

Roteiro4

Giancarlo Oliveira Teixeira - 222050010

Setembro 2023

1 Lista Sequencial Estática

LE.h

```
/*----- File: Lista.h -----+
|Lista Sequencial Estatica      |
|                                |
|                                |
|                                |
|                                |
| Implementado por Guilherme C. Pena em 12/09/2023      |
| Implementacoes extras e modificacao dos comandos      |
| de impressao feitas por Giancarlo O. Teixeira      |
+-----+ */

#ifndef LISTA_H
#define LISTA_H

#include <stdio.h>
#include <stdlib.h>

#define MAX 100

typedef struct{
    int qtd;
    int dados[MAX];
}Lista;

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

```

void destroiLista(Lista *li){
    if(li != NULL)
        free(li);
}

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

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

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

int insereFim(Lista* li, int elem){
    if(li == NULL) return 0;
    if(!listaCheia(li)){
        li->dados[li->qtd] = elem;
        li->qtd++;
        return 1;
    }else{
        return 0;
    }
}

int insereIni(Lista* li, int elem){
    if(li == NULL) return 0;
    if(!listaCheia(li)){
        int i;
        for(i=li->qtd; i>0; i--){
            li->dados[i] = li->dados[i-1];
        }
        li->dados[0] = elem;
        li->qtd++;
        return 1;
    }else{
        return 0;
    }
}

int imprimeLista(Lista *li){

```

```

    if(li == NULL) return 0;
    int i;
    printf("Elementos: ");
    for(i=0; i<li->qtd; i++){
        printf("%d ", li->dados[i]);
    }
    printf("\n");
    return 1;
}

int removeFim(Lista* li){
    if(li == NULL) return 0;
    if(!listaVazia(li)){
        li->qtd--;
        return 1;
    }else return 0;
}

int removeIni(Lista *li){
    if(li == NULL ) return 0;
    if(!listaVazia(li)){
        int i;
        for(i=0; i<li->qtd-1; i++)
            li->dados[i] = li->dados[i+1];
        li->qtd--;
        return 1;
    } else{
        return 0;
    }
}

/* Implementacoes extras */

int procura(Lista *li, int elem) {
    if(li == NULL ) return 0;
    if(listaVazia(li)) return -1;
    int i;
    for(i = 0; i < li->qtd; i++)
        if(li->dados[i] == elem) return i;
    return -1;
}

int insereOrdenado(Lista *li, int elem) {
    if(li == NULL ) return 0;
    if(!listaCheia(li)) {
        int i, j;
        for(i = 0; i < li->qtd && li->dados[i] <= elem; i++);
        for(j = li->qtd-1; j >= i; j--)
            li->dados[j+1] = li->dados[j];
        li->dados[i] = elem;
    }
}

```

```

        li->qtd++;
        return 1;
    } else {
        return 0;
    }
}

int removeElemento(Lista *li, int elem) {
    if(li == NULL ) return 0;
    if(!listaVazia(li)) {
        int i;
        for(i = 0; i < li->qtd-1 && li->dados[i] != elem; i++);
        if(i == li->qtd) {
            return 0;
        }
        for(i += 1; i < li->qtd; i++)
            li->dados[i-1] = li->dados[i];
        li->qtd--;
        return 1;
    } else {
        return 0;
    }
}

# endif

```

main.c

```

#include <stdio.h>
#include "LE.h"

int main() {
    int a[] = {10, 7, 3, 8, 3, 1, 4, 6, 8, 17};

    Lista *li = criaLista();

    printf("Inserindo ordenado: ");
    for(int i = 0; i < sizeof(a)/sizeof(a[0]); i++) {
        printf("%d ",a[i]);
        insereOrdenado(li,a[i]);
    }
    printf("\n");
    imprimeLista(li);
    printf("\n");

    printf("7 esta presente na lista? %s\n",
        procura(li,7) > -1 ? "sim" : "nao");
    printf("81 esta presente na lista? %s\n",
        procura(li,81) > -1 ? "sim" : "nao");
}

```

```

removeElemento(li,3);
printf("\nSe removeElemento(li,3), 3 esta presente na lista
      ? %s\n",
      procura(li,3) > -1 ? "sim" : "nao");
imprimeLista(li);

removeElemento(li,3);
printf("\nSe removeElemento(li,3), 3 esta presente na lista
      ? %s\n",
      procura(li,3) > -1 ? "sim" : "nao");
imprimeLista(li);

destróiLista(li);

return 0;
}

```

```

giancarlo@giancarlo-desktop: ~/Documents/aeds...
giancarlo@giancarlo-desktop:~/Documents/aeds/roteiro4/1$ ./main
Inserindo ordenado: 10 7 3 8 3 1 4 6 8 17
Elementos: 1 3 3 4 6 7 8 8 10 17

