Estruturas de dados

Listas lineares

CSTSI – Semestre 2

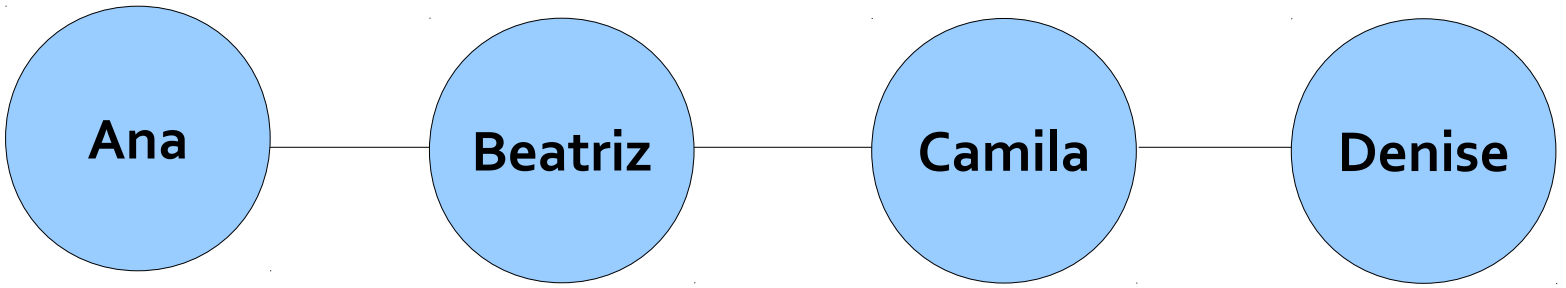
05

Estruturas de dados

Estudo das técnicas de representação e manipulação de dados.

Problema: Obter uma relação dos alunos matriculados na disciplina de Estruturas de dados classificada em ordem alfabética.

Estrutura: **Lista linear**



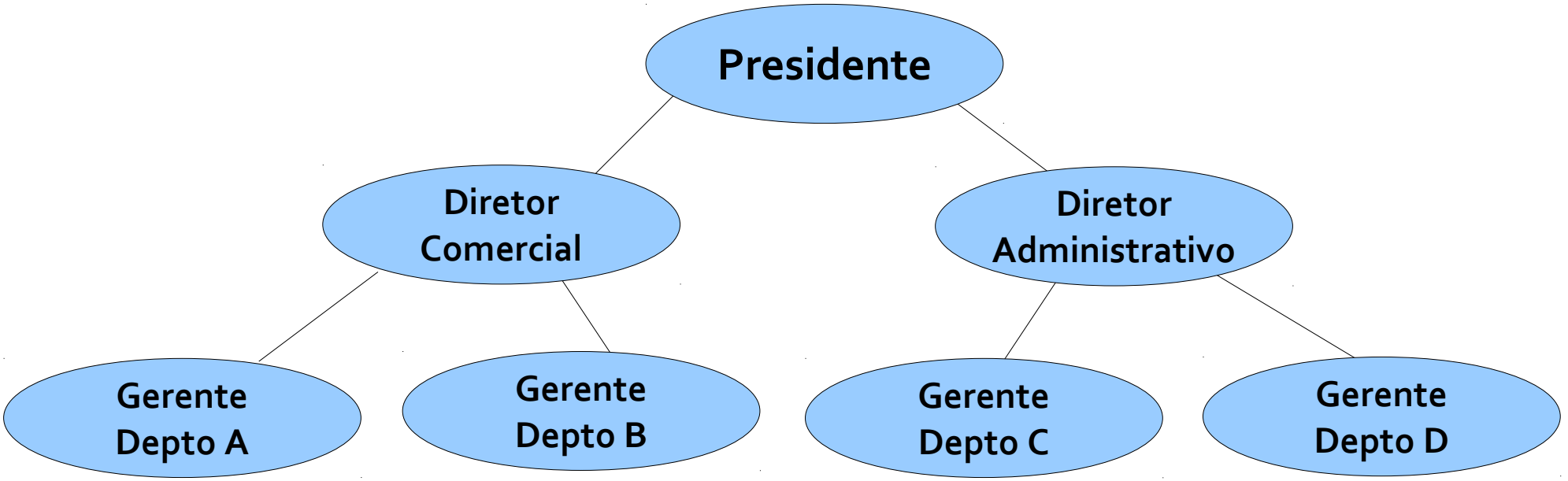
Como representar na memória?

Operações? **inserir**, **remover**, **procurar**, **percorrer**

Estruturas de dados

Problema: O gerente do departamento B é subordinado a quem?

