UNINASSAU | ser educacional

Estrutura de Dados

Anderson Lima

Aula 4

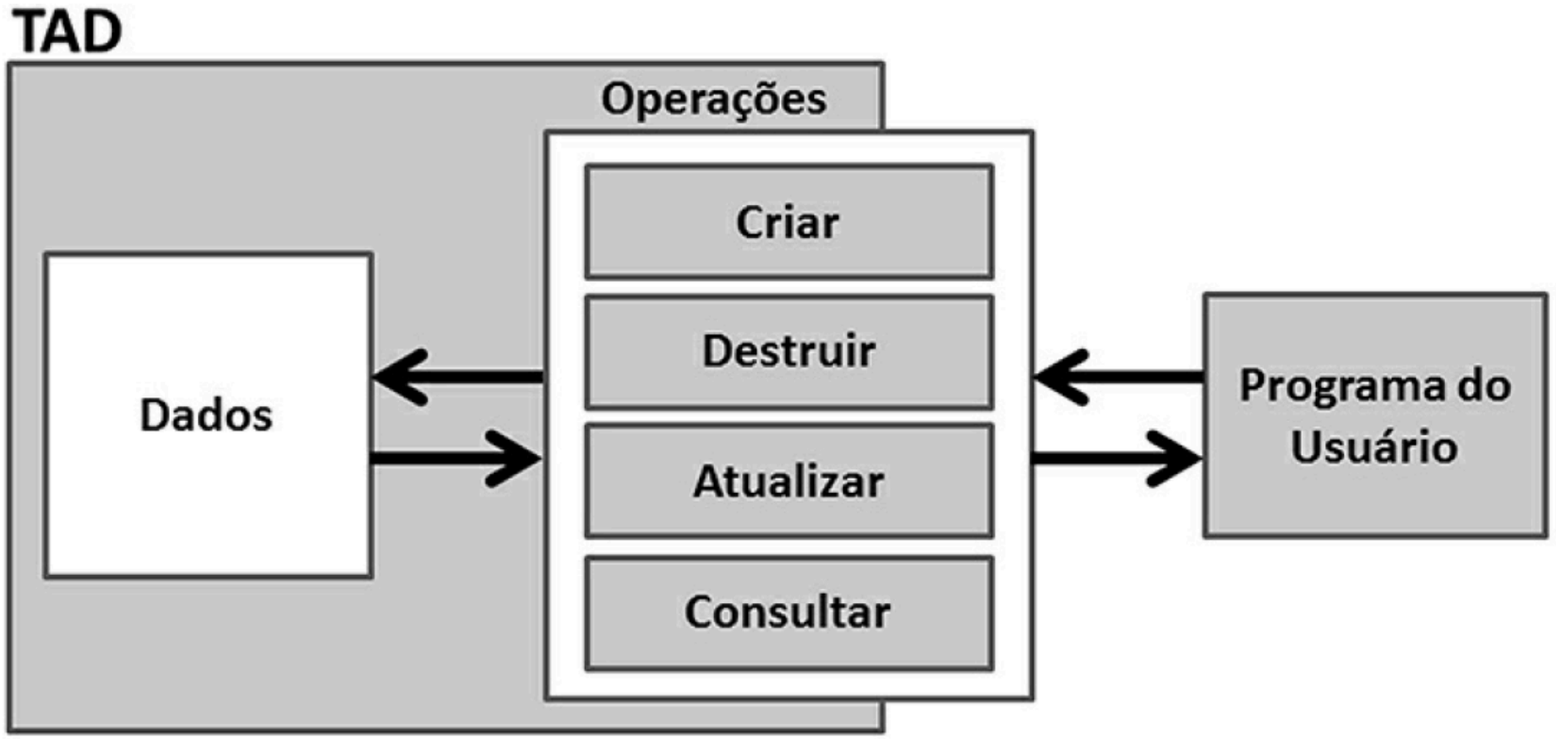
Agenda

- Lista - conceitos
- Lista sequencial estática

No último episódio…

Reprisando

- Prática de Tipo Abstrato de Dados



Reprisando

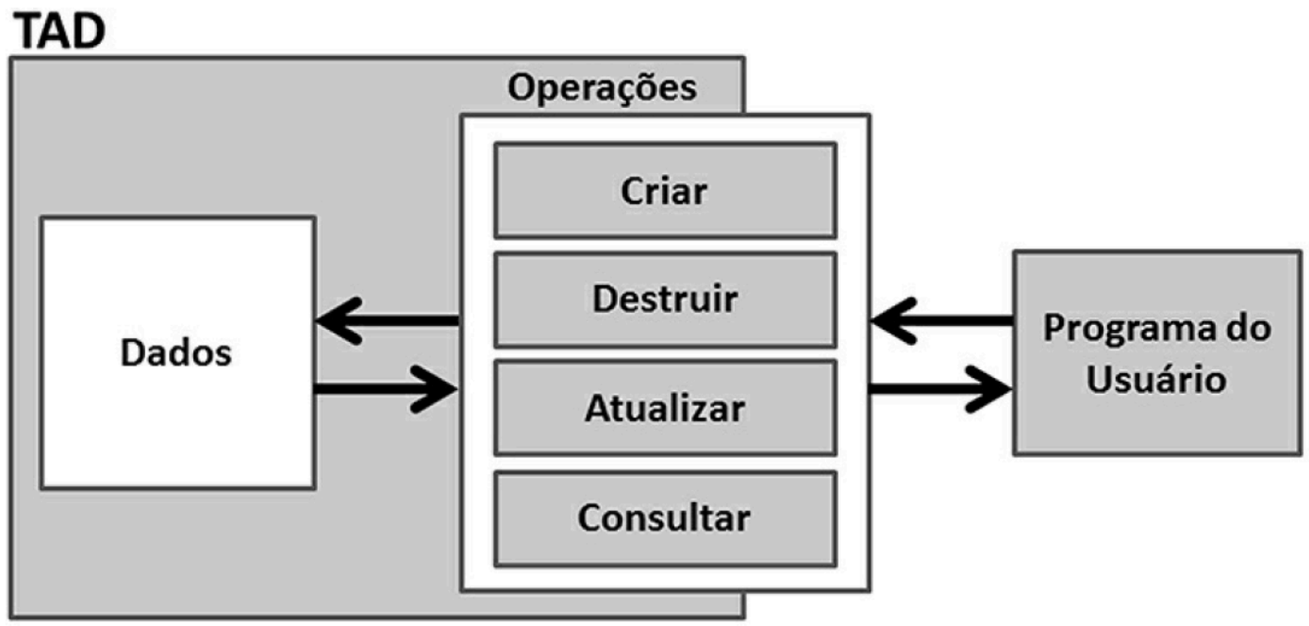
- Prática de Tipo Abstrato de Dados

Dados

- `Ponto`

Operações

- `Ponto_cria`
- `Ponto_libera`
- `Ponto_acessa`
- `Ponto_atribui`
- `Ponto_distancia`



Reprisando

- Prática de Tipo Abstrato de Dados

Arquivo Ponto.h

```
01  typedef struct ponto Ponto;
02  //Cria um novo ponto
03  Ponto* Ponto_cria(float x, float y);
04  //Libera um ponto
05  void Ponto_libera(Ponto* p);
06  //Acessa os valores "x" e "y" de um ponto