7 esta presente na lista? sim
81 esta presente na lista? nao

Se removeElemento(li,3), 3 esta presente na lista? sim
Elementos: 1 3 4 6 7 8 8 10 17

Se removeElemento(li,3), 3 esta presente na lista? nao
Elementos: 1 4 6 7 8 8 10 17
giancarlo@giancarlo-desktop:~/Documents/aeds/roteiro4/1$

```

2 Lista Simplesmente Encadeada

LSE.h

```
/*----- File: Lista.h -----+
|Lista Sequencial Estatica      |
|                                |
|                                |
|                                |
| Implementado por Guilherme C. Pena em 12/09/2023      |
| Implementacoes extras e modificacao dos comandos      |
| de impressao feitas por Giancarlo O. Teixeira      |
+-----+ */

#ifndef LISTASE_H
#define LISTASE_H

#include <stdio.h>
#include <stdlib.h>

typedef struct NO{
    int info;
    struct NO* prox;
}NO;

typedef struct NO* Lista;

Lista* criaLista(){
    Lista *li;
    li = (Lista*) malloc (sizeof(Lista));
    if(li != NULL){
        *li = NULL;
    }
    return li;
}

int listaVazia(Lista *li){
    if(li == NULL) return 1;
    if(*li == NULL) return 1;//True - Vazia!
    return 0;//False - tem elemento!
}

NO* alocarNO(){
    return (NO*) malloc (sizeof(NO));
}

void liberarNO(NO* q){
    free(q);
}
```

```

int insereIni(Lista* li, int elem){
    if(li == NULL) return 0;
    NO* novo = alocarNO();
    if(novo == NULL) return 0;
    novo->info = elem;
    novo->prox = *li;
    *li = novo;
    return 1;
}

int insereFim(Lista* li, int elem){
    if(li == NULL) return 0;
    NO* novo = alocarNO();
    if(novo == NULL) return 0;
    novo->info = elem;
    novo->prox = NULL;
    if(listaVazia(li)){
        *li = novo;
    }else{
        NO* aux = *li;
        while(aux->prox != NULL)
            aux = aux->prox;
        aux->prox = novo;
    }
    return 1;
}

int removeIni(Lista* li){
    if(li == NULL) return 0;
    if(listaVazia(li)) return 0;
    NO* aux = *li;
    *li = aux->prox;
    liberarNO(aux);
    return 1;
}

int removeFim(Lista* li){
    if(li == NULL) return 0;
    if(listaVazia(li)) return 0;
    NO* ant, *aux = *li;
    while(aux->prox != NULL){
        ant = aux;
        aux = aux->prox;
    }
    if(aux == *li)
        *li = aux->prox;
    else
        ant->prox = aux->prox;
    liberarNO(aux);
}

```

```

        return 1;
    }

void imprimeLista(Lista* li){
    if(li == NULL) return;
    if(listaVazia(li)){
        return;
    }
    printf("Elementos: ");
    NO* aux = *li;
    while(aux != NULL){
        printf("%d ", aux->info);
        aux = aux->prox;
    }
    printf("\n");
}

void destroiLista(Lista* li){
    if(li != NULL){
        NO* aux;
        while((*li) != NULL){
            aux = *li;
            *li = (*li)->prox;
            liberarNO(aux);
        }
        free(li);
    }
}

/* Implementacoes extras */

int tamanho(Lista *li) {
    if(li == NULL) return 0;
    int tam = 0;
    NO *aux = *li;
    while(aux != NULL) {
        tam++;
        aux = aux->prox;
    }
    return tam;
}

NO *procura(Lista *li, int elem) {
    if(li == NULL) return NULL;
    NO *aux = *li;
    while(aux && aux->info != elem)
        aux = aux->prox;
    return aux;
}

```



```

int insereOrdenado(Lista *li, int elem) {
    if(li == NULL) return 0;
    NO* novo = alocarNO();
    if(novo == NULL) return 0;
    novo->info = elem;
    novo->prox = NULL;
    if(listaVazia(li)){
        *li = novo;
    }else{
        NO* aux = *li;
        if(aux->info > elem) {
            novo->prox = *li;
            *li = novo;
        } else {
            while(aux->prox && aux->prox->info < elem)
                aux = aux->prox;
            novo->prox = aux->prox;
            aux->prox = novo;
        }
    }
    return 1;
}

int removeElemento(Lista *li, int elem) {
    if(li == NULL) return 0;
    if(listaVazia(li)) return 0;
    NO* tmp, *aux = *li;
    while(aux->prox != NULL && aux->prox->info != elem)
        aux = aux->prox;
    if(aux->prox == NULL) return 0;
    tmp = aux->prox;
    aux->prox = tmp->prox;
    liberarNO(tmp);
    return 1;
}

