para novo elemento (*novo);

Interface da LSE – Insere no Início

- Método para inserir um **novο** elemento no **Início** na Lista
 - Estado Atual – Lista Vazia

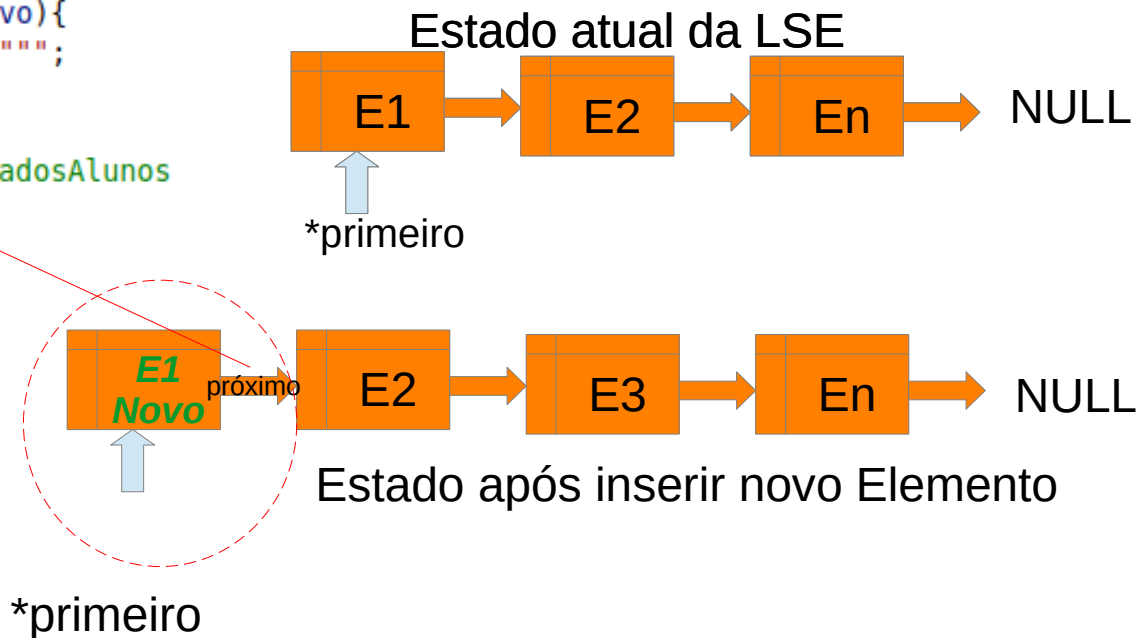


```
void insereNoInicio(LSE *ls, DadosAluno *novο){  
    ""Insero o Elemento no inicio da Lista"";  
    if(ls->primeiro == NULL) //Lista Vazia  
        novο->proximo = NULL;  
    else //Lista Com DadosAlunos  
        novο->proximo = ls->primeiro;  
    ls->primeiro = novο;  
    ls->n_elementos++;  
}
```

Interface da LSE – Insere no Início

- Método para inserir um **novο** elemento no **Início** na Lista
 - Estado Atual – Lista COM ELEMENTOS

```
void insereNoInicio(LSE *ls, DadosAluno *novo){  
    ""Insero o Elemento no inicio da Lista"";  
    if(ls->primeiro == NULL) //Lista Vazia  
        novo->proximo = NULL;  
    else //Lista Com DadosAlunos  
        novo->proximo = ls->primeiro;  
    ls->primeiro = novo;  
    ls->n_elementos++;  
}
```



Interface da LSE – Insere no Início

- Programa principal inserir novo elemento;

```
int main(){  
  
    //cria a lista de alunos de matemática  
    LSE *ListaMatematica = criaListaLSE();  
  
    //cria o elemento de LSE chamado pedro  
    DadosAluno *pedro = cadastraAluno("Pedro",44,1123301);  
  
    //chamada da funcao para cadastrar novo aluno  
    insereNoInicio(ListaMatematica,pedro);  
}
```

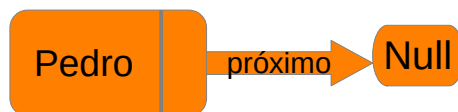
Novo aluno



Interface da LSE – Insere no Início

- Método para inserir um *novο* elemento no *Início* na Lista

Estado atual da LSE



*primeiro

```
insereNoInicio(ListaMatematica, cadastraAluno("Pedro", 44, 1123301));  
insereNoInicio(ListaMatematica, cadastraAluno("Paulo", 34, 1123302));  
insereNoInicio(ListaMatematica, cadastraAluno("Maria", 54, 1123303));  
insereNoInicio(ListaMatematica, cadastraAluno("Paula", 24, 1123304));
```

