

Archivos binarios

Definición

Un archivo es una colección de datos que se encuentran almacenados de manera permanente en algún dispositivo externo del computador (memoria secundaria) como el disco duro, dispositivo o memoria USB, CDs, etc. En realidad, todo lo que se almacena en el computador se considera un archivo, así por ejemplo si escribimos una carta en algún procesador de palabras y luego la guardamos, esta carta se constituye en un archivo, si escribimos un programa en un editor de textos y lo guardamos, éste también es un archivo, de igual manera el programa ejecutable (con extensión .exe) también constituye un archivo, y así una fotografía digital, un video en un CD, una hoja de cálculo también son archivos.

En otros capítulos se ha trabajado con archivos de texto, se ha visto la manera tan práctica que representa manejar un archivo de textos para el ingreso y la salida de datos y también se han apreciado las limitaciones que tienen estos archivos, sin embargo, los archivos de textos son sólo un tipo de archivo. Lo que estudiaremos en este capítulo es otro tipo de archivo, los archivos binarios, que sin alejarse de la definición de lo que es un archivo, por la forma en que se almacenan los datos difiere sustancialmente de lo que es un archivo de textos y por lo tanto la manera de procesarlos también será diferente.

Formas en las que se puede almacenar información en un archivo

Cuando se habla de un archivo, desde el punto de vista de la programación, no es del todo correcto hablar de "tipos de archivos". Esto porque en realidad un archivo es sólo una colección de bytes consecutivos almacenados en el computador, y en ese sentido todos los archivos son lo mismo para el computador. Lo que hace la diferencia es el formato cómo se almacena la información; es así que podemos distinguir dos modelos diferentes, los conocidos como "archivos de texto" y los denominados "archivos binarios".

En los archivos de texto, la información sufre una transformación. Cuando se guarda un dato en un archivo de texto, el valor es convertido a una cadena de caracteres antes de almacenarse en el computador. Recuerde que los datos en un programa se almacenan en variables y éstas son posiciones de memoria, pues bien, como sabemos en la memoria del computador la información (por ejemplo, un número) se guarda en una representación binaria; sin embargo, cuando abrimos un archivo de texto no vemos esta representación binaria sino el número como lo entendemos, en otras palabras, una secuencia de dígitos. Entienda entonces que cuando en un programa se hace un asignación de la forma **a = 93751;** donde "**a**" es una variable definida de tipo entero (**int**), lo que se hace es colocar en la posición de memoria relacionada a la variable **a**, la representación binaria del número, esto es el valor de **0011 0111 0110 1110 0000 0001 0000 0000** (representación de 4 bytes), luego cuando se envía el contenido de esta variable al archivo de textos, se toma esta

representación binaria, se transforma en la cadena '**93751**' y cada carácter que la conforma es colocado en el archivo. Observe que en la variable se emplean 4 bytes para almacenar el número sin embargo al archivo se envían 5 caracteres o bytes; si el número hubiera sido **751**, al archivo sólo se enviarían 3 bytes y si fuera **193751** se enviarían 6 bytes.

Cuando se hace el proceso inverso, es decir cuando se lee información del archivo, el sistema toma uno a uno los caracteres del archivo los convierte a una representación binaria y finalmente lo almacena en la variable. En el ejemplo anterior, se toman los 5 caracteres y se transforman en 4 bytes.

En los archivos binarios, la información no se transforma cuando es colocada en el archivo. La representación binaria del dato, tal como se encuentra en la memoria asignada a la variable, es llevada y almacenada en el archivo. Esta es una diferencia muy fuerte con respecto a los archivos de texto que influirá de manera significativa en la forma cómo se accederá a la información del archivo. Una de las cosas que podemos apreciar en esta forma de almacenar los datos en el archivo es que cuando llevamos un valor, por ejemplo, de una variable de tipo **int**, serán los 4 bytes que conforman la variable lo que se enviarán al archivo, sin importar el número de cifras que tenga el valor. Esto es, si el número fuera **93751**, la información almacenada en la variable que será llevada al archivo será **0011 0111 0110 1110 0000 0001 0000 0000**, si fuera **193751** lo que se almacenará será **1101 0111 1111 0100 0000 0010 0000 0000**, y si el número fuera **751** lo que se almacenará en el archivo será **1110 1111 0000 0010 0000 0000 0000 0000**, 4 bytes en todos los casos.

Diferencias funcionales entre un archivo de textos y uno binario

La forma cómo se almacena la información en los archivos afectará en gran medida la forma cómo se accederán a ellos, a continuación, presentaremos estos aspectos:

Separación de los datos:

Dado que los datos en un archivo de texto son secuencias de caracteres y por lo tanto la cantidad de bytes para almacenar un valor dependerá de la cantidad de cifras que tiene el valor, se deben definir separadores (caracteres especiales) que permitan al sistema saber dónde empieza y dónde termina el dato. Estos separadores son el espacio en blanco (' '), el tabulador ('\t') y el cambio de línea ('\n').

Por otro lado, en los archivos binarios lo que se guarda es la misma secuencia de bytes que tiene la representación numérica del dato en la memoria y que la cantidad de bytes que se guarda en el archivo depende del tipo de dato y no de la cantidad de cifras que tiene el número. Por esta razón, los archivos binarios no requieren de separadores, si se guarda en el archivo un valor entero (**int**), para recuperarlo sólo se requiere extraer del archivo 4 bytes para tener el número completo, luego los siguientes 4 bytes que se encuentren en el archivo pueden perfectamente corresponder al siguiente dato sin la necesidad de separadores. En un archivo de

textos la única forma de detectar que se culminó la lectura de un valor entero será cuando se detecte el separador.

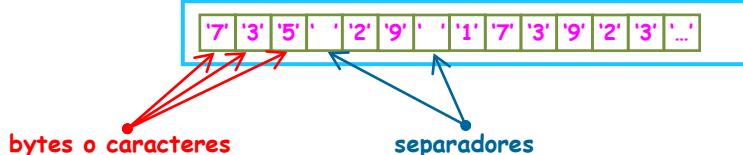
A continuación, se muestran estas diferencias:

```
int a, b, c;
a = 735;
b = 29;
c = 173923;
...
```

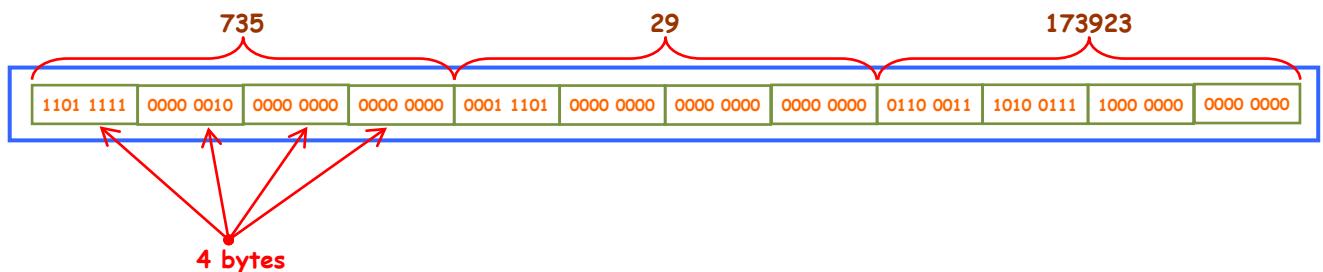
En memoria:

a	11011111 00000010 00000000 00000000
b	00011101 00000000 00000000 00000000
c	01100011 10100111 10000000 00000000

Al enviar los datos a un archivo de textos, la información quedará de la siguiente manera:



Al enviar los datos a un archivo binario, la información quedará así:



Acceso a los datos:

En un archivo de texto, la única forma de acceder a los datos es de manera secuencial, la razón es muy simple, si los dos primeros datos de un archivo de textos no son relevantes pero el tercero sí, para poder leer el tercer dato se requiere leer antes los otros dos, esto porque no hay manera de saber la ubicación del tercer dato para poder colocarse allí y leerlo, se debe leer el primer dato y una vez que se detecte el separador se sabrá que se ha leído el primer valor, de igual manera para el segundo, y recién allí se sabrá que se está listo para leer el tercer dato. Por lo tanto, en un archivo de textos no se puede leer un dato sin antes haber leído los anteriores, a esto se le denomina "**Acceso Secuencial**". La forma cómo se almacenó los datos limita, pues, esta acción.

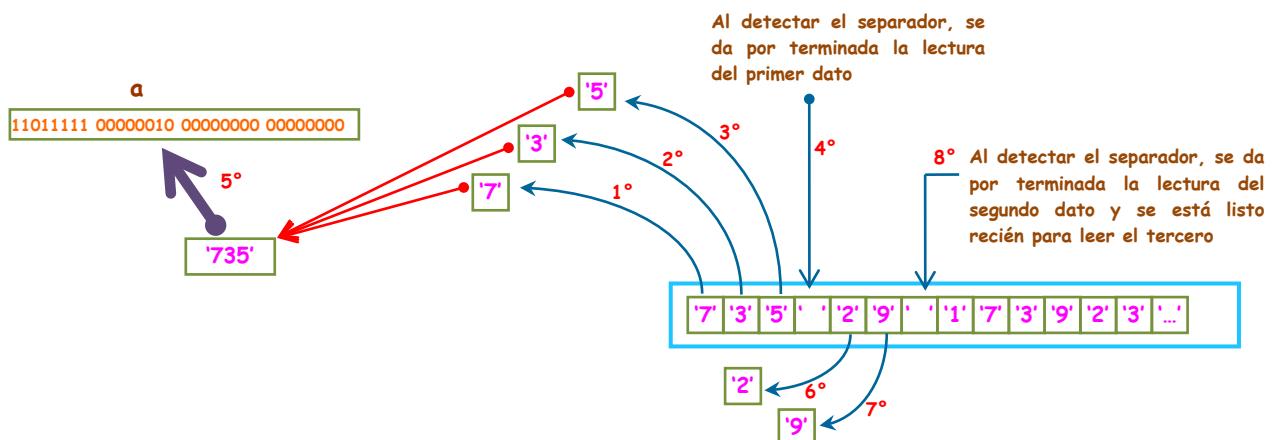
En los archivos binarios, como el espacio almacenado por los datos depende del tipo de dato y no del número de cifras, es muy fácil poder calcular la posición en que se ubica el dato que nos interesa obtener, por ejemplo, si tenemos almacenados en un archivo una serie de números enteros y queremos extraer el tercer valor, sólo debemos calcular cuántos bytes hay desde el inicio del archivo hasta el inicio del dato. En este caso hay dos valores antes del dato, que, multiplicado por 4 bytes, que es lo que ocupa un entero en memoria, nos dan 8 bytes. Entonces no tendremos que leer los dos datos

que se encuentran antes del que nos interesa, sólo debemos desplazarnos 8 bytes desde el inicio del archivo para ubicarnos en el dato que nos interesa y leerlo sin tocar los anteriores. Esta forma de acceder a los datos de un archivo se denomina "**Acceso Directo**".

En resumen, se puede afirmar que en los archivos de textos los datos se acceden de manera secuencial, sin embargo, en los archivos binarios los datos se pueden acceder de manera secuencial, pero también de manera directa.

A continuación, se presenta gráficamente la manera cómo se extrae la información de los archivos:

En los archivos de textos, la única forma de acceder la información es de manera secuencial:



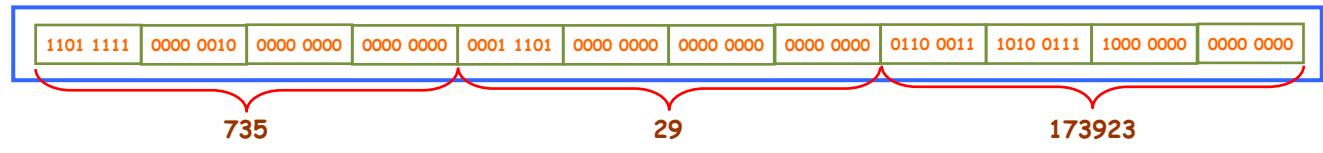
En los archivos binarios, se puede calcular la posición del dato en el archivo, desplazar el indicador del archivo y luego proceder con su lectura sin haber leído los datos que lo preceden:

Al abrirlo, el indicador del archivo se coloca al inicio.

Se calcula la posición del dato en el archivo (2×4 bytes = 8 bytes) y se desplaza el indicador del archivo

3°

Ya se está listo para leer el dato sin haber leído los anteriores.



Actualización de datos:

La actualización de datos en un archivo se refiere a modificar la información que se tiene guardada en el archivo. Para poder realizar esto se requiere de tres pasos, primero leer el dato que se quiere modificar del archivo, luego se tiene que actualizar o modificar el dato en memoria y finalmente volverlo a guardar en el archivo.