#endif

```

main.c

```

#include <stdio.h>
#include "LSE.h"

int main() {
    int a[] = {10, 7, 3, 8, 3, 1, 4, 6, 8, 17};

    Lista *li = criaLista();

    printf("Inserindo ordenado: ");
}

```

```

    for(int i = 0; i < sizeof(a)/sizeof(a[0]); i++) {
        printf("%d ",a[i]);
        insereOrdenado(li,a[i]);
    }
    printf("\n");
    imprimeLista(li);
    printf("Tamanho: %d\n", tamanho(li));
    printf("\n");

    printf("7 esta presente na lista? %s\n",
           procura(li,7) ? "sim" : "nao");
    printf("81 esta presente na lista? %s\n",
           procura(li,81) ? "sim" : "nao");

    removeElemento(li,3);
    printf("\nSe removeElemento(li,3), 3 esta presente na lista
           ? %s\n",
           procura(li,3) ? "sim" : "nao");
    imprimeLista(li);

    removeElemento(li,3);
    printf("\nSe removeElemento(li,3), 3 esta presente na lista
           ? %s\n",
           procura(li,3) ? "sim" : "nao");
    imprimeLista(li);

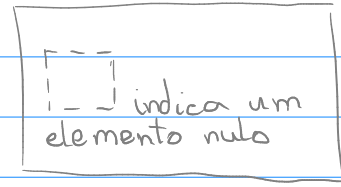
    destroiLista(li);

    return 0;
}

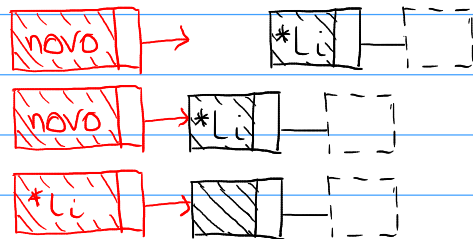
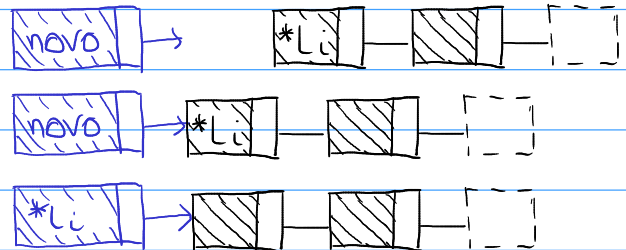
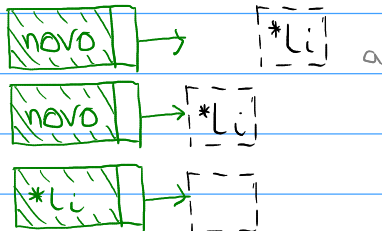
```

```
giancarlo@giancarlo-desktop: ~/Documents/aeds...  
giancarlo@giancarlo-desktop:~/Documents/aeds/roteiro4/1$ ./main  
Inserindo ordenado: 10 7 3 8 3 1 4 6 8 17  
Elementos: 1 3 3 4 6 7 8 8 10 17  
Tamanho: 10  
  
7 esta presente na lista? sim  
81 esta presente na lista? nao  
  
Se removeElemento(l1,3), 3 esta presente na lista? sim  
Elementos: 1 3 4 6 7 8 8 10 17  
  
