1º)

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct Tnumero

{

char nome[100],rg[16];

struct Tnumero \*prox,\*ant;

}TContato;

int procurar(TContato \*L,char nome[]);

void criarNO(TContato \*L,char nome[]); // DOne !

void Adicionar(TContato \*\*a,char nome[]); // Done !

void Cancelar\_Registro(TContato \*\*a);

void Exibir\_Lista(TContato \*a);

int qtdPessoas(TContato \*a);

void Remover\_Pessoa(TContato \*\*a,char rg[]);

void Remover\_Tudo(TContato \*\*a);

int main(){

TContato \*Lista=NULL;

int resp;

char op,rgaux[16],nomeaux[100];

do{

fflush(stdin);

printf("1-Registrar nova pessoa no predio\n"

"2-Cancelar ultimo Registro\n"

"3-Listar pessoas do predio\n"

"4-Verificar pessoa no predio\n"

"5-Exibir Total de pessoas no predio\n"

"6-Remover pessoa do predio\n"

"7-Sair\n"

"Digite sua opcao: ");

op = getchar();fflush(stdin);

switch(op){

case '1':

printf("\nDigite o RG da pessoa: ");

gets(rgaux);

Adicionar(&Lista,rgaux);

break;

case '2':

Cancelar\_Registro(&Lista);

break;

case '3':

Exibir\_Lista(Lista);

break;

case '4':

printf("\nDigite o nome do contato: ");

gets(nomeaux);

resp = procurar(Lista,nomeaux);

if(resp == 1)

printf("\nPessoa Encontrada no predio !\n");

else

printf("\nPessoa nao se encontrada no predio !\n");

break;

case '5':

resp = qtdPessoas(Lista);

printf("\nExistem %d pessoas no predio.\n",resp);

break;

case '6':

printf("\nDigite o RG da pessoa: ");

gets(rgaux);

Remover\_Pessoa(&Lista,rgaux);

break;

case '7':

break;

default:

printf("\nOpcao invalida!\n");

break;

}

}while(op!='7');

Remover\_Tudo(&Lista);

return 0;

}

int procurar(TContato \*L,char nome[]){

TContato \*aux ,\*aux2;

while(L!=NULL){

if(strcmp(nome,L->nome) == 0)

return 1;

if(strcmp(nome,L->nome)<0)

break;

L=L->prox;

}

return 0;

}

void criarNO(TContato \*\*L,char rg[]){

\*L = (TContato \*) malloc(sizeof(TContato));

strcpy((\*L)->rg,rg);fflush(stdin);

printf("Digite o Nome : ");

gets((\*L)->nome);fflush(stdin);

}

void Adicionar(TContato \*\*a,char rg[]){// add no fim, [L.Duplamente.E] (Sem repetidos)

TContato \*novo,\*aux,\*aux2;

int resp;

if(\*a == NULL){ // lista vazia

criarNO(&novo,rg);

novo->prox = NULL;

novo->ant = NULL;

\*a=novo;

}

else{

aux = \*a;

if(strcmp(aux->rg,rg)==0){//verificar o 1º

printf("\nRG repetido! Cadastro Cancelado!\n");

return;

}

aux2 = aux->prox;

while(aux2!=NULL){

if(strcmp(aux2->rg,rg)==0){

printf("\nRG repetido! Cadastro Cancelado!\n");

return;

}

aux = aux2;

aux2 = aux2->prox;

}

criarNO(&novo,rg);

aux->prox = novo;

novo ->prox = NULL;

novo ->ant = aux;

}

}

void Cancelar\_Registro(TContato \*\*a){// remover no fim , L.Duplamente.E

TContato \*aux;

int i=0;

if(\*a == NULL){// lista vazia

printf("\nLista Vazia!\n");

}

else{

aux = \*a;

if(aux ->ant == NULL){ // 1 nó

\*a = NULL;

free(aux);

printf("\nUltimo Registro Cancelado!\n");

return;

}

else{// geral

while(aux->prox!=NULL){

aux = aux ->prox;

}

aux->ant->prox == NULL;

free(aux);

}

}

}

int qtdPessoas(TContato \*a){

int qtd=0;

while(a!=NULL){

qtd ++;

a = a->prox;

}

return qtd;

