ACTIVIDAD 2. CAMPOS DE DIMENSIÓN.

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 objeto del mismo tipo, sin necesidad de crear una lista en memoria principal. La estrategia de almacenamiento es campos de dimensión para registros de longitud variable.

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 (string), entero (llave primaria) y un arreglo de hasta 10 objetos de clase B (objetos almacenados NO modificables) que contiene atributos de tipo string y booleano.

2. Objetivos:

1. Manipulación de archivos binarios.
2. Simular las acciones de una lista utilizando un archivo para el almacenamiento de registros de longitud variable.
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:

Para comenzar con el desarrollo de esta actividad debo mencionar los nombres de los atributos de mi clase A o de mis registros.

En mi caso mi atributo de tipo string lo llamé Nombre, mi llave primaria que es de tipo entero la llamé Llave y finalmente mi arreglo de objetos de clase B lo llamé ObjB, a los objetos de clase B a almacenar cuyos atributos de tipo string y booleano los llame Cadena para el string y Booleano para el bool.

Como un objeto de clase A puede tener hasta 10 objetos B aproveche el atributo booleano de los B para saber cuantos objetos B hay en un A, ya que los A no tienen un contador para saber cuántos B tienen.

Al crear un objeto A cree los 10 objetos B pero con el valor booleano en false, esto lo hice para que cuando el usuario decida cuantos B quiere en A solo la cantidad seleccionada cambien a true, este true significara que son los objetos B que serán tomados en cuenta.

Ejemplo: el usuario quiere 2 objetos B en su objeto A entonces la posición 0 y 1 del arreglo de 10 objetos B su valor booleano será true.

Parte 1. Insertar.

Para poder explicar la inserción primero explicare mi proceso para crear un objeto de clase A (incluyendo sus elementos clase B) y el tratamiento de sus datos para posteriormente ser mandados al archivo binario.

En esta practica hice un buen uso del Objeto de clase A que podemos tener en memoria principal, ya que cuando creaba el objeto lo hacia en la memoria principal, para esto véase el **apéndice 5** Crear Objeto, este pide al usuario dar un nombre y llave al objeto de clase A , además de pedir cuantos objetos de clase B tendrá, obviamente para esto debe hacer algunos procedimientos previos para evitar errores, el primero es reiniciar los valores del objeto de memoria principal para evitar errores para esto véase el **apéndice 4**, también tenemos el método de validar la unicidad de la llave del registro, para esto se necesita previamente copear el contenido del archivo principal a uno secundario ya que es el que usa el método de verificación de unicidad de la llave véase **apéndice 2 y 3.**

La mayoría de los métodos usados en el programa que tienen que revisar el contenido de los registros dentro del archivo utilizan un método llamado cargar registro, véase **apéndice 1**, estos métodos leen el archivo principal y lo almacenan en un string, este string contiene todo el archivo binario por lo que facilita su manejo, se le pasa esta cadena y un índice a el método crear registro para que este lea un registro de la cadena y lo cargue a memoria principal para posterior mente devolver un índice correspondiente al siguiente registro o el final del archivo, el registro cargado a memoria se utilizara para realizar operaciones necesarias, en el caso de verificación de la unicidad de la llave es comparar la llave del registro en memoria principal por la propuesta por el usuario.

Una vez creado el objeto a insertar pasamos a llamar a los métodos insertar al inicio o final, véase **apéndice 6 y 7**, estos lo que hacen es llenar en un string el contenido previo a la inserción del archivo, esto para añadirlo en la nueva version del archivo, pero con el nuevo elemento.

Por ejemplo, para insertar al inicio se inserta primero el registro de memoria principal y posterior mente el string con los otros registros que previamente estaban en el archivo.

Para insertar al final es igual pero primero se inserta el string de los archivos y luego el registro de memoria principal.

Parte 2 Mostrar registros.

Para mostrar un registro es un proceso simple, haciendo uso del método buscar registro véase **apéndice 10** pedimos al usuario la llave del registro a mostrar, ahora pasamos el contenido del archivo principal a un string que mandaremos al método crear registro para que este carge a memoria el primer registro, posterior mente compara si las llaves coinciden, si no lo hacen se vuelve a llamar cargar registro y leerá el siguiente registro del string del archivo, realizara esto hasta que encuentre coincidencia con la llave, cuando lo encuentre simplemente mostrara el contenido de los atributos del objeto de clase A.

Para mostrar todos los registros el proceso es igual, pero a diferencia de mostrar un solo registro este mostrara todos los registros sin excepción, ya que el método cargar registro se ejecutará hasta que termine el string del archivo que serian todos los registros del archivo principal.

