UNINASSAU | ser educacional

Estrutura de Dados

Anderson Lima

Aula 6

Agenda

- Lista sequencial estática - demais operações

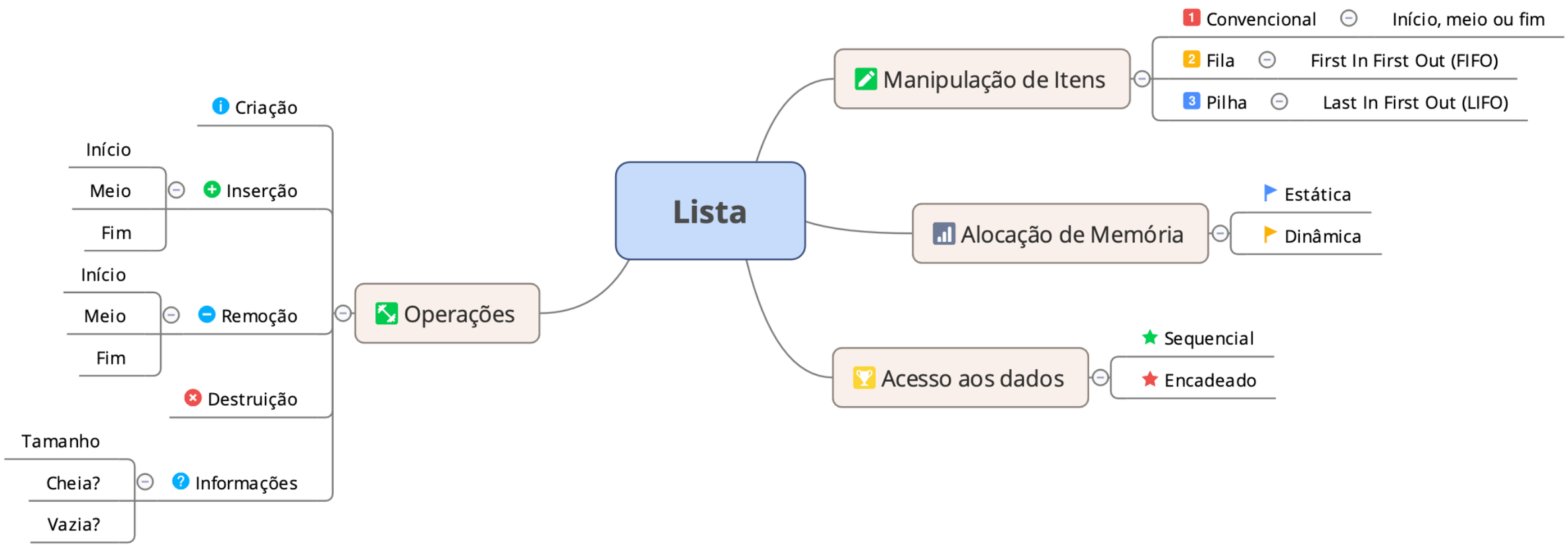
No último episódio…

Reprisando

Lista - Conceito

“Estrutura de dados linear utilizada para armazenar e organizar dados em um computador”. (BACKES, 2016)

