Ejercicios integradores

Los problemas planteados en esta práctica reúnen todos los conceptos dados en la asignatura Informática. Por cada uno de ellos se debe realizar el análisis del problema, escribir el algoritmo que permita resolverlo utilizando modularización, estructuras de datos y archivos, si fuera necesario, y su posterior codificación en Lenguaje C, ejecución, prueba y verificación del correcto funcionamiento del programa.

1- El archivo **ALT_RIO.DAT** contiene las mediciones históricas de altura de un río medidos a lo largo de 500 kilómetros de recorrido, un ejemplo de su contenido es:

Kilómetro	Altura(metros)	Año
150	6	1978
57	2	1975
2	7	2000
150	4	1983
2	8	2004
487	6	2012
…	…	…

Se observa que las mediciones pueden haber sido tomadas varias veces en el mismo lugar, en diferentes años (por lo tanto no se sabe cuántas líneas tiene este archivo).

Se pide escribir un algoritmo que a partir de la información disponible en el archivo calcule e informe al usuario por pantalla los siguientes ítems:

a) Cantidad de mediciones existentes para cada kilómetro, mediante la llamada al subalgoritmo CANT_VECES.

b) Posición (kilómetro) del cual se disponen más mediciones, mediante el subalgoritmo MAS_VECES.

c) Solicitar al usuario que ingrese un determinado kilómetro para consultar el promedio de todas las alturas existentes para ese kilómetro, mediante el subalgoritmo CONSULTA.

d) Mostrar las mediciones del archivo ordenadas en forma ascendente de acuerdo a las alturas del río, mediante el subalgoritmo ORDENA.

2- Un negocio de artículos de computación tiene registradas las ventas por tarjeta de crédito del mes en un archivo de nombre **VTARJ.DAT**. Por experiencia se sabe que no hay más de 500 ventas en un mes. En este archivo, cada uno de sus registros (renglones) está compuesto por: el número de tarjeta (entero) y el importe (real). Sus datos no están ordenados de ninguna manera.

Se necesita escribir un algoritmo que lea los datos del archivo y mediante un subalgoritmo

PROME muestre por pantalla el promedio de ventas del mes. Además, mediante otro subalgoritmo **TOP10** se deben encontrar las 10 mayores ventas para guardarlas en un archivo llamado **T10.DAT**.

En una segunda etapa, se desean procesar además, los datos del archivo **VTARJ2.DAT** provenientes de una sucursal. Se requiere confeccionar un archivo conteniendo el listado unificado de las ventas (nro. de tarjeta - importe) ordenado en forma decreciente según los montos.

3- Completar el siguiente algoritmo con las consignas que se presentan a continuación del mismo.

```
{ ...aquí van las declaraciones de subalgoritmos... }

Tipo estructura mat
     caracter(8) legajo
     real       notas[3]
Fin tipo

ALGORITMO condiciones
{ ...aquí van las declaraciones de las variables... }

//Inicio
INGRESADATOS (Datos, Tot)
Escribir( "Se ingresaron", Tot ,"alumnos" )
Op ← 'L'
Repetir mientras (Op <> 'F')
     Escribir( "L : Buscar por Legajo")
     Escribir( "F : Fin")
     Escribir( "Elija una opción")
     Leer( Op )
     Si ( Op <> 'F' ) entonces
          Escribir ("Ingrese el legajo")
          Leer (Legajo)
          Pos ← BUSCA (Datos, Legajo, Tot)
          Si (Pos = 0) entonces
               Escribir("No existe un alumno con ese dato")
          sino
               Escribir("Condición Final: ",CONDI (Datos,
          Pos))
          fin si
     fin si
fin mientras
Fin
```

Declarar y desarrollar los subalgoritmos (función o subrutina, según corresponda) con los

parámetros correspondientes y respetando los llamados:

INGRESADATOS(): permite cargar el legajo del alumno y hasta 3 notas de cada uno de ellos. Esta información se encuentra, sin ningún orden, en el archivo **NOTAS.DAT**, donde cada registro está compuesto por los siguientes campos: legajo del alumno (cadena de 8 caracteres), materia (cadena de 12 caracteres sin espacios), nota obtenida en la materia (real). No necesariamente se dispone de la nota de todos los alumnos y todas las materias.

BUSCA(): devuelve la posición del alumno con ese legajo, o 0 si no existen datos del alumno. Implementar mediante búsqueda dicotómica.

CONDI(): devuelve 'P', 'I' o 'L' , según las notas del alumno ubicado en la posición Pos.

Observaciones: Tener en cuenta que no se conoce la cantidad total de alumnos, pero se sabe que no supera los 150.

Condiciones:

'P' promovido (el promedio es superior a 7 y ninguna nota es inferior a 5)

'I' condición intermedia (no promovido. Además el promedio es superior a 5 y ninguna nota es inferior a 4)

'L' libre (no es 'P' ni 'I')

4- Una librería cuenta con un archivo, de nombre LIBROS.DAT, donde almacena la información de los libros con los cuales trabaja. Se sabe que en dicho archivo no figuran libros repetidos y que a lo sumo pueden existir 500 libros registrados.
De cada libro se tiene:

Código de identificación (entero).
Nombre (cadena de a lo sumo 30 caracteres).
Autor (cadena de a lo sumo 30 caracteres).
Calificación (entero entre 1 y 10).
Existencia (entero).
Precio (real).