Estado atual da LSE – Após inserção



*primeiro

Interface da LSE – Mostra Lista

- **MostraLista()**

- Mostra todos elementos a partir do primeiro elemento;
- Necessário para varrer a lista (iterar);

```
void mostraLista(LSE *ls){  
    DadosAluno *aux = ls->primeiro;  
    while(aux != NULL){  
        mostraAluno(aux);  
        aux = aux->proximo;  
    }  
    printf("\nFim da Lista\n");  
}
```

Repetição

Do Primeiro

Até o último

Listas Simplesmente Encadeada

- MostraLista()

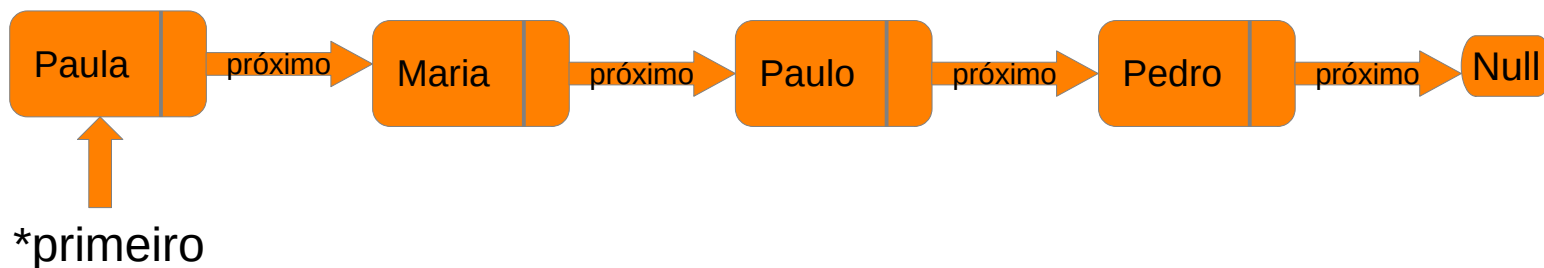
```
void mostraLista(LSE *ls){  
    DadosAluno *aux = ls->primeiro;  
    while(aux != NULL){  
        mostraAluno(aux);  
        aux = aux->proximo;  
    }  
    printf("\nFim da Lista\n");  
}
```

Início da Lista

Nome:Paula	Idade:24	Matricula:1123304
Nome:Maria	Idade:54	Matricula:1123303
Nome:Paulo	Idade:34	Matricula:1123302
Nome:Pedro	Idade:44	Matricula:1123301

