Algoritmos e Estruturas de Dados I

Profª Priscilla Abreu

priscilla.abreu@ime.uerj.br

2022.1

Algoritmos e Estruturas de Dados I

Roteiro da aula

- Listas lineares
  - Listas Sequenciais
    - Ordenadas

Listas
Revisando...

Algoritmos e Estruturas de Dados I

Listas

O QUE É UMA LISTA?

Consideraremos como listas conjuntos sem repetições!

Uma lista é um conjunto de dados relacionados, e de número variável de elementos.

- Exemplo:
  - Lista de alunos de uma turma;
  - Lista de aprovados em um concurso;
  - Lista de produtos de uma loja;
  - ...

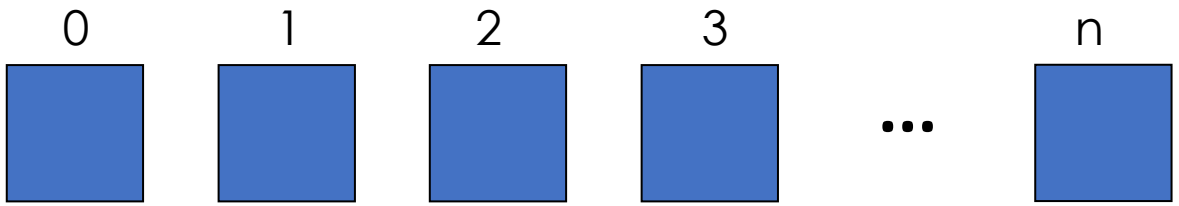
Cada elemento da lista terá uma chave – identificador único.

Algoritmos e Estruturas de Dados I

Listas Linear

Estrutura que permite representar um conjunto de dados de forma a preservar a relação de ordem existente entre eles.

Temos um primeiro elemento, segundo elemento, ..., n-ésimo elemento.



Algoritmos e Estruturas de Dados I

Listas Linear

Listas lineares

Listas lineares gerais
SEM restrição de inserção e remoção de elementos

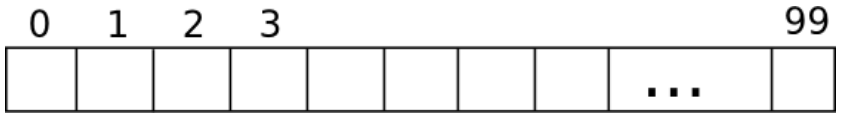
Listas particulares
COM restrição de inserção e remoção de elementos

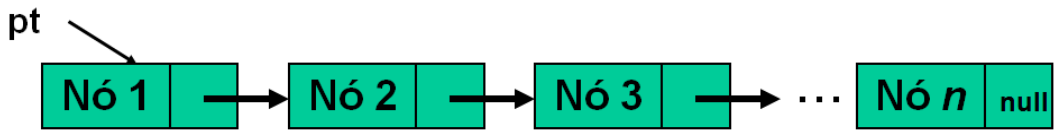
Algoritmos e Estruturas de Dados I

O tipo de armazenamento de uma lista linear pode ser classificado de acordo com a posição relativa na memória (contínua ou não) de cada dois nós consecutivos na lista.

Existem dois tipos de alocação:

- Alocação sequencial
- Alocação encadeada





Algoritmos e Estruturas de Dados I

Implementação de Listas Lineares varia também dependendo da ordem das chaves:

- **Não-Ordenada**: elementos são armazenados em ordem arbitrária, não tendo relação com os valores das respectivas chaves;

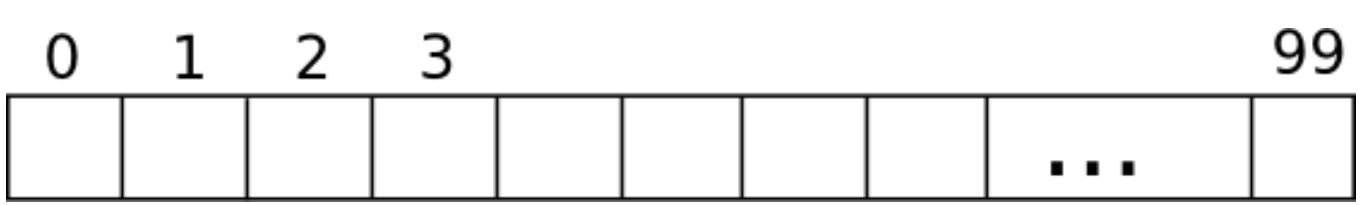
- **Ordenada**: elementos são armazenados de acordo com o valor da chave (em geral, ordem não decrescente).

