



Universidade Federal do ABC
Centro de Matemática, Computação e Cognição

Listas

Monael Pinheiro Ribeiro, D.Sc.

Listas

- Definição

- **lista**, s.f. 1. Catalogo, relação, rol. 2. Listra. 3. Tira comprida e estreita de pano ou papel. 4. Relação de nomes de pessoas ou coisas; listagem. 5. Cédula de votação. 6. Cardápio, menu
- **lista** s. f. 1. Tira (estreita e comprida). 2. Risca (em tecido) de cor diferente do fundo. 3. Galão ou fita que adorna uma peça do fato ou do uniforme. 4. Beta. 5. Série escrita de nomes de pessoas ou de coisas. 6. Relação dos prémios da lotaria. 7. Papel com o nome de um ou vários candidatos; voto. lista civil: dotação anual de um chefe de Estado. lista de pratos ou de preços: cardápio, menu. lista telefonica: livro com o endereço e o número de telefone dos assinantes de uma companhia telefonica.

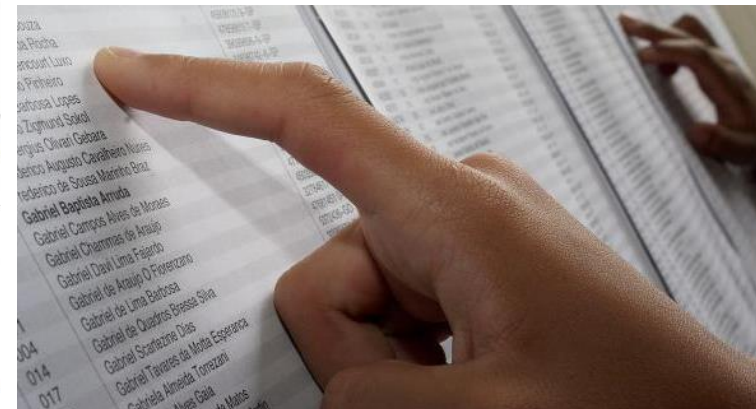
- Exemples:



- 100-BROTH/*Caldo*
- 101-COXINH OF MEAT/*Coxinha de Carne*
- 102-COXINH OF CHICKEN/*Coxinha de Frango*
- 103-COXINH OF CHICKEN CATUPIRY/*Coxinha de Frango Catupiry*
- 104-CROISSANT OF CHICKEN/*Croissant de Frango*
- 105-MIXING CROISSANT/*Croissant Misto*
- 106-EMPADA OF CHICKEN/*Empada de Frango*
- 107-TURNED PAGES MIXING/*Folheado Misto*
- 108-ROLLED OF CHEESE/*Enroladinho de Queijo*
- 109-ROLLED MIXING/*Enroladinho Misto*
- 110-ROLLED SAUSAGE/*Enroladinho Salsicha*
- 111-ESFIRRA OF MEAT/*Esfirra de Carne*
- 112-ESFIRRA OF CHICKEN/*Esfirra de Frango*
- 113-TURNED PAGES OF CHICKEN/*Folheado de Frango*
- 114-TURNED PAGES OF CHEESE/*Folheado de Queijo*
- 115-KIBE/*Kibe*
- 116-STOCKING LASANHA/*Meia Lasanha*
- 117-HALF CANELONE/*Meio Canelone*
- 119-MINI CALZONE/*Mini Calzone*
- 121-BREAD WITH CHEESE/*Pão com Queijo*
- 122-BREAD WITH CHEESE/*Pão de Queijo*
- 123-ITALIAN BREAD/*Pão Italiano*
- 124-MIXING BREAD/*Pão Misto*
- 125-PIZZA (PORTION)/*Pizza (porção)*
- 126-RISOLE OF MEAT/*Risole de Carne*
- 127-RISOLE OF CHICKEN/*Risole de Frango*
- 128-NATURAL SANDWICH/*Sanduche Natural*
- 129-TUNA PIE/*Torta de Atum*
- 130-CHICKEN PIE/*Torta de Frango*
- 131-PIE OF CHICKEN CATUPIRY/*Torta de Frango Catupiry*
- 134-CRAYON/*Paetel*
- 456-MIXING SANDWICH/*Sanduche misto*



LISTA DE PRESEÇA - CONSULTA PÚBLICA 01/SMP/ATSI/2006 - Sistema de Gerenciamento da Fiscalização - SGF - 25/04/2006					
REPRESENTANTE	CARGO	EMPRESA	TELEFONE	E-MAIL	ASSINATURA
MARCELO A. J. SILVA	DELEGADO	TECNOLOGIA	3053.7431	marcelo@tecnologia.com.br	[Assinatura]
MARCO MUSIC	delegado	PROACOL	5504.0003	marco@proacol.com.br	[Assinatura]
FELIPE A. MARIANO JR.	advogado	EMULIT	9251.1215	felipe@emulit.com.br	[Assinatura]
MARCELO A. MARINI	advogado	MMP	3401.020	marcelo@mmp.com.br	[Assinatura]
CARLOS ALBERTO M. MOISÉS	PROCURADOR	IRISA	9124.8023	carlos@irisa.com.br	[Assinatura]
ALEXANDRE S. DRAZVICKAS	diretor	OPERATECH	3186.6191	alexandre@operatech.com.br	[Assinatura]
SERGIO NUNES	Gerente	Sinepro	9181.8055	sergio@sinopro.com.br	[Assinatura]
SILVANO MELLO	SECURITY	VIVO EMPRESAS	9384.1821	silvano@vivoempresas.com.br	[Assinatura]
PAULO MAGALHÃES	Gerente	ATLUS	9493.2241	paulo@atlus.com.br	[Assinatura]
RICARDO GONÇALVES	superintendente	TOLLIS	5504.9134	ricardo@tollis.com.br	[Assinatura]
Alfonso A. S. Pereira	Gerente	SAP	3150.7400	alfonso@sap.com.br	[Assinatura]
RICHARDO ALBANO	advogado técnico	Terra m	3194.1919	richardo@terra.com.br	[Assinatura]
JOÃO WALTER	Gerente-Suplente	CapTelnet	2165.6516	joao@captelnet.com.br	[Assinatura]
MARIO GUERREIRO	Gerente	HEXUSOLUÇÃO	3068.8216	mario@hexusolucao.com.br	[Assinatura]
LUIS CARLOS MURIEL	Diretor Executivo	Sinart Denada	8184.1009	luis@sinartdenada.com.br	[Assinatura]
PAULO CARANI	advogado	Miguellet	5504.2408	paulo@miguellet.com.br	[Assinatura]
CARLOS MARQUES	Consultor	TECNOCONSULTING	3102.2114	carlos@tecnoconsulting.com.br	[Assinatura]
GUILLERMO RAMALHO	Diretor executivo	SPRING WIRELESS	3478.2121	guillermo@springwireless.com.br	[Assinatura]
ALFREDO GARCIA LIMA	Diretor executivo	WORLDWIDE TELECOM	3470.7718	alfredo@worldwidetel.com.br	[Assinatura]
ALVARO BERNARDI	advogado	WORLDWIDE TELECOM	338.5711	alvaro@worldwidetel.com.br	[Assinatura]
FABIO GONCALVES JR.	adv. class	Conilium	3182.0663	fabio@conilium.com.br	[Assinatura]
Sérgio Costa	advogado	BS&F	3459.5191	sergio@bsf.com.br	[Assinatura]



Listas

- Uma lista é uma estrutura de dados em que os elementos estão organizados em uma ordem linear. Uma lista pode ser:
 - **Simplesmente Ligada**: A partir de um elemento da lista não se alcança o elemento anterior.
 - **Duplamente Ligada**: A partir de um elemento da lista se alcança o elemento anterior.
 - **Ordenada**: A ordem linear da lista corresponde à ordem das chaves.
 - **Não ordenada**: Os elementos aparecem em qualquer ordem.
 - **Circular**: A partir do primeiro elemento da lista se alcança o último. E a partir do último elemento da lista se alcança o primeiro.
 - **Não circular**: A partir do primeiro não se alcança o último. E do último elemento não se alcança o primeiro.

Listas

- Quanto a implementação, as listas podem ser:
 - **Estáticas**: Os elementos são armazenados em um vetor.



- **Dinâmicas**: Os elementos são alocados dinamicamente conforme necessidade. Cada elemento armazena os dados e um ponteiro para o próximo elemento da lista.



Listas

- Estrutura de Dados do tipo Lista:
 - Lista Estática Sequencial (LES)
 - Lista Estática Encadeada (LEE)
 - Lista Dinâmica Encadeada (LDE)
 - Lista Dinâmica Duplamente Encadeada (LDDE)



Universidade Federal do ABC
Centro de Matemática, Computação e Cognição

Lista Estática Sequencial

Monael Pinheiro Ribeiro, D.Sc.

Lista Estática Sequencial

- Seja:
 - **L** uma lista com **n** elementos e
 - **i** um índice da lista tal que $0 \leq i \leq n-1$
- Características de uma Lista **L** do tipo LES:
 - Os elementos da lista estão ordenados através de um campo chave;
 - São armazenados fisicamente em posições consecutivas;
 - A inserção de um elemento na posição **L[i]** causa o deslocamento a direita do elemento de **L[i]** ao último elemento;
 - A eliminação do elemento **L[i]** requer o deslocamento à esquerda do elemento **L[i+1]** ao último;

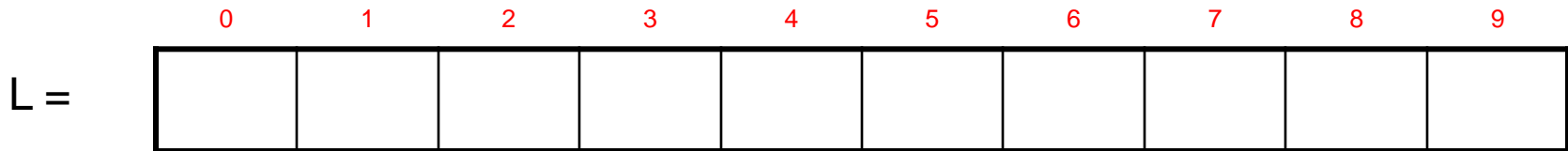
Lista Estática Sequencial

- Operações Básicas
 - Inserção de um elemento na Lista
 - Eliminação de um elemento da Lista
 - Consulta da pertinência de um elemento na Lista

Lista Estática Sequencial

- Funcionamento (Inserção)

Quantidade: 0

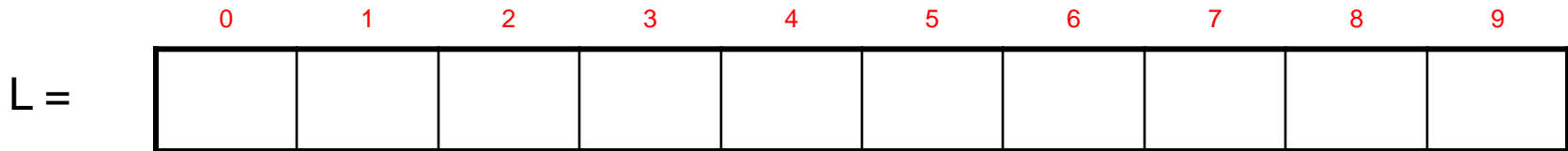


Para fins de simplificação do exemplo de funcionamento, usaremos um vetor de inteiros onde os próprios números inseridos são considerado a chave. Em uma aplicação ter-se-ia um vetor de estruturas com algum membro sendo a chave.