07  int Ponto_acessa(Ponto* p, float* x, float* y);
08  //Atribui os valores "x" e "y" a um ponto
09  int Ponto_atribui(Ponto* p, float x, float y);
10  //Calcula a distância entre dois pontos
11  float Ponto_distancia(Ponto* p1, Ponto* p2);
```

Arquivo Ponto.c

```
01  #include <stdlib.h>
02  #include <math.h>
03  #include "Ponto.h" //inclui os Protótipos
04  struct ponto{//Definição do tipo de dados
05    float x;
06    float y;
07  };
```

Listas - Conceitos

Listas

Conceito

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

Listas

Tipos de Listas

Quanto à <u>manipulação de itens</u>

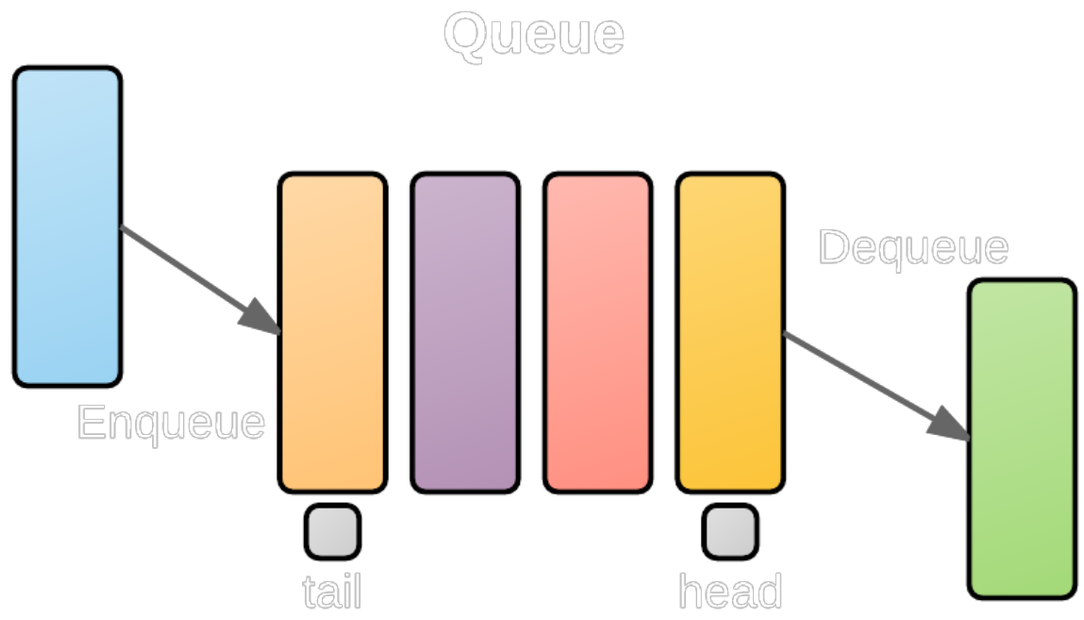
- **Lista Convencional**
  Os elementos pode ser inseridos ou removidos de qualquer parte da estrutura.

