**Instituto Tecnológico y de Estudios Superiores de Occidente**

Departamento de Electrónica, Sistemas e Informática

Ingeniería en Sistemas Computacionales



PROGRAMACIÓN CON MEMORIA DINÁMICA

Tarea 2. Memoria Dinámica y Archivos

Autor: Paterno Materno, Nombre

12 de junio de 2018. Tlaquepaque, Jalisco,

Presentación: 5 pts.

Funcionalidad: 40 pts.

Pruebas: 20 pts.

- Falta describir las pruebas (escenario, y resultados de la experimentación).

- Tienes algunos detalles en el tema de funcionalidad, modificar y eliminar datos en archivos.

**Instrucciónes para entrega de tarea**

Esta tarea, como el resto, es ***IMPRESCINDIBLE*** entregar los entregables de esta actividad de la siguiente manera:

* **Reporte:** vía *moodle* en **un archivo PDF**.
* **Código:** vía su repositorio **Github**.

La evaluación de la tarea comprende:

* 10% para la presentación
* 60% para la funcionalidad
* 30% para las pruebas

Es necesario responder el apartado de conclusiones, pero no se trata de llenarlo con paja. Si no se aprendió nada al hacer la práctica, es preferible escribir eso. Si el apartado queda vacío, se restarán puntos al porcentaje de presentación.

**Objetivo de la actividad**

El objetivo de la tarea es que el alumno aplique los conocimientos y habilidades adquiridos en el tema de manejo de memoria dinámica y archivos utilizando el lenguaje ANSI C.

**Descripción del problema**

Ahora tienes los conocimientos para enfrentarte a un nuevo proyecto llamado **MyDB**. En este proyecto vas a recrear una parte de un sistema de transacciones bancarias. Para esto vas a requerir del uso de:

* Estructuras
* Funciones y paso de parámetros
* Apuntadores
* Memoria Dinámica
* Archivos binarios

El sistema **MyDB** al ser ejecutado deberá mostrar al usuario una interfaz con el siguiente menú principal:

**<< Sistema MyDB >>**

1. Clientes
2. Cuentas
3. Transacciones
4. Salir

El sistema **MyDB** debe realizar automáticamente, las siguientes operaciones:

1. Si el sistema **MyDB** se ejecutó por primera vez, este deberá crear tres archivos binarios: **clientes.dat**, **cuentas.dat** y **transacciones.dat**. Para esto el sistema debe solicitar al usuario indicar la **ruta de acceso** (por ejemplo, c:\\carpeta\\) en donde se desea crear los archivos (esta información deberá ser almacenada en un archivo de texto llamado **mydb.sys**).

**Clientes**

La opción **Clientes** debe mostrar un submenú con las siguientes opciones:

* **Nuevo** cliente Registra los datos de un nuevo cliente del banco
* **Buscar** cliente Permite consultar la información de un usuario a partir de su

id\_cliente.

* **Eliminar** cliente Si existe, elimina un usuario deseado del sistema. Esto implica que

deben Borrarse las cuentas registradas a nombre del usuario

(utilice id\_usuario para buscar).

* **Imprimir** clientes Imprime la información de todos los clientes registrados en el

sistema.

La información que el sistema requiere almacenar sobre cada cliente es la siguiente:

* Id\_usuario (es un número entero que se genera de manera consecutiva, clave única)
* Nombre
* Apellido materno
* Apellido paternoid
* Fecha de nacimiento (tipo de dato estructurado: dd/mm/aaaa)

Para gestionar la información de los clientes, defina un tipo de dato estructurado llamado **Usuario**, utilice instancias de Usuario para capturar la información desde el teclado y posteriormente guardarlo en el archivo usuario.dat.

Un ejemplo del contenido que se estará almacenando en el archivo **usuario.dat** es el siguiente:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **id\_usuario** | **nombre** | **apellido\_paterno** | **apellido\_materno** | **fecha\_nacimiento** |
| 1 | Ricardo | Perez | Perez | {3,10, 2010} |
| 2 | Luis | Rodriguez | Mejía | {2,7, 2005} |
| 3 | Gabriela | Martínez | Aguilar | {7,11,2015} |

**Importante:** considere que no pueden existir datos **id\_usuario** repetidos y que es un valor autonúmerico. Adicionalmente, recuerde que al inicio el archivo no tendrá datos.

**Cuentas**

La opción **Cuentas** debe mostrar un submenú con las siguientes opciones:

* **Nueva** cuenta Registra una cuenta nueva a nombre de un usuario, utilice

**id\_cliente** para relacionar el usuario y la cuenta. Antes de

crear la nueva cuenta se debe verificar que el usuario exista en el

sistema. Adicionalmente, se debe indicar el saldo con el que se

abre la cuenta. Por ejemplo; $1000.