Lista Estática Sequencial

- Funcionamento (Inserção)

Quantidade: 0

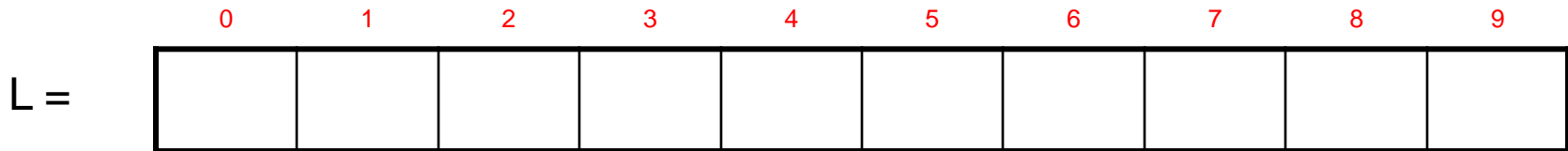


Item a ser inserido: 78

Lista Estática Sequencial

- Funcionamento (Inserção)

Quantidade: 0



Quantidade igual a ZERO, significa que a lista está vazia.

Portanto:

Item a ser inserido: 78

- Inserir o elemento na posição Quantidade
- Incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção)

Quantidade: 1

L =

0	1	2	3	4	5	6	7	8	9
78									

Quantidade igual a ZERO, significa que a lista está vazia.

Portanto:

Item a ser inserido: 78

- Inserir o elemento na posição Quantidade
- Incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção)

Quantidade: 1

L =

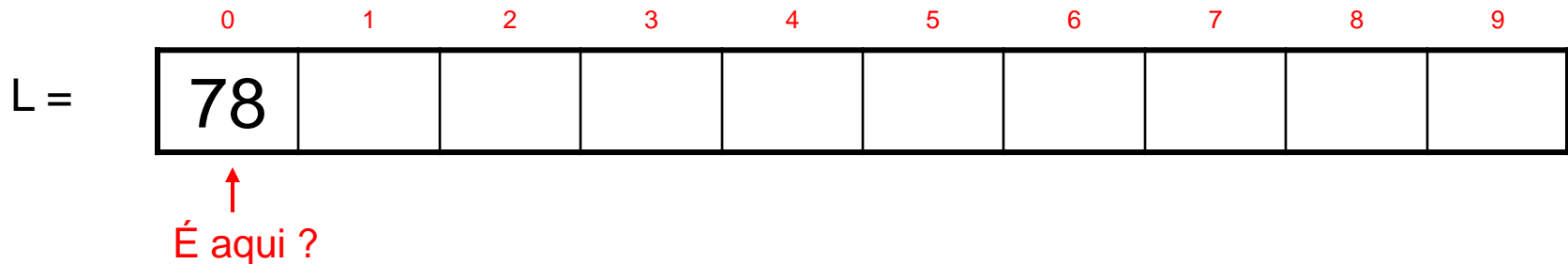
0	1	2	3	4	5	6	7	8	9
78									

Item a ser inserido: 92

Lista Estática Sequencial

- Funcionamento (Inserção)

Quantidade: 1



Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

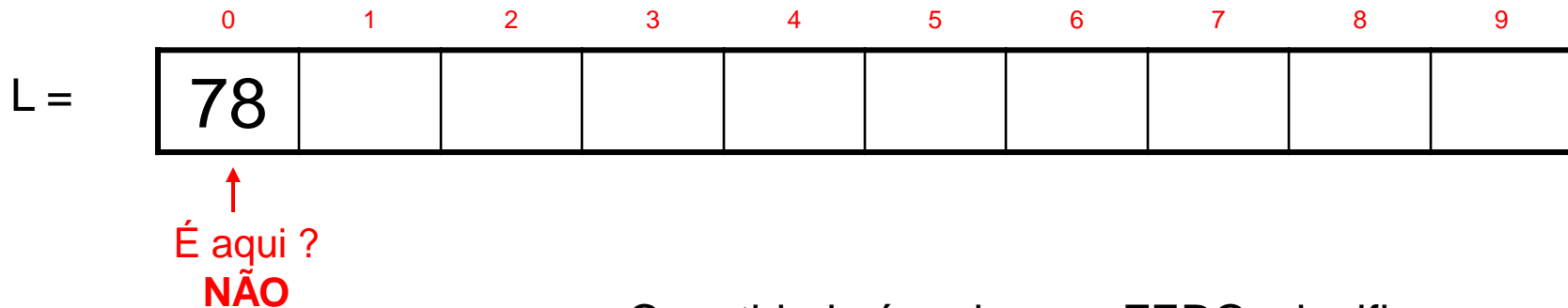
Item a ser inserido: 92

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção)

Quantidade: 1



Item a ser inserido: 92

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção)

Quantidade: 2

L =

0	1	2	3	4	5	6	7	8	9
78	92								

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

Item a ser inserido: 92

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção)

Quantidade: 2

L =

0	1	2	3	4	5	6	7	8	9
78	92								

Item a ser inserido: 81

Lista Estática Sequencial

- Funcionamento (Inserção)

Quantidade: 2

L =

0	1	2	3	4	5	6	7	8	9
78	92								

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

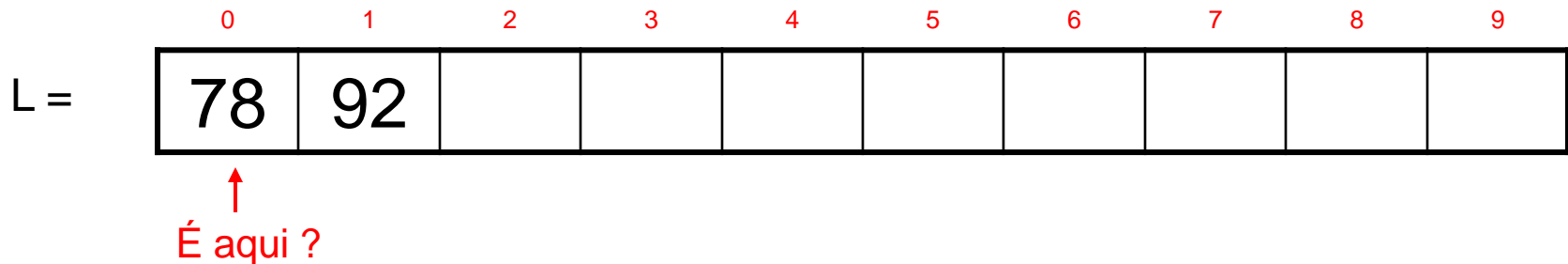
Item a ser inserido: 81

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção)

Quantidade: 2



Item a ser inserido: 81

Quantidade é maior que ZERO, significa que a há itens na lista.

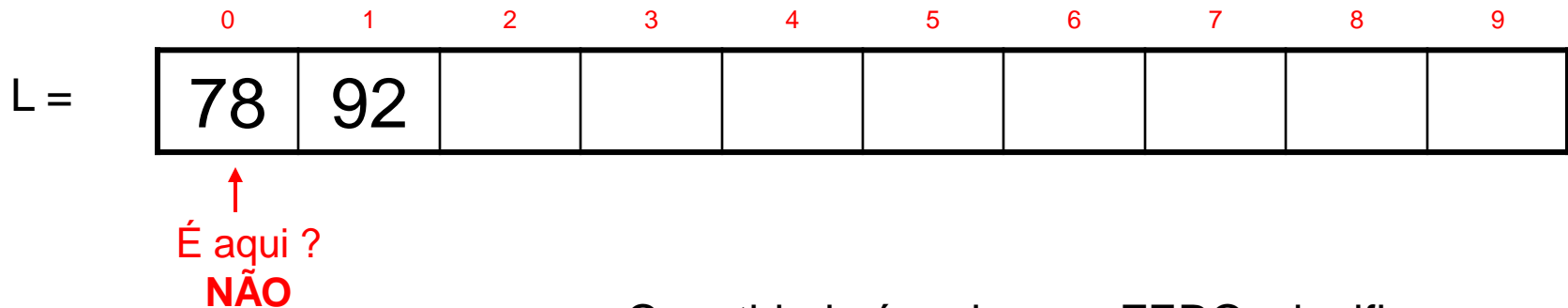
Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção)

Quantidade: 2



Item a ser inserido: 81

Quantidade é maior que ZERO, significa que a há itens na lista.

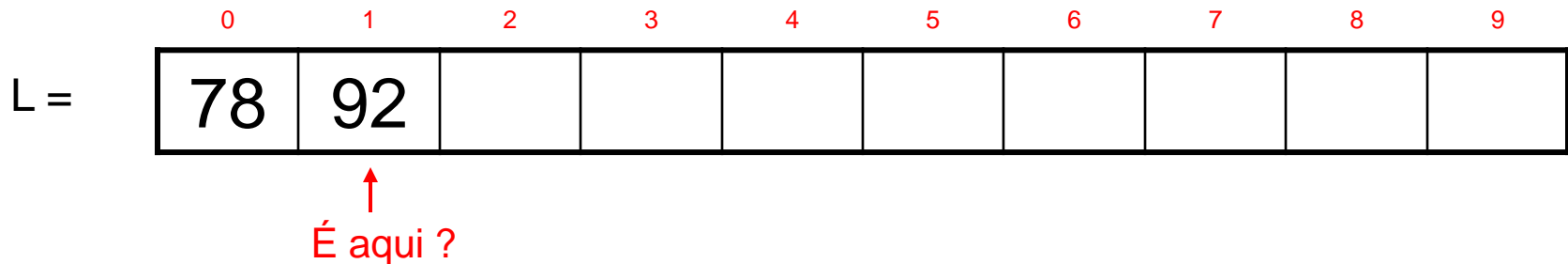
Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção)

Quantidade: 2



Item a ser inserido: 81

Quantidade é maior que ZERO, significa que a há itens na lista.