Tipos de Listas

Quanto à <u>manipulação de itens</u>

- **Fila (FIFO)**
  Os elementos só podem ser inseridos no final e acessados ou removidos do início.

First **I**n, **F**irst **O**ut
Primeiro a entrar, primeiro a sair.

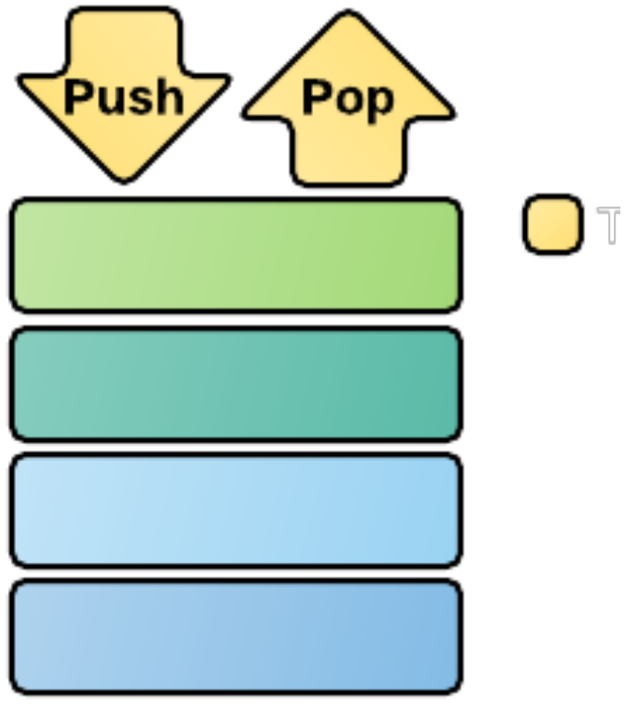


Tipos de Listas

Quanto à <u>manipulação de itens</u>

- **Pilha (LIFO)**
  Os elementos só podem ser inseridos, acessados ou removidos no final da estrutura.

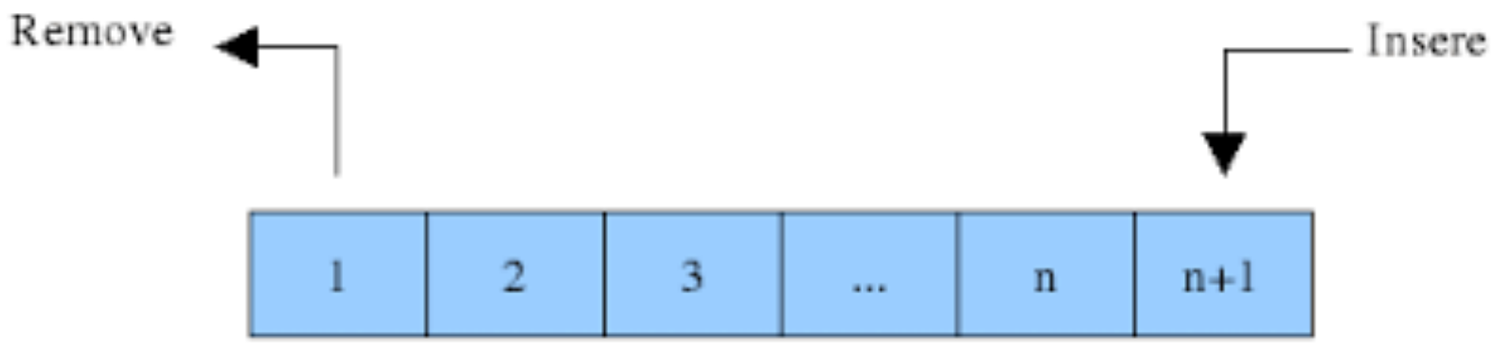
  Last **I**n, **F**irst **O**ut
  Último a entrar, primeiro a sair.



