

# ACTIVIDAD 1. DELIMITADORES.

## 1. Planteamiento del problema.

Diseñar una solución informática que demuestre la comprensión y uso de ficheros para la representación en memoria secundaria de una lista. El almacenamiento de los datos debe realizarse con la estrategia de delimitadores. La forma del fichero será:

```
at1Reg1|at2Reg1|at3.1Reg1|at3.2Reg1|at3.3Reg1|at3.4Reg1  
at1Reg2|at2Reg2|at3.1Reg2|at3.2Reg2|at3.3Reg2|at3.4Reg2 . . .  
at1RegN|at2RegN|at3.1RegN|at3.2RegN|at3.3RegN|at3.4RegN
```

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 4 enteros (el arreglo siempre contiene datos). La llave primaria es un campo que se debe solicitar, y por esto, resulta necesario asegurarse que esa llave primaria no exista.

## 2. Objetivos:

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

## 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.

## 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 clave primaria que es de tipo entero la llamé ID y finalmente mi arreglo de valores enteros lo llamé Calificaciones, quise darle un enfoque a registro de calificaciones de alumnos al programa, aunque claro esto solo sería la excusa perfecta para implementar esta actividad de registros con delimitadores.

### Parte 1. Insertar y validar llave primaria.

Inicialmente para insertar un registro en cualquier parte del archivo es necesario tener claro los datos del registro que queremos agregar al archivo, por lo que como se puede observar en el **apéndice 1.1** el método insertar recopila la información correspondiente a un registro y la almacena en un string que retorna al final de su método, en esta función la parte importante está en el momento de pedir el ID al usuario, ya que nuestro programa no debe aceptar un ID que sea igual a un ID ya existente dentro del archivo.

También se inserta entre cada atributo del registro un carácter delimitador que en este caso sería '|' se inserta entre cada atributo excepto al final del atributo final ya que allí iría un salto de línea que delimitaría el final de nuestro registro.

Para validar el ID véase el **apéndice 1.4** que es el método de validación de ID, se trata de un método que compara en un archivo auxiliar que contiene los registros previos a la inserción de registro a realizar, lo que hace es que va y lee cada atributo ID de los registros y los compara con la propuesta de ID por parte del usuario, si coincide esta propuesta de ID con uno de los ID existentes el método regresara un booleano falso que da a entender que ese ID ya existe y que se debe pedir nuevamente un ID que no se encuentre previamente en el archivo.

Claro que si el método reviso todos los registros y no encontró una coincidencia retornara un booleano verdadero dando a entender que el ID propuesto por el usuario es válido y único por lo que se puede usar.

Para insertar al inicio véase el **apéndice 1.2** en el copiamos todos los registros existentes en el archivo principal a uno secundario. Posteriormente realizamos una escritura en el archivo utilizando el método de insertar visto en **apéndice 1.1**, esta escritura se hace con el método predeterminado de escritura de ofstream que borra lo existente del archivo e inserta al inicio del fichero, de esta manera ya tenemos el registro insertado al inicio, posterior mente se fueron insertando en orden los registros que ya teníamos previamente y que se guardaron en nuestro archivo auxiliar esta vez usando el método de escritura APP que inserta al final del archivo pero sin eliminar la información existente por lo que nuestro registro insertado al inicio se mantendrá.

Para insertar al final el trabajo se simplifica ya que con el método de escritura APP se inserta un registro al final sin problema, pero, para que se validen correctamente los datos véase **apéndice 1.4** debemos cargar los registros actuales a un archivo secundario como se muestra en el **apéndice 1.3** antes de la inserción al final en el archivo.

## Parte 2 Mostrar registros.

Para mostrar los registros, opte por irlos imprimiendo con forme se va leyendo el archivo desde el inicio hasta el final, como se puede ver en el **apéndice 2**, se usa un método de lectura básico en el que cada línea de del archivo que es a su vez un registro se lee en una variable llamada row, la cual se usa para leer cada registro, con ayuda de los ciclos While podemos ir leyendo cada atributo del registro hasta que se encuentre con un delimitador, el orden en que se guardaron los atributos de los archivos como se muestra en **apéndice 1.1** es el siguiente:

```
ID|Nombre|Calificacion1| Calificacion2| Calificacion3| Calificacion4(Salto_de_linea)
```

Por lo que el método mostrar lee cada atributo y cuando lo termina de leer lo imprime y sigue con el siguiente atributo hasta llegar al final del registro, después de esto si el archivo ya está en su posición final termina el método, si no leemos un registro nuevo y lo guardamos en row para repetir el proceso antes mencionado.