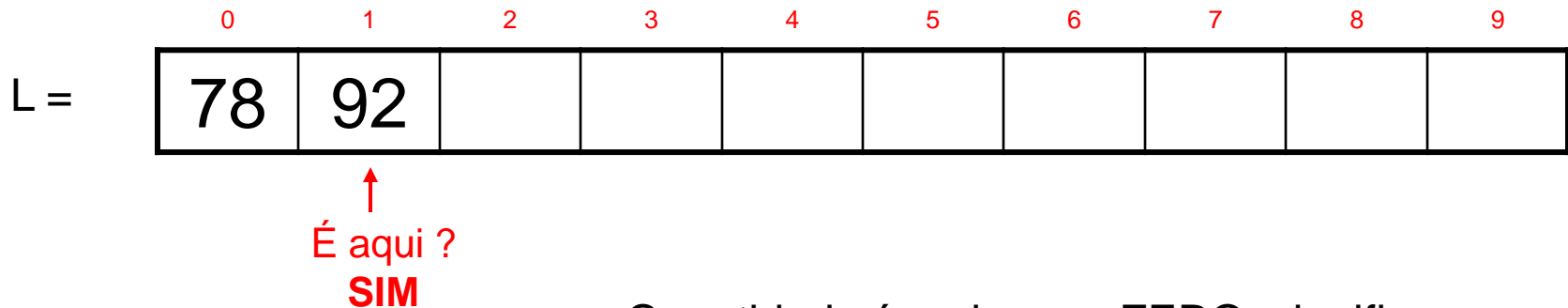
Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção)

Quantidade: 2



Item a ser inserido: 81

Quantidade é maior que ZERO, significa que a há itens na lista.

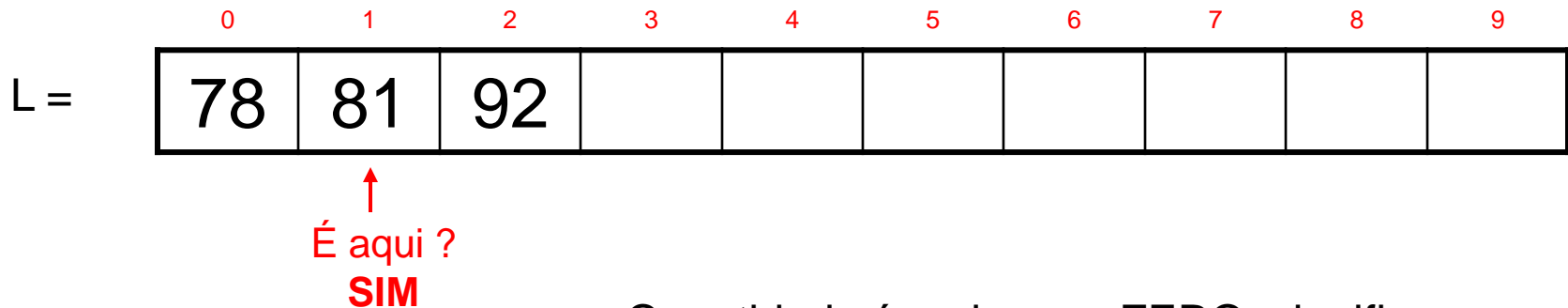
Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção)

Quantidade: 3



Item a ser inserido: 81

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção)

Quantidade: 3

L =

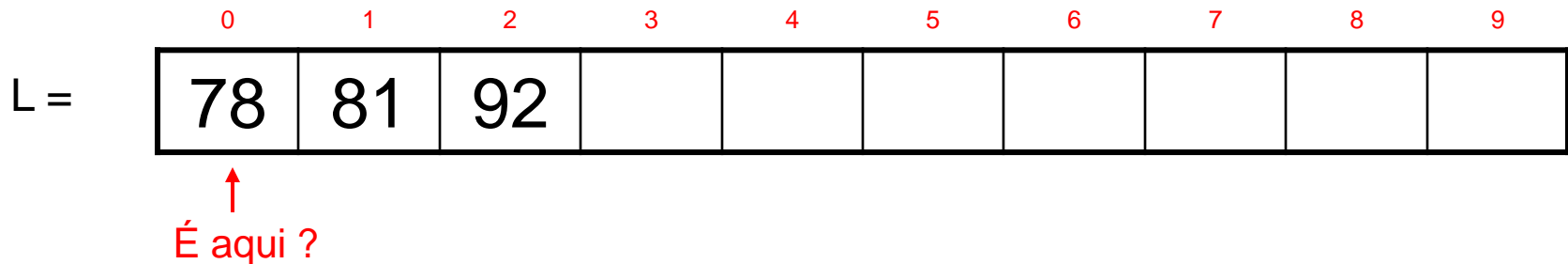
0	1	2	3	4	5	6	7	8	9
78	81	92							

Item a ser inserido: 45

Lista Estática Sequencial

- Funcionamento (Inserção)

Quantidade: 3



Item a ser inserido: 45

Quantidade é maior que ZERO, significa que a há itens na lista.

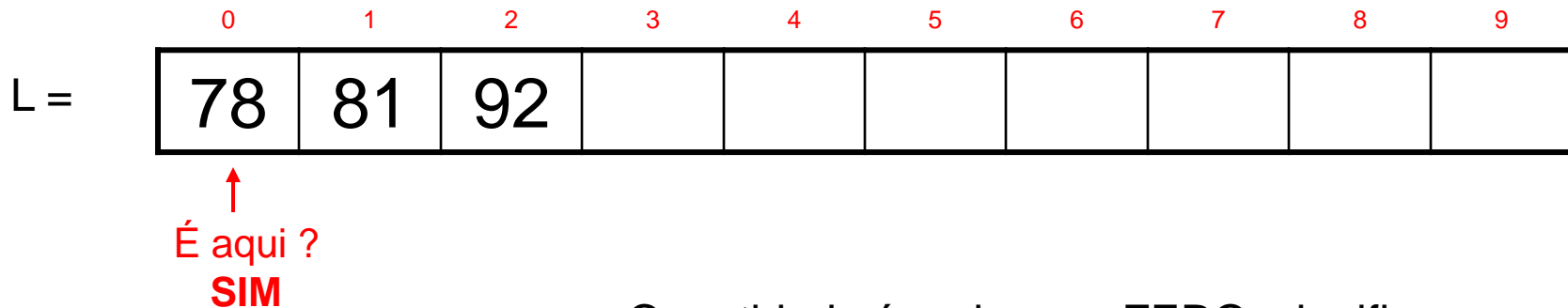
Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção)

Quantidade: 3



Item a ser inserido: 45

Quantidade é maior que ZERO, significa que a há itens na lista.

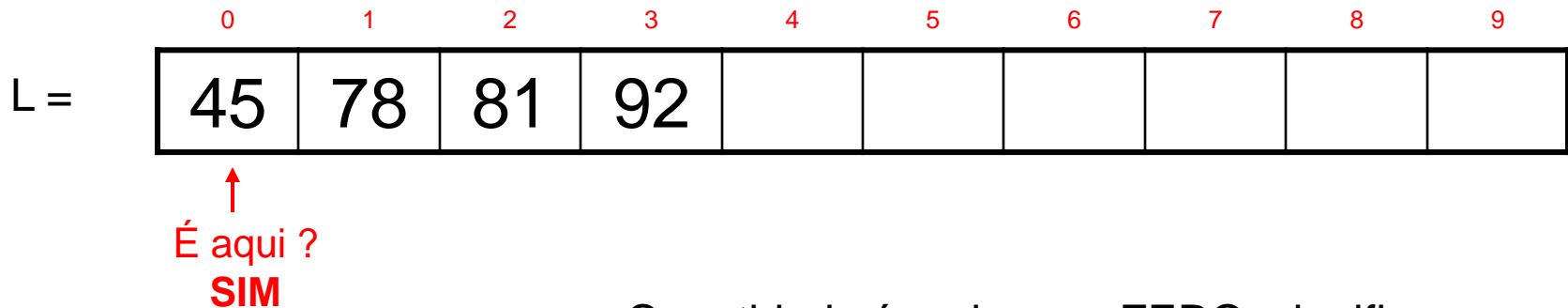
Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção)

Quantidade: 4



Item a ser inserido: 45

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção)
 - Observe cada situação da inserção
 - Inserir no início da Lista
 - Vazia
 - Não Vazia
 - Inserir no meio da Lista
 - Inserir no final da Lista

Lista Estática Sequencial

- Funcionamento (Inserção no início da Lista)

Quantidade: 4

L =

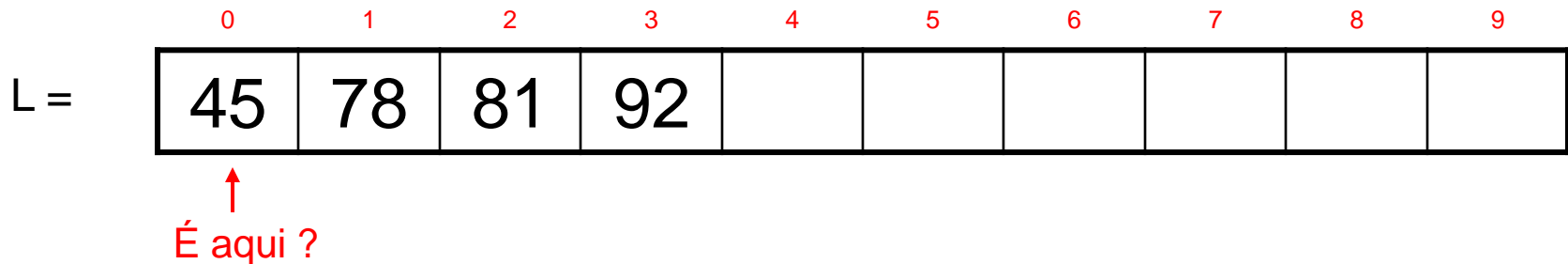
0	1	2	3	4	5	6	7	8	9
45	78	81	92						

Item a ser inserido: 36

Lista Estática Sequencial

- Funcionamento (Inserção no início da Lista)

Quantidade: 4



Item a ser inserido: 36

Quantidade é maior que ZERO, significa que a há itens na lista.

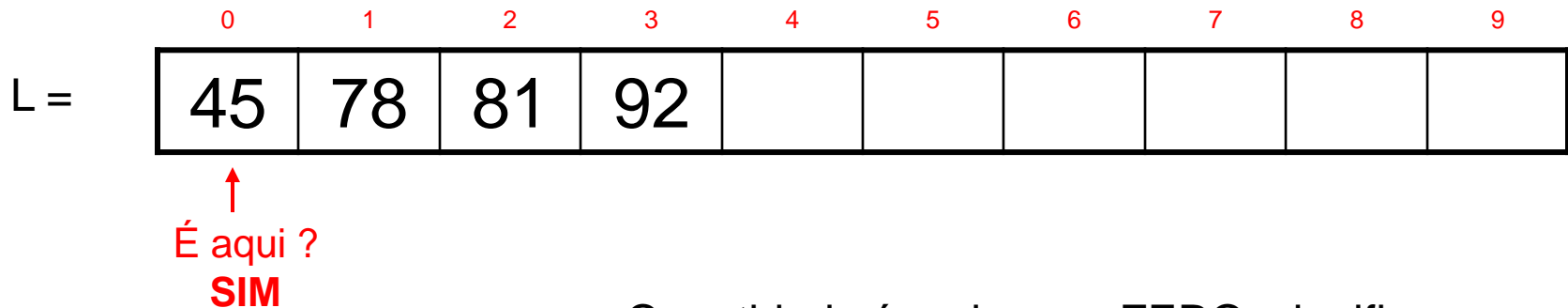
Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no início da Lista)

Quantidade: 4



Item a ser inserido: 36

Quantidade é maior que ZERO, significa que a há itens na lista.

