#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "pilaD.h"

// No usa constantes para definir el tamaño de las cadenas

// Debería haber indicado que TElementoP estaría en “pilaD.h”

// no define TPila

// Todos los enteros podrían haber sido unsigned short

typedef struct{

char cod\_materia[6];

int cuatrimestre;

unsigned int creditos;

}TElementoP;

typedef struct nodoS{

char correlativas[6]; // Nombre poco representativo

struct nodoS \*sig;

}nodoS;

typedef nodoS \*TsubL;

typedef struct nodo{

char cod\_materia[6];

char nombre[26];

int anio;

int cuatrimestre;

char tipo;

unsigned int creditos;

TsubL subL;

struct nodo \* sig;

}nodo;

typedef nodo \*TLista;

typedef struct nodoD{

int anio;

TPila p;

struct nodoD \*ant,\*sig;

}nodoD;

typedef nodoD \*PnodoD;

typedef struct{

PnodoD pri,ult;

}TListaD;

void cargaLista(TLista \*);

void puntoiA(TLista ,TLista ,char []); //sobra un parámetro, mal modularizado

void puntoiB(TLista ,TListaD \*);

void puntoiC(TLista \*,TListaD ,TPila \*,char [],int); // pila parámetro? Un entero?, mal modularizado

int main(){

TLista l=NULL,busca;

TListaD ld;

TPila p=NULL;

char cod\_mat[6];

// Sería mejor inicializar en la función que genera a la lista

ld.pri = NULL;

ld.ult = NULL;

cargaLista(&l);

printf("codigo de materia\n");

scanf("%s",cod\_mat);

// La búsqueda la debía hacer la función puntoiA asumiendo que existe la asignatura

busca = l;

// ¿Busca mientras sean iguales?

while(busca != NULL && strcmp(busca->cod\_materia,cod\_mat) == 0)

busca = busca->sig;

if(busca){ //la consigna dice que siempre existe

puntoiA(l,busca,cod\_mat);

}

puntoiB(l,&ld);

printf("codigo de materia para eliminar\n");

scanf("%s",cod\_mat);

puntoiC(&l,ld,&p,cod\_mat,0);

return 0;

}

void cargaLista(TLista \*l){

//desarrollo

}

// Debía recibir solo el código de asignatura como parámetro, no también el nodo

void puntoiA(TLista l,TLista materia,char cod\_materia[]){

FILE \*arch;

char archivo[10];

TLista aux;

TsubL auxS = materia->subL;

float acumCreditos = 0.0;

int contCorrelativas = 0;

strcpy(archivo,cod\_materia);

strcat(archivo,".txt");

arch = fopen(archivo,"w"); // No controla la apertura del archivo

fprintf(arch,"%s %d %d\n",materia->nombre,materia->anio,materia->cuatrimestre);

if(auxS){

while(auxS!=NULL){

aux = l; //vuelvo a la cabecera para poder buscar ya que no estan ordenados

// Podría haber asumido que la asignatura existe

while(aux!=NULL && strcmp(aux->cod\_materia,auxS->correlativas) != 0)

aux=aux->sig;

if(aux != NULL && strcmp(aux->cod\_materia,auxS->correlativas) != 0){

fprintf(arch,"%s\n",aux->nombre); // Todas las materias debían estar en la misma línea

acumCreditos += aux->creditos;

contCorrelativas++;

}

auxS = auxS->sig;

}

fprintf(arch,"%2.2f\n",acumCreditos/contCorrelativas);

}else{

fprintf(arch,"NO TIENE CORRELATIVAS ANTERIORES\n");

} //{} de mas

fclose(arch);

}

void puntoiB(TLista l,TListaD \*ld){

TLista aux = l;

PnodoD nuevo,auxd;

TElementoP obligatoria;

// Mucho código repetido en la solución, podría haberse estructurado mejor

while(aux){

//insercion del año en lista doble

if(ld->pri == NULL || aux->anio < ld->pri->anio){

nuevo = (PnodoD)malloc(sizeof(nodoD));

nuevo->anio = aux->anio;

IniciaP(&nuevo->p);

// Ya hizo el análisis de inserción y aún no sabe si la materia era obligatoria o no

if(aux->tipo == 'O'){ // si es obligatoria inserto en la pila

strcpy(obligatoria.cod\_materia,aux->cod\_materia);

obligatoria.creditos = aux->creditos;

obligatoria.cuatrimestre = aux->cuatrimestre;

poneP(&nuevo->p,obligatoria);

}

nuevo->ant = NULL;

nuevo->sig = ld->pri;

if (ld->pri == NULL)

ld->ult = nuevo;

else

ld->pri->ant = nuevo;

ld->pri = nuevo;

