Ejercicios de práctica para Examen Libre

Los problemas planteados en esta práctica reúnen la totalidad de los conceptos vertidos en la asignatura Informática. Para cada uno de ellos realizar el análisis del problema, escribir el algoritmo que permita resolverlo, y su posterior codificación en Lenguaje C, ejecución, prueba y verificación del correcto funcionamiento del programa.

1) Una fábrica de fuentes de alimentación produce 20 modelos diferentes. Al finalizar el mes se deben procesar ciertos archivos con información del stock, lo producido y lo vendido. Los archivos disponibles son los siguientes:

STOCK.DAT: Contiene la información de la cantidad de fuentes de alimentación de cada modelo que tiene la fábrica al iniciar el mes. Los campos de este archivo son: cod (el código de fuente, entero entre 1 y 20), modelo (cadena de 5 caracteres con el modelo de la fuente), cantidad (la cantidad de fuentes de ese modelo que hay en stock),

PROD.DAT: contiene información de lo producido por día y por modelo. Los campos son: dia (número de día), cod (el código de fuente, entero entre 1 y 20), modelo (cadena de 5 caracteres con el modelo de la fuente), cantidad (la cantidad de fuentes de ese modelo que se produjo ese día)

VENTAS.DAT: contiene información de lo vendido por día y por modelo. Los campos son: dia (número de día), cod (el código de fuente, entero entre 1 y 20), modelo (cadena de 5 caracteres con el modelo de la fuente), cantidad (la cantidad de fuentes de ese modelo que se vendió ese día)

No necesariamente hubo producción todos los días así como tampoco necesariamente se produjeron todos los modelos todos los días. La misma observación vale para las ventas.

Se solicita hacer el análisis del problema y el algoritmo en pseudocódigo, que primero procese los datos de los archivos, almacenando los datos procesados en la/s estructura/s de datos que considere pertinentes acorde a lo que se solicita resolver, y que luego, través de un menú de opciones, permita:

1. Mostrar stock (mostrar el stock de cada fuente al final del mes, ordenado por cantidad, en forma descendente)

2. Calcular ingresos (debe mostrar el total de ingresos a partir de los datos de las cantidades vendidas y los precios de cada fuente; estos últimos datos deberán leerse desde teclado)

3. Mostrar el promedio de ventas (cantidad de unidades por venta)

4. Ingresar un modelo de fuente y mostrar el stock del mismo

0. Salir

2) Una industria petroquímica posee 30 tanques. De cada tanque se dispone de la siguiente información, la cual está contenida en el archivo **TANQUES.DAT**: número del tanque (valor entre 1 y 30), nombre del producto que almacena (hasta 20 caracteres), capacidad total en litros (valor entero de 6 dígitos) y existencia actual en litros (valor entero de 6 dígitos). La información contenida en este archivo no necesariamente está ordenada por número de tanque. Se asume que toda la información contenida es correcta.

Por otro lado, se dispone del archivo **MOVIM.DAT** que contiene el detalle de los movimientos (entradas y salidas) producidos durante el último mes en los tanques de la industria. Cada registro de este archivo contiene los siguientes datos: número de tanque, día del mes, tipo de movimiento (1 carácter) (E=entrada o S=salida) y cantidad en litros. No necesariamente todos los tanques tuvieron movimiento todos los días del mes. Además, los datos contenidos en este archivo no se encuentran ordenados con criterio alguno. Se asume un mes de 30 días.

Es factible que alguno de los datos contenidos en este archivo no sea correcto. Por ejemplo, que el número del tanque o el número de día, no se encuentre dentro del rango esperado, o que la cantidad sea cero o negativa, o que el tipo de movimiento no sea E o S.