Estrutura: **Árvore**



Como representar na memória?

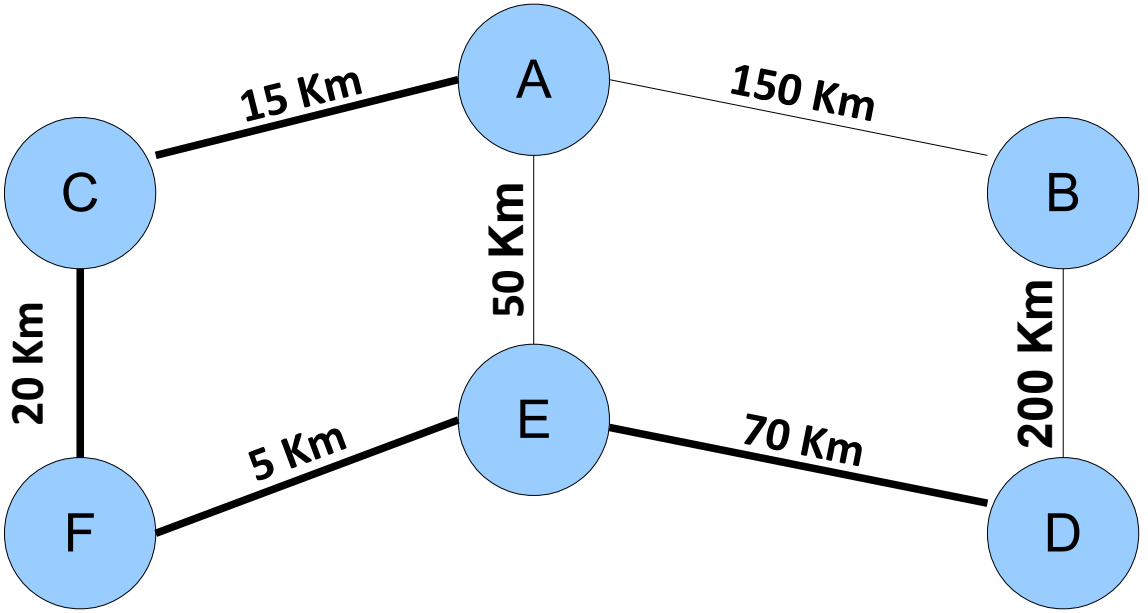
Operações? **inserir**, **remover**, **procurar**, **percorrer**

Estruturas de dados

Problema: Dada uma relação de cidades e as respectivas distâncias entre elas, determinar o caminho mais curto entre duas destas cidades.

Ex: Qual o menor percurso entre A e D?

Estrutura: **Grafo**



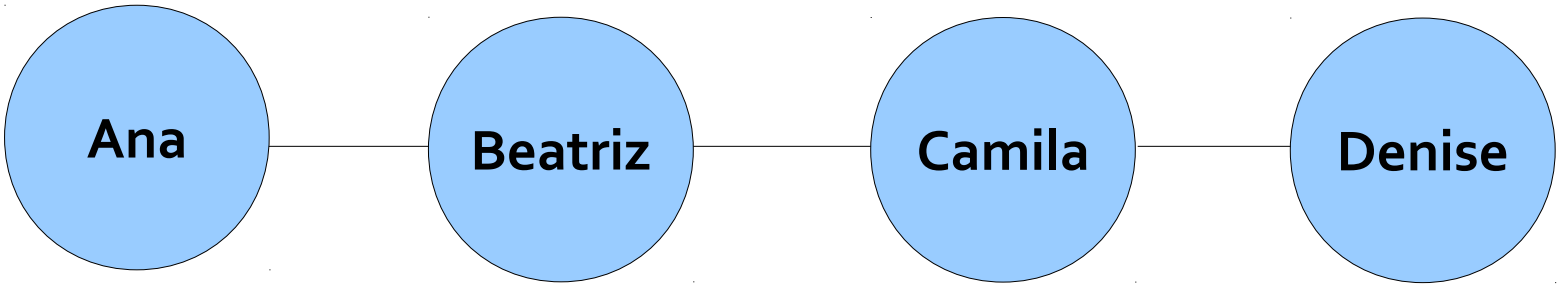
Como representar na memória?

