ESTRUTURA DE DADOS: LISTAS

Prof. Jean Nunes

LISTAS

1 – Definição e Características

2 – Lista Estática Sequential

3 – Lista Dinâmica Encadeada

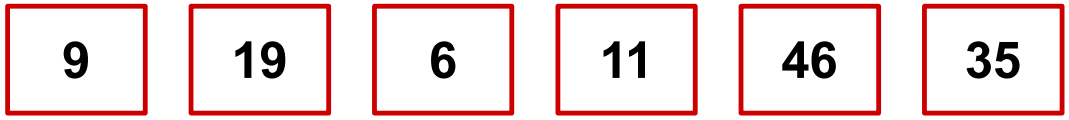
4 – Lista Dinâmica Duplamente Encadeada

5 – Listas Circulares

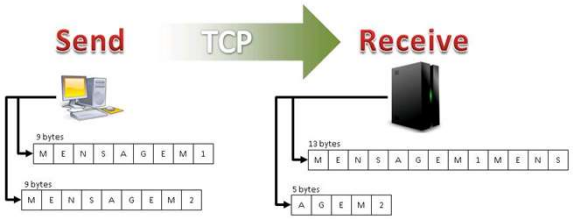
LISTAS

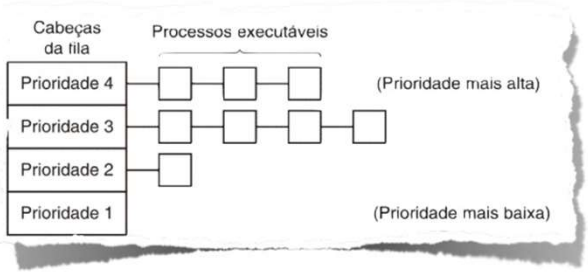
Uma estrutura do tipo “Lista” é uma sequência de elementos do mesmo tipo.

LISTA com os números da Mega Sena do próximo sorteio



- Uma “lista” pode possuir n (n >= 0) elementos (nós).
- Se n = 0, sabemos que a “lista” está vazia.





Employees

Name	Title	Office Phone	MobilePager	Description
Mike Chastain		8014156055	8015101060	Project Manager
John Maki				Project Manager
Cory Moore		8014156000		Project Manager
Kelly Hyvonen		8014156000		Estimator
Chris Gryzbowski		8014156000		Project Manager
Steve Kieffer		8014156089	8013817970	Other
Jeff Arnold		8014156050	8014300871	Project Manager
John Stalder		8014155943	8013810884	Estimator
Eric Hunter		8014156082	8017263365	Other
Michael Sant		8017967314	8013814742	Estimator

1 2 3

Add a Customer

LISTAS

Aplicações

LISTAS

Operações básicas

Criar lista

Inserir um elemento na lista

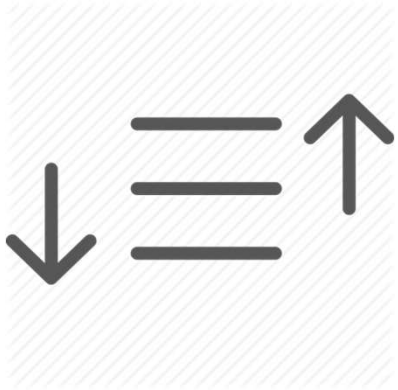
Excluir um elemento da lista

Acessar um elemento na lista

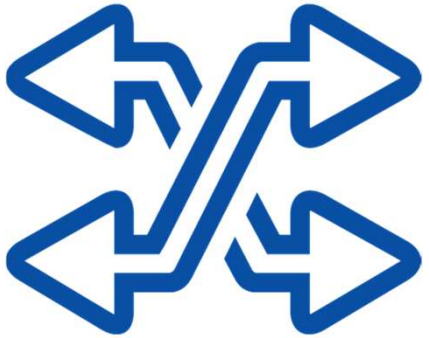
Destruir lista

LISTAS

Essas operações dependem do tipo de alocação de memória usada



Estática

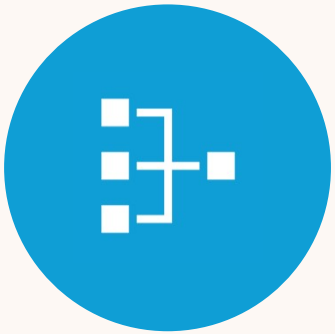


Dinâmica

LISTAS: Alocação Estática

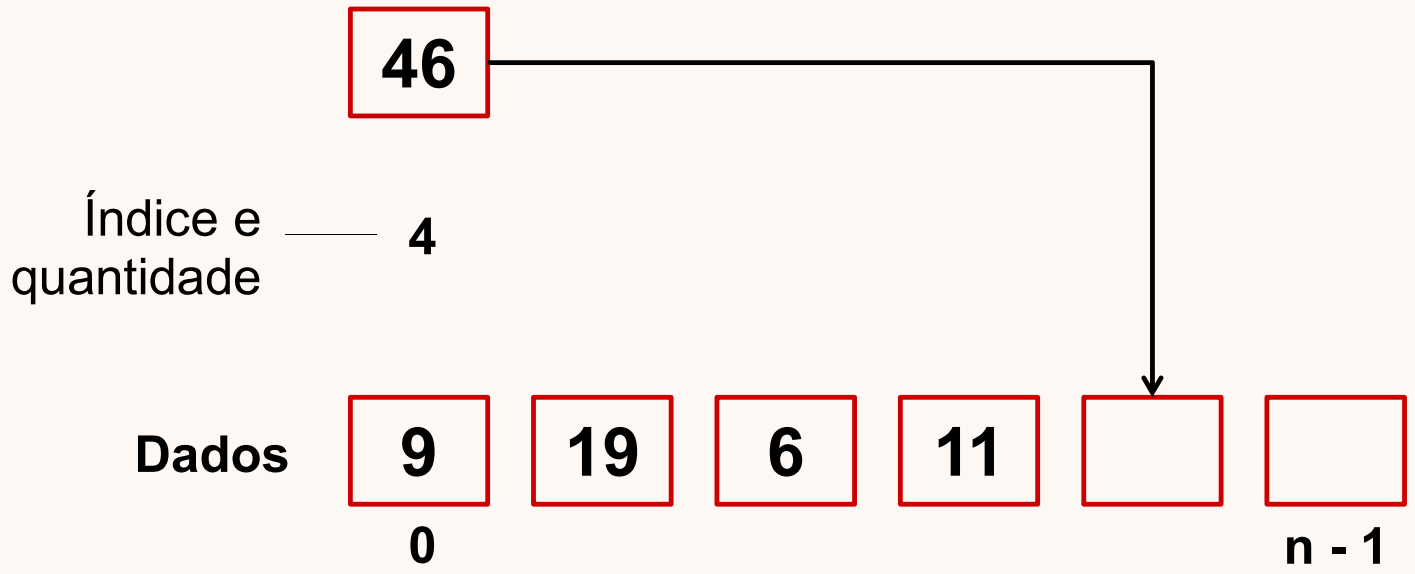
O espaço de memória é alocado no momento da compilação.