Se solicita hacer el análisis del problema y el algoritmo en pseudocódigo, para que a partir de la lectura de los datos contenidos en los archivos **TANQUES.DAT** y **MOVIM.DAT**, se:

- Cree un archivo **ERRORES.DAT** con todos los registros no válidos del archivo **MOVIM.DAT** por tener uno o varios de sus datos incorrectos.
- Genere un listado que se emita por pantalla, ordenado en forma decreciente por capacidad del tanque, que muestre los siguientes datos: número de tanque, nombre del producto almacenado, capacidad y existencia actual. No se deberán mostrar los tanques que estén vacíos. Se asume que todas las capacidades son diferentes.
- Muestre por pantalla el número de tanque que tenga menor disponibilidad (cantidad de litros que le faltan para estar completo). Se asume que solamente habrá un tanque.

---

3) Un sistema de gestión de una biblioteca genera diversos archivos donde guarda toda la información relacionada a las clásicas tareas de administración de libros y el control de los préstamos.

Se dispone de los siguientes archivos extraídos de este sistema:

listado-JULIO.txt: Contiene la información de todos los libros que posee la biblioteca al comenzar el mes de julio. Cada registro lógico del archivo está compuesto de 5 campos:

código (entero), título (cadena de 100 caracteres), autor (cadena de 50 caracteres), año de edición (entero) y cantidad total de ejemplares (entero). Los registros están ordenados por código de menor a mayor.

mov-JULIO.txt: contiene información sobre todos los movimientos (préstamos y devoluciones) de libros durante todo el mes de julio. Cada registro lógico del archivo está compuesto por dos campos: código del libro (entero) y tipo de movimiento (caracter, 'P' por préstamo / 'D' por devolución). Este archivo está ordenado cronológicamente. Tener en cuenta que los códigos de libros se pueden repetir en diferentes registros al prestarse varios libros del mismo tipo. Además, no se sabe cuantos registros tiene el archivo.

Notas: Puede ocurrir que algunos libros no hayan tenido movimientos en el período considerado. Se sabe que la biblioteca posee 532 libros. El código del libro es un número entero cualesquiera, no necesariamente consecutivo.

Se solicita realizar un algoritmo en pseudocódigo que primero cargue la información necesaria de estos archivos en estructura/s de datos a través del llamado a una subrutina de nombre CARGA que debe devolver dicha/s estructura/s al algoritmo principal. Luego presente un menú de opciones para realizar las siguientes 4 tareas:

1- Listar código, título y autor de los libros prestados al finalizar el mes de julio, ordenados de forma creciente por código. Para este fin implemente una subrutina LISTADO que reciba la/s estructura/s generadas por CARGA y muestre en pantalla el listado solicitado.

2- Buscar por código si un libro está disponible al finalizar el mes de julio. Debe implementar una búsqueda dicotómica definiendo una función BUSCAR que tome como parámetro el código del libro buscado, la/s estructura/s de datos generadas por CARGA y devuelva: -1 (si el código es inexistente) o x, siendo x la cantidad de ejemplares disponibles de dicho libro (puede ser 0). El algoritmo principal mostrará un mensaje en pantalla al usuario: "Código inexistente", "x ejemplares disponibles" o "Libro no disponible" en el caso que no haya mas ejemplares en disposición.

3- Generar un archivo con la lista de libros que se encuentran en préstamos al finalizar el mes de julio. Debe implementar una subrutina GENERA que tomando como parámetro la/s estructura/s de datos generadas por CARGA, genere el archivo prest-JULIO.txt donde se guarde un resumen de los libros que se encuentran en préstamo al momento de finalizar el mes de JULIO. Cada registro lógico del archivo debe guardar 2 campos: código del libro prestado (entero), cantidad de ejemplares prestados (enteros).

4- Salir del Menú.

4) Control y gestión de un estacionamiento automático de automóviles.

Un estacionamiento dispone de 180 lugares numerados del 1 al 180.

En el mismo, un sistema de cámaras realiza un registro automático de la patente del automóvil y la hora de ingreso y de egreso del estacionamiento. Esta información es procesada, y junto con el número de cochera ocupada por el vehículo, se guarda en el archivo de nombre ingresos.txt

Los datos por registro del archivo son:

patente (7 caracteres)
número de cochera (entero de 1 a 180)
minutos totales de estadía del vehículo (entero)

Tener en cuenta que el número de cochera puede repetirse varias veces o ninguna dentro del archivo y además no se sabe cuantos registros tiene el archivo.