Lista linear

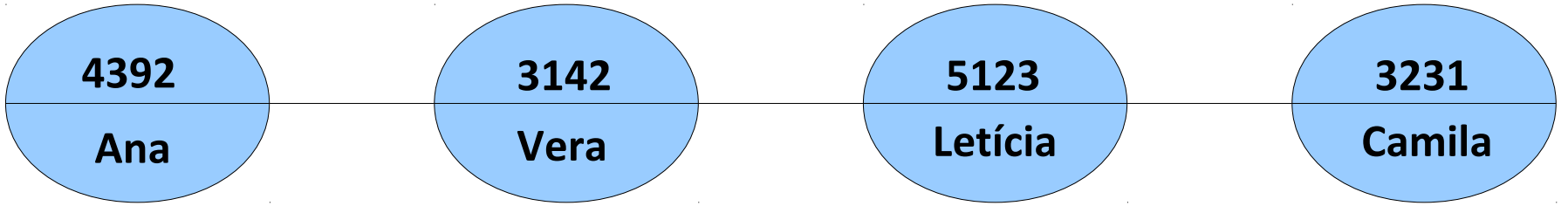
É uma estrutura caracterizada por uma sequência de elementos, que mantém entre si uma relação de ordem $\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_n}$ com n>=o, tal que:

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

Se **n = o** então a lista é vazia.



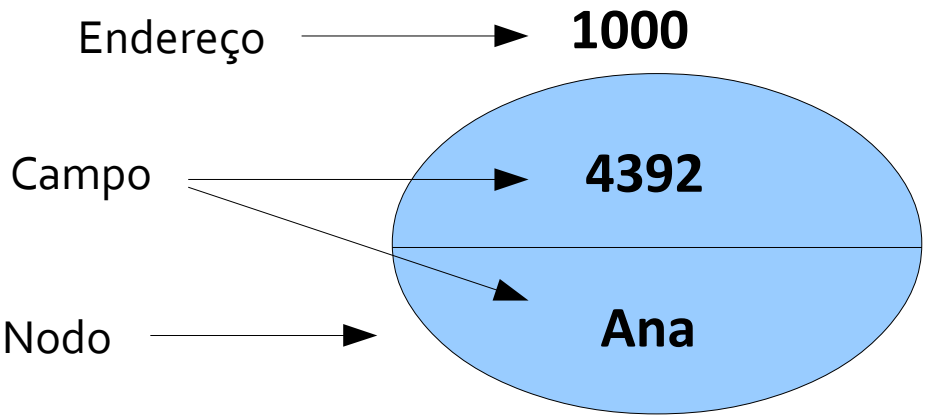
Lista linear



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:

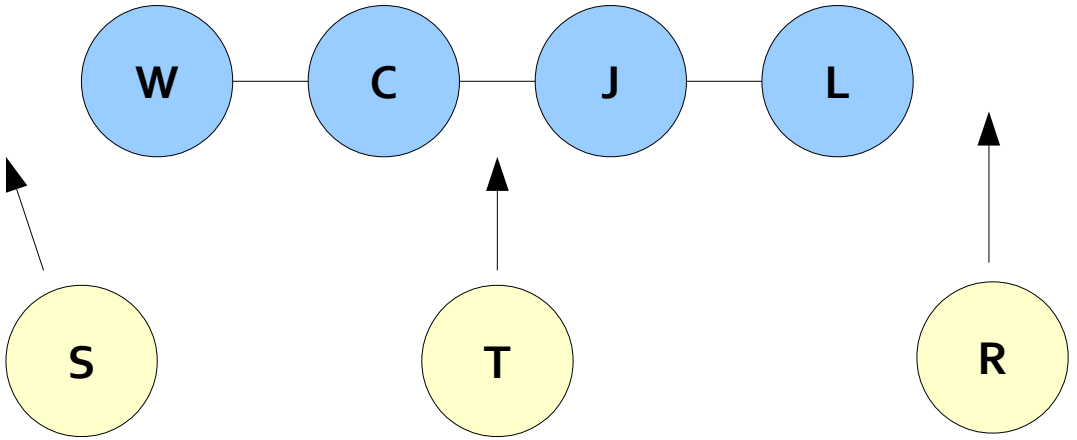
- Criar uma lista;
- Verificar se uma lista está vazia;
- Obter o tamanho de uma lista;
- Obter/modificar o valor do elemento de uma determinada posição da lista;
- Obter a posição de um elemento cujo valor é dado;
- Acessar um elemento da lista;
- Inserir um elemento na lista;
- Remover um elemento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