Tipos de Listas

Quanto à <u>alocação de memória</u>

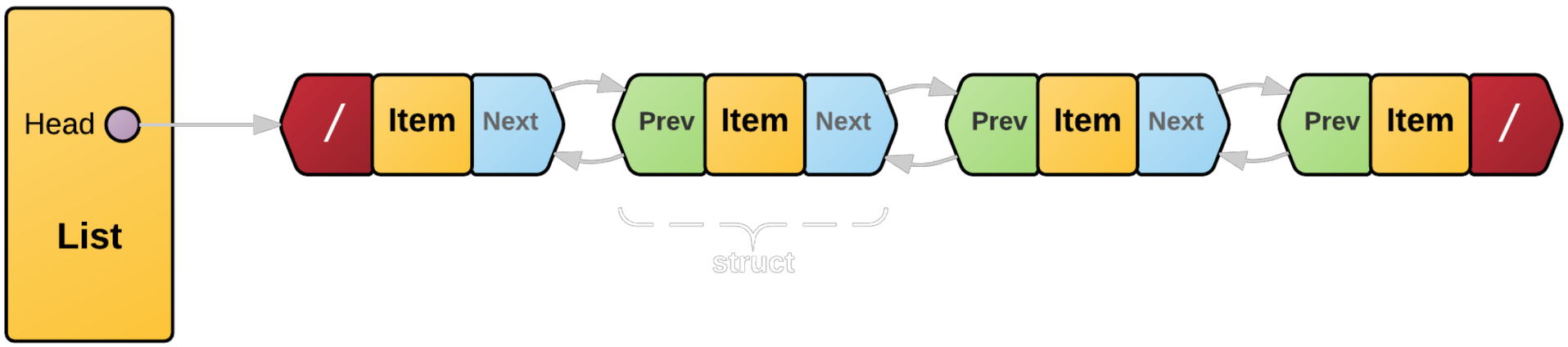
- **Alocação estática**
  A alocação é definida no momento da compilação do programa, sendo necessário estabelecer o número máximo que a lista deverá possuir.



Tipos de Listas

Quanto à <u>alocação de memória</u>

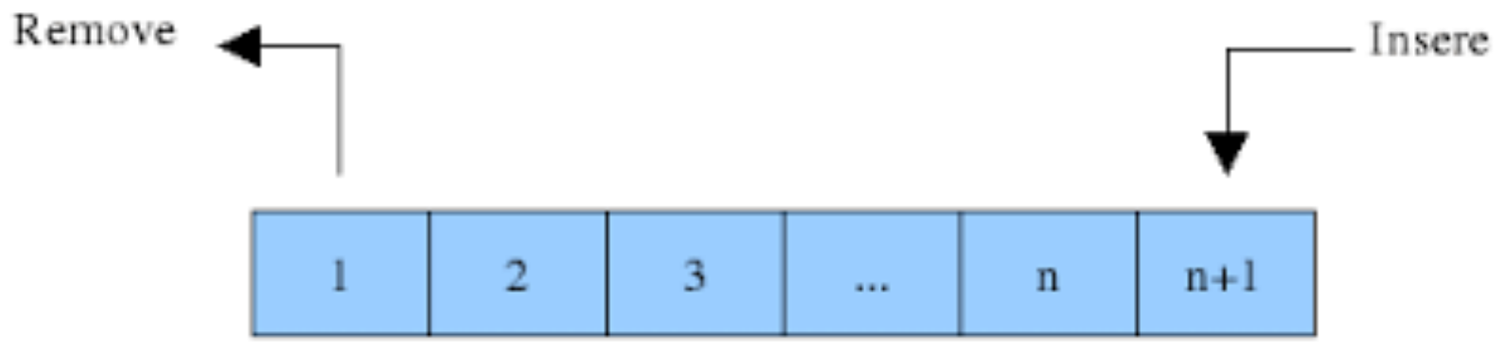
- **Alocação dinâmica**
  A alocação de memória é feita durante o tempo de execução, crescendo a estrutura à medida que novos elementos são adicionados.



Tipos de Listas

Quanto ao <u>acesso aos dados</u>

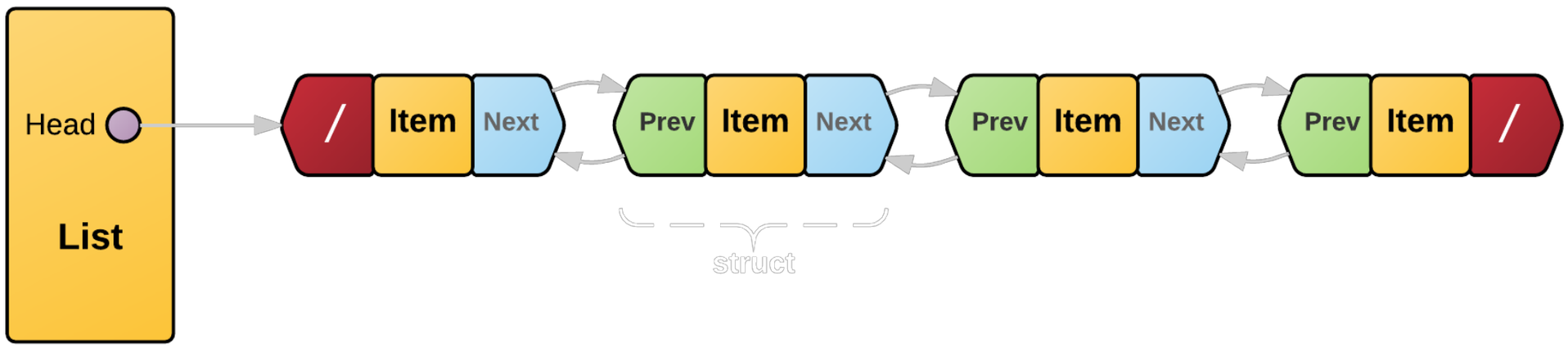
- **Acesso Sequencial**
  Os elementos são armazenados de forma consecutiva na memória (array ou vetor).



