file:phila00.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "pilha00.h"

FormatoLst * criarLista(){
   FormatoLst * nova = (FormatoLst *)malloc(sizeof(FormatoLst));
   nova->IndIniLst = NULL;
   nova->IndFimLst = NULL;
   return nova;
}


FormatoPlha * criarPilha(){
   FormatoPlha * novapilha = (FormatoPlha *)malloc(sizeof(FormatoPlha));
   novapilha->argLista =  criarLista();
   return novapilha;
}

int insereInicio(int valor, FormatoLst * argLista){
    FormatoNo * novoprimeiro = (FormatoNo *)malloc(sizeof(FormatoNo));
     novoprimeiro->info = valor;
    novoprimeiro->link = argLista->IndIniLst;
    argLista->IndIniLst = novoprimeiro;
     if(argLista->IndFimLst == NULL){ //argLista->IndIniLst->link == NULL ou novoprimeiro->link == NULL
        argLista->IndFimLst = argLista->IndIniLst;
     }
     return 0;
}


void insereFinal(int valor, FormatoLst * argLista){
   FormatoNo * novoultimo = (FormatoNo *)malloc(sizeof(FormatoNo));
   novoultimo->info = valor;
   novoultimo->link = NULL;
   if(argLista->IndIniLst == NULL){ //Condição dada quando a lista está vazia antes de inserir um
elemento
           argLista->IndIniLst = novoultimo;
   }else{
           argLista->IndFimLst->link = novoultimo;
   }
   argLista->IndFimLst = novoultimo;
}


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;
}


void empilhar(int valor, FormatoPlha * p){
        insereInicio(valor, p->argLista);
}


int desempilhar(FormatoPlha * p){
    return removeInicio(p->argLista);
}


int estaVaziaPilha(FormatoPlha * p){
    if(p->argLista->IndIniLst == NULL){
       return 1;
   }
   return 0;
}
```

file:phila00.h

```
typedef struct elemento{
        int info;
        struct elemento * link;
        }FormatoNo;


typedef struct lista{
        FormatoNo * IndIniLst;
        FormatoNo * IndFimLst;
}FormatoLst;


typedef struct pilha{
        FormatoLst * argLista;
}FormatoPlha;


FormatoLst * criarLista();
FormatoPlha * criarPilha();
int insereIndIniLst(int valor, FormatoLst * argLista);
void insereFinal(int valor, FormatoLst * argLista);
int removeIndIniLst(FormatoLst * argLista);
int estaVazia(FormatoLst * argLista);
int removeFinal(FormatoLst * argLista);
void inserir(int pos, int valor, FormatoLst * argLista);
int remover(int pos, FormatoLst * argLista);
void empilhar(int valor, FormatoPlha * p);
int desempilhar(FormatoPlha * p);
int estaVaziaPilha(FormatoPlha * p);
int main (int argc, char *argv[]);
```

file: main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "pilha00.h"

int main (int argc, char *argv[]){
    /*FormatoLst * l = (FormatoLst *)malloc(sizeof(FormatoLst));
```

```
    insereFinal(1,l);
    insereFinal(2,l);
    insereFinal(3,l);
    inserir(2,30,l);
    //printf("Numero removido pos 2: %d\n",remover(2,l));

    while(!estaVazia(l)){
       int rmv = removeIndIniLst(l);
       printf("Numero removido: %d\n",rmv);
    }*/
    FormatoPlha * p1 = criarPilha();
    empilhar(1,p1);
    empilhar(2,p1);
    while(!estaVaziaPilha(p1)){
      printf("Numero desempilhado: %d\n",desempilhar(p1));
   }


    printf("IndFimLst!!!\n");
}
```