ACTIVIDAD 3 REGISTROS DE LONGITUD FIJA.

Orlando Agustín Barba Palacios

ESTRUCTURAS DE DATOS 2 SECCION D06

1. Planteamiento del problema.

Diseñar una solución informática que demuestre la comprensión y uso de ficheros para el almacenamiento en memoria secundaria de más de un registro de longitud fija del mismo tipo, sin necesidad de crear una lista en memoria principal y volcar la lista completa al fichero. La estrategia de almacenamiento es registros de longitud fija.

El sistema debe considerar el almacenamiento de objetos de una clase A, los objetos de la clase A deben contar con atributos de tipo: cadena (de hasta 20 caracteres con o sin espacios), entero (llave primaria), un contador de objetos de clase B y un arreglo de hasta 5 objetos de clase B que contiene atributos de tipo double y boolean. Las modificaciones realizadas en un registro específico deben sobrescribir solamente los bytes correspondientes al objeto montado a memoria principal y modificado.

2. Objetivos:

1. Manipulación de archivos de texto plano.
2. Simular las acciones de una lista utilizando un archivo binario para el almacenamiento de registros de longitud fija.
3. Comprender el almacenamiento de datos en bytes en un archivo binario.

3. Marco teórico.

Para esta actividad se requiere tener un paradigma de programación orientada a objetos y dominar la programación estructurada, así como el funcionamiento y la aplicación de los ciclos While y For.

La comprensión de conceptos como bit, bytes, archivos binarios (.bin). Comprender el

funcionamiento de una lista para hacer uso de sus métodos de manera correcta.

4. Desarrollo.

Introducción:

En esta actividad se nos pidió almacenar registros en un archivo binario, a diferencia de la actividad anterior que fue con registros de longitud fija ahora son de longitud fija lo cual hace más fácil la lectura y escritura de un registro en el archivo, sin mencionar que hace más intuitivo el proceso de programación.

Parte 1. Insertar y validar llave primaria.

La inserción es bastante simple, primero añadimos los datos a nuestro registro en memoria principal para posteriormente insertarlo, primero quisiera explicar como se hace la verificación de unicidad de la llave ya que es un aspecto muy importante al momento de añadir un nuevo registro.

Como podemos ver en el apéndice 4 se hace una lectura de cada registro del archivo, cada que lee un registro lo carga a memoria principal para luego hacer la comparación entre la llave del registro y la nueva llave que se quiere usar, si coincide significa que la nueva llave

no se puede usar porque ya existe, si no coincide lee otro registro y así lo sigue repitiendo hasta que termine de leer el archivo o encuentre una coincidencia, si no encuentra coincidencia significa que se puede usar esa nueva llave para un nuevo registro.

Para insertar tenemos insertar al inicio (apéndice 2) e insertar al final (apéndice 1)

Lo que hice fue en el caso de inserción al final abrir el archivo en modo binario y app para escribir al final del archivo sin borrar lo que tenia previamente y para insertar al inicio guarde todo el archivo previo a la inserción en un string para después insertar mi nuevo registro solo en modo de apertura bin para que se insertara desde cero y al inicio para después insertar ese string con los datos previos y de esta forma tener los mismos registros pero ahora con un registro nuevo al inicio.

Parte 2 Mostrar registros.

En esta ocasión tenemos 2 formas de mostrar los registros, primero explicare el método MostrarUNO apéndice 7, para este método pedimos la llave del registro a mostrar, para mostrarlo deberemos buscarlo por lo que pasaremos esa llave a el método buscar que podemos ver en el apéndice 3 que funciona de manera muy similar al verificar llave ya que lee todos los registros en busca de el registro, pero, si se encuentra además de cargar a memoria el registro nos devuelve la posición del bite en donde comienza ese registro en el archivo, este dato nos será útil para otro método pero de momento nos conformamos con cargar en memoria el registro a mostrar. Una vez encontrado el registro simplemente se muestra y listo, para mostrar todos los registros es igual solo que imprime todos los registros en vez de que solo imprima el que queremos buscar.

Parte 3 Eliminar

Para eliminar opte por un archivo secundario con los registros actuales previos a la eliminación, para esto use el método llenar secundario véase apéndice 8, una vez con ese registro secundario empecé en el método de eliminación que consiste en pedir la llave del registro a eliminar para posteriormente ir leyendo el archivo secundario e ir comparando sus registros, si no había coincidencia los añadía al archivo principal pero si se encontraba con el registro a eliminar simplemente no lo escribe para que en la nueva version del archivo principal no este ese registro.