Una vez leído un dato de un archivo, el indicador del archivo está listo para leer el siguiente valor, ya que está colocado en el byte siguiente al dato leído. Para poder modificar el dato leído y finalmente guardar la modificación en el archivo, se tiene que retroceder el indicador del archivo y colocarlo al inicio del dato leído, luego se debe

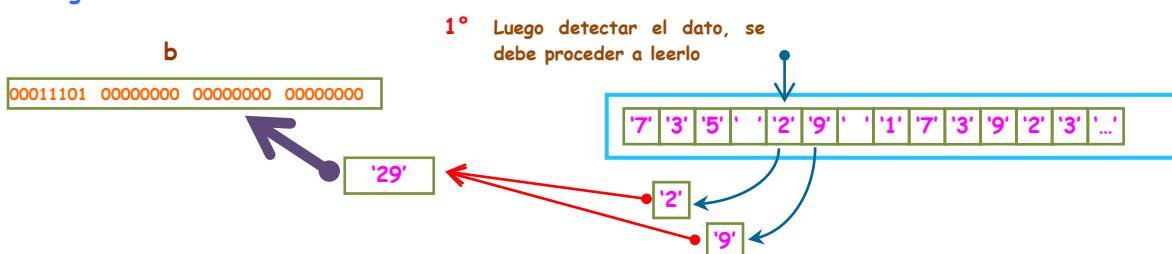
proceder a guardar el dato en el archivo a partir de esa posición. Aquí es donde se presenta un problema en los archivos de textos, imagínese que el dato leído tiene dos cifras y que luego de su modificación el número quedara con cinco dígitos. Resultaría que los dos primeros dígitos del número ocuparían las dos posiciones originales del número, el tercero se colocaría en la posición del separador y las dos últimas se colocarían en las posiciones de las dos primeras cifras del siguiente dato, el resultado sería un daño irreparable al archivo.

Este problema no se presenta con los archivos binarios por lo que ya se ha comentado, la información en el archivo no depende del número de cifras, luego no importa si el número crece o se reduce, siempre ocupará el mismo espacio. Por lo tanto, en este tipo de archivo la actualización de datos se puede realizar sin complicación alguna.

Son por estas razones que algunos lenguajes de programación no permiten realizar actualizaciones en los archivos de texto, sin embargo, en el caso de lenguajes como el C y el C++ esto sí se permite, pero dejan al programador la responsabilidad de controlar los errores que se puedan producir.

A continuación, se ilustra este proceso:

En los archivos de textos, las acciones que se deberían seguir para actualizar un dato serían las siguientes:

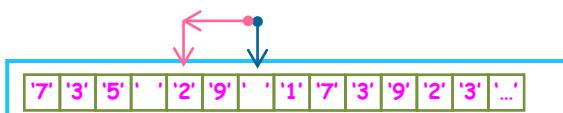


- 1º Luego detectar el dato, se debe proceder a leerlo
- 2º A continuación, se debe proceder a modificar el dato

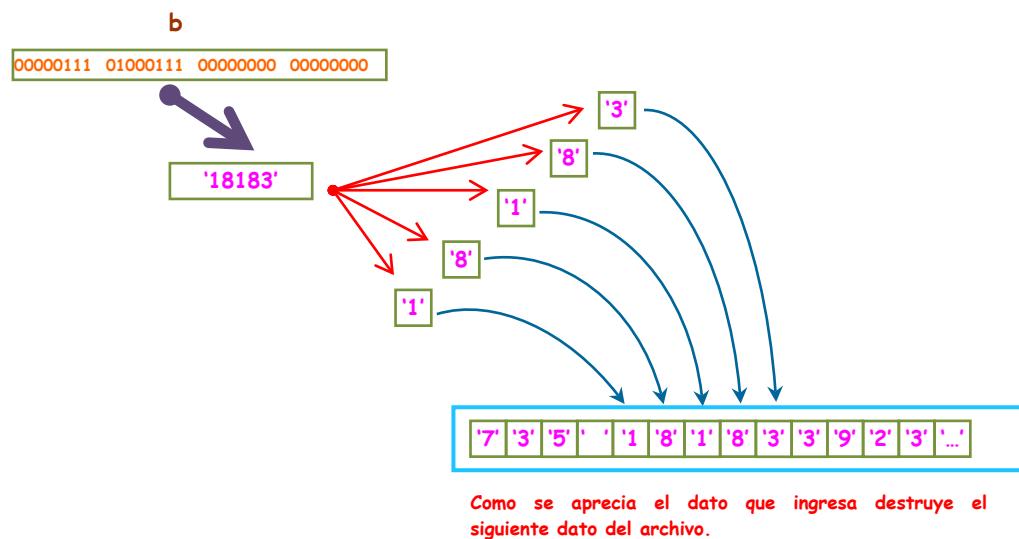
b *= 627; // b ← 18183

b
`00000111 01000111 00000000 00000000`

- 3º Luego se debe, de alguna forma, regresar el indicador del archivo para colocarse nuevamente en la posición del dato leído

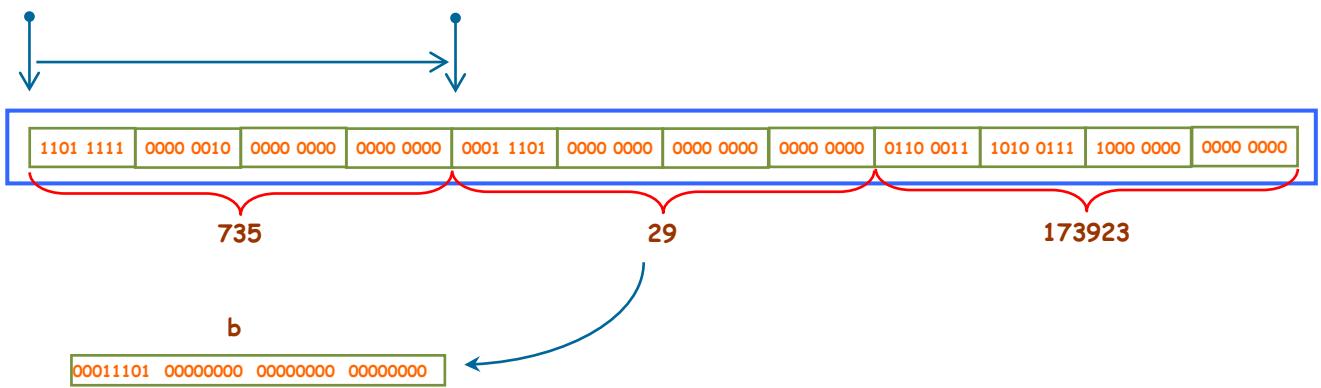


- 4 Finalmente se debe proceder a grabar el dato modificado en el archivo



En los archivos binarios las acciones son similares, pero aquí no hay peligro de estropear el archivo:

- 1º Luego colocar el indicador del archivo en el dato, se debe proceder a leerlo



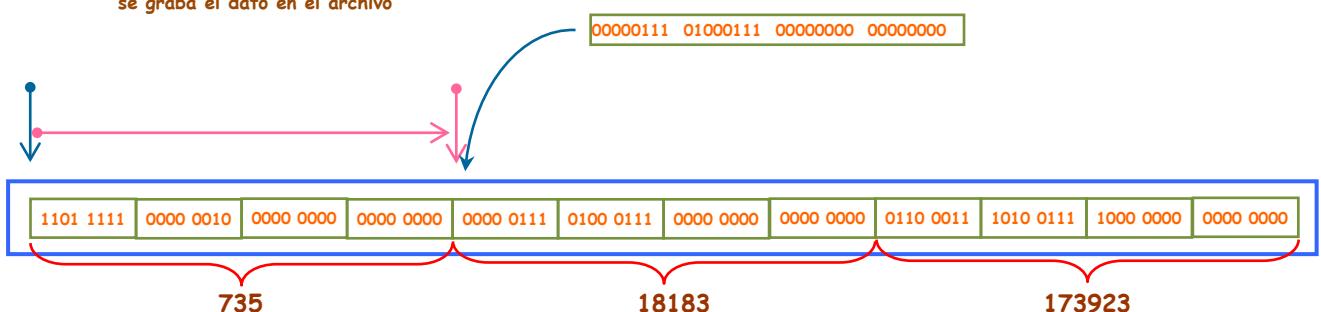
- 2º A continuación, se debe proceder a modificar el dato, igual como se hizo con el archivo de texto

$b := b * 627; // b \leftarrow 18183$

b

00000111 01000111 00000000 00000000

- 3º Finalmente se coloca nuevamente el indicador del archivo en la posición del segundo dato y se graba el dato en el archivo



El dato que ingresa no altera los otros datos del archivo

Para poder explotar estas características funcionales de los archivos binarios, es necesario tener una idea clara de qué tipo de datos vamos a guardar en el archivo y en particular que la información que vayamos a guardar en el archivo sea homogénea en cuanto al tamaño de los datos. Esto último parece trivial porque lo primero que se bien a la mente es lo que se dijo al principio acerca que un valor de tipo *int* o *float* se guardará siempre en 4 bytes¹ y un valor *double* en 8, etc., sin embargo cuando se piensa en esto muchas veces se dejan de lado a las cadenas de caracteres y allí es donde aparecen los problemas. El manejo de memoria dinámica y exacta en el manejo de cadenas de caracteres es muy útil y eficiente cuando estamos hablando de datos en la memoria principal del computador, sin embargo, esto, llevado a un archivo binario, es un grave error ya que si guardamos cadenas de caracteres de diferente longitud en un archivo binario no solo tendremos problemas para leer el archivo, sino que no podremos explotar las características de acceso directo, actualización de datos, etc. Por esta razón, al igual que cuando se trabaja con un arreglo estático, las cadenas de caracteres que se piensen guardar en un archivo deben definirse en un tamaño fijo y a la hora de llevarla al archivo binario, debe guardar íntegramente los elementos de esta cadena, así no se haya llenado la totalidad de los elementos de la cadena.

Definición y apertura de variables de archivo:

La definición de variables de archivo no cambia no cambia significativamente, pero hay detalles que hay que tomar en cuenta debido a que los archivos binarios pueden realizar operaciones diferentes a la que se realizan en un archivo de texto. A continuación, mostramos estos detalles:

Apertura para leer datos de un archivo binario

```
ifstream archBin("nombreDelArchivo.bin", ios::in|ios::bin);
if (not archBin.is_open()){
    cout<<left<<"ERROR: No se pudo abrir el archivo nombreDelArchivo.bin"<<endl;
    exit(1);
}
```

Este proceso abre un archivo binario para leer datos de él, el archivo debe existir, de lo contrario se levantará una bandera de error por lo que el programa se debe interrumpir. Note que la diferencia radica en la extensión ".bin" del nombre del archivo y en la operación "|ios::bin".

Apertura para escribir datos en un archivo binario

```
ofstream archBin("nombreDelArchivo.bin", ios::out|ios::bin);
if (not archBin.is_open()){
    cout<<left<<"ERROR: No se pudo abrir el archivo nombreDelArchivo.bin"<<endl;
    exit(1);
}
```

Igual que el anterior.

¹ Es posible que en futuras versiones de C/C++ estos valores puedan cambiar, pero por ahora esos valores son un estándar.

Apertura para actualizar los datos en un archivo binario

```
fstream archBin("nombreDelArchivo.bin", ios::in|ios::out|ios::bin);
if (not archBin.is_open()){
    cout<<left<<"ERROR: No se pudo abrir el archivo nombreDelArchivo.bin"<<endl;
    exit(1);
}
```

Aquí se presentan mayores diferencias ya que en los archivos de textos es muy complicado realizar actualizaciones de datos en archivos y por lo tanto no se realizan. Vemos que, por actualización se entiende por *leer* un dato del archivo, modificarlo y finalmente *escribirlo* en el mismo lugar, por esta razón, la variable de archivo no puede ser *ifstream* (porque no se va a leer únicamente) ni *ofstream* (porque no se va a escribir únicamente), por lo que se usa un tipo de dato neutro como *fstream*. Por otro lado, como se va realizar las dos operaciones, el modo de apertura debe ser *ios::in|ios::out|ios::bin*.

Debe entender que solo se puede actualizar un archivo que tenga datos, por lo que, si el archivo no existe, no se le podrá abrir de este modo, primero habrá que crearlo, escribiendo datos en él, cerrarlo y luego volverlo a abrir en este modo.

Entrada y Salida de datos de un archivo binario

Se presentan aquí grandes diferencias con respecto a los archivos de texto. A continuación, se indicarán estas diferencias.

Los operadores *<<* y *>>* ya no se emplearán para la impresión y lectura de los archivos binarios. Esto se debe a que al realizar la operación *arch << ...*, se toma la representación binaria del dato en memoria y la transforma en una cadena de caracteres, mientras que en la operación *arch >> ...*, toma los caracteres del archivo y los transforma a una representación binaria. Los archivos binarios, como ya se indicó, no transforman la información.