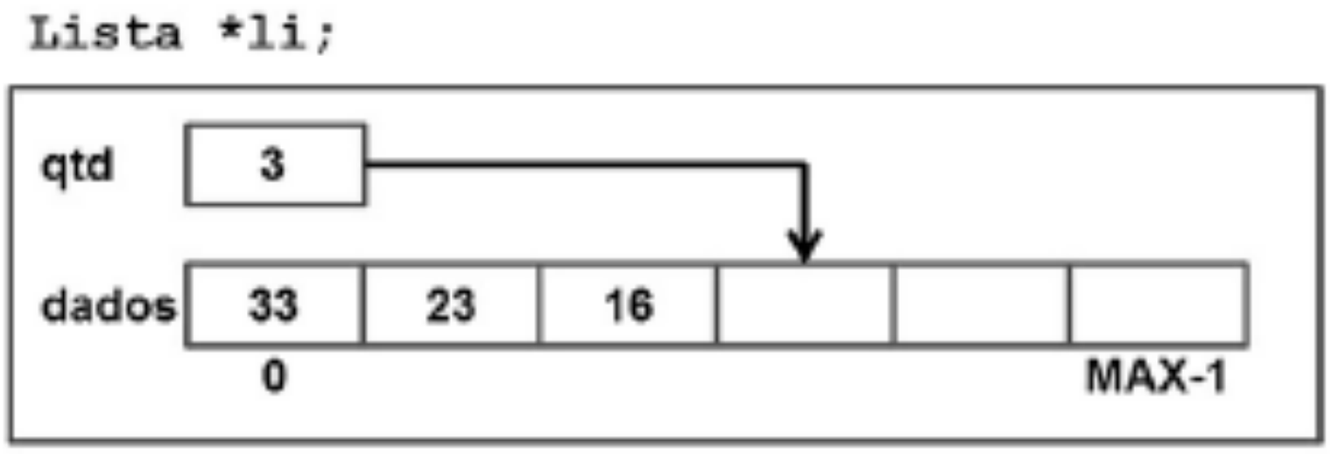
Reprisando


Lista
Manipulação de Itens
1 Convencional
Início, meio ou fim
2 Fila
First In First Out (FIFO)
3 Pilha
Last In First Out (LIFO)
Alocação de Memória
Estática
Dinâmica
Acesso aos dados
Sequencial
Encadeado
Operações
Criação
Início
Meio
Fim
Inserção
Início
Meio
Fim
Remoção
Destruição
Tamanho
Cheia?
Vazia?
Informações

Reprisando

Lista Sequencial Estática

- Tipo de lista que utiliza alocação estática e acesso sequencial dos elementos.
- Implementada usando-se um **array** ou **vetor**.



Reprisando

Lista Sequencial Estática

ListaSequencial.h

Arquivo ListaSequencial.h

```
01  #define MAX 100
02  struct aluno{
03      int matricula;
04      char nome[30];
05      float n1,n2,n3;
06  };
07  typedef struct lista Lista;
08
09  Lista* cria_lista();
10  void libera_lista(Lista* li);
11  int busca_lista_pos(Lista* li, int pos, struct aluno *al);
12  int busca_lista_mat(Lista* li, int mat, struct aluno *al);
13  int insere_lista_final(Lista* li, struct aluno al);
14  int insere_lista_inicio(Lista* li, struct aluno al);
15  int insere_lista_ordenada(Lista* li, struct aluno al);
16  int remove_lista(Lista* li, int mat);
17  int remove_lista_inicio(Lista* li);
18  int remove_lista_final(Lista* li);
19  int tamanho_lista(Lista* li);
20  int lista_cheia(Lista* li);
21  int lista_vazia(Lista* li);
```

Reprisando

Lista Sequencial Estática

Operações

- **cria_lista**
- **libera_lista**
- busca_lista_pos
- busca_lista_mat
- **insere_lista_final**
- insere_lista_inicio

Operações

- insere_lista_ordenada
- remove_lista
- remove_lista_inicio
- remove_lista_final
- **tamanho_lista**
- **lista_cheia**
- **lista_vazia**

Lista Sequencial Estática

Lista Sequencial Estática

Operações Complementares

- busca_lista_pos
- busca_lista_mat
- insere_lista_inicio
- insere_lista_ordenada
- remove_lista
- remove_lista_inicio
- remove_lista_final

