

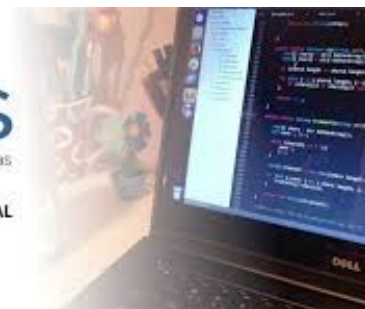
Tecnologia em Análise e Desenvolvimento de Sistemas - TADS

Estrutura de Dados I

Atividades Pedagógicas Não Presenciais – APNP 2020

Prof. Luciano Vargas Gonçalves

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



Estrutura da Dados

- **Aula 4 – Estruturas Dinâmicas**

Sumário

- **Estrutura de Dados**

- **Estáticas:**

- Tamanho Fixo;
 - Vetores e Matrizes

- **Dinâmicas:**

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

Aplicação

- **Exemplo:**
 - Agenda de Celular;
 - Lista de Chamadas de alunos;
 - Pessoas para o atendimento médico;
 - Fila de Caixa (Banco);
- **Armazena dados complexos;**
 - Necessitam de uma estrutura própria e uma estrutura de controle.
 - Crescem e decrescem conforme o sistema evolui (dinâmica);

Separação

- **Duas partes distintas:**
 - **Estrutura da Informação:**
 - Armazena as informações
 - Os dados;
 - Conteúdo armazenado;
 - **Interface:**
 - Estrutura que gerência os dados;
 - Controla as ações sobre os dados;
 - Insere, Remove, Consulta e Atualiza;

Interface

- **Usuário:**

- O usuário não precisa ficar ciente da estrutura de armazenamento;
- Apenas deve conhecer os comandos de acesso (interface):
 - Inserir, remover, alterar e etc.



Interface do Usuário

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;

Listas

- É uma coleção de **elementos** (Nó) do mesmo tipo, dispostos linearmente, que podem ou não seguir determinada organização, por exemplo:
 - [E1,E2,E3,E4,E5,....,En]
 - Onde n seja $n \geq 0$;



Elementos Encadeado (Ligados) →

Listas

- **Listas Encadeadas:**

- Simples (Deslocamento (acesso) em *um sentido*)

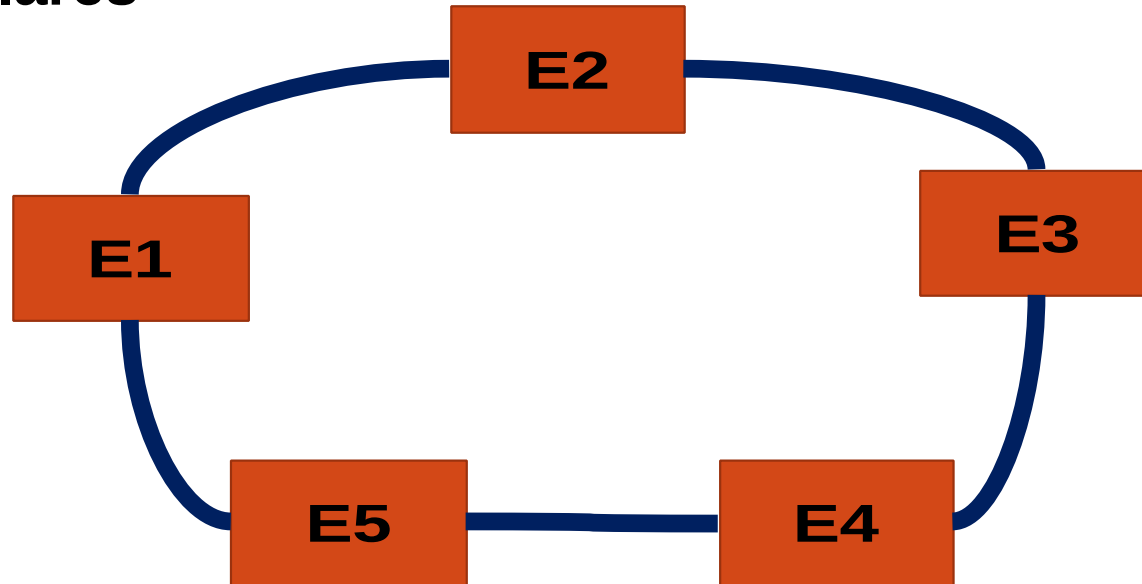


- Duplamente (Deslocamento em *ambos sentidos*)



