



INSTITUTO FEDERAL
Sul-rio-grandense

Câmpus
Pelotas

EDUCAÇÃO
PÚBLICA
100%
GRATUITA

Estrutura de Dados

Aula 7

Estrutura: LISTAS

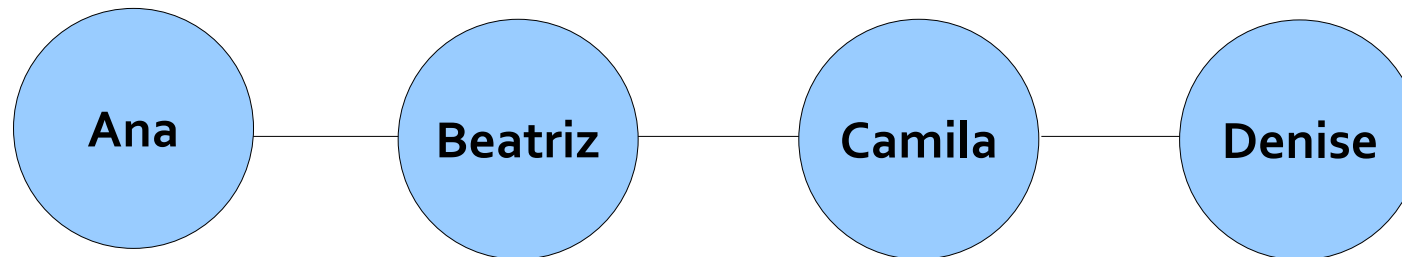
Representadas por Contiguidade Física
ListaCF

Lista Linear

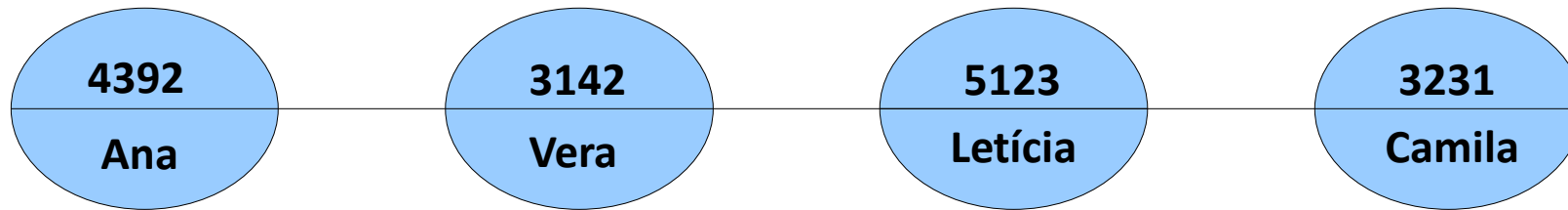
É uma estrutura caracterizada por uma sequência de elementos, que mantêm entre si uma relação de ordem e_1, e_2, \dots, e_n com $n \geq 0$, tal que:

- e_1 é o primeiro elemento da lista
- e_n é o último elemento da lista
- e_k , $1 < k < n$ é precedido pelo elemento e_{k-1} e seguido por e_{k+1}

Se $n = 0$ então a lista é vazia.



