Curso: Ciência da Computação

Disciplina: Estrutura de Dados 1

Professor: Clayton Zambon

centro universitário unifacvest

3. Lista

3.0.Definição;
3.1.Estática Sequencial;

3. Lista
3.0. Definição

Lista

- Podemos Inserir elementos em uma lista, remover, procurar, ordenar;
- Exemplo: lista de músicas, lista de compras;

APARELHO AUDIOVISUAL DE KARAOKÊ

LISTA DE MÚSICAS DO iVideokê

Ordem alfabética pelo idioma / cantor

INTÉRPRETE	CÓD.	TOM	TÍTULO	AUTOR	INÍCIO DA LETRA	PACOTE
Jota Quest	1234	D	SEIS E TRINTA	M.Buzelin-M.Túlio-PFonseca-PJ-R.Flausino	Pareço contigo normal e do avesso	16B
Jota Quest	1439	E	ÚNICO OLHAR	Jota Quest-G.Mesquita	Então me coloco à sua frente	17C
Jota Quest	2915	E	MAIS PERTO DE MIM	Jota Quest-Giovane Mesquita-Fernando Eugênio	Quem sabe um dia ainda consiga	22D
Jota Quest	15381	Gm	MANDOU BEM	Brian-Gigi-Barnes-PJ-Lara-Rog-Fons-Flaus	Uh uh... Você mandou bem cuidou de	25B
Jota Quest	15513	B	TUDO ESTÁ PARADO	Gessinger-Buzelin-Lara-Fonseca-PJ-Flausino	Ô ô ô ô... Tudo está parado diz aê	26A
Jota Quest	15640	G	WAITING FOR YOU (SHINE ON, SHINE ON)	J.Quest-J.Barnes-Q.Space	Shine on shine on luz do sol um novo dia	26D
Jota Quest	4112	G	FÁCIL	Jota Quest	Tudo é tão bom e azul e calmo como s	3E
Jota Quest	4213	G	SEMPRE ASSIM	M.Túlio-R.Flausino	Sete e quinze eu acordo e começo a	3G
Jota Quest	4287	Dbm	O VENTO	M.Buzelin	Voe por todo mar e volte aqui	4A
Jota Quest	4489	Am	OXIGÊNIO	Rogério Flausino	Mesmo com a fumaça Dá para ver	4F
Jota Quest	6032	G	TELE-FOME	Paulinho Pedra Azul	Não alimento amor Por telefone	4G
Jota Quest	6107	A	O QUE EU TAMBÉM NÃO ENTENDO	R.Flausino-F.Mello	Essa não é mais uma carta de amor	5A
Jota Quest	6172	G	DIAS MELHORES	R.Flausino	Vivemos esperando dias melhores	5C
Jota Quest	6423	Em	NA MORAL	R.Flausino-W.Sideral	Na moral...vivendo de folia e caos	6A
Jota Quest	6509	E	SÓ HOJE	F.Mello-R.Flausino	Hoje eu preciso te encontrar de qualquer jeito	6C
Jota Quest	6603	F	AMOR MAIOR	Rogério Flausino	Eu quero ficar só mas comigo só eu não	6F
Jota Quest	6691	D	DO SEU LADO	Nando Reis	La la la... Faz muito tempo mas eu me lembro	7B
Jota Quest	7200	A	O SOL	Antônio Júlio Nastácia	Ei dor... eu não te escuto mais	9C
Jota Quest	7051	A	ALÉM DO HORIZONTE	R.Carlos-E.Carlos	Além do horizonte existe um lugar	9F
Joyce	7969	E	FORA DE HORA	Joyce	Fora de hora o meu coração pega a pensar	12D
Joyce	1448	C	CLAREANA	Joyce Moreno-Maurício Maestro	Um coração de mel de melão	17D
Juan Luis Guerra	3239	G	ROMANCE ROSA	Vs. J.G.Guerra-A.Reis	Eu te dei uma rosa que encontrei no	1B
Juca Novaes - Lenine	2918	E	MEIO ALMODÓVAR	Juca Novaes	Oh oh oh oh... Foi só um ensaio	22D
Juizo Final	5602	E	ATLETAS DE CRISTO	Ricardo-Fábio	Atletas atletas de Cristo	EVA
Juliana Baroni	9430	Em	DANCE, DANCE, DANCE	Paulo Anhaia-Rick Bonadio	Tantos desafios tanto a percorrer	13A
Juliana Baroni	9524	B	QUERO TE ENCONTRAR	Paulo Anhaia-Rick Bonadio	É triste saber que me acostumei a chorar	13D
Juliana Baroni	9418	A	CIDADE TRISTE	Clio-J.Jorgensen-Rampac	A dor de te ver ir embora	13E
Juliana Baroni - Lorenzo Martins	9492	D	NÃO HÁ HORA NEM LUGAR	Paulo Anhaia-Rick Bonadio	De que adianta reclamar dizer é o fim	13E
Juliana Diniz	9314	A	PARA FICAR	Juliana Diniz	Menino o seu amor foi-se embora por amor que	12D
Julieta Venegas - Marisa Monte	9687	C	ILUSION	Marisa Monte	Uma vez eu tive uma ilusão e não soube o que	14C
Julio Iglesias	7941	F#m	DEVANEIOS	L.Gardey-Erasmo Carlos	Ou me queres ou me deixas não dá mais pra	12C

3. Lista
3.0. Definição

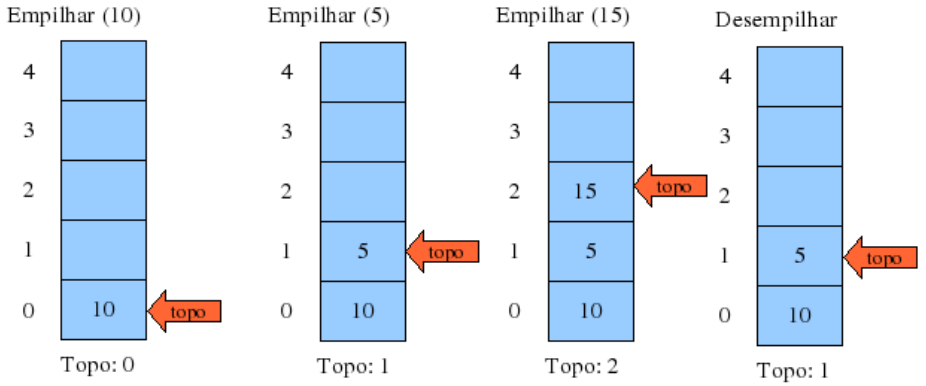
Fila

- Podemos inserir elementos sempre no fim da fila, Remover elementos do início da fila, procurar elementos na fila;
- Primeiro que entra é o primeiro que sai (FIFO);
- Exemplo: Fila de Banco, fila de cartório, fila para bater o ponto;

3. Lista
3.0. Definição

Pilha

- Podemos inserir elementos sempre no fim da pilha, Remover elementos do fim da pilha, procurar elementos na pilha;
- Último que entra é o primeiro que sai (FILO);
- Exemplo: Pilha de Livros;



3. Lista
3.0. Definição

Estruturas ESTÁTICA X DINÂMICA

- Estática:

- Serão utilizados ARRAYS (vetores);

- Dinâmica:

- Serão utilizados PONTEIROS;

Em ambas as estruturas utilizaremos STRUCTS.

3. Lista
3.0. Definição

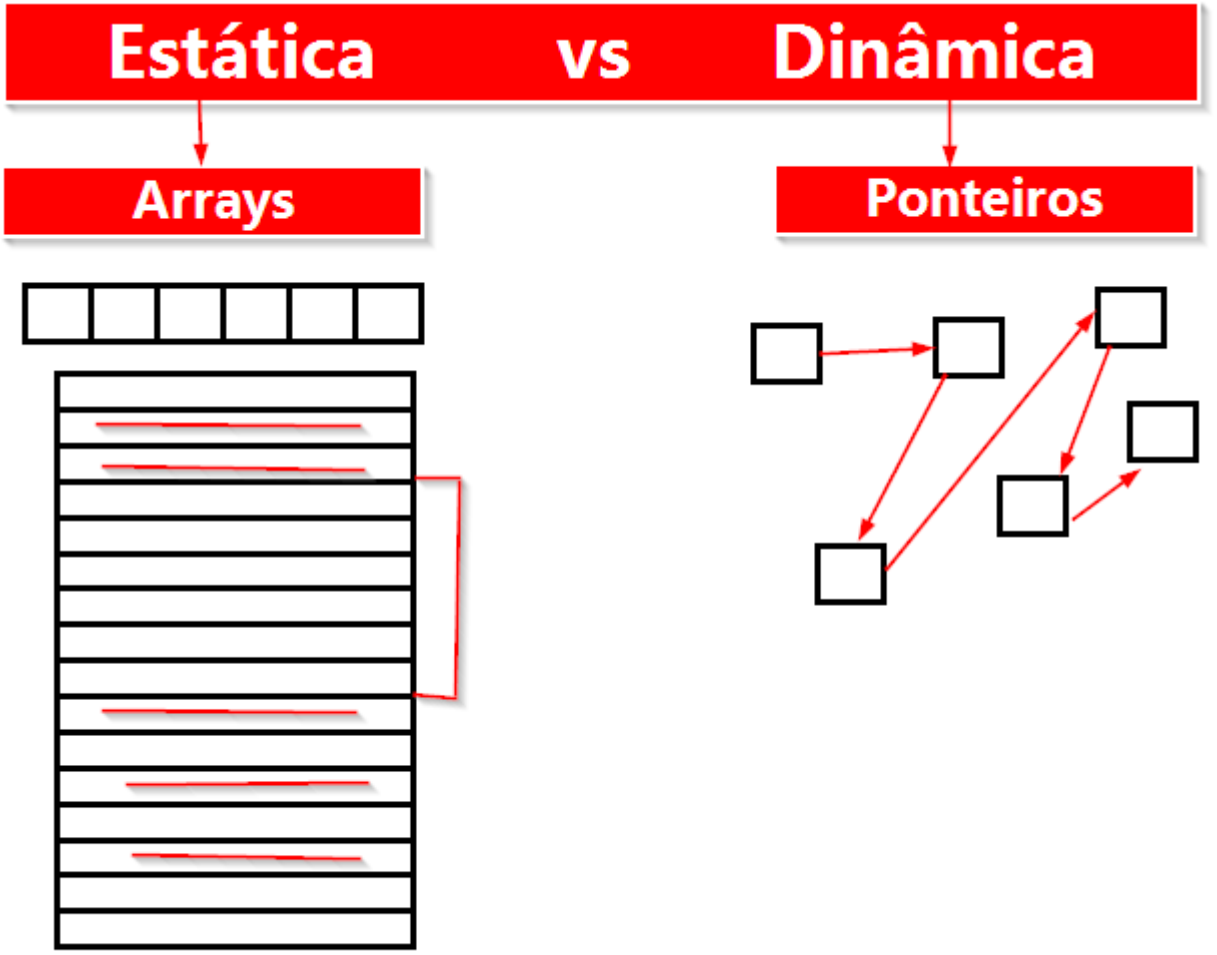
ESTÁTICA X DINÂMICA

- Estática:

- Consecutivos na memória;

- Dinâmica:

- Criado um por um não consecutivo na memória;



3. Lista
3.0. Definição

ESTÁTICA X DINÂMICA

- Estática: exemplo lista de chamada de alunos

- Vantagem:
  - Busca mais rápida;
- Desvantagem:
  - Inserir e remover elementos;
  - Espaço de memória alocado;

- Dinâmica: exemplo lista de músicas

- Vantagem:
  - Inserir e remover elementos;
- Desvantagem:
  - Busca mais demorada;

3. Lista
3.0. Definição

ESTÁTICA X DINÂMICA - Perguntas

- Uma estrutura Estática Possui?

- ( ) Ponteiros como base;
- ( ) Arrays como base;

- Oque a estrutura dinâmica faz mais rápido do que a estática?

- ( ) Buscar Elementos;
- ( ) Inserir elementos;

- Marque a resposta Falsa:

- ( ) Na Lista dinâmica os elementos estão posicionados aleatoriamente na memória.
- ( ) Na lista estática a quantidade de elementos possíveis depende do tamanho do array;
- ( ) Na lista dinâmica o array permite que se crie quantos elementos forem necessários;
- ( ) Na lista dinâmica o uso de ponteiros deixa a busca mais lenta do que a lista estática;

3. Lista
3.0. Definição

Lista

- É uma sequência de elementos, ordenados ou não, de um mesmo tipo.

- Seus elementos possuem estrutura interna abstraída, ou seja, sua complexidade é arbitrária e não afeta o seu funcionamento.

– Não teremos acesso direto aos dados, somente as funções que manipulam os mesmos.

3. Lista
3.0. Definição

Lista

- Uma lista pode possuir N elementos com N >= 0.

- Se N = 0 então a lista está vazia.

- Aplicações de Listas:

- Cadastro de Funcionários;
- Itens de Estoque.

3. Lista
3.0. Definição

Lista

- Operações básicas (funções) que podem ser feitas com Listas:

- - Criar a lista;
- - Inserir de um elemento;
- - Remover de um elemento;
- - Acessar a um elemento;
- - Destruir a lista;
- - Ordenar a lista;

- As operações acima dependem do tipo de alocação de memória usada:

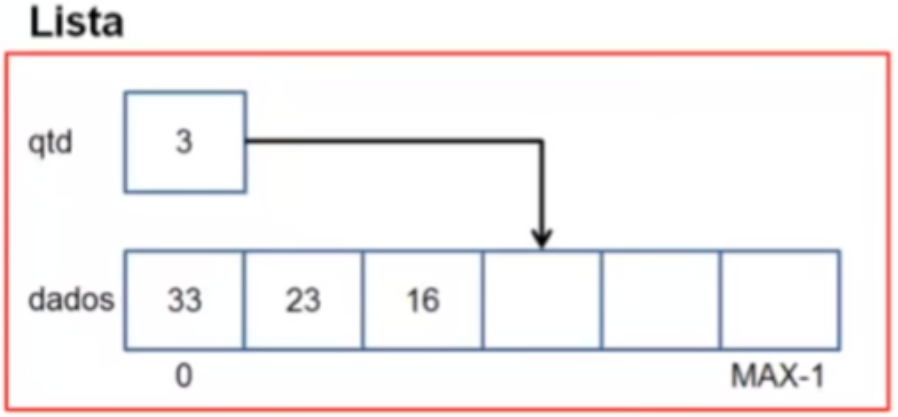
- - Estática;
- - Dinâmica;

3. Lista
3.0. Definição

Lista

- Lista com alocação ESTÁTICA:

- O espaço de memória é alocado no momento da compilação;
- Exige a definição do número máximo de elementos da lista;
- Acesso sequencial: os elementos ficam armazenados de forma consecutiva na memória;

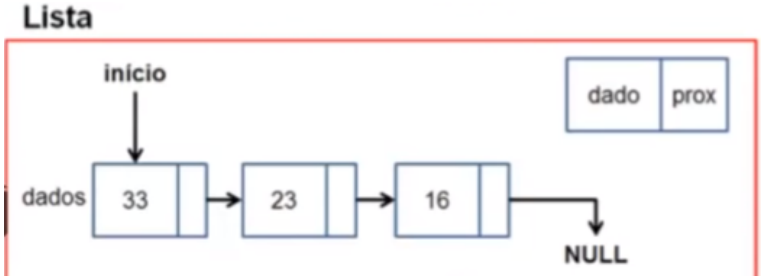


3. Lista
3.0. Definição

Lista

Lista com alocação DINÂMICA:

- O espaço de memória é alocado em tempo de execução;
- A lista cresce à medida que novos elementos são armazenados, e diminui à medida que elementos são removidos;
- Acesso encadeado: cada elemento pode estar em uma área distinta da memória. Para acessar um elemento, é preciso percorrer todos os seus antecessores na Lista;

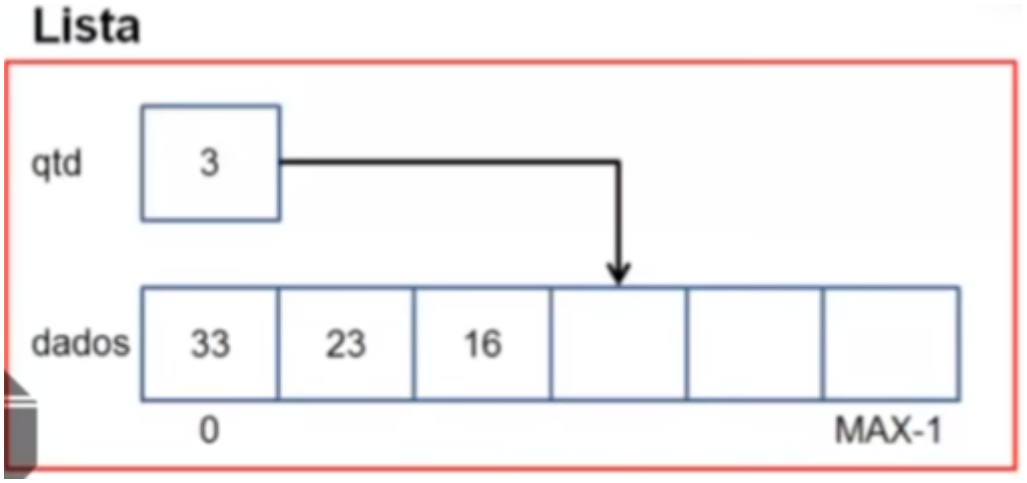


3. Lista
3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

- Tipo de lista onde o sucessor de um elemento ocupa a posição física seguinte do mesmo;

- Neste tipo de lista são utilizados arrays;



3. Lista
3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

Vantagens do uso de Arrays

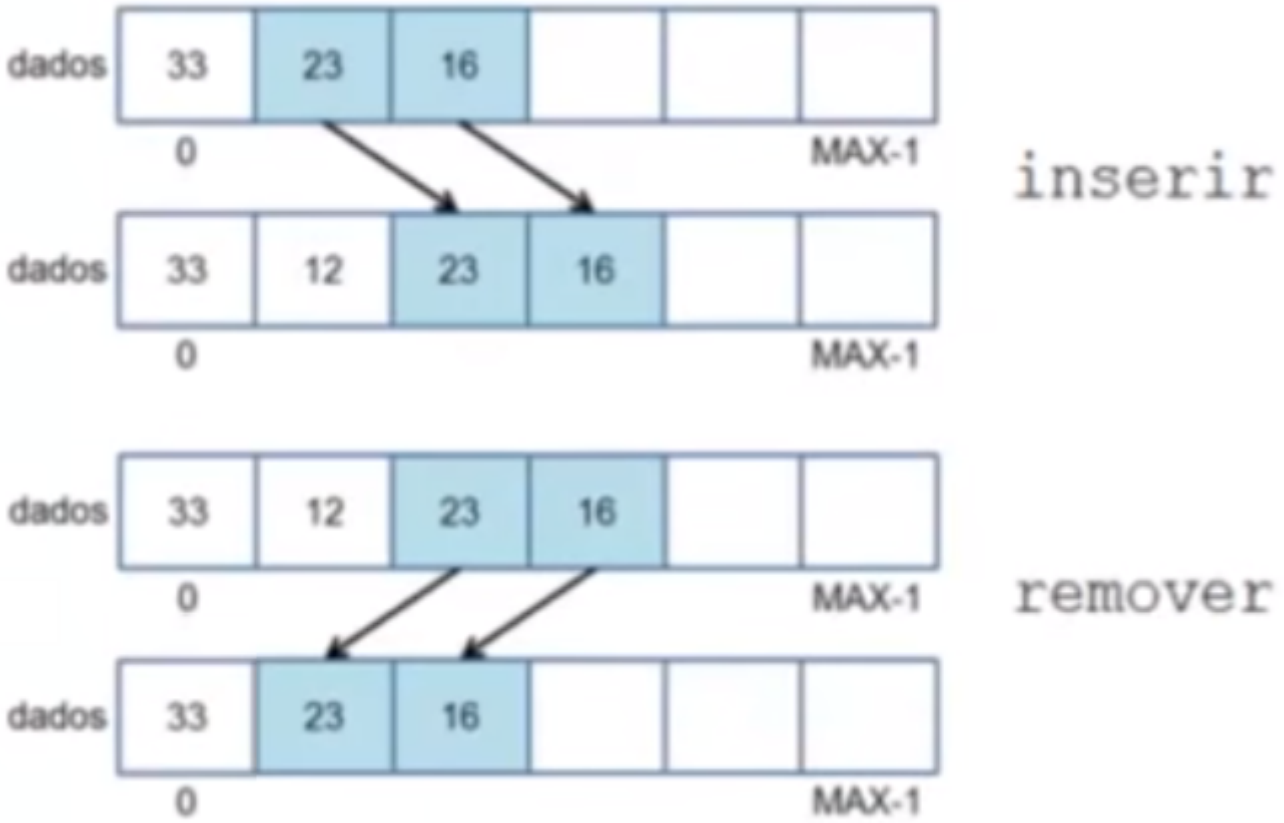
- Acesso rápido e direto aos elementos através dos índices;
- Tempo constante para acessar um elemento;
- Facilidade em modificar informações;

Desvantagens do uso de Arrays

- Definição prévia do tamanho do array;
- Dificuldade para inserir e remover um elemento entre outros dois. Neste caso é necessário deslocar os elementos;

3. Lista
3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática



3. Lista
3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

Quando utilizar essa Lista?