Listas

- **Listas Simplesmente Encadeadas:**
 - **Circulares**



Listas

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

Listas - Elemento

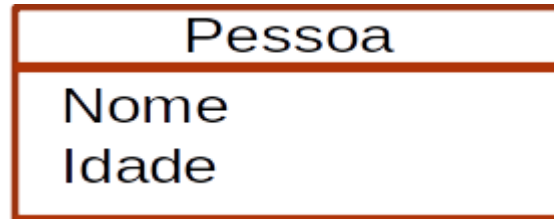
- Elemento (Nó) é a estrutura de dados que irá armazenar as informações dos integrantes;
 - A Lista de chamada armazena Alunos;
 - A fila de Banco armazena Pessoas;
 - Fila no Pedágio armazena Carros;



Elemento de armazenamento

Listas Simplesmente Encadeada- LSE

- Exemplo: Lista de Chamada
 - O elemento será uma pessoa;
 - Onde iremos armazenar as informações de uma pessoa:
 - Nome e idade;



Elemento Pessoa

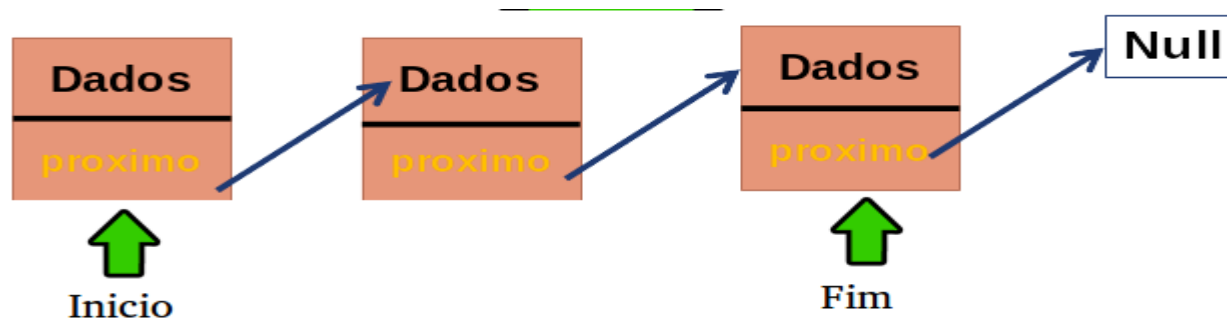
Listas Simplesmente Encadeada - LSE

- Lista pode ter de 0 ou mais elementos encadeados;
- Acesso no INÍCIO e navegação em apenas uma direção, até alcançar o FIM;
- Cada elemento tem uma apontamento para o próximo elemento



Listas Simplesmente Encadeada - LSE

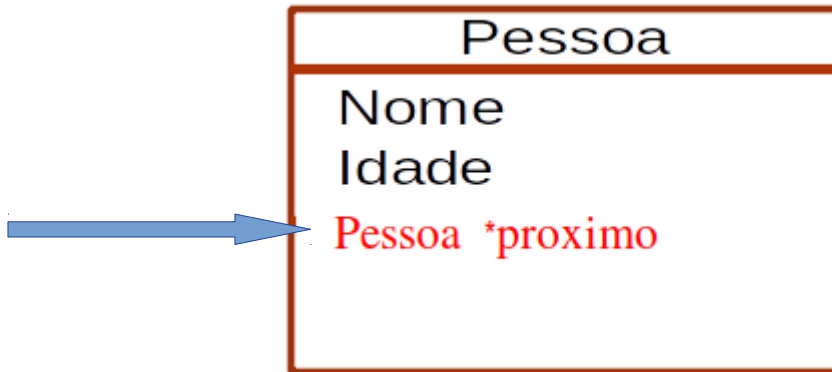
- Precisamos realizar os devidos apontamentos;
- Armazenar uma referência para o próximo em cada elemento, sendo o elo de ligação dos elementos;
- Podemos contextualizar e pensar em pessoas de mãos dadas;



Lista Simplesmente Encadeada

Listas Simplesmente Encadeada - LSE

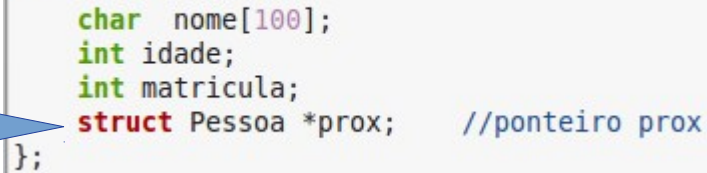
- Para apontar para o próximo elemento iremos criar uma apontador que irá armazenar uma referência para o “próximo” elemento (Pessoa).
- Próximo é um atributo do tipo Pessoa;