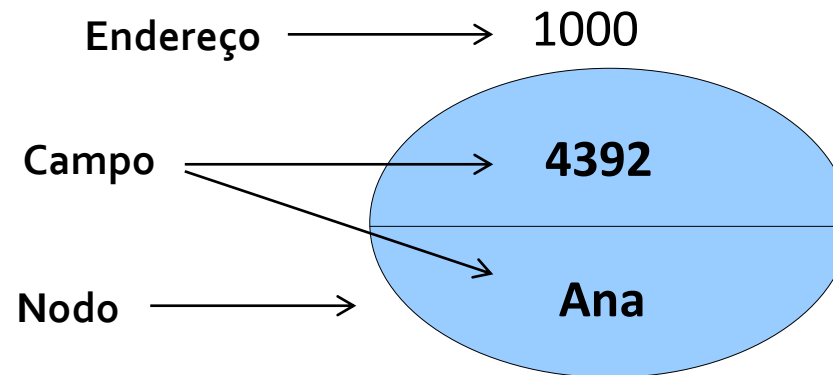
Lista Linear



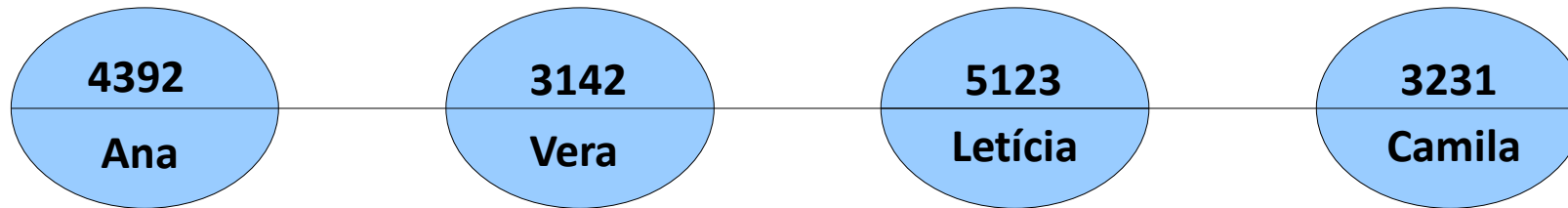
Nodo ou nó: É a unidade básica de informação de uma certa estrutura.

Campo de um nodo: É a subdivisão de um nodo.

Endereço de um nodo: É a localização do nodo dentro de uma área de armazenamento.



Lista Linear



Operações:

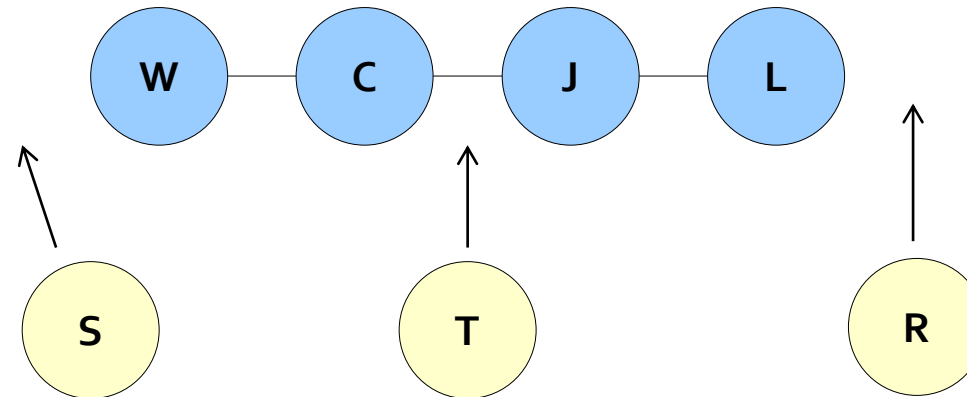
- Criar uma lista;
- Verificar se uma lista está vazia;
- Obter o tamanho de uma lista;
- Obter/modificar o valor do elemento de uma determinada posição da lista;
- Obter a posição de um elemento cujo valor é dado;
- Acessar um elemento da lista;
- Inserir um elemento na lista;
- Remover um elemento da lista;
- Percorrer a lista;
- Concatenar duas listas;
- Classificar os elementos de uma lista;
- Outras ...

Lista Linear: Operações

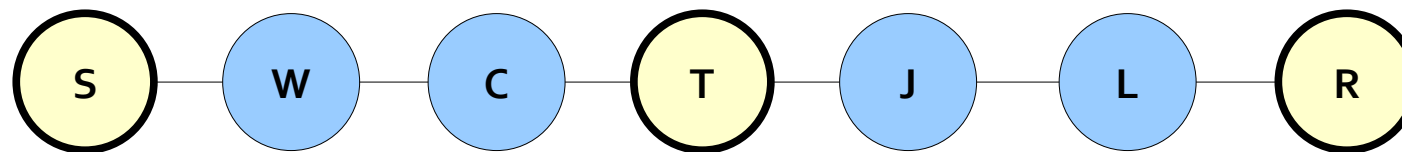
Operações:

- **Inserção:** Acrescentar um novo nodo na lista.