Se solicita escribir un algoritmo/programa que:

1- Llame a una subrutina de nombre CARGA que abra el archivo ingresos.txt y a partir de dicha información almacene en una estructura de datos adecuada el tiempo total ocupado por cada cochera. La estructura utilizada debe ser pasada por parámetros al programa principal.

2- Llame a una subrutina de nombre TPO_XCOCHERA que recibiendo por parámetros la estructura generada en CARGA, muestre “nro. de cochera y tiempo ocupada” dentro del mismo subalgoritmo.

3- Llame a una función de nombre NUNCA que recibiendo por parámetros la estructura generada en CARGA determine cuántas cocheras nunca fueron ocupadas. El resultado debe ser mostrado en el programa principal.

4- Llame a una subrutina de nombre MAS_REQUERIDAS que recibiendo por parámetros la estructura generada en CARGA determine cuales cocheras se utilizaron por más de 500 minutos, es decir, las más requeridas. Mostrar dicho listado dentro de la misma subrutina.

Escribir los subalgoritmos CARGA, TPO_XCOCHERA, NUNCA y MAS_REQUERIDAS con los parámetros correspondientes.

5) Se cuenta con las mediciones de radiación solar (irradiancia) de todos los días del año 2017, para la ciudad de Rosario. A partir de dichos datos, se desea analizarlos 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: nro. entero, **Día**: nro. entero, **Irradiancia**: nro. real, **Horas solares diarias**: nro. entero

Se pide diseñar un algoritmo que deberá procesar y cargar la información necesaria de este archivo en una/s estructura/s de datos adecuadas 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 algoritmo principal.

Luego, se debe mostrar por pantalla el siguiente menú:

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

Cada uno de los ítems del menú deben:

1- Invocar a una subrutina de nombre **LISTADO** que muestre por pantalla un listado indicando la irradiancia promedio de cada mes. Este listado debe estar ordenado por promedio en forma descendente, además debe figurar también los números del mes correspondiente.

2- Mostrar la irradiancia promedio diaria de un mes específico que el algoritmo principal debe pedir al usuario (invocar una función de nombre **DIARIA**).

3- Muestra el mes con mayor cantidad de horas de luz solar (invocar una función de nombre **MAYOR**)

4- Da por finalizado el programa.

NOTA IMPORTANTE: Los subalgoritmos de los ítems 1) 2) y 3) del menú deben tener como parámetro por lo menos la/s estructura/s devuelta/s por **CARGA** y no debe volver a leerse el archivo.

6) Una empresa nos solicita que realicemos un algoritmo para la liquidación de sueldos de sus operarios.

Para ello dispone del archivo **empleados.dat** cuyo registro lógico está compuesto de 4 campos: código de operario (entero), apellido (carácter 25), nombre (carácter 25), sueldo (real), el mismo se encuentra ordenado por código de operario. A la vez, cuenta con otro archivo **hsextras.dat**, cada registro lógico del archivo está compuesto por 4 campos: código de operario (entero), número de mes (entero de 1 a 12), año (entero) y cantidad acumulada de horas extras (entero). No todos los operarios hacen horas extras todos los meses. Para un determinado mes y año, el código de operario aparecerá una única vez con su correspondiente cantidad acumulada de horas extras. Este archivo no tiene ningún orden.

Se solicita hacer un algoritmo principal que llame a la subrutina **CARGA** que permita la carga del archivo **empelados.dat** en una adecuada estructura de datos que se deberá devolver al algoritmo principal. La empresa no tiene más de 70 operarios.
Luego el algoritmo principal deberá quedarse en un lazo de repetición mostrando el siguiente menú de opciones:

*** Menú Liquidación de Sueldos ***
1- Alta Hs. Extras
2- Listado Liquidación
1- Salir

ALTA: subrutina que permite el alta de nuevos registros en el archivo **hsextras.dat**. Solicitar el ingreso por teclado de un número de mes y año a dar de alta. Posteriormente solicitar en forma reiterada: un código de operario y la cantidad de horas extras trabajadas por ese operario (validar que sean mayor que cero), en el mes y año ingresados previamente. Finalizar el ingreso con código de operario = 0.