Se solicita realizar las siguientes acciones:

a) En una subrutina mostrar el contenido del archivo LIBROS.DAT.

b) Escribir un subalgoritmo que recibiendo un código de identificación desde el algoritmo principal lo busque en los libros cargados en el archivo y devuelva al algoritmo principal su título, existencia y precio. Estos datos deberán mostrarse luego, por pantalla. En caso de no encontrarlo, informar dicha situación con un mensaje.

c) Ingresando una calificación mínima y un valor de límite superior de precio, mostrar todas las obras, con sus datos, que cumplan con los requerimientos ingresados.

5- A principio del mes de junio, el director comercial de una concesionaria de autos de la ciudad nos encargó que lo ayudemos a obtener la información necesaria para evaluar el rendimiento de los vendedores durante los pasados 3 meses. Para esto nos facilitó el archivo donde se encuentran todas las ventas de este período (ventas3-4-5.dat). Los registros del archivo responden al siguiente diseño:

- Identificación del vendedor (***ID***): entero consecutivo desde el 1. Hay 38 vendedores.
- Modelo de auto (***MOD***): entero entre 1 y 8.
- Día, mes y año de la venta (***DD***, ***MM***, ***AA***): 3 enteros.
- Descuento realizado en la venta (***DTO***): valor con decimales (entre 0 (0%) y 0,3 (30%)).

En cada venta, se vende un único auto.

Realizar *un bosquejo del archivo de ventas escribiendo, al menos, 5 ejemplos de registros. Entre paréntesis se encuentran los nombres de campos/variables sugeridos.*

Además, contamos con un listado de precios, en dólares, de los 8 modelos de autos y el director nos informa, que los precios han permanecido sin variaciones desde principio de año.

Con toda esta información nos solicita la obtención de determinados indicadores.

Se pide confeccionar un algoritmo en pseudocódigo que:

a) Realice la carga de precios, a partir de teclado, en una estructura de datos de nombre PRECIOS. Se ingresarán los precios de los 8 modelos de autos, aunque no necesariamente ordenados por número de modelo. Validar el ingreso del modelo adecuadamente.

Realizar *un bosquejo de esta estructura de datos.*

b) Invoque a una subrutina de nombre **CARGA** que deje disponible en el algoritmo principal, a través de una estructura de datos apropiada, solamente la información procesada y necesaria del archivo *ventas3-4-5.dat* para ser utilizada en los procesos posteriores.

Realizar *un bosquejo de la estructura de datos que devuelve la subrutina* **CARGA***.*

c) Muestre (desde el algoritmo principal) el número de vendedor y el monto total correspondiente, del vendedor que obtuvo el mayor monto total de ventas. La determinación de este vendedor se deberá hacer a través de una función **MAYOR**. Se asume que no habrá duplicados.

d) Invoque al subalgoritmo **PORCENTAJES**, que listará ordenado por número de vendedor en forma decreciente, el porcentaje que representa la cantidad total de unidades vendidas por cada uno de ellos, sobre la cantidad total de unidades vendidas por todos los vendedores de la concesionaria. Sólo se deberán listar los vendedores cuyo porcentaje supere a un porcentaje que el usuario indicará por teclado, en el algoritmo principal.

6- Se cuenta con las mediciones de radiación solar (irradiancia) de todos los días del año 2017, para la ciudad de Rosario. Se desea analizar estos datos para realizar un estudio del rendimiento que puede obtenerse de una instalación eléctrica fotovoltaica.
Se cuenta con un archivo de nombre **DATOS.DAT** compuesto por 4 campos:

Mes: número entero,
Día: número entero,
Irradiancia: número real,
Minutos solares diarios: número entero.

Se pide diseñar un algoritmo que deberá procesar y cargar la información necesaria de este archivo en una/s estructura/s de datos adecuada/s para la resolución de los siguientes ítems, a través del llamado a una subrutina de nombre **CARGA**, que debe devolver dicha/s estructura/s de datos al programa principal.
Luego, se debe mostrar por pantalla el siguiente menú:

****IRRADIANCIA ROSARIO****
1- Irradiancia promedio mensual
2- Irradiancia mes específico
3- Mes mayor cant. horas solares
4- Finalizar

Cada uno de los ítems del menú deben permitir:
1- Mostrar por pantalla un listado que indique la irradiancia promedio diaria de cada mes.
2- Mostrar la irradiancia promedio diaria de un mes específico que el algoritmo principal debe pedir al usuario.
3- Mostrar el mes con mayor cantidad de horas de luz solar (invocar una función de nombre **MAYOR**)
4- Dar por finalizado el programa.

Los ítems del menú deben resolverse con los datos de las estructuras pasadas por **CARGA** (no debe volver a leerse el archivo).

7- El servicio meteorológico nacional dispone de un archivo de nombre **TEMPS.DAT** que contiene información sobre las temperaturas máximas registradas en cada uno de los 31 días del mes de mayo en las 50 ciudades principales del país.

Las ciudades están codificadas mediante un número entero del 1 al 50.
El archivo contiene los siguientes datos: nro. de día (entero), código de la ciudad (entero) y temperatura máxima (real). Los datos en el archivo no presentan orden alguno.

Se solicita hacer el algoritmo principal y los subalgoritmos correspondientes en pseudocódigo o Lenguaje C que resuelvan lo siguiente a partir del archivo **TEMPS.DAT**.