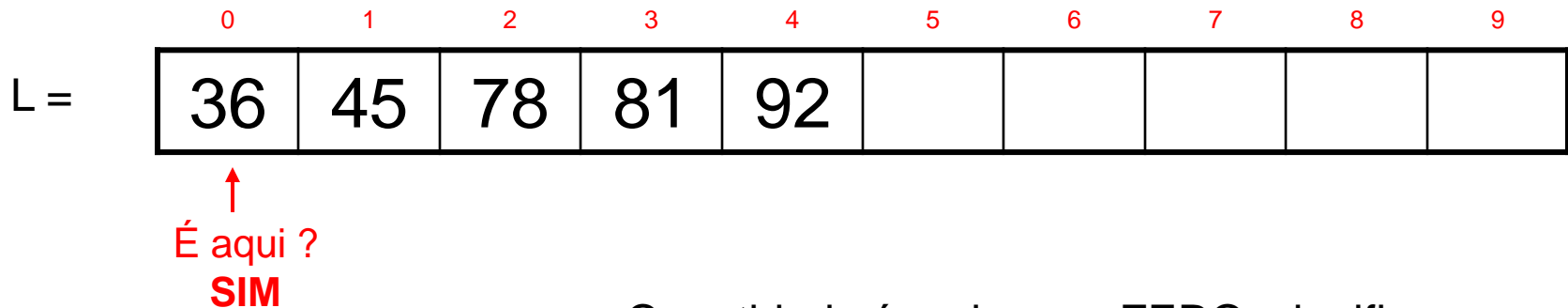
Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no início da Lista)

Quantidade: 5



Item a ser inserido: 36

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no final da Lista)

Quantidade: 5

L =

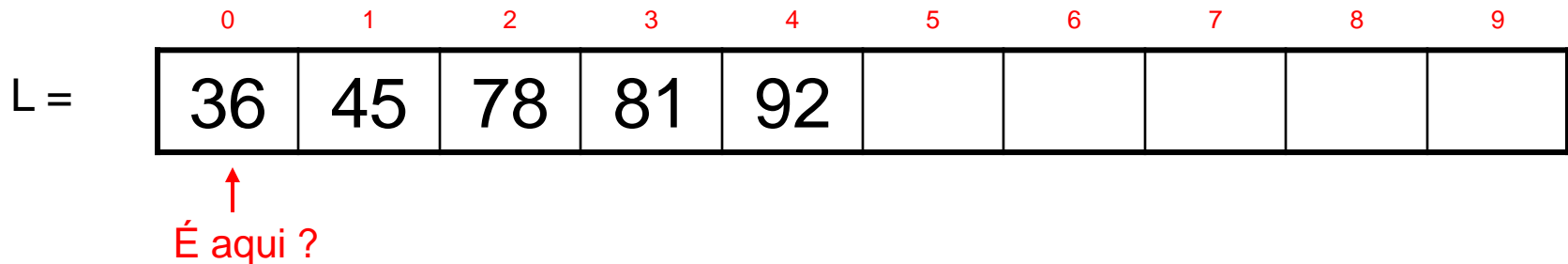
0	1	2	3	4	5	6	7	8	9
36	45	78	81	92					

Item a ser inserido: 98

Lista Estática Sequencial

- Funcionamento (Inserção no final da Lista)

Quantidade: 5



Item a ser inserido: 98

Quantidade é maior que ZERO, significa que a há itens na lista.

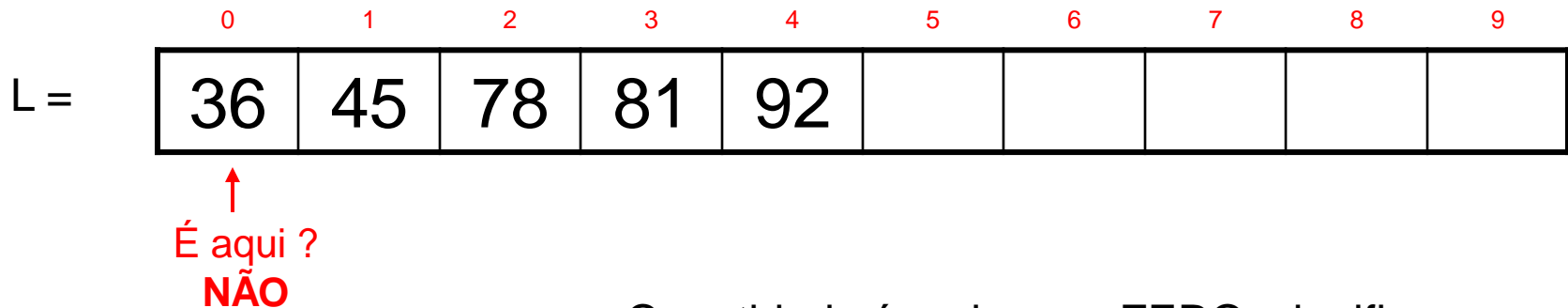
Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no final da Lista)

Quantidade: 5



Item a ser inserido: 98

Quantidade é maior que ZERO, significa que a há itens na lista.

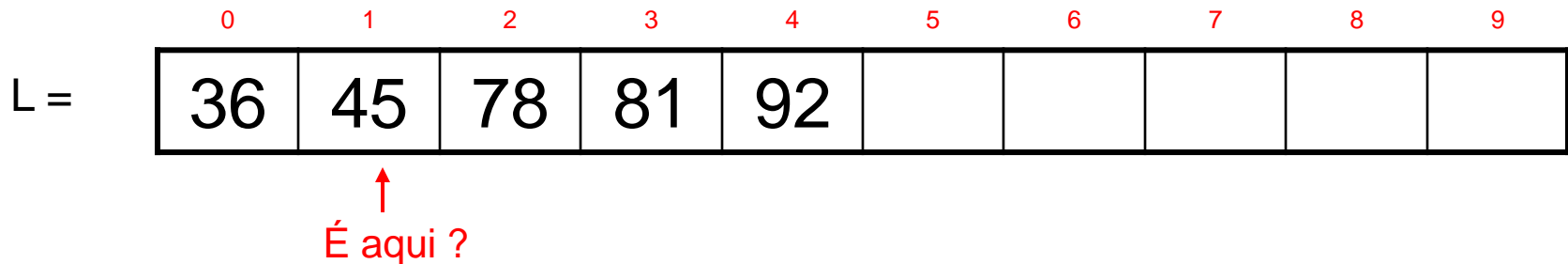
Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no final da Lista)

Quantidade: 5



Item a ser inserido: 98

Quantidade é maior que ZERO, significa que a há itens na lista.

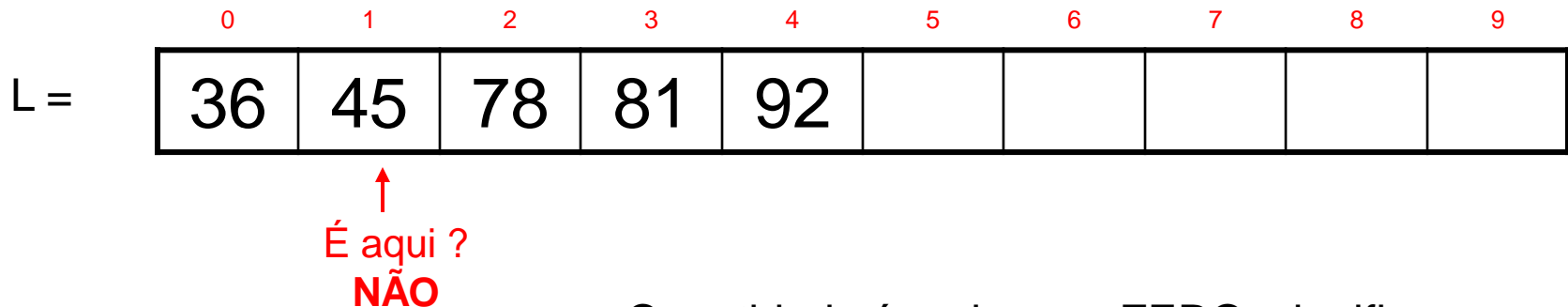
Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no final da Lista)

Quantidade: 5



Item a ser inserido: 98

Quantidade é maior que ZERO, significa que a há itens na lista.

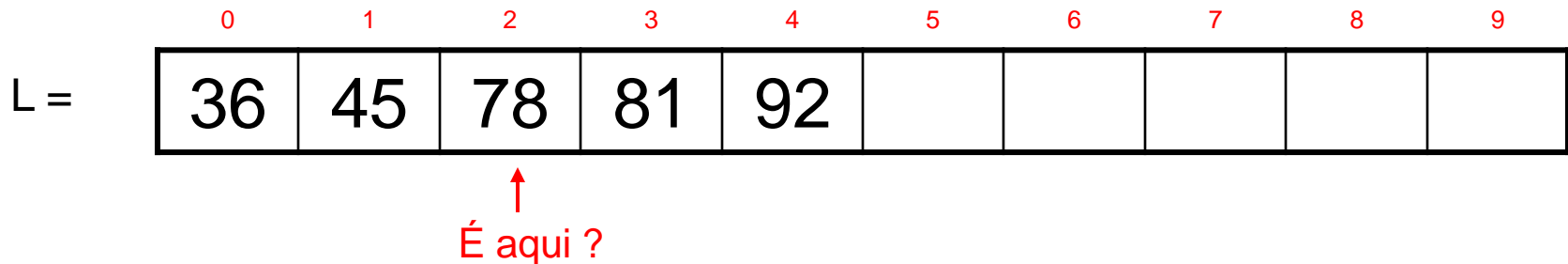
Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no final da Lista)

Quantidade: 5



Item a ser inserido: 98

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no final da Lista)

Quantidade: 5

L =

0	1	2	3	4	5	6	7	8	9
36	45	78	81	92					

↑
É aqui ?
NÃO

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

Item a ser inserido: 98

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no final da Lista)

Quantidade: 5

L =

0	1	2	3	4	5	6	7	8	9
36	45	78	81	92					

↑
É aqui ?