Exige a definição do número máximo de elementos da “lista”



O acesso é sequencial: os elementos são consecutivos na memória.

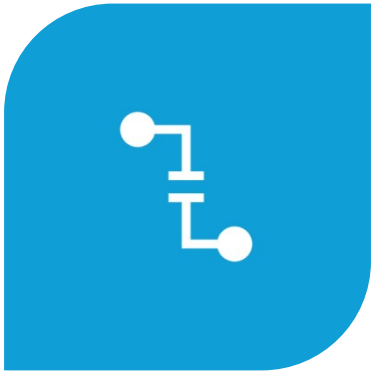
LISTAS: Alocação Estática



LISTAS: Alocação Dinâmica

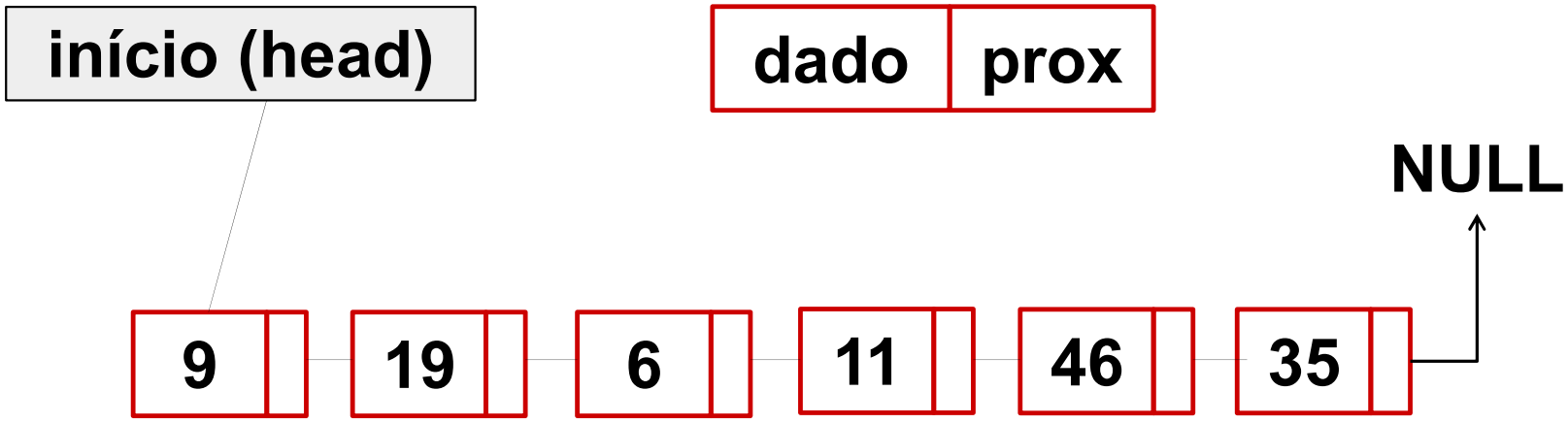
O espaço de memória é alocado em tempo de execução.

A “lista” cresce à medida que novos elementos são adicionados, e diminui à medida que os elementos são removidos.



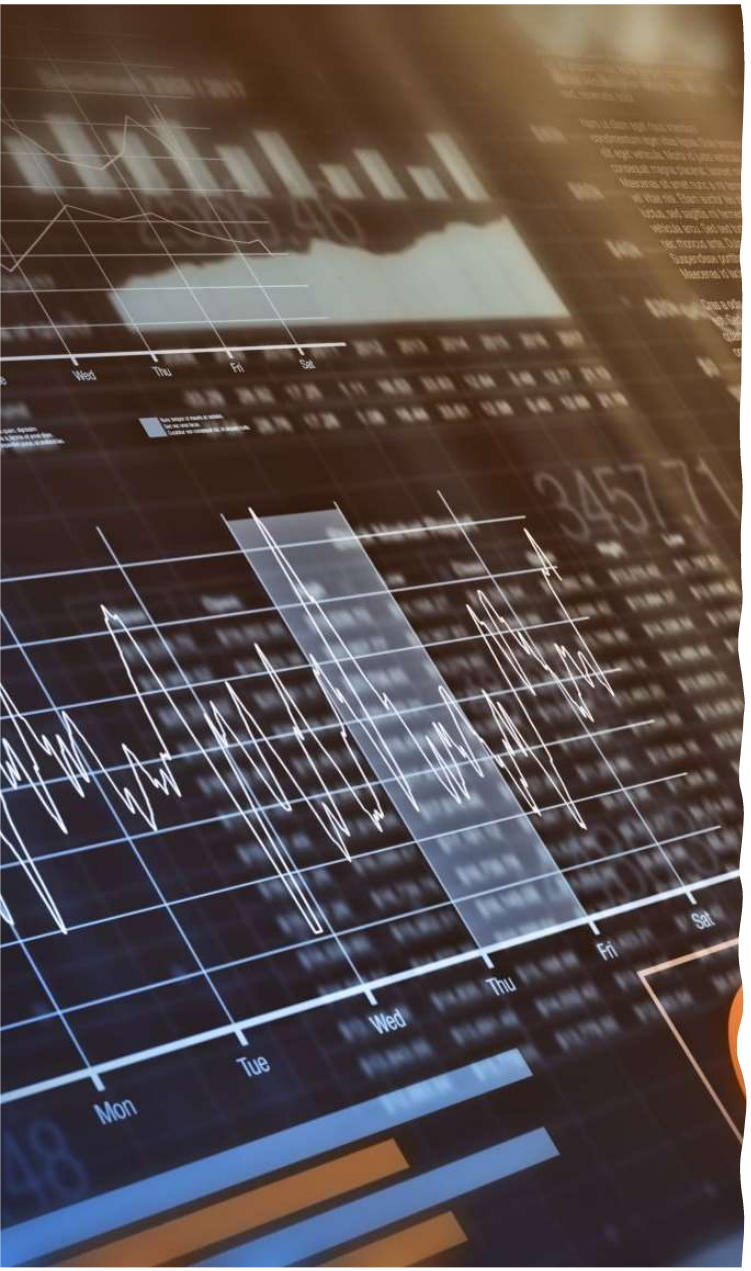
O acesso é encadeado: cada elemento pode estar em uma área distinta da memória. Acesso por ponteiros (não índices).

