#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX 31

#define LARGO\_COD 8

#define LARGO\_NOM 21

#define DIAS 30

#define MESES 12

typedef struct {

int dia; //shortint

float tMin, tMax;

} tElementop;

typedef struct pila{

tElementop datos[MAX];

int tope;

} tPila;

// No se pedían los prototipos de los op de pila

void iniciaP(tPila \*p);

void poneP(tPila \*p, tElementop x);

void sacaP(tPila \*p, tElementop \*x);

tElementop consultaP(tPila p);

int vaciaP(tPila p); //Implementaciones en pila.c, los tipos de telemento y tpila y estos prototipos deberian estar en pila.h

//Deberia ir #include "pila.h"

typedef struct nodo {

char codPos[LARGO\_COD];

char nombre[LARGO\_NOM];

tPila mediciones;

struct nodo\* sig;

} nodo;

typedef nodo\* TLista;

typedef struct nodoD {

char nombre[LARGO\_NOM];

int dia;

float tMin, tMax, rgTemp;

struct nodoD \*sig, \*ant;

} nodoD;

typedef nodoD\* PNodo;

typedef struct {

PNodo pri, ult;

} TlistaDoble;

// Podrían haber sido prototipos las funciones incompletas siguientes

void cargaLista(TLista\* l, char\* nomArch) {

//Se carga la lista desde archivo

}

void cargaMt(TLista lista, char mt[DIAS][MESES][LARGO\_COD]) {

//Se carga mt en base a la lista de ciudades ya cargada

}

void ingresaCp(char\* codPos) {

//Se ingresa cp por algun medio, consola si no hay un parametro para que fuera de archivo

}

void ingresaTemp(float\* temp) {

//Se ingresa temp por algun medio, consola si no hay un parametro para que fuera de archivo

}

void insertaNodoDobleOrdenado(char\* nombre, int dia, float tMin, float tMax, TListaDoble\* l) {

PNodo act, aux;

float rgTemp = tMax - tMin;

aux = (PNodo) malloc(sizeof(nodoD));

strcpy(aux->nombre, nombre);

aux->dia = dia;

aux->tMin = tMin;

aux->tMax = tMax;

aux->rgTemp = rgTemp;

if (l->pri == NULL || rgTemp < l->pri->rgTemp) {

aux->ant = NULL;

aux->sig = l->pri;

if (l->pri == NULL)

l->ult = aux;

else

l->pri->ant = aux;

l->pri = aux;

}

else if (rgTemp > l->ult->rgTemp) {

aux->ant = l->ult;

aux->sig = NULL;

l->ult->sig = aux;

l->ult = aux;

}

else {

act = l->pri;

while (rgTemp > act->rgTemp)

act = act->sig;

aux->sig = act;

aux->ant = act->ant;

act->ant->sig = aux;

act->ant = aux;

}

}

void generaListaDoble(TLista lista, TListaDoble\* listaDoble) {

tElementop medicion;

tPila pilaAux;

int condicion;

float sumaMin;

int cant;

TLista ciudadesMayorTemp[10]; //Asumo como maximo hay 10 ciudades con la misma temp que sea la mayor de todas, se puede modificar

int cantMayorTemp;

float mayorTemp;

float mayorTempActual;

int i;

sumaMin = 0;

cant = 0;

cantMayorTemp = 0;

mayorTemp = -999;

iniciaP(&pilaAux);

while (lista != NULL) {

mayorTempActual = -9999;

while (!vaciaP(lista->mediciones)) {

sacaP(&lista->mediciones, &medicion);

cant++;

sumaMin += medicion.tMin;

if (medicion.tMax > mayorTempActual)

mayorTempActual = medicion.tMax;

// Innecesario guardar el resultado en una variable, con poner la expresión como condición bastaba

condicion = medicion.tMax - medicion.tMin <= medicion.tMin;

if (condicion) {

insertaNodoDobleOrdenado(lista->nombre, medicion.dia, medicion.tMin, medicion.tMax, listaDoble);

}

else {

poneP(&pilaAux, medicion);

}

}

while (!vaciaP(pilaAux)) {

sacaP(&pilaAux, &medicion);

poneP(&lista->mediciones, medicion);