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

Item a ser inserido: 98

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no final da Lista)

Quantidade: 5

L =

0	1	2	3	4	5	6	7	8	9
36	45	78	81	92					

↑
É aqui ?
NÃO

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

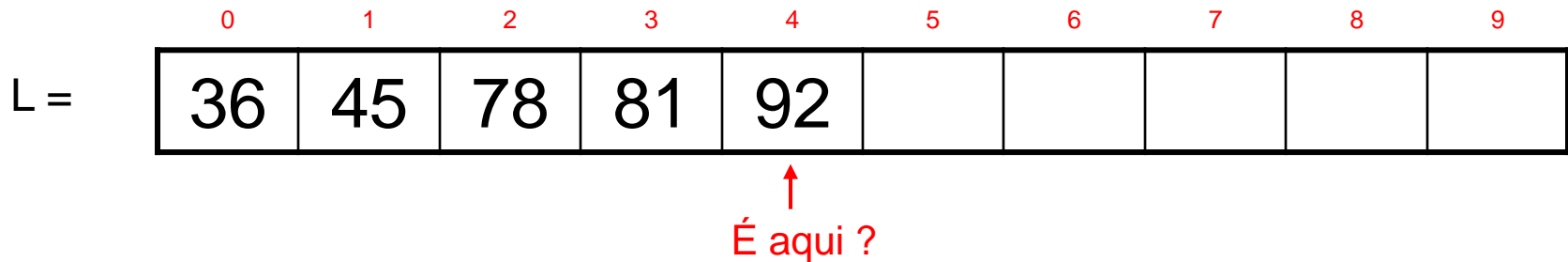
Item a ser inserido: 98

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no final da Lista)

Quantidade: 5



Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

Item a ser inserido: 98

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no final da Lista)

Quantidade: 5

L =

0	1	2	3	4	5	6	7	8	9
36	45	78	81	92					

↑
É aqui ?
NÃO

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

Item a ser inserido: 98

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no final da Lista)

Quantidade: 6

L =

0	1	2	3	4	5	6	7	8	9
36	45	78	81	92	98				

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

Item a ser inserido: 98

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no meio da Lista)

Quantidade: 6

L =

0	1	2	3	4	5	6	7	8	9
36	45	78	81	92	98				

Item a ser inserido: 80

Lista Estática Sequencial

- Funcionamento (Inserção no meio da Lista)

Quantidade: 6

L =

0	1	2	3	4	5	6	7	8	9
36	45	78	81	92	98				

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

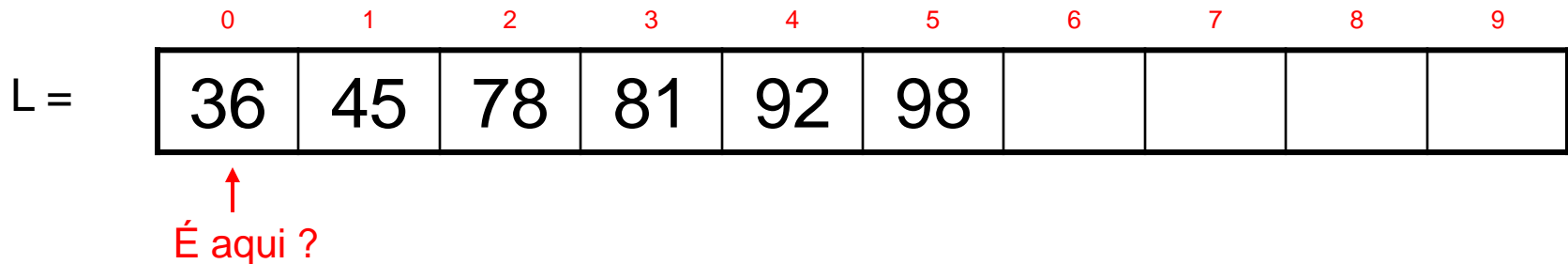
Item a ser inserido: 80

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no meio da Lista)

Quantidade: 6



Item a ser inserido: 80

Quantidade é maior que ZERO, significa que a há itens na lista.

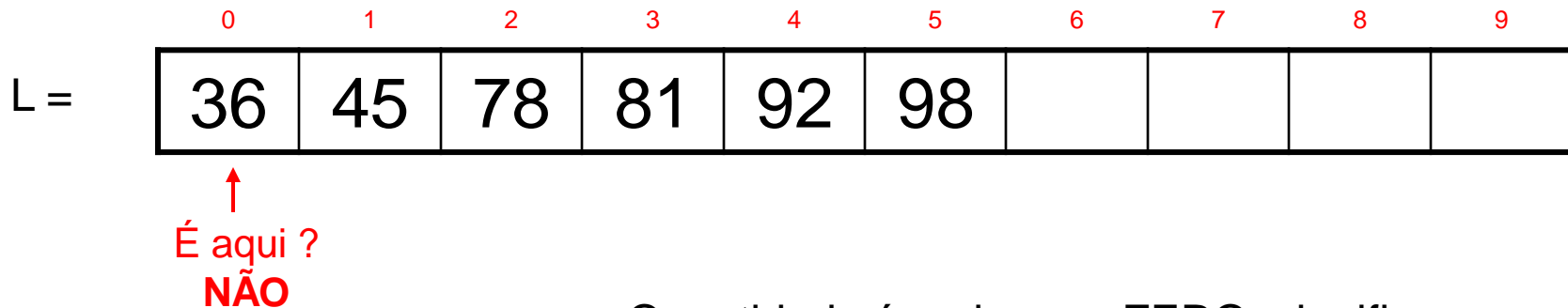
Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no meio da Lista)

Quantidade: 6



Item a ser inserido: 80

Quantidade é maior que ZERO, significa que a há itens na lista.

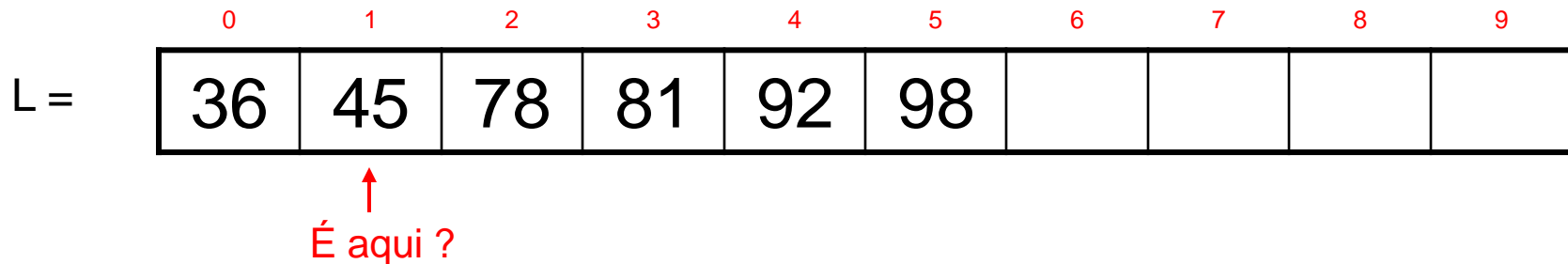
Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no meio da Lista)

Quantidade: 6



Item a ser inserido: 80

Quantidade é maior que ZERO, significa que a há itens na lista.

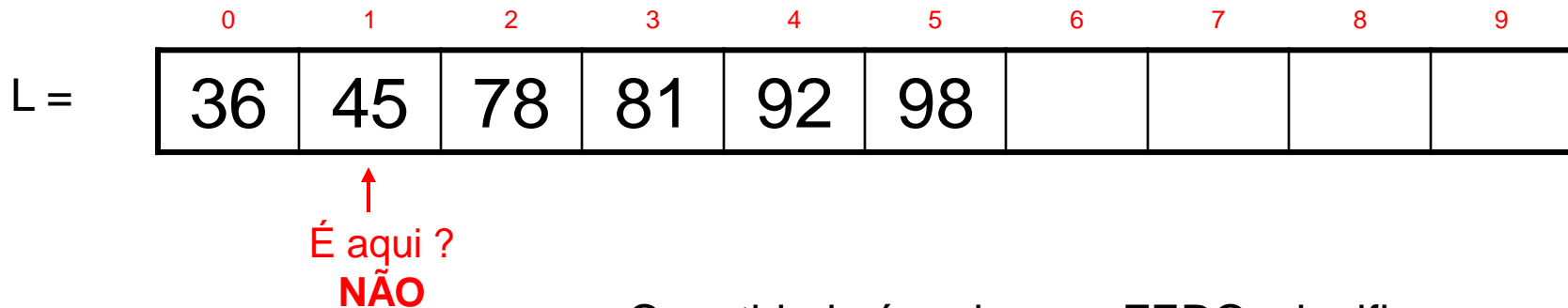
Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no meio da Lista)

Quantidade: 6



Item a ser inserido: 80

Quantidade é maior que ZERO, significa que a há itens na lista.

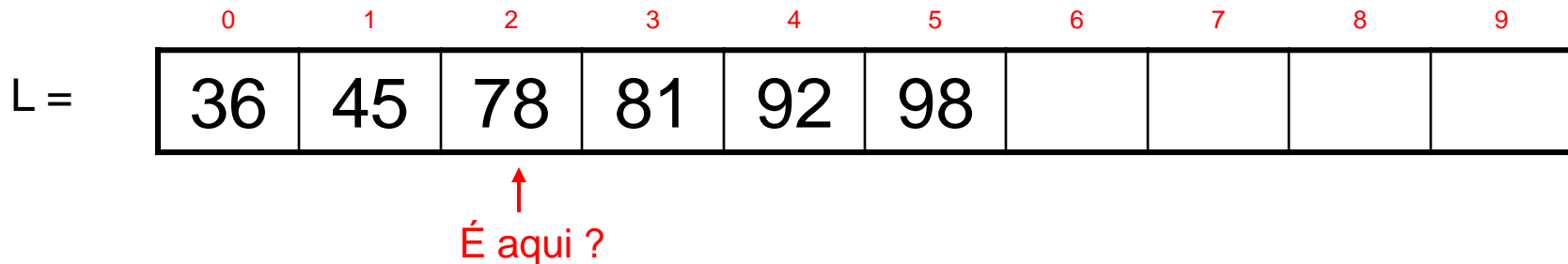
Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no meio da Lista)

Quantidade: 6



Item a ser inserido: 80

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no meio da Lista)

Quantidade: 6

L =

0	1	2	3	4	5	6	7	8	9
36	45	78	81	92	98				

↑
É aqui ?
NÃO

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

Item a ser inserido: 80

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no meio da Lista)

Quantidade: 6

L =

0	1	2	3	4	5	6	7	8	9
36	45	78	81	92	98				

↑
É aqui ?

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

Item a ser inserido: 80

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no meio da Lista)

Quantidade: 6

L =

0	1	2	3	4	5	6	7	8	9
36	45	78	81	92	98				

↑
É aqui ?

SIM

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

Item a ser inserido: 80

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento (Inserção no meio da Lista)

Quantidade: 7

L =

0	1	2	3	4	5	6	7	8	9
36	45	78	80	81	92	98			

↑
É aqui ?