## Parte 3 Búsqueda de registros.

Para buscar un registro se uso una estructura similar a la del método de verificación de unicidad de ID (véase **apéndice 1.4**), el método requiere recibir como parámetro una cadena con el ID del registro a buscar, si encuentra ese ID correspondiente a un registro del archivo va a cargar a memoria del objeto de clase A ese registro a sus atributos a manera de recuperación del registro que buscamos, para posteriormente devolver un valor booleano verdadero afirmando que se encontró el registro, en cambio si no se encuentra el ID en el archivo devolverá un falso.

#### Parte 4 Eliminar

Para el método eliminar (Véase **apéndice 3**) se pide al usuario el ID del registro a eliminar para después andar llamar a la función de búsqueda (**Apéndice 2**) en la que si no se encuentra el registro el programa lo dirá y saldrá del método eliminar.

En cambio, si lo encuentra precederá a empezar la eliminación del registro, para esto, sabemos que en la función de búsqueda cuando encuentra el registro a buscar almacena en memoria del objeto el registro, por lo que usaremos esos datos para eliminar el registro en el archivo.

Primero vamos a escribir cada uno de los registros del archivo principal en el archivo secundario, pero con una condición, se va a escribir el registro en archivo auxiliar si y solo si el ID del archivo a escribir no es igual al ID que tiene nuestro objeto (Que es el que queremos eliminar), de esta manera tendremos al final en el archivo auxiliar los registros originales menos el registro que se quería eliminar.

Finalmente pasamos todos los registros de el archivo auxiliar al principal (véase **apéndice 4**), todo esto renovando los archivos con el método de escritura por defecto que borra el contenido existente y agrega solo el nuevo, de esta manera tenemos la eliminación del registro exitosa.

#### Parte 5 Modificar

Para modificar se utiliza prácticamente la misma estructura que se uso para eliminar, solo que los detalles a resaltar son los siguientes: el método después de buscar el registro, encontrarlo y cargarlo a los atributos del objeto se muestra un menú que permite modificar el Nombre(string) o las Calificaciones(enteros), una vez editados los valores de los atributos del objeto podríamos decir que el registro se a modificado exitosamente, solo quedaría remplazar ese mismo registro por el registro modificado.

Como se puede ver en el apéndice 5 se trasladan todos los registros del archivo original al secundario, pero en este caso a diferencia del método eliminar (**apéndice 3**) la condición no ignorara el registro si no que va a insertar el registro modificado desde los atributos del objeto, de esta manera el orden en el archivo auxiliar al final será el mismo de antes y el registro a modificar estará modificado con éxito.

Una vez mas trasladamos el archivo auxiliar al archivo principal (**apéndice 4**) y el método habría terminado con éxito la modificación de un registro.

## 5. Pruebas y resultados.

Prueba de Inserción:

Previamente tenemos en el archivo principal 2 registros:

1|Orlando|90|91|92|93

2|Oscar|80|81|82|83

Vamos a insertar al inicio pero al principio intentare usar los ID existentes para el nuevo registro:

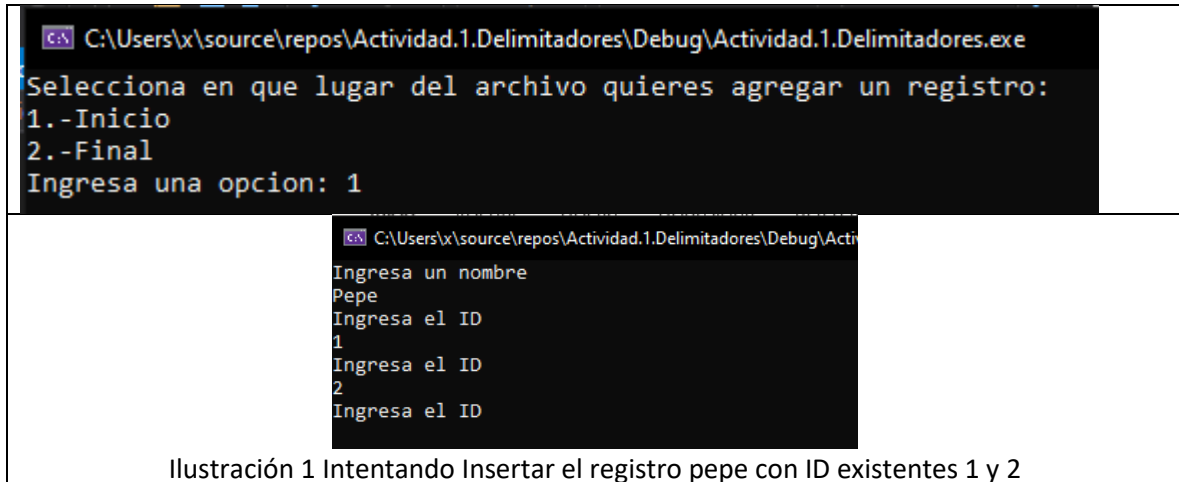


Ilustración 1 Intentando Insertar el registro pepe con ID existentes 1 y 2

Como podemos observar en la ilustración 1 no me deja ingresar un id existente, siempre que quiero hacerlo