Tipos de Listas

Quanto ao <u>acesso aos dados</u>

- **Acesso Encadeado**
  Os elementos são armazenados em áreas distintas da memória, sendo necessário que ele possua o endereço do próximo elemento, de modo a permitir sua localização.



Operações

Principais operações de uma lista

- Criação da lista
- Inserção de elemento
  - Início
  - Final
  - No meio
- Remoção de elemento
  - Início
  - Final
  - No meio
- Destruição da lista

Operações

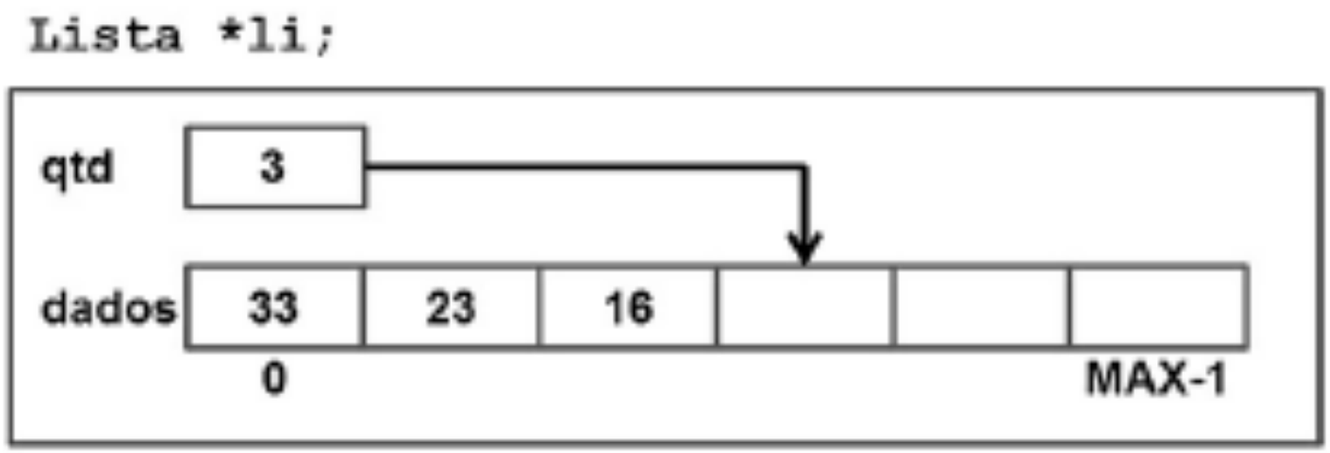
Principais operações de uma lista

- Informações
  - Tamanho
  - Está cheia?
  - Está vazia?

Listas Sequencial Estática

Lista Sequencial Estática

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



Lista Sequencial Estática

Vantagens

- Acesso rápido aos elementos
- Tempo constante de acesso aos elementos
- Facilidade para modificar as informações

Desvantagens

- Definição prévia do tamanho do array
- Dificuldade de inserir elementos no meio da lista