Con cada ingreso de código de operario se deberá verificar que el mismo sea válido, para ello se invocará a la función **BUSQUEDADICO** que recibiendo como parámetros la estructura generada en la subrutina CARGA y el código de un operario, esta función mediante búsqueda dicotómica deberá encontrar el código buscado y devolver un valor a su elección, este valor indicará si lo encontró o no, informando por pantalla sólo en el caso que no haya sido encontrado: “código de operario inexistente”.

Si es un código válido, pero ya fue ingresado ese código, mes y año en el archivo hsextras.dat se deberá informar “El registro ya existe en el archivo hsextras.dat”, en caso contrario se agregarán los datos en un nuevo registro en el archivo hsextras.dat.

LISTADO: subrutina que solicita el ingreso por teclado de tres datos: un número de mes (entero), un año a liquidar (entero) y el valor de la hora extra (real). Luego mostrar por pantalla en forma ordenada por código de operario: el código del operario, el apellido y nombre, el sueldo, la cantidad acumulada de horas extras y el neto a cobrar

(sueldo+cantidad*valor hora extra) del mes y año ingresados.

Al salir del menú llamar a la subrutina **GENERA** que deberá generar un nuevo archivo de nombre **estadísticas2017.dat** con la cantidad total de horas acumuladas en cada mes del año actual. El registro lógico del archivo tendrá 2 campos: número de mes (entero) y cantidad total de horas extras acumuladas (entero). La información de grabará en forma ordenada cantidad total de horas extras acumuladas, previamente se deberá invocar a la subrutina **ORDENA.**

---

7) A fines de este año comenzará a disputarse una competencia entre los equipos de distintas naciones del mundo y debemos realizar el algoritmo que permita obtener el podio donde se deberá mostrar el nombre del equipo y su puesto respectivo. Para esto contamos con cierta información brindada por la organización del evento.

La clasificación se realizó en mayo de este año y la información se encuentra en el archivo seleccionados.txt. En el mismo se encuentran los datos de los 32 países seleccionados: el código del equipo (número entero entre 1 y 32 sin repeticiones) y el nombre del equipo (ningún nombre de nación supera los 20 caracteres).

Los 3 primeros registros son los siguientes:

Código de equipo	Nombre del equipo
3	Argentina
21	Japón
17	Estados_Unidos
…	…

Por otro lado, al finalizar la competencia nos otorgarán las planillas con la información de cada encuentro deportivo ordenado de manera temporal. La planilla tiene 5 columnas, a saber:
Fecha: 3 enteros: día, mes y año. Código del equipo 1. Código del equipo 2.
Goles realizados por el equipo 1. Goles realizados por el equipo 2.

Los datos de estas planillas se ingresan por teclado y no se sabe la cantidad de encuentros disputados.

Se solicita realizar un algoritmo en pseudocódigo o Lenguaje C que permita realizar las siguientes tareas:

1) En el algoritmo principal cargar en una estructura de datos adecuada de nombre **EQUIPOS** los nombres de las naciones participantes, ordenados por número de código. Dibujar dicha estructura.

2) Luego invocar una subrutina de nombre **CARGA** que permita, a partir de la planilla, almacenar en otra estructura de datos de nombre **PUNTAJES** los puntos que cada equipo

obtuvo, sabiendo qué:

*Si un equipo realizó más goles que el otro, ese equipo tendrá 5 puntos y el otro 0.
*Si tienen igual cantidad de goles realizados, ambos reciben 2 puntos.

3) Luego presente un menú de opciones como el siguiente:

MENÚ
Opción 1: Los 3 Campeones.
Opción 2: Los Mayores puntajes.
Opción 3: Puntos de un país.
Opción 4: Salir.

Opción 1: Invocar una subrutina de nombre ORDENAR que, recibiendo sólo las estructuras EQUIPOS y PUNTAJES, devuelva esas mismas estructuras ordenadas por cantidad de puntos de modo tal que se pueda mostrar en el algoritmo principal los 3 primeros equipos (nombre) y su puntaje.