Se removeElemento(l1,3), 3 esta presente na lista? nao  
Elementos: 1 4 6 7 8 8 10 17  
giancarlo@giancarlo-desktop:~/Documents/aeds/roteiro4/1$
```

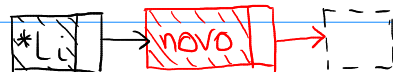
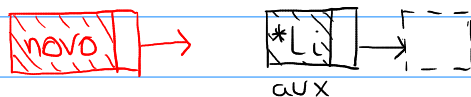
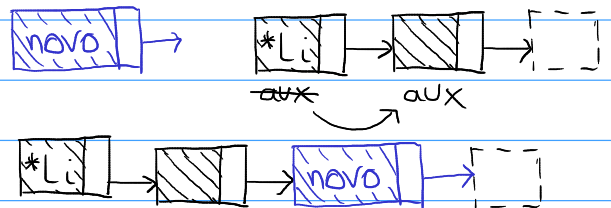
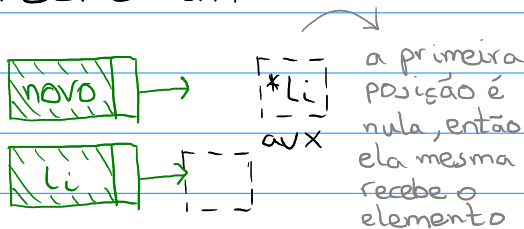
RASTREIO LSE



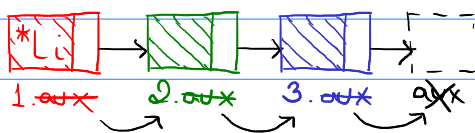
insereIni



insereFim

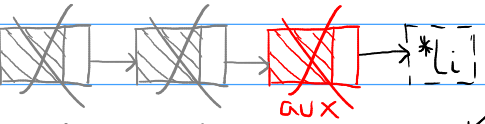
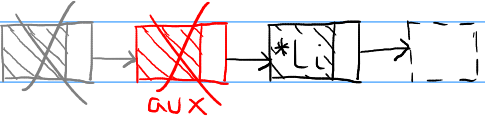
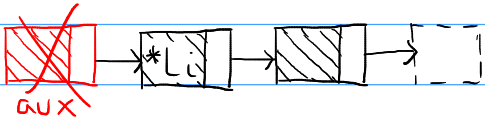
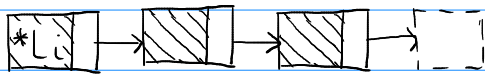


imprime



TERMINAL
1. elemento 1
2. elemento 2
3. elemento 3

destrói lista



3 Lista Duplamente Encadeada

LDE.h

```
/*----- File: Lista.h -----+
|Lista Sequencial Estatica      |
|                                |
|                                |
|                                |
| Implementado por Guilherme C. Pena em 12/09/2023      |
| Implementacoes extras e modificacao dos comandos      |
| de impressao feitas por Giancarlo O. Teixeira      |
+-----+ */

#ifndef LDE_H
#define LDE_H

#include <stdio.h>
#include <stdlib.h>

typedef struct NO{
    int info;
    struct NO* prox;
    struct NO* ant;
}NO;

typedef struct NO* Lista;

Lista* criaLista(){
    Lista *li;
    li = (Lista*) malloc (sizeof(Lista));
    if(li != NULL){
        *li = NULL;
    }
    return li;
}

int listaVazia(Lista *li){
    if(li == NULL) return 1;
    if(*li == NULL) return 1;//True - Vazia!
    return 0;//False - tem elemento!
}

NO* alocarNO(){
    return (NO*) malloc (sizeof(NO));
}

void liberarNO(NO* q){
    free(q);
}
```

```

}

int insereIni(Lista* li, int elem){
    if(li == NULL) return 0;
    NO* novo = alocarNO();
    if(novo == NULL) return 0;
    novo->info = elem;
    novo->prox = *li;
    novo->ant = NULL;
    if(!listaVazia(li))
        (*li)->ant = novo;
    *li = novo;
    return 1;
}

int insereFim(Lista* li, int elem){
    if(li == NULL) return 0;
    NO* novo = alocarNO();
    if(novo == NULL) return 0;
    novo->info = elem;
    novo->prox = NULL;
    if(listaVazia(li)){
        novo->ant = NULL;
        *li = novo;
    }else{
        NO* aux = *li;
        while(aux->prox != NULL)
            aux = aux->prox;
        aux->prox = novo;
        novo->ant = aux;
    }
    return 1;
}

int removeIni(Lista *li){
    if(li == NULL) return 0;
    if(listaVazia(li)) return 0;
    NO* aux = *li;
    *li = aux->prox;
    if(aux->prox != NULL)
        aux->prox->ant = NULL;
    liberarNO(aux);
    return 1;
}

int removeFim(Lista *li){
    if(li == NULL) return 0;
    if(listaVazia(li)) return 0;
    NO* aux = *li;
    while(aux->prox != NULL)

```

```

        aux = aux->prox;
    if(aux->ant == NULL)
        *li = aux->prox;
    else
        aux->ant->prox = NULL;
    liberarNO(aux);
    return 1;
}

void imprimeLista(Lista* li){
    if(li == NULL) return;
    if(listaVazia(li)){
        printf("Lista Vazia!\n");
        return;
    }
    printf("Elementos: ");
    NO* aux = *li;
    while(aux != NULL){
        printf("%d ", aux->info);
        aux = aux->prox;
    }
    printf("\n");
}

void destroiLista(Lista* li){
    if(li != NULL){
        NO* aux;
        while((*li) != NULL){
            aux = *li;
            *li = (*li)->prox;
            liberarNO(aux);
        }
        free(li);
    }
}