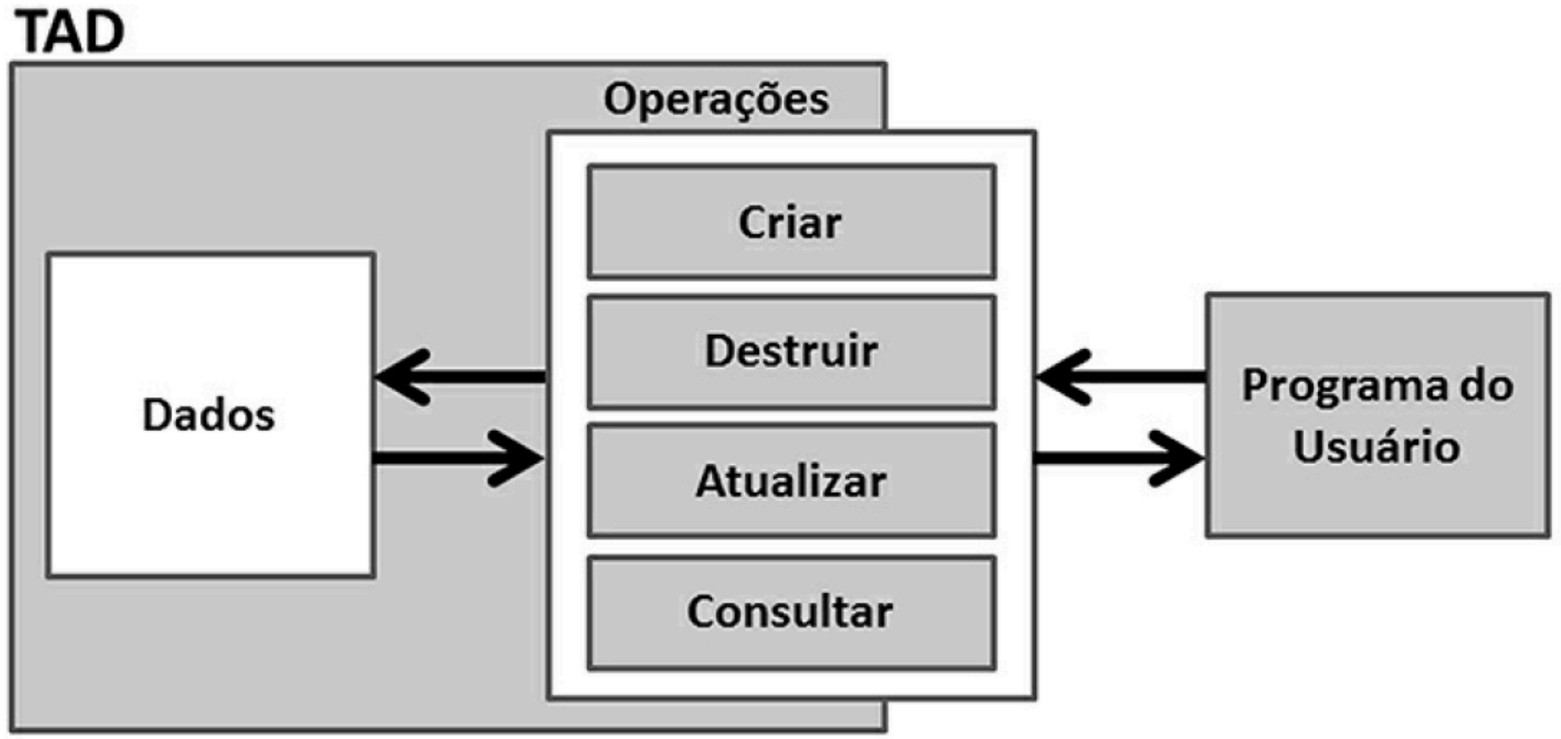
Lista Sequencial Estática

Indicação

- Listas pequenas
- Inserção e remoção no final da lista
- Tamanho máximo bem definido
- A operação de busca é a mais frequente

Lista Sequencial Estática

Implementação



Lista Sequencial Estática

Implementação

Dados

- constante MAX
- Tipo de dados a ser armazenado
- Definição do tipo LISTA

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

Header

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);
```

Lista Sequencial Estática

Implementação

Arquivo ListaSequencial.c

```
01  #include <stdio.h>
02  #include <stdlib.h>
03  #include "ListaSequencial.h" //inclui os protótipos
04  //Definição do tipo lista
05  struct lista{
06      int qtd;
07      struct aluno dados[MAX];
08  };
```

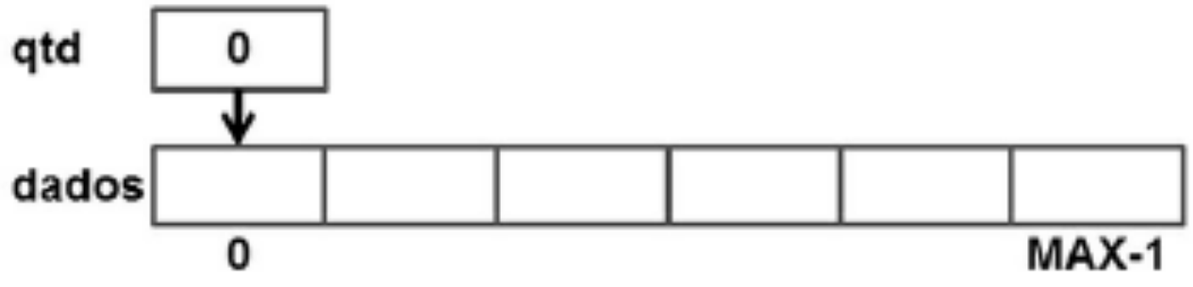
ListaSequencial.c

Lista Sequencial Estática

Criando uma lista

Criando uma lista