}

if (mayorTempActual >= mayorTemp) {

// Si encuentra una nueva temperatura mayor tiene que descartar todas las ciudades previas volviendo a 0 el indice

mayorTemp = mayorTempActual;

ciudadesMayorTemp[cantMayorTemp] = lista;

cantMayorTemp++;

}

lista = lista->sig;

}

if (cant > 0) {

printf("Temp. Min promedio: %g\n", sumaMin / cant);

printf("Ciudad/es con la mayor temperatura, igual a %g grados:\n", mayorTemp);

}

else {

printf("No hay mediciones\n");

}

// Sería más adecuado haberlo incluído en la rama verdadera del if anterior

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

puts(ciudadesMayorTemp[i]->nombre);

}

int cuentaVecesMaximaRec(char mt[DIAS][MESES][LARGO\_COD], char\* codPos, int iCorte, int jCorte, int i, int j, int n) {

if (i == iCorte && j == jCorte) { //Los cortes deberian ser >= 0 y <= que los indices de los dias/meses maximos para quien llame a la funcion

return 0;

}

else if (i == -1)

return cuentaVecesMaximaRec(mt, codPos, iCorte, jCorte, n - 1, j - 1, n);

else {

if (strcmp(codPos, mt[i][j]) == 0)

return cuentaVecesMaximaRec(mt, codPos, iCorte, jCorte, i - 1, j, n) + 1;

else

return cuentaVecesMaximaRec(mt, codPos, iCorte, jCorte, i - 1, j, n);

}

}

int cuentaVecesMaximaEnOtonio(char mt[DIAS][MESES][LARGO\_COD], char\* codPos) {

return cuentaVecesMaximaRec(mt, codPos, 20, 2, 20, 5, DIAS); //Empieza en 21/6, corta en 21/3

}

void eliminaCiudad(char\* codPos, float temp, TLista\* lista, int\* eliminada) {

// Cuidado con definir el tamaño de un arreglo con una expresión, solo es válido en el estándar de C si la expresión puede calcularse en tiempo de compilación (lo cual se cumple aquí dado que LARGO\_NOM es una macro)

char nomArch[LARGO\_NOM + 4]; //4 espacios para ".dat"

tElementop medicion;

FILE\* arch;

TLista ant, act = \*lista;

\*eliminada = 0;

while (act != NULL && strcmp(codPos, act->codPos) != 0) {

ant = act;

act = act->sig;

}

if (act != NULL) {

strcpy(nomArch, act->nombre);

strcat(nomArch, ".dat");

arch = fopen(nomArch, "wb");

if (arch != NULL) { //Asumo que si no se puede crear el archivo, no deberia borrarse la ciudad

while (!vaciaP(act->mediciones)) {

sacaP(&act->mediciones, &medicion);

if (medicion.tMin < temp) {

fwrite(&medicion, sizeof(tElementop), 1, arch);

}

}

if (act == \*lista)

\*lista = act->sig;

else

ant->sig = act->sig;

free(act);

fclose(arch);

\*eliminada = 1;

}

}

}

int main()

{

TLista listaCiudades = NULL;

TListaDoble listaDobleMediciones;

char mt[DIAS][MESES][LARGO\_COD];

char codPos[LARGO\_COD];

float temp;

int eliminada;

// Podrían haberse inicializado mejor en la función correspondiente

listaDobleMediciones.pri = listaDobleMediciones.ult = NULL;

cargaLista(&listaCiudades, "datos.txt");

cargaMt(listaCiudades, mt);

generaListaDoble(listaCiudades, &listaDobleMediciones); //Aunque una ciudad de la lista simple quede sin mediciones, no se indica que deberia eliminarse

ingresaCp(codPos);

printf("Para el codigo ingresado, en otonio esa ciudad tuvo la temp max %d veces\n", cuentaVecesMaximaEnOtonio(mt, codPos));

ingresaCp(codPos);

ingresaTemp(&temp);

eliminaCiudad(codPos, temp, &listaCiudades, &eliminada);

if (eliminada)

printf("Se elimino la ciudad con codigo postal ingresado, con exito\n");

else

printf("No se pudo eliminar la ciudad con codigo postal ingresado\n"); //No se distingue falla en la apertura del archivo o codigo postal incorrecto

return 0;

}