file:Lista01.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "lista01.h"

int insereInicio(int valor, FormatoLst* argLista){
    FormatoNo * novoElemento = (FormatoNo *)malloc(sizeof(FormatoNo));
    novoElemento->Info = valor;
    novoElemento->Link = argLista->IndIniLst;
    argLista->IndIniLst = novoElemento;
     if(argLista->IndFimLst== NULL){
        argLista->IndFimLst= argLista->IndIniLst;
        printf("insereInicio: Condição dada quando a lista está vazia antes de inserir um elemento.
\n");
     }
     return 0;
}


void insereFinal(int valor, FormatoLst * argLista){
   FormatoNo * novoElemento = (FormatoNo *)malloc(sizeof(FormatoNo));
   novoElemento->Info = valor;
   novoElemento->Link = NULL;
   if(argLista->IndIniLst == NULL){
          argLista->IndIniLst = novoElemento;
          printf("insereFinalLst: Condição dada quando a lista está vazia antes de inserir um
elemento \n");
   }
   else{
      argLista->IndFimLst->Link = novoElemento;
   }
   argLista->IndFimLst = novoElemento;
}

int removeInicio(FormatoLst * argLista){
    if(argLista->IndIniLst == NULL){
       return -1;
    }
    int tmp = argLista->IndIniLst->Info;
    FormatoNo * removido = argLista->IndIniLst;
    argLista->IndIniLst = argLista->IndIniLst->Link;
    free(removido);
    if(argLista->IndIniLst == NULL){
        argLista->IndFimLst= NULL;
    }
    return tmp;
}

int estaVazia(FormatoLst * argLista){
    if(argLista->IndIniLst == NULL){
       return 1;
    }
    return 0;
}

int removeFinal(FormatoLst * argLista){
    if(argLista->IndIniLst == NULL){
       return -1;
    }
    int tmp = argLista->IndFimLst->Info;
    FormatoNo * ultimo = argLista->IndIniLst;
    FormatoNo * penultimo = NULL;
    while(ultimo->Link != NULL){
         penultimo = ultimo;
         ultimo = ultimo->Link;
    }
    if(penultimo != NULL){
```

```
        penultimo->Link = NULL;
        argLista->IndFimLst = penultimo;

    }else{//lista possui apenas um elemento
        argLista->IndIniLst = NULL;
        argLista->IndFimLst = NULL;
    }
    free(ultimo);
    return tmp;

}

void inserir(int pos, int valor, FormatoLst * argLista){
     if( pos <= 0){
        insereInicio(valor,argLista);
     }else{
         FormatoNo * atual = argLista->IndIniLst;
         int i = 0;
         for(i = 0; i < (pos-1) && atual != NULL ;i++){
               atual = atual->Link;
         }
         if(atual == NULL || atual == argLista->IndFimLst){
              insereFinal(valor,argLista);
         } else{
              FormatoNo * novo = (FormatoNo *)malloc(sizeof(FormatoNo));
              novo->Info = valor;
              novo->Link = atual->Link;
              atual->Link = novo;
         }
     }
}


int remover(int pos, FormatoLst * argLista){
        if(estaVazia(argLista)){
               return -1;
        }
        if(pos < 0){
               return -1;
        }
        FormatoNo * removido = argLista->IndIniLst;
        FormatoNo * ant_removido = NULL;
        int i = 0;
        for (i = 0; i < pos && removido != NULL; i++) {
               ant_removido = removido;
               removido = removido->Link;
        }
        if(removido != NULL){
               if(removido == argLista->IndIniLst){
                      argLista->IndIniLst = removido->Link;
               }else{
                      ant_removido->Link = removido->Link;
               }
               if(removido == argLista->IndFimLst){
                      argLista->IndFimLst = ant_removido;
               }
               int tmp = removido->Info;
               free(removido);
               return tmp;
        }
        return -1;
}


int busca(int valor, FormatoLst * argLista){
    FormatoNo * atual = argLista->IndIniLst;
    FormatoNo * anterior = NULL;
    while(atual != NULL && atual->Info != valor){
       anterior = atual;
       atual = atual->Link;
```

```
    }
    if(atual == NULL){
        return -1;
    }else if (atual == argLista->IndIniLst){
        return atual->Info;
    }else if (atual == argLista->IndFimLst){
       argLista->IndFimLst = anterior;
    }
    anterior->Link = atual->Link;
    atual->Link = argLista->IndIniLst;
    argLista->IndIniLst = atual;
    return atual->Info;
}

int busca1(int valor, FormatoLst * argLista){
    FormatoNo * atual = argLista->IndIniLst;
    int cont = 0;
    while(atual != NULL && atual->Info != valor){

       atual = atual->Link;
       cont++;
    }
    if(atual == NULL){
        return -1;
    }
    int ret = remover(cont,argLista);
    insereInicio(ret,argLista);
    return ret;

}


void imprimeLista(FormatoLst * argLista){
   FormatoNo * atual = argLista->IndIniLst;
    printf("Lista:");
    while(atual != NULL){
        printf(" %d",atual->Info);
        atual = atual->Link;
    }
    printf("\n");

}
```

file:Lista01.h

```
typedef struct elemento{int Info; struct elemento * Link;} FormatoNo; // Representa um nó da lista
typedef struct lista{FormatoNo * IndIniLst;  FormatoNo * IndFimLst; }FormatoLst; // Representa uma
lista

//Função para inserir elemento no inicio da lista
int insereInicio(int valor, FormatoLst* argLista);
//Função para inserir elemento no final da lista
void insereFinal(int valor, FormatoLst * argLista);

int removeInicio(FormatoLst * argLista);
int estaVazia(FormatoLst * argLista);
int removeFinal(FormatoLst * argLista);
void inserir(int pos, int valor, FormatoLst * argLista);
int remover(int pos, FormatoLst * argLista);
int busca(int valor, FormatoLst * argLista);
int busca1(int valor, FormatoLst * argLista);
void imprimeLista(FormatoLst * argLista);
```

file:main.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "lista01.h"

int main(int argc, char* argv[])
{

FormatoLst * argLista = (FormatoLst *)malloc(sizeof(FormatoLst));
insereInicio(5,argLista);
insereFinal(1,argLista);
insereFinal(5,argLista);
insereFinal(15,argLista);
imprimeLista(argLista);
inserir(1,30,argLista);
imprimeLista(argLista);
printf("Busca pelo 30: %d\n",busca1(30,argLista));
imprimeLista(argLista);
printf("Numero removido pos 2: %d\n",remover(2,argLista));
imprimeLista(argLista);
printf("IndFimLstLst!!!\n");
}
```