```
01  Lista* cria_lista(){
02      Lista *li;
03      li = (Lista*) malloc(sizeof(struct lista));
04      if(li != NULL)
05          li->qtd = 0;
06      return li;
07  }
```



Lista Sequencial Estática

Destruindo uma lista

Destruindo uma lista

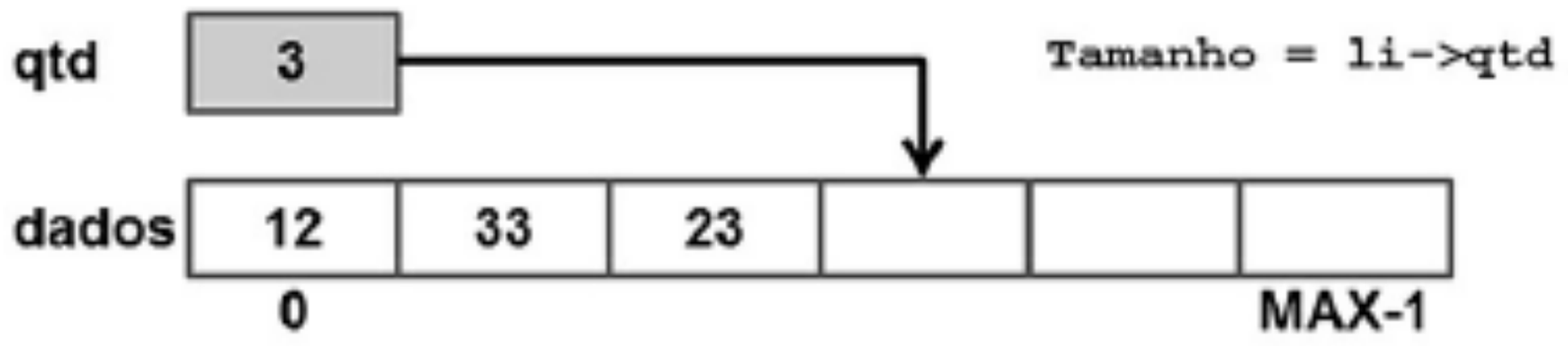
```
01  void libera_lista(Lista* li){
02      free(li);
03  }
```

Lista Sequencial Estática

Informações Básicas - Tamanho da Lista

Tamanho da lista

```
01  int tamanho_lista(Lista* li){
02      if(li == NULL)
03          return -1;
04      else
05          return li->qtd;
06  }
```

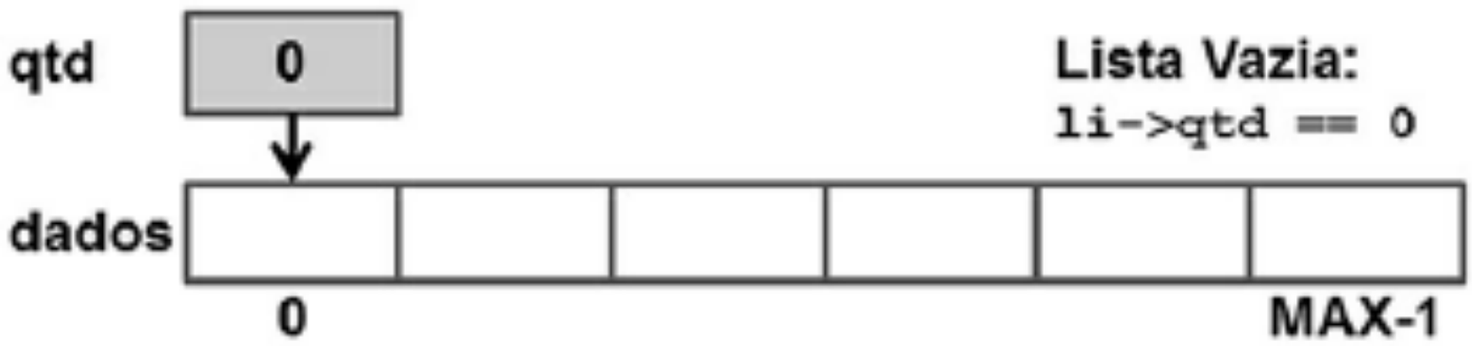


Lista Sequencial Estática

Informações Básicas - Lista Vazia

Retornando se a lista está vazia