Dividir a sala em 3 grupos

Lista Sequencial Estática

GRUPO 1

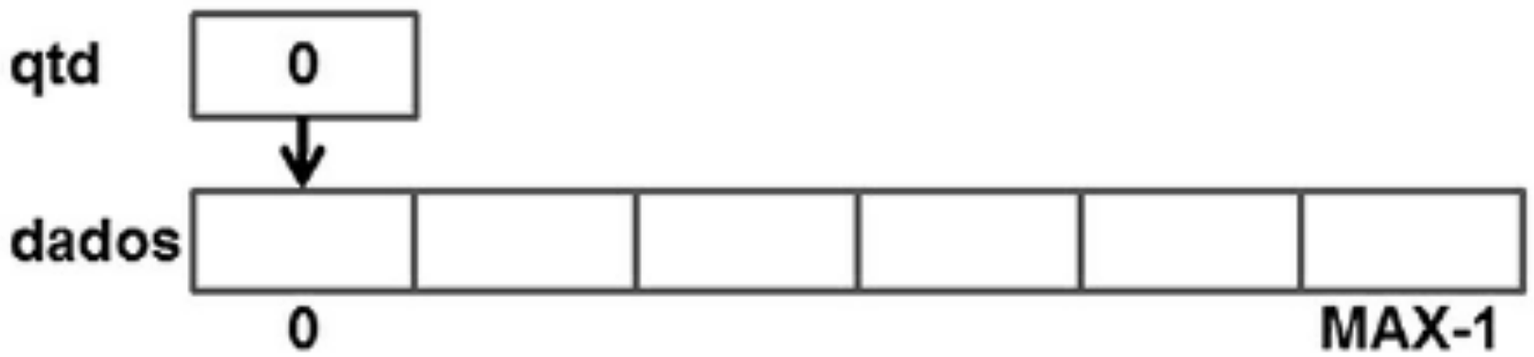
- busca_lista_pos
- insere_lista_inicio

GRUPO 2

- busca_lista_mat
- remove_lista_final

GRUPO 3

- remove_lista
- remove_lista_inicio



Lista Sequencial Estática

GRUPO 1

```
int busca_lista_pos(Lista* li, int pos, struct aluno *info);
int insere_lista_inicio(Lista* li, struct aluno info);
```

GRUPO 2

```
int busca_lista_mat(Lista* li, int mat, struct aluno *info);
int remove_lista_final(Lista* li);
```

GRUPO 3

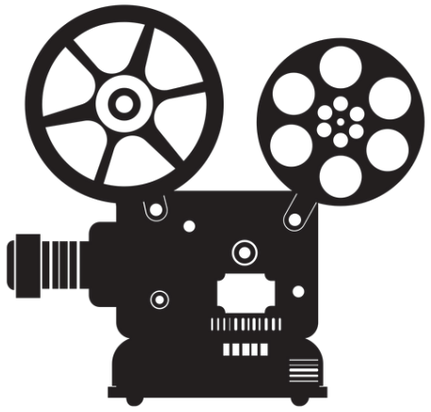
```
int remove_lista(Lista* li, int mat);
int remove_lista_inicio(Lista* li);
```

15 min

Lista Sequencial Estática

GRUPO 1

```
int busca_lista_pos(Lista* li, int pos, struct aluno *info);
int insere_lista_inicio(Lista* li, struct aluno info);
```



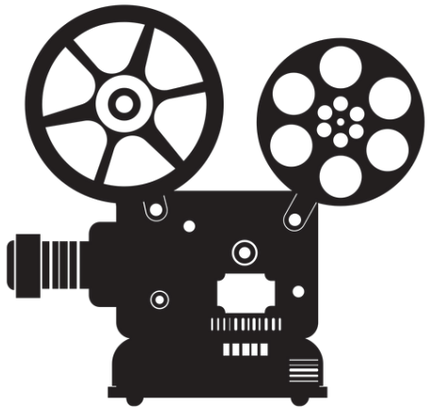
APRESENTAÇÃO

5 min

Lista Sequencial Estática

GRUPO 2

```
int busca_lista_mat(Lista* li, int mat, struct aluno *info);
int remove_lista_final(Lista* li);
```



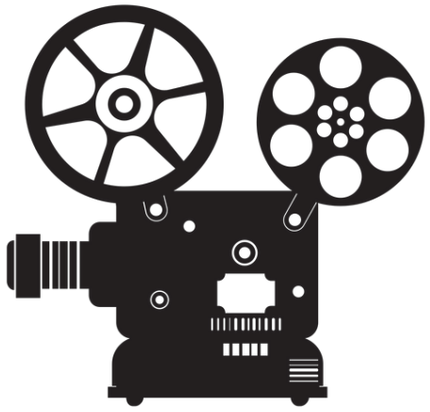
APRESENTAÇÃO

5 min

Lista Sequencial Estática

GRUPO 3

```
int remove_lista(Lista* li, int mat);
int remove_lista_inicio(Lista* li);
```