Listas Lineares Sequenciais Ordenadas

Algoritmos e Estruturas de Dados I

Lista Linear Sequencial

- Uso de vetores.



  - MAX é a quantidade máxima de elementos que a lista poderá armazenar.
  - **n representa o número de elementos.**

Algoritmos e Estruturas de Dados I

Lista Linear Sequencial

estrutura no:
    chave: <inteiro>

estrutura listaSeq:
    valores [1..MAX]: no
    n: inteiro

```
const int MAX = 10
typedef struct no{
    int chave;
}no;

typedef struct listaSeq{
    no valores[MAX];
    int n;
} listaSeq;
```

Algoritmos e Estruturas de Dados I

Operações:

- Busca:
  - Listas ordenadas

Algoritmos e Estruturas de Dados I

Operações:

- Busca – lista ordenada:

```
função BuscaBin(ref L: ListaSeq, c: inteiro): Inteiro
    var inf, sup, m: inteiro
    inf ← 1
    sup ← L.n
        enquanto inf ≤ sup faça
            m ← [(inf + sup)/2)]
            se L.valores[m].chave = c então
                retornar (m)
            senão
                se L.valores[m].chave < c então
                    inf ← m + 1
                senão
                    sup ← m - 1
        retornar (0)
```

Tempo:
Melhor Caso: θ(1)
Pior Caso: θ(lg n)

```
int buscaBin (listaSeq *L, int c) {
    int inf, sup, meio;
    inf = 0;
    sup=(L->n)-1;
    while (inf<= sup){
                meio= floor((inf+sup)/2);
                if (c == L->valores[meio].chave)
                    return meio;
                else
                    if (x > L->valores[meio].chave)
                        inf = meio + 1;
                    else
                        sup= meio -1;
   }
   return -1;
}
```

Inserindo elementos...

Algoritmos e Estruturas de Dados I

Operações:

- Inserção:
  - Listas ordenadas

Algoritmos e Estruturas de Dados I

Operações:

- Inserção:
  - Listas ordenadas

Valor a ser inserido

c

16

L.n : 4

1	2	3	4	5	6	7	8	9	10
15	20	26	38						

Algoritmos e Estruturas de Dados I

Operações:

- Inserção:
  - Listas ordenadas

Valor a ser inserido

c

16

Tem espaço disponível?

L.n : 4

1	2	3	4	5	6	7	8	9	10
15	20	26	38						

Algoritmos e Estruturas de Dados I

Operações:

- Inserção:
  - Listas ordenadas

Valor a ser inserido

c

16

O elemento já está cadastrado?

