Archivos de binarios - Actualizado 2024

1. Escriba un programa que permita a un usuario ingresar n valores float y se guarden en un archivo, el número n lo informa al comienzo del programa. Al finalizar el programa debe listar todos los números guardados en el archivo.
2. Agregue al código anterior la funcionalidad para poder modificar un número, solicitando al usuario en qué posición está el número que quiere modificar y cuál es el nuevo valor. Los cambios deben ser guardados en el archivo.
3. Dados dos archivos de enteros confeccione un programa que guarde en un tercer archivo el contenido de ambos intercalado de a un número.
4. Dados dos archivos de enteros, previamente ordenados ascendentemente, confeccione un programa que guarde en un tercer archivo el contenido de ambos ordenado **ascendentemente** – Utilice el algoritmo de apareo. Puede adaptar el visto en el ejercicio 7 de la guía de arrays.
5. Dados dos archivos de enteros, previamente ordenados ascendentemente, confeccione un programa que guarde en un tercer archivo el contenido de ambos ordenados **descendentemente**. – Utilice el algoritmo de apareo. Puede adaptar el visto en el ejercicio 7 de la guía de arrays.

6. Escriba un programa que guarde en un archivo tres alumnos y luego los liste por pantalla. El alumno es una estructura que contiene:

   Nombre
   Apellido
   Legajo
   DNI
   Fecha de nacimiento

7. Dado un archivo que contiene ítems de facturas, agrupados por número de factura, confeccione un corte de control que muestre por pantalla el total del importe de cada factura. Por ejemplo, teniendo:

```
struct FacturaItem
{
        int nrofactura;
        int nroitem;
        float precio;
};
```

   Y guardando en un archivo los siguientes datos:
   vec[0].nrofactura = 100; vec[0].precio = 700;
   vec[1].nrofactura = 100; vec[1].precio = 1200.5;
   vec[2].nrofactura = 102; vec[2].precio = 340.75;

```
vec[3].nrofactura = 102; vec[3].precio = 120.15;
vec[4].nrofactura = 102; vec[4].precio = 340.75;
vec[5].nrofactura = 201; vec[5].precio = 81.20;
vec[6].nrofactura = 201; vec[6].precio = 15.70;
fwrite(vec,sizeof(FacturaItem),7,f);
```

Una posible salida por pantalla de la ejecución del programa solicitado sería:

El total de la factura nro 100 es $1900.5

El total de la factura nro 102 es $801.65

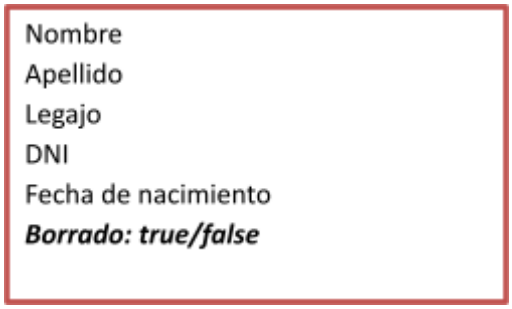
El total de la factura nro 201 es $96.9

8. Extender el programa del ejercicio 6 agregando un menú principal con las siguientes opciones:

- Agregar un alumno
- Listar todos los alumnos
- Buscar alumno por legajo
- Buscar alumno por DNI
- Ordenar archivo por legajo.
- Salir

Implemente la funcionalidad para cada opción en subprogramas.

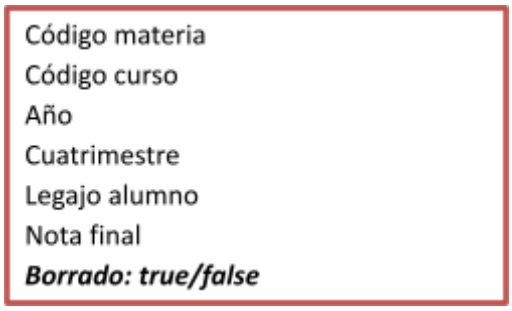
9. Agregar la funcionalidad de borrar un alumno. El borrado puede hacerse lógico, es decir poniendo una bandera en el registro que esté activo:



Recuerde agregar la opción borrar en el menú y actualizar las funciones de búsqueda y lista para que no muestren registros borrados.

¿Qué debería hacer si desea borrar físicamente del archivo un registro?

10. Se extiende el programa anterior con un manejo de archivo de cursadas. El programa debe permitir ingresar una cursada de un alumno a una materia utilizando la siguiente estructura:



Extienda el menú para poder ingresar la cursada de un alumno. Si el alumno no existe, el programa debe mostrar un mensaje de error “Alumno inexistente en archivo”.

Además debe también listar todas las cursadas de un alumno dado.