Listas Simplesmente Encadeada - LSE

- **Elemento de armazenamento**
 - Elemento Pessoa armazena os dados de uma pessoa e aponta para próxima Pessoa;

```
// Estrutura para Elementos
struct Pessoa{
    char nome[100];
    int idade;
    int matricula;
    struct Pessoa *prox;    //ponteiro prox
};
```



Interface da LSE

- Definição da Lista
 - Nós mais importantes (Extremos)
 - Pontos de acesso aos demais elementos da Lista
 - Primeiro e Último da lista
 - Os métodos de controle da lista;
 - Inserir, remover e consultar elemento



Interface da LSE

- Definição da Estrutura da Lista

```
#include <stdio.h>
#include <stdlib.h>

// Estrutura para Elementos
struct Pessoa{
    char nome[100];
    int idade;
    int matricula;
    struct Pessoa *prox;    //ponteiro prox
};
```



Extremos da lista

Interface da LSE

- Definição da interface Lista

```
struct Lista{  
    struct Pessoa *primeiro;  
    struct Pessoa *ultimo;  
    int np;  
};
```

```
typedef struct Pessoa pessoa;  
typedef struct Lista lista;
```

Extremos da lista



Interface da LSE

- **A Lista de pessoas terá as funções:**
 - **Inserir_No_Inicio (Pessoa psNova)**
 - Recebe uma Pessoa (psNova) e insere no início da lista
 - **Inserir_No_Fim (Pessoa psNova)**
 - Recebe uma Pessoa (psNova) e insere no fim da lista
 - **ContaElementos()**
 - Retorna o número de Elementos da Lista
 - **ExibirLista ()**
 - Mostra todos elementos da Lista
 - **Remove_No_Inicio ()**
 - Remove o primeiro Elemento da Lista
 - **Remove_No_Fim ()**
 - Remove o último Elemento da Lista

Interface da LSE

- **Função Cria a Lista – Interface da Lista**
 - Inicialização da lista
 - Recebe um ponteiro para lista L;

```
void cria_lista(lista *l){  
    l->primeiro = NULL;  
    l->ultimo = NULL;  
    l->np = 0;  
}
```

Interface da LSE

- **Função novaPessoa – Cria um elemento tipo Pessoa**
 - Cria um elemento Pessoa para inserir na lista;
 - Retorna um ponteiro pessoa;

```

pessoa* novaPessoa(){
    pessoa *novo = (pessoa *)malloc(sizeof(pessoa));
    if(!novo){
        printf("Sem Memoria disponivel!\n");
        exit(0);
    }
    printf("Novo aluno - Nome: ");
    scanf ("%s",novo->nome);
    printf("Novo aluno - Idade: ");
    scanf ("%d", &novo->idade);
    novo->matricula=matricula++;

    return novo;
}

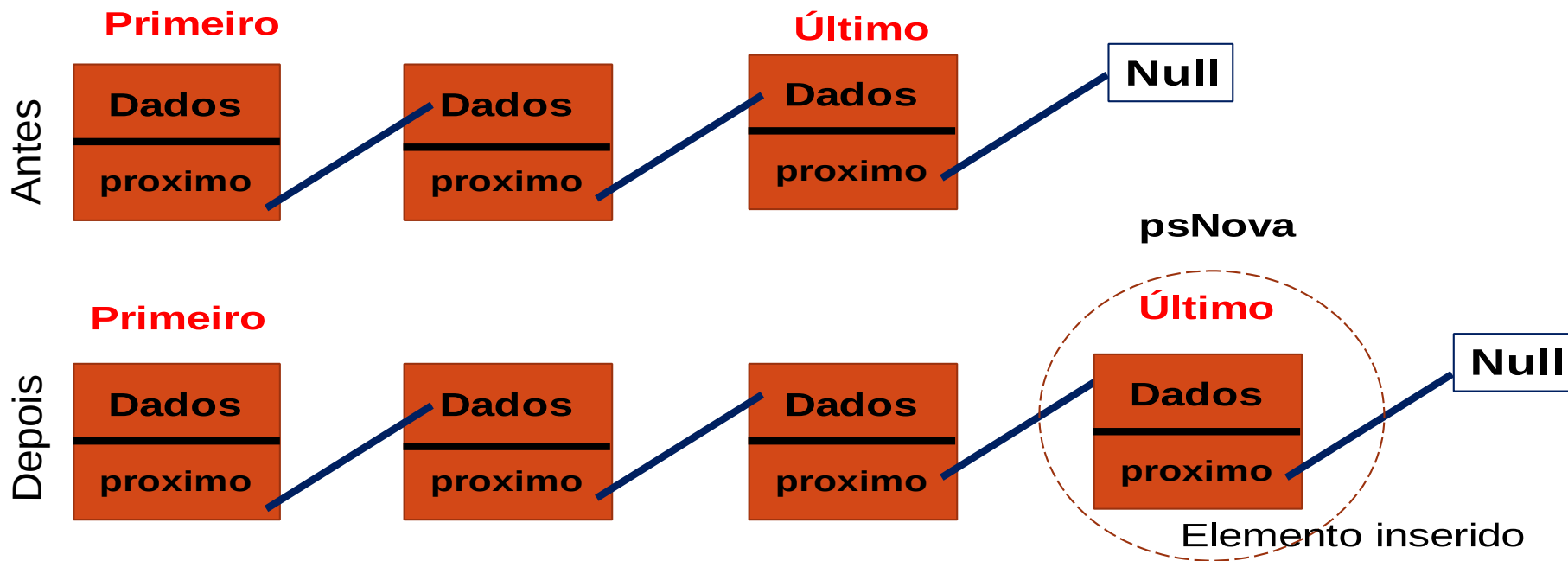
```