SIM

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

Item a ser inserido: 80

- procurar a posição para inserir o elemento.
- ao encontrar inserir o elemento.
- incrementar a Quantidade em uma unidade

Lista Estática Sequencial

- Funcionamento

Quantidade: 7

L =

0	1	2	3	4	5	6	7	8	9
36	45	78	80	81	92	98			

Lista Estática Sequencial

- Inserção: (Lembrete...)
 - Caso 1: Inserir primeiro elemento em uma lista vazia:
 - Checar a quantidade, se for Zero adicionar no índice 0 da lista.
 - Caso 2: Inserir primeiro elemento de uma lista não vazia:
 - Checar a quantidade, se for maior que Zero, então desloque todos os itens da lista um item para a direita e adicione o item novo no índice 0 da lista.
 - Caso 3: Inserir o último elemento de uma lista:
 - Quando chegar ao final da lista (atual igual quantidade). Adicione o item novo no índice quantidade da lista (última posição).
 - Caso 4: Inserir no meio da lista:
 - Quando encontra a posição de inserção do novo elemento. Desloque todos os elementos do final da lista até a posição de inserção um índice para direita e adicione o novo item na posição correta.

Lista Estática Sequencial

- Funcionamento

Quantidade: 7

L =

0	1	2	3	4	5	6	7	8	9
36	45	78	80	81	92	98			

Lista Estática Sequencial

- Funcionamento

Quantidade: 7

L =

0	1	2	3	4	5	6	7	8	9
36	45	78	80	81	92	98			

Como faremos a consulta de um item na lista L ?

Lista Estática Sequencial

- Funcionamento

Quantidade: 7

L =

0	1	2	3	4	5	6	7	8	9
36	45	78	80	81	92	98			

Como faremos a consulta de um item na lista L ?

- Busca Sequencial (Linear)

Lista Estática Sequencial

- Funcionamento

Quantidade: 7

L =

0	1	2	3	4	5	6	7	8	9
36	45	78	80	81	92	98			

Como faremos a consulta de um item na lista L ?

- Busca Sequencial (Linear)
- Busca Binária

Lista Estática Sequencial

- Funcionamento (Remoção)
 - Observe cada situação da remoção
 - Remover do início da Lista
 - Remover do meio da Lista
 - Remover do final da Lista

Lista Estática Sequencial

- Funcionamento (Remoção do início da Lista)

Quantidade: 7

L =

0	1	2	3	4	5	6	7	8	9
36	45	78	80	81	92	98			

Item a ser removido: 36

Lista Estática Sequencial

- Funcionamento (Remoção do início da Lista)

Quantidade: 7

L =

0	1	2	3	4	5	6	7	8	9
36	45	78	80	81	92	98			

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

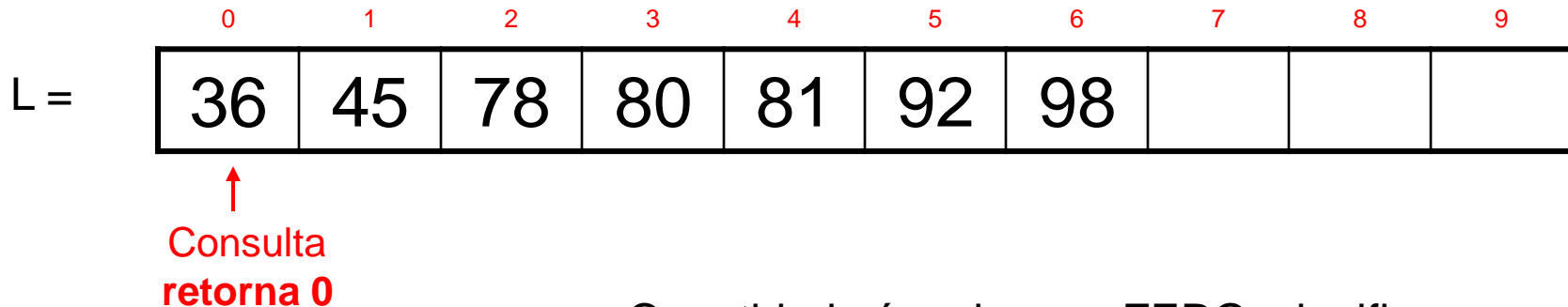
Item a ser removido: 36

- Invoque a rotina de busca para o elemento a ser removido.
- Caso o elemento esteja na lista, remova-o e decremente Quantidade de uma unidade
- Caso contrário, exiba uma mensagem.

Lista Estática Sequencial

- Funcionamento (Remoção do início da Lista)

Quantidade: 7



Item a ser removido: 36

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

- Invoque a rotina de busca para o elemento a ser removido.
- Caso o elemento esteja na lista, remova-o e decremente Quantidade de uma unidade
- Caso contrário, exiba uma mensagem.

Lista Estática Sequencial

- Funcionamento (Remoção do início da Lista)

Quantidade: 6

L =

0	1	2	3	4	5	6	7	8	9
45	78	80	81	92	98				

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

Item a ser removido: 36

- Invoque a rotina de busca para o elemento a ser removido.
- Caso o elemento esteja na lista, remova-o e decremente Quantidade de uma unidade
- Caso contrário, exiba uma mensagem.

Lista Estática Sequencial

- Funcionamento (Remoção do fim da Lista)

Quantidade: 6

L =

0	1	2	3	4	5	6	7	8	9
45	78	80	81	92	98				

Item a ser removido: 98

Lista Estática Sequencial

- Funcionamento (Remoção do fim da Lista)

Quantidade: 6

L =

0	1	2	3	4	5	6	7	8	9
45	78	80	81	92	98				

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

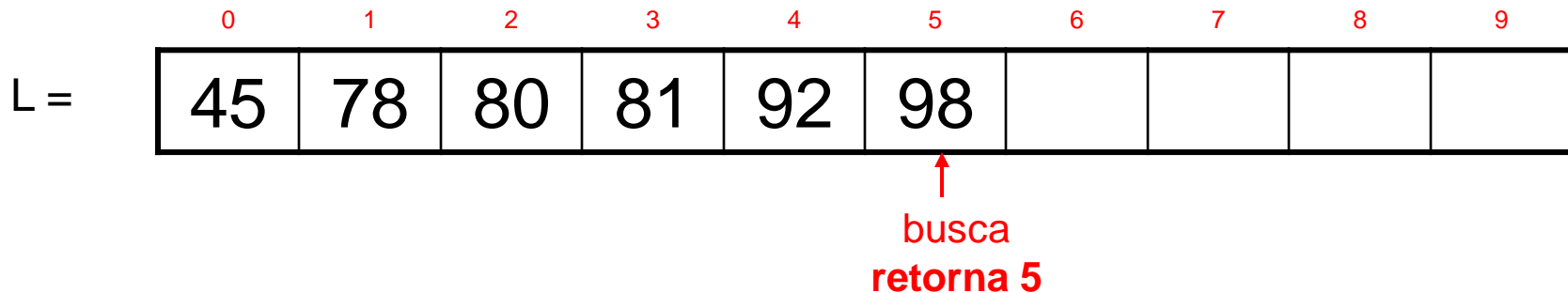
Item a ser removido: 98

- Invoque a rotina de busca para o elemento a ser removido.
- Caso o elemento esteja na lista, remova-o e decremente Quantidade de uma unidade
- Caso contrário, exiba uma mensagem.

Lista Estática Sequencial

- Funcionamento (Remoção do fim da Lista)

Quantidade: 6



Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

Item a ser removido: 98

- Invoque a rotina de busca para o elemento a ser removido.
- Caso o elemento esteja na lista, remova-o e decremente Quantidade de uma unidade
- Caso contrário, exiba uma mensagem.

Lista Estática Sequencial

- Funcionamento (Remoção do fim da Lista)

Quantidade: 5

L =

0	1	2	3	4	5	6	7	8	9
45	78	80	81	92					

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

Item a ser removido: 98

- Invoque a rotina de busca para o elemento a ser removido.
- Caso o elemento esteja na lista, remova-o e decremente Quantidade de uma unidade
- Caso contrário, exiba uma mensagem.

Lista Estática Sequencial

- Funcionamento (Remoção do meio da Lista)

Quantidade: 5

L =

0	1	2	3	4	5	6	7	8	9
45	78	80	81	92					

Item a ser removido: 80

Lista Estática Sequencial

- Funcionamento (Remoção do meio da Lista)

Quantidade: 5

L =

0	1	2	3	4	5	6	7	8	9
45	78	80	81	92					

Item a ser removido: 80

Quantidade é maior que ZERO, significa que a há itens na lista.

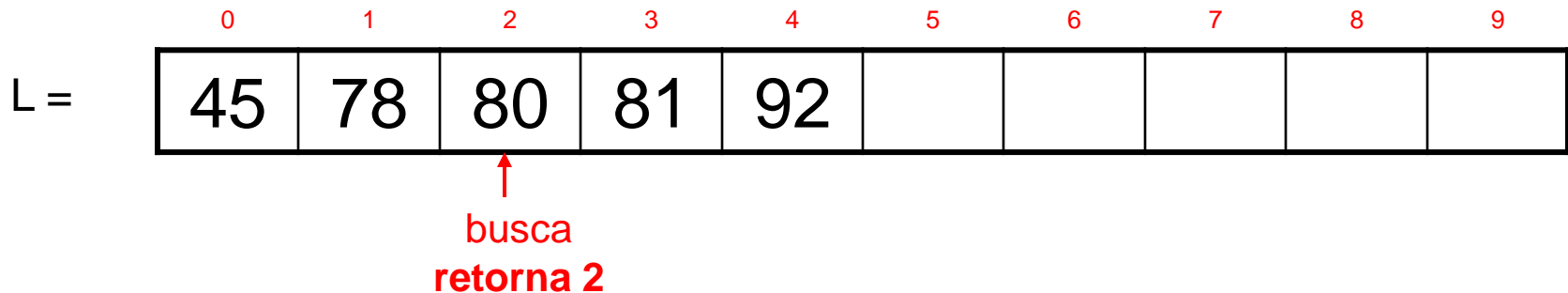
Portanto:

- Invoque a rotina de busca para o elemento a ser removido.
- Caso o elemento esteja na lista, remova-o e decremente Quantidade de uma unidade
- Caso contrário, exiba uma mensagem.

Lista Estática Sequencial

- Funcionamento (Remoção do meio da Lista)

Quantidade: 5



Item a ser removido: 80

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

- Invoque a rotina de busca para o elemento a ser removido.
- Caso o elemento esteja na lista, remova-o e decremente Quantidade de uma unidade
- Caso contrário, exiba uma mensagem.

Lista Estática Sequencial

- Funcionamento (Remoção do meio da Lista)

Quantidade: 4

L =

0	1	2	3	4	5	6	7	8	9
45	78	81	92						

Quantidade é maior que ZERO, significa que a há itens na lista.

Portanto:

Item a ser removido: 80

- Invoque a rotina de busca para o elemento a ser removido.
- Caso o elemento esteja na lista, remova-o e decremente Quantidade de uma unidade
- Caso contrário, exiba uma mensagem.

Lista Estática Sequencial

- Estruturas:
 - tLista
 - tItem → Por simplificação do exemplo nossos itens serão inteiros.

Lista Estática Sequencial

- Implementação da Estrutura da Lista

```
1. struct tLista
2. {
3.     int *itens;
4.     int quantidade;
5.     int tamanho;
6. };
```

Por simplificação os
itens são um vetor de
inteiros.