LISTAS: Alocação Dinâmica



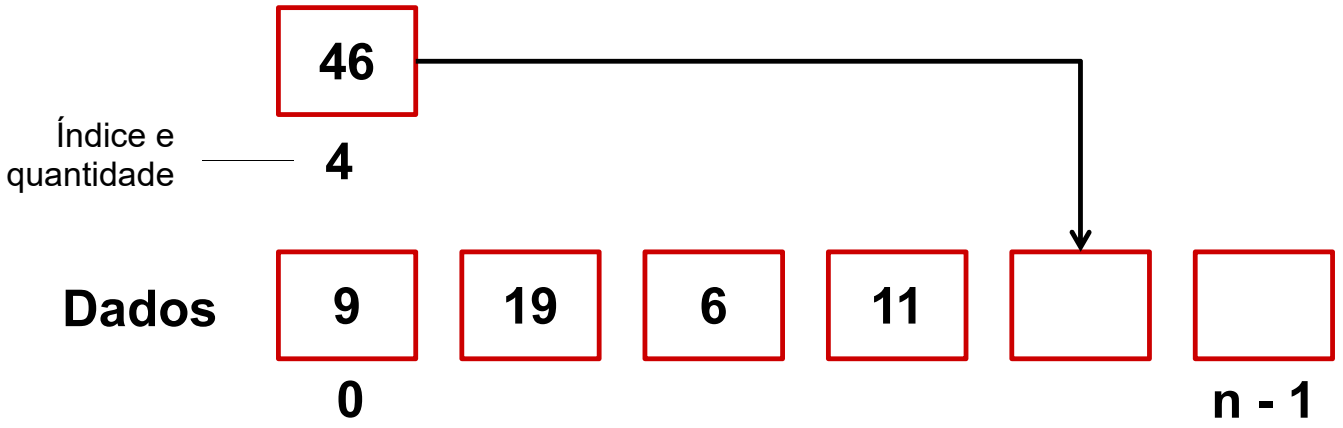
LISTAS

Lista Estática Sequencial ou Lista Estática Linear



LISTA ESTÁTICA SEQUENCIAL

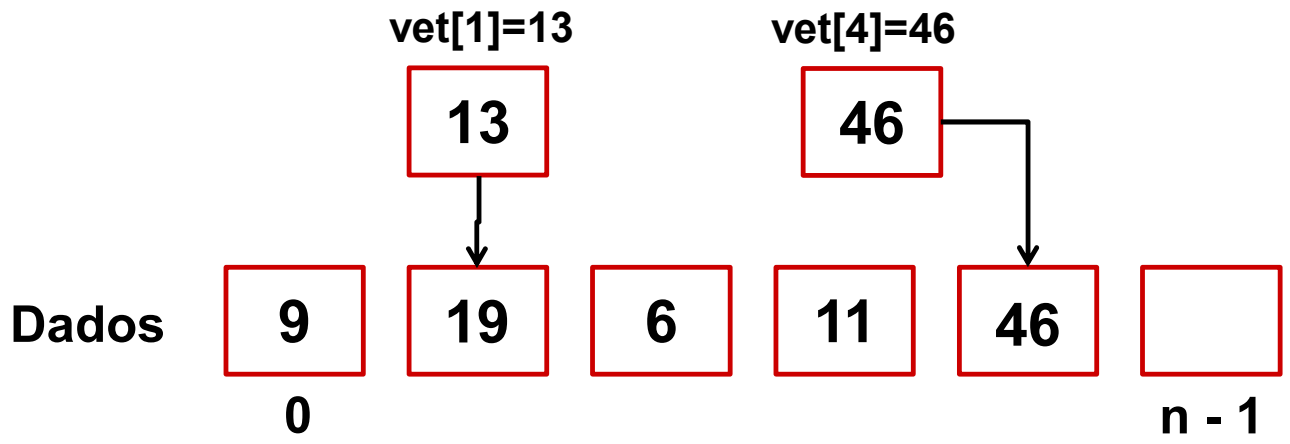
Tipo de lista onde o sucessor de um elemento (nó) ocupa a posição física seguinte do mesmo (uso de um array).



LISTA ESTÁTICA SEQUENCIAL

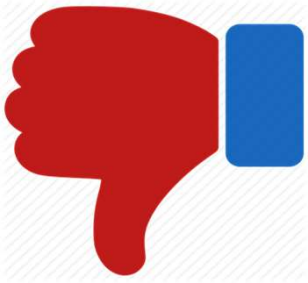
Vantagens: Acesso rápido e direto aos elementos através de um índice.