/* Implementacoes extras */

int tamanho(Lista *li) {
    if(li == NULL) return 0;
    int tam = 0;
    NO *aux = *li;
    while(aux != NULL) {
        tam++;
        aux = aux->prox;
    }
    return tam;
}

NO *procura(Lista *li, int elem) {

```



```

    if(li == NULL) return NULL;
    NO *aux = *li;
    while(aux && aux->info != elem)
        aux = aux->prox;
    return aux;
}

int insereOrdenado(Lista *li, int elem) {
    if(li == NULL) return 0;
    NO* novo = alocarNO();
    if(novo == NULL) return 0;
    novo->info = elem;
    novo->prox = NULL;
    if(listaVazia(li)){
        novo->ant = NULL;
        *li = novo;
    }else{
        NO* aux = *li;
        if(aux->info > elem) {
            novo->prox = *li;
            novo->ant = NULL;
            *li = novo;
        } else {
            while(aux->prox && aux->prox->info < elem)
                aux = aux->prox;
            novo->prox = aux->prox;
            novo->ant = aux;
            if(aux->prox) aux->prox->ant = novo;
            aux->prox = novo;
        }
    }
    return 1;
}

int removeElemento(Lista *li, int elem) {
    if(li == NULL) return 0;
    if(listaVazia(li)) return 0;
    NO* tmp, *aux = *li;
    while(aux->prox != NULL && aux->prox->info != elem)
        aux = aux->prox;
    if(aux->prox == NULL) return 0;
    tmp = aux->prox;
    aux->prox = tmp->prox;
    tmp->prox->ant = aux;
    liberarNO(tmp);
    return 1;
}

#endif

```

main.c

```
#include <stdio.h>
#include "LSE.h"

int main() {
    int a[] = {10, 7, 3, 8, 3, 1, 4, 6, 8, 17};

    Lista *li = criaLista();

    printf("Inserindo ordenado: ");
    for(int i = 0; i < sizeof(a)/sizeof(a[0]); i++) {
        printf("%d ", a[i]);
        insereOrdenado(li, a[i]);
    }
    printf("\n");
    imprimeLista(li);
    printf("Tamanho: %d\n", tamanho(li));
    printf("\n");

    printf("7 esta presente na lista? %s\n",
           procura(li, 7) ? "sim" : "nao");
    printf("81 esta presente na lista? %s\n",
           procura(li, 81) ? "sim" : "nao");

    removeElemento(li, 3);
    printf("\nSe removeElemento(li, 3), 3 esta presente na lista\n",
           ? %s\n",
           procura(li, 3) ? "sim" : "nao");
    imprimeLista(li);

    removeElemento(li, 3);
    printf("\nSe removeElemento(li, 3), 3 esta presente na lista\n",
           ? %s\n",
           procura(li, 3) ? "sim" : "nao");
    imprimeLista(li);

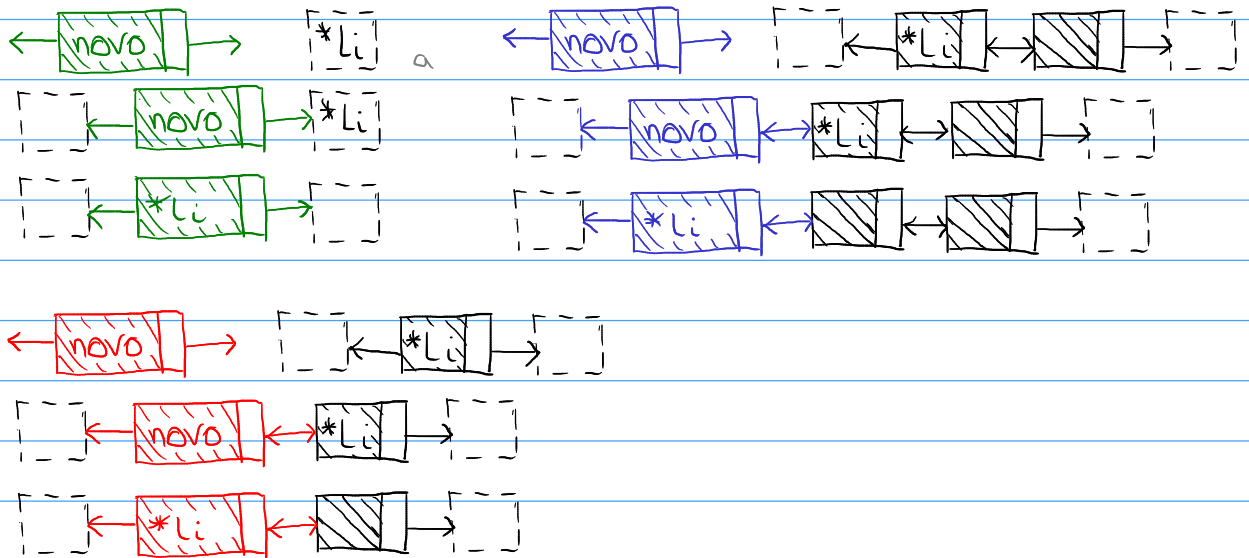
    destroiLista(li);