Los métodos que se emplearán para la lectura e impresión de datos en archivos binarios serán *read* y *write*, cuya sintaxis y modo de empleo es completamente diferente a las empleadas en archivos de textos.

El método *read* se emplea de la siguiente manera:

```
archBin.read(reinterpret_cast<char*>(&var), sizeof(T));
```

En donde *reinterpret_cast<char*>* es un operador que busca poder manipular la información contenida en la variable *var*, *bute por byte*. La expresión *(&var)* le indicará al método *read* la dirección de memoria donde colocará debe colocar la información que leerá del archivo, la operación *sizeof(T)* le indicará la cantidad de bytes que tomará del archivo, ahí se puede colocar una expresión más compleja como por ejemplo *sizeof(T)*n*.

Si observa bien, verá que se trata de una operación muy versátil, ya que solo requiere una dirección de memoria y una cantidad de bytes y con esa información es capaz de leer en una operación cualquier tipo de dato, desde un entero hasta una estructura

(*struct*), lo más compleja que se le pueda ocurrir, incluso podrá llenar un arreglo, de cualquier tipo en una sola operación.

El método *write* tiene una sintaxis similar a la de *read*, ésta es:

```
archBin.write(reinterpret_cast<const char*>(&var), sizeof(T));
```

Como se observa la única diferencia es la del operador *const*. Este método toma de la memoria del computador, indicada por la dirección de la variable *&var*, uno a uno los bytes que se desean guardar y los lleva a archivo. Esta operación la hace sin fijarse en el tipo de dato que está transfiriendo, por esta razón la función requiere del operador *reinterpret_cast*. El uso del operador *const* se debe porque el método no modifica el contenido de la variable *var*. Como se verá más adelante, esto hace igualmente versátil el empleo de esta función.

Acceso secuencial a un archivo binario

Como hemos dicho, un archivo binario puede manejarse tanto de manera secuencial como de manera directa, en este punto analizaremos el primer caso ya que es la forma natural de acceso a todo tipo de archivo y porque es la más sencilla.

El siguiente programa muestra la manera cómo llenar un archivo con valores enteros de manera secuencial para luego leerlos del archivo binario y mostrarlos. Hay que tener en cuenta que como lo que se guarda en un archivo binario es una copia exacta del dato en memoria (representación binaria), no se puede crear un archivo binario mediante un editor de palabras como sí se puede hacer con los archivos de texto. Sólo se podrá crear un archivo binario mediante un programa. Tampoco se podrá entender la información del archivo binario si se abre el archivo con un editor de palabras. Los datos se leerán desde un archivo de textos como el que se muestra a continuación:

12	14	10	8				
1034	1253	1177	90731	921	501	1891	1025
15	16	12	14	18	15		
1407	1503						
251	512	973	1015	6002			
18	15	17	16	18	19	20	
...							

El programa para leer esos datos y que los guarde en un archivo binario se muestra a continuación:

```
#include "FuncionesDeArchivosBinarios.h"

int main(int argc, char** argv) {
    crearArchivoBinario("DatosEnteros.txt","DatosEnteros.bin");
    mostrarContenidoDeArchivoBinario("DatosEnteros.bin");
    return 0;
}

#ifndef FUNCIONESDEARCHIVOSBINARIOS_H
#define FUNCIONESDEARCHIVOSBINARIOS_H

void crearArchivoBinario(const char *,const char *);
void mostrarContenidoDeArchivoBinario(const char *);

#endif /* FUNCIONESDEARCHIVOSBINARIOS_H */

#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;
```

```

#include "FuncionesDeArchivosBinarios.h"

void crearArchivoBinario(const char *nombArchDatos, const char *nombArchBin) {
    ifstream archDatos(nombArchDatos, ios::in);
    if(not archDatos.is_open()){
        cout << left << "ERROR: No se pudo abrir el archivo: "<<nombArchDatos<<endl;
        exit(1);
    }

    ofstream archBin(nombArchBin, ios::out);
    if(not archBin.is_open()){
        cout << left << "ERROR: No se pudo abrir el archivo: "<<nombArchBin<<endl;
        exit(1);
    }
    int dato;
    while(true){
        archDatos >> dato;
        if(archDatos.eof())break;
        archBin.write(reinterpret_cast<const char*>(&dato), sizeof(int));
    }
}

void mostrarContenidoDeArchivoBinario(const char *nombArchBin) {
    ifstream archBin(nombArchBin, ios::in);
    if(not archBin.is_open()){
        cout << left << "ERROR: No se pudo abrir el archivo: "<<nombArchBin<<endl;
        exit(1);
    }
    int dato;
    while(true){
        archBin.read(reinterpret_cast<char*>(&dato), sizeof(int));
        if(archBin.eof())break;
        cout << right << setw(8) << dato;
    }
    cout << endl;
}

```

Al ejecutar este programa podremos apreciar en la pantalla los valores que se almacenaron en el archivo.

Una segunda aplicación que presentamos a continuación consistirá en la creación de un archivo binario en la que coloquemos mayor cantidad de información. En este caso los datos los vamos a sacar de un archivo de textos en donde se encuentre la información de un grupo de personas que podrían ser los empleados de una compañía. En este archivo de textos tendrá un formato de tipo *CSV* (*Comma Separated Values*) y en cada línea tendremos el DNI (valor entero), el nombre (cadena de caracteres) y el sueldo (valor de punto flotante), el archivo es similar al que se muestra a continuación:

63540121,Perez Garcia Juan Carlos,3756.20
73991124,Saavedra Gomez Maria Luisa,7600.00
17229900,Li Wong Ada, 2890.75
...

Como se indicó, el programa leerá la información de este archivo de textos y lo guardará de manera secuencial en otro archivo, pero bajo el formato binario.

Para poder implementar este programa debemos tomar en cuenta las características de los archivos binarios, de modo que luego se le pueda explotar al máximo. En este sentido, lo primero que debemos considerar es que cada dato que se guarde en el archivo binario debe tener un tamaño uniforme en bytes, por eso debemos manejar la

información de una persona como una unidad (entidad), esto es como una estructura (*struct*) compuesto por tres campos (dni, nombre y sueldo).

Una vez decidido que se va a trabajar con una variable de tipo *struct*, se debe solucionar el problema de las cadenas de caracteres. Los campos que definen cadenas de caracteres en una estructura que será empleada para manipular un archivo binario deben definirse obligatoriamente como un arreglo estático, esto garantizará que el texto leído del archivo o grabado en él se haga de manera correcta, si se define como un arreglo dinámico, lo único que se guardará en el archivo es la dirección de memoria donde se encuentra el texto en la memoria RAM del computador y no el texto, por lo que luego no lo podrá recuperar. Por otro lado, el definir la cadena de caracteres como un arreglo estático garantizará que lo que se guarde en el archivo tenga la misma cantidad de bytes, manteniéndose la homogeneidad en el tamaño de los datos.

El siguiente programa contempla estas cosas, observe la forma cómo se define la estructura y sobre todo la manera cómo se debe definir las cadenas de caracteres.

```
#ifndef ESTRUCTURAPERSONA_H
#define ESTRUCTURAPERSONA_H

    struct Persona{
        int dni;
        char nombre[60];
        double sueldo;
    };

#endif /* ESTRUCTURAPERSONA_H */
#include "FuncionesDeArchivosBinarios.h"

int main(int argc, char** argv) {
    crearArchivoBinario("Personal.csv","Personal.bin");
    mostrarContenidoDeArchivoBinario("Personal.bin");
    return 0;
}

#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;
#include "EstructuraPersona.h"
#include "FuncionesDeArchivosBinarios.h"

void crearArchivoBinario(const char *nombArchPersonal,const char *nombArchBin) {
    ifstream archDatos(nombArchPersonal, ios::in);
    if(not archDatos.is_open()){
        cout<<left<<"ERROR: No se pudo abrir el archivo: "<<nombArchPersonal<<endl;
        exit(1);
    }

    ofstream archBin(nombArchBin, ios::out|ios::binary);
    if(not archBin.is_open()){
        cout <<left<<"ERROR: No se pudo abrir el archivo: "<<nombArchBin<<endl;
        exit(1);
    }
    struct Persona persona;
    while(true){
        leePersona(archDatos,persona);
        if(archDatos.eof())break;
        archBin.write(reinterpret_cast<const char*>(&persona),sizeof(struct Persona));
    }
}
```

```

void leePersona(ifstream &archDatos, struct Persona &persona){
    archDatos >> persona.dni;
    if(archDatos.eof())return;
    archDatos.get();
    archDatos.getline(persona.nombre,60,',');
    archDatos>>persona.sueldo;
}

void mostrarContenidoDeArchivoBinario(const char *nombArchBin){
    ifstream archBin(nombArchBin, ios::in|ios::binary);
    if(not archBin.is_open()){
        cout << left << "ERROR: No se pudo abrir el archivo: "<<nombArchBin<<endl;
        exit(1);
    }
    struct Persona persona;
    cout.precision(2);
    cout<<fixed;
    cout<<sizeof(struct Persona)<<endl;
    while(true){
        archBin.read(reinterpret_cast<char*>(&persona), sizeof(struct Persona));
        if(archBin.eof())break;
        cout<<left<<setw(10)<<persona.dni<<setw(35)<<persona.nombre
            <<right<<setw(10)<<persona.sueldo<<endl;
    }
    cout<<endl;
}

```

Acceso directo a un archivo binario

Hasta ahora el manejo que hemos hecho de los archivos binarios difiere muy poco de lo que hicimos con los archivos de textos. Por un lado, hemos leído uno a uno los datos y los hemos guardado uno a continuación del otro en el archivo binario, y, por otro lado, lo que hemos hecho es leer los datos del archivo binario secuencialmente y los hemos mostrado en la pantalla. Esto se hace porque la manera natural de manejar cualquier tipo de archivo es de manera secuencial y por lo tanto no tenemos que hacer algo extraordinario para poder acceder de esa forma a los datos de un archivo, cualquiera sea su formato.

En este punto, lo que vamos a mostrar es otra forma de acceder a los datos de un archivo, este modo se denomina "acceso directo". Esta manera de acceso no es exclusiva de los archivos binarios, sin embargo, el emplearla en otro tipo de archivo como en los archivos de texto implicará realizar operaciones muy complejas que se escapan a la finalidad de este curso, por eso el acceso directo a archivos lo circunscribiremos sólo a archivos binarios.

Una de las cosas en la que debemos enfocarnos para realizar este tipo de acceso es en las funciones que permitan realizar esta labor, por eso a continuación las describiremos:

Método seekg

Cuando se abre un archivo, el indicador del archivo se coloca al inicio del mismo, este indicador marca la posición a partir de donde se extraerá o grabará la información del o en el archivo. Conforme se extrae o graba en el archivo, este indicador se va desplazando hacia delante de modo que cuando se termina la operación, el indicador del archivo queda listo para realizar otra operación a partir del siguiente registro.

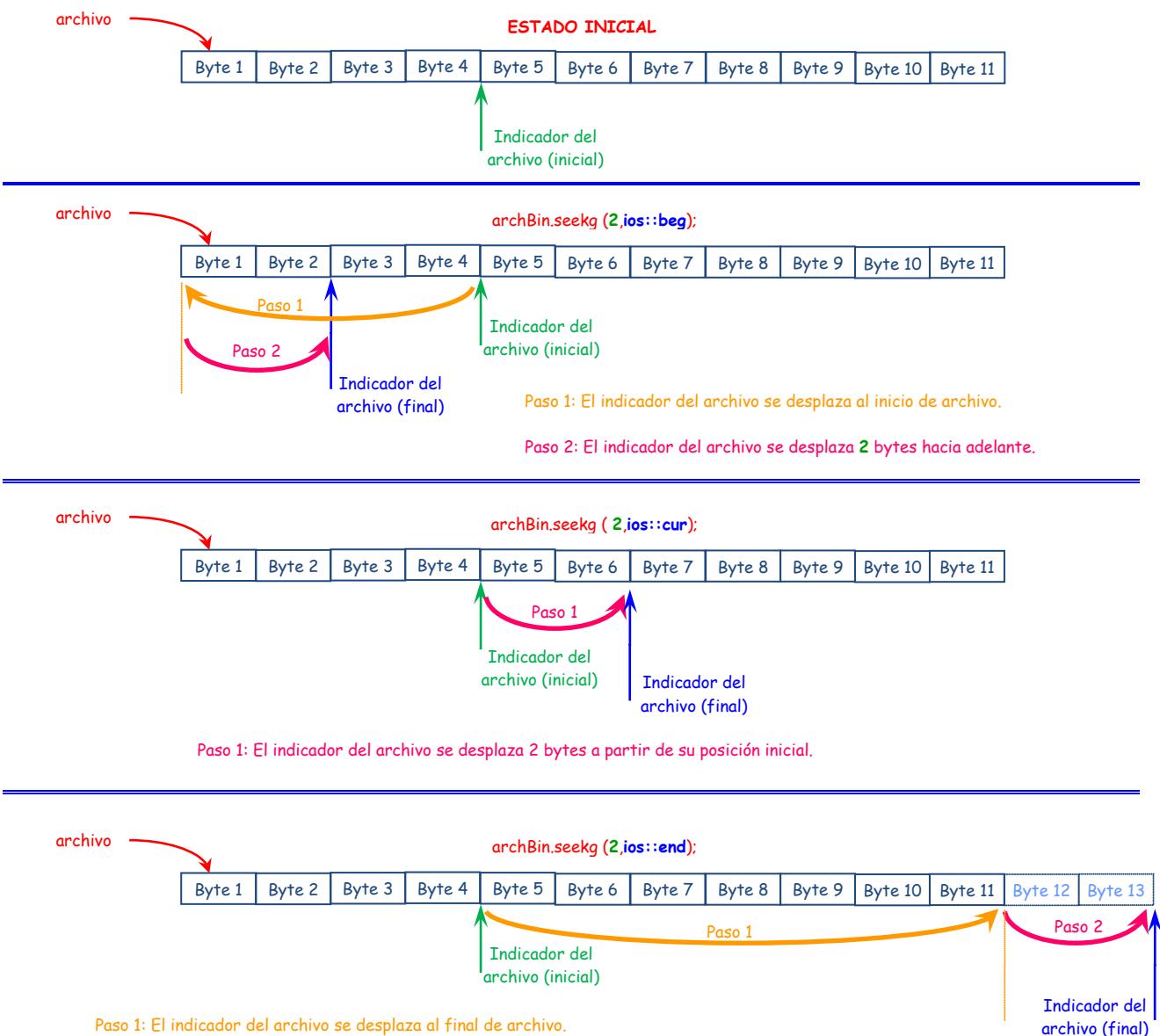
El método `seekg` permite desplazar el indicador del archivo a voluntad en el archivo.