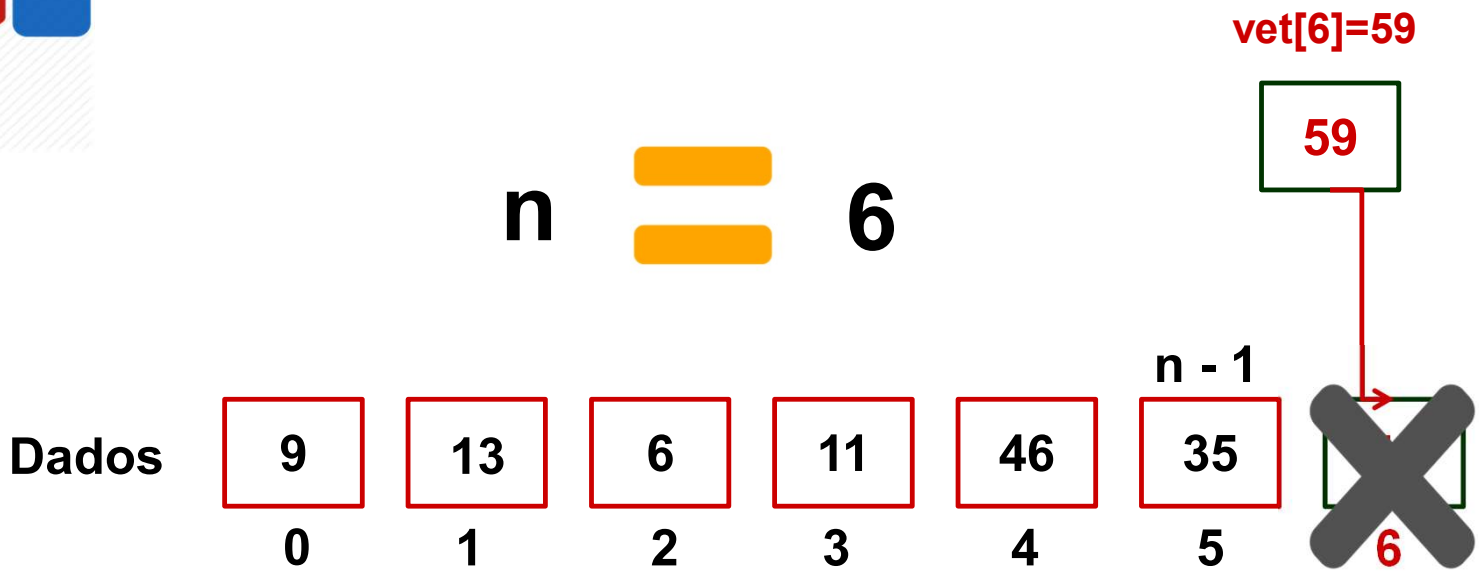
- Tempo constante para acessar um elemento.
- Facilidade de modificar informações.



LISTA ESTÁTICA SEQUENCIAL

Desvantagens: Definição prévia do tamanho do array.

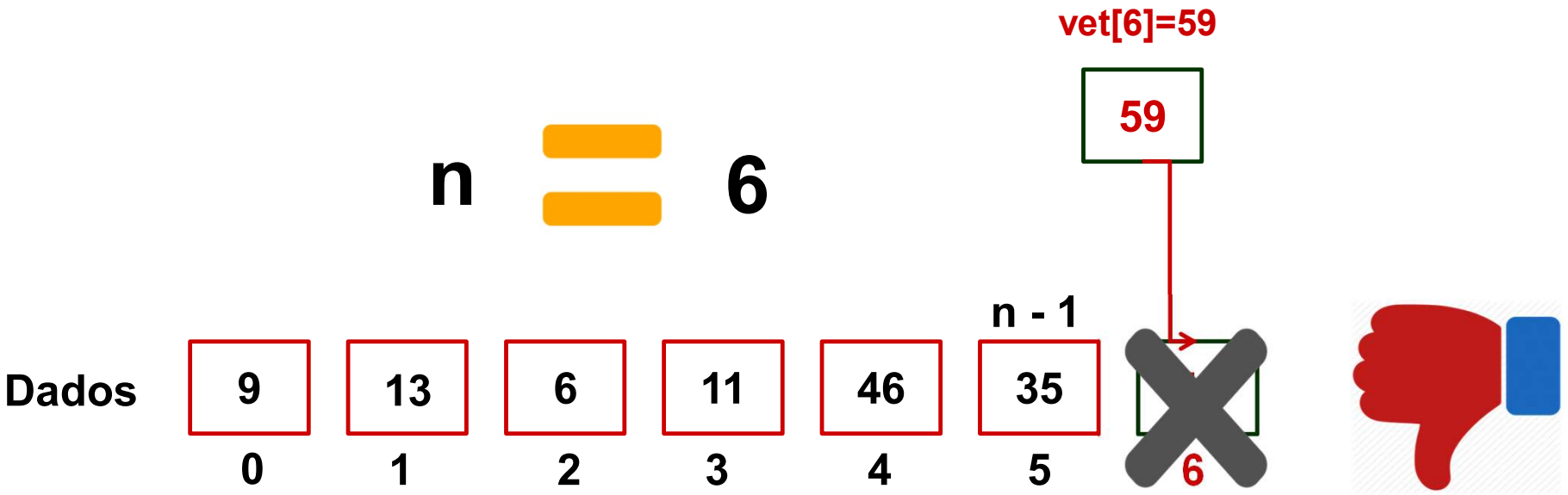




LISTA ESTÁTICA SEQUENCIAL

Desvantagens: Dificuldade para **inserir** um elemento entre outros dois.