Antes da operação:



Depois da operação:

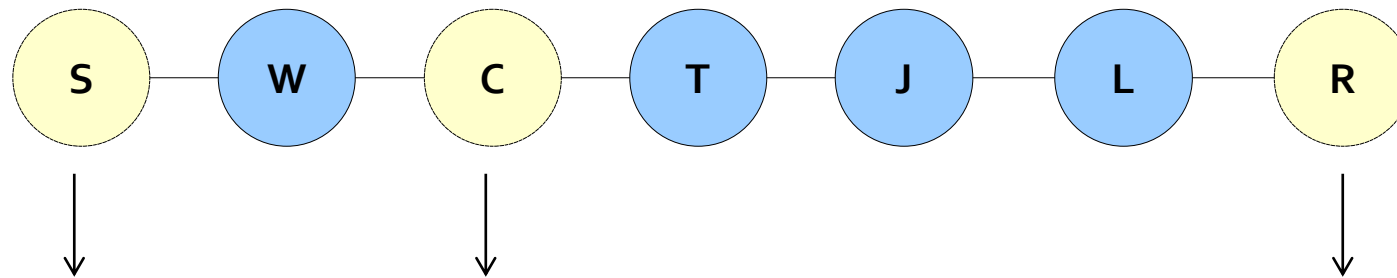


Lista Linear: Operações

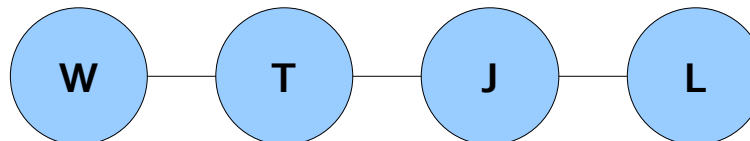
Operações:

- **Remoção:** Retirar um nodo da lista.

Antes da operação:



Depois da operação:

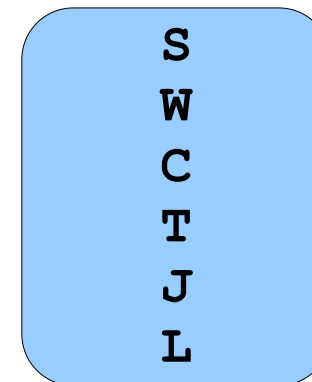
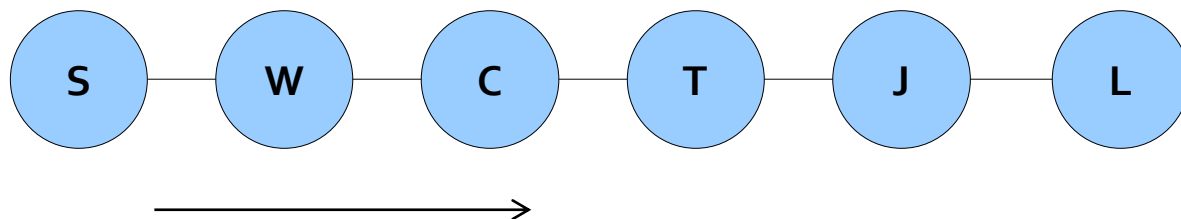


Lista Linear: Operações

Operações:

- **Percurso:** Percorrer todos os nodos da lista para executar algum operação sobre seus elementos.