Lista linear

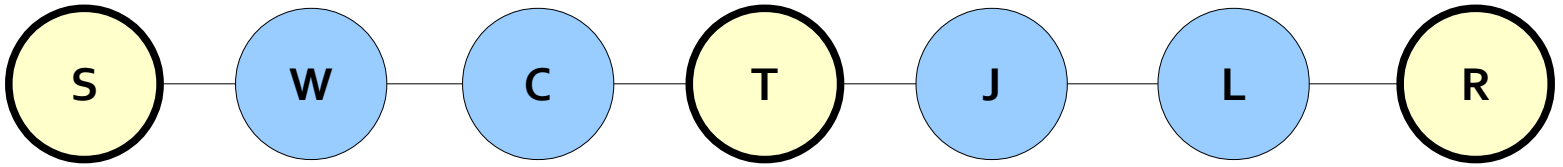
Operações:

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

Antes da operação:



Depois da operação:

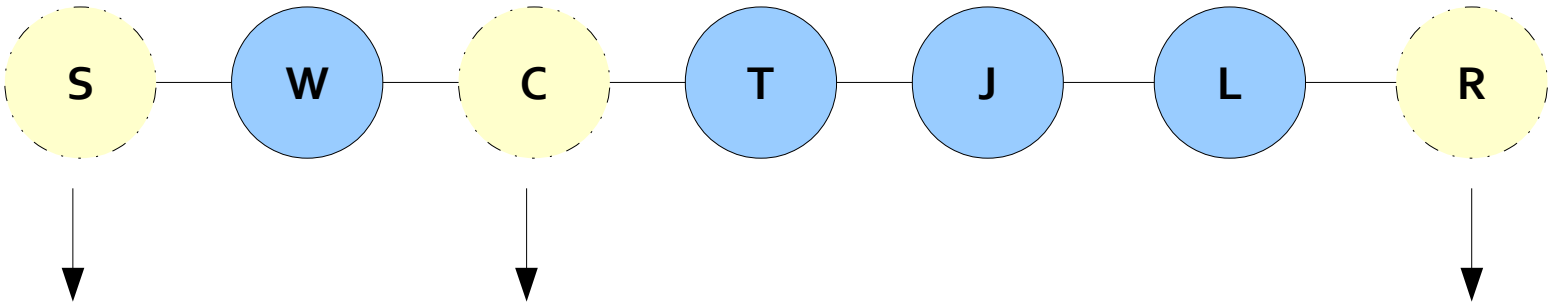


Lista linear

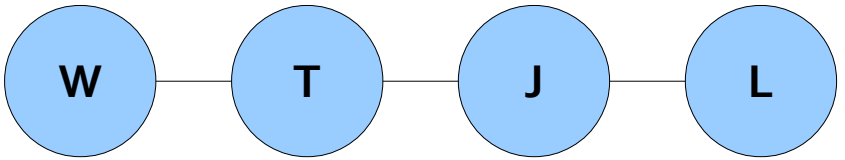
Operações:

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

Antes da operação:



Depois da operação:

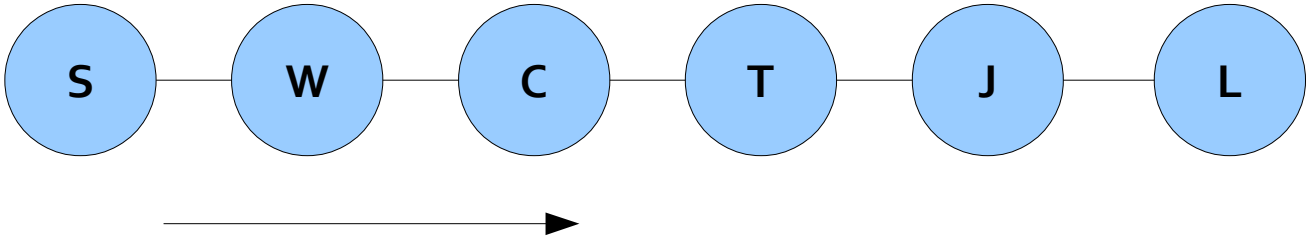


Lista linear

Operações:

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

Ex: Exibir todos os nodos da lista.

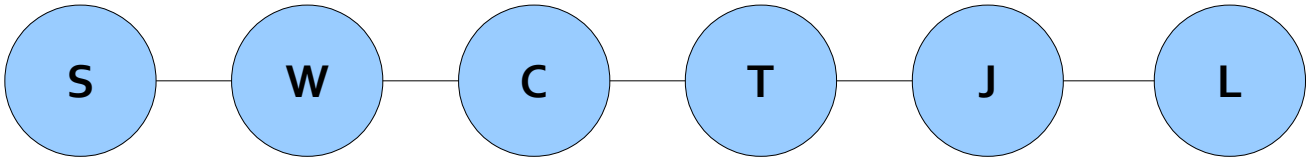



```
S
W
C
T
J
L
```

Lista linear

Operações:

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




```
Existe S?
   SIM
Existe T?
   SIM
Existe K?
   NÃO
```

Lista linear

Implementação:

- Representação por contiguidade física.