Reserva de Memória

Interface da LSE

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

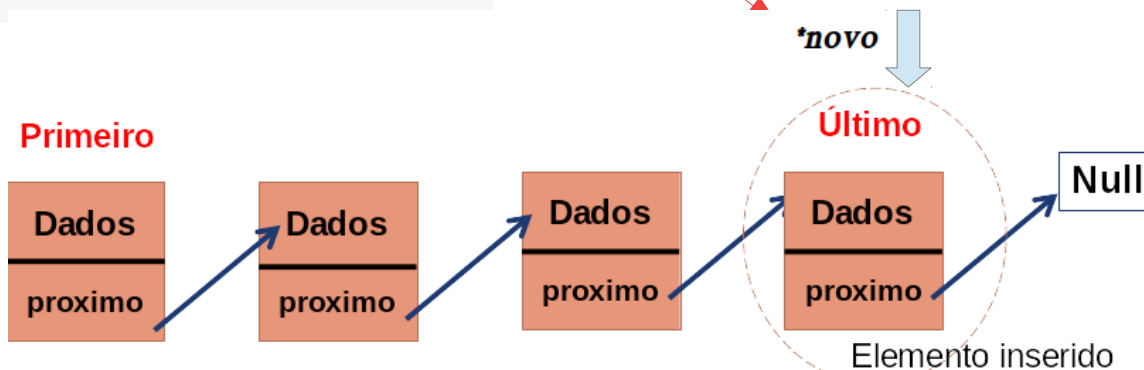


Listas Simplesmente Encadeada

Inserir um elemento no *Final* na Lista

```
void insereFim(lista *l, pessoa *novo){  
    novo->prox = NULL;  
    if(l->primeiro == NULL)  
        l->primeiro = novo;  
    else{  
        l->ultimo->prox = novo;  
    }  
    l->ultimo = novo;  
    l->np++;  
}
```

Ponteiro para Lista e Elemento



Listas Simplesmente Encadeada

Inserir um elemento no *Final* na Lista

```
void insereFim(lista *l, pessoa *novo){  
    novo->prox = NULL;  
    if(l->primeiro == NULL)   
        l->primeiro = novo;  
    else{  
        l->ultimo->prox = novo;  
    }  
    l->ultimo = novo;  
    l->np++;  
}
```