Opción 2: Se debe solicitar al usuario un valor entero V. Luego, mediante el uso de una función de nombre SUPERAN, cuyos parámetros son los resultados obtenidos en ORDENAR y el valor V, determinar cuántos equipos obtuvieron V puntos o más. Mostrar el resultado en el algoritmo principal con un mensaje claro para el usuario.

Opción 3: dado el nombre de un país mostrar su puntaje.

Para ello primero se debe solicitar al usuario un nombre P de un país y se llamará a una función de nombre BÚSQUEDA que recibiendo las estructuras EQUIPOS y PUNTAJES, y el nombre P, realice una búsqueda para hallar el puntaje del país P. Considerar que el nombre P ingresado puede estar mal escrito y por lo tanto no se lo encuentre.

Mostrar el resultado en el algoritmo principal con un mensaje claro para el usuario.

Opción 4: fin del algoritmo

Aclaraciones:
*Ningún archivo tiene algún orden establecido.
*Se garantiza que los 3 equipos del podio no tienen los mismos puntos, ni entre ellos ni con ninguno de los demás equipos.

Ejercicios resueltos

En la resolución:

- Utilizar al menos un subprograma de tipo subrutina y otro de tipo función.
- El intercambio de datos entre el programa principal y los subalgoritmos se debe realizar mediante parámetros.

1) Un kiosco vende cubanitos rellenos de 3 sabores distintos: chocolate, dulce de leche y pasta de maní.
Los cubanitos se venden en paquetitos individuales de 1 cubanito o en cajas de 6 cubanitos. Las cajas pueden contener cubanitos todos del mismo sabor o de sabores mezclados (2 de cada sabor). Las cajas cuestan $50 y los paquetes individuales cuestan $10.
Se dispone de un archivo llamado **VENTAS.DAT** que contiene el detalle de las ventas realizadas en el último mes, con los siguientes datos: día del mes, tipo de empaque (Unitario/Caja), sabor (Chocolate/Dulce de leche/Pasta de maní/Mezcla) y cantidad vendida.
La información de este archivo no se encuentra ordenada con criterio alguno. Puede ser que existan días para los cuales no haya habido ninguna venta. Incluso, para un mismo día, puede existir más de un registro, ya que los registros no contienen totales por día, sino que representan el detalle de cada venta realizada. Se asume un mes de 31 días.

Se solicita hacer el análisis del problema y el algoritmo en pseudocódigo, que primero realice la carga de los datos del archivo **VENTAS.DAT** en la/s estructura/s de datos que considere pertinentes acorde a lo que se solicita resolver, y que luego, través de un menú de opciones, permita:

a) Para un determinado día del mes que el usuario indicará por teclado, conocer cuántos cubanitos se vendieron de cada sabor. Se deberá validar el día de forma conveniente.
b) Generar un archivo denominado **PORCENTAJES.DAT** que contenga, para los 31 días del mes, los porcentajes de ventas de cada uno de los 3 sabores. Los datos a grabar en dicho archivo son: día del mes y los tres porcentajes (chocolate, dulce de leche y pasta de maní).
c) Mostrar por pantalla, un listado ordenado en forma decreciente según la recaudación en pesos. Informar número de día e importe recaudado. No mostrar los días en los cuales la recaudación resultara nula. Este listado deberá también mostrar al final del mismo, el importe total recaudado.
d) Informar si en el mes se vendieron más cajas, más paquetitos individuales o igual cantidad de ambos tipos de empaque.