  - Os nodos são armazenados em endereços contíguos, ou igualmente distanciados um do outro.

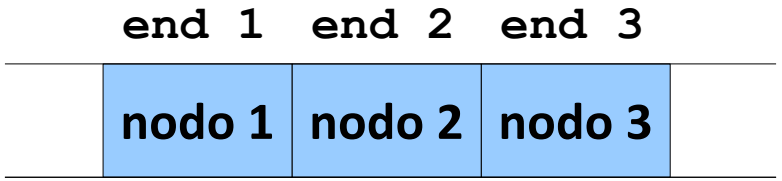
- Representação por encadeamento.

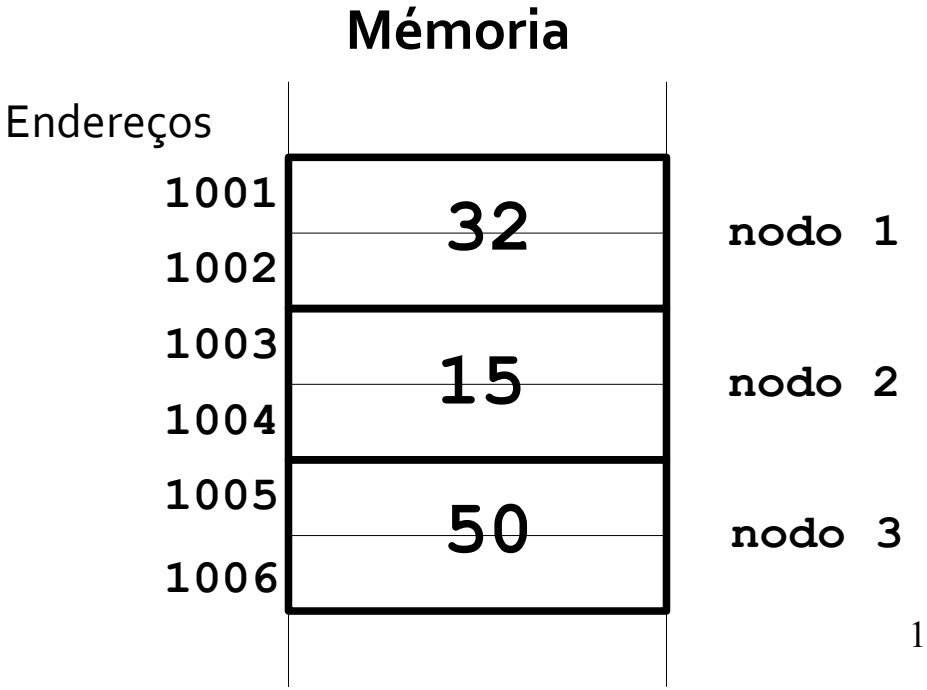
  - Os nodos são armazenados em endereços que não mantém qualquer relação entre si. Os relacionamentos entre os nodos são representados por meio de ligações físicas explícitas.

Lista linear

Representação 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.