```
01  int lista_vazia(Lista* li){
02      if(li == NULL)
03          return -1;
04      return (li->qtd == 0);
05  }
```

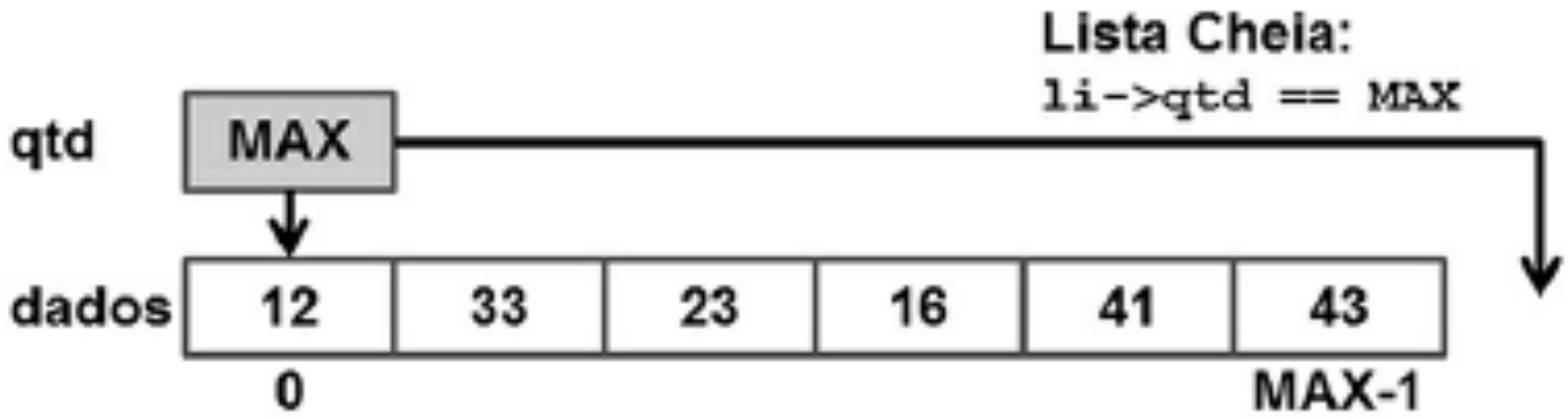


Lista Sequencial Estática

Informações Básicas - Lista Cheia

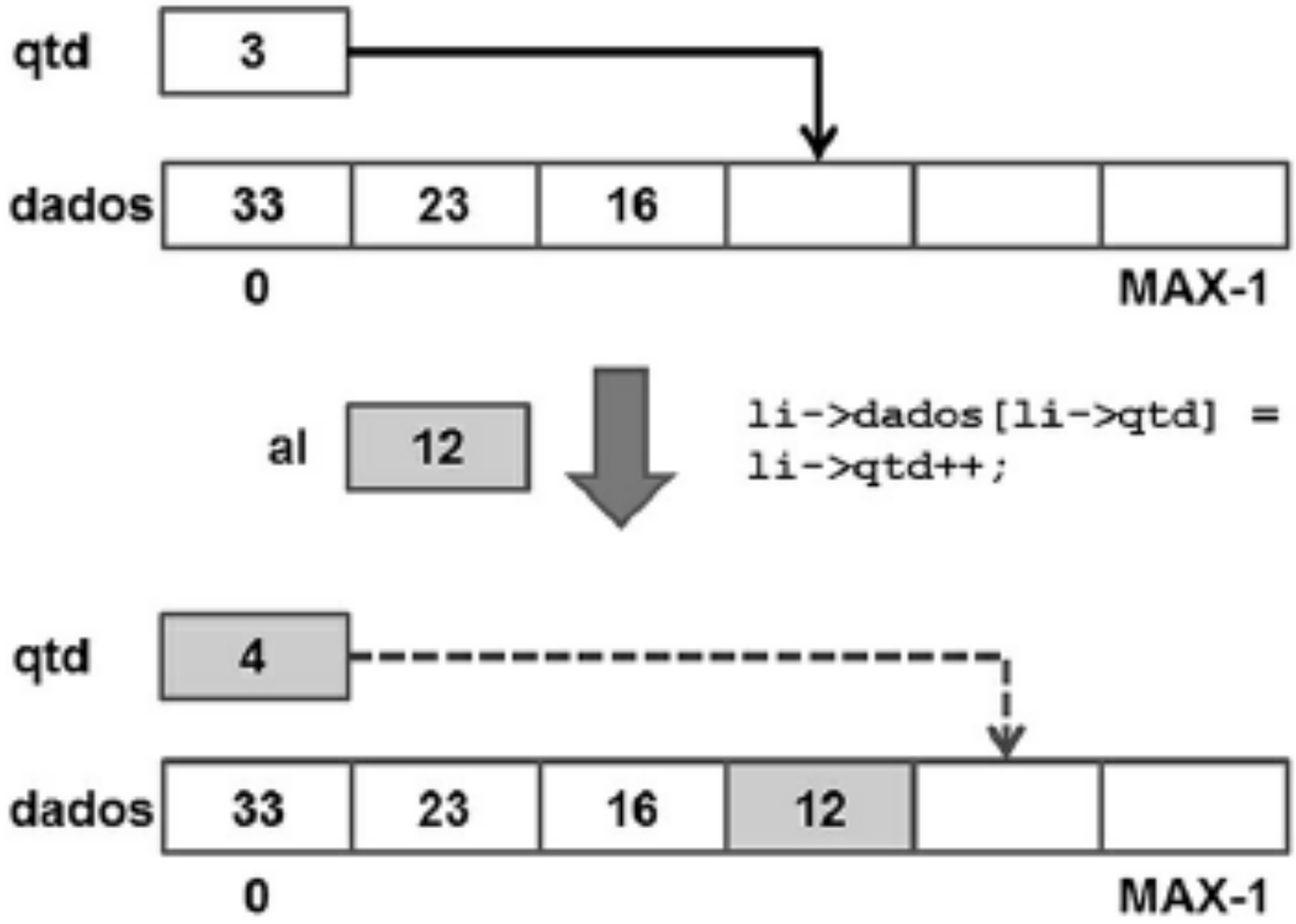
Retornando se a lista está cheia

```
01  int lista_cheia(Lista* li){
02      if(li == NULL)
03          return -1;
04      return (li->qtd == MAX);
05  }
```



Lista Sequencial Estática

Inserindo elemento no final da Lista



Lista Sequencial Estática

Inserindo elemento no final da Lista

Inserindo um elemento no final da lista

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

Cenas do próximo capítulo…

O que nós vimos hoje?

- Conceitos básicos de lista
- Lista sequencial estática

Na próxima aula...

- Prática de lista sequencial estática - 1a parte

Anderson Lima

andclima@gmail.com