El método `seekg` tiene la siguiente sintaxis:

```
archBin.seekg(n, ios::beg);  
archBin.seekg(n, ios::cur);  
archBin.seekg(n, ios::end);
```

Donde `archBin` es el nombre de la variable de archivo que queremos procesar, `n` es un valor entero que indica la cantidad de bytes que queremos desplazar el indicador del archivo, `ios::beg` indica que el desplazamiento se realizará desde el inicio del archivo, `ios::cur` indica que el desplazamiento se realizará desde la posición actual del indicador del archivo y la constante `ios::end` indica que el desplazamiento se realizará desde el final del archivo.

La figura siguiente muestra un archivo binario. Si se produjeron operaciones de lectura (o escritura como veremos más adelante), el indicador del archivo se desplazó. A partir de allí se puede observar cómo trabaja la función `fseek`.



Método tellg

El método `tellg` devuelve la cantidad de bytes que hay entre el inicio del archivo y la posición del indicador del archivo.

El método `tellg` tiene la siguiente sintaxis:

`n = archBin.tellg();`

La figura siguiente muestra cómo trabaja el método `tellg`:



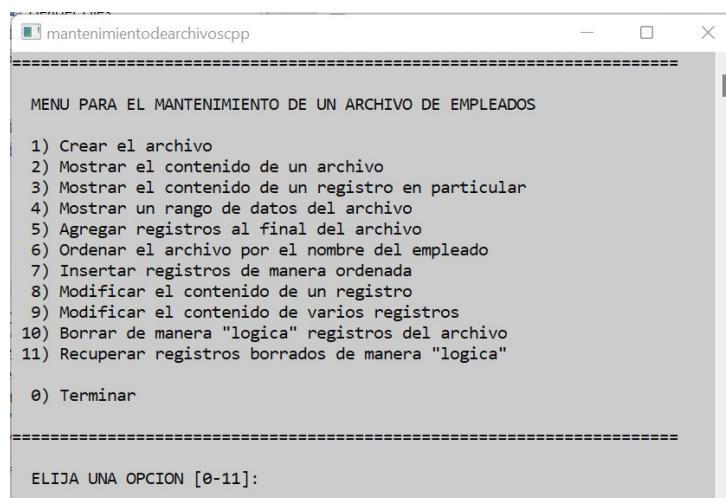
Aplicaciones que emplean archivos binarios

Mantenimiento de archivos

La aplicación que vamos a desarrollar a continuación se concentrará en la realización de diversas operaciones en un archivo binario, la idea es que en él podamos crear un archivo, lo podamos modificar, actualizando los registros, insertando nuevos registros o eliminándolos, también lo ordenaremos y realizaremos diversos reportes.

Como la aplicación abordará muchas tareas dentro de un archivo, la forma como se desarrollará el programa estará dada en función de un menú de opciones. La idea es mostrar en la pantalla la lista de tareas que podrá realizar el programa, luego el usuario podrá elegir cualquiera de ellas y en el orden que desee.

La figura siguiente muestra lo que se quiere hacer:



El código para la elaboración de un menú de opciones es una tarea sencilla, a continuación, se presenta el código que al ser ejecutado muestre el menú de opciones que se ve en la figura anterior.

```

/*
 * Proyecto: MantenimientoDeArchivosCPP
 * Archivo: main.cpp
 * Autor: J. Miguel Guanira E.
 *
 * Created on 16 de agosto de 2022, 08:26 PM
 */

```

```

#include "BibMantenimientoDeArchivos.h"

int main(int argc, char** argv) {
    menuDeOpciones(); // El programa principal, sólo llamará al
                        // procedimiento que mostrará el menú
    return 0;
}

```

Mostrar un menú de opciones no implica sólo imprimir la lista de opciones, esta impresión se debe hacer dentro de una iteración en la que el usuario vea las opciones, elija una de ellas y la ejecute. Una vez ejecutada la opción se debe repetir todas las acciones inclusive la impresión del menú. Esto último debido a que, al ejecutar una opción, el reporte que se puede generar podría impedir que se vean las opciones del menú.

La iteración hará que el usuario pueda trabajar con todas las tareas que quiera realizar sin tener que ejecutar el programa por cada una de ellas.

Finalmente, una de las opciones del menú permitirá al usuario dar por terminada la ejecución del programa.

```

/*
 * Proyecto: MantenimientoDeArchivosCPP
 * Archivo: BibMantenimientoDeArchivos.cpp
 * Autor: J. Miguel Guanira E.
 *
 * Created on 16 de agosto de 2022, 08:31 PM
 */

```

```

#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;
#include <cstring>
#include <cctype>
#include "EstructuraPersona.h"
#include "BibMantenimientoDeArchivos.h"
#define MAX_CAR_LIN 70

void menuDeOpciones() {
    int opcion;
    cout << endl << endl;

```

```

while(true){
    imprimeLinea('=', MAX_CAR_LIN);
    cout<<endl;
    cout<<
        " MENU PARA EL MANTENIMIENTO DE UN ARCHIVO DE EMPLEADOS"
        <<endl<<endl;
    cout<<" 1) Crear el archivo"<<endl;
    cout<<" 2) Mostrar el contenido de un archivo"<<endl;
    cout<<" 3) Mostrar el contenido de un registro en particular"
        <<endl;
    cout<<" 4) Mostrar un rango de datos del archivo" <<endl;
    cout<<" 5) Agregar registros al final del archivo"<<endl;
    cout<<" 6) Ordenar el archivo por el nombre del empleado"
        <<endl;
    cout<<" 7) Insertar registros de manera ordenada" <<endl;
    cout<<" 8) Modificar el contenido de un registro" <<endl;
    cout<<" 9) Modificar el contenido de varios registros" <<endl;
    cout<<"10) Borrar de manera \"logica\" registros del archivo"
        <<endl;
    cout<<"11) Recuperar registros borrados de manera \"logica\""
        <<endl;
    cout<<endl<<" 0) Terminar"<<endl<<endl;
    imprimeLinea('=', MAX_CAR_LIN);
    while(1){
        cout<<endl;
        cout<<"  ELIJA UNA OPCION [0-11]: ";
        cin >> opcion;
        if (cin.fail() or opcion < 0 or opcion > 11){
            cout<<endl;
            cout<<"  ERROR: NO INGRESO UNA OPCION VALIDA"
                <<endl<<endl;
            imprimeLinea('=', MAX_CAR_LIN);
            cin.clear();
            while(cin.get() != '\n'); //Limpia el buffer de entrada
        }
        else {
            while(cin.get() != '\n'); //Limpia el buffer de entrada
            break;
        }
    }
    if(opcion == 0) break;
    else ejecutarOpcion(opcion);
}
imprimeLinea('=', MAX_CAR_LIN); // Ve implementaciones al final del proyecto
cout<<endl;
cout<<"FIN DEL PROGRAMA"<<endl<<endl;
imprimeLinea('=', MAX_CAR_LIN);
}

```

Una vez que el usuario elija una opción válida, el programa llamará a otro procedimiento que ejecute selectivamente la opción deseada

```

void ejecutarOpcion(int opcion) {
    switch (opcion) {
        case 1:
            crearArchivo();
            break;
        case 2:
            mostrarArchivo();
            break;
        case 3:
            mostrarRegistro();
            break;
        case 4:
            mostrarRangoDeRegistros();
            break;
        case 5:
            agregarAlFinalDelArchivo();
            break;
        case 6:
            ordenarArchivo();
            break;
        case 7:
            insertarOrdenado();
            break;
        case 8:
            modificaRegistro();
            break;
        case 9:
            modificaRegistros();
            break;
        case 10:
            eliminacionLogicaDeRegistros();
            break;
        case 11:
            recuperacionDeRegistros();
    }
}

```

Como se ha podido observar no se han colocado argumentos a las funciones de las opciones, esto porque el C++ define un tipo de dato diferente de acuerdo con la operación que se desee realizar, por esta razón no podemos pasar la variable de archivo como parámetro. Por otro lado, esta forma permite que el programa sea más didáctico y no se tenga que ver una opción para entender otra. Esta decisión traerá como consecuencia que se tenga que repetir parte del código en los diferentes procedimientos, pero como he dicho, el que se pueda entender mejor estos procesos es la prioridad en este ejemplo.

Primero definiremos la estructura de datos a emplear en el programa:

```

/*
 * Proyecto: MantenimientoDeArchivosCPP
 * Archivo: EstructuraPersona.h
 * Autor: J. Miguel Guanira E.
 *
 * Created on 16 de agosto de 2022, 08:35 PM
 */

#ifndef ESTRUCTURAPERSONA_H
#define ESTRUCTURAPERSONA_H

struct Persona{
    bool activo; // Indicará si el registro está activo (true) o se borró
                 // de manera lógica (false)
    int dni;
    char nombre[60];
    double sueldo;
};

#endif /* ESTRUCTURAPERSONA_H */

```

Opción 1: Crear el archivo

La creación de un archivo binario ya se vio en un ejemplo anterior, así que aquí sólo se presentará el código.

```

void crearArchivo(){
    struct Persona persona;;
    char sn;

    imprimeLinea('=',70);
    cout<<endl<<endl;
    cout<<" ESTA OPCION INICIALIZARA EL ARCHIVO BINARIO" <<endl<<endl;
    cout<<" Desea continuar? (S/N): ";
    while(1){
        sn = cin.get();
        sn = toupper(sn);
        while(getchar() != '\n'); //Limpia el buffer de entrada
        if (sn != 'S' and sn != 'N'){
            cout<<endl<<" Solo puede ingresar s/S/n/N"<<endl;
            cout<<" Desea continuar? (S/N): ";
        }
        else break;
    }
    if (sn == 'S') {
        ifstream archTxt("Personal.csv",ios::in);
        if(not archTxt.is_open()){
            cout<<"ERROR: No se pudo abrir el archivo Personal.csv" <<endl;
            exit(1);
        }
        ofstream archBin("Personal.bin",ios::out|ios::binary);
    }
}

```

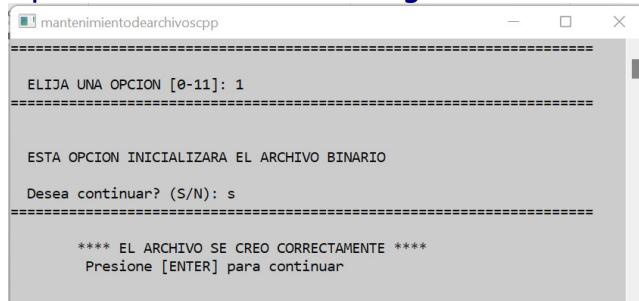
```

        if(not archTxt.is_open()){
            cout<<"ERROR: No se pudo abrir el archivo Personal.bin" <<endl;
            exit(1);
        }
        persona.activo = true; //Todos los registros estarán activos inicialmente
        while (1){
            leerPersona(archTxt,persona);
            if(archTxt.eof())break;
            archBin.write(reinterpret_cast<const char*>(&persona)
                          ,sizeof(struct Persona));
        }
        imprimeLinea('=',70);
        cout<<endl;
        cout<<right<<setw(50)<<"**** EL ARCHIVO SE CREO CORRECTAMENTE ****"
            <<endl;
        cout<<right<<setw(40)<<"Presione [ENTER] para continuar" <<endl;
        cin.get();
    }
}

void leerPersona(ifstream &archTxt,struct Persona &persona){
    archTxt>>persona.dni;
    if(archTxt.eof())return;
    archTxt.get();
    archTxt.getline(persona.nombre,60,',');
    archTxt>>persona.sueldo;
}