El algoritmo/programa principal deberá:

1- Procesar la información del archivo a través de la invocación a una subrutina de nombre **LECTURA**, que deberá devolver al principal la estructura de datos apropiada que contenga las temperaturas máximas registradas en cada uno de los 31 días del mes de mayo para cada una de las 50 ciudades.

2- Solicitar por teclado un código de ciudad y un valor de temperatura **T**, y mediante la llamada a una función de nombre **CUANTOS** muestre la cantidad de días con temperaturas mayores a T que registra esa ciudad. Los parámetros de esta función deberán ser el código de la ciudad, la temperatura T y la estructura obtenida en **LECTURA**.

Este punto debe repetirse hasta que el usuario ingrese un número 0 como código de ciudad.

3- Deberá listar qué día se produjo la mayor temperatura en cada una de las 50 ciudades. Considerar que existe un único máximo por ciudad.

Se deberá mostrar: código de ciudad, temperatura mayor y dia en el cual fue registrada. Este listado deberá estar ordenado según la temperatura en forma descendente.

Además, se debe grabar este listado en un archivo de nombre **MAXMAYO18.DAT.**

8- Dibuje todas las estructuras necesarias y desarrolle el algoritmo en pseudocódigo correspondiente para resolver el siguiente enunciado.

Se solicita hacer un algoritmo que permita obtener y listar: todos los legajos y su correspondiente promedio, ordenado por promedio en orden decreciente.

Para ello se dispone del archivo **alumnos.dat** cuyos registros lógico están compuesto de 3 campos: legajo (carácter 8), apellido (carácter 25), nombres (carácter 25).

A la vez, cuenta con otro archivo **examenes.dat**, cada registro lógico del archivo está compuesto por 6 campos: legajo (carácter 8), código de asignatura (carácter 5), día (entero de 1 a 31), mes (entero de 1 a 12), año (entero) del examen y nota (real). No se sabe

cuántos registros tiene el archivo **examenes.dat**

Deberá desarrollar las siguientes subrutinas que se invocarán desde el algoritmo principal:

CARGA subrutina que permite la carga de los datos del archivo **alumnos.dat** en adecuada/s estructura/s datos que se deberá/n devolver al algoritmo principal. En el archivo **alumnos.dat** hay exactamente 80 alumnos registrados.

PROMEDIOS: subrutina que recibe la/s estructura/s de datos devueltas por CARGA y permite calcular de cada alumno, su correspondiente promedio. La/s estructura/s se actualizará/n y deberá/n ser devuelta/s al algoritmo principal.

Para calcular el promedio, se tienen como datos los exámenes de cada alumno que se encuentran en el archivo **examenes.dat**. Cada vez que se lee un registro del archivo se deberá verificar que el *legajo* leído sea válido, para ello se invocará a la función **BUSCAR** que recibiendo como parámetros la estructura generada en la subrutina CARGA y el *legajo*, esta función mediante búsqueda secuencial deberá encontrar el *legajo* buscado y devolver un valor a su elección, este valor indicará si lo encontró o no.

Si el *legajo* se encontró (es válido), entonces se contabilizarán y acumularán sus notas, en caso contrario, si el *legajo* no se encontró, no se realizará ninguna acción.

Una vez procesados todos los datos del archivo **examenes.dat** se obtendrá el promedio de cada *legajo*. Tener en cuenta que hay alumnos que no rindieron ninguna asignatura y no se encuentran en el archivo **examenes.dat**.

ORDENA subrutina que recibiendo la/s estructura/s de datos devueltas por PROMEDIOS, la/s ordena por promedio en orden decreciente. Luego muestra por pantalla el legajo y su correspondiente promedio.

9- En una planta agroquímica se llevan a cabo procesos que deben ser continuamente monitoreados, de modo de garantizar que las condiciones sean óptimas para que cada uno de ellos pueda llevarse a cabo con normalidad.

En particular en esta planta química se realizan mediciones de cuatro tipos:

- Temperatura: medida en grados centígrados [ºC]
- Presión: medida en bares [Bar]
- Nivel: medido en metros[m]
- Caudal: medido en litros por hora [l/h]

Las mediciones de un día se guardan en el archivo *mediciones.txt*, y para cada una de ellas se guardan los siguientes datos:

ID del sensor - Tipo de sensor - Cota inferior - Cota superior - Medición

Donde:

- *ID del sensor*: es un número que identifica a dicho sensor en la planta.
- *Tipo de sensor*: temperatura (T), presión (P), nivel (N) ó caudal(C).
- *Cota inferior*: valor por debajo del cual no debería encontrarse la variable medida.
- *Cota superior*: valor por encima del cual no debería encontrarse la variable medida.
- *Medición*: el valor obtenido de la medición en su correspondiente unidad.

A modo de ejemplo, se muestran a continuación unas posibles líneas de dicho archivo.

205	T	80	90	87
314	P	100	124	127

De donde, la primera línea debe interpretarse como: el sensor 205, que es de temperatura, cuya cota inferior es de 80ºC y la superior de 90ºC, midió 87 ºC. Del mismo modo, la segunda línea debe interpretarse como sigue: el sensor 314, que es de presión, cuya cota inferior es de 100 bares y la superior de 124 bares, midió 127 bares.

A partir de esta información, se desea implementar un algoritmo que llame a los siguientes subalgoritmos uno a continuación del otro.

1) Subalgoritmo FILTRO:
Recibiendo un tipo de sensor del algoritmo principal (T, P, N o C), mostrar sólo las mediciones de ese tipo que figuran en el archivo *mediciones.txt*. Luego devolver al algoritmo principal un arreglo de 4 elementos, donde cada elemento del arreglo contenga la cantidad de mediciones de cada uno de los cuatro tipos. Estos cuatro valores deberán ser mostrados en el algoritmo principal.