Parte 3 Eliminar.

Para eliminar utilizamos el método con el mismo nombre, véase el **apéndice 8**, tiene la misma estructura que el de mostrar registro (**apéndice 10**) solo que en vez de mostrar omite su escritura.

Lo que hace es cargar el archivo actual principal y lo va escribiendo nuevamente ingresando todos los registros exceptuando el que queremos eliminar, haciendo que su eliminación se haga correctamente.

Parte 4 Modificar

Para modificar un registro se hizo lo mismo que en el apéndice 8 de eliminación solo que en vez de omitir el registro este se edita/ modifica y ya que esta modificado se ingresa en la misma posición que tenia antes de su modificación, puede ver el método modificar en el **apéndice 9**.

5. Pruebas y resultados.

Empezaremos teniendo en mente que hay ya 2 registros en el archivo binario:

```
Mostrando Registros
.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.
Registro con Llave# 1
Nombre: Orlando
ObjetoB# 1 Nombre: Mate
ObjetoB# 2 Nombre: Progra
.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.
Registro con Llave# 2
Nombre: Oscar
ObjetoB# 1 Nombre: Historia
.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.
```

Procedemos entonces a ir al menú principal del programa y ver sus opciones

```
Menu
1.-Agregar
2.-Eliminar
3.-Modificar
4.-Mostrar Registro
5.-Mostrar Registros
6.-Salir
Ingresa una Opcion:
```

Vamos a la opción agregar, la cual nos dirá si queremos insertar al inicio o final:

```
Selecciona en que lugar del archivo quieres agregar un registro:
1.-Inicio
2.-Final
Ingresa una opcion:
```

Nosotros ingresaremos al finar un registro llamado Israel de llave 3 con 1 objeto B llamado Espanol:

```
Ingresa la llave del objeto de clase A (debe ser nueva)
3
Ingresa el nombre del objeto de clase A a crear
Israel
Cuantos objetos Clase B tendra? max 10
1
Objeto B numero: 1
Ingresa el nombre del objeto de clase B
Espanol
```

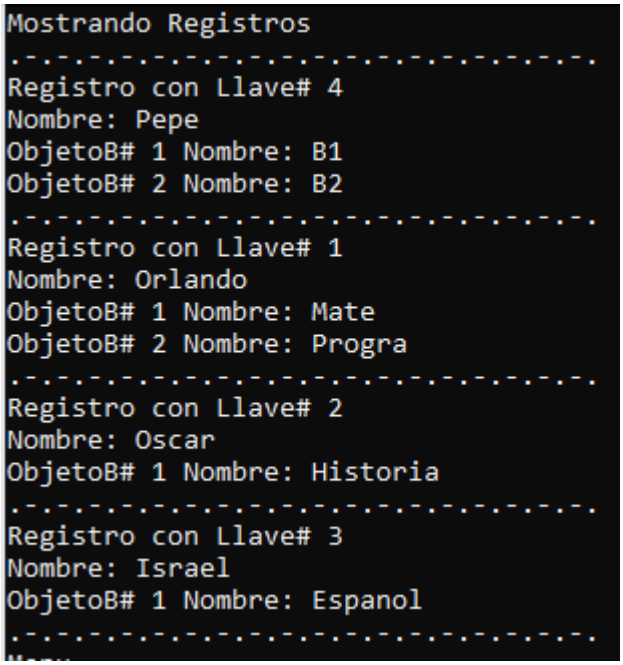
Ahora que se ingreso lo comprobaremos con la opción mostrar registros:

```
Mostrando Registros
.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.
Registro con Llave# 1
Nombre: Orlando
ObjetoB# 1 Nombre: Mate
ObjetoB# 2 Nombre: Progra
.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.
Registro con Llave# 2
Nombre: Oscar
ObjetoB# 1 Nombre: Historia
.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.
Registro con Llave# 3
Nombre: Israel
ObjetoB# 1 Nombre: Espanol
.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.
```

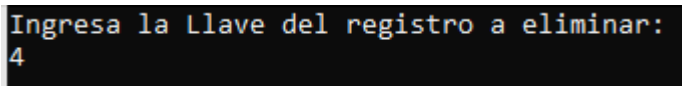
Ahora insertaremos al inicio a pepe llave 4 2 objetos B que serian B1 y B2:

```
Ingresa la llave del objeto de clase A (debe ser nueva)
4
Ingresa el nombre del objeto de clase A a crear
Pepe
Cuantos objetos Clase B tendra? max 10
2
Objeto B numero: 1
Ingresa el nombre del objeto de clase B
B1
Objeto B numero: 2
Ingresa el nombre del objeto de clase B
B2
```