Testa Lista vazia

Atualiza o ponteiro para último elemento

Listas Simplesmente Encadeada

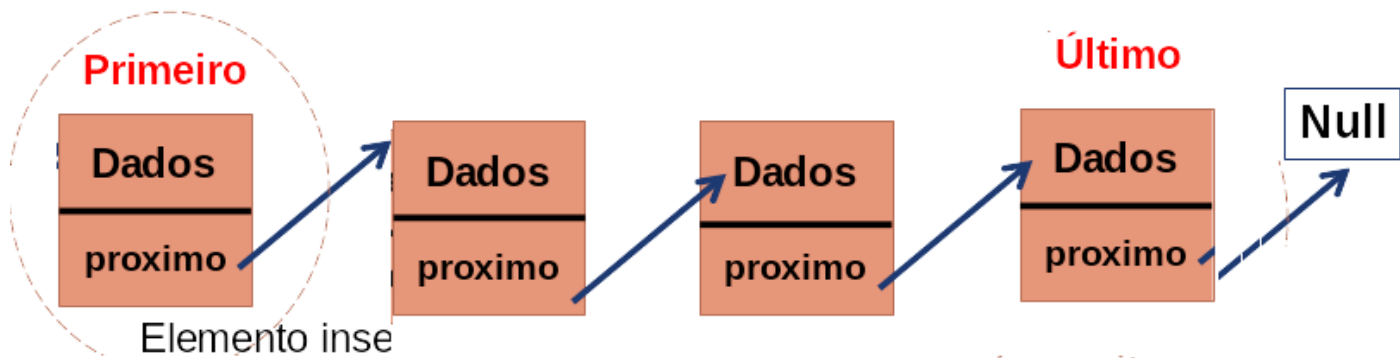
Inserir um elemento no **INÍCIO** na Lista

```
void insereInicio(lista *l, pessoa *novo){  
    if(l->primeiro == NULL)  
        l->ultimo = novo; ;  
    else{  
        novo->prox = l->primeiro;  
    }  
    l->primeiro = novo;  
    l->np++;  
}
```

Testa Lista vazia

Atualiza o ponteiro para o primeiro elemento

**novo*



Listas Simplesmente Encadeada

- A Lista de pessoas terá as funções:
 - **Inserir_No_Fim(Pessoa psNova)**
- **Falta implementar**
 - ExibirLista()
 - Mostra todos elementos da Lista
 - Remove_No_Inicio ()
 - Remove o primeiro Elemento da Lista

Listas Simplesmente Encadeada

- **ExibirLista()**

```
void mostraLista(lista *l){  
    if (l->np == 0){  
        printf("Lista Vazia!\n\n");  
        return ;  
    }  
    pessoa *tmp = l->primeiro;  
    do{  
        printf("Nome= %s,   idade = %d e matricula = %d\n ",tmp->nome, tmp->idade,tmp->matricula);  
        tmp = tmp->prox;  
    } while(tmp != NULL);  
}
```

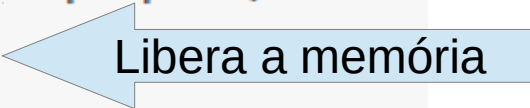
Ponteiro Auxiliar

Avanço ou Incremento

Listas Simplesmente Encadeada

- **Remove_No_Inicio ()**
 - Remove o primeiro Elemento da Lista

```
int removeInicio(lista *l){  
    if(l->np>0){  
        pessoa *p;  
        p = l->primeiro;  
        l->primeiro = p->prox;  
        free(p);  
        l->np--;  
        return 1;  
    }  
  
    return 0;  
}
```



Libera a memória

Implementação LSE.

Precisamos do “main()” implementar uma aplicação fazendo uso do LSE

```
#include <stdio.h>
#include <stdlib.h>
#include "lib.h"

lista *lp;

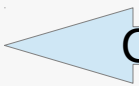
void main(void){
    lp = (lista * ) malloc (sizeof(lista));
    cria_lista(lp);

    mostraLista(lp); //MOSTRA LISTA VAZIA

    pessoa *p1;
    p1 = novaPessoa(); //INSERE A PRIMEIRA PESSOA
    insereFim(lp,p1);

    insereFim(lp,novaPessoa()); //INSERE A SEGUNDA PESSOA
    insereInicio(lp,novaPessoa()); //INSERE A TERCEIRA PESSOA

    mostraLista(lp);
}
```



Cabeça da Lista

Implementação LSE.

- Teste com lista de 3 pessoas
 - Dois insere no fim e um insere no início

```
ogramas/Aula3 $ ./roda
Novo aluno - Nome: ada
Novo aluno - Idade: 22
Novo aluno - Matricula: 11
Novo aluno - Nome: baa
Novo aluno - Idade: 33
Novo aluno - Matricula: 21
Novo aluno - Nome: cas
Novo aluno - Idade: 333
Novo aluno - Matricula: 11
Nome= cas,      idade = 333 e matricula = 11
Nome= ada,      idade = 22 e matricula = 11
Nome= baa,      idade = 33 e matricula = 21
```


Code::Blocks

- Exemplo da Lista de Números

–