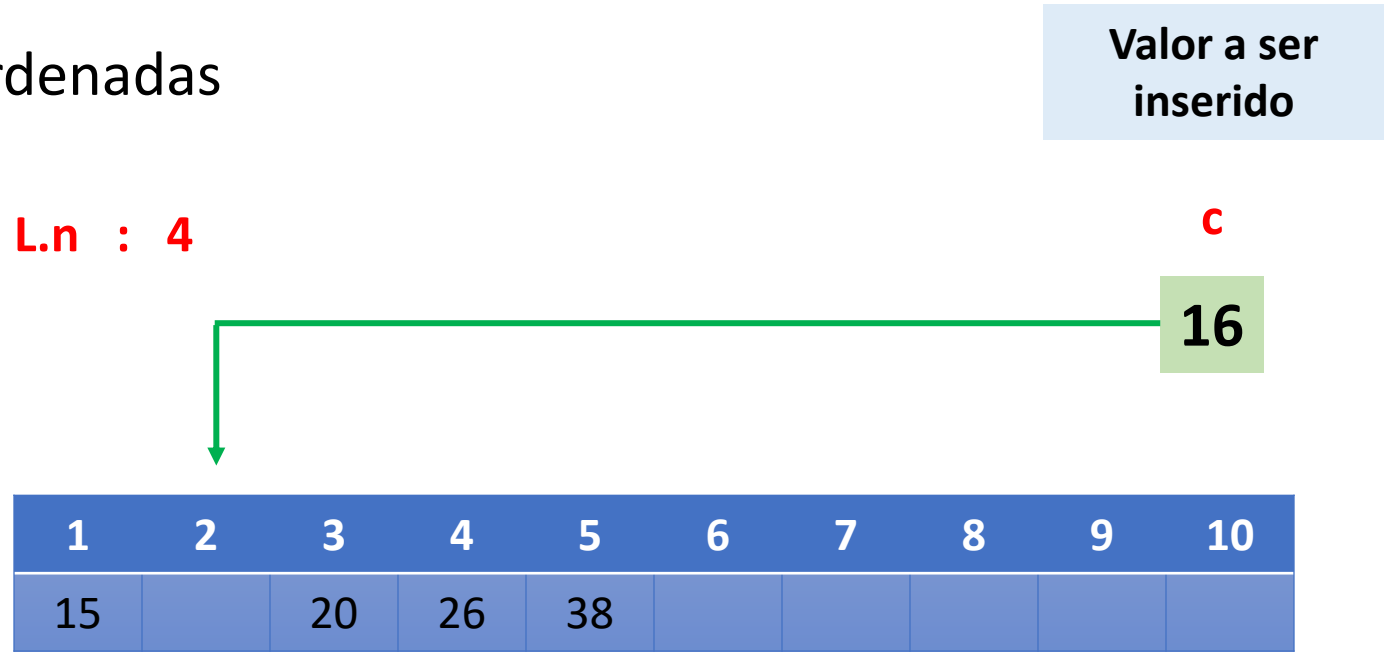
L.n : 4

1	2	3	4	5	6	7	8	9	10
15	20	26	38						

Algoritmos e Estruturas de Dados I

Operações:

- Inserção:
  - Listas ordenadas



Algoritmos e Estruturas de Dados I

Operações:

- Inserção:
  - Listas ordenadas

Valor a ser inserido

i

4

L.n : 4

c

16

1	2	3	4	5	6	7	8	9	10
15	20	26	38						

Algoritmos e Estruturas de Dados I

Operações:

- Inserção:
  - Listas ordenadas

Valor a ser inserido

c

16

i

4

L.n : 4

1	2	3	4	5	6	7	8	9	10
15	20	26	38	38					

Algoritmos e Estruturas de Dados I

Operações:

- Inserção:
  - Listas ordenadas

Valor a ser inserido

c

16

i

3

L.n : 4

1	2	3	4	5	6	7	8	9	10
15	20	26	26	38					

Algoritmos e Estruturas de Dados I

Operações:

- Inserção:
  - Listas ordenadas

Valor a ser inserido

i

2

L.n : 4

c

16

1	2	3	4	5	6	7	8	9	10
15	20	20	26	38					

Algoritmos e Estruturas de Dados I

Operações:

- Inserção:
  - Listas ordenadas

Valor a ser inserido

i

1

L.n : 4

c

16

1	2	3	4	5	6	7	8	9	10
15	16	20	26	38					

Algoritmos e Estruturas de Dados I

Operações:

- Inserção:
  - Listas ordenadas

L.valores[i+1] = chave

L.n : 5

c

16

1	2	3	4	5	6	7	8	9	10
15	16	20	26	38					

Algoritmos e Estruturas de Dados I

Operações:

- Inserção – lista não-ordenada:

```
procedimento insere(ref L: ListaSeq, c: inteiro)
    se L.n < MAX então
        se buscaBin(L, c) = -1 então
            i ← n
            enquanto (L.valores[i].chave > c e i>0) faça
                L.valores[i+1].chave ← L.valores[i].chave
                i ← i - 1
            L.valores[i+1].chave := c
            n := n+1
        senão
            escreva(“elemento já existe na lista”)
    senão
        escreva("Overflow")
```

Tempo:
Melhor caso: θ(lg n)
Pior caso: θ(n)

```
void inserir(listaSeq *L, int c){
    int i;
    if (L->n < MAX) {
            if (buscaBin(L,c)==-1){
                i = (L->n) -1;
                while (L->valores[i].chave > c && i>-1){
                    L->valores[i+1].chave= L->valores[i].chave;
                    i--;
                }
                L->valores[i+1].chave = c;
                L->n = L->n + 1;
            }
            else
                printf(“Elemento já cadastrado!”);
    }
    else
            printf("\nLista cheia!\n");
}
```

Removendo elementos...