- Quando precisar armazenar pequena quantidade de dados;
- Inserção/Remoção apenas no final da lista;
- Tamanho máximo bem definido;
- A busca é a operação mais frequente;

3. Lista
3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

- Alocação ESTÁTICA de uma Lista:

- Para criar um lista iremos precisar criar duas STRUCTS;
- 1) Controlador;
  - Irá controlar o Tamanho da lista e a posição dos elementos;
- 2) Elementos da lista;
  - Desta forma será possível guardar vários tipos de dados;

3. Lista
3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

Implementando uma Lista Sequencial Estática

- Arquivo "ListaSequencial.h": definir:

- Protótipos das funções;
- O tipo de dados armazenado na lista;
- O ponteiro lista;
- Tamanho do vetor usado na lista.

- Arquivo "ListaSequencial.c": definir:

- O tipo de dados lista;
- Implementar as funções;

- Arquivo "main.c": definir:

- Fazer as chamadas das funções;

3. Lista
3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

- Implementando as funções de Criar e Liberar uma Lista Estática

- Criando a STRUCT que irá armazenar os Elementos da Lista;
- Criando a STRUCT Controlador;
- Definindo o ponteiro Lista;
- Implementar a função cria_lista;
- Implementar a função libera_lista;

3. Lista
3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

```c
//Arquivo ListaSequencial.h
#define MAX 10 //Tamanho que será nosso vetor

/*Definindo um Tipo Aluno
Elementos da Lista: desta forma será possível guardar
vários tipos de dados*/
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

//Definindo uma struct Lista
typedef struct lista Lista;
```

```c
//Arquivo ListaSequencial.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h" //inclui os Protótipos

/*Implementando a Struct do tipo lista
Controlador: irpa controlar o tamanho da lista
e a posição dos elementos*/
struct lista{
    int qtd;
    struct aluno dados[MAX];
};
```

```c
//main.c : Programa Principal
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h"

int main(){
    Lista* li; //Declarando um ponteiro do tipo Lista
}
```

3. Lista
3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

```c
//Arquivo ListaSequencial.h

#define MAX 10 //Tamanho que será nosso vetor

/*Definindo um Tipo Aluno
Elementos da Lista: desta forma será possível guardar
vários tipos de dados*/
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct lista Lista; //Definindo uma struct Lista

Lista* cria_lista(); //Definindo Função de criar Lista
```

```c
//main.c : Programa Principal
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h"

int main(){
    Lista* li = cria_lista();
}
```

```c
//Arquivo ListaSequencial.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h" //inclui os Protótipos

/*Implementando a Struct do tipo lista
Controlador: irpa controlar o tamanho da lista
e a posição dos elementos*/
struct lista{
    int qtd;
    struct aluno dados[MAX];
};

//Implementando a função criar lista
Lista* cria_lista(){
    Lista *li; //Declara ponteiro
    li = (Lista*)malloc(sizeof(struct lista));
    if(li != NULL)
        li->qtd = 0;
    return li;
}
```

3. Lista
3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

```c
//Arquivo ListaSequencial.h

#define MAX 10 //Tamanho que será nosso vetor

/*Definindo um Tipo Aluno
Elementos da Lista: desta forma será possível guardar
vários tipos de dados*/
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct lista Lista; //Definindo uma struct Lista

Lista* cria_lista(); //Definindo Função de criar Lista
void libera_lista(Lista* li); //Definindo função libera lista
```

```c
//main.c : Programa Principal
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h"

int main(){
    Lista* li = cria_lista();
    libera_lista(li);
}
```

```c
//Arquivo ListaSequencial.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h" //inclui os Protótipos

/*Implementando a Struct do tipo lista
Controlador: ira controlar o tamanho da lista
e a posição dos elementos*/
struct lista{
    int qtd;
    struct aluno dados[MAX];
};

//Implementando a função criar lista
Lista* cria_lista(){
    Lista *li; //Declara ponteiro
    li = (Lista*)malloc(sizeof(struct lista));
    if(li != NULL)
        li->qtd = 0;
    return li;
}
//Implementando a função libera lista
void libera_lista(Lista* li){
    free(li);
}
```

3. Lista
3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

- Implementando informações básicas sobre a lista:

- Tamanho;
- Lista está cheia?
- Lista está vazia?

3. Lista

3.1. Lista Sequencial Estática

```c
//Arquivo ListaSequencial.h

#define MAX 10 //Tamanho que será nosso vetor

/*Definindo um Tipo Aluno
Elementos da Lista: desta forma será possível guardar
vários tipos de dados*/
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};

typedef struct lista Lista; //Definindo uma struct Lista

Lista* cria_lista(); //Definindo Função
void libera_lista(Lista* li); //Definindo função
int tamanho_lista(Lista* li); //Definindo função
```

```c
//main.c : Programa Principal
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h"

int main(){
    Lista* li = cria_lista();
    libera_lista(li);
    int x = tamanho_lista(li);
}
```

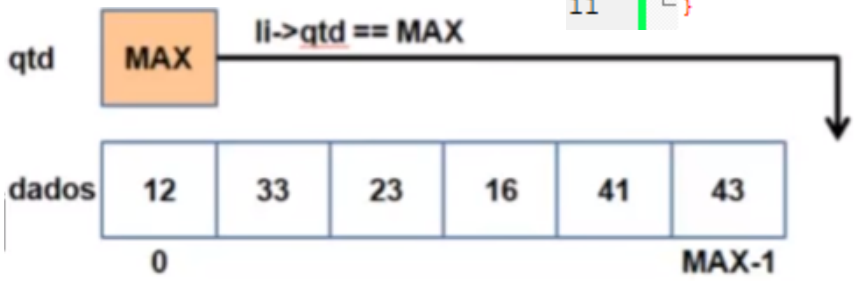

```c
//Arquivo ListaSequencial.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h" //inclui os Protótipos

/*Implementando a Struct do tipo lista
Controlador: ira controlar o tamanho da lista
e a posição dos elementos*/
struct lista{
    int qtd;
    struct aluno dados[MAX];
};

//Implementando a função criar lista
Lista* cria_lista(){
    Lista *li; //Declara ponteiro
    li = (Lista*)malloc(sizeof(struct lista));
    if(li != NULL)
        li->qtd = 0;
    return li;
}
//Implementando a função libera lista
void libera_lista(Lista* li){
    free(li);
}
//Implementando a funcao Tamanho da lista
int tamanho_lista(Lista* li){
    if(li == NULL)
        return -1;
    else
        return li->qtd;
}
```

3. Lista

3.1. Lista Sequencial Estática

```c
//Arquivo ListaSequencial.h
//Tamanho que será nosso vetor
#define MAX 10

/*Definindo um Tipo Aluno
Elementos da Lista: desta forma será possível guardar
vários tipos de dados*/
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};
//Definindo uma struct Lista
typedef struct lista Lista;

Lista* cria_lista();
void libera_lista(Lista* li);
int tamanho_lista(Lista* li);
int lista_cheia(Lista* li);
```

```c
//main.c : Programa Principal
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h"

int main(){
    Lista* li = cria_lista();
    libera_lista(li);
    int x = tamanho_lista(li);
    x = lista_cheia(li);
}
```



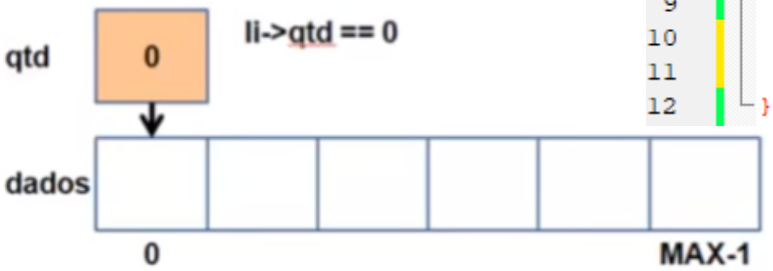

```c
//Arquivo ListaSequencial.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h" //inclui os Protótipos

/*Implementando a Struct do tipo lista
Controlador: ira controlar o tamanho da lista
e a posição dos elementos*/
struct lista{
    int qtd;
    struct aluno dados[MAX];
};

//Implementando a função criar lista
Lista* cria_lista(){
    Lista *li; //Declara ponteiro
    li = (Lista*)malloc(sizeof(struct lista));
    if(li != NULL)
        li->qtd = 0;
    return li;
}
//Implementando a função libera lista
void libera_lista(Lista* li){
    free(li);
}
//Implementando a funcao Tamanho da lista
int tamanho_lista(Lista* li){
    if(li == NULL)
        return -1;
    else
        return li->qtd;
}
//Implementando a funcao lista Cheia
int lista_cheia(Lista* li){
    if(li == NULL)
        return -1;
    return (li->qtd == MAX);
}
```