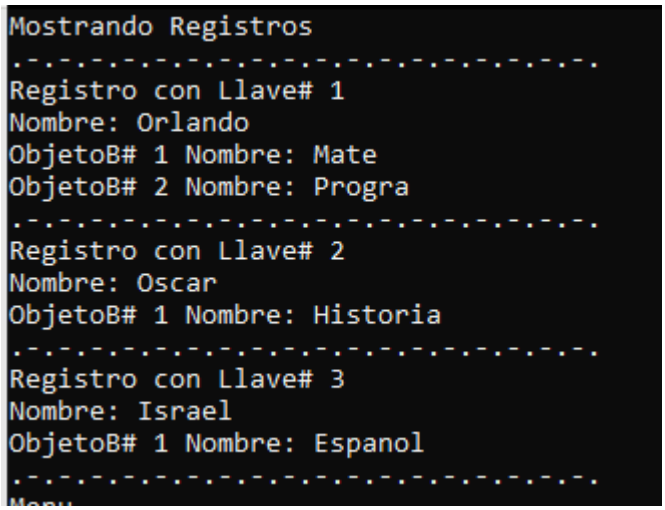
Mostremos para comprobar:



Ahora eliminaremos a Pepe que es llave 4 porque me parece que no lo necesitamos en el registro así que seleccionare esa opción en el menú.



Mostramos para ver si se elimino:



Ahora modificare el nombre de Israel de llave 3 a Gabriel:

```
Ingresa la Llave del registro a Modificar:
3
Se encontro el registro a Modificar
Que deseas modificar?
1.-Nombre
2.-Llave
Ingresa una opcion:
1
Ingresa el nuevo nombre
Gabriel
```

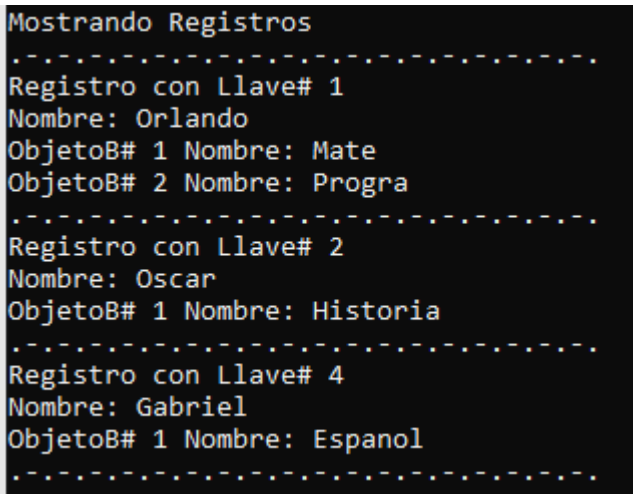
Mostramos para verificar:

```
Mostrando Registros
.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.
Registro con Llave# 1
Nombre: Orlando
ObjetoB# 1 Nombre: Mate
ObjetoB# 2 Nombre: Progra
.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.
Registro con Llave# 2
Nombre: Oscar
ObjetoB# 1 Nombre: Historia
.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.
Registro con Llave# 3
Nombre: Gabriel
ObjetoB# 1 Nombre: Espanol
.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.
```

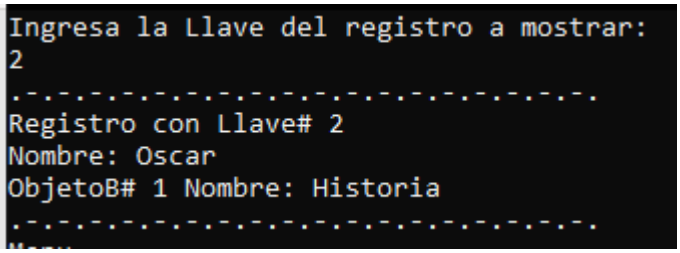
Ahora modificaremos la llave de Gabriel a una llave 4 en vez de 3:

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

Mostramos para verificar:



Ahora mostraremos un solo registro a partir de su llave, me apetece mostrar la llave 2 correspondiente a Oscar:



6. Conclusiones.

La practica al igual que la anterior que eran delimitadores es sencilla una vez que entiendes como manejar los archivos aunque al ser ahora binario se complica un poco mas el manejo aunque me parece una buena forma de almacenamiento si fuera de longitud fija el cual no es el caso, al ser de campos de dimensión de longitud variable el hecho de escribir es fácil pero leer es otra historia, en esta ocasión opte por almacenar todo el archivo en una cadena para poder manipular los datos con mejor control pero aun así creo que no es la mejor opción, se me dificulto bastante la serialización y la recuperación de los datos, fue algo que me llevo tiempo pero al final puedo decir que se lograron los objetivos.