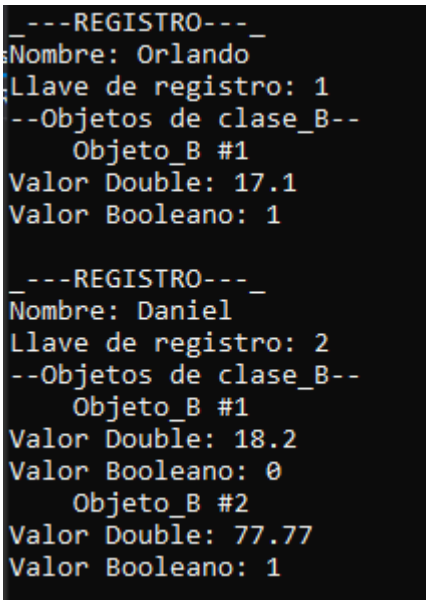
Parte 4 Modificar

Para modificar un registro(apéndice 6) pido la llave del registro a modificar para después pasarlo por el método de búsqueda(apéndice 3) que recordemos que nos carga a memoria principal el registro a buscar y nos regresa su posición en el registro, una vez que tenemos en memoria principal el registro le hacemos las modificaciones necesarias y ya que esta listo se escribe en el archivo pero usando el método de escriture bin y app, para no perder datos del registro y situando nuestro puntero del archivo en la posición del registro que habíamos buscado previamente para así sobrescribir los datos de el registro

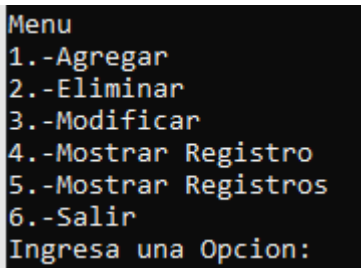
viejo por el nuevo y ya modificado, esto no ocasiona conflicto ya que recordemos que al ser registros de longitud fija sus dimensiones son las mismas en bites y por tanto no hay perdida o alteración alguna en los otros registros.

5. Pruebas y resultados.

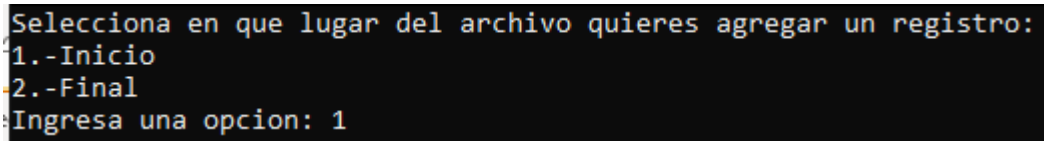
Para realizar las pruebas se debe saber que tenemos en el archivo ya 2 registros:



Bien, ahora trabajaremos sobre los datos, tenemos el siguiente menú:



Vamos a insertar al inicio un registro de nombre pepe con llave 4 y 2 objetos B uno con un booleano verdadero y otro falso:



```
Ingresa La Llave del objeto A:
4
Ingresa La cadena:
pepe
Cuantos objetos de clase B quieres?:
2
Objeto B# 1
Ingresa Double:
21.2
Ingresa booleano 1 o 0:
1
Objeto B# 2
Ingresa Double:
34.56
Ingresa booleano 1 o 0:
0
```

Ahora mostremos los registros para comprobar:

```
_---REGISTRO---_
Nombre: pepe
Llave de registro: 4
--Objetos de clase_B--
    Objeto_B #1
Valor Double: 21.2
Valor Booleano: 1
    Objeto_B #2
Valor Double: 34.56
Valor Booleano: 0

_---REGISTRO---_
Nombre: Orlando
Llave de registro: 1
--Objetos de clase_B--
    Objeto_B #1
Valor Double: 17.1
Valor Booleano: 1

_---REGISTRO---_
Nombre: Daniel
Llave de registro: 2
--Objetos de clase_B--
    Objeto_B #1
Valor Double: 18.2
Valor Booleano: 0
    Objeto_B #2
Valor Double: 77.77
Valor Booleano: 1
```

Ahora insertemos al final un registro de nombre Israel con llave 3 y objetos B con booleano verdadero:

```
Ingresa La Llave del objeto A:
3
Ingresa La cadena:
Israel
Cuantos objetos de clase B quieres?:
1
Objeto B# 1
Ingresa Double:
39.444
Ingresa booleano 1 o 0:
1
```