3. Lista

3.1. Lista Sequencial Estática

```c
//Arquivo ListaSequencial.h
//Tamanho que será nosso vetor
#define MAX 10

/*Definindo um Tipo Aluno
Elementos da Lista: desta forma será possível guardar
vários tipos de dados*/
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};
//Definindo uma struct Lista
typedef struct lista Lista;

Lista* cria_lista();
void libera_lista(Lista* li);
int tamanho_lista(Lista* li);
int lista_cheia(Lista* li);
int lista_vazia(Lista* li);
```

```c
//main.c : Programa Principal
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h"

int main(){
    Lista* li = cria_lista();
    libera_lista(li);
    int x = tamanho_lista(li);
    x = lista_cheia(li);
    x = lista_vazia(li);
}
```




```c
//Arquivo ListaSequencial.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h" //inclui os Protótipos

/*Implementando a Struct do tipo lista
Controlador: ira controlar o tamanho da lista
e a posição dos elementos*/
struct lista{
    int qtd;
    struct aluno dados[MAX];
};

//Implementando a função criar lista
Lista* cria_lista(){
    Lista *li; //Declara ponteiro
    li = (Lista*)malloc(sizeof(struct lista));
    if(li != NULL)
        li->qtd = 0;
    return li;
}
//Implementando a função libera lista
void libera_lista(Lista* li){
    free(li);
}
//Implementando a funcao Tamanho da lista
int tamanho_lista(Lista* li){
    if(li == NULL)
        return -1;
    else
        return li->qtd;
}
//Implementando a funcao lista Cheia
int lista_cheia(Lista* li){
    if(li == NULL)
        return -1;
    return (li->qtd == MAX);
}
//Implementando a funcao lista Vazia
int lista_vazia(Lista* li){
    if(li == NULL)
        return -1;
    return (li->qtd == 0);
}
```

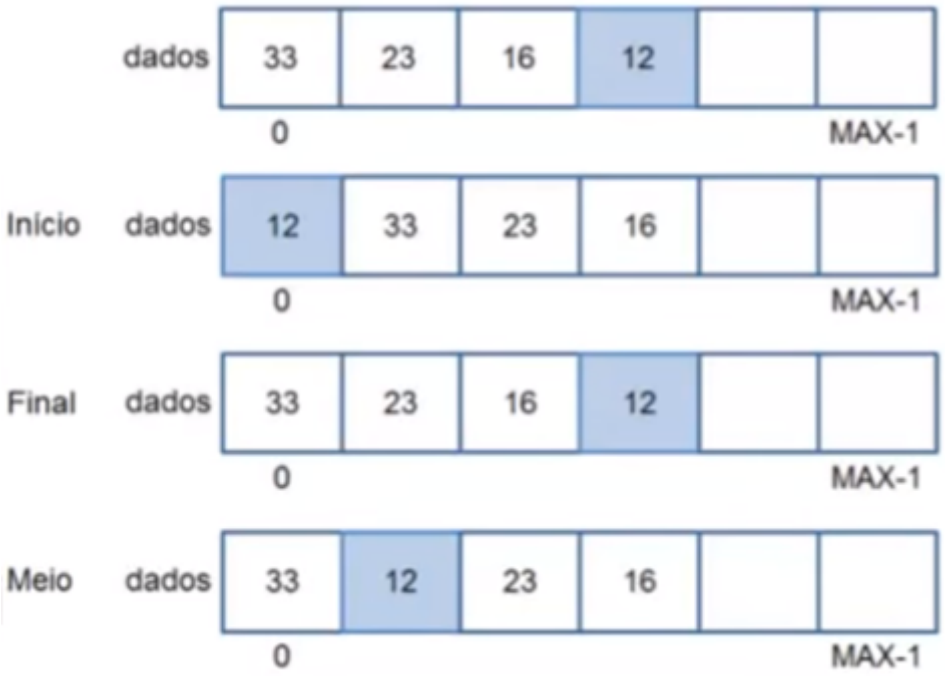
3. Lista
3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

- Implementação de INSERIR elementos em uma Lista Estática;

- Existem três tipos de inserção:

- No início;
- No meio;
- No fim;

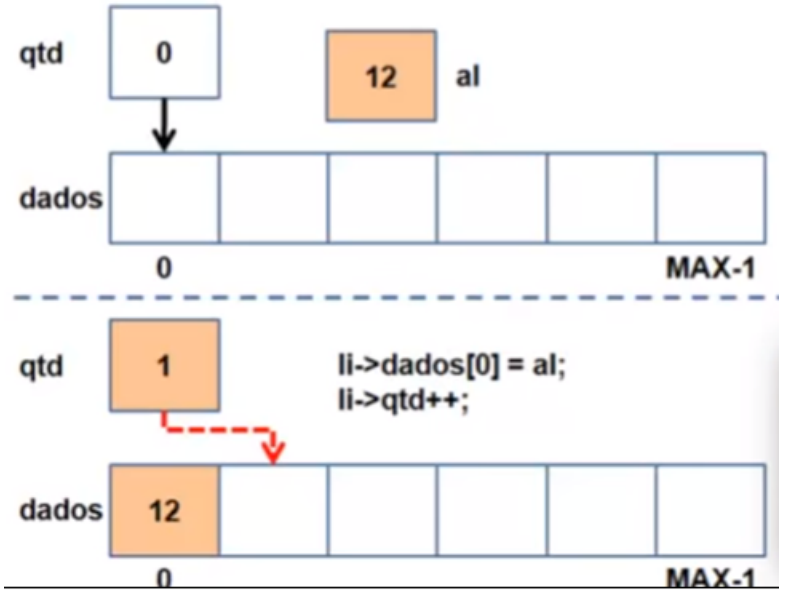


3. Lista
3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

- Existe o caso onde a inserção é feita em uma lista que está vazia.

- Não é possível inserir em uma lista cheia.



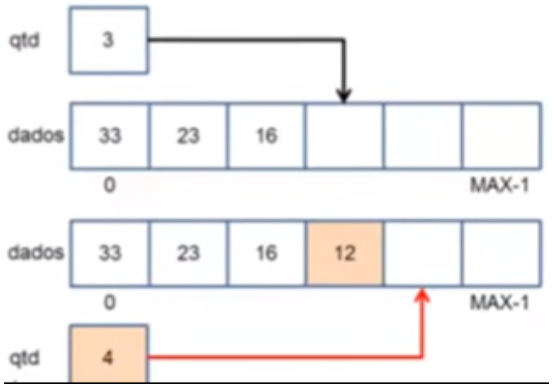
3. Lista

3.1. Lista Sequencial Estática

```c
//main.c : Programa Principal
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h"

int main(){
    Lista* li = cria_lista();
    aluno* dados_aluno;
    libera_lista(li);
    int x = tamanho_lista(li);
    x = lista_cheia(li);
    x = lista_vazia(li);
    int insere = insere_lista_final(li, dados_aluno);
}
```

```c
//Arquivo ListaSequencial.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h" //inclui os Protótipos

/*Implementando a Struct do tipo lista
Controlador: ira controlar o tamanho da lista
e a posição dos elementos*/
struct lista{
    int qtd;
    struct aluno dados[MAX];
};

//Implementando a funcao insere_lista_final
int insere_lista_final(Lista* li, struct aluno al){
    if(li == NULL)
        return 0;
    if(li->qtd == MAX)//lista cheia
        return 0;
    li->dados[li->qtd] = al;
    li->qtd++;
    return 1;
}
```

```c
//Arquivo ListaSequencial.h
//Tamanho que será nosso vetor
#define MAX 10

/*Definindo um Tipo Aluno
Elementos da Lista: desta forma será possível guardar
vários tipos de dados*/
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};
//Definindo uma struct Lista
typedef struct lista Lista;

Lista* cria_lista();
void libera_lista(Lista* li);
int tamanho_lista(Lista* li);
int lista_cheia(Lista* li);
int lista_vazia(Lista* li);
int insere_lista_final(Lista* li, struct aluno al);
```



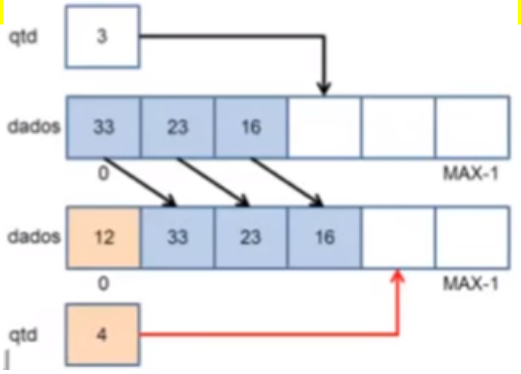
3. Lista

3.1. Lista Sequencial Estática

```c
//main.c : Programa Principal
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h"

int main(){
    Lista* li = cria_lista();
    aluno* dados_aluno;
    libera_lista(li);
    int x = tamanho_lista(li);
    x = lista_cheia(li);
    x = lista_vazia(li);
    int insere = insere_lista_final(li, dados_aluno);
    insere = insere_lista_inicio(li, dados_aluno);
}
```

```c
//Arquivo ListaSequencial.h
//Tamanho que será nosso vetor
#define MAX 10

/*Definindo um Tipo Aluno
Elementos da Lista: desta forma será possível guardar
vários tipos de dados*/
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};
//Definindo uma struct Lista
typedef struct lista Lista;

Lista* cria_lista();
void libera_lista(Lista* li);
int tamanho_lista(Lista* li);
int lista_cheia(Lista* li);
int lista_vazia(Lista* li);
int insere_lista_final(Lista* li, struct aluno al);
int insere_lista_inicio(Lista* li, struct aluno al);
```

```c
//Arquivo ListaSequencial.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h" //inclui os Protótipos

/*Implementando a Struct do tipo lista
Controlador: ira controlar o tamanho da lista
e a posição dos elementos*/
struct lista{
    int qtd;
    struct aluno dados[MAX];
};

//Implementando a funcao insere_lista_inicio
int insere_lista_inicio(Lista* li, struct aluno al){
    if(li == NULL)
        return 0;
    if(li->qtd == MAX)//lista cheia
        return 0;
    int i;
    for(i=li->qtd-1; i>=0; i--)
        li->dados[i+1] = li->dados[i];
    li->dados[0] = al;
    li->qtd++;
    return 1;
}
```