APRESENTAÇÃO

5 min

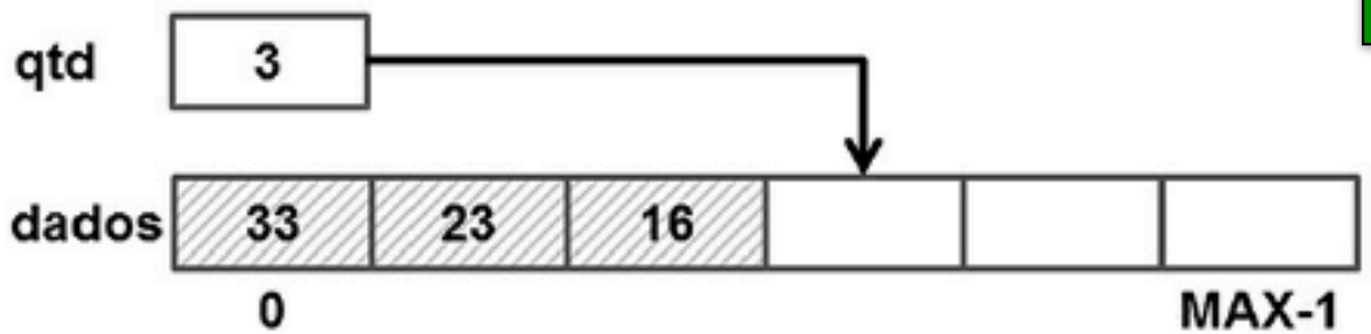
Gabarito

UNINASSAU

Lista Sequencial Estática

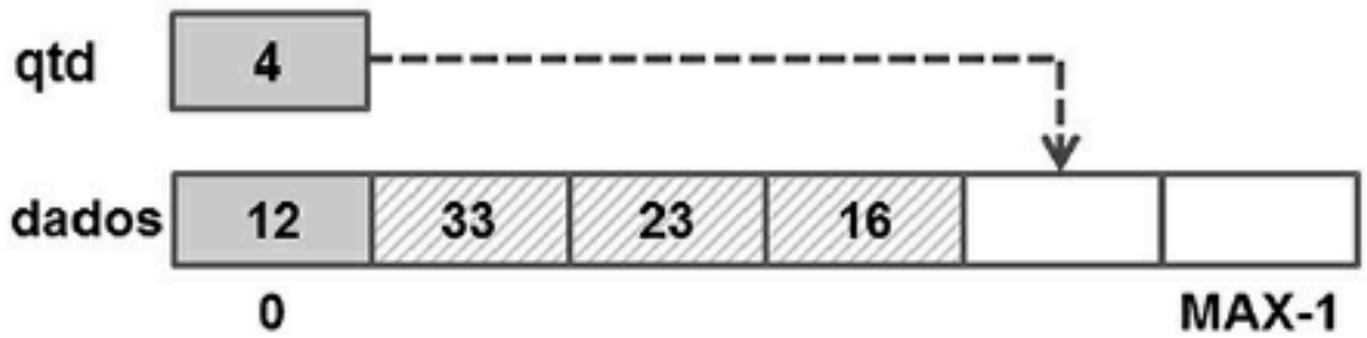
Inserindo elemento no início

GRUPO 1



Desloca os elementos uma posição para frente e insere:

```
for(i=li->qtd-1; i>=0; i--)
   li->dados[i+1] = li->dados[i];
li->dados[0] = al;
li->qtd++;
```



Lista Sequencial Estática

Inserindo elemento no início

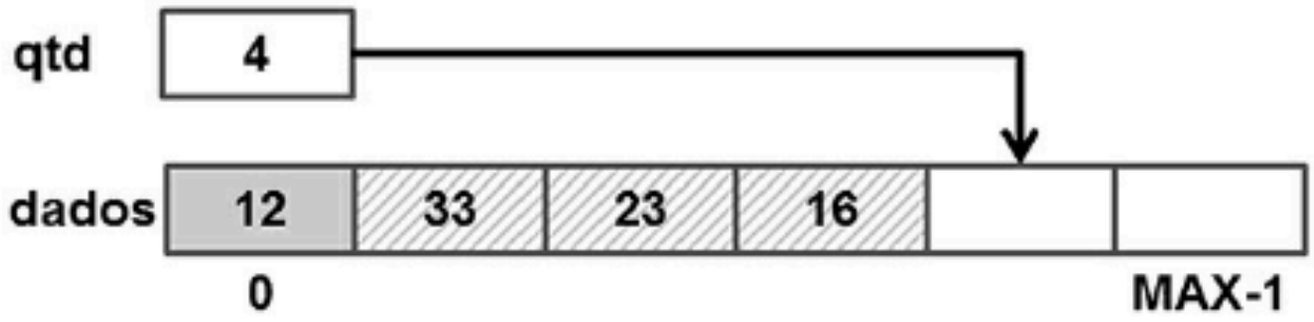
Inserindo um elemento no início da lista

```
01  int insere_lista_inicio(Lista* li, struct aluno al){
02      if(li == NULL)
03          return 0;
04      if(li->qtd == MAX)//lista cheia
05          return 0;
06      int i;
07      for(i=li->qtd-1; i>=0; i--)
08          li->dados[i+1] = li->dados[i];
09      li->dados[0] = al;
10      li->qtd++;
11      return 1;
12  }
```