    return 0;
}
```

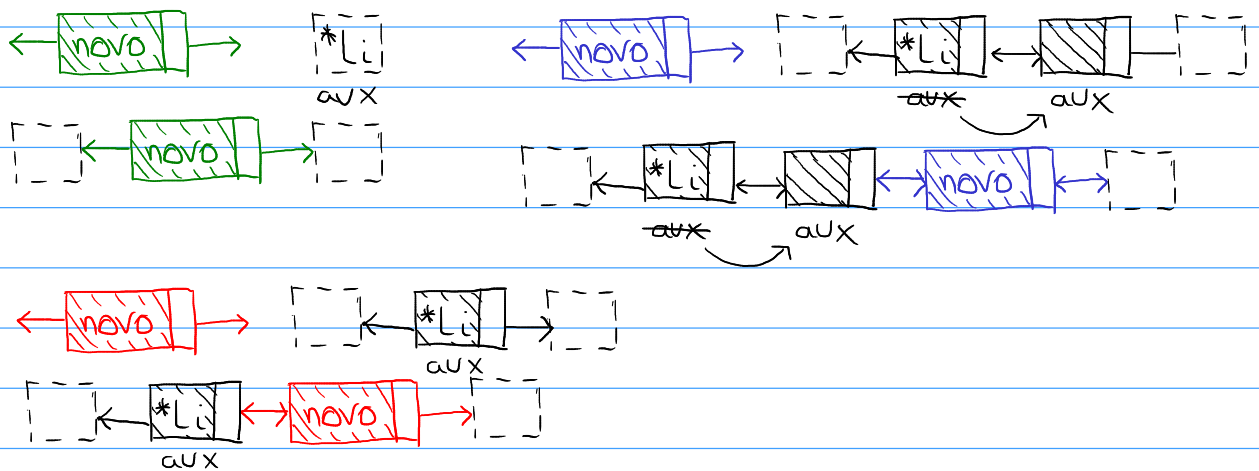
```
giancarlo@giancarlo-desktop: ~/Documents/aeds...  
giancarlo@giancarlo-desktop:~/Documents/aeds/roteiro4/1$ ./main  
Inserindo ordenado: 10 7 3 8 3 1 4 6 8 17  
Elementos: 1 3 3 4 6 7 8 8 10 17  
Tamanho: 10  
  
7 esta presente na lista? sim  
81 esta presente na lista? nao  
  
Se removeElemento(l1,3), 3 esta presente na lista? sim  
Elementos: 1 3 4 6 7 8 8 10 17  
  
Se removeElemento(l1,3), 3 esta presente na lista? nao  
Elementos: 1 4 6 7 8 8 10 17  
giancarlo@giancarlo-desktop:~/Documents/aeds/roteiro4/1$
```

RASTREIO LDE

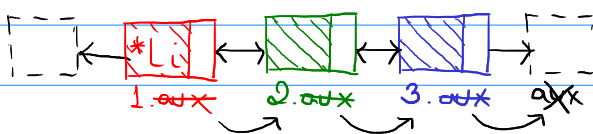
insereIni



insereFim

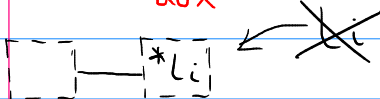
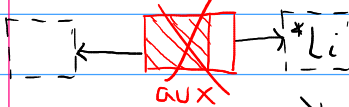
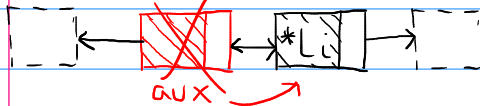
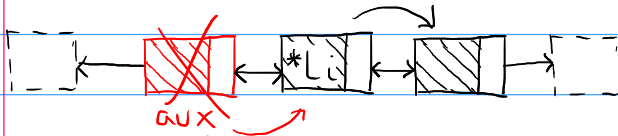
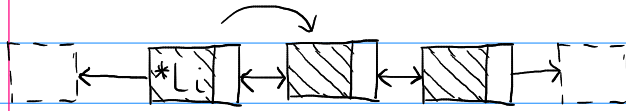


imprime



TERMINAL
1. elemento 1
2. elemento 2
3. elemento 3

destrói lista



4 Lista Circular Simplesmente Encadeada

LCSE.h