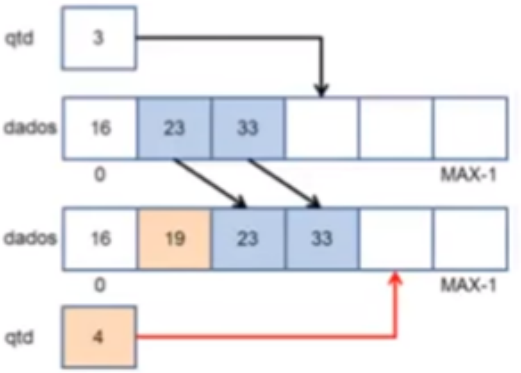
3. Lista

3.1. Lista Sequencial Estática

```c
//main.c : Programa Principal
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h"

int main(){
    Lista* li = cria_lista();
    aluno* dados_aluno;
    libera_lista(li);
    int x = tamanho_lista(li);
    x = lista_cheia(li);
    x = lista_vazia(li);
    int insere = insere_lista_final(li, dados_aluno);
    insere = insere_lista_inicio(li, dados_aluno);
    insere = insere_lista_ordenada(li, dados_aluno);
}
```

```c
//Arquivo ListaSequencial.h
//Tamanho que será nosso vetor
#define MAX 10

/*Definindo um Tipo Aluno
Elementos da Lista: desta forma será possível guardar
vários tipos de dados*/
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};
//Definindo uma struct Lista
typedef struct lista Lista;

Lista* cria_lista();
void libera_lista(Lista* li);
int tamanho_lista(Lista* li);
int lista_cheia(Lista* li);
int lista_vazia(Lista* li);
int insere_lista_final(Lista* li, struct aluno al);
int insere_lista_inicio(Lista* li, struct aluno al);
int insere_lista_ordenada(Lista* li, struct aluno al);
```

```c
//Arquivo ListaSequencial.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h" //inclui os Protótipos

/*Implementando a Struct do tipo lista
Controlador: ira controlar o tamanho da lista
e a posição dos elementos*/
struct lista{
    int qtd;
    struct aluno dados[MAX];
};

//Implementando a funcao insere_lista_ordenada
int insere_lista_ordenada(Lista* li, struct aluno al){
    if(li == NULL)
        return 0;
    if(li->qtd == MAX)//lista cheia
        return 0;
    int k,i = 0;
    while(i<li->qtd && li->dados[i].matricula < al.matricula)
        i++;

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

    li->dados[i] = al;
    li->qtd++;
    return 1;
}
```



3. Lista
3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

- Implementação para EXIBIR elementos em uma Lista Estática;

3. Lista

3.1. Lista Sequencial Estática

```c
//main.c : Programa Principal
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h"

int main(){
    Lista* li = cria_lista();
     aluno* dados_aluno;
     libera_lista(li);
     int matricula_aluno, posicao;
     int x = tamanho_lista(li);
     x = lista_cheia(li);
     x = lista_vazia(li);
     int insere = insere_lista_final(li, dados_aluno);
     insere = insere_lista_inicio(li, dados_aluno);
     insere = insere_lista_ordenada(li, dados_aluno);
     imprime_lista(li);
}
```

```c
//Arquivo ListaSequencial.h
//Tamanho que será nosso vetor
#define MAX 10

/*Definindo um Tipo Aluno
Elementos da Lista: desta forma será possível guardar
vários tipos de dados*/
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};
//Definindo uma struct Lista
typedef struct lista Lista;

Lista* cria_lista();
void libera_lista(Lista* li);
int tamanho_lista(Lista* li);
int lista_cheia(Lista* li);
int lista_vazia(Lista* li);
int insere_lista_final(Lista* li, struct aluno al);
int insere_lista_inicio(Lista* li, struct aluno al);
int insere_lista_ordenada(Lista* li, struct aluno al);
void imprime_lista(Lista* li);
```

```c
//Arquivo ListaSequencial.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h" //inclui os Protótipos

/*Implementando a Struct do tipo lista
Controlador: ira controlar o tamanho da lista
e a posição dos elementos*/
struct lista{
    int qtd;
    struct aluno dados[MAX];
};
//Implementacao para exibir a lista
void imprime_lista(Lista* li){
    if(li == NULL)
        return;
    int i;
    printf("--------------------------------\n");
    for(i=0; i< li->qtd; i++){
        printf("Matricula: %d\n",li->dados[i].matricula);
        printf("Nome: %s\n",li->dados[i].nome);
        printf("Notas: %.2f %.2f %.2f\n",li->dados[i].n1,
                                         li->dados[i].n2,
                                         li->dados[i].n3);
        printf("--------------------------------\n");
    }
}
```

3. Lista
3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

EXERCÍCIO:

- Fazer um menu para acessar as funções;
- Fazer as implementações descritas anteriormente;
- Popular a lista com informações do aluno na lista;

3. Lista

3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

EXERCÍCIO:

```
ACADEMICO: CLAYTON ZAMBON


      ################################################
      #       MENU - Lista Estatica Sequencial       #
      #                                              #
      #       Digite a opcao desejada                #
      #                                              #
      #        a = Inserir elemento no Final         #
      #        b = Inserir elemento no Inicio        #
      #        c = Inserir elemento Ordenado         #
      #        d = Remover elemento do Final         #
      #        e = Remover elemento do Inicio        #
      #        f = Remover elemento                  #
      #        g = Consultar elemento pela posicao   #
      #        h = Consultar elemento pela Matricula #
      #        i = Exibir lista                      #
      #        j = Exibir Tamanho da Lista           #
      #        k = Sair                              #
      #                                              #
      ################################################

      Escolha uma opcao >>: _
```

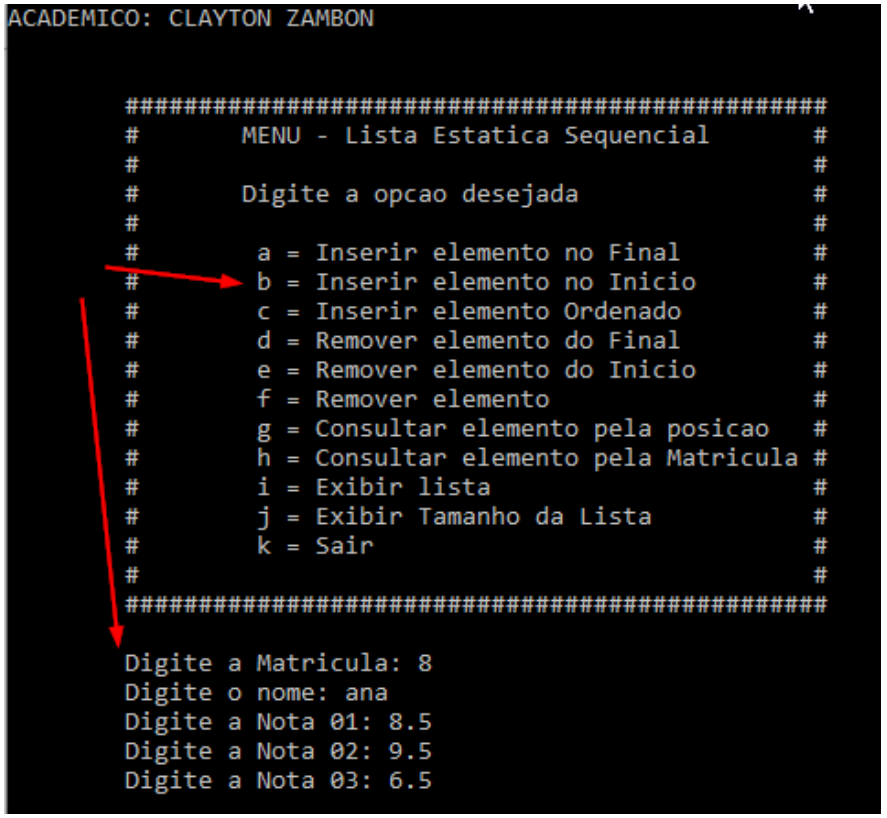
```
ACADEMICO: CLAYTON ZAMBON


      ################################################
      #       MENU - Lista Estatica Sequencial       #
      #                                              #
      #       Digite a opcao desejada                #
      #                                              #
      #        a = Inserir elemento no Final         #
      #        b = Inserir elemento no Inicio        #
      #        c = Inserir elemento Ordenado         #
      #        d = Remover elemento do Final         #
      #        e = Remover elemento do Inicio        #
      #        f = Remover elemento                  #
      #        g = Consultar elemento pela posicao   #
      #        h = Consultar elemento pela Matricula #
      #        i = Exibir lista                      #
      #        j = Exibir Tamanho da Lista           #
      #        k = Sair                              #
      #                                              #
      ################################################

      Digite a Matricula: 1
      Digite o nome: pedro
      Digite a Nota 01: 7.5
      Digite a Nota 02: 6.5
      Digite a Nota 03: 8.5
```

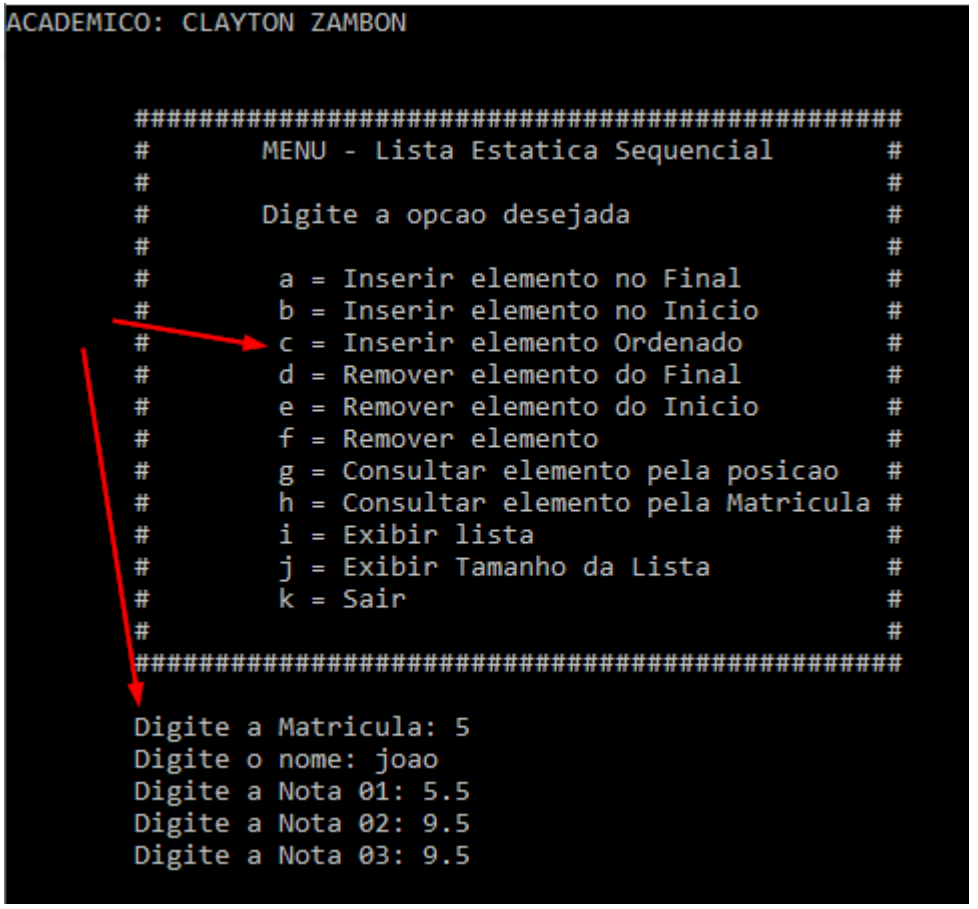
3. Lista

3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

EXERCÍCIO:





3. Lista

3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

EXERCÍCIO:

```
ACADEMICO: CLAYTON ZAMBON


       ################################################
       #       MENU - Lista Estatica Sequencial       #
       #                                              #
       #       Digite a opcao desejada                #
       #                                              #
       #        a = Inserir elemento no Final         #
       #        b = Inserir elemento no Inicio        #
       #        c = Inserir elemento Ordenado         #
       #        d = Remover elemento do Final         #
       #        e = Remover elemento do Inicio        #
       #        f = Remover elemento                  #
       #        g = Consultar elemento pela posicao   #
       #        h = Consultar elemento pela Matricula #
       #        i = Exibir lista                      #
       #        j = Exibir Tamanho da Lista           #
       #        k = Sair                              #
       #                                              #
       ################################################

       O tamanho da Lista eh: 3

Escolha outra opcao para continuar >>: _
```

```
ACADEMICO: CLAYTON ZAMBON


       ################################################
       #       MENU - Lista Estatica Sequencial       #
       #                                              #
       #       Digite a opcao desejada                #
       #                                              #
       #        a = Inserir elemento no Final         #
       #        b = Inserir elemento no Inicio        #
       #        c = Inserir elemento Ordenado         #
       #        d = Remover elemento do Final         #
       #        e = Remover elemento do Inicio        #
       #        f = Remover elemento                  #
       #        g = Consultar elemento pela posicao   #
       #        h = Consultar elemento pela Matricula #
       #        i = Exibir lista                      #
       #        j = Exibir Tamanho da Lista           #
       #        k = Sair                              #
       #                                              #
       ################################################

       ------------------------------
Matricula: 5
Nome: joao
Notas: 5.50 9.50 9.50
------------------------------
Matricula: 8
Nome: ana
Notas: 8.50 9.50 6.50
------------------------------
Matricula: 1
Nome: pedro
Notas: 7.50 6.50 8.50
------------------------------

Escolha outra opcao para continuar >>:
```

3. Lista
3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

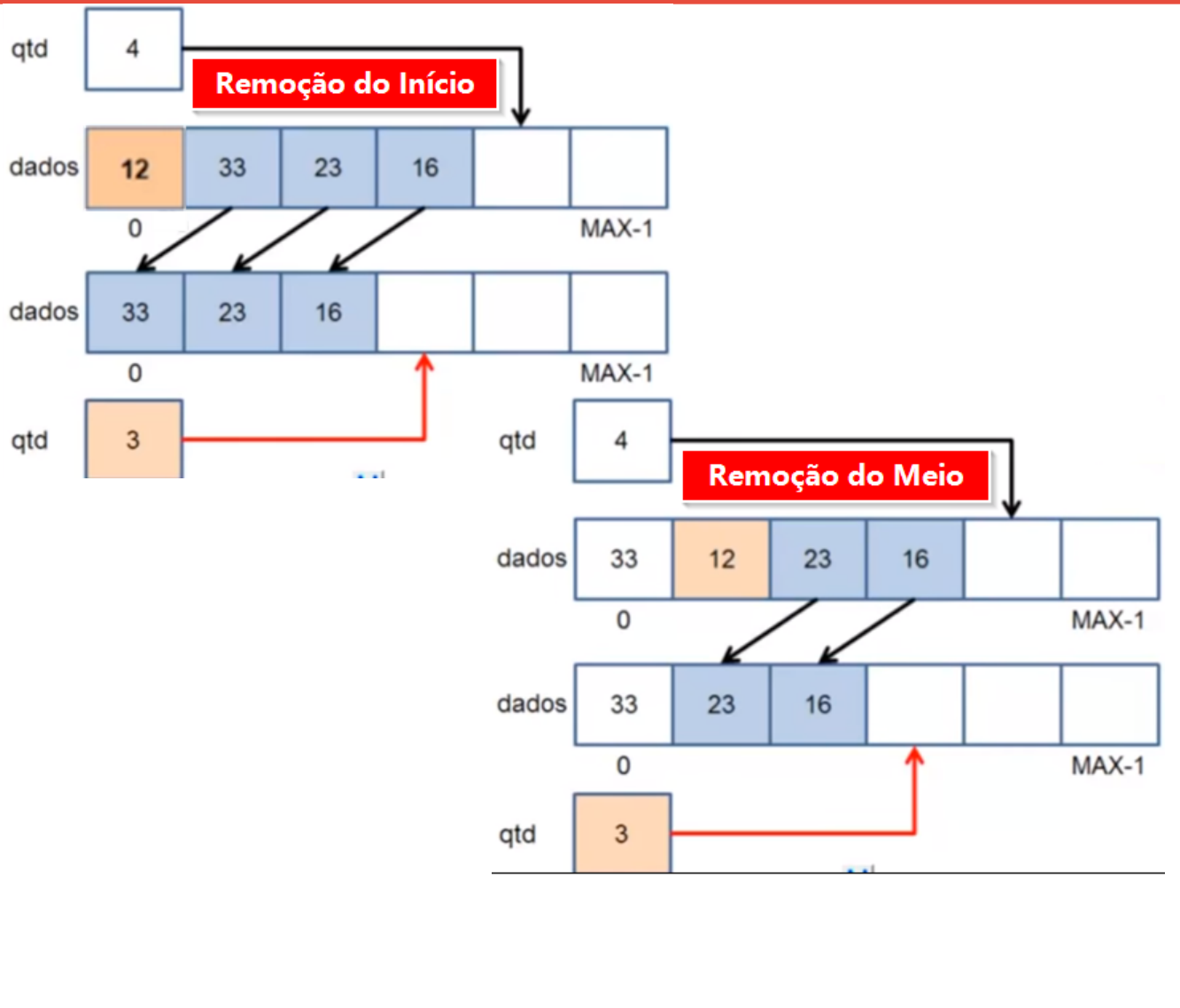
- Implementação de REMOÇÃO elementos em uma Lista Estática;

- Existem três tipos de REMOÇÃO:

- Do início;
- Do meio;
- Do fim;

3. Lista

3.1. Lista Sequencial Estática



3. Lista
3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

- Os 3 tipos de REMOÇÃO trabalham juntos;

- A remoção sempre remove um elemento específico da lista, o qual pode estar no início, no meio ou no final da lista;

- Não é possível remover elementos de uma lista vazia;

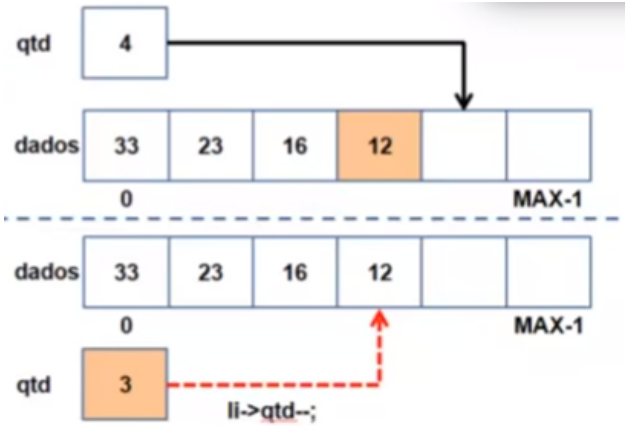
3. Lista

3.1. Lista Sequencial Estática

```c
//main.c : Programa Principal
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h"

int main(){
    Lista* li = cria_lista();
    aluno* dados_aluno;
    libera_lista(li);
    int x = tamanho_lista(li);
    x = lista_cheia(li);
    x = lista_vazia(li);
    int insere = insere_lista_final(li, dados_aluno);
    insere = insere_lista_inicio(li, dados_aluno);
    insere = insere_lista_ordenada(li, dados_aluno);
    int remove = remove_lista_final(li);
}
```

```c
//Arquivo ListaSequencial.h
//Tamanho que será nosso vetor
#define MAX 10

/*Definindo um Tipo Aluno
Elementos da Lista: desta forma será possível guardar
vários tipos de dados*/
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};
//Definindo uma struct Lista
typedef struct lista Lista;

Lista* cria_lista();
void libera_lista(Lista* li);
int tamanho_lista(Lista* li);
int lista_cheia(Lista* li);
int lista_vazia(Lista* li);
int insere_lista_final(Lista* li, struct aluno al);
int insere_lista_inicio(Lista* li, struct aluno al);
int insere_lista_ordenada(Lista* li, struct aluno al);
int remove_lista_final(Lista* li);
```

```c
//Arquivo ListaSequencial.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h" //inclui os Protótipos

/*Implementando a Struct do tipo lista
Controlador: ira controlar o tamanho da lista
e a posição dos elementos*/
struct lista{
    int qtd;
    struct aluno dados[MAX];
};

//Implementando a funcao remove_lista_final
int remove_lista_final(Lista* li){
    if(li == NULL)
        return 0;
    if(li->qtd == 0)
        return 0;
    li->qtd--;
    return 1;
}
```