Lista Sequencial Estática

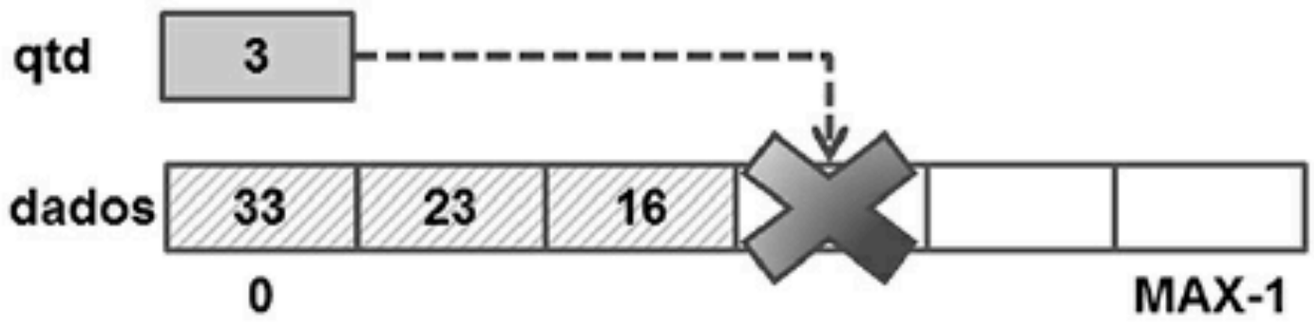
Removendo elemento do início

GRUPO 3



Desloca os elementos uma posição para trás:

```
for(k=0; k< li->qtd-1; k++)
    li->dados[k] = li->dados[k+1];
li->qtd--;
```



Lista Sequencial Estática

Removendo elemento do início

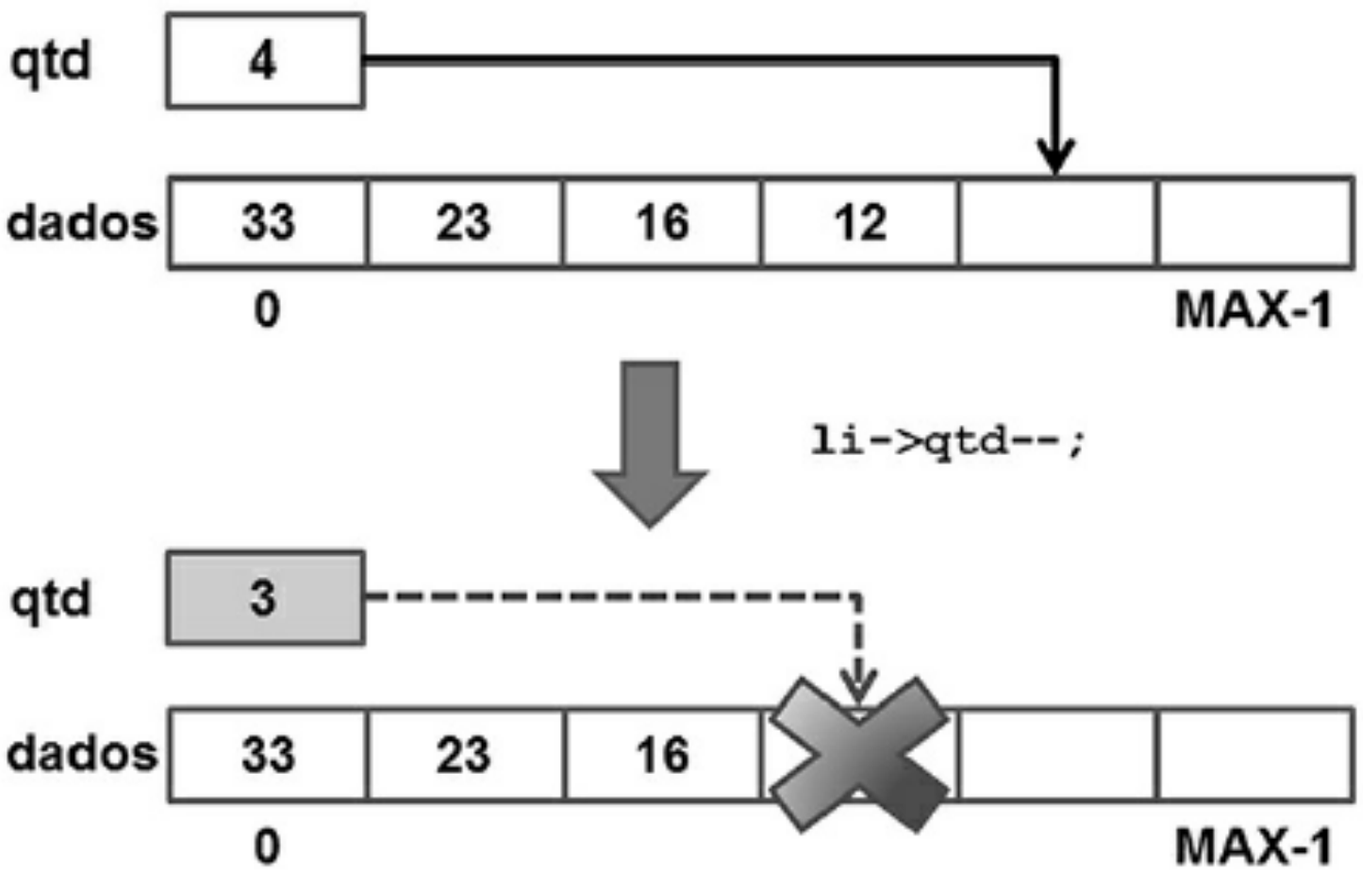
Removendo um elemento do início da lista

```
01  int remove_lista_inicio(Lista* li){
02      if(li == NULL)
03          return 0;
04      if(li->qtd == 0)//lista vazia
05          return 0;
06      int k = 0;
07      for(k=0; k< li->qtd-1; k++)
08          li->dados[k] = li->dados[k+1];
09      li->qtd--;
10      return 1;
11  }
```