2) Subalgoritmo GEN_AL:
Podemos observar que en el caso de la primera línea del archivo *mediciones.txt*, la medición se encuentra dentro de las cotas definidas, mientras que en el segundo caso esto no sucede. Cuando una variable se encuentra fuera de su rango admisible, esto debe ser informado a través de una *alarma*, que consiste en enviar la medición al área de control de planta escribiéndola en el archivo *alarmas.txt* tal y como figura en el archivo de origen, es decir, el renglón completo.

Este subalgoritmo debe verificar las mediciones para determinar si los sensores están fuera de sus límites y generar un señal de *alarma* si corresponde escribiendo cada una en el archivo *alarmas.txt*.

3) Subalgoritmo ALARMAS:
Debe contar la cantidad de alarmas de sensores de presión que han sido guardadas en el subalgoritmo GEN_AL y devolver ese valor para ser informado en el algoritmo principal.

10- Con el objetivo de analizar los resultados de una competencia de matemática en la que participaron 50 estudiantes (todos los estudiantes están cursando el último año del profesorado de matemática), se almacenaron los resultados de la competencia en un archivo de nombre **RESULTADOS.DAT**. Cada estudiante tenía que resolver 5 ejercicios.
Los datos del archivo son:

Número de estudiante (un número entero de 1 a 50 que permite identificar a cada competidor)

Número de ejercicio (un número entero de 1 a 5 que permite identificar a cada ejercicio)

Nota (un número real)

Realizar un algoritmo principal y los subalgoritmos **CREAR, MAYOR** y **PORCENTAJE.**

CREAR: debe procesar los datos del archivo **RESULTADOS.DAT** a fin de guardar en una estructura de datos adecuada de nombre RES, de cada estudiante, la calificación obtenida (es decir, la suma de cada una de las notas obtenidas en cada ejercicio). La/s estructura/s deberán ser devueltas al programa principal.

MAYOR que recibiendo la/s estructura/s devueltas por el sugalgorimo CREAR, muestre en pantalla el o los estudiantes que obtuvieron la mayor clasificación. En pantalla deberá mostrar: “La máxima clasificación obtenida en la competencia fue ***valor mayor*** y la obtuvo el estudiante ***nro. estudiante*** “, o

“La máxima clasificación obtenida en la competencia fue ***valor mayor*** y la obtuvieron los estudiantes: “

“Número de estudiante” ***nro. estudiante***

PORCENTAJE que recibiendo la/s estructura/s devueltas por el sugalgorimo CREAR, calcule y devuelva al programa principal el porcentaje de estudiantes desaprobados (desaprobado es una nota <6).

Observaciones:
- No es necesario realizar un menú de opciones.
- Al menos un subalgoritmo debe ser una función.

11- Con el objetivo de analizar la frecuencia de uso de las teclas de un teclado, durante una semana un software mide el uso de las teclas del mismo. El software mide 27 teclas. Las teclas que serán sometidas a control, se ingresan por teclado y se almacenan en memoria en una estructura adecuada.

Día a día, por cada tecla, se almacena la información en un archivo de nombre **MEDICIONES.DAT**.

Los datos del archivo son:

Día de la semana (un número entero de 1 a 7)

Letra medida (un carácter que es el que se midió)

Cantidad de veces que se presionó en ese día (número entero)

Realizar un algoritmo principal y los subalgoritmos CREAR, CARGAR y MAYOR.

CREAR: debe solicitar el ingreso por teclado de cada una de las 27 letras a controlar y guardarlas en una estructura de datos apropiada de nombre LET que deberá ser devuelta al programa principal.

CARGAR: debe procesar los datos del archivo *MEDICIONES.DAT* a fin de guardar en una estructura de datos adecuada de nombre USO, la letra y el uso semanal de la tecla (la cantidad total de veces que esa letra se usó en la semana). La/s estructura/s deberán ser devueltas al programa principal.

MAYOR que recibiendo la/s estructura/s devueltas por el sugalgorimo CARGAR, devuelva la posición en donde se encuentra la tecla de mayor uso (se supone que hay una sola letra con mayor uso), y muestre en el algoritmo principal el siguiente mensaje: "La tecla más usada fue: ***letra***. Se presionó ***uso*** veces"

Observaciones:
- No es necesario realizar un menú de opciones.
- Al menos un subalgoritmo debe ser una función.

Ejercicios Resueltos

1- Una librería cuenta con un archivo, de nombre **LIBROS.DAT**, donde almacena la información de los libros con los cuales trabaja. Se sabe que en dicho archivo no figuran libros repetidos y que a lo sumo pueden existir 500 libros registrados.
De cada libro se tiene:

- Código de identificación (entero).
- Nombre (cadena de a lo sumo 30 caracteres).
- Autor (cadena de a lo sumo 30 caracteres).
- Calificación (entero entre 1 y 10).
- Existencia (entero).
- Precio (real).

Se solicita realizar las siguientes acciones:
a) En una subrutina mostrar el contenido del archivo **LIBROS.DAT**.
b) Escribir un subalgoritmo que recibiendo un código de identificación desde el algoritmo principal lo busque en los libros cargados en el archivo y devuelva al algoritmo principal su título, existencia y precio. Estos datos deberán mostrarse luego, por pantalla. En caso de no encontrarlo, informar dicha situación con un mensaje.
c) Ingresando una calificación mínima y un valor de límite superior de precio, mostrar todas las obras, con sus datos, que cumplan con los requerimientos ingresados.