```
/*----- File: Lista.h -----+
/Lista Sequencial Estatica      /
/
/
/
/ Implementado por Guilherme C. Pena em 12/09/2023      /
/ Implementacoes extras e modificacao dos comandos      /
/ de impressao feitas por Giancarlo O. Teixeira      /
+-----+ */

#ifndef LDE_H
#define LDE_H

#include <stdio.h>
#include <stdlib.h>

typedef struct NO{
    int info;
    struct NO* prox;
    struct NO* ant;
}NO;

typedef struct NO* Lista;

Lista* criaLista(){
    Lista *li;
    li = (Lista*) malloc (sizeof(Lista));
    if(li != NULL){
        *li = NULL;
    }
    return li;
}

int listaVazia(Lista *li){
    if(li == NULL) return 1;
    if(*li == NULL) return 1;//True - Vazia!
    return 0;//False - tem elemento!
}

NO* alocarNO(){
    return (NO*) malloc (sizeof(NO));
}

void liberarNO(NO* q){
    free(q);
}
```

```

}

int insereIni(Lista* li, int elem){
    if(li == NULL) return 0;
    NO* novo = alocarNO();
    if(novo == NULL) return 0;
    novo->info = elem;
    novo->prox = *li;
    novo->ant = NULL;
    if(!listaVazia(li))
        (*li)->ant = novo;
    *li = novo;
    return 1;
}

int insereFim(Lista* li, int elem){
    if(li == NULL) return 0;
    NO* novo = alocarNO();
    if(novo == NULL) return 0;
    novo->info = elem;
    novo->prox = NULL;
    if(listaVazia(li)){
        novo->ant = NULL;
        *li = novo;
    }else{
        NO* aux = *li;
        while(aux->prox != NULL)
            aux = aux->prox;
        aux->prox = novo;
        novo->ant = aux;
    }
    return 1;
}

int removeIni(Lista *li){
    if(li == NULL) return 0;
    if(listaVazia(li)) return 0;
    NO* aux = *li;
    *li = aux->prox;
    if(aux->prox != NULL)
        aux->prox->ant = NULL;
    liberarNO(aux);
    return 1;
}

int removeFim(Lista *li){
    if(li == NULL) return 0;
    if(listaVazia(li)) return 0;
    NO* aux = *li;
    while(aux->prox != NULL)

```

```

        aux = aux->prox;
    if(aux->ant == NULL)
        *li = aux->prox;
    else
        aux->ant->prox = NULL;
    liberarNO(aux);
    return 1;
}

void imprimeLista(Lista* li){
    if(li == NULL) return;
    if(listaVazia(li)){
        printf("Lista Vazia!\n");
        return;
    }
    printf("Elementos: ");
    NO* aux = *li;
    while(aux != NULL){
        printf("%d ", aux->info);
        aux = aux->prox;
    }
    printf("\n");
}

void destroiLista(Lista* li){
    if(li != NULL){
        NO* aux;
        while((*li) != NULL){
            aux = *li;
            *li = (*li)->prox;
            liberarNO(aux);
        }
        free(li);
    }
}

/* Implementacoes extras */

int tamanho(Lista *li) {
    if(li == NULL) return 0;
    int tam = 0;
    NO *aux = *li;
    while(aux != NULL) {
        tam++;
        aux = aux->prox;
    }
    return tam;
}

NO *procura(Lista *li, int elem) {

```



```

    if(li == NULL) return NULL;
    NO *aux = *li;
    while(aux && aux->info != elem)
        aux = aux->prox;
    return aux;
}

int insereOrdenado(Lista *li, int elem) {
    if(li == NULL) return 0;
    NO* novo = alocarNO();
    if(novo == NULL) return 0;
    novo->info = elem;
    novo->prox = NULL;
    if(listaVazia(li)){
        novo->ant = NULL;
        *li = novo;
    }else{
        NO* aux = *li;
        if(aux->info > elem) {
            novo->prox = *li;
            novo->ant = NULL;
            *li = novo;
        } else {
            while(aux->prox && aux->prox->info < elem)
                aux = aux->prox;
            novo->prox = aux->prox;
            novo->ant = aux;
            if(aux->prox) aux->prox->ant = novo;
            aux->prox = novo;
        }
    }
    return 1;
}

int removeElemento(Lista *li, int elem) {
    if(li == NULL) return 0;
    if(listaVazia(li)) return 0;
    NO* tmp, *aux = *li;
    while(aux->prox != NULL && aux->prox->info != elem)
        aux = aux->prox;
    if(aux->prox == NULL) return 0;
    tmp = aux->prox;
    aux->prox = tmp->prox;
    tmp->prox->ant = aux;
    liberarNO(tmp);
    return 1;
}