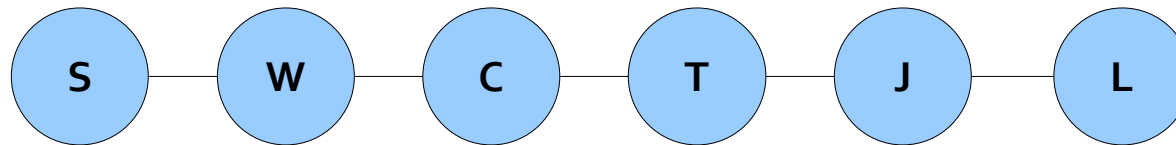
Ex: Exibir todos os nodos da lista.



Lista Linear: Operações

Operações:

- **Procura:** Procurar na lista um nodo que contenha uma informação desejada.



Existe S?

SIM

Existe T?

SIM

Existe K?

NÃO

Lista Linear: Representação

Implementação:

- Representação por **contiguidade física**.
 - Os nodos são armazenados em endereços contíguos, ou igualmente distanciados um do outro.
- Representação **por encadeamento**.
 - Os nodos são armazenados em endereços que não mantém qualquer relação entre si. Os relacionamentos entre os nodos são representados por meio de ligações físicas explícitas.

Lista Linear: Representação

Representação por **Contiguidade Física**:

- Os nodos são armazenados em endereços contíguos, ou igualmente distanciados um do outro.
- Os relacionamentos são representados pela disposição física dos componentes na memória.
- A posição na estrutura lógica determina a posição na estrutura física.

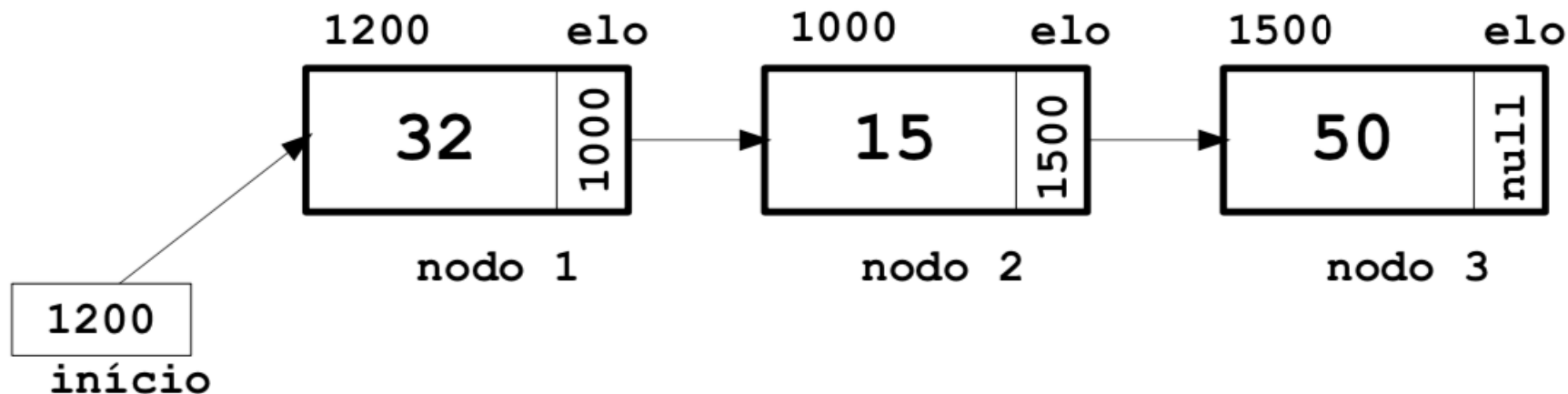
end 1	end 2	end 3
nodo 1	nodo 2	nodo 3

Mémoria		
Endereços		
1001	32	nodo 1
1002		
1003	15	nodo 2
1004		
1005	50	nodo 3
1006		

Lista Linear: Representação

Representação por **Encadeamento**:

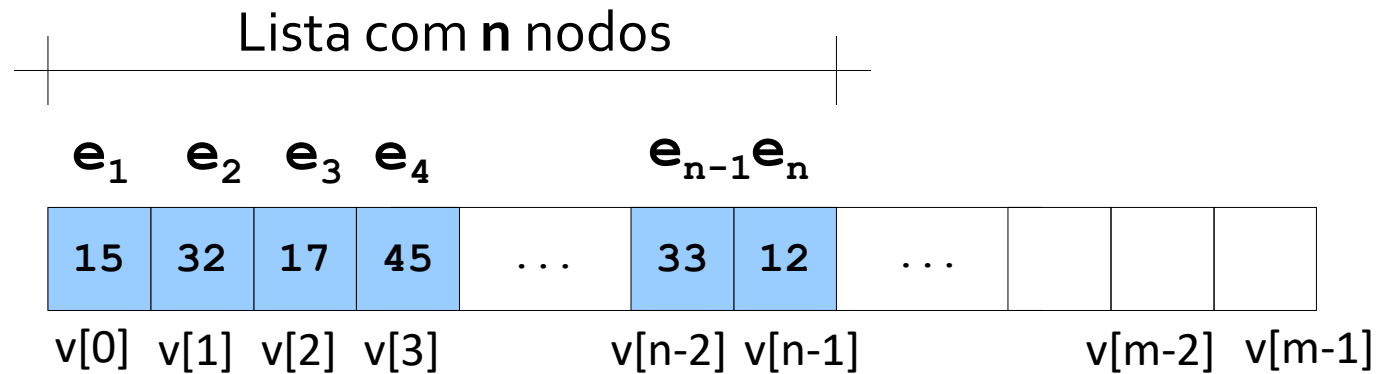
- A disposição física dos nodos independe de sua posição na estrutura lógica.
- Os relacionamentos são representados por elos que são ligações físicas explícitas.
- O valor contido em um campo de elo é o endereço de outro nodo.



Lista : Contiguidade Física

Implementação: Contiguidade física

- A lista é representada como parte de um vetor de **m** elementos.



n : Quantidade de nodos da lista.

v : Vetor que armazena os nodos da lista.

m : Capacidade (máxima do vetor).

Lista : Contiguidade Física

Tipo: **ListaCF**

Uma lista representada por contiguidade física.

```
#define    SUCESSO        0           /* Códigos de erro */
#define    LISTA_VAZIA  1
#define    LISTA_CHEIA  2
#define    CODIGO_INEXISTENTE  3
#define    MAX_NODOS    5

typedef struct {
    int idade;
} Dado;

typedef struct {
    Dado v[MAX_NODOS];
    int n;           /* Quantidade de elementos */
} ListaCF;
```

Lista : Operações

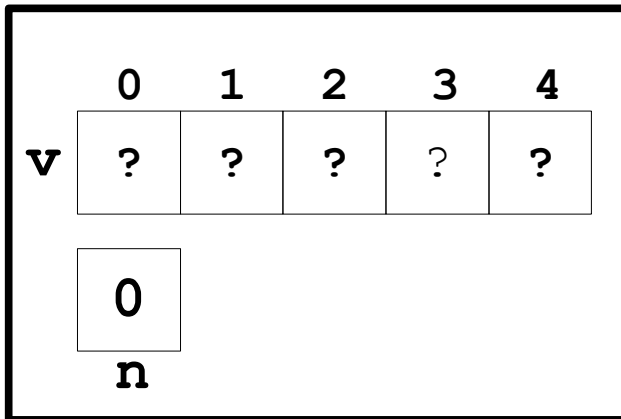
Algumas operações:

```
void criaLista (ListaCF *lt);  
int  incluiNoFim(ListaCF *lt, Dado d);  
void exhibe(ListaCF lt);  
int  excluiDoInicio (ListaCF *lt, Dado *d);
```



Lista : CriaLista

Cria uma lista
E/S: uma lista



lista

```
int main()
{
    ListaCF lista;

    criaLista(&lista);
    return 0;
}

void        criaLista
(ListaCF *lt)
{
    lt->n = 0;
}
```



