**MAIN**

#include <stdio.h>

#include <stdlib.h>

#include "ListaDinEncadDupla.h"

int main(){

struct aluno al, a[4] = {{2,"Andre",9.5,7.8,8.5},

{4,"Ricardo",7.5,8.7,6.8},

{1,"Bianca",9.7,6.7,8.4},

{3,"Ana",5.7,6.1,7.4}};

Lista\* li = cria\_lista();

printf("Tamanho: %d\n\n\n\n",tamanho\_lista(li));

int i;

for(i=0; i < 4; i++)

insere\_lista\_ordenada(li,a[i]);

imprime\_lista(li);

printf("\n\n\n\n Tamanho: %d\n",tamanho\_lista(li));

//remove\_lista(li,3);

for(i=0; i < 4; i++){

remove\_lista\_final(li);

imprime\_lista(li);

printf("\nTamanho: %d\n\n\n\n",tamanho\_lista(li));

}

for(i=0; i < 4; i++)

insere\_lista\_ordenada(li,a[i]);

imprime\_lista(li);

libera\_lista(li);

system("pause");

return 0;

}

**ListaDinEncad**Dupla**.C**

#include <stdio.h>

#include <stdlib.h>

#include "ListaDinEncadDupla.h" //inclui os Protótipos

//Definição do tipo lista

struct elemento{

struct elemento \*ant;

struct aluno dados;

struct elemento \*prox;

};

typedef struct elemento Elem;

Lista\* cria\_lista(){

Lista\* li = (Lista\*) malloc(sizeof(Lista));

if(li != NULL)

\*li = NULL;

return li;

}

void libera\_lista(Lista\* li){

if(li != NULL){

Elem\* no;

while((\*li) != NULL){

no = \*li;

\*li = (\*li)->prox;

free(no);

}

free(li);

}

}

int consulta\_lista\_pos(Lista\* li, int pos, struct aluno \*al){

if(li == NULL || pos <= 0)

return 0;

Elem \*no = \*li;

int i = 1;

while(no != NULL && i < pos){

no = no->prox;

i++;

}

if(no == NULL)

return 0;

else{

\*al = no->dados;

return 1;

}

}

int consulta\_lista\_mat(Lista\* li, int mat, struct aluno \*al){

if(li == NULL)

return 0;

Elem \*no = \*li;

while(no != NULL && no->dados.matricula != mat){

no = no->prox;

}

if(no == NULL)

return 0;

else{

\*al = no->dados;

return 1;

}

}

int insere\_lista\_final(Lista\* li, struct aluno al){

if(li == NULL)

return 0;

Elem \*no;

no = (Elem\*) malloc(sizeof(Elem));

if(no == NULL)

return 0;

no->dados = al;

no->prox = NULL;

if((\*li) == NULL){//lista vazia: insere início

no->ant = NULL;

\*li = no;

}else{

Elem \*aux;

aux = \*li;

while(aux->prox != NULL){

aux = aux->prox;

}

aux->prox = no;

no->ant = aux;

}

return 1;

}

int insere\_lista\_inicio(Lista\* li, struct aluno al){

if(li == NULL)

return 0;

Elem\* no;

no = (Elem\*) malloc(sizeof(Elem));

if(no == NULL)

return 0;

no->dados = al;

no->prox = (\*li);

no->ant = NULL;

if(\*li != NULL)//lista não vazia: apontar para o anterior!

(\*li)->ant = no;

\*li = no;

return 1;

}

int insere\_lista\_ordenada(Lista\* li, struct aluno al){

if(li == NULL)

return 0;

Elem \*no = (Elem\*) malloc(sizeof(Elem));

if(no == NULL)

return 0;

no->dados = al;

if((\*li) == NULL){//lista vazia: insere início

no->prox = NULL;

no->ant = NULL;

\*li = no;

return 1;

}

else{

Elem \*ante, \*atual = \*li;

while(atual != NULL && atual->dados.matricula < al.matricula){

ante = atual;

atual = atual->prox;

}

if(atual == \*li){//insere início

no->ant = NULL;

(\*li)->ant = no;

no->prox = (\*li);

\*li = no;

}else{

no->prox = ante->prox;

no->ant = ante;

ante->prox = no;

if(atual != NULL)

atual->ant = no;

}

return 1;

}

}

int remove\_lista(Lista\* li, int mat){//TERMINAR

if(li == NULL)

return 0;

if((\*li) == NULL)//lista vazia

return 0;

Elem \*no = \*li;

while(no != NULL && no->dados.matricula != mat){

no = no->prox;

}

if(no == NULL)//não encontrado

return 0;

if(no->ant == NULL)//remover o primeiro?

\*li = no->prox;

else

no->ant->prox = no->prox;

if(no->prox != NULL)//não é o último?

no->prox->ant = no->ant;

free(no);

return 1;

}

int remove\_lista\_inicio(Lista\* li){

if(li == NULL)

return 0;

if((\*li) == NULL)//lista vazia

return 0;

Elem \*no = \*li;

\*li = no->prox;

if(no->prox != NULL)

no->prox->ant = NULL;

free(no);

return 1;

}

int remove\_lista\_final(Lista\* li){

if(li == NULL)

return 0;

if((\*li) == NULL)//lista vazia

return 0;

Elem \*no = \*li;

while(no->prox != NULL)

no = no->prox;

if(no->ant == NULL)//remover o primeiro e único

\*li = no->prox;

else

no->ant->prox = NULL;

free(no);

return 1;

}

int tamanho\_lista(Lista\* li){

if(li == NULL)

return 0;

int cont = 0;

Elem\* no = \*li;

while(no != NULL){

cont++;

no = no->prox;

}

return cont;

}

int lista\_cheia(Lista\* li){

return 0;

}

int lista\_vazia(Lista\* li){

if(li == NULL)

return 1;

if(\*li == NULL)

return 1;

return 0;

}

void imprime\_lista(Lista\* li){

if(li == NULL)

return;

Elem\* no = \*li;

while(no != NULL){

printf("Matricula: %d\n",no->dados.matricula);

printf("Nome: %s\n",no->dados.nome);

printf("Notas: %f %f %f\n",no->dados.n1,

no->dados.n2,

no->dados.n3);

printf("-------------------------------\n");

no = no->prox;

}

}

ListaDinEncadDupla.h

//Arquivo ListaDinEncadDupla.h

struct aluno{

int matricula;

char nome[30];

float n1,n2,n3;

};

typedef struct elemento\* Lista;

Lista\* cria\_lista();

void libera\_lista(Lista\* li);

int consulta\_lista\_pos(Lista\* li, int pos, struct aluno \*al);

int consulta\_lista\_mat(Lista\* li, int mat, struct aluno \*al);

int insere\_lista\_final(Lista\* li, struct aluno al);

int insere\_lista\_inicio(Lista\* li, struct aluno al);

int insere\_lista\_ordenada(Lista\* li, struct aluno al);

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\_vazia(Lista\* li);

void imprime\_lista(Lista\* li);