PRACTICA 9 ALGORITMIA **Prof. Alejandro Frankel**
2020

ALGORITMOS DE APAREO Y CORTE DE CONTROL CON ARCHIVOS Y ARREGLOS

Nota grabación de cadenas de texto: cuando tenga que grabar caracteres en los archivos binarios, no trabaje con strings sino con cadenas de caracteres (char).
Por ejemplo si quiere incluir una descripción de 20 caracteres definirá char Desc [21]

91)
Se dispone de dos vectores (puede cargarlos a partir de archivos), uno con la inscripción de alumnos a exámenes finales **VecFINALES** y otro con las inscripciones del día de hoy **VecDIAFINALES.DAT**, ambos ordenados ascendente por código de materia y con el siguiente diseño:

a.1Nro de legajo (8 dígitos) a.2 Código de materia (6 dígitos) a.3 ApeNom(25caract)

Se pide desarrollar un programa que genere un nuevo vector de inscripciones a finales **VecFINALESACT** resultante del apareo de los dos archivos anteriores, con el mismo orden y diseño.
Luego de generado el vector, grábelo con un módulo a un archivo **VecFINALESACT.DAT**

92)
El dueño de un local de venta de libros desea relevar el stock que posee en el local y en el depósito, para realizar las compras del mes.
Para ello cuenta con dos archivos:

a) **StockEnLocal.dat**, ordenado por código del libro, con un registro por cada libro que se encuentra en el local, con el siguiente diseño:
a.1) Código de libro (4 dígitos) a.2) Título del libro (20 caracteres)
a.3) Stock en el local (entero) a.4) Genero (2 dígitos)

b) **StockEnDeposito.dat**, ordenado por código del libro, con un registro por cada libro que se encuentra en el depósito, con el siguiente diseño:
b.1) Código de libro (4 dígitos) b.2) Stock en depósito (entero)

Se pide desarrollar la metodología necesaria para escribir un algoritmo que emita un listado ordenado por código de libro, con un renglón por cada libro que tenga faltante en stock sea en depósito, local o en ambos lugares, con el siguiente formato:
Libros faltantes

Código	Observación
9999	Falta en depósito

9999 Falta en local
9999 Falta en local y en depósito

Total de libros en falta: 9999999

93)
Dado el archivo **„ALUMNOS.dat"** con los datos personales de alumnos ordenado por legajo, con el siguiente diseño:

a.1 Legajo (8 dígitos)	a.2 Apellido y nombre (30 caracteres)
a.3 Domicilio (20 caracteres)	a.4 Código postal (4 dígitos)
a.5 Teléfono (10 caracteres)	a.6 Año de ingreso (4 dígitos)

Y otro archivo con el mismo orden que el mencionado llamado **„NOVEDADES.dat"**, con las actualizaciones (altas, bajas, y modificaciones) a ser aplicadas, donde cada registro contiene además de todos los campos de Alumnos.dat un código de operación **(„A"= Alta, „B"= Baja, „M"= Modificación)**.

Se pide desarrollar todos los pasos necesarios para realizar un programa que genere un archivo actualizado **„ALUMACTU.dat"** con el mismo diseño

94)
Una aplicación para descargas de Series por la red posee la información en un archivo binario, **Episodios.dat**, con un registro por cada episodio, ordenado ascendente por **Id_Serie**, con el siguiente diseño:

1) Id_Serie (9 dígitos)	2) Número de temporada (1..12)
3) Número de episodio (byte)	4) Cantidad de descargas (longint)

Se pide desarrollar la metodología necesaria para escribir un algoritmo que emita el siguiente listado:

Listado de Descargas de Series

Serie: 999999999

Temporada	N. de Episodio	Cant. descargas
99	999	9999999
99	999	9999999

……..
Cant. Total de Episodios de la Serie: 99999
Total descargas de la Serie: 999999999

Serie: 999999999

Temporada	N. de Episodio	Cant. descargas
99	999	9999999
99	999	9999999

……..
Cant. Total de Episodios de la Serie: 99999
Total descargas de la Serie: 999999999

…

……..

Pie de impresión del total de Series:

Cantidad Total de series: 9999
Cantidad Total de descargas: 999999999

94') Sólo para quien está más avanzado; incluye corte de control por doble campo
Una aplicación para descargas de Series por la red posee la información en un archivo binario, **Episodios.dat**, con un registro por cada episodio, ordenado ascendente por **Id_Serie y Número de temporada**, con el siguiente diseño:

1) Id_Serie (9 dígitos) 2) Número de temporada (1..12)
3) Número de episodio (byte) 4) Cantidad de descargas (longint)

Se pide desarrollar la metodología necesaria para escribir un algoritmo que emita el siguiente listado:

Listado de Descargas de Series
Serie: 999999999

Temporada: 99

N. de Episodio	Cant. descargas
999	9999999
999	9999999

……..
Cant. Total de Episodios de la Temporada: 99999
Total descargas de la temporada:999999999

Temporada: 99

N. de Episodio	Cant. descargas
999	9999999
999	9999999

……..
Cant. Total de Episodios de la Temporada: 99999
Total descargas de la temporada:999999999

…

Pie de impresión por Serie:

Cant. Total de Episodios de la Serie: 99999
Total descargas de la Serie: 999999999

(próx serie)
Serie: 999999999

Temporada: 99

N. de Episodio	Cant. descargas
999	9999999
999	9999999

……..
Cant. Total de Episodios de la Temporada: 99999
Total descargas de la temporada:999999999

…

……..

Pie de impresión del total de Series:

Cantidad Total de series: 9999
Cantidad Total de descargas: 999999999

95) PARA ENTREGAR EN FORMA GRUPAL

Un comercio cuenta con un archivo binario de **no más de 5.000** ventas de la semana conteniendo **Cód Producto (3 díg) Desc (20 caracteres) Cant Vendida (entero) y Total Facturado en la venta (decimal)**
El archivo se encuentra **agrupado** por Cód de Producto.

Se pide obtener:

Por cada producto:
Promedio de cantidades vendidas de las ventas individual
Total facturado del producto
Cantidad de ventas de más de $3.000

Del total de productos:
Cantidad de **productos** que facturaron en total más de $5.000.

Listado de Productos que cuestan más de **$1.000**, **ordenados por Cód Producto**, con el siguiente formato

Cód Producto	**Desc**	**Pu**
999	XX..XX	99.999,99
999	XX..XX	99.999,99
..	**..**	**..**

El corte de control (los ciclos) deberán estar desarrollado en el main para poder apreciarlo bien. Dentro del corte de control es válido que haya llamada a módulos para resolver distintas partes que están fuera del ciclo o entre ciclos.

Una vez generado el vector con los productos que cumplen tener Precio Unitario (sin repetición de producto obviamente), invocará al final del main un módulo **Ordenar** que ordenará el vector por el campo pedido con el método de ordenamiento que prefiera.

Sobre el final del main desarrolle un ciclo de proceso que permita al usuario ingresar sucesivos Cód de Producto hasta 0, y que por cada uno invoque a una función **BusqBin** que lo busque en forma binaria en el vector de productos generado y ordenado previamente, y que devolverá la Descripción y Precio unitario de dicho producto o señal -1 si no está, e imprima por cada uno dichos datos o un cartel de que no se encuentra dicho producto.