```

La salida, eliminando la parte del menú, será la siguiente:



Opción 2: Mostrar el contenido de un archivo

Aquí sólo hay que mostrar los datos de manera tabular, cosa que ya se hizo en otro ejemplo. Lo que sí se debe resaltar aquí, es que en el código se verifica si el registro fue borrado de manera lógica. El borrado lógico de un registro se hace porque por alguna razón se decide que se ha de eliminarse del archivo, pero si se elimina de manera física estos datos ya no se podrán recuperar si se cometió un error o el empleado, en este caso, se reincorpora luego de un tiempo. En este sentido esta opción no imprimirá los registros borrados de manera lógica (con el campo "activo" en *falso*). Al final se indicará si el archivo tiene registros borrados.

```

void mostrarArchivo(){
    ifstream archBin("Personal.bin",ios::in|ios::binary);
    if(not archBin.is_open()){
        cout<<"ERROR: No se pudo abrir el archivo Personal.bin" <<endl;
        exit(1);
    }
    struct Persona persona;;
    int i=0;
    cout.precision(2);
    cout<<fixed;
    cout << endl;
    imprimeLinea('=',70);
    cout<<right<<setw(4)<<"No."<<setw(6)<<"DNI"
        <<setw(12)<<"NOMBRE"
        <<setw(43)<<"SUELDO"<<endl;
    imprimeLinea('=',70);
    while(1){
        archBin.read(reinterpret_cast<char*>(&persona),
                     sizeof(struct Persona));
        if(archBin.eof()) break;
        if(persona.activo){
            cout<<right<<setw(3)<<i+1<<")   "
                <<left<<setw(10)<<persona.dni
                <<setw(40)<<persona.nombre
                <<right<<setw(10)<<persona.sueldo<<endl;
        }
        i++;
    }
    cout<<endl<<endl;
    cout<<setw(40)<<"Presione [ENTER] para continuar";
    getchar();
}

```

La salida de este módulo para un archivo que no ha borrado lógicamente registros será como se muestra a continuación:

No.	DNI	NOMBRE	SUELDO
1)	97514234	RAFFO ARIAS SURAMI ANNIE	2590.42
2)	15523742	DIAZ CORREA IRMA VIRGINIA	347.22
3)	90001464	VALVERDE FLORES IRMA YDALI	1097.12
4)	70022841	CHUMPITAZ IBEROS RONAL	1023.32
5)	27366789	CHUNG LEE ALVARO	2825.37
6)	405590370	PEREZ FARFAN RICHARD SANDRO	6866.48
7)	26358230	CABRERA LUNA SHIRLEY ISABEL	782.57
8)	44660461	PINTO CALIXTO SHIRLEY	3506.80
9)	84018683	ROMERO PALOMARES GLORIA TATIANA	4328.31
10)	88641921	SANTISTEBAN MEZA ROSA	8468.73
11)	22794073	MENDOZA EGUSQUIZA ROSARIO	1846.57
12)	85428483	SALAZAR CORRALES LORENA IRMA	2695.60
13)	37232995	REYES TANG EDWARD	2476.16
14)	48179147	CERNA PAIRAZAMAN FIORELLA MARGOT	5301.58
15)	94424972	FORZANZ CARO ALONSO	3862.59
16)	70719467	SHIRAKAWA MIRANDA MADELEINE	8609.19
17)	16565040	ARROYO GORDILLO MARIA HELI	3831.02
18)	22766834	HERNANDEZ BARRIOS EMILIO	5086.21
19)	82000163	VEGA GARCIA JANET REBECA	2432.14
20)	87895755	SHIRAKAWA ORTEGA ANGEL	3789.89
21)	45968606	WONG MARTEL MADELEINE	6146.95
22)	57823550	DIAZ ANTEZANO MAGALI SILVANA	5389.56
23)	53219729	MEJIA DIAZ BLANCA MILAGROS	4291.45
24)	74294791	RAFFO CHUNG ADRIA GLORIA	7685.69
25)	92136229	CASTRO SUAREZ ROLANDO	2240.96

Presione [ENTER] para continuar

Opción 3: Mostrar el contenido de un registro en particular

En esta opción podremos apreciar las propiedades de acceso directo que tienen los archivos binarios, el usuario aquí podrá ver el contenido de un registro en particular. El código también verificará si el registro fue dado de baja (fue borrado lógicamente). Este procedimiento no sólo se limitará a mostrar un registro, sino que también permitirá ver todos los que desee el usuario.

```
void mostrarRegistro() {
    ifstream archBin("Personal.bin",ios::in|ios::binary);
    if(not archBin.is_open()){
        cout<<"ERROR: No se pudo abrir el archivo Personal.bin" <<endl;
        exit(1);
    }
    struct Persona persona;;
    int n, tamReg, tamArch, numReg;
    char s;
    tamReg = sizeof(struct Persona);
    datosDelArchivo(archBin,tamReg, tamArch, numReg);
    // Ve implementacíos al final del proyecto
    while (1){
        cout<<endl<<"Ingrese el numero de registro que quiere ver (entre 1 y "
           <<numReg<<") : ";
        cin>>n;
        if (n >= 1 && n <= numReg) {
            archBin.seekg( (n-1)*tamReg, ios::beg);
            archBin.read(reinterpret_cast<char*>(&persona),
                         tamReg);
            cout<<endl;
            imprimeLinea('=',70);
        }
    }
}
```

```

cout<<right<<setw(4)<<"No. "<<setw(6)<<"DNI"
    <<setw(12)<<"NOMBRE"
    <<setw(43)<<"SUELDO"<<endl;
imprimeLinea('=', 70);
cout<<right<<setw(3)<<n<<" "
    <<left<<setw(10)<<persona.dni
    <<setw(40)<<persona.nombre
    <<right<<setw(10)<<persona.sueldo<<endl;

if (persona.activo==false)
    cout<<endl<<setw(40)
        <<"***** REGISTRO DADO DE BAJA *****"<<endl;
imprimeLinea('=', 70);
while(cin.get() !='\n');
cout<<endl
    <<" Si desea ver otro registro ingrese S, de lo contrario ENTER: ";
s = cin.get();
s = toupper(s);
if (s!='S')break;
}
}
cout<<endl<<endl;
cout<<setw(40)<<"Presione [ENTER] para continuar";
getchar();
}

```

La ejecución de este módulo se presenta a continuación:

```

mantenimientodearchivoscpp
ELIJA UNA OPCION [0-11]: 3
Ingrese el numero de registro que quiere ver (entre 1 y 25): 17
=====
No.   DNI      NOMBRE          SUELDO
=====
17) 16565040  ARROYO GORDILLO MARIA HELI      3831.02
=====

Si desea ver otro registro ingrese S, de lo contrario ENTER: s
Ingrese el numero de registro que quiere ver (entre 1 y 25): 22
=====
No.   DNI      NOMBRE          SUELDO
=====
22) 57823550  DIAZ ANTEZANO MAGALI SILVANA      5389.56
=====

Si desea ver otro registro ingrese S, de lo contrario ENTER: s
Ingrese el numero de registro que quiere ver (entre 1 y 25): 14
=====
No.   DNI      NOMBRE          SUELDO
=====
14) 48179147  CERNA PAIRAZAMAN FIORELLA MARGOT      5301.58
=====

Si desea ver otro registro ingrese S, de lo contrario ENTER:

Presione [ENTER] para continuar

```

Opción 4: Mostrar un rango de datos del archivo

Como su nombre lo indica se mostrará en esta parte un grupo de registros.

```
void mostrarRangoDeRegistros(){
    ifstream archBin("Personal.bin",ios::in|ios::binary);
    if(not archBin.is_open()){
        cout<<"ERROR: No se pudo abrir el archivo Personal.bin" <<endl;
        exit(1);
    }
    struct Persona persona;;
    int tamReg, tamArch, numReg, inicio, fin;
    char s, rango[20];
    tamReg = sizeof(struct Persona);
    datosDelArchivo(archBin,tamReg, tamArch, numReg);
    cout<<endl<<endl;
    cout<<" Segun el siguiente formato:"<<endl;
    cout<<"     Para los primeros N registros: -n"<<endl;
    cout<<"     Para los ultimos N registros : n-"<<endl;
    cout<<"     Para un rango:                      m-n"<<endl;
    cout<<"     Ingrese el rango: ";
    cin.getline(rango,20);
    inicio = 0;
    fin = numReg;
    separarRango(rango, inicio, fin);
    if (inicio <=fin and inicio>=0 and inicio<=numReg and
        fin>=0 and fin<=numReg){
        archBin.seekg(inicio*tamReg,ios::beg);
        cout<<endl;
        imprimeLinea('=',70);
        cout<<right<<setw(4)<<"No."<<setw(6)<<"DNI"
            <<setw(12)<<"NOMBRE"<<setw(43)<<"SUELDO"<<endl;
        imprimeLinea('=',70);
        while (inicio < fin) {
            archBin.read(reinterpret_cast<char*>(&persona),
                         tamReg);
            if (persona.activo)
                cout<<right<<setw(3)<<inicio+1<<" "
                    <<left<<setw(10)<<persona.dni
                    <<setw(40)<<persona.nombre
                    <<right<<setw(10)<<persona.sueldo<<endl;
            inicio++;
        }
    }
    else {
        imprimeLinea('=',70);
        cout<<endl<<" ERROR: NO INGRESO UN RANGO VALIDO" <<endl<<endl;
    }
    imprimeLinea('=',70);
    cout<<endl<<endl<<setw(40)<<"Presione [ENTER] para continuar";
    cin.get();
}
```

```

void separarRango(char *rango, int &inicio, int &fin){
    int i=0;
    while(rango[i]){
        if(rango[i]=='-') break;
        i++;
    }
    if(i != strlen(rango)-1){
        fin = atoi(&rango[i+1]);
    }
    if(i!=0){
        rango[i] = 0; //Se coloca el cero para poder sacar el rango de inicio
        inicio = atoi(rango)-1;
    }
}

```

A continuación, veremos las tres formas en que se puede solicitar este reporte:
Los primeros n registros:

No.	DNI	NOMBRE	SUELDO
1)	97514234	RAFFO ARIAS SURAMI ANNIE	2590.42
2)	15523742	DIAZ CORREA IRMA VIRGINIA	347.22
3)	90001464	VALVERDE FLORES IRMA YDALI	1097.12
4)	70022841	CHUMPIAZ IBEROS RONAL	1023.32
5)	27366789	CHUNG LEE ALVARO	2825.37
6)	40590370	PEREZ FARFAN RICHARD SANDRO	6866.48
7)	26358230	CABRERA LUNA SHIRLEY ISABEL	782.57

Desde el registro n hasta el registro m:

No.	DNI	NOMBRE	SUELDO
5)	27366789	CHUNG LEE ALVARO	2825.37
6)	40590370	PEREZ FARFAN RICHARD SANDRO	6866.48
7)	26358230	CABRERA LUNA SHIRLEY ISABEL	782.57
8)	44660461	PINTO CALIXTO SHIRLEY	3506.8
9)	84018683	ROMERO PALOMARES GLORIA TATIANA	4328.31
10)	88641921	SANTISTEBAN MEZA ROSA	8468.73
11)	22794073	MENDOZA EGUSQUIZA ROSARIO	1846.57
12)	85428483	SALAZAR CORRALES LORENA IRMA	2695.6
13)	37232995	REYES TANG EDWARD	2476.16
14)	48179147	CERNA PAIRAZAMAN FIORELLA MARGOT	5301.58
15)	94424972	FORZANI CARO ALONSO	3862.59

Los últimos n registros:



Opción 5: Agregar registros al final del archivo

En esta operación vamos a agregar registros al archivo, el proceso en este caso colocará los nuevos registros al final del archivo, por lo que luego tendrá que ordenarlo si es que desea mantener el archivo organizado.

```
void agregarAlFinalDelArchivo(){
    fstream archBin("Personal.bin",
                    ios::in|ios::out|ios::binary);
    //Vamos a actualizarlo
    if(not archBin.is_open()){
        cout<<"ERROR: No se pudo abrir el archivo Personal.bin" <<endl;
        exit(1);
    }
    struct Persona persona;;
    int tamReg, tamArch, numReg, n=0;
    tamReg = sizeof(struct Persona);
    archBin.seekg(0, ios::end); // Nos movemos al final del archivo
    imprimeLinea('=',70);
    cout<<endl<<" INGRESE LOS NUEVOS REGISTROS:"<<endl<<endl;
    while(true){
        cout<<" DNI (0 para terminar): ";
        cin>>persona.dni;
        while(cin.get()!='\n');
        if (persona.dni == 0)break;
        cout<<" Nombre : ";
        cin.getline(persona.nombre, 60);
        cout<<" Sueldo : ";
        cin>>persona.sueldo;
        persona.activo = true;
        archBin.write(reinterpret_cast<const char*>(&persona),
                      tamReg); // Escribimos al final del archivo
        n++;
    }
    if (n > 0){
        if (n == 1) cout<<endl<<" SE AGREGO UN REGISTRO"<<endl;
        else cout<<endl<<" SE AGREGARON "<<n<<" REGISTROS"<<endl;
        cout<<endl<<endl;
    }
}
```

```

        cout<<setw(40)<<"***** EL ARCHIVO PUDO QUEDAR DESORDENADO *****"<<endl;
    }
    imprimeLinea('=', 70);
    cout<<endl<<endl;
    cout<<setw(40)<<"Presione [ENTER] para continuar";
    cin.get();
}

```

El resultado es el siguiente:

```

mantenimientodearchivoscpp
=====
ELIJA UNA OPCION [0-11]: 5
=====
INGRESE LOS NUEVOS REGISTROS:
DNI (0 para terminar): 77889955
Nombre : RONCAL NEYRA ANA CECILIA
Sueldo : 3987.45
DNI (0 para terminar): 12213443
Nombre : BARZOLA QUISPE NAOMI ALEXANDRA
Sueldo : 1500.50
DNI (0 para terminar): 99009923
Nombre : ZURITA YANES PAULA VALENTINA
Sueldo : 2500.00
DNI (0 para terminar): 0

SE AGREGARON 3 REGISTROS

**** EL ARCHIVO PUDO QUEDAR DESORDENADO *****
=====

Presione [ENTER] para continuar

```



```

mantenimientodearchivoscpp
=====
ELIJA UNA OPCION [0-11]: 2
=====
No. DNI      NOMBRE          SUELDO
=====
1) 97514234 RAFFO ARIAS SURAMI ANNIE      2590.42
2) 15523742 DIAZ CORREA IRMA VIRGINIA     347.22
3) 90001464 VALVERDE FLORES IRMA YDALI   1097.12
4) 70022841 CHUMPITAZ IBEROS RONAL       1023.32
5) 27366789 CHUNG LEE ALVARO             2825.37
6) 40598370 PEREZ FARFAN RICHARD SANDRO  6866.48
7) 26358230 CABRERA LUNA SHIRLEY ISABEL  782.57
8) 44660461 PINTO CALIXTO SHIRLEY         3506.80
9) 84018683 ROMERO PALOMARES GLORIA TATIANA 4328.31
10) 88641921 SANTISTEBAN MEZA ROSA        8468.73
11) 22794073 MENDOZA EGUSQUIZA ROSARIO    1846.57
12) 85428483 SALAZAR CORRALES LORENA IRMA  2695.60
13) 37232995 REYES TANG EDWARD           2476.16
14) 48179147 CERNA PAIRAZAMAN FIORELLA MARGOT 5301.58
15) 94424972 FORZANI CARO ALONSO          3862.59
16) 70719467 SHIRAKAWA MIRANDA MADELEINE    8609.19
17) 16565048 ARROYO GORDILLO MARIA HELI    3831.02
18) 22766834 HERNANDEZ BARRIOS EMILIO     5086.21
19) 82000163 VEGA GARCIA JANET REBECA      2432.14
20) 87895755 SHIRAKAWA ORTEGA ANGEL        3789.89
21) 45968606 WONG MARTEL MADELEINE        6146.95
22) 57823550 DIAZ ANTEZANO MAGALI SILVANA  5389.56
23) 53219729 MEJIA DIAZ BLANCA MILAGROS    4291.45
24) 74294791 RAFFO CHUNG ADRIA GLORIA     7685.69
25) 92136229 CASTRO SUAREZ ROLANDO       2240.96
26) 77889955 RONCAL NEYRA ANA CECILIA      3987.45
27) 12213443 BARZOLA QUISPE NAOMI ALEXANDRA 1500.50
28) 99009923 ZURITA YANES PAULA VALENTINA  2500.00

Presione [ENTER] para continuar

```

Si se desea que el archivo se ordene, se deberá elegir la opción adecuada para clasificarlo nuevamente y luego volverlo a mostrar.

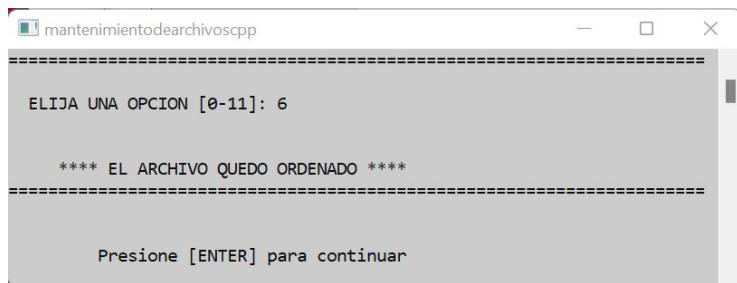
Opción 6: Ordenar el archivo por el nombre del empleado

Esta opción va a ordenar todos los registros del archivo, la operación alterará el orden en que se encuentran los registros del archivo, por lo que la operación será irreversible.

Debe observar los límites que usa el algoritmo en sus iteraciones, también ver que no se requiere una variable auxiliar para intercambiar los datos como se hace con arreglos, tampoco una función de intercambio.

```
void ordenarArchivo(){
    fstream archBin("Personal.bin",
                    ios::in|ios::out|ios::binary);
    if(not archBin.is_open()){
        cout<<"ERROR: No se pudo abrir el archivo Personal.bin" <<endl;
        exit(1);
    }
    struct Persona personaI, personaK;
    int tamReg, tamArch, numReg;
    tamReg = sizeof(struct Persona);
    datosDelArchivo(archBin,tamReg, tamArch, numReg);
    for(int i=0; i< numReg-1; i++)
        for(int k= i+1; k < numReg; k++) {
            archBin.seekg(i*tamReg,ios::beg);
            archBin.read(reinterpret_cast<char*>(&personaI),
                          tamReg);
            archBin.seekg(k*tamReg,ios::beg);
            archBin.read(reinterpret_cast<char*>(&personaK),
                          tamReg);
            if(strcmp(personaI.nombre,personaK.nombre)>0) {
                archBin.seekg(i*tamReg,ios::beg);
                archBin.write(reinterpret_cast
                                <const char*>(&personaK),
                                tamReg);
                archBin.seekg(k*tamReg,ios::beg);
                archBin.write(reinterpret_cast
                                <const char*>(&personaI),
                                tamReg);
                archBin.flush();
            }
        }
    cout<<endl<<endl;
    cout<<setw(40)<<"***** EL ARCHIVO QUEDO ORDENADO *****"<<endl;
    imprimeLinea('=',70);
    cout<<endl<<endl;
    cout<<setw(40)<<"Presione [ENTER] para continuar";
    cin.get();
}
```

Luego de ejecutar esta opción se verá por pantalla:



Luego se puede pedir al programa que muestre el contenido del archivo y esto se obtendrá:

No.	DNI	NOMBRE	SUELDO
1)	16565048	ARROYO GORDILLO MARIA HELI	3831.02
2)	12213443	BARZOLA QUISPE NAOMI ALEXANDRA	1500.50
3)	26358230	CABRERA LUNA SHIRLEY ISABEL	782.57
4)	92136229	CASTRO SUAREZ ROLANDO	2240.96
5)	48179147	CERNA PAIRAZAMAN FIORELLA MARGOT	5301.58
6)	70022841	CHUMPIATA IBEROS RONAL	1023.32
7)	27366789	CHUNG LEE ALVARO	2825.37
8)	57823558	DIAZ ANTEZANO MAGALI SILVANA	5389.56
9)	15523742	DIAZ CORREA IRMA VIRGINIA	347.22
10)	94424972	FORZANZ CARO ALONSO	3862.59
11)	22766834	HERNANDEZ BARRIOS EMILIO	5086.21
12)	53219729	MEJIA DIAZ BLANCA MILAGROS	4291.45
13)	22794073	MENDOZA EGUSQUIZA ROSARIO	1846.57
14)	40598370	PEREZ FARFAN RICHARD SANDRO	6866.48
15)	44668461	PINTO CALIXTO SHIRLEY	3506.80
16)	97514234	RAFFO ARIAS SURAMI ANNIE	2590.42
17)	74294791	RAFFO CHUNG ADRIA GLORIA	7685.69
18)	37232995	REYES TANG EDWARD	2476.16
19)	84018683	ROMERO PALOMARES GLORIA TATIANA	4328.31
20)	77889955	RONCAL NEYRA ANA CECILIA	3987.45
21)	85428483	SALAZAR CORRALES LORENA IRMA	2695.60
22)	88641921	SANTISTEBAN MEZA ROSA	8468.73
23)	70719467	SHIRAKAWA MIRANDA MADELEINE	8609.19
24)	87895755	SHIRAKAWA ORTEGA ANGEL	3789.89
25)	90001464	VALVERDE FLORES IRMA YDALI	1097.12
26)	82000163	VEGA GARCIA JANET REBECA	2432.14
27)	45968606	WONG MARTEL MADELEINE	6146.95
28)	99009923	ZURITA YANES PAULA VALENTINA	2500.00

Opción 7: Insertar registros de manera ordenada

En este caso agregaremos nuevos registros al archivo, pero el proceso desplazará los registros existentes e insertando el nuevo registro en la posición adecuada de modo que se mantenga siempre el archivo ordenado.

```
void insertarOrdenado(){
    fstream archBin("Personal.bin",
                    ios::in|ios::out|ios::binary);
    if(not archBin.is_open()){
        cout<<"ERROR: No se pudo abrir el archivo Personal.bin" <<endl;
        exit(1);
    }
    struct Persona persona, per;
    int tamReg, tamArch, numReg, n=0, numIns=0;

    tamReg = sizeof(struct Persona);
    datosDelArchivo(archBin,tamReg, tamArch, numReg);
    imprimeLinea('=',70);
    cout<<endl<<" INGRESE LOS NUEVOS REGISTROS:"<<endl<<endl;
```

```

while(true) {
    cout<<" DNI (0 para terminar): ";
    cin>>persona.dni;
    while(cin.get()!='\n');
    if (persona.dni == 0)break;
    cout<<" Nombre : ";
    cin.getline(persona.nombre, 60);
    cout<<" Sueldo : ";
    cin>>persona.sueldo;
    persona.activo = true;
    // Desplazamos los registros para ubicar el nuevo
    n = numReg-1;
    while(n>=0){
        archBin.seekg(n*tamReg,ios::beg);
        archBin.read(reinterpret_cast<char*>(&per),
                     tamReg);
        if (strcmp(per.nombre, persona.nombre)>0){
            // Avanzamos un registro
            archBin.seekg((n+1)*tamReg,ios::beg);
            archBin.write(reinterpret_cast
                           <const char*>(&per),tamReg);
            archBin.flush();
            n--;
        }
        else break;
    }
    n++; //Colocamos en posición el indicador del archivo
    numReg++;
    archBin.seekg(n*tamReg,ios::beg);
    archBin.write(reinterpret_cast<const char*>(&persona),
                  tamReg);
    archBin.flush();
    numIns++;
}
if (numIns > 0){
    if (numIns == 1) cout<<endl<<" SE AGREGO UN REGISTRO" <<endl;
    else cout<<endl<<" SE AGREGARON "<<numIns<<" REGISTROS"
              <<endl <<endl;
    cout<<endl<<endl<<setw(40)
                  <<"***** EL ARCHIVO SE MANTIENE ORDENADO *****" <<endl;
}
imprimeLinea('=',70);
cout<<endl<<endl;
cout<<setw(40)<<"Presione [ENTER] para continuar"<<endl;
cin.get();
}

```

El resultado es el siguiente:

```

ELIJA UNA OPCION [0-11]: 7
=====
INGRESE LOS NUEVOS REGISTROS:

DNI (0 para terminar): 57553055
Nombre : RAMIREZ CARRILLO WALTER SAUL
Sueldo : 3001.03
DNI (0 para terminar): 77755511
Nombre : FARIAS VILCA NAOMI VALENTINA
Sueldo : 1509.99
DNI (0 para terminar): 0

SE AGREGARON 2 REGISTROS

**** EL ARCHIVO SE MANTIENE ORDENADO ****
=====

Presione [ENTER] para continuar