Lista Sequencial Estática

Removendo elemento do final

GRUPO 2



Lista Sequencial Estática

Removendo elemento do final

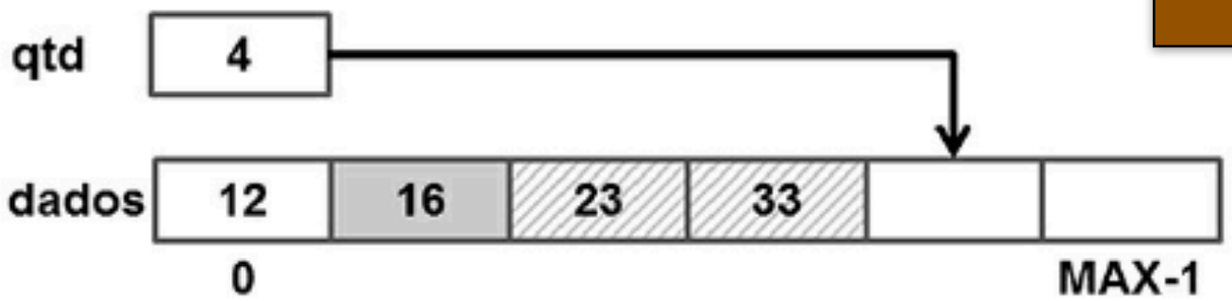
Removendo um elemento do final da lista

```
01  int remove_lista_final(Lista* li){
02      if(li == NULL)
03          return 0;
04      if(li->qtd == 0)//lista vazia
05          return 0;
06      li->qtd--;
07      return 1;
08  }
```

Lista Sequencial Estática

Removendo um elemento específico

GRUPO 3

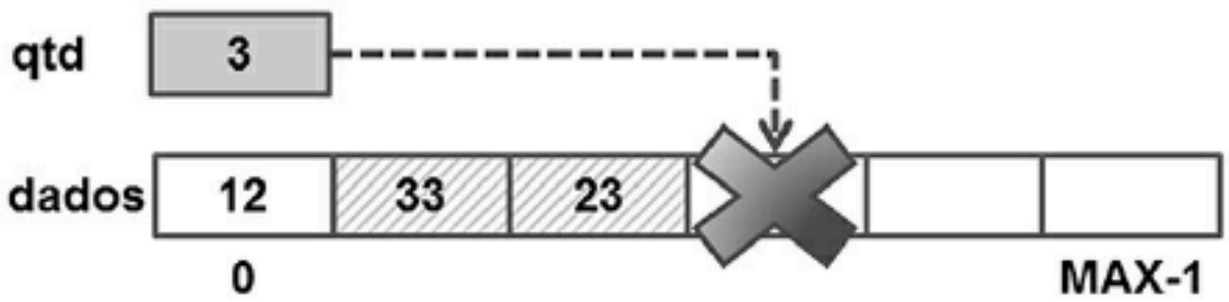


Procura elemento a ser removido:

```
while(i<li->qtd && li->dados[i].matricula != mat)
    i++;
```

