

Listas Sequenciais

/* Programa que manipula uma lista ligada com inserção no final e remoção tanto na cabeça,
Quanto no final. */

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define nroNo 20
#define lv -1

struct tNo {
    int info;
    int prox;
};

struct tNo no[nroNo];
int disp = 0; // inicializando a memoria
int p, l1;

void aloca_memoria()
{
    int k;
    for (k=0; k < (nroNo - 1); k++){
        no[k].prox = k + 1;
        printf("%d ",no[k].prox);
    }
    no[nroNo - 1].prox = lv;
    disp = 0;
    l1 = lv;
}

int pega_no()
{
    int i;
    if(disp == lv) {
        printf("lista cheia\n");
        getch();
        exit(1);
    }
    else{
        i = disp;
        /* no[i].prox = lv; */
        disp = no[disp].prox;
    }
    return(i);
}

void libera_no(int x)
{
    no[x].prox = disp;
```

Listas Sequenciais

```
        disp = x;
        return;
    }

/* funcao que insere um elemento no final */
int insere_fim(int l, int x)
{
    int p, aux;
    p = l;
    aux = pega_no();
    no[aux].info = x;
    no[aux].prox = lv;
    if (p == lv) {
        l = aux;
    }
    else {
        while(no[p].prox != lv) {
            p = no[p].prox;
        }
        no[p].prox = aux;
    }
    return(l);
}

/* funcao que remove o ultimo elemento */
int remove_ultimo(int l)
{
    int p, aux, aux2, x;
    p = l;
    aux2 = lv;
    aux = lv;
    if(p != lv){
        // lista nao eh vazia
        while(p != lv) {
            aux2 = aux;
            aux = p;
            p = no[p].prox;
        }
        if(aux == lv){
            // so tem 1 elemento na lista
            x = no[aux].info;
            libera_no(aux);
            printf("\nElemento removido = %d",x);
            p = lv;
        }
        else{
            // tem mais de um elemento na lista
            x = no[aux].info;
            no[aux2].prox = lv;
            libera_no(aux);
        }
    }
}
```



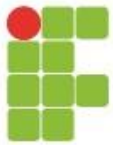
Listas Sequenciais

```
        printf("\nElemento removido = %d",x);
        p = l;
    }
}
else {
    printf("lista vazia");
    p = lv;
}
return(p);
}
```

```
/* funcao que remove o primeiro elemento */
int remove_cabeca(int l)
{
    int p, aux, aux2, x;
    p = l;
    if(p != lv){
        // lista nao eh vazia
        x = no[p].info;
        p = no[p].prox;
        libera_no(l);
        printf("\nElemento removido = %d",x);
        l = p;
    }
    else {
        printf("lista vazia");
    }
    return(l);
}
```

```
/* funcao que imprime a lista */
void imprime(int l)
{
    int q;
    q = l;
    printf("\n ---- lista atual ----");
    printf("\n[ ");
    while(q != lv){
        printf("%d ",no[q].info);
        q = no[q].prox;
    }
    printf(" ]");
}
```

```
int main()
{
    int x;
    system("cls");
    printf("tecle enter");
    getch();
}
```



Listas Sequenciais

```
printf("\n----- Alocação da Memória ----- \n");
aloca_memoria();
printf("\n Disp = %d L1 = %d ", disp, l1);
printf("\n----- Inserção dos elementos na lista -----");
l1 = lv;
printf("\n V= 10, Disp_p = %d L1 = %d ", disp, l1);
l1 = insere_fim(l1, 10);
printf("\n V= 20, Disp_p = %d L1 = %d ", disp, l1);
l1 = insere_fim(l1, 20);
printf("\n V= 30, Disp_p = %d L1 = %d ", disp, l1);
l1 = insere_fim(l1, 30);
printf("\n V= 40, Disp_p = %d L1 = %d ", disp, l1);
l1 = insere_fim(l1, 40);
printf("\n V= 50, Disp_p = %d L1 = %d ", disp, l1);
l1 = insere_fim(l1, 50);
printf("\n ");
getch();
printf("\n----- Impressão da lista -----");
imprime(l1);
printf("\n----- Remoção dos elementos na lista -----");
printf("\n Removendo o último elemento da lista");
l1 = remove_ultimo(l1);
imprime(l1);
printf("\n Removendo o primeiro elemento da lista");
l1 = remove_cabeca(l1);
imprime(l1);
printf("\n Removendo o último elemento da lista");
l1 = remove_ultimo(l1);
//printf("\n Elemento removido = %d ", x);
imprime(l1);
printf("\n Removendo o primeiro elemento da lista");
l1 = remove_cabeca(l1);
//printf("\n Elemento removido = %d ", x);
printf("\n----- Impressão da lista -----");
imprime(l1);
printf("\n -- fim --");
getch();
```

}