Resolución:
Para leer el contenido del archivo y luego trabajar con el mismo, podemos proceder de varias maneras. Por un lado, es posible trabajar siempre leyendo el archivo cada vez que sea necesario, utilizar su información (en cuyo caso se lo leerá usando variables simples o un registro). Por otro lado, es posible trabajar cargando el contenido de dicho archivo a memoria (es decir guardando todo su contenido en arreglos de variables simple o en un arreglo de registros).

A continuación, se muestra la resolución del problema utilizando la primera lógica descripta y utilizando variables simples:

```
Algoritmo problema_libros
Variables:
        entero: ID, c_min, ex, band
        real: p_max, prec
        caracter: tit[30]
Inicio
        // ítem a
        MOSTRAR()
        // ítem b
        band ← 0
        Escribir("Ingrese un código de identificación a buscar")
        Leer(ID)
        BUSCAR(ID, band, ex, prec, tit)
        Si(band <> 0) entonces
      Escribir(nom, ex, prec)
```

```
        sino
          Escribir("Libro no encontrado")
        fin si
        // ítem c
        Escribir("Ingrese una calificación mínima a buscar")
        Leer(c_min)
        Escribir("Ingrese un precio máximo")
        Leer(p_max)
        FILTRAR(c_min, p_max)
Fin

Subrutina MOSTRAR()
Variables
   entero: id, calif, ex
   caracter: nom[30], aut[30]
   real: precio
Inicio
   abrir(f1, "LIBROS.DAT", lectura)
   Leer(f1, id, nom, aut, calif, ex, precio)
   Repetir mientras(no(eof(f1)))
         Escribir(id, nom, aut, calif, ex, precio)
         Leer(f1, id, nom, aut, calif, ex, precio)
   fin mientras
   cerrar(f1)
fin subrutina

Subrutina BUSCAR(E: entero: ID_libro; S: entero: band, ex, real: precio, caracter(30): nom)
Variables:
        entero: id, calif
        caracter:  aut[30]
Inicio
        abrir(f1, "LIBROS.DAT", lectura)
        Leer(f1, id, nom, aut, calif, ex, precio)
        Repetir mientras(no(eof(f1)) y ID_libro <> id)
          Leer(f1, id, nom, aut, calif, ex, precio)
        fin mientras
        cerrar(f1)
        Si(ID_libro = id) entonces
          band ← 1
        fin si
fin subrutina

Subrutina FILTRAR(E: entero: c_min, real:p_max)
Variables
        entero: id, calif, ex, band
        caracter: nom[30], aut[30]
        real: precio
```

```
Inicio
        band ← 0
        abrir(f1, "LIBROS.DAT", lectura)
        Repetir mientras(no(eof(f1)))
          Leer(f1, id, nom, aut, calif, ex, precio)
          Si(calif >= c_min y precio <= p_max) entonces
       Escribir(id, nom, aut, calif, ex, precio)
       Si(band = 0) entonces
         band ← 1
       fin si
          fin si
        fin mientras
        cerrar(f1)
        Si(band = 0) entonces
          Escribir("Ningún libro cumple con el criterio ingresado")
        fin si
fin subrutina
```

Su codificación en lenguaje C es:

```
#include <stdio.h>

void MOSTRAR()
{
   int id, calif, ex;
   char nom[30], aut[30];
   float precio;
   FILE *f1;

   f1 = fopen("LIBROS.DAT","r");
   fscanf(f1,"%d%s%s%d%d%f", &id, nom, aut, &calif, &ex, &precio);
   while(!feof(f1))
   {
         printf("%d %s %s %d %d %g\n", id, nom, aut, calif, ex, precio);
         fscanf(f1,"%d%s%s%d%d%f", &id, nom, aut, &calif, &ex, &precio);
   }
   fclose(f1);
}

void BUSCAR(int ID_libro, int *band, int *ex, float *precio, char nom[30])
{
   int id, calif;
   char aut[30];
   FILE *f1;

   f1 = fopen("LIBROS.DAT","r");
   fscanf(f1,"%d%s%s%d%d%f", &id, nom, aut, &calif, ex, precio);
```

```
   while(!feof(f1) && ID_libro != id)
         fscanf(f1,"%d%s%s%d%d%f", &id, nom, aut, &calif, ex, precio);
   fclose(f1);
   if(ID_libro == id) *band = 1;
}

void FILTRAR(int c_min, float p_max)
{
   int id, calif, ex, band = 0;
   char nom[30], aut[30];
   float precio;
   FILE *f1;

   f1 = fopen("LIBROS.DAT","r");
   fscanf(f1,"%d%s%s%d%d%f", &id, nom, aut, &calif, &ex, &precio);
   while(!feof(f1))
   {
         if(calif >= c_min && precio <= p_max)
         {
               printf("%d %s %s %d %d %f\n", id, nom, aut, calif, ex, precio);
               if(band == 0) band = 1;
         }
         fscanf(f1,"%d%s%s%d%d%f", &id, nom, aut, &calif, &ex, &precio);
   }
   fclose(f1);
   if(band == 0) printf("Ning%cn libro cumple con el criterio ingresado\n", 163);
}

int main()
{
   int ID, c_min, ex, band;
   float p_max, prec;
   char tit[30];

   // ítem a
   MOSTRAR();
   // ítem b
   band = 0;
   printf("Ingrese un c%cdigo de identificaci%Cn a buscar\n", 162, 162);
   scanf("%d", &ID);
   BUSCAR(ID, &band, &ex, &prec, tit);
   if(band != 0)
         printf("%s\t%d\t%g\n", tit, ex, prec);
   else
         printf("Libro no encontrado\n");
   // ítem c
   printf("Ingrese una calificaci%cn m%cnima a buscar\n", 162, 161);
```

```
   scanf("%d", &c_min);
   printf("Ingrese un precio m%cximo\n", 160);
   scanf("%f", &p_max);
   FILTRAR(c_min, p_max);
   return 0;
}
```