Algoritmos e Estruturas de Dados I

Operações:

- Remoção:
  - Listas ordenadas

Algoritmos e Estruturas de Dados I

Operações:

- Remoção:

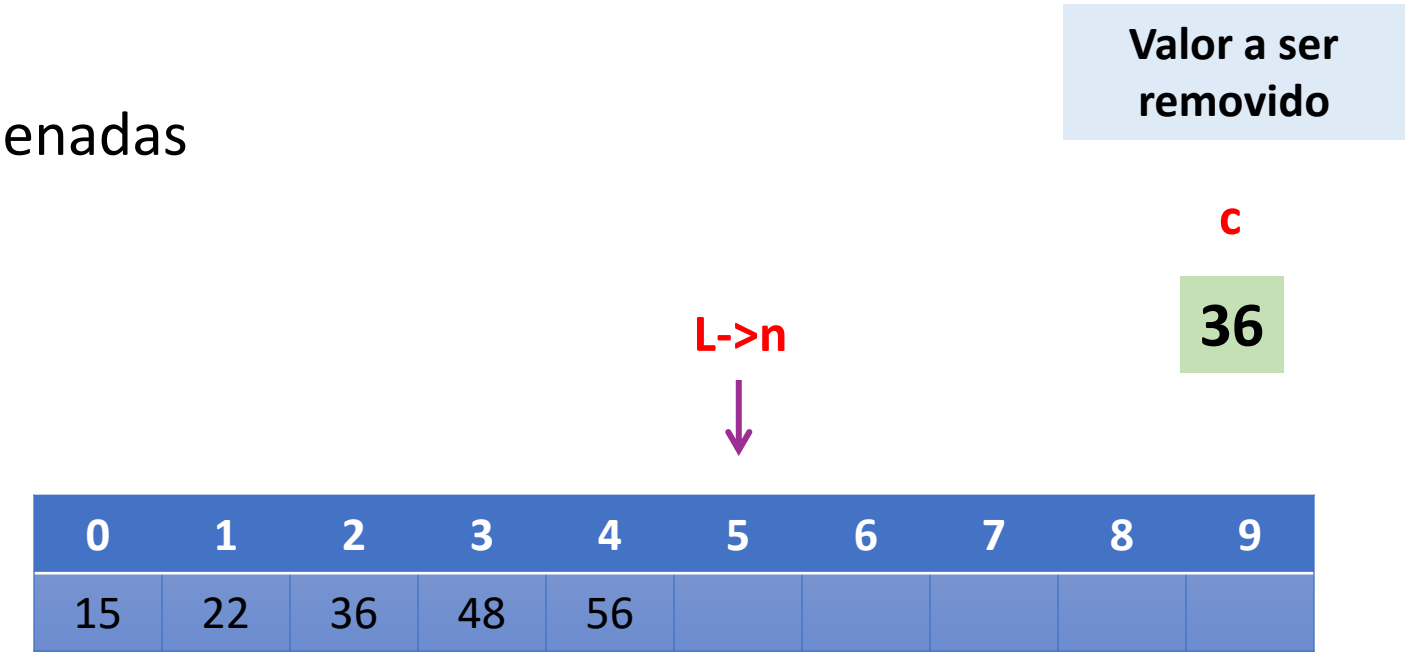
Remoção não ocorre de fato

simulação

Algoritmos e Estruturas de Dados I

Operações:

- Remoção:
  - Listas ordenadas



Algoritmos e Estruturas de Dados I

Operações:

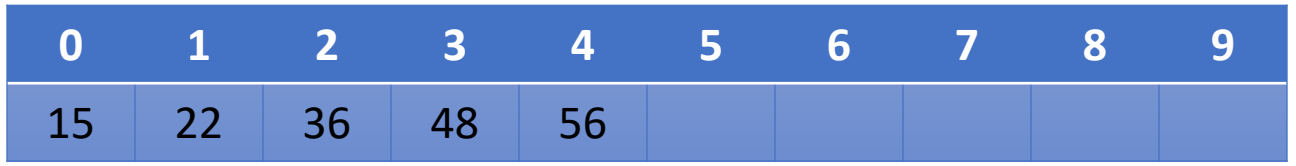
- Remoção:
  - Listas não-ordenadas

A lista está vazia?

c

36

L->n



Algoritmos e Estruturas de Dados I

Operações:

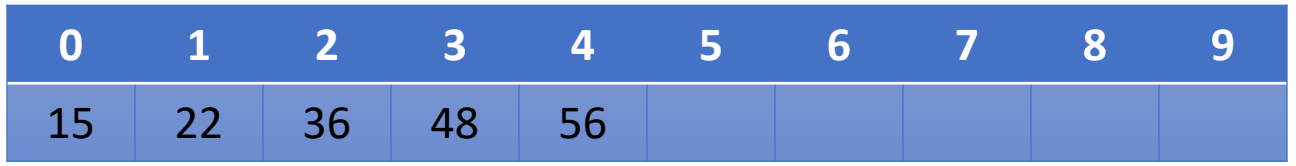
- Remoção:
  - Listas não-ordenadas

O valor a ser removido encontra-se na lista?

c

36

L->n

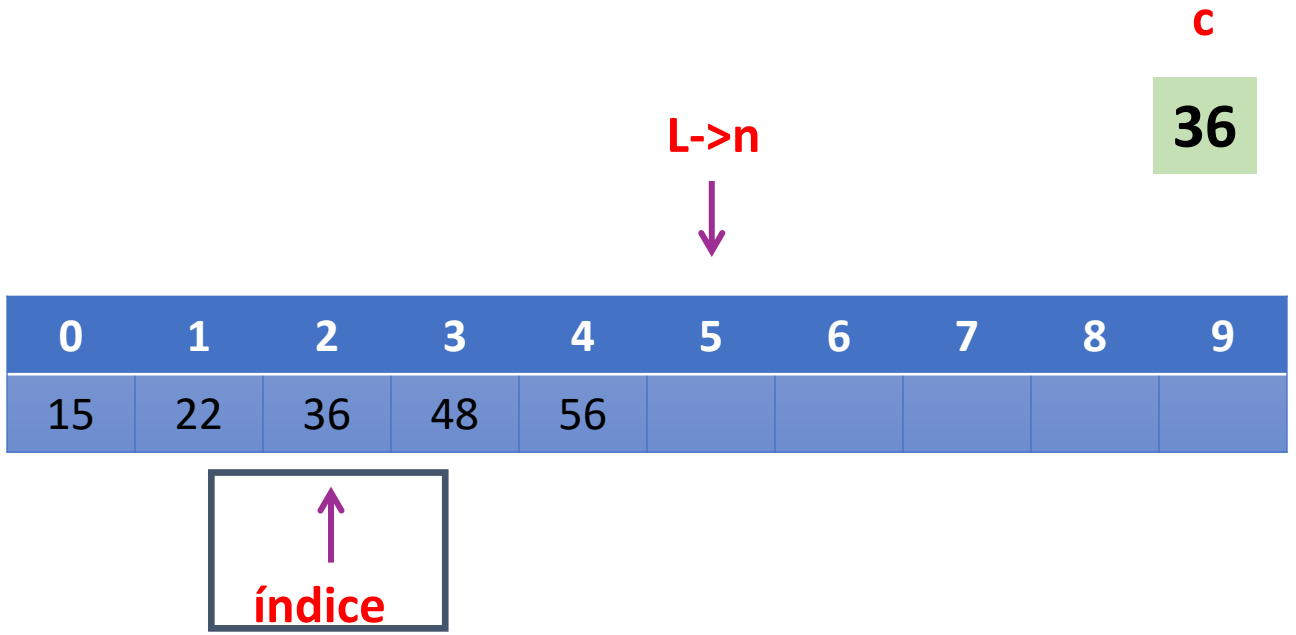


Algoritmos e Estruturas de Dados I

Operações:

- Remoção:
  - Listas não-ordenadas

O valor a ser removido encontra-se na lista?