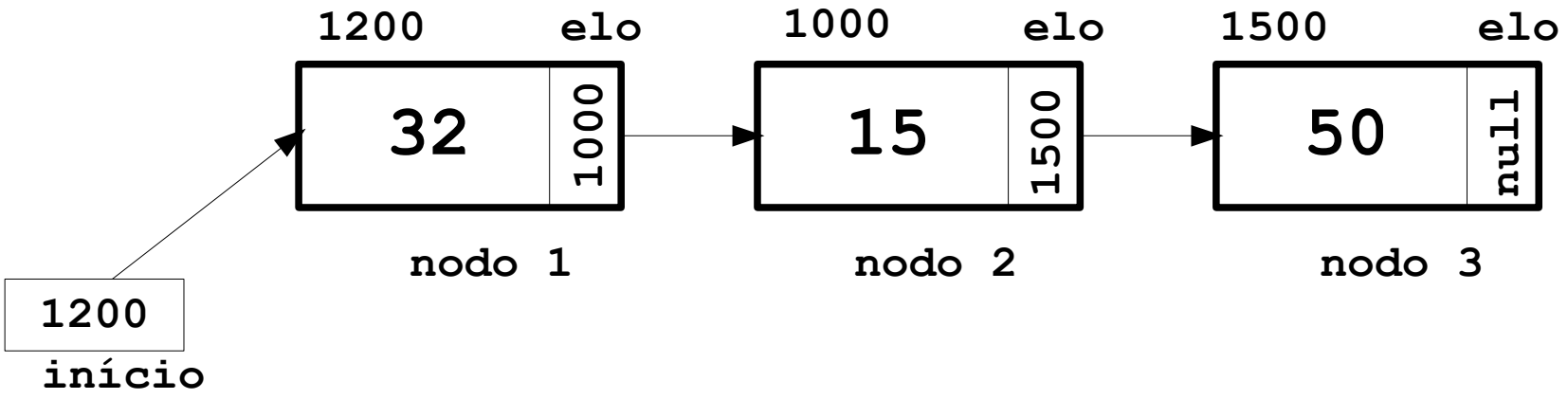




Lista linear

Representação por encadeamento

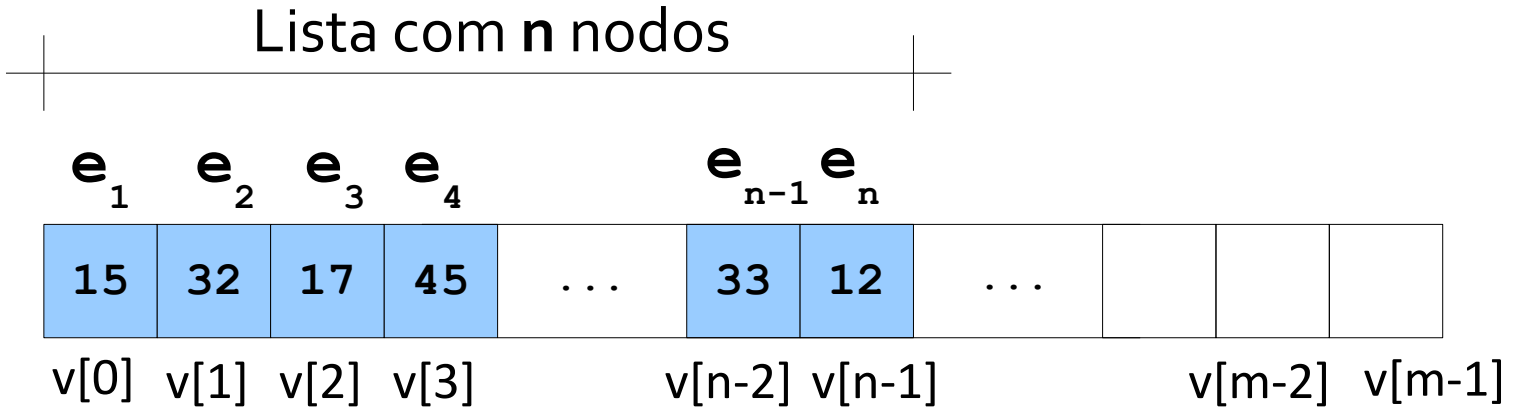
- A disposição física dos nodos independe de sua posição na estrutura lógica.
- Os relacionamentos são representados por elos que são ligações físicas explícitas.
- O valor contido em um campo de elo é o endereço de outro nodo.



LL : 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).

LL : 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   MAX_NODOS  5

typedef struct {
      int idade;
} Dado;

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

LL : Contiguidade física

Algumas operações:

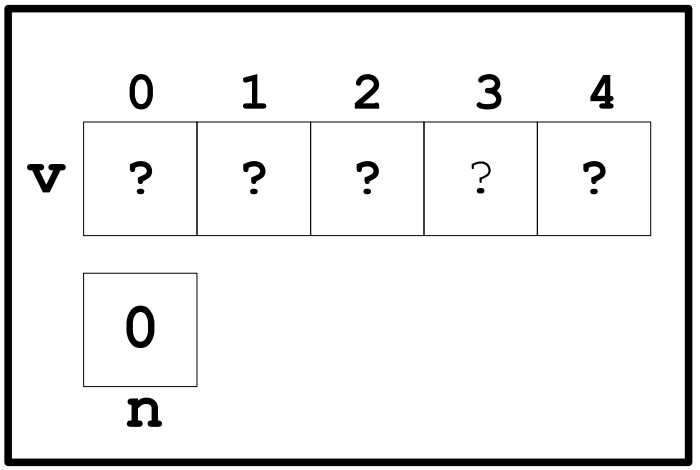
```
void criaLista (ListaCF *lt);
int  incluiNoFim(ListaCF *lt, Dado d);
void exibe(ListaCF lt);
int  excluiDoInicio (ListaCF *lt, Dado *d);
```

LL : Contiguidade física

Cria uma lista

E/S: Uma lista

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



lista

```
int main() {
    ListaCF lista;

    criaLista(&lista);
    return 0;
}
```

LL : Contiguidade física

Incluir no final da lista:

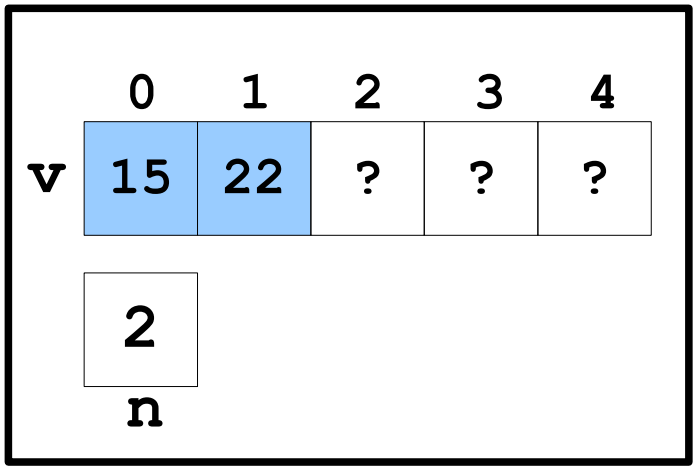
E/S: Uma lista

Entrada: O dado que será incluído.

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

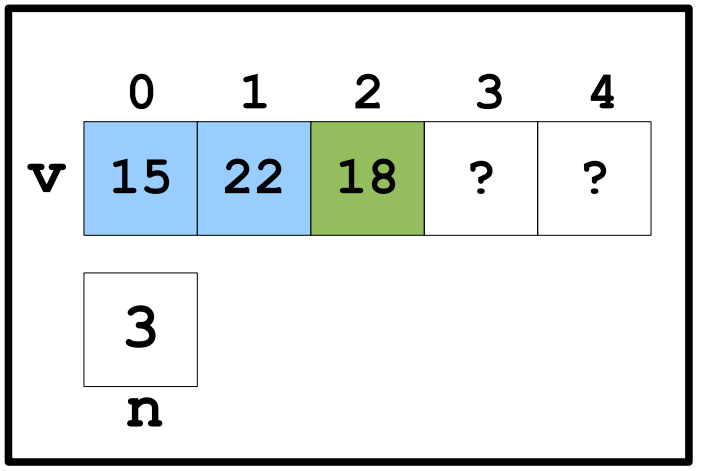
Inclusão do dado: **`idade=18`** em uma lista com 2 nodos

Antes:



`lista`

Depois:



`lista`

LL : Contiguidade física

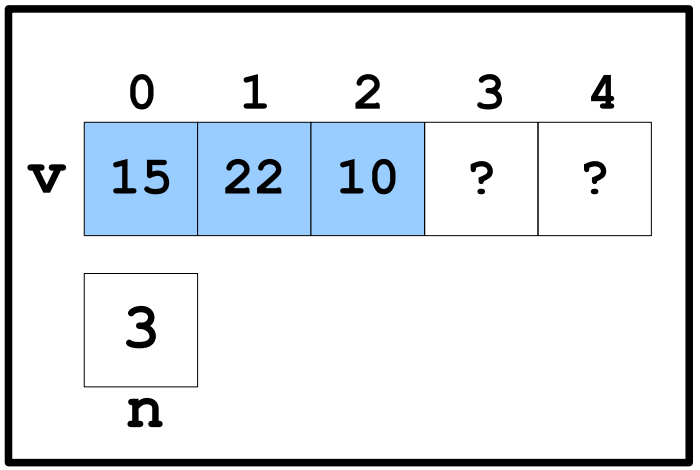
```
...
int incluiNoFim(ListaCF *lt, Dado d) {
   if (lt->n==MAX_NODOS)
      return LISTA_CHEIA;
   else {
      lt->v[lt->n] = d;
      lt->n++;
      return SUCESSO;
   }
}

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;
}
```

LL : Contiguidade física

Exibir o conteúdo da lista

Entrada: Nenhuma.

Retorno: Nenhuma.



lista

```
15
22
10
```

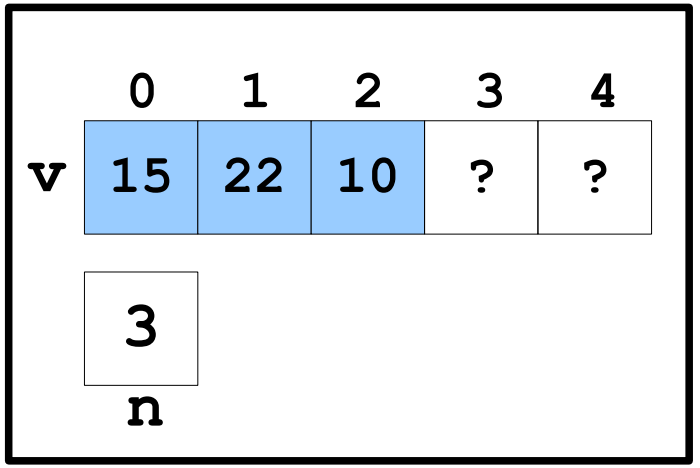
```
...
void exibe(ListaCF lt) {
   int i;
   for(i=0; i<lt.n; i++)
      printf("%d\n",lt.v[i].idade;
}

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;
}
```

