	<b>ROTEIRO DE PRÁTICA EM LABORATÓRIO</b> ESTRUTURA DE DADOS – 2019.1 PROF. ANDERSON LIMA		
<b>ATIVIDADE</b>	LABORATÓRIO DE LISTA	<b>DATA</b>	26/02/2019

## OBJETIVO

Demonstrar os conceitos básicos de lista sequencial e implementar código com as operações básicas iniciais.

## PREPARAÇÃO

- I. Abrir o **CodeBlocks**.
- II. Criar um novo projeto.  
 Create a new project – Console Application – Next – C  
 Project Title: projeto-lista  
 Folder: <diretorio-projeto>  
 Finish

## ROTEIRO

1. Criar o arquivo de header (ListaSequencial.h)
  - File – New – File... – C/C++ header – Go
  - Filename with full path: <diretorio-projeto>/ListaSequencial.h
  - Finish
2. Remover o conteúdo do arquivo gerado automaticamente.
3. Declarar no arquivo de cabeçalho os dados e operações que serão usados na lista.

### ListaSequencia.h

```
#define MAX 100
struct aluno {
    int matricula;
    char nome[30];
    float n1, n2, n3;
};

typedef struct lista Lista;

Lista* cria_lista();
void libera_lista(Lista* li);

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



## ROTEIRO DE PRÁTICA EM LABORATÓRIO

ESTRUTURA DE DADOS – 2019.1

PROF. ANDERSON LIMA

### ATIVIDADE

LABORATÓRIO DE LISTA

DATA

26/02/2019

```
int insere_lista_inicio(Lista* li, struct aluno info);
int insere_lista_final(Lista* li, struct aluno info);
int insere_lista_ordenada(Lista* li, struct aluno info);
```

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

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

#### 4. Criar o arquivo de implementação (ListaSequencial.c)

- File – New – File... – C/C++ source – Go – C – Next
- Filename with full path: <diretorio-projeto>/ListaSequencial.c
- Finish

#### 5. Iniciar codificação do corpo do TAD lista.

##### ListaSequencia.c


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

struct lista {
    struct aluno dados[MAX];
    int qtd;
};
```

#### 6. Implementar a função para **criação** da lista.

##### ListaSequencia.c (continuação)

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

	<b>ROTEIRO DE PRÁTICA EM LABORATÓRIO</b> ESTRUTURA DE DADOS – 2019.1 PROF. ANDERSON LIMA		
<b>ATIVIDADE</b>	LABORATÓRIO DE LISTA	<b>DATA</b>	26/02/2019

7. Implementar a função para **destruição** da lista.

**ListaSequencia.c** (continuação)

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

8. Implementar a função para verificar o **tamanho** da lista.

**ListaSequencia.c** (continuação)

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

9. Implementar a função para verificar se a lista está **vazia**.


**ListaSequencia.c** (continuação)

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

10. Implementar a função para verificar se a lista está **cheia**.

**ListaSequencia.c** (continuação)

```
int lista_cheia(Lista* li) {
    if (li == NULL)
        return 0;
    else
        return (li->qtd == MAX);
}
```

	<b>ROTEIRO DE PRÁTICA EM LABORATÓRIO</b> ESTRUTURA DE DADOS – 2019.1 PROF. ANDERSON LIMA		
<b>ATIVIDADE</b>	LABORATÓRIO DE LISTA	<b>DATA</b>	26/02/2019

11. Implementar a função para **inserir um elemento no final** da lista.

**ListaSequencia.c** (continuação)

```
int insere_lista_final(Lista* li, struct aluno info) {
    if (li == NULL)
        return 0;
    if (li->qtd == MAX)
        return 0;
    li->dados[li->qtd] = info;
    li->qtd++;
    return 1;
}
```

12. Alterar o arquivo **main.c** para testar a estrutura de dados do tipo lista.

**main.c**

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

int main() {
    printf("=====\n");
    printf("Implementacao de Lista Sequencial\n");
    printf("=====\n");
    printf("\n\n");

    Lista* listaAluno;
    listaAluno = cria_lista();

    struct aluno aluno1;
    aluno1.matricula = 10;
    strcpy(aluno1.nome, "Fulano de Tal");
    aluno1.n1 = 7.5;
    aluno1.n2 = 8.1;
    aluno1.n3 = 9.0;

    insere_lista_final(listaAluno, aluno1);

    struct aluno aluno2;
    aluno2.matricula = 20;
    strcpy(aluno2.nome, "Beltrano Pereira");
    aluno2.n1 = 5.0;
    aluno2.n2 = 8.0;
    aluno2.n3 = 10.0;
```



## ROTEIRO DE PRÁTICA EM LABORATÓRIO

ESTRUTURA DE DADOS – 2019.1

PROF. ANDERSON LIMA

### ATIVIDADE

LABORATÓRIO DE LISTA

### DATA

26/02/2019

```
insere_lista_final(listaAluno, aluno2);

printf("Tamanho da lista: %d \n", tamanho_lista(listaAluno);

return 0;
}
```

## DESAFIO

- I. Criar uma operação para imprimir o conteúdo da lista no seguinte formato:

```
----- Dados da lista -----
Pos 00 ->| 10 Fulano de Tal 5.0 6.0 7.0 |<-
Pos 01 ->| 20 Beltrano Pereira 8.0 7.0 9.0 |<-
Pos 02 ->| 30 Sicrano Silva 10.0 9.2 8.1 |<-
-----
Tamanho: 3
-----
```

- II. Criar um menu para permitir escolher operações para ser realizada na lista, no seguinte formato:

MENU PRINCIPAL

```
-----
1: Inserir aluno
2: Imprimir lista
0: Sair
-----
```

Informe a opção desejada: <opcao>

- III. Permitir ao operador informar os dados do aluno ao escolher a opção 1 no menu principal (inserir aluno) e adicionar o aluno à lista.

```
Informe a matricula: <matricula>
Informe o nome: <nome>
Informe a 1a nota: <nota1>
Informe a 2a nota: <nota2>
Informe a 3a nota: <nota3>
```