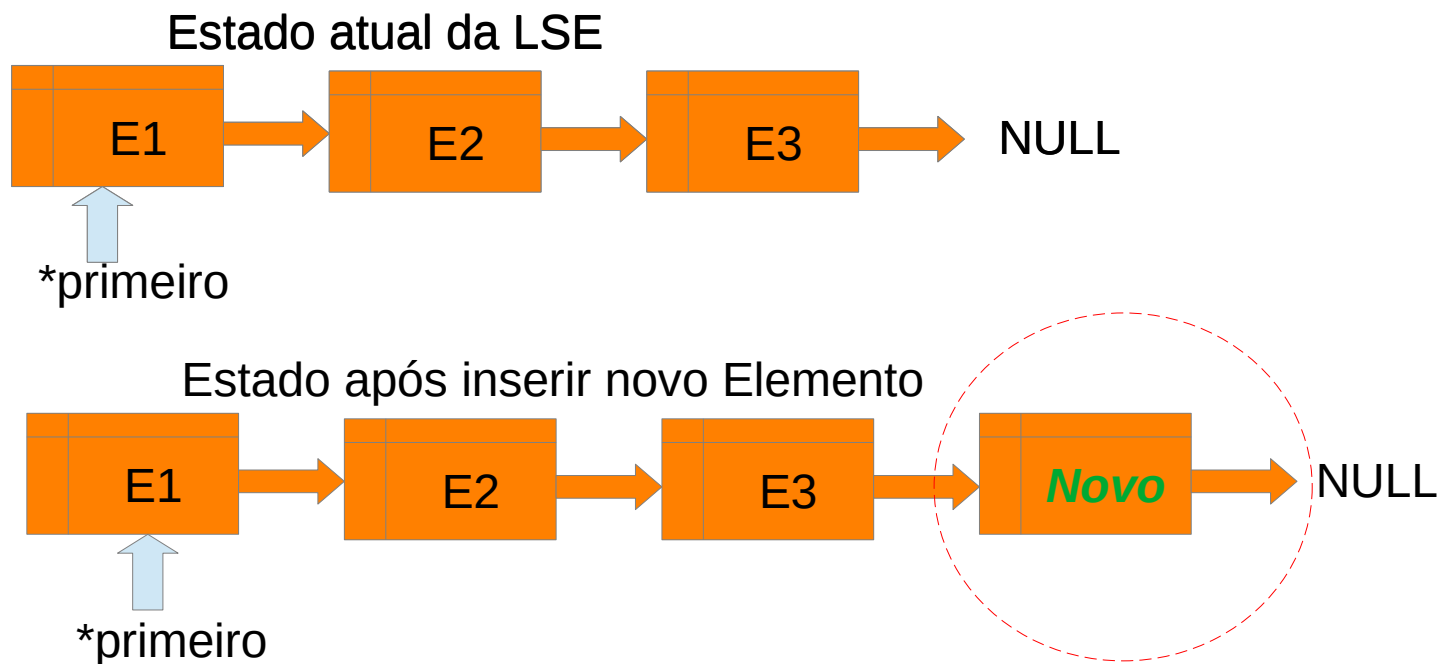
Fim da Lista

Saída no terminal



Interface da LSE – Insere no FIM

- Método para inserir um **novο** elemento no **Final** na Lista

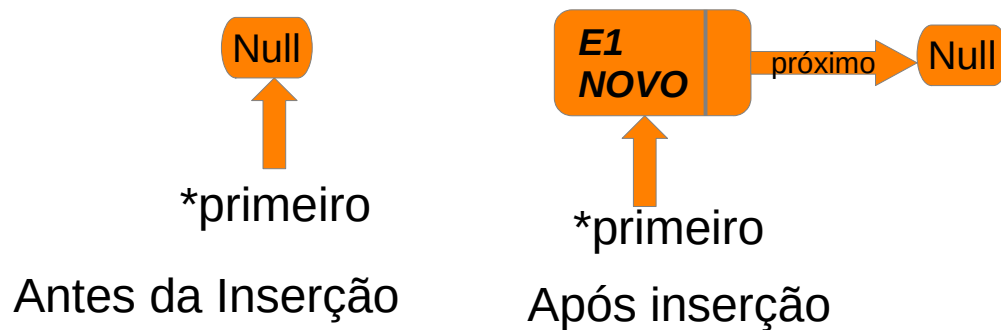


Interface da LSE – Insere no FIM

- **Inserir um elemento no *FIM* da Lista**
 - Função InsereNoFim()
 - Recebe um ponteiro de lista *ls e um ponteiro de DadosAluno *novo;
 - Função sem retorno;
 - Duas possibilidades para inserção :
 - 1) Lista Vazia antes da inserção ($n_elementos = 0$);
 - 2) Lista com elementos ($n_elementos > 0$);

Interface da LSE – Insere no FIM

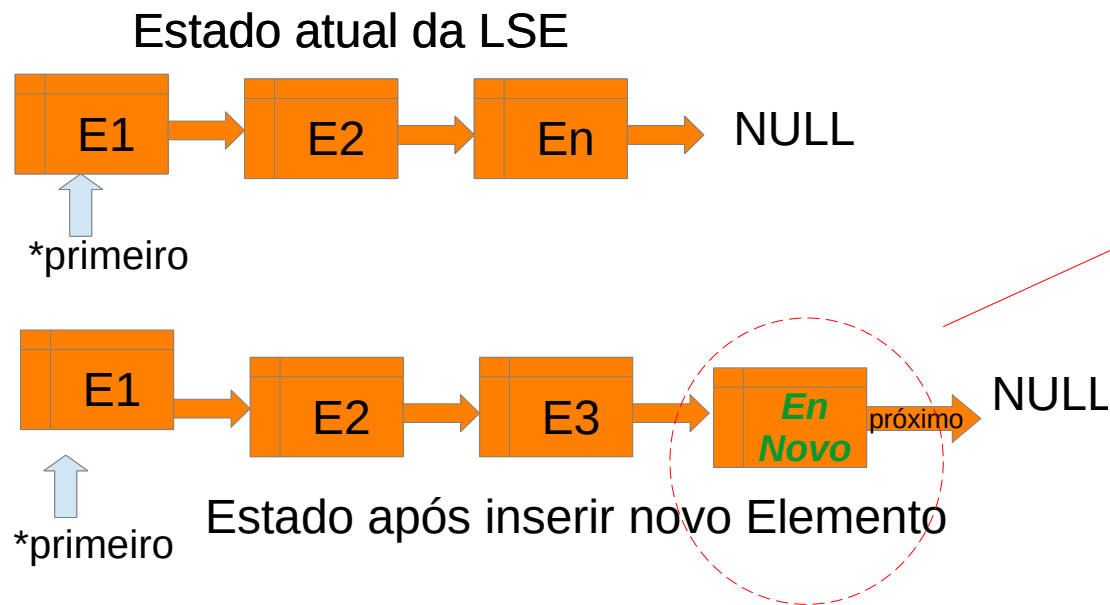
- Método para inserir um **novο** elemento no **FIM** na Lista
 - Estado Atual – Lista Vazia
 - Procedimento idêntico ao Insere no Início



```
void insereNoFim(LSE *ls, DadosAluno *novo){  
    if(ls->primeiro == NULL)  
        insereNoInicio(ls,novo);  
    else{ ...  
}
```