7. Apéndice(s).

Apéndice 1.-Cargar registro.

```
int Clase_A::CargarRegistro(string Cadena, int Indice)
{
	ReiniciarMemoria();
	Indice++;
```

```
	Llave = Cadena[Indice] - 48;
	Indice++;
	int TamNombre = Cadena[Indice] - 48;
	Indice++;
	for (int i = 0; i < TamNombre; i++) {
		Nombre += ".";
	}
	for (int i = 0; i < TamNombre; i++) {
		Nombre[i] = Cadena[Indice];
		Indice++;
	}
	int NumeroB = Cadena[Indice] - 48;
	for (int i = 0; i < NumeroB; i++) {
		Indice++;
		TamNombre = Cadena[Indice] - 48;
		Indice++;
		for (int j = 0; j < TamNombre; j++) {
			ObjB[i].Cadena += ".";
		}
		for (int j = 0; j < TamNombre; j++) {
			ObjB[i].Cadena[j] = Cadena[Indice];
			Indice++;
		}
		ObjB[i].Booleano = true;
	}
	return Indice;
}
```

Apéndice 2.-validar llave.

```
bool Clase_A::ValidarLlave(int Propuesta)
```

```
{
	string Documento;
	ifstream Archivo("Auxiliar.bin", ios::binary);
	Archivo >> Documento;
	Archivo.close();
	int TamDocumento = Documento.length();
	int Indice = -1;
	while (Indice < TamDocumento - 1) {
		Indice = CargarRegistro(Documento, Indice);
		if (Llave == Propuesta) {
			return false;//No se puede usar llave propuesta
		}
	}
	return true;//se puede usar llave propuesta
}
```

Apéndice 3.-llenar auxiliar.

```
void Clase_A::llenarAuxiliar()
{
	string Documento;
	ifstream Archivo("principal.bin", ios::binary);
	Archivo >> Documento;
	Archivo.close();
	ofstream ArchivoAUX("Auxiliar.bin", ios::binary);
	ArchivoAUX << Documento;
	ArchivoAUX.close();
}
```

Apéndice 4.-Reiniciar memoria.

```
void Clase_A::ReiniciarMemoria()
{
```

```
	Nombre = "";
	Llave = -1;
	Clase_B Nuevo;
	for (int i = 0; i < 10; i++) {
		ObjB[i] = Nuevo;
	}
}
```

Apéndice 5.- Crear objeto.

```
void Clase_A::CrearObjeto()
{
	ReiniciarMemoria();
	llenarAuxiliar();
	int Objb,propuesta;
	bool Unica = false;
	while(Unica==false){
	cout << "Ingresa la llave del objeto de clase A (debe ser nueva)"
<< endl;
	cin >> propuesta;
	Unica = ValidarLlave(propuesta);
	}
	ReiniciarMemoria();
	cout << "Ingresa el nombre del objeto de clase A a crear" << endl;
	cin >> Nombre;
	Llave = propuesta;
	cout << "Cuantos objetos Clase B tendra? max 10" << endl;
	cin >> Objb;
	for (int i = 0; i < Objb; i++) {
		cout << "Objeto B numero: "<< i+1 << endl;
		ObjB[i].Modificar();
	}
```

```
}
```

Apéndice 6.-Insertar al Final.

```
void Clase_A::InsertarFinal()
{
	llenarAuxiliar();
	string Documento;
	ifstream ArchivoAUX("Auxiliar.bin", ios::binary);
	ArchivoAUX >> Documento;
	ArchivoAUX.close();
	CrearObjeto();
	ofstream Archivo("principal.bin", ios::binary);
	Archivo << Documento;
	int Tamnombre = Nombre.length();
	int CantB = 0;
	Archivo << Llave;
	Archivo << Tamnombre;
	Archivo << Nombre;
	for (int i = 0; i < 10; i++) {
		if (ObjB[i].Validador()) {
			CantB++;
		}
	}
	Archivo << CantB;
	for (int i = 0; i < 10; i++) {
		ObjB[i].Guardar(Archivo);
	}
	Archivo.close();
}
```