Desloca os elementos uma posição para trás:

```
for(k=i; k< li->qtd-1; k++)
    li->dados[k] = li->dados[k+1];
li->qtd--;
```



Lista Sequencial Estática

Removendo um elemento específico

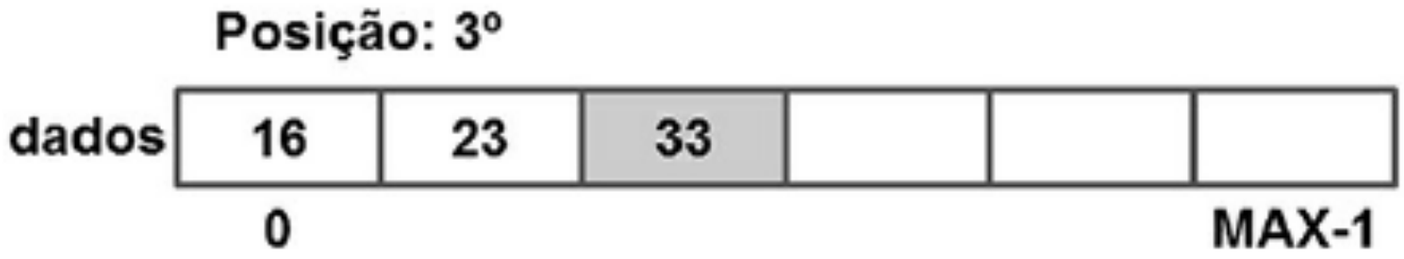
Removendo um elemento específico da lista

```
01  int remove_lista(Lista* li, int mat){
02      if(li == NULL)
03          return 0;
04      if(li->qtd == 0)//lista vazia
05          return 0;
06      int k,i = 0;
07      while(i<li->qtd && li->dados[i].matricula != mat)
08          i++;
09      if(i == li->qtd)//elemento não encontrado
10          return 0;
11
12      for(k=i; k< li->qtd-1; k++)
13          li->dados[k] = li->dados[k+1];
14      li->qtd--;
15      return 1;
16  }
```

Lista Sequencial Estática

Busca um elemento por posição

GRUPO 1




```
*al = li->dados[pos-1];
```

Busca um elemento por posição

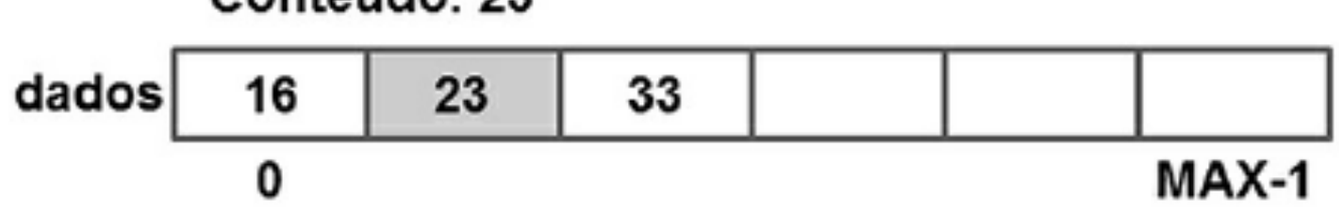
```
01  int busca_lista_pos(Lista* li,int pos,struct aluno *al){
02      if(li == NULL || pos <= 0 ||  pos > li->qtd)
03          return 0;
04      *al = li->dados[pos-1];
05      return 1;
06  }
```

Lista Sequencial Estática

Busca um elemento por matrícula

GRUPO 2

Conteúdo: 23



Busca pelo elemento:

```
while(i<li->qtd && li->dados[i].matricula != mat)
        i++;
```