Interface da LSE – Insere no FIM

- Método para inserir um **novο** elemento no **FIM** na Lista
 - Estado Atual – Lista COM ELEMENTOS



```
void insereNoFim(LSE *ls, DadosAluno *novo){  
    if(ls->primeiro == NULL)  
        insereNoInicio(ls,novo);  
    else{  
        DadosAluno *aux = ls->primeiro;  
        while(aux->proximo != NULL){  
            aux = aux->proximo;  
        }  
        novo->proximo = NULL; //ultimo  
        aux->proximo = novo;  
        ls->n_elementos++;  
    }  
}
```


Programa para Inserir Alunos na Turma

- Programa para armazenar vários alunos
 - Insere 4 alunos na turma de matemática e mostra a lista;

```
int main(){  
  
    LSE *ListaMatematica = criaListaLSE();  
  
    insereNoInicio(ListaMatematica, cadastraAluno("Paulo", 44, 1123301));  
    insereNoInicio(ListaMatematica, cadastraAluno("Pedro", 34, 1123301));  
    insereNoFim(ListaMatematica, cadastraAluno("Maria", 54, 1123301));  
    insereNoFim(ListaMatematica, cadastraAluno("Paula", 24, 1123301));  
  
    mostraLista(ListaMatematica);  
}
```

Programa para Inserir Alunos na Turma

- Programa para armazenar vários alunos:
 - Execução do programa;

```
int main(){  
  
    LSE *ListaMatematica = criaListaLSE();  
  
    insereNoInicio(ListaMatematica, cadastraAluno("Paulo", 44, 1123301));  
    insereNoInicio(ListaMatematica, cadastraAluno("Pedro", 34, 1123301));  
    insereNoFim(ListaMatematica, cadastraAluno("Maria", 54, 1123301));  
    insereNoFim(ListaMatematica, cadastraAluno("Paula", 24, 1123301));  
  
    mostraLista(ListaMatematica);  
}
```

Nome: Pedro	Idade: 34	Matricula: 1123301
Nome: Paulo	Idade: 44	Matricula: 1123301
Nome: Maria	Idade: 54	Matricula: 1123301
Nome: Paula	Idade: 24	Matricula: 1123301

Saída no Terminal

Funções de Remoção

- Como Inserir um novo elemento na Lista, Funções:
 - **RemoçãoNoInicio():**
 - Insere novo elemento na posição E0;
 - **RemoveNoFim()**
 - Insere novo elemento na posição En;
 - **RemoveNaPosição(p)**
 - Insere novo elemento na posição Ep;
- **Condições a serem avaliadas:**
 - Lista vazia ($n_elementos = 0$);
 - Lista com elementos ($n_elementos > 0$);

Interface da LSE – Insere no FIM

- **RemoveNoInicio ()**

- Remove o primeiro Elemento da Lista e retorna o endereço do elemento removido;

```
DadosAluno* removeNoInicio(LSE *ls){
    DadosAluno *aux = ls->primeiro;
    if(aux == NULL)
        printf("\nErro - Lista Vazia\n");
    else{
        ls->primeiro = aux->proximo;
        ls->n_elementos--;
    }
    return aux;
}
```

Listas Simplesmente Encadeada

- RemoveNoInicio ()

```
printf("\nRemove no Início:");  
DadosAluno *removido = removeNoInicio(ListaMatematica);  
mostraLista(ListaMatematica);
```

Nome:Pedro	Idade:34	Matricula:1123301
Nome:Paulo	Idade:44	Matricula:1123301
Nome:Maria	Idade:54	Matricula:1123301
Nome:Paula	Idade:24	Matricula:1123301

Fim da Lista

Remove no Início:

Nome:Paulo	Idade:44	Matricula:1123301
Nome:Maria	Idade:54	Matricula:1123301
Nome:Paula	Idade:24	Matricula:1123301

Fim da Lista



Primeiro
Elemento
mudou

Tarefa

- Implementar as Funções da Lista LSE:
 - RemoveNoFim():
 - Recebe um ponteiro de lista e retorna uma endereço de DadosAluno;
 - RemoveNaPosição():
 - Recebe um ponteiro de lista, uma posição e retorna uma endereço de DadosAluno;
 - InsereNaPosição(int pos):
 - Recebe um ponteiro de lista, uma posição(pos) e um endereço de DadosAluno;
 - Função sem retorno.