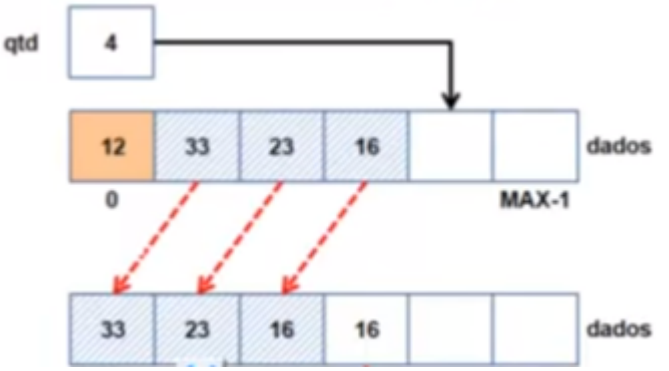
3. Lista

3.1. Lista Sequencial Estática

```c
//main.c : Programa Principal
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h"

int main(){
    Lista* li = cria_lista();
    aluno* dados_aluno;
    libera_lista(li);
    int x = tamanho_lista(li);
    x = lista_cheia(li);
    x = lista_vazia(li);
    int insere = insere_lista_final(li, dados_aluno);
    insere = insere_lista_inicio(li, dados_aluno);
    insere = insere_lista_ordenada(li, dados_aluno);
    int remove = remove_lista_final(li);
    remove = remove_lista_inicio(li);
}
```

```c
//Arquivo ListaSequencial.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h" //inclui os Protótipos

/*Implementando a Struct do tipo lista
Controlador: ira controlar o tamanho da lista
e a posição dos elementos*/
struct lista{
    int qtd;
    struct aluno dados[MAX];
};
//Implementando a funcao remove_lista_inicio
int remove_lista_inicio(Lista* li){
    if(li == NULL)
        return 0;
    if(li->qtd == 0)
        return 0;
    int k = 0;
    for(k=0; k< li->qtd-1; k++)
        li->dados[k] = li->dados[k+1];
    li->qtd--;
    return 1;
}
```

```c
//Arquivo ListaSequencial.h
//Tamanho que será nosso vetor
#define MAX 10

/*Definindo um Tipo Aluno
Elementos da Lista: desta forma será possível guardar
vários tipos de dados*/
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};
//Definindo uma struct Lista
typedef struct lista Lista;

Lista* cria_lista();
void libera_lista(Lista* li);
int tamanho_lista(Lista* li);
int lista_cheia(Lista* li);
int lista_vazia(Lista* li);
int insere_lista_final(Lista* li, struct aluno al);
int insere_lista_inicio(Lista* li, struct aluno al);
int insere_lista_ordenada(Lista* li, struct aluno al);
int remove_lista_final(Lista* li);
int remove_lista_inicio(Lista* li);
```



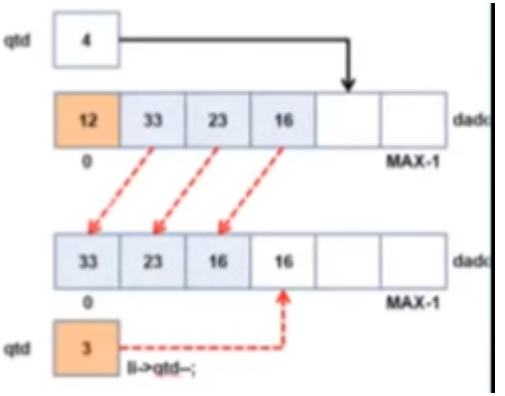
3. Lista

3.1. Lista Sequencial Estática

```c
int main(){
    Lista* li = cria_lista();
    aluno* dados_aluno;
    libera_lista(li);
    int x = tamanho_lista(li);
    x = lista_cheia(li);
    x = lista_vazia(li);
    int insere = insere_lista_final(li, dados_aluno);
    insere = insere_lista_inicio(li, dados_aluno);
    insere = insere_lista_ordenada(li, dados_aluno);
    int remove = remove_lista_final(li);
    remove = remove_lista_inicio(li);
    remove = remove_lista(li, matricula_aluno);
}
```

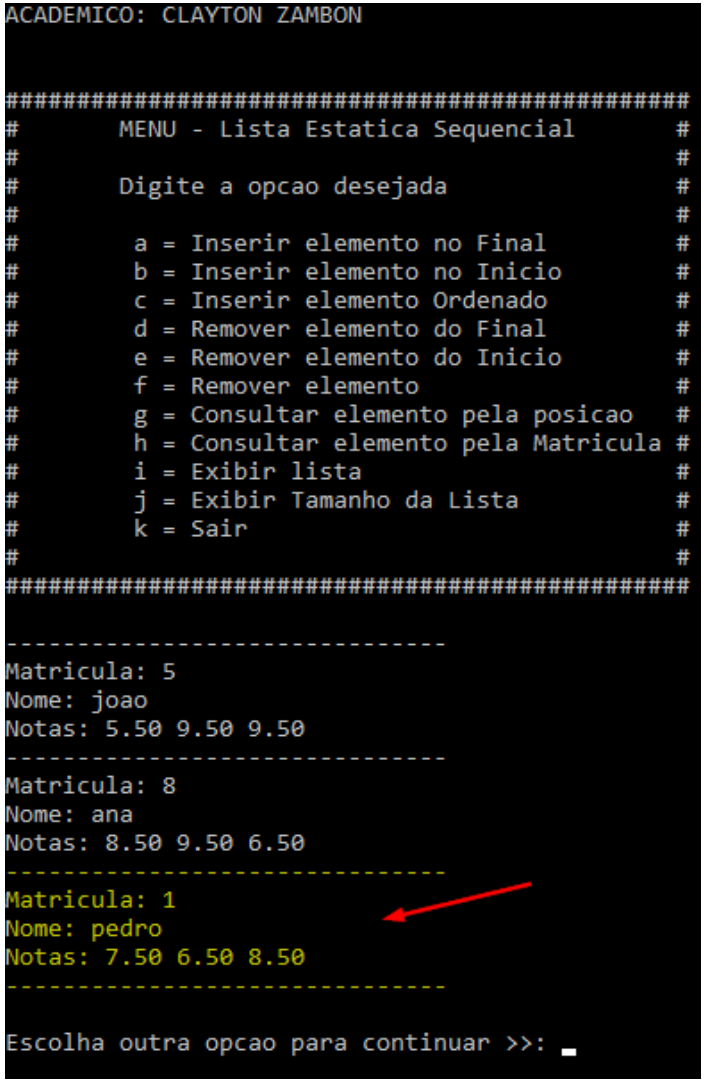
```c
//Arquivo ListaSequencial.h
//Tamanho que será nosso vetor
#define MAX 10

/*Definindo um Tipo Aluno
Elementos da Lista: desta forma será possível guardar
vários tipos de dados*/
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};
//Definindo uma struct Lista
typedef struct lista Lista;

Lista* cria_lista();
void libera_lista(Lista* li);
int tamanho_lista(Lista* li);
int lista_cheia(Lista* li);
int lista_vazia(Lista* li);
int insere_lista_final(Lista* li, struct aluno al);
int insere_lista_inicio(Lista* li, struct aluno al);
int insere_lista_ordenada(Lista* li, struct aluno al);
int remove_lista_final(Lista* li);
int remove_lista_inicio(Lista* li);
int remove_lista(Lista* li, int mat);
```

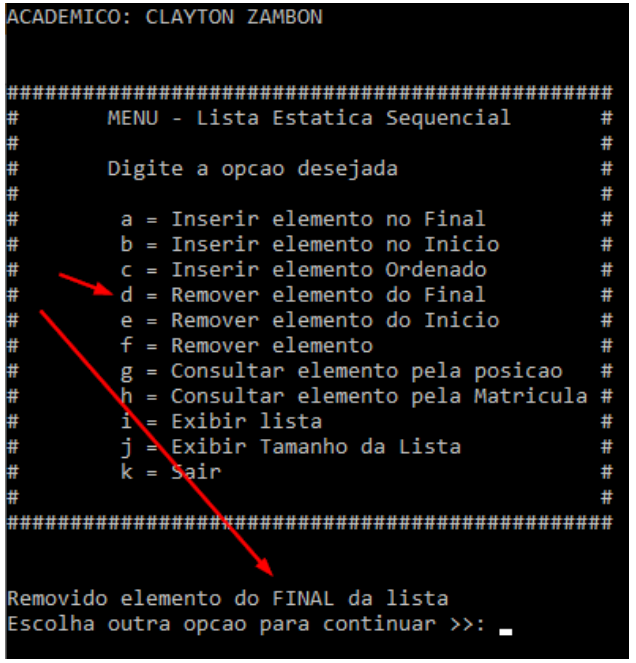
```c
//Arquivo ListaSequencial.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h" //inclui os Protótipos

/*Implementando a Struct do tipo lista
Controlador: ira controlar o tamanho da lista
e a posição dos elementos*/
struct lista{
    int qtd;
    struct aluno dados[MAX];
};
//Implementando a funcao remove_lista
int remove_lista(Lista* li, int mat){
    if(li == NULL)
        return 0;
    if(li->qtd == 0)
        return 0;
    int k,i = 0;
    while(i<li->qtd && li->dados[i].matricula != mat)
        i++;
    if(i == li->qtd)//elemento nao encontrado
        return 0;

    for(k=i; k< li->qtd-1; k++)
        li->dados[k] = li->dados[k+1];
    li->qtd--;
    printf("\nRemovida a matricula %d da lista", mat);
    return 1;
}
```

