DOCUMENTACIÓN EN C - FINANCIERA - PERALTA MEZA ISMAEL - TP FINAL

El programa principal está desarrollado casi en su totalidad sobre una librería propia, la cual tiene todas las funciones necesarias para el procesamiento de datos (altas, bajas, modificaciones, búsquedas, listados). Trabajando sobre el archivo anterior que tenía el cliente "prestamos.csv" y guardando todos los procesos principales en el archivo binario "creditos.dat". Esta biblioteca se maneja con el main.c posee un switch que procesa la función de menú.
Las validaciones de ingresos por teclado se realizaron principalmente con ciclos while en algunos casos en particulares con funciones auxiliares (por ejemplo: contieneNumeros() - existe_registro()).

A continuación se dará detalles de las funciones más importantes del programa sobre la financiera: ***(Para información más exactos ver código con sus respectivos comentarios****)*

- **vista_datos_previos:**

Me permite listar el archivo "prestamos.csv" mediante funciones extraídas de string.h y stdio.h. Leo y separó en por campos el documento entre ";" para hacer una visualización correcta del mismo en la consola. Cree variables auxiliares para darles formato con atof() y así poder bajar los decimales a 2.

- **crear_archivo_binario:**

Crea el binario "créditos.dat", es la única función que lo crea, necesaria para las demás.

- **migrar_datos:**

Similar a la parte de lectura de "prestamos.csv", con la diferencia de que ahora tengo que pasar los campos leídos al archivo binario "creditos.dat" guardandolo en campos un formato determinado por mi "struct datos clientes". Uso funciones atoi() y atof() para asignar si son enteros o flotantes, strcpy para guardar las cadenas. Por otro lado según los requerimientos:

  - Al realizar la migración aparte de copiar los datos de CSV se le asignó un campo activo por defecto = 1
  - Se creó una función auxiliar obtener_nombre_mes() que según el número de mes leído devuelve la abreviación correspondiente en minúscula (esta función la use varias veces)
  - Se separó en nombre y apellido en columnas distintas usando el espacio en blanco entre ambos " "
  - Los apellidos y tipo de créditos pasaron a mayúscula usando la función strupr() de la librería string.h

- **vista_archivo_nuevo**:

Con esta función realizo impresiones en pantalla de las 4 opciones que tiene el usuario, construido en un switch. La reutilizo a lo largo del uso del programa ya que puedo visualizar a medida que se modifica algún registro .
En todos los casos se recorre de forma secuencial el archivo binario pero mostrando con condicionales lo que requiere el usuario, con las correctas validaciones para que no ingrese números distintos a los solicitados, además reconfirmar si desea hacer la acción con condicionales.

- **ingresar_clientes:**

Se realiza el ingreso de nuevo registros/clientes por teclado en el archivo binario, tiene que ingresar la posición u Orden al que pertenece en el registro el cual válido con la función aux existe_registro() de forma directa (fseek, para ubicar el puntero al principio del registro que necesito) y siendo también mayor a 0. Posterior se piden todos los datos que necesito para llenar el struct, los campos str como nombre y apellido uso contieneNumeros() para validar el campo sean todas letras. Hay campos que se completan con operaciones por ejemplo el IVA está seteado como #define IVA 0.21. Se realiza una previsualización de todos los datos

cargados y posterior un condicional que me sirve para saber si el usuario desea cargar o no el nuevo registro.

- **busqueda:**

Mediante un switch y 2 opciones que tiene el usuario para ingresar:

  - Case 1: Válido que el registro existe y es mayor que cero, para posteriormente realizar la busqueda de forma secuencial en el archivo binario recorriendo las posiciones obtenidas (NroReg) con ftell en un while con un i++.
  - Case 2: Idem anterior, pero esta vez uso strcmp == 0 para ver si el apellido ingresado es igual al que busco.

- **modificar:**

Mediante un switch y 2 opciones que tiene el usuario para ingresar, modifica campos ya cargados del binario.Se pide el número de posición que desea modificar (validando idem puntos anteriores) y mediante selección directa cambia los campos importe y tipoCred. Se hace una revisión previa de los datos, mostrando el registro completo y en ambos caso confirma el usuario si quiere modificar el cambio de X (campo nombre/apellido). Pisando los valores anteriores con scanf o strcpy para cadenas.

- **baja_logica:**

Cumple la función de modificar el campo activo que está por defecto en 1 a 0. El cliente elige mediante el número de orden el registro que se va a dar de baja, se realiza una pre-visualización para que posteriormente confirme si lo desea. Es de forma directa la búsqueda del registro

- **baja_fisica**

Por último esta función se encarga de migrar los datos activos = 1 a un nuevo binario temporario y los activos = 0 (los que se dieron de baja con baja_logica()) al nuevo archivo de texto clientes_bajas<fecha>.xyz. Posteriormente renombro los archivos temporarios a los definitivos como estaban originalmente y borro los anteriores todo con rename, remove. Hago una lectura final del nuevo archivo de texto.

<u>Algunas aclaraciones:</u>

- ❖ En la función vista_archivo_nuevo() va a figurar un case 5, el cual quise usar para listar el archivo de bajas.xyz pero no me funciono correctamente por más que nombrara correctamente al archivo en la línea del código N° 693. Pude emitir bien después de crear el archivo en la misma función de baja_fisica cumpliendo con el punto 13 del tp. (me queda pendiente a mejorar pero no llego con el tiempo).

- ❖ Me falto hacer alguna función que valide cuando el usuario ingresa una letra donde tendría que ir un número. Lo dejé para más adelante y se me complicaron otras cosas, lo dejo pendiente para mejorar. Probe con una función de la librería math.h (isnan()) pero no dio resultado, deje documentado el intento. Voy a internar con isdigit de ctype.h más adelante

- ❖ Cuando ingresó los años y leí la consigna de que se tiene que tomar de referencia el año más chico del csv y el año actual como rango intérprete 2022 a 2023, si es incorrecto por favor informar para modificar.

- Otra mejora pendiente es la de funcion baja_fisica(), todo funciona perfectamente siempre y cuando el usuario no cree un nuevo archivo de baja el mismo día, eso me desconfigura todas las posiciones, lo voy a mejorar.

- Última aclaración, probe muchas veces el archivo, borrando y creando nuevos. En ciertos momentos el archivo binario no se creaba con 0 kb (vacío), sino que se guardaba basura y cuando hacia la migración se migran bien los datos pero quedaba la basura (si lo borraba y volvía a crear estaba bien) no se si es por mi maquina pero de 20 pruebas 3 me paso lo mismo en diferentes momentos. Por favor tener en cuenta en caso de que suceda. Gracias