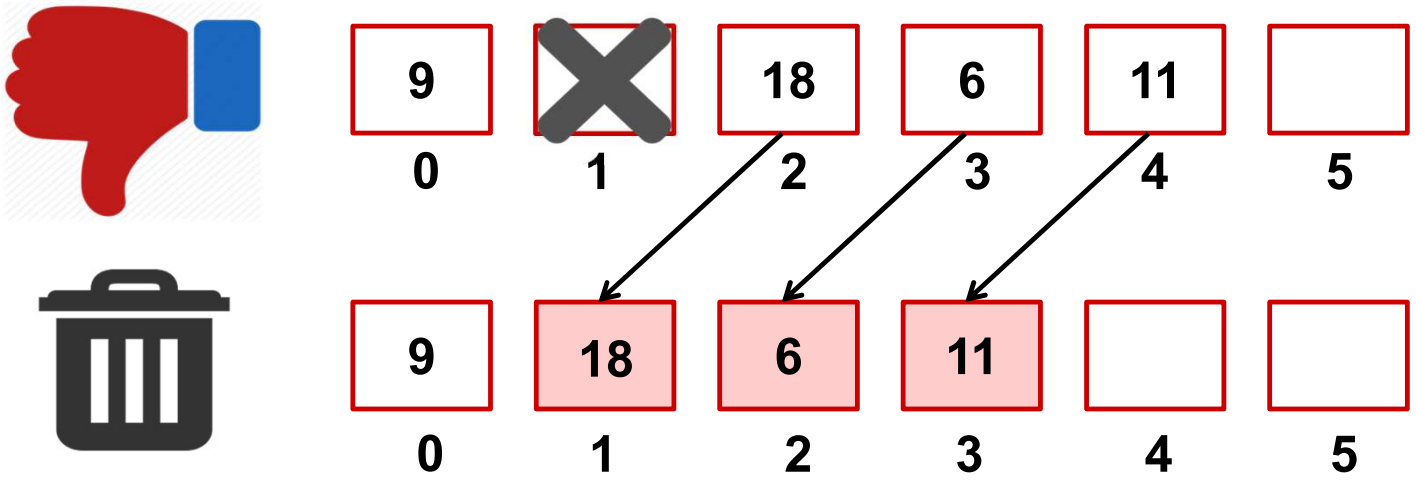
- Exige deslocamento, principalmente em lista ordenada.



LISTA ESTÁTICA SEQUENCIAL

Desvantagens: Dificuldade para **remover** um elemento entre outros dois.

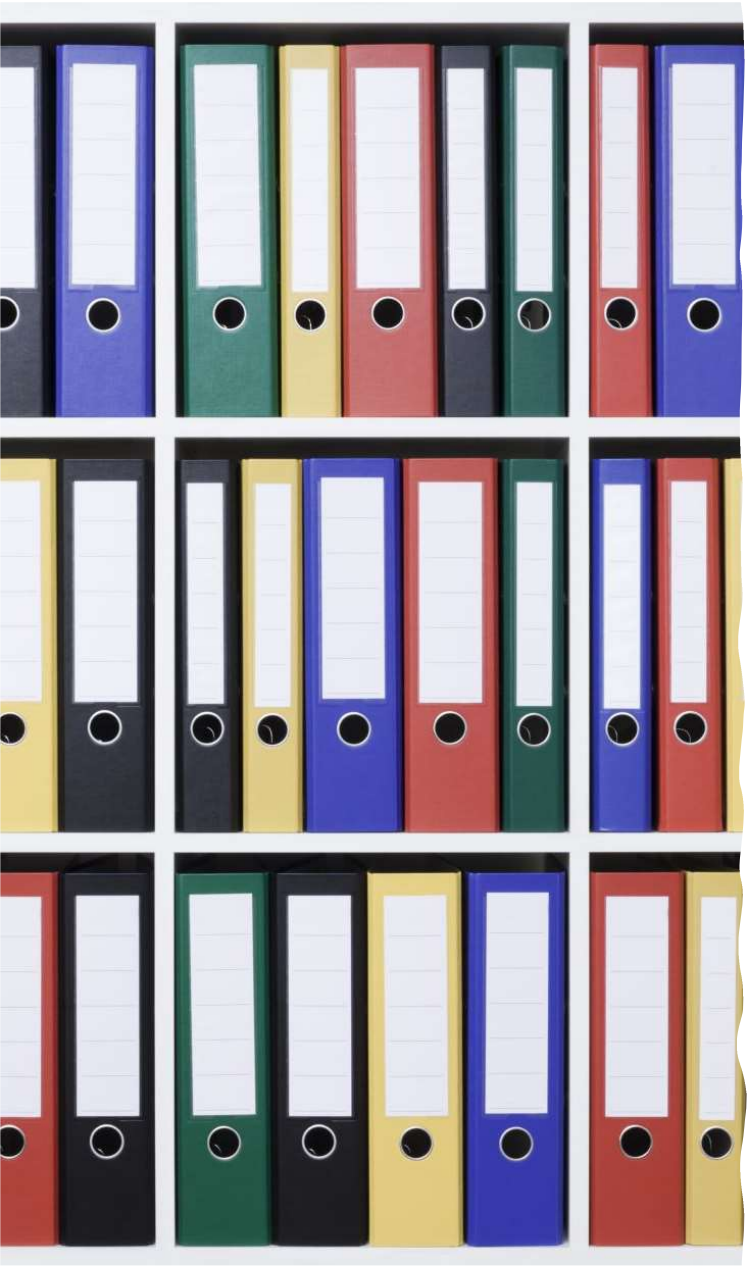
- Exige deslocamento, principalmente em lista ordenada.



LISTA ESTÁTICA SEQUENCIAL

Quando utilizar?