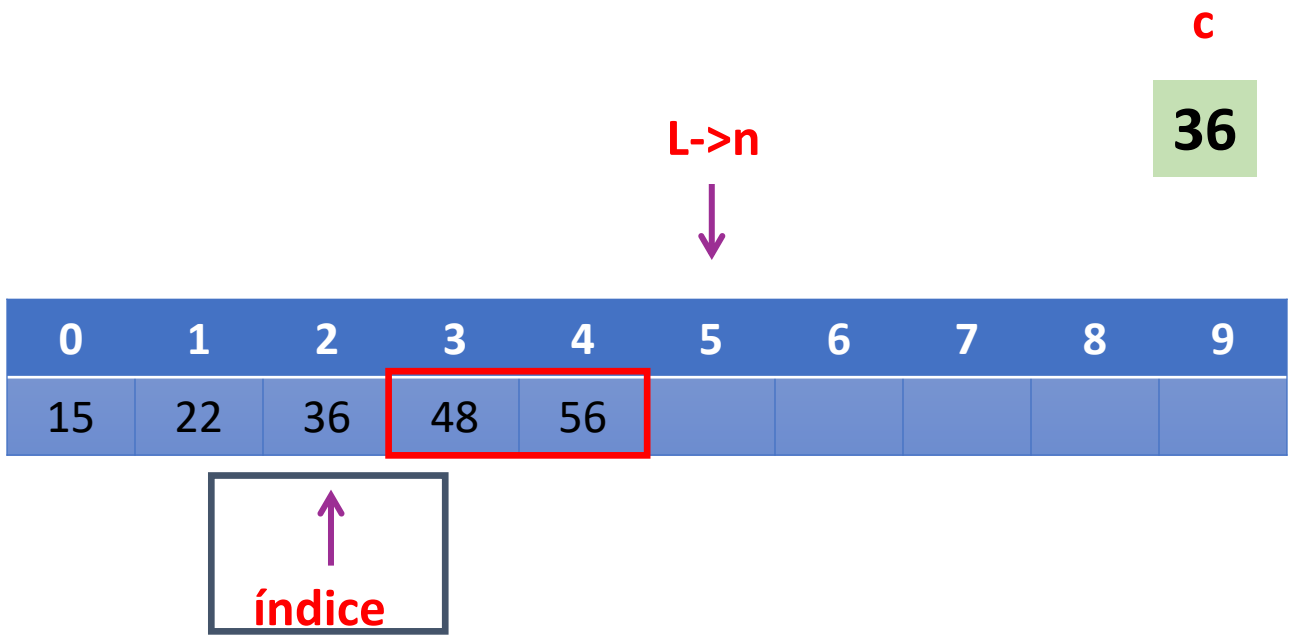


Algoritmos e Estruturas de Dados I

Operações:

- Remoção:
  - Listas não-ordenadas

Vamos arrastar à esquerda os elementos que estão depois do que será removido.

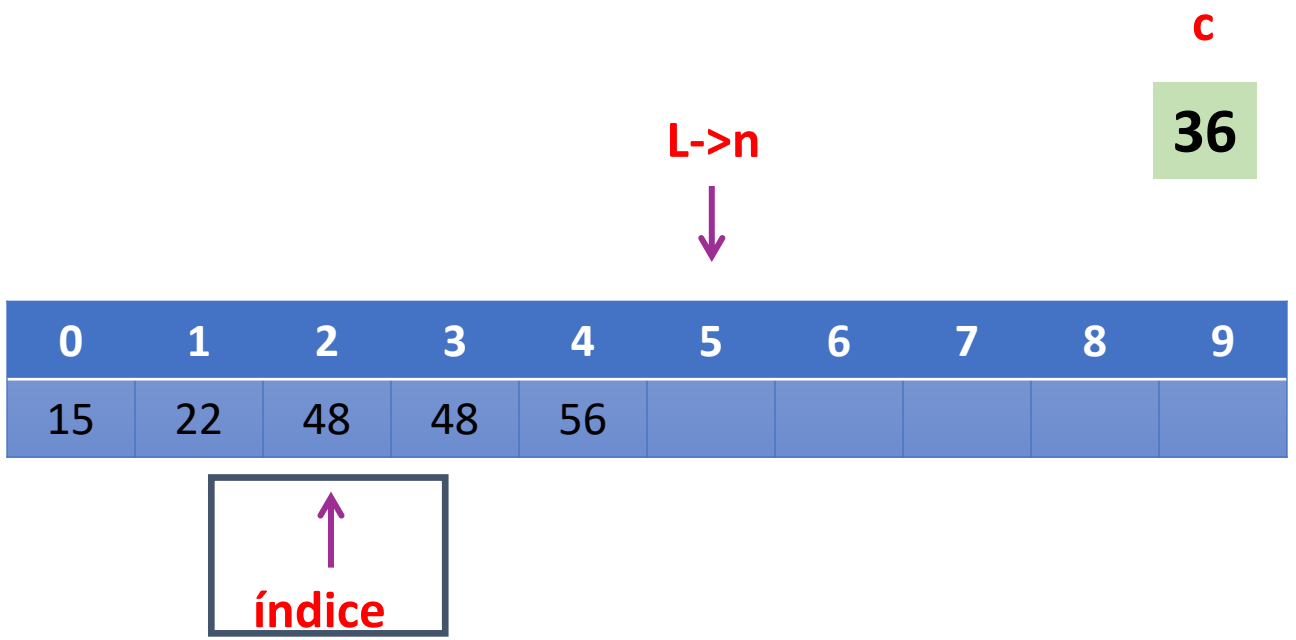


Algoritmos e Estruturas de Dados I

Operações:

- Remoção:
  - Listas não-ordenadas

Vamos arrastar à esquerda os elementos que estão depois do que será removido.