me regresa a el formulario para volver a dar un ID valido, esto demuestra el funcionamiento de la validación de unicidad de llave primaria/ID.

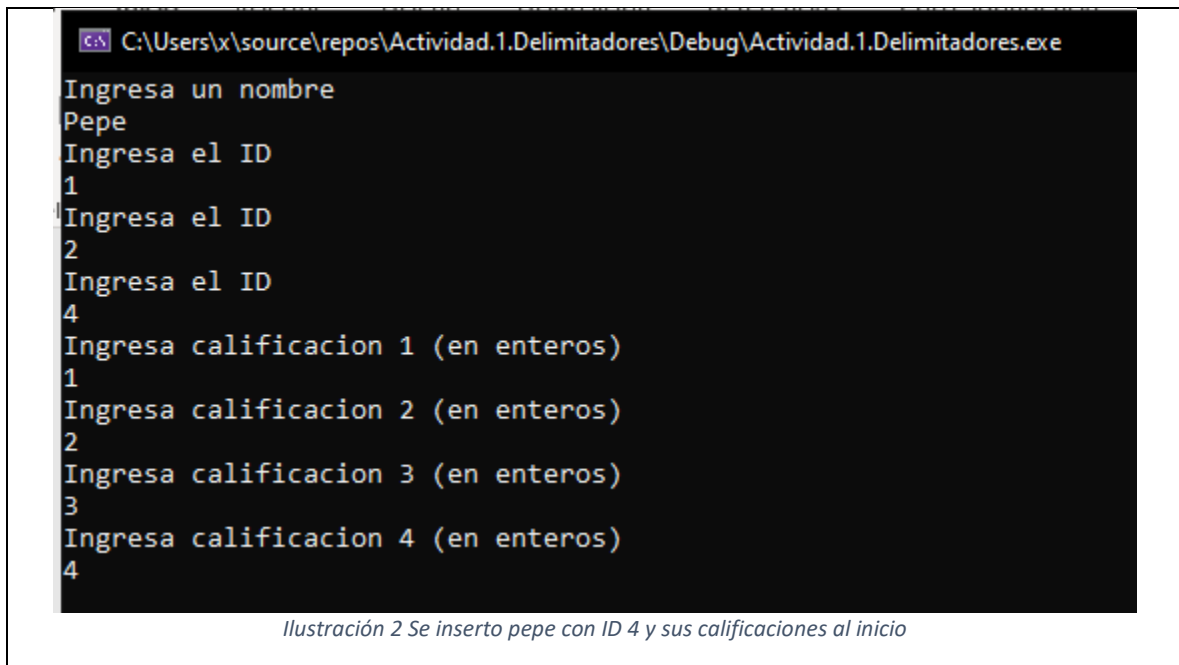
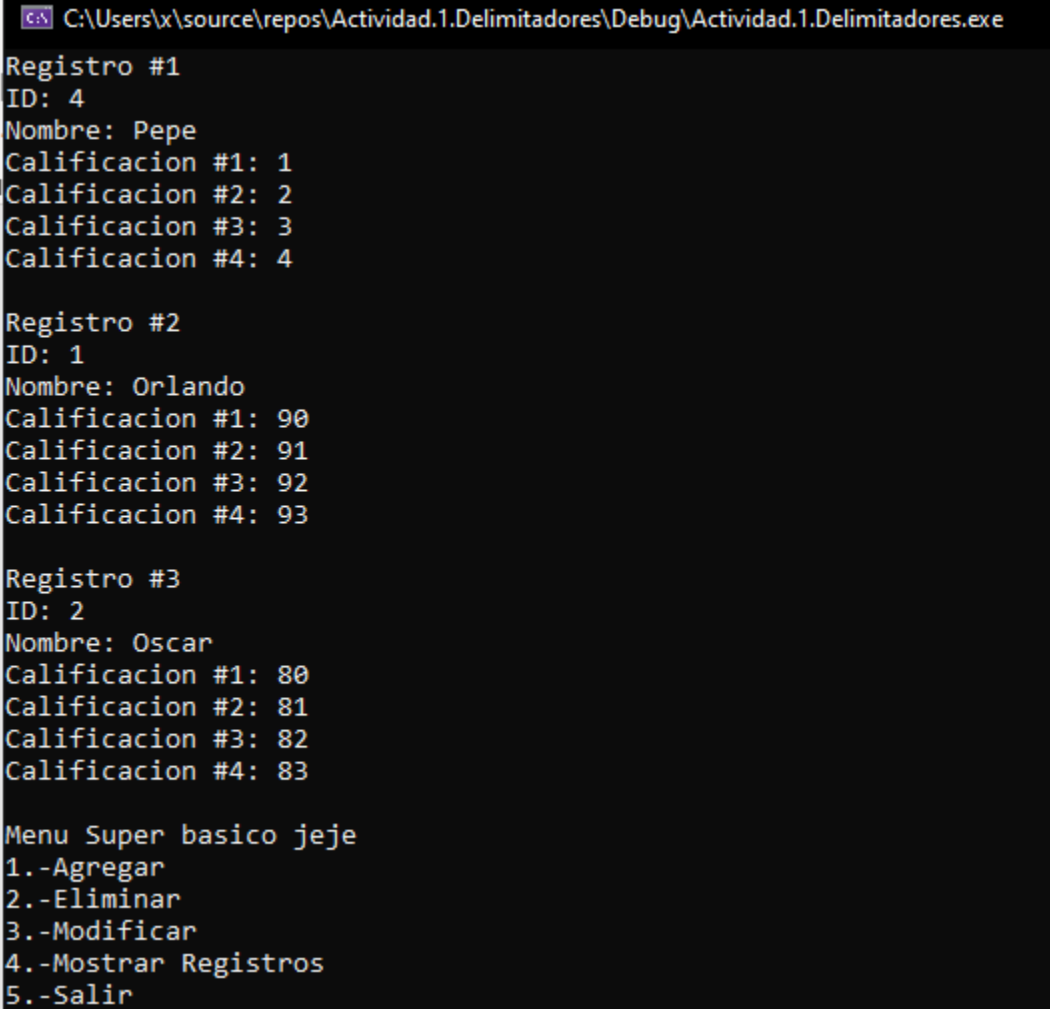