En caso de utilizar la primera lógica descripta y utilizando un registro simple, el algoritmo principal permanecerá igual, y los subalgoritmos resultarán:

```
Tipo
        Registro libro
        entero: id, calif, ex
        caracter: nom[30], aut[30]
        real: precio
        fin registro

Subrutina MOSTRAR()
Variables
        libro: L
Inicio
        abrir(f1, "LIBROS.DAT", lectura)
        Leer(f1, L.id, L.nom, L.aut, L.calif, L.ex, L.precio)
        Repetir mientras(no(eof(f1)))
          Escribir(L.id, L.nom, L.aut, L.calif, L.ex, L.precio)
          Leer(f1, L.id, L.nom, L.aut, L.calif, L.ex, L.precio)
        fin mientras
        cerrar(f1)
fin subrutina

Subrutina BUSCAR(E: entero: ID_libro; S: entero: band, ex, real: precio, caracter(30): nom)
Variables:
        libro: L
Inicio
        abrir(f1, "LIBROS.DAT", lectura)
        Leer(f1, L.id, L.nom, L.aut, L.calif, L.ex, L.precio)
        Repetir mientras(no(eof(f1)) y ID_libro <> L.id)
          Leer(f1, L.id, L.nom, L.aut, L.calif, L.ex, L.precio)
        fin mientras
        cerrar(f1)
        Si(ID_libro = id) entonces
          band ← 1
          ex ← L.ex
```

```
            precio ← L.precio
            nom ← L.nom
        fin si
fin subrutina

Subrutina FILTRAR(E: entero: c_min, real:p_max)
Variables
        libro: L
        entero: band
Inicio
        band ← 0
        abrir(f1, "LIBROS.DAT", lectura)
        Leer(f1, L.id, L.nom, L.aut, L.calif, L.ex, L.precio)
        Repetir mientras(no(eof(f1)))
          Si(L.calif >= c_min y L.precio <= p_max) entonces
        Escribir(L.id, L.nom, L.aut, L.calif, L.ex, L.precio)
        Si(band = 0) entonces
                band ← 1
        fin si
          fin si
          Leer(f1, L.id, L.nom, L.aut, L.calif, L.ex, L.precio)
        fin mientras
        cerrar(f1)
        Si(band = 0) entonces
          Escribir("Ningún libro cumple con el criterio ingresado")
        fin si
fin subrutina
```

Cuya codificación en lenguaje C es:

```
#include <stdio.h>
#include <string.h>

struct libro
{
   int id, calif, ex;
   char nom[30], aut[30];
   float precio;
};



void MOSTRAR()
```

```
{
   struct libro L;
   FILE *f1;

   f1 = fopen("LIBROS.DAT","r");
   fscanf(f1,"%d%s%s%d%d%f", &L.id, L.nom, L.aut, &L.calif, &L.ex, &L.precio);
   while(!feof(f1))
   {
         printf("%d %s %s %d %d %g\n", L.id, L.nom, L.aut, L.calif, L.ex, L.precio);
         fscanf(f1,"%d%s%s%d%d%f", &L.id, L.nom, L.aut, &L.calif, &L.ex, &L.precio);
   }
   fclose(f1);
}

void BUSCAR(int ID_libro, int *band, int *ex, float *precio, char nom[30])
{
   struct libro L;
   FILE *f1;

   f1 = fopen("LIBROS.DAT","r");
   fscanf(f1,"%d%s%s%d%d%f", &L.id, L.nom, L.aut, &L.calif, &L.ex, &L.precio);
   while(!feof(f1) && ID_libro != L.id)
         fscanf(f1,"%d%s%s%d%d%f", &L.id, L.nom, L.aut, &L.calif, &L.ex, &L.precio);
   fclose(f1);
   if(ID_libro == L.id)
   {
         *band = 1;
         *ex = L.ex;
         *precio = L.precio;
         strcpy(nom, L.nom);
   }
}

void FILTRAR(int c_min, float p_max)
{
   struct libro L;
   int band = 0;
   FILE *f1;

   f1 = fopen("LIBROS.DAT","r");
   fscanf(f1,"%d%s%s%d%d%f", &L.id, L.nom, L.aut, &L.calif, &L.ex, &L.precio);
   while(!feof(f1))
   {
```

```
        if(L.calif >= c_min && L.precio <= p_max)
        {
                printf("%d %s %s %d %d %g\n", L.id, L.nom, L.aut, L.calif, L.ex, L.precio);
                if(band == 0) band = 1;
        }
        fscanf(f1,"%d%s%s%d%d%f", &L.id, L.nom, L.aut, &L.calif, &L.ex, &L.precio);
   }
   fclose(f1);
   if(band == 0)
        printf("Ning%Cn libro cumple con el criterio ingresado\n", 163);
}
```