Achou o elemento:

```
*al = li->dados[i];
```

Lista Sequencial Estática

Busca um elemento por matrícula

Busca um elemento por conteúdo

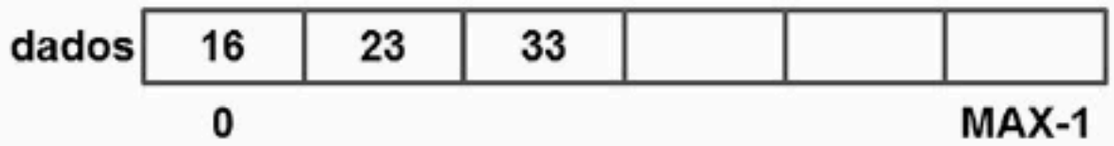
```
01  int busca_lista_mat(Lista* li, int mat, struct aluno *al){
02      if(li == NULL)
03          return 0;
04      int i = 0;
05      while(i<li->qtd && li->dados[i].matricula != mat)
06          i++;
07      if(i == li->qtd)//elemento não encontrado
08          return 0;
09
10      *al = li->dados[i];
11      return 1;
12  }
```

Lista Sequencial Estática

Inserindo elemento em lista ordenada

BÔNUS

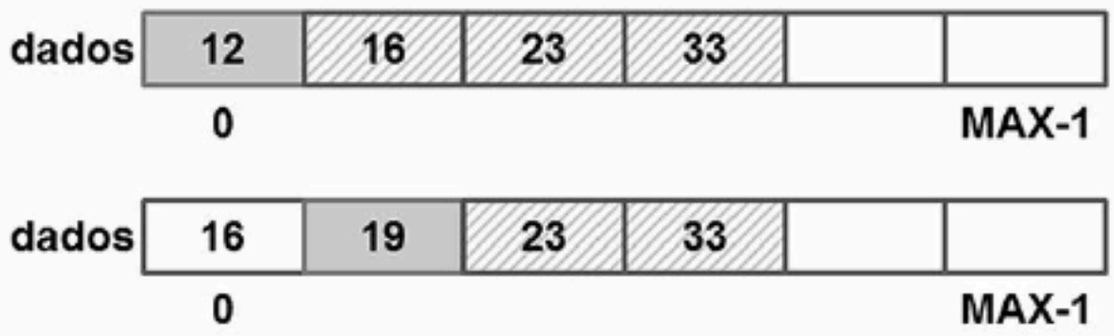
Lista inicial
Busca onde Inserir:




```
int k,i = 0;
while(i < li->qtd && li->dados[i].matricula < al.matricula)
   i++;
```

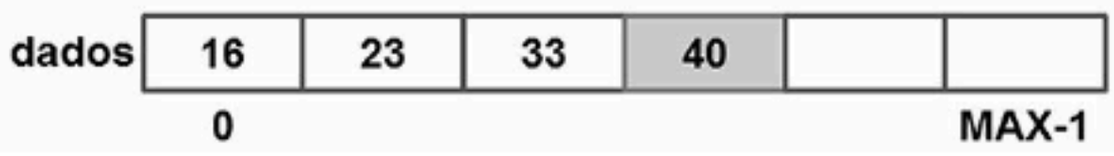
Inserção no início ou no meio: desloca elementos

```
for(k=li->qtd-1; k >= i; k--)
  li->dados[k+1] = li->dados[k];
```



Inserir elemento

```
li->dados[i] = al;
li->qtd++;
```



Lista Sequencial Estática

Inserindo elemento em lista ordenada

Inserindo um elemento de forma ordenada na lista

```
01  int insere_lista_ordenada(Lista* li,struct aluno al){
02    if(li == NULL)
03        return 0;
04    if(li->qtd == MAX)//lista cheia
05        return 0;
06    int k,i = 0;
07    while(i<li->qtd && li->dados[i].matricula < al.matricula)
08        i++;
09
10    for(k=li->qtd-1; k >= i; k--)
11        li->dados[k+1] = li->dados[k];
12
13    li->dados[i] = al;
14    li->qtd++;
15    return 1;
16  }
```

Cenas do próximo capítulo…

O que nós vimos hoje?

- Lista sequencial estática - Operações Complementares
  - busca_lista_pos
  - busca_lista_mat
  - insere_lista_inicio
  - insere_lista_ordenada
  - remove_lista
  - remove_lista_inicio
  - remove_lista_final

Na próxima aula...

- Prática de lista sequencial estática - 2a parte

Anderson Lima

andclima@gmail.com