Ilustración 2 Se inserto pepe con ID 4 y sus calificaciones al inicio

Al insertar el registro de “Pepe” al inicio comprobaremos que se insertó usando la opción de mostrar registros:



```
C:\Users\x\source\repos\Actividad.1.Delimitadores\Debug\Actividad.1.Delimitadores.exe

Registro #1
ID: 4
Nombre: Pepe
Calificacion #1: 1
Calificacion #2: 2
Calificacion #3: 3
Calificacion #4: 4

Registro #2
ID: 1
Nombre: Orlando
Calificacion #1: 90
Calificacion #2: 91
Calificacion #3: 92
Calificacion #4: 93

Registro #3
ID: 2
Nombre: Oscar
Calificacion #1: 80
Calificacion #2: 81
Calificacion #3: 82
Calificacion #4: 83

Menu Super basico jeje
1.-Agregar
2.-Eliminar
3.-Modificar
4.-Mostrar Registros
5.-Salir
```

*Ilustración 3 Mostrando Registros actuales*

Podemos observar en la ilustración 3 que Pepe se inserto al inicio del archivo sin problema, a continuación insertare un registro al final del archivo.



En la ilustración 4 se muestra la inserción al final de Alan, ahora mostrare los registros para comprobar.

```
C:\Users\x\source\repos\Actividad.1.Delimitadores\Debug\Actividad.1.Delimitadores.exe

Registro #1
ID: 4
Nombre: Pepe
Calificacion #1: 1
Calificacion #2: 2
Calificacion #3: 3
Calificacion #4: 4

Registro #2
ID: 1
Nombre: Orlando
Calificacion #1: 90
Calificacion #2: 91
Calificacion #3: 92
Calificacion #4: 93

Registro #3
ID: 2
Nombre: Oscar
Calificacion #1: 80
Calificacion #2: 81
Calificacion #3: 82
Calificacion #4: 83

Registro #4
ID: 3
Nombre: Alan
Calificacion #1: 67
Calificacion #2: 68
Calificacion #3: 69
Calificacion #4: 70
```

*Ilustración 5 Alan insertado al final con éxito*

La inserción al final fue exitosa, ahora quiero cambiar el nombre de Pepe a Jose, vamos a usar la opción de modificar el registro de Pepe es el ID 4.



```
C:\Users\x\source\repos\Actividad.1.Delimitadores\Debug\Actividad.1.Delimitadores.exe
Modificar Registro
Ingresa el ID del registro a modificar:
4
Registro:
ID#4 Nombre: Pepe
Calificacion #1: 1    Calificacion #2: 2
Calificacion #3: 3    Calificacion #4: 4

Que quieres modificar del Registro?
1.-Nombre
2.-Calificaciones
Ingresa una opcion: 1
Ingresa el nuevo nombre:
Jose
```

*Ilustración 6 Realizando la búsqueda del registro 4 y asignando el nuevo nombre*

Al realizar la acción de la ilustración 6 habremos cambiado el registro de Pepe a Jose, recordemos que su ID es el 4 y que actualmente es el primer registro escrito en el archivo.

```
C:\Users\x\source\repos\Actividad.1.Delimitadores\Debug\Actividad.1.Delimitadores.exe

Registro #1
ID: 4
Nombre: Jose
Calificacion #1: 1
Calificacion #2: 2
Calificacion #3: 3
Calificacion #4: 4

Registro #2
ID: 1
Nombre: Orlando
Calificacion #1: 90
Calificacion #2: 91
Calificacion #3: 92
Calificacion #4: 93

Registro #3
ID: 2
Nombre: Oscar
Calificacion #1: 80
Calificacion #2: 81
Calificacion #3: 82
Calificacion #4: 83

Registro #4
ID: 3
Nombre: Alan
Calificacion #1: 67
Calificacion #2: 68
Calificacion #3: 69
Calificacion #4: 70
```

*Ilustración 7 Cambio de nombre del registro 4 exitoso*

En la ilustración 7 vemos el cambio de nombre exitoso en el registro con ID 4, ahora a ese mismo registro le daremos calificaciones reales, por lo que ahora modificaremos las calificaciones del registro con ID 4.

```
C:\Users\x\source\repos\Actividad.1.Delimitadores\Debug\Actividad.1.Delimitadores.exe
Modificar Registro
Ingresa el ID del registro a modificar:
4
Registro:
ID#4 Nombre: Jose
Calificacion #1: 1    Calificacion #2: 2
Calificacion #3: 3    Calificacion #4: 4

Que quieres modificar del Registro?
1.-Nombre
2.-Calificaciones
Ingresa una opcion: 2
Ingresa la nueva calificacion#:1
90
Ingresa la nueva calificacion#:2
80
Ingresa la nueva calificacion#:3
70
Ingresa la nueva calificacion#:4
60
```

*Ilustración 8 Modificando Calificaciones de registro con ID 4*

En la ilustración 8 modificamos las calificaciones del registro con ID 4, ahora veremos si la modificación funciona.

```
C:\Users\x\source\repos\Actividad.1.Delimitadores\Debug\Actividad.1.Delimitadores.exe

Registro #1
ID: 4
Nombre: Jose
Calificacion #1: 90
Calificacion #2: 80
Calificacion #3: 70
Calificacion #4: 60

Registro #2
ID: 1
Nombre: Orlando
Calificacion #1: 90
Calificacion #2: 91
Calificacion #3: 92
Calificacion #4: 93

Registro #3
ID: 2
Nombre: Oscar
Calificacion #1: 80
Calificacion #2: 81
Calificacion #3: 82
Calificacion #4: 83

Registro #4
ID: 3
Nombre: Alan
Calificacion #1: 67
Calificacion #2: 68
Calificacion #3: 69
Calificacion #4: 70
```