A continuación, se muestra la resolución del problema utilizando la segunda lógica descripta y utilizando arreglos de variables simples:

```
Algoritmo problema_libros
Variables:
        entero: ID, c_min, id[500], calif[500], ex[500], cant, band, ind
        real: p_max, precio[500]
        caracter(30): nom[500], aut[500]
Inicio
        // ítem a
        MOSTRAR(id, nom, aut, calif, ex, precio, cant)
        // ítem b
        // ítem b
        band ← 0
        Escribir("Ingrese un código de identificación a buscar")
        Leer(ID)
        BUSCAR(ID, id, cant, band, ind)
        Si(band <> 0) entonces
                Escribir(nom[ind], ex[ind], prec[ind])
         sino
                Escribir("Libro no encontrado")
        fin si
        // ítem c
        Escribir("Ingrese una calificación mínima a buscar")
        Leer(c_min)
        Escribir("Ingrese un precio máximo")
        Leer(p_max)
        FILTRAR(c_min, p_max, id, nom, aut, calif, ex, precio, cant)
Fin
```

```
Subrutina MOSTRAR(S: entero: id[500], caracter: nom[500][30], aut[500][30], entero:
calif[500], ex[500], real: precio[500], entero: cant)
Variables
        entero: i
Inicio
        abrir(f1, "LIBROS.DAT", lectura)
        i ← 1
        Leer(f1, id[i], nom[i], aut[i], calif[i], ex[i], precio[i])
        Repetir mientras(no(eof(f1)))
          i ← i + 1
          Leer(f1, id[i], nom[i], aut[i], calif[i], ex[i], precio[i])
        fin mientras
        cant ← i
        cerrar(f1)
        Repetir para i ← 1, cant
          Escribir(id[i], nom[i], aut[i], calif[i], ex[i], precio[i])
        fin para
fin subrutina

Subrutina BUSCAR(E: entero: ID_libro, id[500], cant; S: entero: band, ind)
Variables:
        entero: i
Inicio
        i ← 1
        Repetir mientras(i <= cant y ID_libro <> id[i])
          i ← i + 1
        fin mientras
        Si(ID_libro = id[i]) entonces
          band ← 1
          ind ← i
        fin si
fin subrutina

Subrutina FILTRAR(E: entero: c_min, real: p_max, entero: id[500], caracter: nom[500][30],
aut[500][30], entero: calif[500], ex[500], real: precio[500], entero: cant)
Variables
        entero: i
Inicio
        band ← 0
        Repetir para i ← 1, cant
          Si(calif[i] >= c_min y precio[i] <= p_max) entonces
        Escribir(id[i], nom[i], aut[i], calif[i], ex[i], precio[i])
        Si(band = 0) entonces
```

```
                band ← 1
        fin si
          fin si
        fin para
        Si(band = 0) entonces
          Escribir("Ningún libro cumple con el criterio ingresado")
        fin si
fin subrutina
```

Su codificación en lenguaje C es:

```
#include <stdio.h>

void MOSTRAR(int id[500], char nom[500][30], char aut[500][30], int calif[500], int ex[500],
float precio[500], int *cant)
{
   int i = 0;
   FILE *f1;

   f1 = fopen("LIBROS.DAT","r");
   fscanf(f1,"%d%s%s%d%d%f", &id[i], nom[i], aut[i], &calif[i], &ex[i], &precio[i]);
   while(!feof(f1))
   {
         i++;
         fscanf(f1,"%d%s%s%d%d%f", &id[i], nom[i], aut[i], &calif[i], &ex[i], &precio[i]);
   }
   fclose(f1);
   *cant = i;
   for(i=0; i<*cant; i++)
         printf("%d %s %s %d %d %g\n", id[i], nom[i], aut[i], calif[i], ex[i], precio[i]);
}

void BUSCAR(int ID_libro, int id[500], int cant, int *band, int *ind)
{
   int i = 0;

   while(i<cant && ID_libro != id[i]) i++;
   if(ID_libro == id[i])
   {
         *band = 1;
         *ind = i;
   }
}
```

```
void FILTRAR(int c_min, float p_max, int id[500], char nom[500][30], char aut[500][30], int
calif[500], int ex[500], float precio[500], int cant)
{
   int i, band = 0;

   for(i=0; i<cant; i++)
   {
         if(calif[i] >= c_min && precio[i] <= p_max)
         {
               printf("%d %s %s %d %d %g\n", id[i], nom[i], aut[i], calif[i], ex[i], precio[i]);
               if(band == 0) band = 1;
         }
   }
   if(band == 0) printf("Ningún libro cumple con el criterio ingresado\n");
}

int main()
{
   int ID, c_min, band, id[500], calif[500], ex[500], cant, ind;
   float p_max, precio[500];
   char nom[500][30], aut[500][30];

   // ítem a
   MOSTRAR(id, nom, aut, calif, ex, precio, &cant);
   // ítem b
   band = 0;
   printf("Ingrese un c%cdigo de identificaci%cn a buscar\n", 162, 162);
   scanf("%d", &ID);
   BUSCAR(ID, id, cant, &band, &ind);
   if(band != 0)
         printf("%s\t%d\t%g\n", nom[ind], ex[ind], precio[ind]);
   else
         printf("Libro no encontrado\n");
   // ítem c
   printf("Ingrese una calificaci%cn m%cnima a buscar\n", 162, 161);
   scanf("%d", &c_min);
   printf("Ingrese un precio m%cximo\n", 160);
   scanf("%f", &p_max);
   FILTRAR(c_min, p_max, id, nom, aut, calif, ex, precio, cant);
   return 0;
}
```