* **Buscar** cuenta Permite consultar en pantalla la información de una cuenta en el

sistema a partir de su **id\_cuenta**. En pantalla debe mostrarse:

**id\_cuenta**, **nombre de cliente**, **saldo de la cuenta**.

* **Eliminar** cuenta Si existe, elimina la cuenta deseada en el sistema.
* **Imprimir** cuentas Imprime la información de todas las cuentas registradas en el

sistema. En pantalla debe mostrarse un listado con la siguiente información de las cuentas: **id\_cuenta**, **nombre de cliente**, **saldo de la cuenta**.

La información que el sistema requiere almacenar sobre cada cuenta es la siguiente:

* id\_cuenta (es un número entero que se genera de manera consecutiva, clave única)
* id\_usuario (indica a quien pertenece la cuenta)
* Saldo
* Fecha de apertura (tipo de dato estructurado: dd/mm/aaaa)

Para gestionar la información de las cuentas, defina un tipo de dato estructurado llamado **Cuenta**, utilice instancias de **Cuenta** para capturar la información desde el teclado y posteriormente guardarlo en el archivo **cuenta.dat**.

Un ejemplo del contenido que se estará almacenando en el archivo **cuenta.dat** es el siguiente:

|  |  |  |  |
| --- | --- | --- | --- |
| **id\_cuenta** | **Id\_usuario** | **Saldo** | **fecha\_apertura** |
| 1 | 1 | Perez | {12,6, 2018} |
| 2 | 2 | Rodriguez | {2,7, 2018} |
| 3 | 1 | Martínez | {7,3,2018} |

**Importante:** considere que no pueden existir valores de **id\_cuenta** repetidos y que es un valor autonúmerico. Adicionalmente, observe que un usuario puede tener más de una cuenta.

**Transacciones**

La opción **Transacciones** debe mostrar un submenú con las siguientes opciones:

* **Depósito** Permite incrementar el saldo de la cuenta, para esto el sistema

requiere: **id\_cuenta**, **monto a depositar** (valide que la cuenta exista).

* **Retiro** Permite a un cliente disponer del dinero que tiene una cuenta

bancaria. Para esto el sistema requiere: **id\_cuenta**, **monto a retirar**

(valide que la cuenta existe y que tiene fondos suficientes).

* **Transferencia** Permite a un cliente transferir dinero de una cuenta origen a una

cuenta destino. Para esto el sistema requiere: **id\_cuenta origen**,

**id\_cuenta destino**, **monto a transferir** (valide que existan ambas

cuentas y que la cuenta origen tiene fondos suficientes).

La información que el sistema requiere almacenar sobre cada transacción es la siguiente:

* id\_transacción (es un número entero que se genera de manera consecutiva, no se puede repetir)
* Tipo de operación (depósito, retiro, transferencia)
* Cuenta origen
* Cuenta destino (se utiliza para las operaciones de transferencia, en otro caso, NULL)
* Fecha de la transacción
* Monto de la transacción

Para gestionar la información de las trasferencias, defina un tipo de dato estructurado llamado **Transferencia**, utilice instancias de Transferencia para capturar la información desde el teclado y posteriormente guardarlo en el archivo transferencia.dat.

Un ejemplo del contenido que se estará almacenando en el archivo **transferencia.dat** es el siguiente:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **id\_transaccion** | **tipo\_transaccion** | **Id\_cuenta\_origen** | **Id\_cuenta\_destino** | **fecha\_transaccion** | **monto\_transaccion** |
| 1 | Retiro | 1 | Null | {12,6, 2018} | $100 |
| 2 | Deposito | 2 | Null | {12,6, 2018} | $5000 |
| 3 | Transferencia | 2 | 1 | {12,6,2018} | $1500 |

**Importante:** considere que no pueden existir datos **id\_transaccion** repetidos y que es un valor autonúmerico. Adicionalmente, recuerde que al inicio el archivo no tendrá datos y que los saldos de las cuentas deberán afectarse por las transacciones realizadas.

Solución del alumno, pruebas y Conclusiones

Código fuente

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

FILE \*fd,\*fd2,\*fd3,\*fd4;

typedef struct{

int dd;

int mm;

int aaaa;

}fecha;

typedef struct{

int idUsuario;

char nombre[20];

char apellidop[20];

char apellidom[20];

fecha nacimiento;

}Usuario;pusuario

typedef struct{

int dd;

int mm;

int aaa;

}FechaApertura;

typedef struct{

int cuenta;

int usuario;

char saldo[20];

fechaApertura FA;

}Cuenta;

int main(void){

int opcion;

char ruta[100];

char rutaUsuario[100];

char rutaCuentas[100];

char rutaTransacciones[100];

char rutamydb[100];

printf("Ingresar ruta de acceso\n");

gets(ruta);

// Asignacion de ruta de Usuario

strcpy(ruta,strcat(ruta, "\\"));

strcpy(rutaUsuario,ruta);

strcpy(rutaCuentas,ruta);

strcpy(rutaTransacciones,ruta);

strcpy(rutamydb,ruta);

strcat(rutaUsuario, "\\usuario.dat");

// Asignacion ruta Cuentas

strcat(rutaCuentas, "\\cuentas.dat");