*Ilustración 9 Calificaciones de registro con ID 4 modificadas con éxito.*

Como se ve en la ilustración 9 Se pudieron modificar las calificaciones de el registro de ID 4, ahora para finalizar de comprobar las funcionalidades del programa eliminaremos un registro, en este caso borraremos el registro de Oscar ID 2, resulta que Oscar nunca existió por lo que solo abarca espacio innecesario en el registro por lo que eliminaremos su registro.

```
C:\Users\x\source\repos\Actividad.1.Delimitadores\Debug\Actividad.1.Delimitadores.exe

Eliminar Registro
Ingresa el ID del registro que quieres eliminar:
2
```

*Ilustración 10 Eliminando Registro con ID 2*

En la ilustración 10 vemos que se va a eliminar el registro con ID 2, ahora veremos si es verdad que se borro.

Como podemos ver la eliminación de un registro funciona perfectamente.

Con eso comprobaría la funcionabilidad de mi programa cumpliendo con todos los requerimientos funcionales de la actividad 1, a su vez es recomendable ver el video en el que se muestra el funcionamiento del programa.

## 6. Conclusiones.

Esta práctica es relativamente fácil en cuanto entiendes como funcionan la lectura y escritura de los archivos, en este caso los delimitadores ayudan bastante a identificar los atributos o registros con los que se deben interactuar a lo largo del programa, tuve problemas en el aspecto de usar 2 archivos a la vez en este caso el archivo principal y el secundario ya que el método que usaba para leer o escribir en ellos era vital ya que si usaba un método de lectura o escritura equivocado provocaba registros duplicados o registros escritos a medias y esto por consiguiente daba errores, es por eso que resalto el hecho de comprender bien el manejo de los archivos para hacer la actividad.

## 7. Apéndice(s).

### 1.1 Método Insertar:

```

string Clase_A::Insertar()
{
    string Nombre,ID;
    int C1,C2,C3,C4;
    cout << "Ingresa un nombre" << endl;
    cin >> Nombre;
    do {
        cout << "Ingresa el ID" << endl;
        cin >> ID;
        if (Unicidad_de_ID(ID)) {
            break;
        }
    } while (true);

    cout << "Ingresa calificacion 1 (en enteros)" << endl;
    cin >> C1;
    cout << "Ingresa calificacion 2 (en enteros)" << endl;
    cin >> C2;
    cout << "Ingresa calificacion 3 (en enteros)" << endl;
    cin >> C3;
    cout << "Ingresa calificacion 4 (en enteros)" << endl;
    cin >> C4;
    stringstream ss;
    ss << ID << "|" << Nombre << "|" << C1 << "|" << C2 << "|" << C3 <<
    "|" << C4;
    return ss.str();
}

```

## 1.2 Método Insertar al Inicio:

```

void Clase_A::Insertar_al_Inicio()
{
    string row;
    ifstream Aoriginal;

```

```

        Aoriginal.open("iny.txt");
        Aoriginal >> row;

        ofstream Aauxiliar("Auxiliar.txt");
        while (!Aoriginal.eof()) {
            Aauxiliar << row << endl;
            Aoriginal >> row;
        }

        Aauxiliar.close();
        Aoriginal.close();

        ofstream fout("iny.txt");
        fout << Insertar() << endl;
        fout.close();

        ifstream ArchivoAuxiliar;
        ArchivoAuxiliar.open("Auxiliar.txt");
        ArchivoAuxiliar >> row;

        ofstream ArchivoPrincipal("iny.txt", ios::app);
        while (!ArchivoAuxiliar.eof()) {
            ArchivoPrincipal << row << endl;
            ArchivoAuxiliar >> row;
        }

        ArchivoPrincipal.close();
        ArchivoAuxiliar.close();
    }
}

```

### 1.3 Método Insertar al Final:

```

void Clase_A::Insertar_al_Final()
{
    string row;

    ifstream Aoriginal;
    Aoriginal.open("iny.txt");
    Aoriginal >> row;

    ofstream Aauxiliar("Auxiliar.txt");
    while (!Aoriginal.eof()) {

```

```

        Aauxiliar << row << endl;
        Aoriginal >> row;
    }
    Aauxiliar.close();
    Aoriginal.close();
    ofstream fout("iny.txt", ios::app);
    fout << Insertar() << endl;
    fout.close();
}