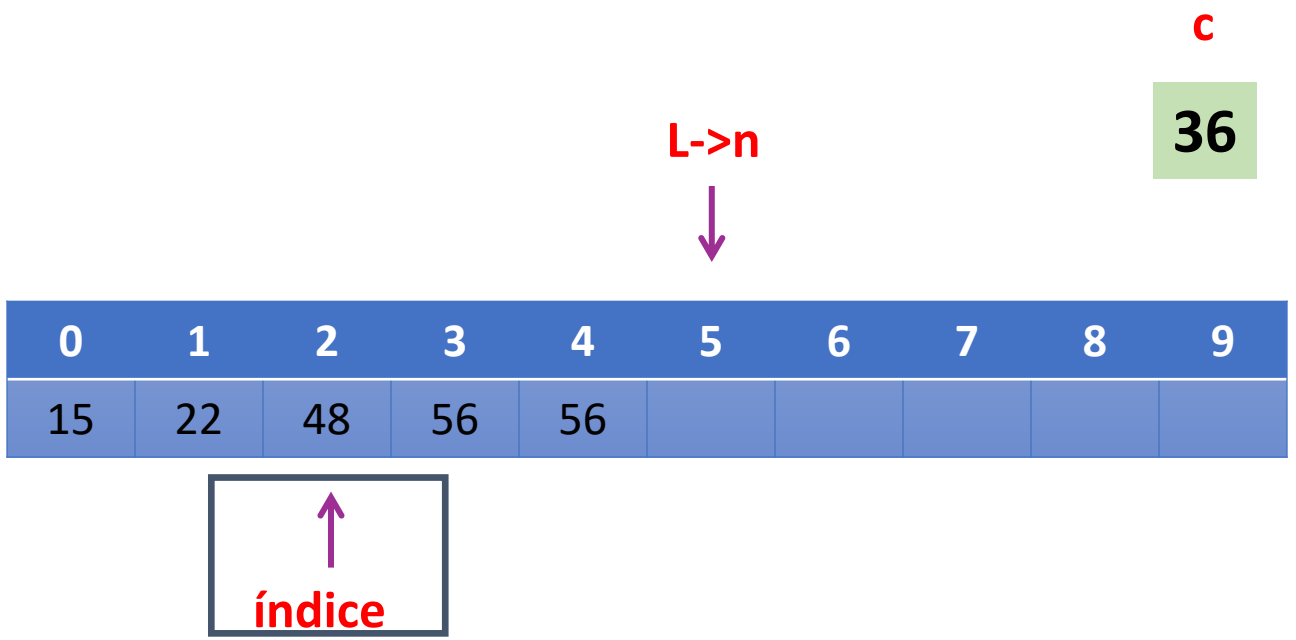


Algoritmos e Estruturas de Dados I

Operações:

- Remoção:
  - Listas não-ordenadas

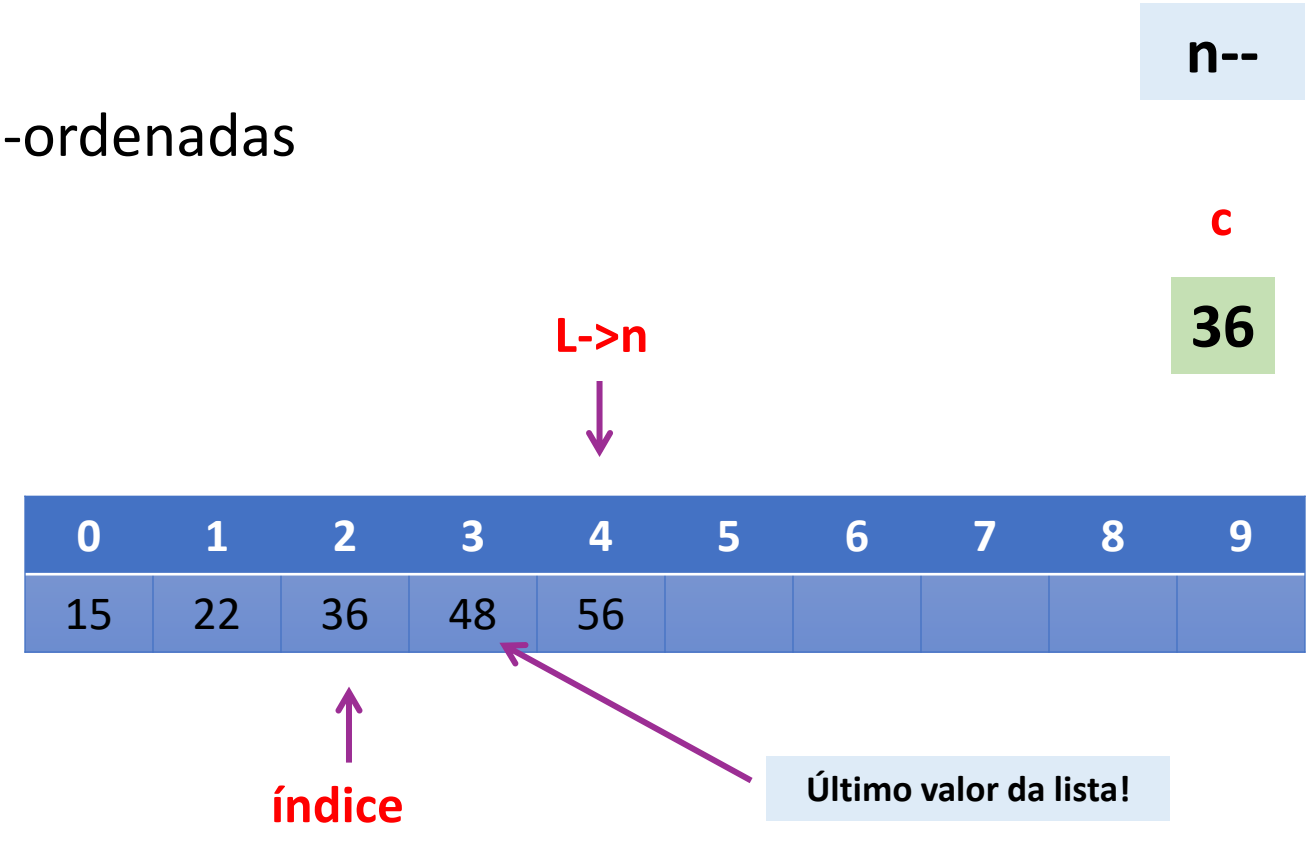
Vamos arrastar à esquerda os elementos que estão depois do que será removido.



Algoritmos e Estruturas de Dados I

Operações:

- Remoção:
  - Listas não-ordenadas



Algoritmos e Estruturas de Dados I

Operações:

- Remoção – lista ordenada:

```
procedimento remove(ref L: ListaSeq, c: inteiro)
    i, índice: inteiro
    se L.n > 1 então
        indice ← buscaPosicao(L,c);
        se (indice ≠ 0) então
            L.valores[indice] ← L.valores[L.n]
            L.n  ←  L.n -1
        senão
            escreva(“Elemento não existente!”)
    senão
        escreva(“Lista vazia")
```

Tempo:
Melhor caso: θ(lg n)
Pior caso: θ(n)

```
void remover(listaSeq *L, int c){
    int i, indice;
    if (L->n > 0) {
        indice= buscaBin(L,c);
        if (indice != -1){
            for (i=indice; i < (L->n)-1; i++)
                L->valores[i].chave=L->valores[i+1].chave;
            (L->n)--;
        }
        else
            printf("\nElemento não existe\n");
    }
    else
        printf("\nLista vazia!\n");
}
```

Algoritmos e Estruturas de Dados I

Exercício

Utilizando as funcionalidades trabalhadas sobre listas sequenciais, faça um programa para gerenciar os dados dos alunos de uma turma. A capacidade máxima de alunos é 30 e os dados a serem armazenados são: matrícula, nome, cpf, telefone. Você deve apresentar um menu de opções com as funcionalidades disponibilizadas. Considere que a lista não armazena os dados de modo ordenado.

Exemplo:

*** Sistema de alunos ***

1- Cadastrar aluno

2- Buscar aluno pela matrícula

3- Exibir os dados dos alunos

4- Remover um aluno

5- Sair

FIM