En pseudocódigo

```
Subrutina cargar_datos (S: entero datos[31][5], entero stat )
Variables:
        entero: dia , cantidad
        caracter: tipo[10], sabor[15]
```

```
        archivo: f
        entero: t, s, i , j
Inicio
    stat←0
    abrir(f, "ventas.dat","lectura")
    Si ( error_al_abrir(f) ) entonces
        Escribir("No se pudo abrir el archivo ventas.dat")
        stat ← 1
    sino
        Repetir para i←1, 31  //inicialización
            Repetir para j←1, 5
                datos[i][j]←0
            fin para
        fin para

        Leer(f, dia, tipo, sabor, cantidad)
        Repetir mientras( no eof(f) )
            t←1    //codificación de tipo en número
            Si ( tipo = "caja" ) entonces
                t←2
            fin si

       Según sea (sabor) //codificación del sabor en número
                caso "chocolate": s←1
                caso "dulce de leche": s←2
                caso "pasta de mani": s←3
                caso "mezcla": s←4
            fin según

            Si (t=1) entonces  //venta unitaria
                datos[dia][s] ← datos[dia][s] + cantidad //cantidad vendida del sabor s
                datos[dia][4] ← datos[dia][4] + cantidad //cant. total de unid. vendidas
            sino  //venta por caja
                Si(s<>4) entonces //venta de cajas del mismo sabor
                    datos[dia][s] ← datos[dia][s] + cantidad*6
                sino
                    datos[dia][1] ← datos[dia][1] + cantidad*2
                    datos[dia][2] ← datos[dia][2] + cantidad*2
                    datos[dia][3] ← datos[dia][3] + cantidad*2
                fin si
                datos[dia][5] ← datos[dia][5] + cantidad //cant. total de cajas vendidas
            fin si
            Leer(f, dia, tipo, sabor, cantidad)
        fin mientras
    fin si
cerrar(f)
Fin subrutina



Subrutina cantidad_cub (E: entero datos[31][5], entero dia,S: entero cantidades[3])
Variables entero: i
Inicio
    Repetir para i←1, 3
        cantidades[i] ← datos[dia][i]
    fin para
Fin subrutina


Subrutina porcentajes (E: entero datos[31][5], S: entero stat)
Variables   archivo: f
```

```
            entero: i
            real: suma
Inicio
    stat ← 0
    abrir(f,"porcentajes.dat","escritura")
    Si ( error_al_abrir(f) ) entonces
        Escribir("No se pudo abrir el archivo porcentajes.dat")
        stat ← 1
    sino
        Repetir para i←1, 31
            suma ← datos[i][4] + datos[i][5]*6
            si (suma <> 0) entonces
                Escribir( f, i, datos[i][1]/suma, datos[i][2]/suma,
datos[i][3]/suma )
            sino
                Escribir(f,i, 0, 0, 0)
            fin si
        fin para
    fin si
    cerrar(f)
Fin subrutina


Subrutina listado (E: entero datos[31][5])
variables   entero: i, j, cual, lugar, max, montos[31][2]
Inicio
    Repetir para i←1, 31
        montos[i][1] ← i
        montos[i][2] ← datos[i][4]*10 + datos[i][5]*50
    fin para

    Repetir para i←1, 30
        max  ← montos[i][2]
        cual  ← montos[i][1]
        lugar ← i
        Repetir para j ← i+1, 31
            Si (montos[j][2] > max) entonces
                max   ← montos[j][2]
                cual  ← montos[j][1]
                lugar ← j
            fin si
        fin para
        montos[lugar][1] ← montos[i][1]
        montos[lugar][2] ← montos[i][2]
        montos[i][1] ← cual
        montos[i][2] ← max
    fin para

    i←1
    Repetir mientras (i<32 y  montos[i][2]>0)
        Escribir("dia: " , montos[i][1] , " monto: ", montos[i][2] )
        i++
    fin mientras
Fin subrutina

Función mayor_venta (entero datos[31][5] ): entero
Variables   entero: i, suma_ind, suma_paq, may
Inicio
    suma_ind ← 0
    suma_paq ← 0
    may ← 0
    Repetir para i←1, 31
```

```
        suma_ind ← suma_ind + datos[i][4]
        suma_paq ← suma_paq + datos[i][5]
    fin para
    Si (suma_ind > suma_paq) entonces
        may ← 1
    sino
        Si (suma_ind < suma_paq) entonces
            may ← 2
        fin si
    fin si
    Devolver(may)
Fin función

Subrutina menu(S: entero opcion)
Inicio
    opcion←1

    Escribir ("1-Cantidad")
    Escribir ("2-Porcentaje")
    Escribir ("3-Listado")
    Escribir ("4-Mayor")
    Escribir ("5-Salir")

    Escribir ("Elija una opción: ")
    Leer(opcion)
    Repetir mientras(no (opcion>0 y opcion<6))
        Escribir ("Opción incorrecta, reingrese: ")
        Leer(opcion)
    fin mientras
Fin subrutina


Algoritmo Cubanitos
Variables entero: opcion, datos[31][5], stat, dia, cantidad[3], may
Inicio
        cargar_datos(datos, stat)
        Si (stat=0) entonces
            menu(opcion)
            Repetir mientras (opcion<>5)
                Según sea (opcion)
                    caso 1:
                        Escribir ("Ingrese el nro de dia: ")
                        Leer(dia)
                        Repetir mientras(no(dia>0 y dia<32))
                            Escribir ("Dia inválido, reingrese")
                            Leer(dia)
                        fin mientras
                        cantidad_cub(datos, dia, cantidad)
                        Escribir("Se vendieron")
                        Escribir("Chocolate: ", cantidad[1])
                        Escribir("Dulce de Leche: ", cantidad[2])
                        Escribir("Pasta de maní: ", cantidad[3])
                    caso 2:
                        porcentajes(datos, stat)
                        Si (stat=0) entonces
                            Escribir ("Se creó el archivo de porcentajes")
                        fin si
                    caso 3:
                        listado(datos)
                    caso 4:
                        may←mayor_venta(datos)
                        Si(may=0) entonces
```

```
                            Escribir ("Se vendieron la misma cantidad de empaques")
                        sino
                            Si (may=1) entonces
                               Escribir ("Se vendieron más paquetes individuales")
                            sino
                               Escribir ("Se vendieron más cajas")
                            fin si
                        fin si
                fin según
                menu(opcion)
            fin mientras
        fin si
Fin Algoritmo
```