EDUCAÇÃO
PÚBLICA

100%
GRATUITA

Lista : IncluiNoFim

Incluir no final da lista:

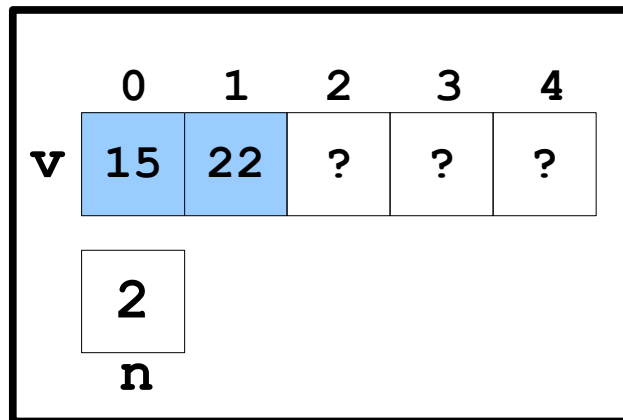
E/S: Uma lista

Entrada: O dado que será incluído.

Retorno: Código de erro **SUCESSO** ou **LISTA_CHEIA** (Overflow).

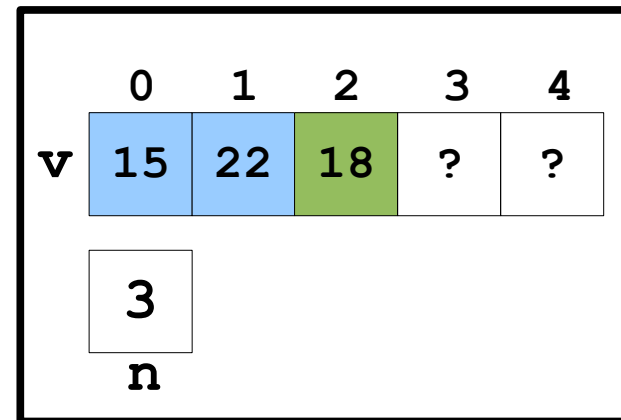
Inclusão do dado: **idade=18** em uma lista com 2 nodos

Antes:



lista

Depois:



lista


```

int main() {
    ListaCF lista; Dado d;

    criaLista(&lista);
    d.idade = 18;
    if (incluiNoFim(&lista,d)==LISTA_CHEIA)
        printf("ERRO: Lista cheia");
    else
        printf("Inclusão OK");
    return 0;
}

```

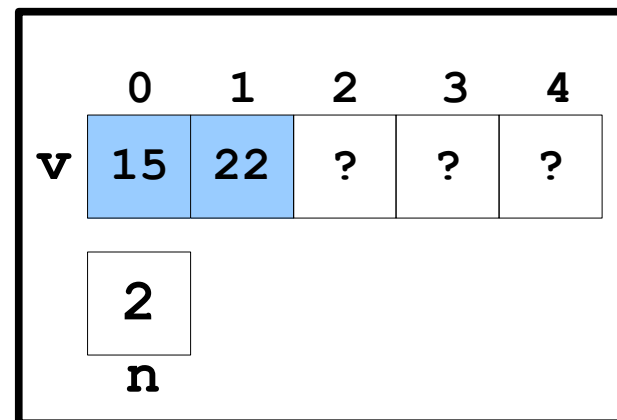
```

...
int incluiNoFim(ListaCF *lt, Dado d) {
    if (lt->n==MAX_NODOS)
        return LISTA_CHEIA;
    else {
        lt->v[lt->n] = d;
        lt->n++;
        return SUCESSO;
    }
}

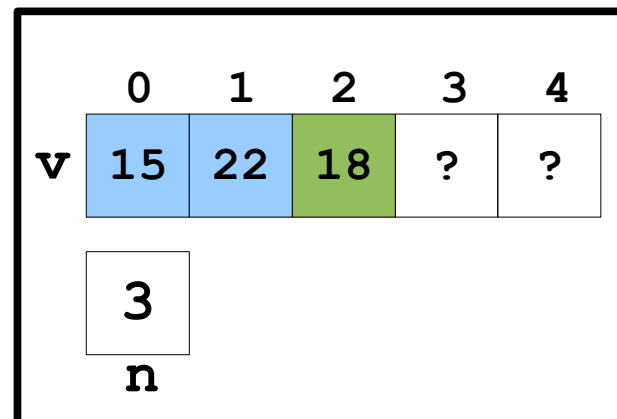
```