- Inicialização da Lista

```
1. void iniciaLista(struct tLista *lista, int n)
2. {
3.     lista->itens = (int*) malloc(n * sizeof(int));
4.     lista->tamanho = n;
5.     lista->quantidade = 0;
6. }
```

Lista Estática Sequencial

- Sugestões para Funções:
 - void iniciaLista(struct tLista *, int);
 - int busca(struct tLista, int);
 - int buscaLinear(struct tLista);
 - int buscaBinaria(struct tLista, int, int, int);
 - int lerItem(void);
 - void insere(struct tLista *, int);
 - void deslocaDir(struct tLista*, int);
 - void consulta(struct tLista, int);
 - int remove(struct tLista *);
 - void deslocaEsq(struct tLista*, int);

Lista Estática Sequencial

- Implementação da Busca

```
1. int busca( struct tLista lista, int it)
2. {
3.     return buscaLinear(lista, it);
4. }
```

```
1. int buscaLinear( struct tLista lista, int it)
2. {
3.     int i, ret = -1;
4.     for(i=0; i<lista.quantidade && ret==-1; i++)
5.     {
6.         if(lista.itens[i] == it)
7.         {
8.             ret = i;
9.         }
10.    }
11.    return ret;
12. }
```

Lista Estática Sequencial

- Implementação da Busca

```
1. int busca( struct tLista lista, int it)
2. {
3.     return buscaBinaria(lista, 0, lista.quantidade, it);
4. }
```

```
1. int buscaBinaria( struct tLista lista, int inicio, int fim, int it)
2. {
3.     int meio = (inicio+fim)/2;
4.     if(inicio <= fim)
5.         if(lista.itens[meio] == it)
6.             return meio;
7.         if(it < lista.itens[meio])
8.             return buscaBinaria(lista, inicio, meio-1, it);
9.         else
10.            return buscaBinaria(lista, meio+1, fim, it);
11.     return -1;
12. }
```


Lista Estática Sequencial

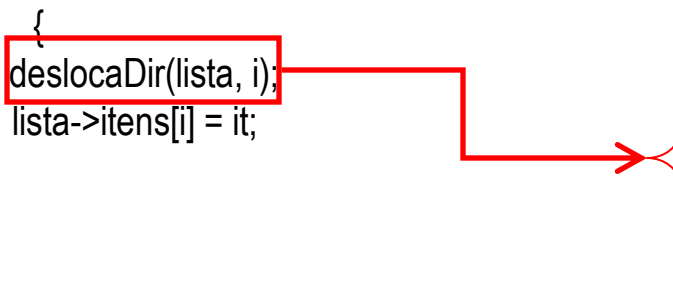
- Implementação da leitura do item.

```
1. int lerItem(void)
2. {
3.     int it;
4.     printf("Informe a chave do item: ");
5.     scanf("%d", &it);
6.     return it;
7. }
```

Lista Estática Sequencial

- Implementação da inserção.

```
1. void insere(struct tLista *lista) {
2.     int i, achei=0, it = lerItem();
3.     if(lista->quantidade != TAMANHO) {
4.         if(lista->quantidade == 0) {
5.             lista->itens[lista->quantidade] = it;
6.         }
7.         else {
8.             for(i=0; i<lista->quantidade && achei==0; i++) {
9.                 if(lista->itens[i] >= it) {
10.                     achei=1;
11.                     if(lista->itens[i] == it) {
12.                         printf("\nChave: %d ja existe na lista!\n", it);
13.                         system("pause");
14.                         return;
15.                     }
16.                     else {
17.                         deslocaDir(lista, i);
18.                         lista->itens[i] = it;
19.                     }
20.                 }
21.             }
22.             if(achei == 0) {
23.                 lista->itens[lista->quantidade] = it;
24.             }
25.         }
26.         lista->quantidade++;
27.     }
28.     else {
29.         printf("\nLista Cheia!\n");
30.         system("pause");
31.     }
32. }
```



```
1. void deslocaDir(struct tLista *lista, int ateh)
2. {
3.     int j;
4.     for(j=lista->quantidade-1; j >= ateh; j -- )
5.         lista->itens[j+1] = lista->itens[j];
6. }
```

Lista Estática Sequencial

- Implementação da inserção.

```
1. void insere(struct tLista *lista) {  
2.     int i, achei=0, it = lerItem();  
3.     if(lista->quantidade != TAMANHO) {  
4.         if(lista->quantidade == 0) {  
5.             lista->itens[lista->quantidade] = it;  
6.         }  
7.         else {  
8.             for(i=0; i<lista->quantidade && achei==0; i++) {  
9.                 if(lista->itens[i] >= it) {  
10.                     achei=1;  
11.                     if(lista->itens[i] == it) {  
12.                         printf("\nChave: %d ja existe na lista!\n", it);  
13.                         system("pause");  
14.                         return;  
15.                     }  
16.                     else {  
17.                         deslocaDir(lista, i);  
18.                         lista->itens[i] = it;  
19.                     }  
20.                 }  
21.             }  
22.         }  
23.     }  
24. }
```

```
22.     if(achei == 0) {  
23.         lista->itens[lista->quantidade] = it;  
24.     }  
25. }  
26. lista->quantidade++;  
27. }  
28. else {  
29.     printf("\nLista Cheia!\n");  
30.     system("pause");  
31. }  
32. }
```

```
1. void deslocaDir(struct tLista *lista, int ateh)  
2. {  
3.     int j;  
4.     for(j=lista->quantidade-1; j >= ateh; j -- )  
5.         lista->itens[j+1] = lista->itens[j];  
6. }
```

Lista Estática Sequencial

- Implementação da inserção.

```
1. void insere(struct tLista *lista) {
2.     int i, achei=0, it = lerItem();
3.     if(lista->quantidade != TAMANHO) {
4.         if(lista->quantidade == 0) {
5.             lista->itens[lista->quantidade] = it;
6.         }
7.         else {
8.             for(i=0; i<lista->quantidade && achei==0; i++) {
9.                 if(lista->itens[i] >= it) {
10.                     achei=1;
11.                     if(lista->itens[i] == it) {
12.                         printf("\nChave: %d ja existe na lista!\n", it);
13.                         system("pause");
14.                         return;
15.                     }
16.                     else {
17.                         deslocaDir(lista, i);
18.                         lista->itens[i] = it;
19.                     }
20.                 }
21.             }
22.             if(achei == 0) {
23.                 lista->itens[lista->quantidade] = it;
24.             }
25.             lista->quantidade++;
26.         }
27.         else {
28.             printf("\nLista Cheia!\n");
29.             system("pause");
30.         }
31.     }
32. }
```

Inserindo o Primeiro da Lista

Diagram illustrating the insertion process:

- Red arrows indicate the flow of execution for inserting the first element (when `lista->quantidade == 0`).
- Blue arrows indicate the flow of execution for inserting an element into the middle of the list (when `achei == 0` and `lista->quantidade > 0`).
- The `deslocaDir` function is called to shift elements to the right to make space for the new element.

```
1. void deslocaDir(struct tLista *lista, int ateh)
2. {
3.     int j;
4.     for(j=lista->quantidade-1; j >= ateh; j -- )
5.         lista->itens[j+1] = lista->itens[j];
6. }
```

Lista Estática Sequencial

- Implementação da inserção.

```
1. void insere(struct tLista *lista) {
2.     int i, achei=0, it = lerItem();
3.     if(lista->quantidade != TAMANHO) {
4.         if(lista->quantidade == 0) {
5.             lista->itens[lista->quantidade] = it;
6.         }
7.         else {
8.             Procurando a posição para inserir
9.             for(i=0; i<lista->quantidade && achei==0; i++) {
10.                 if(lista->itens[i] >= it) {
11.                     achei=1;
12.                     if(lista->itens[i] == it) {
13.                         printf("\nChave: %d ja existe na lista!\n", it);
14.                         system("pause");
15.                         return;
16.                     }
17.                     else {
18.                         deslocaDir(lista, i);
19.                         lista->itens[i] = it;
20.                     }
21.                 }
22.             }
23.         }
24.     }
25. }
```

```
22.     if(achei == 0) {
23.         lista->itens[lista->quantidade] = it;
24.     }
25. }
26. lista->quantidade++;
27. }
28. else {
29.     printf("\nLista Cheia!\n");
30.     system("pause");
31. }
32. }
```

```
1. void deslocaDir(struct tLista *lista, int ateh)
2. {
3.     int j;
4.     for(j=lista->quantidade-1; j >= ateh; j -- )
5.         lista->itens[j+1] = lista->itens[j];
6. }
```

Lista Estática Sequencial

- Implementação da inserção.

```
1. void insere(struct tLista *lista) {
2.     int i, achei=0, it = lerItem();
3.     if(lista->quantidade != TAMANHO) {
4.         if(lista->quantidade == 0) {
5.             lista->itens[lista->quantidade] = it;
6.         }
7.         else {
8.             for(i=0; i<lista->quantidade && achei==0; i++) {
9.                 if(lista->itens[i] >= it) {
10.                     achei=1;
11.                     if(lista->itens[i] == it) {
12.                         printf("\nChave: %d ja existe na lista!\n", it);
13.                         system("pause");
14.                         return;
15.                     }
16.                     else {
17.                         deslocaDir(lista, i);
18.                         lista->itens[i] = it;
19.                     }
20.                 }
21.             }
```

```
22.         if(achei == 0) {
23.             lista->itens[lista->quantidade] = it;
24.         }
25.     }
26.     lista->quantidade++;
27. }
28. else {
29.     printf("\nLista Cheia!\n");
30.     system("pause");
31. }
32. }
```

```
1. void deslocaDir(struct tLista *lista, int ateh)
2. {
3.     int j;
4.     for(j=lista->quantidade-1; j >= ateh; j -- )
5.         lista->itens[j+1] = lista->itens[j];
6. }
```

Se for item repetido

Lista Estática Sequencial

- Implementação da inserção.

```
1. void insere(struct tLista *lista) {  
2.     int i, achei=0, it = lerItem();  
3.     if(lista->quantidade != TAMANHO) {  
4.         if(lista->quantidade == 0) {  
5.             lista->itens[lista->quantidade] = it;  
6.         }  
7.         else {  
8.             for(i=0; i<lista->quantidade && achei==0; i++) {  
9.                 if(lista->itens[i] >= it) {  
10.                     achei=1;  
11.                     if(lista->itens[i] == it) {  
12.                         printf("\nChave: %d ja existe na lista!\n", it);  
13.                         system("pause");  
14.                         return;  
15.                     }  
16.                     Desloca itens e insere na lista  
17.                     deslocaDir(lista, i);  
18.                     lista->itens[i] = it;  
19.                 }  
20.             }  
21.         }  
22.     }  
23. }
```

```
22.     if(achei == 0) {  
23.         lista->itens[lista->quantidade] = it;  
24.     }  
25. }  
26. lista->quantidade++;  
27. }  
28. else {  
29.     printf("\nLista Cheia!\n");  
30.     system("pause");  
31. }  
32. }
```

```
1. void deslocaDir(struct tLista *lista, int ateh)  
2. {  
3.     int j;  
4.     for(j=lista->quantidade-1; j >= ateh; j -- )  
5.         lista->itens[j+1] = lista->itens[j];  
6. }
```