#endif

```

main.c

```
#include <stdio.h>
#include "LCSE.h"

int main() {
    int a[] = {10, 7, 3, 8, 3, 1, 4, 6, 8, 17};

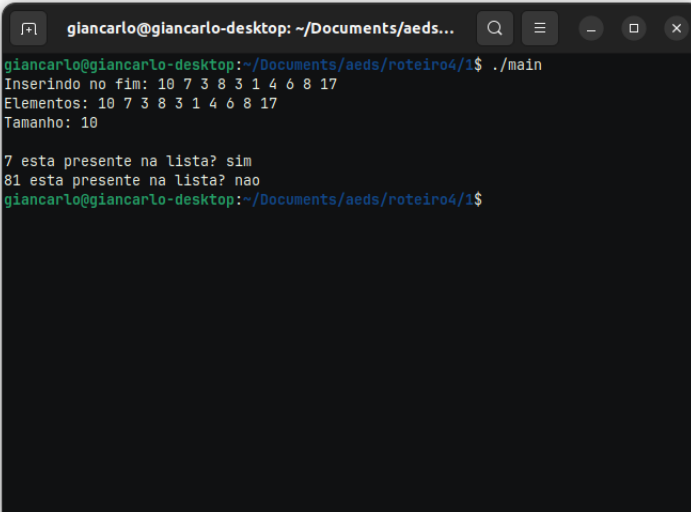
    Lista *li = criaLista();

    printf("Inserindo no fim: ");
    for(int i = 0; i < sizeof(a)/sizeof(a[0]); i++) {
        printf("%d ", a[i]);
        insereFim(li, a[i]);
    }
    printf("\n");
    imprimeLista(li);
    printf("Tamanho: %d\n", tamanho(li));
    printf("\n");

    printf("7 esta presente na lista? %s\n",
        procura(li, 7) ? "sim" : "nao");
    printf("81 esta presente na lista? %s\n",
        procura(li, 81) ? "sim" : "nao");

    destroiLista(li);

    return 0;
}
```

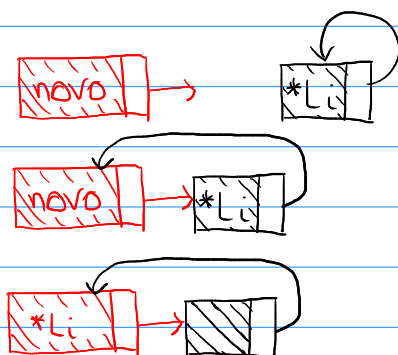
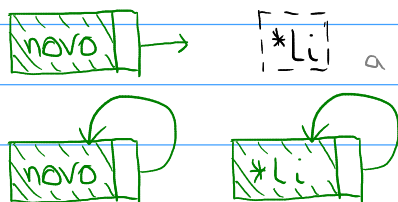


```
giancarlo@giancarlo-desktop: ~/Documents/aeds...
giancarlo@giancarlo-desktop:~/Documents/aeds/roteiro4/1$ ./main
Inserindo no fim: 10 7 3 8 3 1 4 6 8 17
Elementos: 10 7 3 8 3 1 4 6 8 17
Tamanho: 10

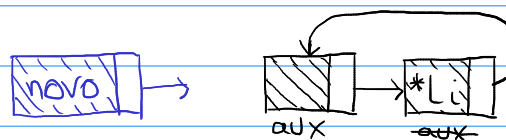
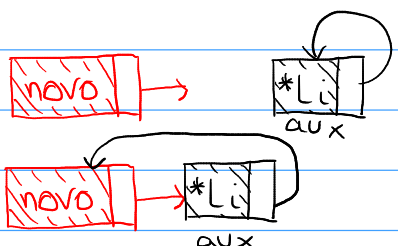
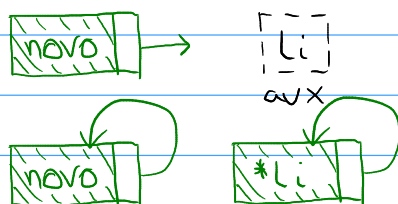
7 esta presente na lista? sim
81 esta presente na lista? nao
giancarlo@giancarlo-desktop:~/Documents/aeds/roteiro4/1$
```

RASTREIO LCSE

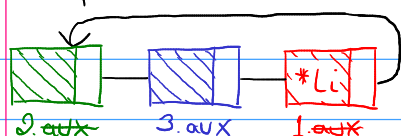
insereIni



insereFim



imprime



TERMINAL	
1.	elemento 1
2.	elemento 2
3.	elemento 3

destrói lista