}

void Exibir\_Lista(TContato \*a){

TContato \*aux = a;

if(a == NULL)

printf("Lista Vazia!\n");

else{

do{

printf("Nome: %s\n",aux->nome);

printf("RG: %s\n",aux->rg);

aux = aux -> prox;

}while(aux != NULL);

}

}

void Remover\_Pessoa(TContato \*\*a,char rg[]){

TContato \*aux;

if(\*a == NULL){

printf("\nLista vazia !\n");

}

else{

aux = \*a;

while(aux != NULL){

if(strcmp(aux->rg,rg)==0){

aux->ant->prox = aux->prox;

aux->prox->ant = aux->ant;

free(aux);

printf("\nRemocao Concluida!\n");

return;

}

aux=aux->prox;

}

printf("\nPessoa nao encontrada! Remocao Cancelada!\n");

}

}

void Remover\_Tudo(TContato \*\*a){

TContato \*aux;

if(\*a != NULL){

do{

aux = \*a;

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

free(aux);

}while(\*a != NULL);

}

}

2º)

Treal \*Copiar(Treal \*a){

Treal \*Nova=NULL,\*aux,\*aux2,\*novo;

if(a != NULL){

aux = a;

Nova = (Treal \*)malloc(sizeof(Treal));

\*Nova = \*a;

aux = a->prox;

aux2 = Nova;

while(aux != NULL){

novo = (Treal \*)malloc(sizeof(Treal));

\*novo = \*aux;

aux2->prox = novo;

novo->ant = aux2;

aux = aux ->prox;

aux2 = novo;

}

}

return Nova;

}

3º)

Treal \*CopiarInversa(Treal \*a){

Treal \*Nova=NULL,\*aux,\*aux2,\*novo;

if(a != NULL){

aux = a;

Nova = (Treal \*)malloc(sizeof(Treal));

\*Nova = \*a;

aux = a->prox;

aux2 = Nova;

Nova ->prox = NULL; // novo

while(aux != NULL){

novo = (Treal \*)malloc(sizeof(Treal));

\*novo = \*aux;

aux2->ant = novo; // mudou

novo->prox = aux2; // mudou

Nova = novo; // novo

aux = aux ->prox;

aux2 = novo;

}

Nova ->ant = NULL; // novo

}

return Nova;

}

4º)

Treal \*Inverter(Treal \*\*a){

Treal \*aux,\*aux2,\*aux3,noAux;

if(\*a==NULL){

printf("Lista vazia!");

}

else if((\*a)->prox==NULL){// ------ 1 unico No

return \*a;

}

else if((\*a)->prox->prox == NULL){// ----- 2 No's

aux = \*a;

aux2 = aux->prox;

aux->ant = aux2;

aux2->prox = aux;

aux->prox = NULL;

aux2->ant = NULL;

\*a = aux2;

return \*a;

}

else{//-------------- Mais de 2 no's (lista par ou impar)

aux = \*a;

aux3 = \*a;

while(aux3->prox != NULL)

aux3= aux3->prox;

//---------------------------// -----Revertendo Extremiadades

noAux = \*aux3;

aux3->ant = aux->ant;

aux3->prox= aux->prox;

aux->ant = noAux.ant;

aux->prox = noAux.prox;

\*a = aux3;//

aux=aux3->prox;

aux3=noAux.ant;

aux3->prox = aux->ant;

aux->ant = \*a;//

//---------------------------

while(aux->prox != aux3->prox){//----Revertendo meio

noAux = \*aux;

aux->ant = aux3->ant;

aux->prox = aux3->prox;

aux3 ->prox =noAux.prox;

aux3 ->ant = noAux.ant;

aux3->ant->prox = aux3;

aux3->prox->ant = aux3;

aux->prox->ant = aux;

aux->ant->prox = aux;

aux3 = aux->ant;

aux = noAux.prox;

if(aux->prox == aux3)

break;

}

if(aux->prox == aux3){// ----- Revertendo 2 ultimos do meio , se a Lista for Par

noAux = \*aux;

aux->ant = aux3;

aux->prox=aux3->prox;

aux3->prox = aux;

aux3->ant = noAux.ant;

aux3->ant->prox = aux3;

aux->prox->ant = aux;

}

return \*a;

}

}