Apéndice 7.-Insertar Inicio (igual que insertar al final solo que se añade el string "Documento" al final en vez de al inicio.

```
ofstream Archivo("principal.bin", ios::binary);
	int Tamnombre = Nombre.length();
	int CantB = 0;
	Archivo << Llave;
	Archivo << Tamnombre;
	Archivo << Nombre;
	for (int i = 0; i < 10; i++) {
		if (ObjB[i].Validador()) {
			CantB++;
		}
	}
	Archivo << CantB;
	for (int i = 0; i < 10; i++) {
		ObjB[i].Guardar(Archivo);
	}
	Archivo << Documento;
	Archivo.close();
```

Apéndice 8.- Eliminar.

```
bool Clase_A::Eliminar()
{
	string Documento;
	ifstream Archivo("principal.bin", ios::binary);
	Archivo >> Documento;
	Archivo.close();
	int TamDocumento = Documento.length();
	int Indice = -1;
	int LlaveEliminar;
	bool Eliminado = false;
	cout << "Ingresa la Llave del registro a eliminar: " << endl;
```

```
	cin >> LlaveEliminar;
	ofstream ArchivoNew("principal.bin", ios::binary);
	while (Indice < TamDocumento - 1) {
		Indice = CargarRegistro(Documento, Indice);
		if (Llave != LlaveEliminar) {
			int Tamnombre = Nombre.length();
			int CantB = 0;
			ArchivoNew << Llave;
			ArchivoNew << Tamnombre;
			ArchivoNew << Nombre;
			for (int i = 0; i < 10; i++) {
				if (ObjB[i].Validador()) {
					CantB++;
				}
			}
			ArchivoNew << CantB;
			for (int i = 0; i < 10; i++) {
				ObjB[i].Guardar(ArchivoNew);
			}
		}
		else
			Eliminado = true;
	}
	return Eliminado;
}
```

Apéndice 9.- Modificar.

```
bool Clase_A::Modificar()
{
	string Documento;
```

```
ifstream Archivo("principal.bin", ios::binary);
Archivo >> Documento;
Archivo.close();
int TamDocumento = Documento.length();
int Indice = -1;
int LlaveMod;
bool Modificado = false;
cout << "Ingresa la Llave del registro a Modificar: " << endl;
cin >> LlaveMod;
ofstream ArchivoNew("principal.bin", ios::binary);
while (Indice < TamDocumento - 1) {
	Indice = CargarRegistro(Documento, Indice);
	if (Llave != LlaveMod) {
		int Tamnombre = Nombre.length();
		int CantB = 0;
		ArchivoNew << Llave;
		ArchivoNew << Tamnombre;
		ArchivoNew << Nombre;
		for (int i = 0; i < 10; i++) {
			if (ObjB[i].Validador()) {
				CantB++;
			}
		}
		ArchivoNew << CantB;
		for (int i = 0; i < 10; i++) {
			ObjB[i].Guardar(ArchivoNew);
		}
	}
	else{
		Modificado = true;
```

```
                MenuModificar();
                int Tamnombre = Nombre.length();
                int CantB = 0;
                ArchivoNew << Llave;
                ArchivoNew << Tamnombre;
                ArchivoNew << Nombre;
                for (int i = 0; i < 10; i++) {
                      if (ObjB[i].Validador()) {
                            CantB++;
                      }
                }
                ArchivoNew << CantB;
                for (int i = 0; i < 10; i++) {
                      ObjB[i].Guardar(ArchivoNew);
                }
          }
      }
      return Modificado;
}
```

Apéndice 10.- Mostrar Registro.

```
void Clase_A::MostrarRegistro()
{
      string Documento;
      ifstream Archivo("principal.bin", ios::binary);
      Archivo >> Documento;
      Archivo.close();
      int TamDocumento = Documento.length();
      int Indice = -1;
      int LlaveMostrar;
```

```
	cout << "Ingresa la Llave del registro a mostrar: " << endl;
	cin >> LlaveMostrar;
	while (Indice < TamDocumento - 1) {
		Indice = CargarRegistro(Documento, Indice);
		if(Llave==LlaveMostrar){
			cout << ".-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-." <<
endl;
			cout << "Registro con Llave# " << Llave << endl;
			cout << "Nombre: " << Nombre << endl;
			for (int i = 0; i < ObjetosB; i++) {
				if (ObjB[i].Booleano) {
					cout << "ObjetoB# " << i + 1 << " Nombre: "
<< ObjB[i].Cadena << endl;
				}
			}
			cout << ".-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-." <<
endl;
		}
	}
}
```