- Listas pequenas
- Em dispositivos com pouca memória
- Quando temos inserção/remoção apenas no final
- O tamanho da lista é bem definido
- A busca utilizando índice é a operação mais frequente



LISTA ESTÁTICA SEQUENCIAL

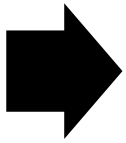
//Arquivo ListaSequencial.h

- Protótipos das funções
- O tipo de dado armazenado na lista
- O ponteiro “lista”
- Tamanho do vetor usado na lista

//Arquivo ListaSequencial.c

- O tipo de dado “lista”
- Implementa as suas funções

LISTA ESTÁTICA SEQUENCIAL



Implementando

//Arquivo ListaSequencial.h

```
#define MAX 10
struct megasena{
    int codigo;
    int concurso;
    char nome[30];
    int n1,n2,n3,n4,n5,n6;
};
typedef struct lista Lista;
Lista* cria_lista();
```

//Arquivo ListaSequencial.c

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

//Definição do tipo lista
struct lista{
    int qtd;
    struct megasena dados[MAX];
};
```

LISTA ESTÁTICA SEQUENCIAL

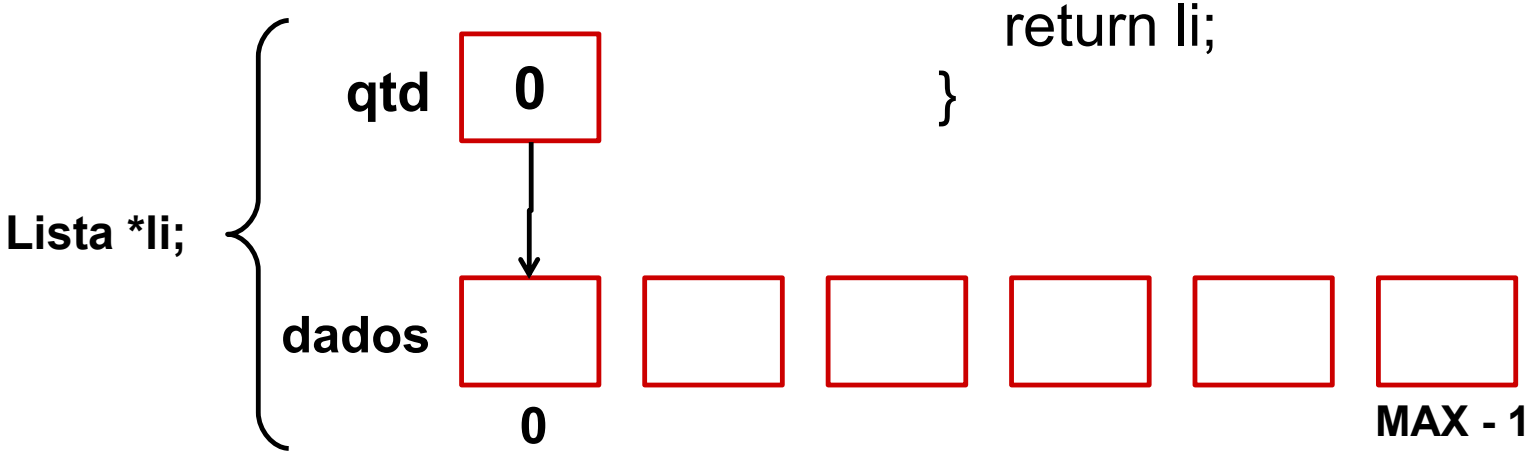
Implementando

//main.c

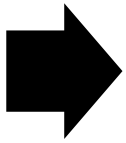
```
#include "ListaSequencial.h"
Lista* li = cria_lista();
```

//Arquivo ListaSequencial.c

```
Lista* cria_lista(){
    Lista *li;
    li = (Lista*) malloc(sizeof(struct lista));
    if(li != NULL)
        li->qtd = 0;
    return li;
}
```



LISTA ESTÁTICA SEQUENCIAL



Implementando

```
//Arquivo ListaSequencial.h
void libera_lista(Lista* li);

//Arquivo ListaSequencial.c
void libera_lista(Lista* li){
    free(li);
}

//main.c
libera_lista(li);
```

LISTA ESTÁTICA SEQUENCIAL

Informações básicas

Tamanho | Lista cheia | Lista vazia

LISTA ESTÁTICA SEQUENCIAL

Informações básicas

Tamanho

//Arquivo ListaSequencial.h

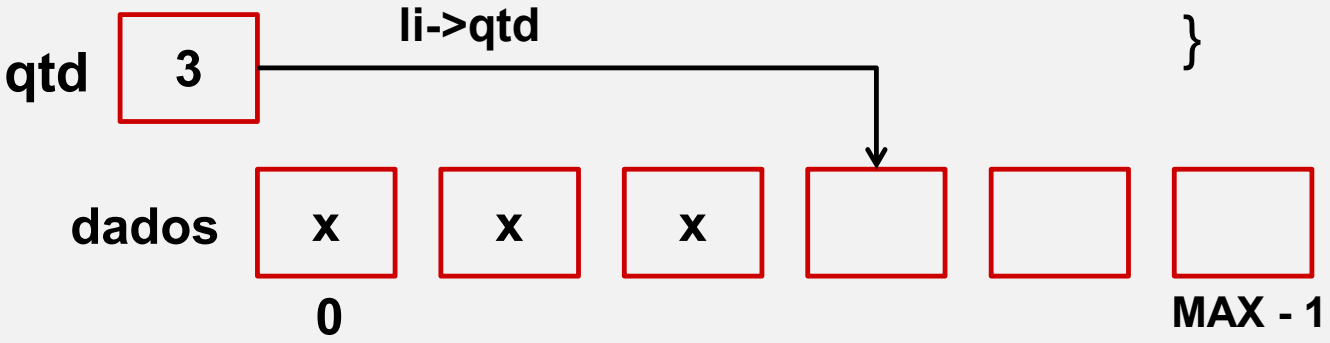
```
int tamanho_lista(Lista* li);
```

//main.c