Codificación en lenguaje C

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

void cargar_datos (int datos[31][5], int *stat ){

int dia , cantidad;
char tipo[10], sabor[15];
FILE *f;
int t, s, i , j;

    *stat = 0;
    f = fopen("ventas.txt","r");
    if(f==NULL){
        printf("No se pudo abrir el archivo ventas.txt");
        *stat = 1;
    } else {

        for(i=0; i<31 ; i++) //inicialización
            for(j=0; j<5 ; j++)
                datos[i][j]=0;


        fscanf(f,"%d %s %s %d", &dia, tipo, sabor, &cantidad);
        while( !feof(f) ){
            t=1; //codificación de tipo en número
            if ( strcmp(tipo,"caja") == 0 ) t=2;

            if ( strcmp(sabor,"chocolate") == 0 ) s=0; //codificación del sabor en
número
            else if ( strcmp(sabor,"dulce_de_leche") == 0 ) s=1;
            else if ( strcmp(sabor,"pasta_de_mani") == 0 ) s=2;
            else if ( strcmp(sabor,"mezcla") == 0 ) s=3;

            dia--;
            if(t==1) {  // venta unitaria
                datos[dia][s] += cantidad; //cantidad vendida del sabor s
                datos[dia][3] += cantidad; //cantidad total de unidades vendidas
            } else {

                if(s!=3){ //venta de cajas del mismo sabor
                    datos[dia][s] += cantidad*6;
                } else {
```

```
                    datos[dia][0] += cantidad*2;
                    datos[dia][1] += cantidad*2;
                    datos[dia][2] += cantidad*2;
                }
                datos[dia][4] += cantidad; //cantidad total de cajas vendidas
            }
            fscanf(f,"%d %s %s %d", &dia, tipo, sabor, &cantidad);
        }
    }
}


void cantidad_cub (int datos[31][5], int dia, int cantidades[3] ){
int i;

    for(i=0; i<3 ; i++)
        cantidades[i] = datos[dia-1][i];