Lista : IncluiNoFim

Antes:



Depois:



Lista : Exibe

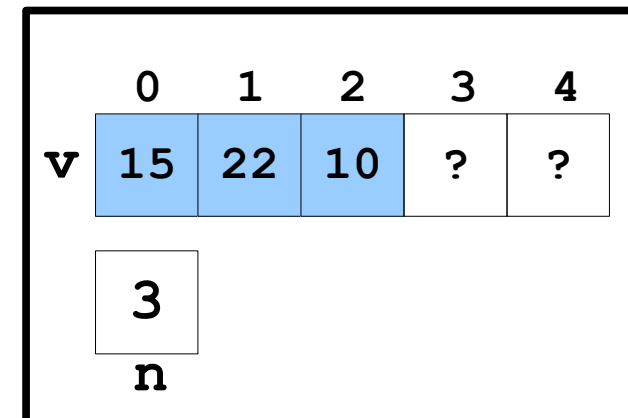
```
int main() {  
    ListCF lista;  Dado d;  
  
    criaLista(&lista);  
    d.idade=15;    incluiNoFim(&lista,d);  
    d.idade=22;    incluiNoFim(&lista,d);  
    d.idade=10;    incluiNoFim(&lista,d);  
    exibe(lista);  
    return 0;  
}
```

```
...  
void exibe(ListaCF lt) {  
    int i;  
    for(i=0; i<lt.n; i++)  
        printf("%d\n",lt.v[i].idade);  
}
```

Exibir o conteúdo da lista

Entrada: Nenhuma.

Retorno: Nenhuma.



lista

15
22
10

Lista : ExcluiDoInicio

Excluir o nodo que está no início da lista:

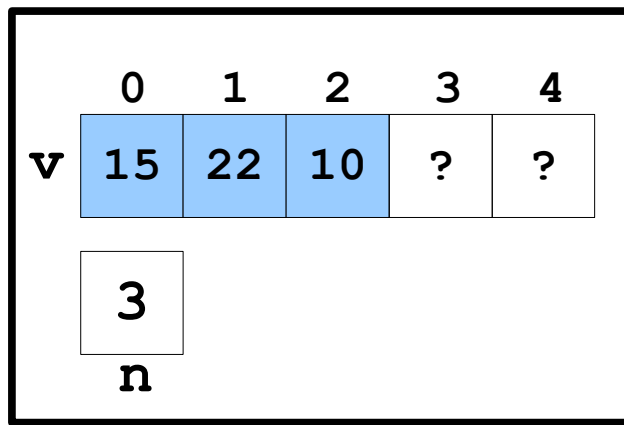
Entrada: Uma lista.

Saída: Nodo excluído.

Retorno: Código de erro **SUCESSO** ou **LISTA_VAZIA** (*Underflow*).