LL : Contiguidade física

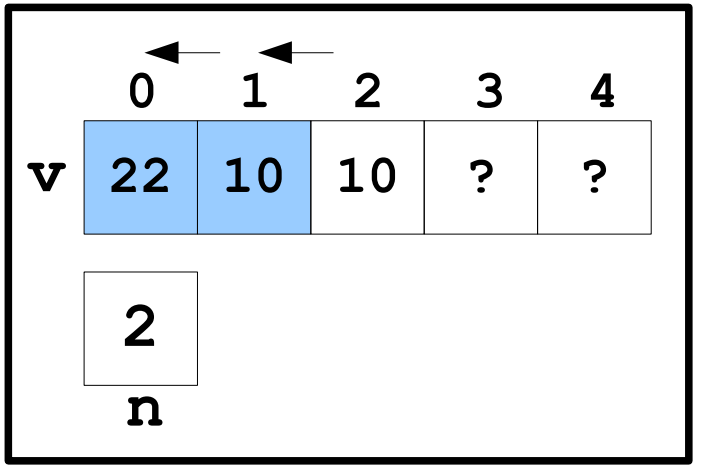
Excluir o nodo que está no início da lista:
Entrada: Nenhuma.
Saída: Nodo excluído.
Retorno: Código de erro **SUCESSO** ou **LISTA_VAZIA** (*Underflow*).

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

Antes:



Depois:



LL : Contiguidade física

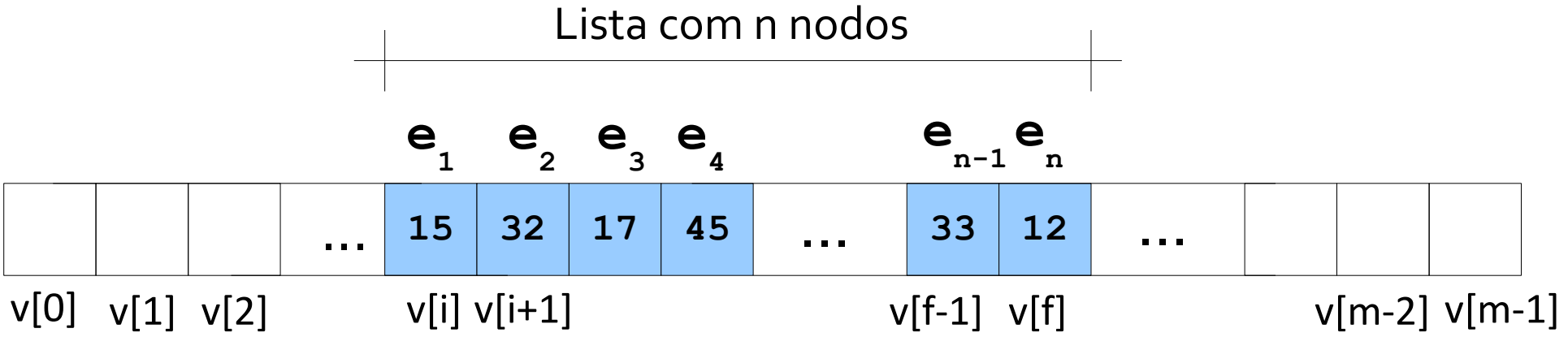
```
...
int excluiDoInicio(ListaCF *lt,Dado *d) {
  int i;
  if (lt->n==0)
     return LISTA_VAZIA;
  else {
     *d = lt->v[0];
     for (i=1; i<lt->n; i++)  // Deslocamento
       lt->v[i-1] = lt->v[i];
     lt->n--;
     return SUCESSO;
  }
}
```

```
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;
}
```

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.

Uma alternativa de representação



i : Índice do primeiro elemento da lista.
f : índice do último elemento da lista.
v : Vetor que armazena os nodos da lista.
m : Capacidade (máxima do vetor).

- Operações de **inclusão** e **exclusão** de nodos podem optar por deslocar para a direita ou esquerda de forma que produza a menor movimentação de nodos.