```
int x = tamanho_lista(li);
```

//Arquivo ListaSequencial.c

```
int tamanho_lista(Lista* li){
    if(li == NULL)
        return -1;
    else
        return li->qtd;
}
```



LISTA ESTÁTICA SEQUENCIAL

Informações básicas

Lista cheia

//Arquivo ListaSequencial.h

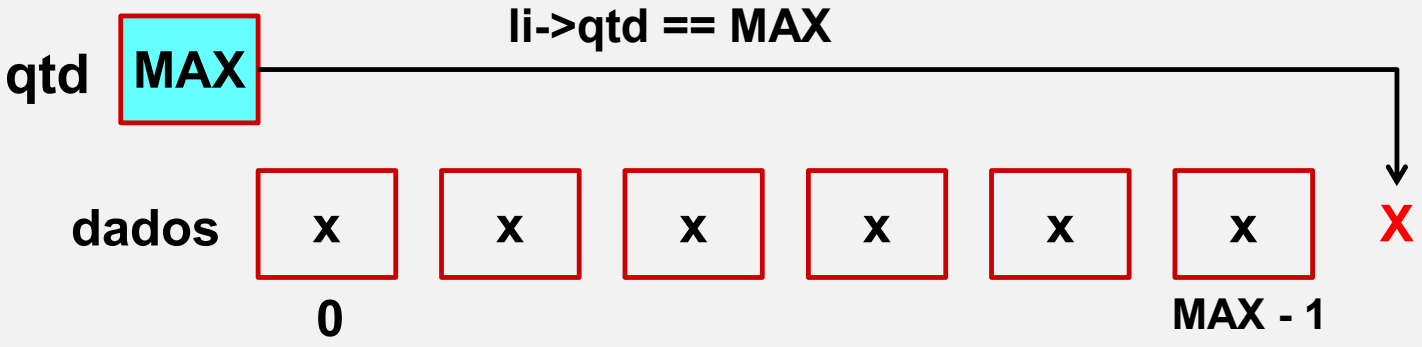
```
int lista_cheia(Lista* li);
```

//main.c

```
if(lista_cheia(li))…
```

//Arquivo ListaSequencial.c

```
int lista_cheia(Lista* li){
    if(li == NULL)
        return -1;
    return (li->qtd == MAX);
}
```



LISTA ESTÁTICA SEQUENCIAL

Informações básicas

Lista vazia

//Arquivo ListaSequencial.h

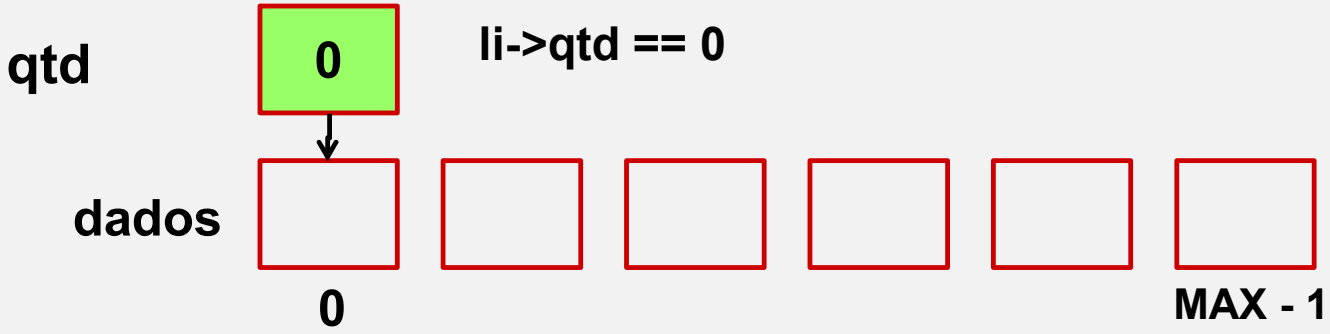
```
int lista_vazia(Lista* li);
```

//main.c