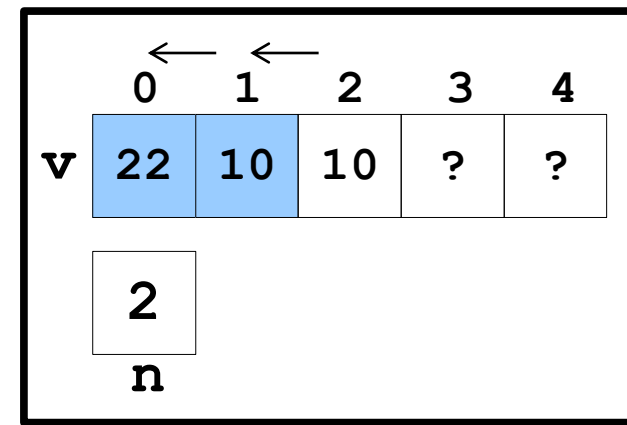
Exclusão do **primeiro** nodo da lista: **15**

Antes:



lista

Depois:

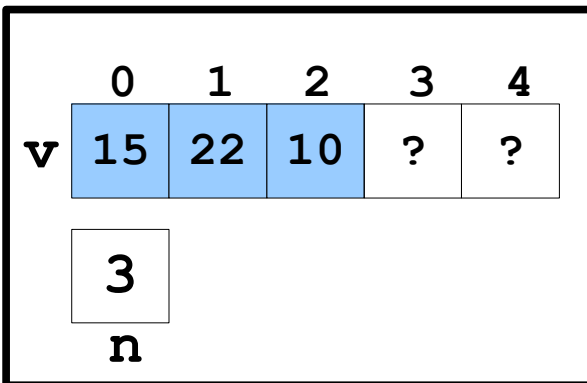


lista

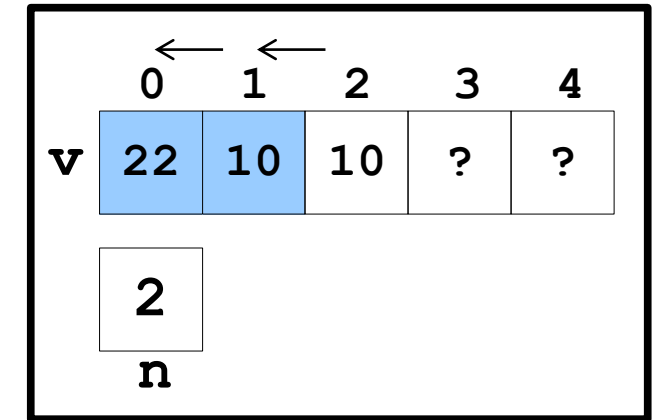
Lista : ExcluiDoInicio

```
...
int excluiDoInicio(ListaCF *lt,Dado *d) {
    int i;
    if (lt->n==0)
        return LISTA_VAZIA;
    else {
        *d = lt->v[0];
        for (i=1; i<lt->n; i++)    // Deslocamento
            lt->v[i-1] = lt->v[i];
        lt->n--;
        return SUCESSO;
    }
}
```

Antes:



Depois:



```
int main() {
    ListaCF lista; Dado d;

    criaLista(&lista);
    d.idade=15; incluiNoFim(&lista,d);
    d.idade=22; incluiNoFim(&lista,d);
    d.idade=10; incluiNoFim(&lista,d);
    if (excluiDoInicio(&lista,&d)==LISTA_VAZIA)
        printf("ERRO: Lista vazia\n");
    else
        printf("Dado excluído: %d\n",d.idade);
    return 0;
}
```

Lista : Operações

Algumas operações:

- Criar Lista
- Exibir
- Incluir no Fim
- Incluir no Início
- Incluir Depois
- Incluir Antes
- Excluir do Início
- Excluir do Fim
- Excluir Nodo
- Pesquisar Nodo
- ...

Representação por Contiguidade Física

- ❖ Permite o acesso randômico.
- ❖ Facilita a transferência de dados (área de memória contígua).
- ❖ O tamanho da lista precisa ser conhecido e alocado antecipadamente.
- ❖ Mantém um espaço de memória ocioso.



INSTITUTO FEDERAL
Sul-rio-grandense

Câmpus
Pelotas

EDUCAÇÃO
PÚBLICA
100%
GRATUITA

Estrutura de Dados

Aula 7

Estrutura: LISTAS

Representadas por Contiguidade Física
ListaCF