}else{

if(ld->ult->anio < aux->anio){

nuevo = (PnodoD)malloc(sizeof(nodoD));

nuevo->anio = aux->anio;

IniciaP(&nuevo->p);

// Ya hizo el análisis de inserción y aún no sabe si la materia era obligatoria o no

if(aux->tipo == 'O'){

strcpy(obligatoria.cod\_materia,aux->cod\_materia);

obligatoria.creditos = aux->creditos;

obligatoria.cuatrimestre = aux->cuatrimestre;

poneP(&nuevo->p,obligatoria);

}

nuevo->sig = NULL;

nuevo->ant = ld->ult;

ld->ult->sig = nuevo;

ld->ult = nuevo;

}else{

auxd = ld->pri;

// auxd no puede ser nulo, sino hubiera insertado por el final

while(auxd != NULL && aux->anio > auxd->anio)

auxd = auxd->sig;

// Ya hizo el análisis de inserción y aún no sabe si la materia era obligatoria o no

if(auxd != NULL && auxd->anio == aux->anio && aux->tipo == 'O'){

strcpy(obligatoria.cod\_materia,aux->cod\_materia);

obligatoria.creditos = aux->creditos;

obligatoria.cuatrimestre = aux->cuatrimestre;

poneP(&auxd->p,obligatoria);

}else{

if(auxd != NULL && aux->anio < auxd->anio){

nuevo = (PnodoD)malloc(sizeof(nodoD));

nuevo->anio = aux->anio;

IniciaP(&nuevo->p);

// Ya hizo el análisis de inserción y aún no sabe si la materia era obligatoria o no

if(aux->tipo == 'O'){

strcpy(obligatoria.cod\_materia,aux->cod\_materia);

obligatoria.creditos = aux->creditos;

obligatoria.cuatrimestre = aux->cuatrimestre;

poneP(&nuevo->p,obligatoria);

}

nuevo->ant = auxd->ant;

nuevo->sig = auxd;

auxd->ant->sig = nuevo;

auxd->ant = nuevo;

}

}

}

}

aux = aux->sig;

}

}

**// NO mezclar el recorrido de una estructura de forma recursiva con el recorrido de otras estructuras QUE INEFICIENTE!! PARAMETROS INNECESARIOS PARA UN RECURSIVIDAD! IF INNECESARIO PARA VER SI ESTA O NO EN EL RECURSIVO**

void puntoiC(TLista \*l,TListaD ld,TPila \*p,char cod\_mat[],int recursivo){

TLista ant = NULL,aux = \*l;

int anio;

TElementoP x;

TsubL auxs,ants,elimsl;

PnodoD auxd;

if(recursivo == 0){ // if (!recursivo)

while(aux != NULL && strcmp(aux->cod\_materia,cod\_mat) != 0){

ant = aux;

aux = aux->sig;

}

if(aux != NULL){

anio = aux->anio;

while(aux->subL != NULL){ //libero todo lo que hay dentro de la sublista de la materia

elimsl = aux->subL;

aux->subL = aux->subL->sig;

free(elimsl);

}

if(ant) //elimino de la lista principal

ant->sig = aux->sig;

else

\*l = (\*l)->sig;

free(aux);

aux = \*l;

//comienzo a buscar en las sublistas

while(aux){

auxs = aux->subL;

ants = NULL;

while(auxs != NULL && strcmp(auxs->correlativas,cod\_mat) != 0){

ants = auxs;

auxs = auxs->sig;

}

if(auxs){

if(ants)

ants->sig = auxs->sig;

else

aux->subL = aux->subL->sig;

free(auxs);

}

aux = aux->sig;

}

// Busca en la lista doble aunque la asignatura no sea obligatoria

if(ld.pri){ //elimino de la lista doble

auxd = ld.pri;

while(auxd != NULL && auxd->anio != anio) //cree que hay años que no van a estar?? Según lo que hizo en a) los años los inserta siempre!!

auxd = auxd->sig;

if(auxd != NULL && !VaciaP(auxd->p))

puntoiC(l,ld,&auxd->p,cod\_mat,1); //llamado recursivo

}

}else{

printf("NO EXISTE LA MATERIA\n");

}

}else{

if(!VaciaP(\*p)){ //eliminacion en la pila

sacaP(p,&x);

// Recorre recursivamente hasta el final de la pila aunque ya hubiese encontrado la asignatura

puntoiC(l,ld,p,cod\_mat,1);

if(strcmp(x.cod\_materia,cod\_mat) != 0)

poneP(p,x);

}

}

}