```
if(lista_vazia(li))
```

//Arquivo ListaSequencial.c

```
int lista_vazia(Lista* li){
    if(li == NULL)
        return -1;
    return (li->qtd == 0);
}
```

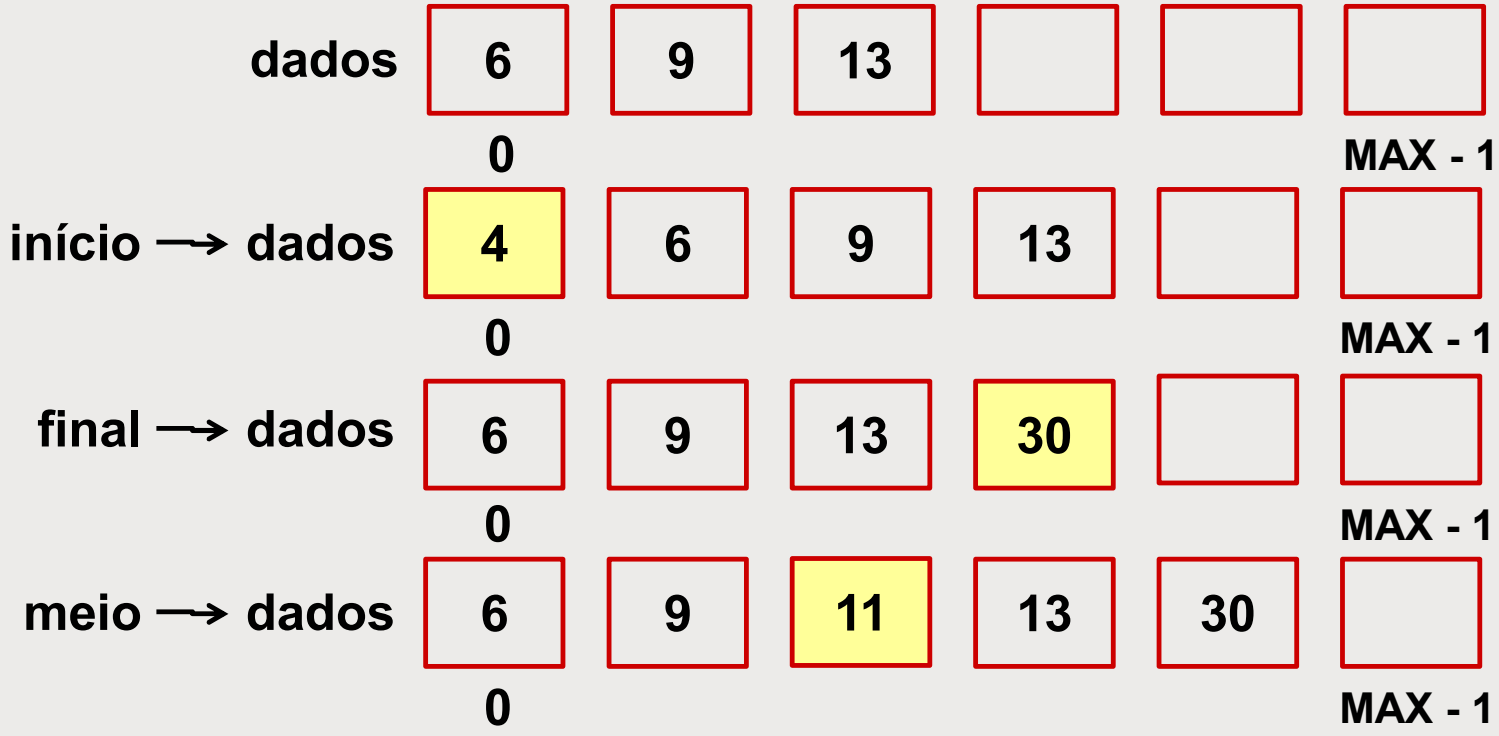


LISTA ESTÁTICA SEQUENCIAL

Inserção

LISTA ESTÁTICA SEQUENCIAL

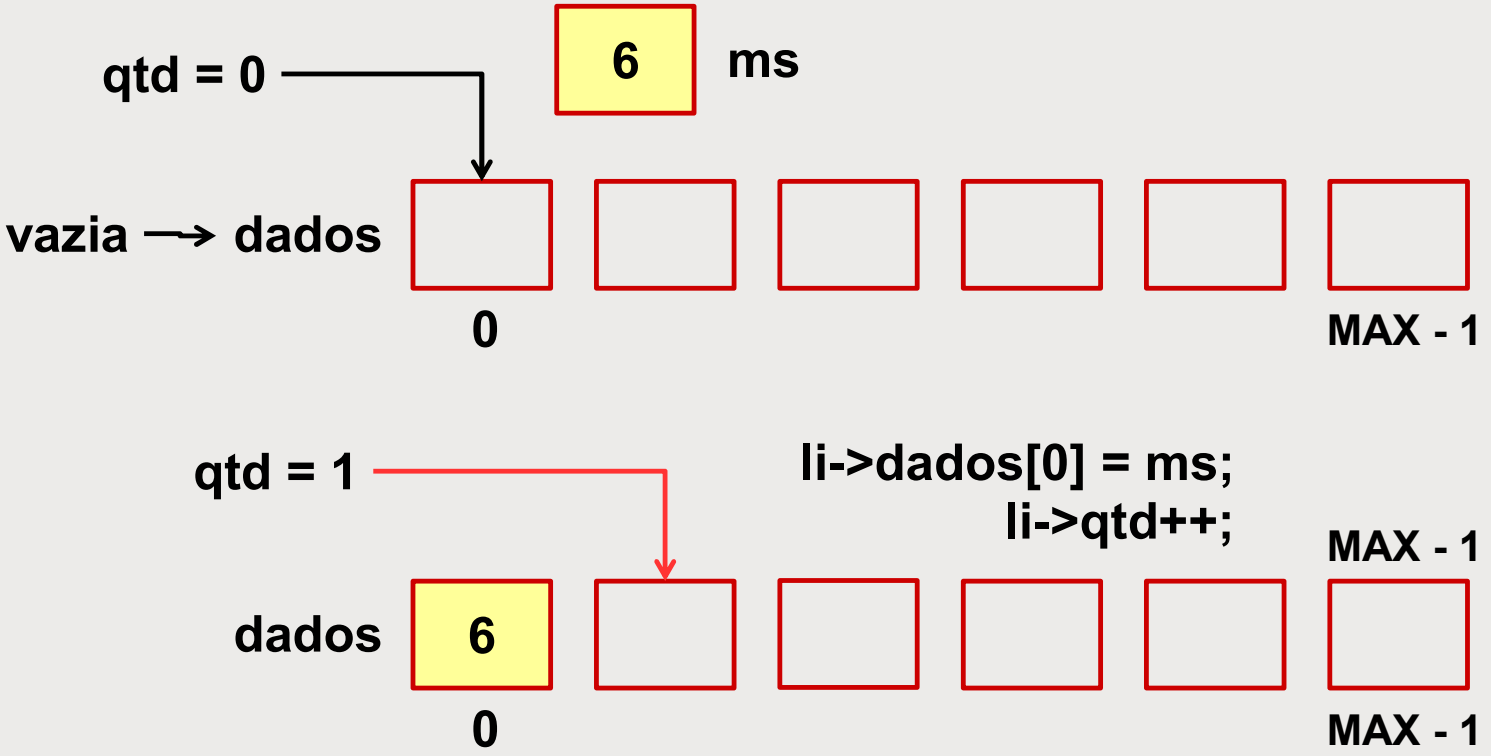
Inserção



LISTA ESTÁTICA SEQUENCIAL

Inserção

Lista vazia



LISTA ESTÁTICA SEQUENCIAL
Inserção, remoção e consulta

Atividade Prática 2: Criar função para

1. inserir elementos na lista (ordenado pelo código)
2. remover primeiro elemento da lista
3. remover último elemento da lista
4. remover elemento através do código
5. consultar elemento pelo índice
6. consultar elemento pelo código