// Asigancion de ruta Transacciones

strcat(rutaTransacciones, "\\transacciones.dat");

strcat(rutamydb, "\\mydb.sys");

puts(rutaUsuario);

puts(rutaCuentas);

puts(rutaTransacciones);

puts(rutamydb);

// Verificacion de archivos

fd =fopen(rutaUsuario,"rb");

fd2=fopen(rutaCuentas,"rb");

fd3=fopen(rutaTransacciones,"rb");

fd4=fopen(rutamydb,"rb");

if(fd==NULL){

printf("Arvhivo Usuario no existe\n");

printf("Creando archivo..\n");

fd =fopen(rutaUsuario,"wb");

}

else {

printf("Se encontro el archivo\n");

printf("Ubicacion: %s",rutaUsuario);

}

if(fd2==NULL){

printf("Arvhivo Cuentas no existe\n");

printf("Creando archivo..\n");

fd2 =fopen(rutaCuentas,"wb");

}

else {

printf("Se encontro el archivo\n");

printf("Ubicacion: %s",rutaCuentas);

}

if(fd3==NULL){

printf("Arvhivo Transacciones no existe\n");

printf("Creando archivo..\n");

fd3 =fopen(rutaTransacciones,"wb");

}

else {

printf("Se encontro el archivo\n");

printf("Ubicacion: %s",rutaTransacciones);

}

if(fd4==NULL){

printf("Arvhivo RUTAS no existe\n");

printf("Creando archivo..\n");

fd3 =fopen(rutaTransacciones,"wb");

}

else {

printf("Se encontro el archivo\n");

printf("Ubicacion: %s",rutaTransacciones);

}

do{

printf("\n<< Sistema MyDB >>\n");

printf("1. lientes\n");

printf("2. Cuentas\n");

printf("3. Transacciones\n");

printf("4. Salir \n");

printf("\n");

scanf("%i",&opcion);

switch(opcion){

int opcion2;

case 1:

system ("cls");

printf("Nuevo Cliente\n");

printf("Buscar Cliente\n");

printf("Eliminar Client\n");

printf("Imprimir Cliente \n");

printf("\n");

scanf("%i",&opcion2);

switch(opcion2){

case 1:

fprintf(fd,"id\_Usuario\t\t\tnombre\t\t\tapellido\_paterno\t\t\tapellido\_materno\t\t\tFecha nacimiento\n\n");

printf("ID Usuario\n");

gets(pusuario.idUsuario);

printf("Nombre\n");

gets(pusuario.nombre);

printf("Apellido paterno\n");

gets(pusuario.apellidop);

printf("Apellido Materno\n");

gets(pusuario.apellidom);

printf("Fecha de nacimiento\n");

gets(pusuario.nacimiento.dd);

gets(pusuario.nacimiento.mm);

gets(pusuario.nacimiento.aaaa);

fwrite(pusuario.idUsuario,1,strlen(pusuario.idUsuario),fd);

printf("\t\t\t");

fwrite(pusuario.nombre,1,strlen(pusuario.nombre),fd);

printf("\t\t\t");

fwrite(pusuario.apellidop,1,strlen(pusuario.apellidop),fd);

printf("\t\t\t");

fwrite(pusuario.apellidom,1,strlen(pusuario.apellidom),fd);

printf("\t\t\t");

fwrite(pusuario.nacimiento.dd,1,strlen(pusuario.nacimiento.dd),fd);

printf(",");

fwrite(pusuario.nacimiento.mm,1,strlen(pusuario.nacimiento.mm),fd);

printf(",");

fwrite(pusuario.nacimiento.aaaa,1,strlen(pusuario.nacimiento.aaaa),fd);

printf("\n");

break;

case 2:

break;

case 3:

break;

case 4:

int c;

while ((c=fgetc(fd))!=EOF){

if(c=='\n')

printf("\n");

}

break;

case 5:

break;

}

break;

case 2:

printf("Nueva Cuenta\n");

printf("Buscar Cuenta\n");

printf("Eliminar Cuenta\ne");

printf("Imprimir cuentas \n");

printf("\n");

scanf("%i",&opcion2);

switch(opcion2){

case 1:

break;

case 2:

break;

case 3:

break;

case 4:

break;

case 5:

break;

}

break;

case 3:

printf("Deposito\n");

printf("Retiro\n");

printf("Transeferencia\n");

printf("\n");

scanf("%i",&opcion2);

switch(opcion2){

case 1:

break;

case 2:

break;

case 3:

break;

case 4:

break;

}

break;

}

printf("hola");

}while(opcion!=4);

return 0;

}

Ejecución

Conclusiones (obligatorio):

* Lo que aprendí́ con esta práctica fue el manejo de archivos en c . Lo que ya sabía era la forma de desplegar el contenido de un archivo en la consola.
* Lo que me costó́ trabajo fue la búsqueda especifica de información en un archivo y como lo solucioné.
* Lo que no pude solucionar fue el remplazo de información en el archivo.