}


void porcentajes(int datos[31][5],int *stat){
FILE *f;
int i;
float suma;

    *stat = 0;
    f = fopen("porcentajes.txt","w");
    if(f == NULL){
        printf("No se pudo abrir el archivo porcentajes.txt");
        *stat = 1;
    } else {

        for(i=0; i<31 ; i++){
            suma = datos[i][3] + datos[i][4]*6;
            if (suma != 0){
                fprintf( f,"%d %f %f %f\n", i+1, datos[i][0]/suma,
datos[i][1]/suma, datos[i][2]/suma );
            } else {
                fprintf( f,"%d %f %f %f\n", i+1, 0.0, 0.0, 0.0 );
            }

        }
    }
    fclose(f);
}


int mayor_venta ( int datos[31][5] ){

int i, suma_ind, suma_paq, may;

    suma_ind = 0;
    suma_paq = 0;
    may = 0;
    for(i=0; i<31 ; i++){
        suma_ind += datos[i][3];
        suma_paq += datos[i][4];
    }
    if( suma_ind>suma_paq){
        may = 1;
    } else {
```

```
       if ( suma_ind < suma_paq) may = 2;
    }
    return(may);
}

void listado (int datos[31][5]){

int i, j, cual, lugar, max, montos[31][2];

    for(i=0; i<31 ; i++){
        montos[i][0] = i+1;
        montos[i][1] = datos[i][3]*10 + datos[i][4]*50;
    }

    for(i=0; i<30 ; i++){
        max   = montos[i][1];
        cual  = montos[i][0];
        lugar = i;
        for(j=i+1; j<31 ; j++){
            if(montos[j][1] > max){
                max   = montos[j][1];
                cual  = montos[j][0];
                lugar = j;
            }
        }
        montos[lugar][0] = montos[i][0];
        montos[lugar][1] = montos[i][1];
        montos[i][0] = cual;
        montos[i][1] = max;
    }

    i=0;
    while (i<31 && montos[i][1]>0){
        printf("\n dia: %d monto: %d ", montos[i][0] , montos[i][1] );
        i++;
    }
}


void  menu(int *opcion) {

    *opcion=1;

    printf ("\n1-Cantidad\n");
    printf ("2-Porcentaje\n");
    printf ("3-Listado\n");
    printf ("4-Mayor\n");
    printf ("5-Salir\n");
    printf ("Elija una opci%cn: ", 162);
    scanf("%d",opcion);
    while(! (*opcion >0 && *opcion<6)){
        printf("Opci%cn incorrecta, reingrese: ", 162);
        scanf("%d",opcion);
    }

}

int main(){
int opcion, datos[31][5], stat, dia, cantidad[3], may;

    cargar_datos(datos, &stat);
    if (!stat){
```

```
        menu(&opcion);
        while (opcion != 5){
            switch(opcion){
                case 1:
                    printf("Ingrese el nro de d%ca: ", 161);
                    scanf("%d",&dia);
                    while(!(dia>0 && dia<32)){
                            printf ("D%ca inv%clido, reingrese: ", 161, 160);
                            scanf("%d",&dia);
                    }
                    cantidad_cub( datos, dia, cantidad );
                    printf("Se vendieron\n");
                    printf("Chocolate: %d \n", cantidad[0]);
                    printf("Dulce de Leche: %d \n", cantidad[1]);
                    printf("Pasta de man%c: %d \n", 161, cantidad[2]);
                    break;
                case 2:
                    porcentajes(datos, &stat);
                    if (!stat)
                            printf ("Se cre%c el archivo de porcentajes\n", 162);
                    break;
                case 3:
                            listado(datos);
                            break;
                case 4:
                            may=mayor_venta(datos);
                            if (!may){
                                printf("Se vendieron la misma cantidad de empaques\n");
                            } else {
                                if (may==1) {
                                  printf("Se vendieron m%cs paquetes individuales\n",
160);
                                } else {
                                  printf("Se vendieron m%cs cajas\n", 160);
                                }
                            }
                            break;
            }
            menu(&opcion);
        }
    }
}
```