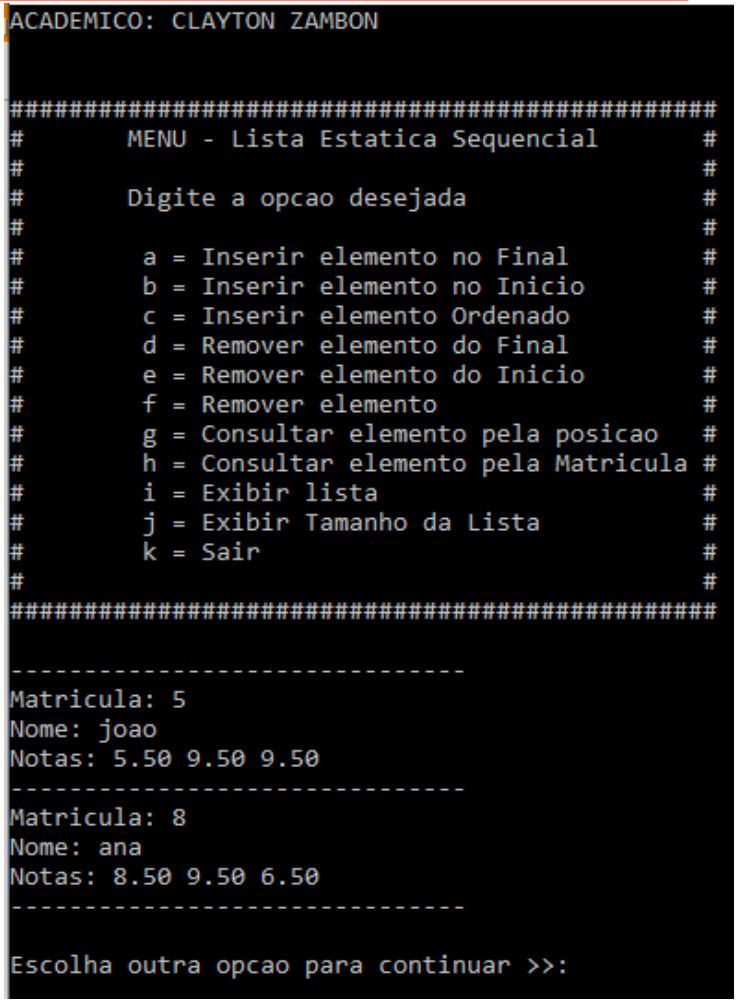


3. Lista

3.1. Lista Sequencial Estática







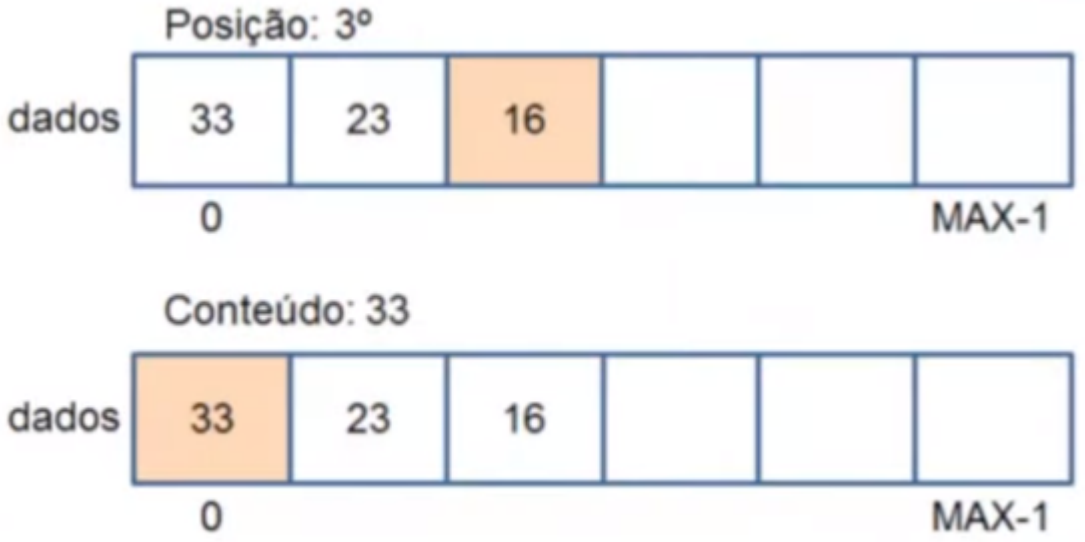
3. Lista
3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

- Implementação de CONSULTA De elementos em uma Lista Estática;

- Existem duas maneiras de consultar um elemento:

- Pela Posição;
- Pelo Conteúdo;



3. Lista

3.1. Lista Sequencial Estática

```c
int main(){
    Lista* li = cria_lista();
     aluno* dados_aluno;
     libera_lista(li);
     int matricula_aluno, posicao;
     int x = tamanho_lista(li);
     x = lista_cheia(li);
     x = lista_vazia(li);
     int insere = insere_lista_final(li, dados_aluno);
     insere = insere_lista_inicio(li, dados_aluno);
     insere = insere_lista_ordenada(li, dados_aluno);
     int remove = remove_lista_final(li);
     remove = remove_lista_inicio(li);
     remove = remove_lista(li, matricula_aluno);
     int consulta = consulta_lista_pos(li, posicao, &dados_aluno);
     consulta = consulta_lista_mat(li, posicao, &dados_aluno);
}
```

```c
//Arquivo ListaSequencial.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h" //inclui os Protótipos

/*Implementando a Struct do tipo lista
Controlador: ira controlar o tamanho da lista
e a posição dos elementos*/
struct lista{
    int qtd;
    struct aluno dados[MAX];
};
//Implementando a funcao consulta_lista_mat
int consulta_lista_mat(Lista* li, int mat, struct aluno *al){
    if(li == NULL)
        return 0;
    int i = 0;
    while(i<li->qtd && li->dados[i].matricula != mat)
        i++;
    if(i == li->qtd)//elemento nao encontrado
        return 0;

    *al = li->dados[i];
    return 1;
}
```

```c
//Arquivo ListaSequencial.h
//Tamanho que será nosso vetor
#define MAX 10

/*Definindo um Tipo Aluno
Elementos da Lista: desta forma será possível guardar
vários tipos de dados*/
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};
//Definindo uma struct Lista
typedef struct lista Lista;

Lista* cria_lista();
void libera_lista(Lista* li);
int tamanho_lista(Lista* li);
int lista_cheia(Lista* li);
int lista_vazia(Lista* li);
int insere_lista_final(Lista* li, struct aluno al);
int insere_lista_inicio(Lista* li, struct aluno al);
int insere_lista_ordenada(Lista* li, struct aluno al);
int remove_lista_final(Lista* li);
int remove_lista_inicio(Lista* li);
int remove_lista(Lista* li, int mat);
int consulta_lista_pos(Lista* li, int pos, struct aluno *al);
int consulta_lista_mat(Lista* li, int mat, struct aluno *al);

```

3. Lista

3.1. Lista Sequencial Estática

```c
int main(){
    /Lista* li = cria_lista();
     aluno* dados_aluno;
     libera_lista(li);
     int matricula_aluno, posicao;
     int x = tamanho_lista(li);
     x = lista_cheia(li);
     x = lista_vazia(li);
     int insere = insere_lista_final(li, dados_aluno);
     insere = insere_lista_inicio(li, dados_aluno);
     insere = insere_lista_ordenada(li, dados_aluno);
     int remove = remove_lista_final(li);
     remove = remove_lista_inicio(li);
     remove = remove_lista(li, matricula_aluno);
     int consulta = consulta_lista_pos(li, posicao, &dados_aluno);
}
```

```c
//Arquivo ListaSequencial.c
#include <stdio.h>
#include <stdlib.h>
#include "ListaSequencial.h" //inclui os Protótipos

/*Implementando a Struct do tipo lista
Controlador: ira controlar o tamanho da lista
e a posição dos elementos*/
struct lista{
    int qtd;
    struct aluno dados[MAX];
};
//Implementando a funcao consulta_lista_pos
int consulta_lista_pos(Lista* li, int pos, struct aluno *al){
    if(li == NULL || pos <= 0 ||  pos > li->qtd)
        return 0;
    *al = li->dados[pos-1];
    return 1;
}
```

```c
//Arquivo ListaSequencial.h
//Tamanho que será nosso vetor
#define MAX 10

/*Definindo um Tipo Aluno
Elementos da Lista: desta forma será possível guardar
vários tipos de dados*/
struct aluno{
    int matricula;
    char nome[30];
    float n1,n2,n3;
};
//Definindo uma struct Lista
typedef struct lista Lista;

Lista* cria_lista();
void libera_lista(Lista* li);
int tamanho_lista(Lista* li);
int lista_cheia(Lista* li);
int lista_vazia(Lista* li);
int insere_lista_final(Lista* li, struct aluno al);
int insere_lista_inicio(Lista* li, struct aluno al);
int insere_lista_ordenada(Lista* li, struct aluno al);
int remove_lista_final(Lista* li);
int remove_lista_inicio(Lista* li);
int remove_lista(Lista* li, int mat);
int consulta_lista_pos(Lista* li, int pos, struct aluno *al);
```

3. Lista
3.1. Lista Sequencial Estática

Lista Sequencial Estática ou Lista Linear Estática

EXERCÍCIO:

- Ao consultar pela posição ou matrícula, exibir as informações do elemento;

- Quando não encontrar um elemento exibir uma mensagem para o usuário;

- Quando não for possível realizar a operação, exibir a mensagem de lista vazia ou lista cheia conforme o caso.

Referências

EDELWEISS, Nina; GALANTE, Renata. Estruturas de Dados. Porto Alegre, BOOKMAN, 2009.

HEINZLE, Roberto. Estruturas de Dados: implementações com C e Pascal. Blumenau, DIRETIVA, 2006.

TENENBAUM, Aron M. Estrutura de Dados usando C. São Paulo, Makron Books, 1995.

FORBELLONE, André Luiz Villar: EBERSPÄCHER, Henri Frederico. Lógica de Programação: a construção de algoritmos e estruturas de dados. 3. ed. São Paulo, PRENTICE HALL, 2005.

KOFFMAN, Elliot B.; WOLFGANG, Paul A. T. Objetos, abstração, estruturas de dados e projeto usando C++. Rio de Janeiro, LTC, 2008.

PEREIRA, Silvio do lago. Estruturas de dados fundamentais: conceitos e aplicações. São Paulo, Érica, 1996.

VILLAS, Marcos Viana et al. Estruturas de dados – Conceitos e técnicas de implementação. Rio de Janeiro, Campus, 1993.

VELOSO, Paulo et al. Estrutura de dados. Rio de Janeiro, Campus, 1996.

Canal do Youtube: Linguagem C Programação Descomplicada

Obrigado!