Tarefa

- LSE.H

```
//Interface para Estrutura Lista Simplesmente Encadeada
```

```
LSE* criaListaLSE();
```

```
DadosAluno* cadastraAluno(char nome[20],int idade, int matricula);
```

```
DadosAluno* lerDadosAluno();
```

```
void insereNoInicio(LSE *ls, DadosAluno *novo);
```

```
void insereNoFim(LSE *ls, DadosAluno *novo);
```

```
void insereNaPosicao(LSE *ls, DadosAluno *novo, int pos);
```

```
DadosAluno* removeNoInicio(LSE *ls);
```

```
DadosAluno* removeNoFim(LSE *ls);
```

```
DadosAluno* removeNaPosicao(LSE *ls, int pos);
```

```
int retornaQuantidadeAluno(LSE *ls);
```

```
void mostraLista(LSE *ls);
```

```
void mostraAluno(DadosAluno *al);
```

```
void mostraPosicao(LSE *ls, int pos);
```

```
void apagaLSE(LSE *ls);
```

```
void apagaAluno(DadosAluno *al);
```

```
int buscaPosicaoOrdenada(LSE *ls, DadosAluno *novo);
```

```
//determinar a posição do novo elemento a ser inserido
```

```
//manterá ordenada a lista;
```

Tarefa

- Função Menu
 - Controla a execução da Lista

```
int menu(LSE *ls){
    printf("1 - Inserir No Início:\n");
    printf("2 - Inserir No Fim:\n");
    printf("3 - Inserir No Posição:\n");
    printf("4 - Remove No Início:\n");
    printf("5 - Remove No Fim:\n");
    printf("6 - Remove No Posição:\n");
    printf("7 - Mostra Lista:\n");
    printf("8 - InsereOrdenado \n");
    printf("0 - Sair Programa:\n");
    int op,pos;
    scanf("%d",&op);
    DadosAluno *novo;
    if(op==1){
        novo = lerDadosAluno();
        insereNoInicio(ls,novo);
    }else if(op == 2){
        novo = lerDadosAluno();
        insereNoFim(ls,novo);
    }else if(op == 3){
        printf("\nInforme a Posição:");
        scanf("%d",&pos);
        novo = lerDadosAluno();
        insereNaPosicao(ls,novo,pos);
    }else if(op == 4){
        novo = removeNoInicio(ls);
    }else if(op == 5){
        novo = removeNoFim(ls);
    }else if(op == 6){
        printf("\nInforme a Posição:");
        scanf("%d",&pos);
        removeNaPosicao(ls,pos);
    }else if(op == 7){
        mostraLista(ls);
    }else{
        printf("\nFim da Execução!\n");
    }
    return op;
}
```


Tarefa

- Execução da Lista

```
luciano@luciano-pc:~/Documentos/IFRS-2021/ED/Code-C/Aula4/Referencia$ ./roda
```

Nome:Pedro	Idade:34	Matricula:1123301
Nome:Paulo	Idade:44	Matricula:1123301
Nome:Maria	Idade:54	Matricula:1123301
Nome:Paula	Idade:24	Matricula:1123301

Fim da Lista

Remove no Início:

Nome:Paulo	Idade:44	Matricula:1123301
Nome:Maria	Idade:54	Matricula:1123301
Nome:Paula	Idade:24	Matricula:1123301

Fim da Lista

- 1 - Inserir No Início:
- 2 - Inserir No Fim:
- 3 - Inserir No Posição:
- 4 - Remove No Início:
- 5 - Remove No Fim:
- 6 - Remove No Posição:
- 7 - Mostra Lista:
- 0 - Mostra Lista:

Material de Apoio

- IFRS – Pergamum
 - SZWARCFITER, Jayme Luiz. Estruturas de dados e seus algoritmos. 3. Rio de Janeiro LTC 2010 1 recurso online ISBN 978-85-216-2995-5.
 - DEITEL, Paul; Deitel, Harvey. C: como programar - 6ª edição. Editora Pearson 850 ISBN 9788576059349.