Finalmente, se muestra la resolución del problema utilizando la segunda lógica descripta y utilizando arreglos de registros:

```
Algoritmo problema_libros
Variables:
        entero: ID, c_min, cant, band, ind
        real: p_max
        libro: L[500]
Inicio
        // ítem a
        MOSTRAR(L, cant)
        // ítem b
        Escribir("Ingrese un código de identificación a buscar")
        Leer(ID)
        BUSCAR(ID, id, cant, band, ind)
        Si(band <> 0) entonces
                Escribir(L[ind].nom, L[ind].ex, L[ind].prec)
         sino
                Escribir("Libro no encontrado")
        fin si
        // ítem c
        Escribir("Ingrese una calificación mínima a buscar")
        Leer(c_min)
        Escribir("Ingrese un precio máximo")
        Leer(p_max)
        FILTRAR(c_min, p_max, L, cant)
Fin

Subrutina MOSTRAR(S: libro: L[500], entero: cant)
Variables
        entero: i
Inicio
        abrir(f1, "LIBROS.DAT", lectura)
        i ← 0
        Repetir mientras(no(eof(f1)))
          i ← i + 1
          Leer(f1, L[i].id, L[i].nom, L[i].aut, L[i].calif, L[i].ex, L[i].precio)
        fin mientras
        cant ← i
        cerrar(f1)
        Repetir para i ← 1, cant
          Escribir(L[i].id, L[i].nom, L[i].aut, L[i].calif, L[i].ex, L[i].precio)
        fin para
```

```
fin subrutina

Subrutina BUSCAR(E: entero: ID_libro, libro: L[500], cant; S: entero: band, ind)
Variables:
        entero: i
Inicio
        i ← 1
        Repetir mientras(i <= cant y ID_libro <> L[i].id)
          i ← i + 1
        fin mientras
        Si(ID_libro = L[i].id) entonces
          band ← 1
          ind ← i
        fin si
fin subrutina

Subrutina FILTRAR(E: entero: c_min, real:p_max, libro: L[500], entero: cant)
Variables
        entero: i
Inicio
        band ← 0
        Repetir para i ← 1, cant
          Si(L[i].calif >= c_min y L[i].precio <= p_max) entonces
        Escribir(L[i].id, L[i].nom, L[i].aut, L[i].calif, L[i].ex, L[i].precio)
        Si(band = 0) entonces
                band ← 1
        fin si
          fin si
        fin para
        Si(band = 0) entonces
          Escribir("Ningún libro cumple con el criterio ingresado")
        fin si
fin subrutina
```

Su codificación en lenguaje C es:

```
#include <stdio.h>

struct libro
{
   int id, calif, ex;
   char nom[30], aut[30];
   float precio;
```

```
};

void MOSTRAR(struct libro L[500], int *cant)
{
   int i = 0;
   FILE *f1;

   f1 = fopen("LIBROS.DAT","r");
   fscanf(f1,"%d%s%s%d%d%f", &L[i].id, L[i].nom, L[i].aut, &L[i].calif, &L[i].ex, &L[i].precio);
   while(!feof(f1))
   {
         i++;
         fscanf(f1,"%d%s%s%d%d%f", &L[i].id, L[i].nom, L[i].aut, &L[i].calif, &L[i].ex,
&L[i].precio);
   }
   fclose(f1);
   *cant = i;
   for(i=0; i<*cant; i++)
         printf("%d %s %s %d %d %g\n", L[i].id, L[i].nom, L[i].aut, L[i].calif, L[i].ex, L[i].precio);
}

void BUSCAR(int ID_libro, struct libro L[500], int cant, int *band, int *ind)
{
   int i = 0;

   while(i<cant && ID_libro != L[i].id) i++;
   if(ID_libro == L[i].id)
   {
         *band = 1;
         *ind = i;
   }
}

void FILTRAR(int c_min, float p_max, struct libro L[500], int cant)
{
   int i, band = 0;

   for(i=0; i<cant; i++)
   {
         if(L[i].calif >= c_min && L[i].precio <= p_max)
         {
                  printf("%d %s %s %d %d %g\n", L[i].id, L[i].nom, L[i].aut, L[i].calif, L[i].ex,
L[i].precio);
```

```
                if(band == 0) band = 1;
            }
        }
        if(band == 0) printf("Ningún libro cumple con el criterio ingresado\n");
}

int main()
{
    int ID, c_min, cant, band, ind;
    float p_max;
    struct libro L[500];

    // ítem a
    MOSTRAR(L, &cant);
    // ítem b
    band = 0;
    printf("Ingrese un c%cdigo de identificaci%cn a buscar\n", 162, 162);
    scanf("%d", &ID);
    BUSCAR(ID, L, cant, &band, &ind);
    if(band != 0)
        printf("%s\t%d\t%g\n", L[ind].nom, L[ind].ex, L[ind].precio);
    else
        printf("Libro no encontrado\n");
    // ítem c
    printf("Ingrese una calificaci%cn m%cnima a buscar\n", 162, 161);
    scanf("%d", &c_min);
    printf("Ingrese un precio m%cximo\n", 160);
    scanf("%f", &p_max);
    FILTRAR(c_min, p_max, L, cant);
    return 0;
}
```