Lista Estática Sequencial

- Implementação da inserção.

```
1. void insere(struct tLista *lista) {
2.     int i, achei=0, it = lerItem();
3.     if(lista->quantidade != TAMANHO) {
4.         if(lista->quantidade == 0) {
5.             lista->itens[lista->quantidade] = it;
6.         }
7.         else {
8.             for(i=0; i<lista->quantidade && achei==0; i++) {
9.                 if(lista->itens[i] >= it) {
10.                     achei=1;
11.                     if(lista->itens[i] == it) {
12.                         printf("\nChave: %d ja existe na lista!\n", it);
13.                         system("pause");
14.                         return;
15.                     }
16.                     else {
17.                         deslocaDir(lista, i);
18.                         lista->itens[i] = it;
19.                     }
20.                 }
21.             }
```

Inserindo o último da lista

```
22.         if(achei == 0) {
23.             lista->itens[lista->quantidade] = it;
24.         }
25.     }
26.     lista->quantidade++;
27. }
28. else {
29.     printf("\nLista Cheia!\n");
30.     system("pause");
31. }
32. }
```

```
1. void deslocaDir(struct tLista *lista, int ateh)
2. {
3.     int j;
4.     for(j=lista->quantidade-1; j >= ateh; j -- )
5.         lista->itens[j+1] = lista->itens[j];
6. }
```


Lista Estática Sequencial

- Implementação da inserção.

```
1. void insere(struct tLista *lista) {  
2.     int i, achei=0, it = lerItem();  
3.     if(lista->quantidade != TAMANHO) {  
4.         if(lista->quantidade == 0) {  
5.             lista->itens[lista->quantidade] = it;  
6.         }  
7.         else {  
8.             for(i=0; i<lista->quantidade && achei==0; i++) {  
9.                 if(lista->itens[i] >= it) {  
10.                     achei=1;  
11.                     if(lista->itens[i] == it) {  
12.                         printf("\nChave: %d ja existe na lista!\n", it);  
13.                         system("pause");  
14.                         return;  
15.                     }  
16.                     else {  
17.                         deslocaDir(lista, i);  
18.                         lista->itens[i] = it;  
19.                     }  
20.                 }  
21.             }  
22.         }  
23.     }  
24. }
```

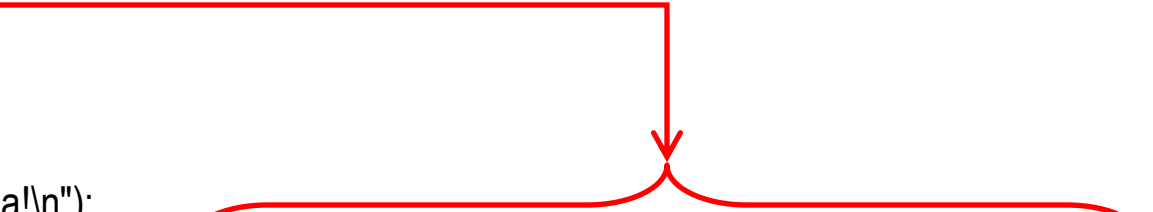
```
22.     if(achei == 0) {  
23.         lista->itens[lista->quantidade] = it;  
24.     }  
25. }  
26. lista->quantidade++;  
27. }  
28. else {  
29.     printf("\nLista Cheia!\n");  
30.     system("pause");  
31. }  
32. }
```

```
1. void deslocaDir(struct tLista *lista, int ateh)  
2. {  
3.     int j;  
4.     for(j=lista->quantidade-1; j >= ateh; j -- )  
5.         lista->itens[j+1] = lista->itens[j];  
6. }
```

Lista Estática Sequencial

- Implementação da remoção.

```
1. void remover(struct tLista *lista) {
2.     int posRem = -1;
3.     if(lista->quantidade != 0) {
4.         posRem = busca(*lista);
5.         if(posRem != -1) {
6.             printf("\nChave: %d excluida com sucesso!\n", lista->itens[posRem]);
7.             deslocaEsq(lista, posRem);
8.             lista->quantidade --;
9.         }
10.    } else {
11.        printf("\nChave nao localizada!\n");
12.    }
13. }
14. else {
15.     printf("\nLista Vazia!\n");
16. }
17. system("pause");
18. }
```

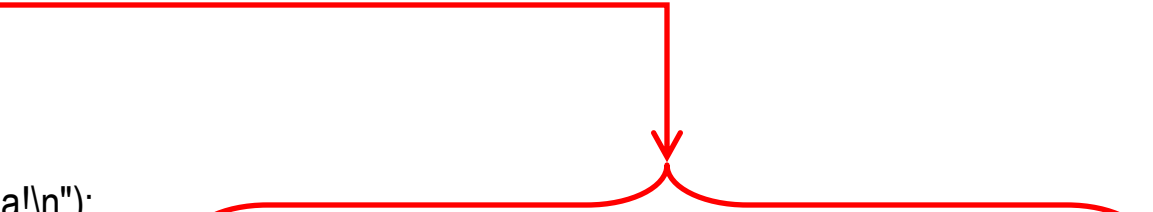


```
1. void deslocaEsq(struct tLista *lista, int de)
2. {
3.     int j;
4.     for(j=de; j < lista->quantidade; j ++ )
5.         lista->itens[j] = lista->itens[j+1];
6. }
```

Lista Estática Sequencial

- Implementação da remoção.

```
1. void remover(struct tLista *lista) {
2.     int posRem = -1;
3.     if(lista->quantidade != 0) {
4.         posRem = busca(*lista); Busca posição do item a remover
5.         if(posRem != -1) {
6.             printf("\nChave: %d excluída com sucesso!\n", lista->itens[posRem]);
7.             deslocaEsq(lista, posRem);
8.             lista->quantidade --;
9.         }
10.    } else {
11.        printf("\nChave não localizada!\n");
12.    }
13. }
14. else {
15.     printf("\nLista Vazia!\n");
16. }
17. system("pause");
18. }
```



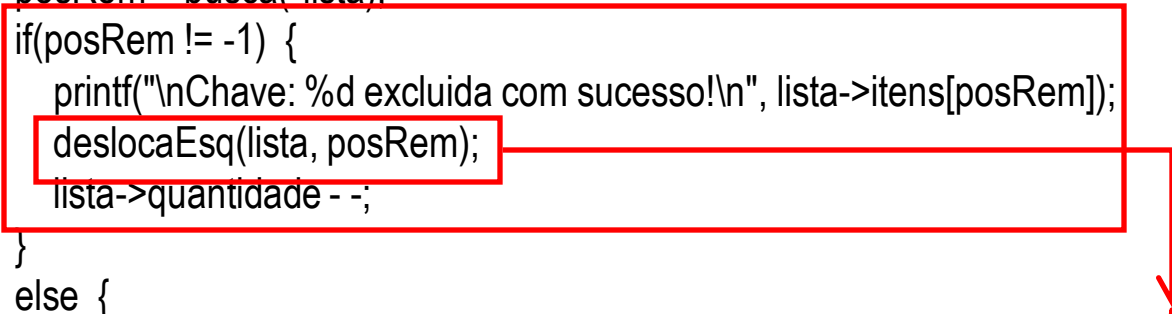
```
1. void deslocaEsq(struct tLista *lista, int de)
2. {
3.     int j;
4.     for(j=de; j < lista->quantidade; j ++ )
5.         lista->itens[j] = lista->itens[j+1];
6. }
```

Lista Estática Sequencial

- Implementação da remoção.

```
1. void remover(struct tLista *lista) {
2.     int posRem = -1;
3.     if(lista->quantidade != 0) {
4.         posRem = busca(*lista);
5.         if(posRem != -1) {
6.             printf("\nChave: %d excluida com sucesso!\n", lista->itens[posRem]);
7.             deslocaEsq(lista, posRem);
8.             lista->quantidade --;
9.         }
10.    } else {
11.        printf("\nChave nao localizada!\n");
12.    }
13. }
14. else {
15.     printf("\nLista Vazia!\n");
16. }
17. system("pause");
18. }
```

Desloca itens e remove

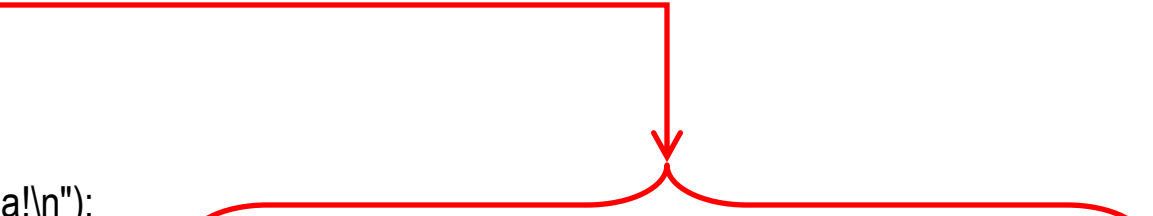


```
1. void deslocaEsq(struct tLista *lista, int de)
2. {
3.     int j;
4.     for(j=de; j < lista->quantidade; j ++ )
5.         lista->itens[j] = lista->itens[j+1];
6. }
```

Lista Estática Sequencial

- Implementação da remoção.

```
1. void remover(struct tLista *lista) {
2.     int posRem = -1;
3.     if(lista->quantidade != 0) {
4.         posRem = busca(*lista);
5.         if(posRem != -1) {
6.             printf("\nChave: %d excluida com sucesso!\n", lista->itens[posRem]);
7.             deslocaEsq(lista, posRem);
8.             lista->quantidade --;
9.         }
10.    } else {
11.        printf("\nChave nao localizada!\n");
12.    }
13. }
14. else {
15.     printf("\nLista Vazia!\n");
16. }
17. system("pause");
18. }
```



```
1. void deslocaEsq(struct tLista *lista, int de)
2. {
3.     int j;
4.     for(j=de; j < lista->quantidade; j ++ )
5.         lista->itens[j] = lista->itens[j+1];
6. }
```

Lista Estática Sequencial

- Implementação da consulta do item.

```
1. void consulta(struct tLista lista) {  
2.     int posCon=-1;  
3.     if(lista.quantidade != 0) {  
4.         posCon = busca(lista);  
5.         if( posCon != -1) {  
6.             /* coloque aqui os printf com todos os dados do registro */  
7.             printf("\nItem Chave: %d\n", lista.itens[posCon]);  
8.         }  
9.         else {  
10.            printf("\nItem com chave nao localizado!\n");  
11.        }  
12.    }  
13.    else {  
14.        printf("\nLista Vazia!\n");  
15.    }  
16.    system("pause");  
17. }
```