Ahora mostremos los registros para comprobar:

```
_---REGISTRO---_
Nombre: pepe
Llave de registro: 4
--Objetos de clase_B--
    Objeto_B #1
Valor Double: 21.2
Valor Booleano: 1
    Objeto_B #2
Valor Double: 34.56
Valor Booleano: 0

_---REGISTRO---_
Nombre: Orlando
Llave de registro: 1
--Objetos de clase_B--
    Objeto_B #1
Valor Double: 17.1
Valor Booleano: 1

_---REGISTRO---_
Nombre: Daniel
Llave de registro: 2
--Objetos de clase_B--
    Objeto_B #1
Valor Double: 18.2
Valor Booleano: 0
    Objeto_B #2
Valor Double: 77.77
Valor Booleano: 1

_---REGISTRO---_
Nombre: Israel
Llave de registro: 3
--Objetos de clase_B--
    Objeto_B #1
Valor Double: 39.444
Valor Booleano: 1
```

Ahora eliminaremos el registro de pepe de llave 4:

```
Ingresa la llave del registro a eliminar:
4
```

Ahora mostremos los registros para comprobar:

```
_---REGISTRO---_
Nombre: Orlando
Llave de registro: 1
--Objetos de clase_B--
    Objeto_B #1
Valor Double: 17.1
Valor Booleano: 1

_---REGISTRO---_
Nombre: Daniel
Llave de registro: 2
--Objetos de clase_B--
    Objeto_B #1
Valor Double: 18.2
Valor Booleano: 0
    Objeto_B #2
Valor Double: 77.77
Valor Booleano: 1

_---REGISTRO---_
Nombre: Israel
Llave de registro: 3
--Objetos de clase_B--
    Objeto_B #1
Valor Double: 39.444
Valor Booleano: 1
```

Ahora modificaremos por completo el registro de llave 3 modificando primero el nombre cambiándolo a Gabriel:

```
Ingresa la llave del registro a modificar:
3
Se encontro el registro a Modificar
Que deseas modificar?
1.-Nombre
2.-Llave
3.-Agregar objeto B
Ingresa una opcion:
1
Ingresa La nueva cadena:
Gabriel
```

También añadiré un objeto de clase B más:

```
Ingresa la llave del registro a modificar:
3
Se encontro el registro a Modificar
Que deseas modificar?
1.-Nombre
2.-Llave
3.-Agregar objeto B
Ingresa una opcion:
3
Objeto B# 2
Ingresa Double:
66.77
Ingresa booleano 1 o 0:
1
```

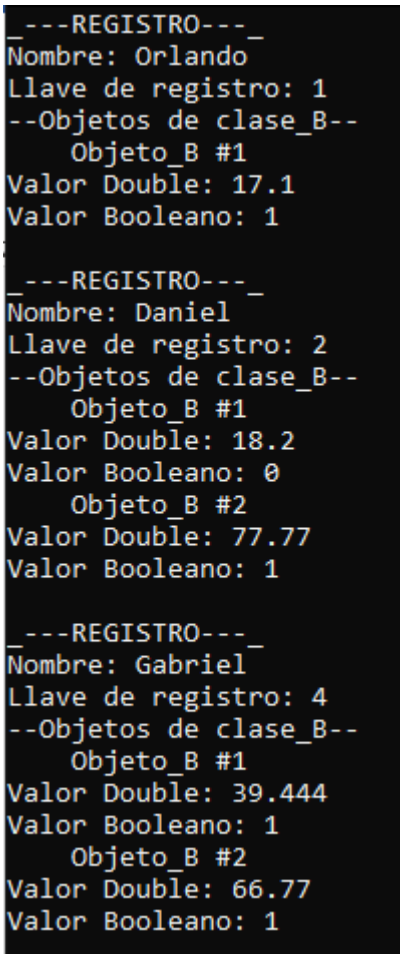
Ahora mostremos los registros para comprobar, pero ahora solamente buscare un solo registro que será el registro con llave 3:

```
Ingresa la llave del registro a mostrar:
3
_---REGISTRO---_
Nombre: Gabriel
Llave de registro: 3
--Objetos de clase_B--
    Objeto_B #1
Valor Double: 39.444
Valor Booleano: 1
    Objeto_B #2
Valor Double: 66.77
Valor Booleano: 1
```