```

#### **1.4 Método de validación de unicidad de llave primaria (ID):**

```

bool Clase_A::Unicidad_de_ID(string ID)
{
    string Clave, row;
    ifstream Lectura;
    Lectura.open("Auxiliar.txt");
    Lectura >> row;
    while (!Lectura.eof()) {
        int contador = 0;
        Clave = "";
        while (row[contador] != '|') {
            Clave += row[contador];
            contador++;
        }
        if (Clave == ID) {
            return false;
        }
        Lectura >> row;
    }
    Lectura.close();
    return true;
}

```

#### **2.- Mostrar registros**



```

void Clase_A::Mostrar_Registros()
{
    ifstream Lectura;
    string Cadena, row;
    int Registro=1;
    Lectura.open("iny.txt");
    Lectura >> row;
    while (!Lectura.eof()) {
        cout << "Registro #" << Registro << endl;
        int contador = 0;
        Cadena = "";
        while (row[contador] != '|') {
            Cadena += row[contador];
            contador++;
        }
        cout << "ID: " << Cadena << endl;
        contador++;
        Cadena = "";
        while (row[contador] != '|') {
            Cadena += row[contador];
            contador++;
        }
        cout << "Nombre: " << Cadena << endl;
        int contador2;
        for (contador2 = 0; contador2 < 3; contador2++) {
            contador++;
            Cadena = "";
            while (row[contador] != '|') {
                Cadena += row[contador];
                contador++;
            }
        }
    }
}

```

```

        cout << "Calificacion #" << contador2+1 << ": " <<
Cadena << endl;
    }
    int Aux = row.size() - contador;
    contador++;
    Cadena = "";
    for (Aux; Aux != 0; Aux--) {
        Cadena += row[contador];
        contador++;
    }
    cout << "Calificacion #4: " << Cadena << endl << endl;
    Registro++;
    Lectura >> row;
}
Lectura.close();
}

```

### 3 Metodo Eliminar

```

void Clase_A::Eliminar_Registros()
{
    string ID_eliminar;
    cout << "Elimidar Registro" << endl;
    cout << "Ingresa el ID del registro que quieres eliminar:" << endl;
    cin >> ID_eliminar;
    if (Buscar_Registros(ID_eliminar)) {
        string row;
        ifstream Aoriginal;
        Aoriginal.open("iny.txt");
        Aoriginal >> row;
        ofstream Aauxiliar("Auxiliar.txt");
        while (!Aoriginal.eof()) {
            string ID_leido = "";
            int contador = 0;

```

```

        while (row[contador] != '|') {
            ID_leido += row[contador];
            contador++;
        }
        if (atoi(ID_leido.c_str()) != this->ID){
            Aauxiliar << row << endl;
        }
        Aoriginal >> row;

    }

    llenar_Archivo_Principal();
    Aauxiliar.close();
    Aoriginal.close();
}

else {
    cout << "No se encontro el registro a eliminar..." << endl;
}

}

```

#### **4 Metodo para llenar el archivo principal usando uno auxiliar**

```

void Clase_A::llenar_Archivo_Principal()
{
    ifstream ArchivoAuxiliar;
    string row;
    ArchivoAuxiliar.open("Auxiliar.txt");
    ArchivoAuxiliar >> row;
    ofstream ArchivoPrincipal("iny.txt");
    while (!ArchivoAuxiliar.eof()) {
        ArchivoPrincipal << row << endl;
        ArchivoAuxiliar >> row;
    }
    ArchivoPrincipal.close();
    ArchivoAuxiliar.close();
}

```

```
}
```

### 5 parte importante del método Modificar:

```
stringstream ss;

    ss << this->ID << "|" << this->Nombre << "|" << this->Calificaciones[0] << "|" << this->Calificaciones[1] << "|" << this->Calificaciones[2] << "|" << this->Calificaciones[3];

    string row;
    ifstream Aoriginal;
    Aoriginal.open("iny.txt");
    Aoriginal >> row;
    ofstream Aauxiliar("Auxiliar.txt");
    while (!Aoriginal.eof()) {
        string ID_leido = "";
        int contador = 0;
        while (row[contador] != '|') {
            ID_leido += row[contador];
            contador++;
        }
        if (atoi(ID_leido.c_str()) != this->ID) {
            Aauxiliar << row << endl;
        }
        else {
            Aauxiliar << ss.str() << endl;
        }
        Aoriginal >> row;
    }

    llenar_Archivo_Principal();
    Aauxiliar.close();
    Aoriginal.close();
```