Ahora cambiare su llave a 4:

```
Ingresa la llave del registro a modificar:
3
Se encontro el registro a Modificar
Que deseas modificar?
1.-Nombre
2.-Llave
3.-Agregar objeto B
Ingresa una opcion:
2
Ingresa la llave del objeto de clase A (debe ser nueva)
1
Ingresa la llave del objeto de clase A (debe ser nueva)
2
Ingresa la llave del objeto de clase A (debe ser nueva)
3
Ingresa la llave del objeto de clase A (debe ser nueva)
4
```

Podemos ver que intente usar las llaves 1,2 y 3 pero no me dejo ya que ya existen y por lo tanto no me dejaría utilizar esas llaves, pero al final me dejo la llave 4 ya que no estaba en algún registro existente, ahora mostrare los registros y comprobare el cambio:



7. Apéndice(s).

1.- Insertar Adelante:

```
void Manejador_De_Archivo::insertarFinal()
{
    CrearObjetoA();
    ofstream fout("fichero.bin", ios::binary | ios::app);
    fout.write((char*)&obj, sizeof(Clase_A));
}
```

2.- Insertar Inicio

```
void Manejador_De_Archivo::insertarInicio()
```

```
{
    ifstream fin("fichero.bin", ios::binary);
    string Archivo;
    fin >> Archivo;
    fin.close();
    CrearObjetoA();
    ofstream fout("fichero.bin", ios::binary);
    fout.write((char*)&obj, sizeof(Clase_A));
    fout << Archivo;
}
```

3.- Buscar

```
int Manejador_De_Archivo::Buscar(int key)
{
    ifstream fin("fichero.bin", ios::binary);
    int fileIndex = 0;
    fin.read((char*)&obj, sizeof(Clase_A));
    while (!fin.eof()) {
        if (obj.DarLlave() == key) {
            return fileIndex;
        }
        fileIndex += sizeof(Clase_A);
        fileIndex = fin.tellg();
        fin.read((char*)&obj, sizeof(Clase_A));
    }
    fin.close();
    return -1;
}
```

4.- verificar Llave

```
bool Manejador_De_Archivo::VerificarLlave(int key)
{
    ifstream fin("fichero.bin", ios::binary);
```

```
    int fileIndex = 0;
    fin.read((char*)&obj, sizeof(Clase_A));
    while (!fin.eof()) {
        if (obj.DarLlave() == key) {
            return false;
        }
        fileIndex += sizeof(Clase_A);
        fileIndex = fin.tellg();
        fin.read((char*)&obj, sizeof(Clase_A));
    }
    fin.close();
    return true;
}
```

5.- Eliminar

```
void Manejador_De_Archivo::Eliminar()
{
    LlenarSecundario();
    int key;
    cout << "Ingresa la llave del registro a eliminar:" << endl;
    cin >> key;
    ifstream fin("Secundario.bin", ios::binary);
    int fileIndex = 0;
    fin.read((char*)&obj, sizeof(Clase_A));
    ofstream fout("fichero.bin", ios::binary);
    while (!fin.eof()) {
        if (obj.DarLlave() != key) {
            fout.write((char*)&obj, sizeof(Clase_A));
        }
        fileIndex += sizeof(Clase_A);
        fileIndex = fin.tellg();
        fin.read((char*)&obj, sizeof(Clase_A));
```

```
    }
    fout.close();
    fin.close();
}
```

6.- Modificar

```
void Manejador_De_Archivo::Modificar()
{
    int key;
    int fileindex;
    cout << "Ingresa la llave del registro a modificar:" << endl;
    cin >> key;
    fileindex = Buscar(key);
    if (fileindex == -1) {
        cout << "No se encontro el registro" << endl;
        return;
    }
    MenuModificar();
    ofstream fout("fichero.bin", ios::binary | ios::in);
    fout.seekp(fileindex, ios::beg);
    fout.write((char*)&obj, sizeof(Clase_A));
}
```

7.- MostrarUNO

```
void Manejador_De_Archivo::MostrarUNO()
{
    int key;
    cout << "Ingresa la llave del registro a mostrar:" << endl;
    cin >> key;
    if (Buscar(key) != -1) {
        cout << "_---REGISTRO---_" << endl;
        obj.Mostrar();
    }
```

```
    else {
        cout << "No se encontro Registro" << endl;
    }
}
```

8.- llenar secundario

```
void Manejador_De_Archivo::LlenarSecundario()
{
    ifstream fin("fichero.bin", ios::binary);
    string Archivo;
    fin >> Archivo;
    fin.close();
    ofstream fout("Secundario.bin", ios::binary);
    fout << Archivo;
}
```