```

```

ELIJA UNA OPCION [0-11]: 2
=====
No. DNI NOMBRE SUELDO
=====
1) 16565040 ARROYO GORDILLO MARIA HELI 3831.02
2) 12213443 BARZOLA QUISPE NAOMI ALEXANDRA 1500.50
3) 26358230 CABRERA LUNA SHIRLEY ISABEL 782.57
4) 92136229 CASTRO SUAREZ ROLANDO 2240.96
5) 48179147 CERNA PAIRAZAMAN FIORELLA MARGOT 5301.58
6) 70022841 CHUMPITAZ IBEROS RONAL 1023.32
7) 27366789 CHUNG LEE ALVARO 2825.37
8) 57823550 DIAZ ANTEZANO MAGALI SILVANA 5389.56
9) 15523742 DIAZ CORREA IRMA VIRGINIA 347.22
10) 77755511 FARIAS VILCA NAOMI VALENTINA 1509.99
11) 94424972 FORZANI CARO ALONSO 3862.59
12) 22766834 HERNANDEZ BARRIOS EMILIO 5086.21
13) 53219729 MEJIA DIAZ BLANCA MILAGROS 4291.45
14) 22794073 MENDOZA EGUSQUIZA ROSARIO 1846.57
15) 40598370 PEREZ FARFAN RICHARD SANDRO 6866.48
16) 44660461 PINTO CALIXTO SHIRLEY 3506.80
17) 97514234 RAFFO ARIAS SURAMI ANNIE 2590.42
18) 74294791 RAFFO CHUNG ADRIA GLORIA 7685.69
19) 57553055 RAMIREZ CARRILLO WALTER SAUL 3001.03
20) 37232995 REYES TANG EDWARD 2476.16
21) 84018683 ROMERO PALOMARES GLORIA TATIANA 4328.31
22) 77889955 RONCAL NEYRA ANA CECILIA 3987.45
23) 85428483 SALAZAR CORRALES LORENA IRMA 2695.60
24) 88641921 SANTISTEBAN MEZA ROSA 8468.73
25) 70719467 SHIRAKAWA MIRANDA MADELEINE 8609.19
26) 87895755 SHIRAKAWA ORTEGA ANGEL 3789.89
27) 90001464 VALVERDE FLORES IRMA YDALI 1097.12
28) 82000163 VEGA GARCIA JANET REBECA 2432.14
29) 45968696 WONG MARTEL MADELEINE 6146.95
30) 99009923 ZURITA YANES PAULA VALENTINA 2500.00

Presione [ENTER] para continuar

```

Opción 8: Modificar el contenido de un registro

Lo que hará el programa aquí será permitir corregir o modificar los registros del archivo, se podrá cambiar cualquiera de los campos de un registro.

```

void modificaRegistro(){
    fstream archBin("Personal.bin",
                    ios::in|ios::out|ios::binary);
    if(not archBin.is_open()){
        cout<<"ERROR: No se pudo abrir el archivo Personal.bin" << endl;
        exit(1);
    }
    struct Persona persona, per;
    int tamReg, tamArch, numReg, n;
    char dato[60], s;
    tamReg = sizeof(struct Persona);

```

```

datosDelArchivo(archBin,tamReg,tamArch,numReg);

while(true) {
    cout<<"Ingrese el numero de registro que quiere modificar (entre 1 y "
        <<numReg<<"): ";
    cin>>n;
    if (n >= 1 and n <= numReg) {
        archBin.seekg((n-1)*tamReg, ios::beg);
        archBin.read(reinterpret_cast<char*>(&persona),
                     tamReg);
        cout<<endl;
        imprimeLinea('=',70);
        cout<<left<<setw(7)<<"No."<<setw(11)<<"DNI"
            <<setw(43)<<"NOMBRE"<<"SUELDO"<<endl;
        imprimeLinea('=',70);
        cout<<right<<setw(3)<<n<<" "
            <<setw(10)<<persona.dni<<"    "
            <<left<<setw(40)<<persona.nombre
            <<right<<setw(10)<<persona.sueldo<<endl;
        if (persona.activo == 0)
            cout<<endl<<right<<setw(40)
                <<"***** REGISTRO DADO DE BAJA *****"<<endl;
        imprimeLinea('=',70);
        cout<<endl<<right<<setw(40)
            <<"***** SE MODIFICARA EL REGISTRO MOSTRADO *****"<<endl;
        cout<<"  Se le solicitara ingresar el valor de cada "
            <<"uno de los campos"<<endl;
        cout<<"  Si no quiere modificarlo solo debe presionar ENTER"
            <<endl;
        while(cin.get() != '\n');
        cout<<endl<<left<<setw(25)<<"  DNI (0 para terminar):";
        cin.getline(dato,60);
        if (strlen(dato) != 0) persona.dni = atoi(dato);
        cout<<left<<setw(25)<<"  Nombre:";
        cin.getline(dato,60);
        if (strlen(dato) != 0)
            strcpy(persona.nombre,dato);
        cout<<left<<setw(25)<<"  Sueldo:";
        cin.getline(dato,60);
        if (strlen(dato) != 0)
            persona.sueldo = atof(dato);
        imprimeLinea('=',70);
        archBin.seekg((n-1)*tamReg, ios::beg);
        archBin.write(reinterpret_cast<const char*>
                      (&persona),tamReg);
        archBin.flush();
        cout<<endl<<right<<setw(50)
            <<"***** SE MODIFICO EL REGISTRO *****";
        cout<<endl<<"Si desea modificar otro registro ingrese S, "
            <<"de lo contrario ENTER:";
        s = cin.get();
}

```

```

        s = toupper(s);
        if (s != 'S')break;
    }
}

imprimeLinea('=', 70);
cout<<endl<<endl<<right<<setw(40)<<"Presione [ENTER] para continuar";
cin.get();
}

```

En la ejecución se apreciará lo siguiente:

```

ELIJA UNA OPCION [0-11]: 8
Ingrese el numero de registro que quiere modificar (entre 1 y 30): 12
=====
No. DNI      NOMBRE          SUELDO
=====
12) 22766834 HERNANDEZ BARRIOS EMILIO   5086.21
=====

**** SE MODIFICARA EL REGISTRO MOSTRADO ****
Se le solicitara ingresar el valor de cada uno de los campos
Si no quiere modificarlo solo debe presionar ENTER

DNI (0 para terminar):
Nombre:           HERNANDEZ BARRIOS EMILIA
Sueldo:
=====

**** SE MODIFICO EL REGISTRO ****
Si desea modificar otro registro ingrese S, de lo contrario ENTER:S
Ingrese el numero de registro que quiere modificar (entre 1 y 30): 20
=====
No. DNI      NOMBRE          SUELDO
=====
20) 37232995 REYES TANG EDWARD       2476.16
=====

**** SE MODIFICARA EL REGISTRO MOSTRADO ****
Se le solicitara ingresar el valor de cada uno de los campos
Si no quiere modificarlo solo debe presionar ENTER

DNI (0 para terminar): 73232995
Nombre:
Sueldo:           3476.16
=====

**** SE MODIFICO EL REGISTRO ****
Si desea modificar otro registro ingrese S, de lo contrario ENTER:
=====

Presione [ENTER] para continuar

```

```

ELIJA UNA OPCION [0-11]: 2
=====
No. DNI      NOMBRE          SUELDO
=====
1) 16565040 ARROYO GORDILLO MARIA HELI   3831.02
2) 12213443 BARZOLA QUISPE NAOMI ALEXANDRA 1500.50
3) 26358230 CABRERA LUNA SHIRLEY ISABEL   782.57
4) 92136229 CASTRO SUAREZ ROLANDO       2240.96
5) 48179147 CERNA PAIRAZAMAN FIORELLA MARGOT 5301.58
6) 70022841 CHUMPITAZ IBEROS RONAL       1023.32
7) 27366789 CHUNG LEE ALVARO             2825.37
8) 57823550 DIAZ ANTEZANO MAGALI SILVANA  5389.56
9) 15523742 DIAZ CORREA IRMA VIRGINIA     347.22
10) 77755511 FARTAS VILCA NAOMI VALENTINA 1509.99
11) 94424972 FORZANI CARO ALONSO         3862.59
12) 22766834 HERNANDEZ BARRIOS EMILIA     5086.21
13) 53219729 MEJIA DIAZ BLANCA MILAGROS  4291.45
14) 22794073 MENDOZA EGUSQUIZA ROSARIO   1846.57
15) 405980370 PEREZ FARFAN RICHARD SANDRO 6866.48
16) 44660461 PINTO CALIXTO SHIRLEY        3506.80
17) 97514234 RAFFO ARIAS SURAMI ANNIE    2590.42
18) 74294791 RAFFO CHUNG ADRIA GLORIA    7685.69
19) 57553055 RAMIREZ CARRILLO WALTER SAUL 3001.03
20) 73232995 REYES TANG EDWARD          3476.12
21) 84918683 ROMERO PALOMARES GLORIA TATIANA 4328.31
22) 77889955 RONCAL NEYRA ANA CECILIA     3987.45
23) 85428483 SALAZAR CORRALES LORENA IRMA  2695.60
24) 88641921 SANTISTEBAN MEZA ROSA        8468.73
25) 70719467 SHIRAKAWA MIRANDA MADELEINE  8609.19
26) 87895755 SHIRAKAWA ORTEGA ANGEL       3789.89
27) 90001464 VALVERDE FLORES IRMA YDALI   1097.12
28) 82000163 VEGA GARCIA JANET REBECA     2432.14
29) 45968606 WONG MARTEL MADELEINE       6146.95
30) 99009923 ZURITA YANES PAULA VALENTINA 2500.00
=====

Presione [ENTER] para continuar

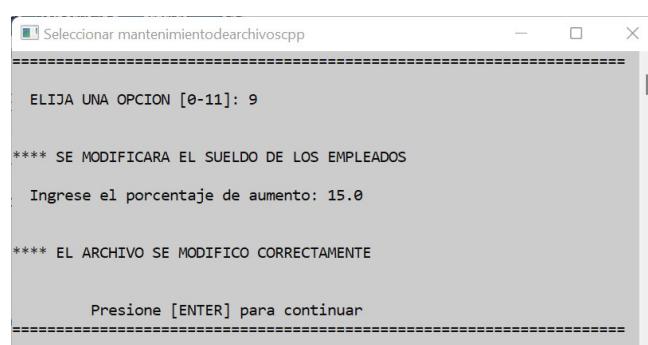
```

Opción 9: Modificar el contenido de varios registros

Aquí lo que se pretende es modificar uno de los campos del archivo, en este caso se decidió modificar el campo correspondiente al sueldo de los empleados, el usuario ingresará el porcentaje de aumento, y éste se les aplicará a todos los sueldos.

```
void modificaRegistros(){
    fstream archBin("Personal.bin",
                    ios::in|ios::out|ios::binary);
    if(not archBin.is_open()){
        cout<<"ERROR: No se pudo abrir el archivo Personal.bin"\;
        exit(1);
    }
    struct Persona persona, per;
    int tamReg, tamArch, numReg, n;
    double porcent;
    tamReg = sizeof(struct Persona);
    datosDelArchivo(archBin,tamReg, tamArch, numReg);
    cout<<endl<<endl<<right
        <<setw(40)<<"***** SE MODIFICARA EL SUELDO DE LOS EMPLEADOS"\;
    cout<<endl<<" Ingrese el porcentaje de aumento: ";
    cin>>porcent;
    porcent = 1 + porcent/100;
    for (int i=0; i< numReg; i++) {
        archBin.seekg(i*tamReg,ios::beg);
        archBin.read(reinterpret_cast<char*>(&persona),
                     tamReg);
        persona.sueldo *= porcent;
        // se regresa una posición para poder registrar la modificación
        archBin.seekg(i*tamReg,ios::beg);
        archBin.write(reinterpret_cast<const char*>(&persona),
                      tamReg);
        archBin.flush();
    }
    cout<<endl<<endl<<right
        <<setw(40)<<"***** EL ARCHIVO SE MODIFICO CORRECTAMENTE"\;
    cout<<endl<<endl<<right
        <<setw(40)<<"Presione [ENTER] para continuar";
    cin.get();
    cout<<endl;
}
```

En la ejecución se apreciará lo siguiente:



ELIJA UNA OPCION [0-11]: 2

No.	DNI	NOMBRE	SUELDO
1)	16565040	ARROYO GORDILLO MARIA HELI	4405.67
2)	12213443	BARZOLA QUISPE NAOMI ALEXANDRA	1725.57
3)	26358230	CABRERA LUNA SHIRLEY ISABEL	899.96
4)	92136229	CASTRO SUAREZ ROLANDO	2577.10
5)	48179147	CERNA PAIRAZAMAN FIORELLA MARGOT	6096.82
6)	70022841	CHUMPITAZ IBEROS RONAL	1176.82
7)	27366789	CHUNG LEE ALVARO	3249.18
8)	57823558	DIAZ ANTEZANO MAGALI SILVANA	6197.99
9)	15523742	DIAZ CORREA IRMA VIRGINIA	399.30
10)	77755511	FARIAS VILCA NAOMI VALENTINA	1736.49
11)	94424972	FORZANI CARO ALONSO	4441.98
12)	22766834	HERNANDEZ BARRIOS EMILIA	5849.14
13)	53219729	MEJIA DIAZ BLANCA MILAGROS	4935.17
14)	22794073	MENDOZA EGUSQUIZA ROSARIO	2123.56
15)	40590370	PEREZ FARFAN RICHARD SANDRO	7896.45
16)	44660461	PINTO CALIXTO SHIRLEY	4032.82
17)	97514234	RAFFO ARIAS SURAMI ANNIE	2978.98
18)	74294791	RAFFO CHUNG ADRIA GLORIA	8838.54
19)	57553055	RAMIREZ CARRILLO WALTER SAUL	3451.18
20)	73232995	REYES TANG EDWARD	3997.58
21)	84018683	ROMERO PALOMARES GLORIA TATIANA	4977.56
22)	77889955	RONCAL NEYRA ANA CECILIA	4585.57
23)	85428483	SALAZAR CORRALES LORENA IRMA	3099.94
24)	88641921	SANTISTEBAN MEZA ROSA	9739.94
25)	70719467	SHIRAKAWA MIRANDA MADELEINE	9900.57
26)	87895755	SHIRAKAWA ORTEGA ANGEL	4358.37
27)	90001464	VALVERDE FLORES IRMA YDALI	1261.69
28)	82000163	VEGA GARCIA JANET REBECA	2796.96
29)	45968696	WONG MARTEL MADELEINE	7068.99
30)	99009923	ZURITA YANES PAULA VALENTINA	2875.00

Presione [ENTER] para continuar

Opción 10: Borrar de manera "lógica" registros del archivo

Como se dijo anteriormente la idea del borrado "lógico" es que el registro no se destruya, lo que se busca de esta forma de borrado es que el registro simplemente desaparezca de los listados que realice el programa y que no se apliquen los cambios que se hagan a los demás registros. Para el usuario este registro ya no existe, sin embargo, permanece en el archivo. La finalidad de hacer esto es que se pueda recuperar en otro momento esta información por cualquier razón.

En la aplicación que hemos realizado se resaltará el hecho que en el archivo haya registros borrados, como se verá en la ejecución de estos módulos. En aplicaciones reales estos registros se toman como si no existieran por lo que se procura que pase desapercibido completamente para el usuario.

```
void eliminacionLogicaDeRegistros() {
    fstream archBin("Personal.bin",
                    ios::in|ios::out|ios::binary);
    if(not archBin.is_open()){
        cout<<"ERROR: No se pudo abrir el archivo Personal.bin"<<endl;
        exit(1);
    }
    struct Persona persona, per;
    int tamReg, tamArch, numReg, n;
    char s;
    tamReg = sizeof(struct Persona);
    datosDelArchivo(archBin,tamReg, tamArch, numReg);
```

```

while(1) {
    cout<<" Ingrese el numero de registro que quiere eliminar (entre 1 y"
        <<setw(4)<<numReg<<") : ";
    cin>>n;
    while(cin.get() != '\n');
    if (n >= 1 and n <= numReg) {
        archBin.seekg((n-1)*tamReg, ios::beg);
        archBin.read(reinterpret_cast<char*>(&persona),
                     tamReg);
        cout<<endl;
        imprimeLinea('=', 70);
        cout<<endl<<right<<"No."<<setw(6)<<"DNI"
            <<setw(14)<<"NOMBRE"
            <<setw(40)<<"SUELDO"<<endl;
        imprimeLinea('=', 70);
        cout<<right<<setw(3)<<n<<") "
            <<setw(10)<<persona.dni<<"      "
            <<left<<setw(40)<<persona.nombre
            <<setw(10)<<persona.sueldo<<endl;
        if (not persona.activo)
            cout<<endl<<right<<setw(40)
                <<"***** REGISTRO DADO DE BAJA *****"<<endl;
        imprimeLinea('=', 70);
        cout<<endl<<" **** SE ELIMINARA EL REGISTRO MOSTRADO ****"<<endl;
        cout<<endl<<" Desea continuar?, ingrese S/N: ";
        s = cin.get();
        s = toupper(s);
        while(cin.get() != '\n');
        if (s=='S'){
            persona.activo = false;
            archBin.seekg((n-1)*tamReg,ios::beg);
            archBin.write(reinterpret_cast<const char*>
                            (&persona),tamReg);
            archBin.flush();
            cout<<endl<<right<<setw(40)
                <<"***** SE ELIMINO EL REGISTRO *****"<<endl;
        }
        else cout<<endl<<right<<setw(40)
            <<"***** SE CANCELO LA OPERACION *****"<<endl;
        cout<<endl<<right<<" Si desea eliminar otro registro ingrese S,"
            <<" de lo contrario ENTER:";
        s = cin.get();
        s = toupper(s);
        if(s != 'S') break;
    }
    imprimeLinea('=', 70);
    cout<<endl<<endl<<right
        <<setw(40)<<"Presione [ENTER] para continuar";
    cin.get();
}

```

La ejecución será la siguiente:

```
ELIJA UNA OPCION [0-11]: 10
Ingrese el numero de registro que quiere eliminar (entre 1 y 30): 15
=====
No. DNI NOMBRE SUELDO
=====
15) 40590370 PEREZ FARFAN RICHARD SANDRO 7896.45
=====

**** SE ELIMINARA EL REGISTRO MOSTRADO ****

Desea continuar?, ingrese S/N: S

**** SE ELIMINO EL REGISTRO ****

Si desea eliminar otro registro ingrese S, de lo contrario ENTER:S
Ingrese el numero de registro que quiere eliminar (entre 1 y 30): 25
=====
No. DNI NOMBRE SUELDO
=====
25) 70719467 SHIRAKAWA MIRANDA MADELEINE 9900.57
=====

**** SE ELIMINARA EL REGISTRO MOSTRADO ****

Desea continuar?, ingrese S/N: S

**** SE ELIMINO EL REGISTRO ****

Si desea eliminar otro registro ingrese S, de lo contrario ENTER:
=====

Presione [ENTER] para continuar
```

Si se desea ver el registro, este programa lo mostrará de la siguiente manera:

```
ELIJA UNA OPCION [0-11]: 3
Ingrese el numero de registro que quiere ver (entre 1 y 30): 15
=====
No. DNI NOMBRE SUELDO
=====
15) 40590370 PEREZ FARFAN RICHARD SANDRO 7896.45
=====

**** REGISTRO DADO DE BAJA ****

Si desea ver otro registro ingrese S, de lo contrario ENTER:
```

El listado completo se mostrará, en este programa, se verá así:

```
ELIJA UNA OPCION [0-11]: 2
=====
No. DNI NOMBRE SUELDO
=====
1) 16565040 ARROYO GORDILLO MARIA HELI 4405.67
2) 12213443 BARZOLA QUISPE NAOMI ALEXANDRA 1725.57
3) 26358230 CABRERA LUNA SHIRLEY ISABEL 899.96
4) 92136229 CASTRO SUAREZ ROLANDO 2577.10
5) 48179147 CERNA PAIRAZAMAN FIORELLA MARGOT 6096.82
6) 70022841 CHUMPITAZ IBEROS RONAL 1176.82
7) 27366789 CHUNG LEE ALVARO 3249.18
8) 57823550 DIAZ ANTEZANO MAGALI SILVANA 6197.99
9) 15523742 DIAZ CORREA IRMA VIRGINIA 399.30
10) 77755511 FARIAS VILCA NAOMI VALENTINA 1736.49
11) 94424972 FORZANI CARO ALONSO 4441.98
12) 22766834 HERNANDEZ BARRIOS EMILIA 5849.14
13) 53219729 MEJIA DIAZ BLANCA MILAGROS 4935.17
14) 22794973 MENDOZA EGUSQUIZA ROSARIO 2123.56
16) 44660461 PINTO CALIXTO SHIRLEY 4032.82
17) 97514234 RAFFO ARIAS SURAMI ANNIE 2978.98
18) 74294791 RAFFO CHUNG ADRIA GLORIA 8838.54
19) 57553055 RAMIREZ CARRILLO WALTER SAUL 3451.18
20) 73232995 REYES TANG EDWARD 3997.58
21) 84018683 ROMERO PALOMARES GLORIA TATIANA 4977.56
22) 77889955 RONCAL NEYRA ANA CECILIA 4585.57
23) 85428483 SALAZAR CORRALES LORENA IRMA 3099.94
24) 88641921 SANTISTEBAN MEZA ROSA 9739.04
26) 87895755 SHIRAKAWA ORTEGA ANGEL 4358.37
27) 90001464 VALVERDE FLORES IRMA YDALI 1261.69
28) 82000163 VEGA GARCIA JANET REBECA 2796.96
29) 45968606 WONG MARTEL MADELEINE 7068.99
30) 99009923 ZURITA YANES PAULA VALENTINA 2875.00
=====

Presione [ENTER] para continuar
```

Opción 11: Recuperar registros borrados de manera "lógica"

Aquí, la opción permitirá que registros que hayan sido borrados "lógicamente" puedan volver a mostrarse. Esta operación es sencilla ya que sólo se debe cambiar el valor lógico de la variable de control.

```
void recuperacionDeRegistros(){
    fstream archBin("Personal.bin",
                    ios::in|ios::out|ios::binary);
    if(not archBin.is_open()){
        cout<<"ERROR: No se pudo abrir el archivo Personal.bin"<<endl;
        exit(1);
    }
    struct Persona persona, per;
    int tamReg, tamArch, numReg, n;
    char s;
    tamReg = sizeof(struct Persona);
    datosDelArchivo(archBin,tamReg, tamArch, numReg);
    while(true){
        cout<<" Ingrese el numero de registro que quiere recuperar (entre 1 y "
            <<setw(4)<<n<<") : ";
        cin>>n;
        while(cin.get()!='\n');
        if (n >= 1 && n <= numReg) {
            archBin.seekg((n-1)*tamReg, ios::beg);
            archBin.read(reinterpret_cast<char*>(&persona),
                         tamReg);
            cout<<endl;
            imprimeLinea('=', 70);
            cout<<endl<<right<<"No. "<<setw(6)<<"DNI"
                <<setw(14)<<"NOMBRE"
                <<setw(40)<<"SUELDO"<<endl;
            imprimeLinea('=', 70);
            cout<<right<<setw(3)<<n<<") "
                <<setw(10)<<persona.dni<<"      "
                <<left<<setw(40)<<persona.nombre
                <<setw(10)<<persona.sueldo<<endl;
            if (not persona.activo){
                cout<<endl<<right<<setw(40)
                    <<"***** REGISTRO DADO DE BAJA *****"<<endl;
                imprimeLinea('=', 70);
                cout<<endl<<" **** SE RECUPERARA EL REGISTRO MOSTRADO ****"
                    <<endl;
                cout<<endl<<" Desea continuar?, ingrese S/N: ";
                s = cin.get();
                s = toupper(s);
                while(cin.get()!='\n');
                if (s=='S'){
                    persona.activo = true;
                    archBin.seekg((n-1)*tamReg,ios::beg);
                    archBin.write(reinterpret_cast
                        <const char*>(&persona),tamReg);
                }
            }
        }
    }
}
```

```

        archBin.flush();
        cout<<endl<<right<<setw(40)
            <<"**** SE RECUPERO EL REGISTRO ****"<<endl;
    }
    else cout<<endl<<right<<setw(40)
        <<"**** SE CANCELO LA OPERACION ****"<<endl;
}
else {
    cout<<endl<<" ****REGISTRO NO HA SIDO DADO DE BAJA,"
        <<" SE CANCELA LA OPERACION ****"<<endl;
}
cout<<endl<<" Si desea eliminar otro registro ingrese S,"
    <<" de lo contrario ENTER: ";
s = cin.get();
s = toupper(s);
if(s != 'S') break;
}
}

imprimeLinea('=',70);
cout<<endl<<endl<<right
    <<setw(40)<<"Presione [ENTER] para continuar";
cin.get();

}

```

La ejecución será la siguiente:

```

=====
ELIJA UNA OPCION [0-11]: 11
Ingrese el numero de registro que quiere recuperar (entre 1 y 30): 10
=====

No.   DNI      NOMBRE          SUELDO
=====
10)  7775511  FARIAS VILCA NAOMI VALENTINA  1736.49

****REGISTRO NO HA SIDO DADO DE BAJA, SE CANCELA LA OPERACION ****

Si desea eliminar otro registro ingrese S, de lo contrario ENTER: s
Ingrese el numero de registro que quiere recuperar (entre 1 y30 ): 15
=====

No.   DNI      NOMBRE          SUELDO
=====
15)  40590370  PEREZ FARFAN RICHARD SANDRO  7896.45

**** REGISTRO DADO DE BAJA ***

=====

**** SE RECUPERARA EL REGISTRO MOSTRADO ***

Desea continuar?, ingrese S/N: s

**** SE RECUPERO EL REGISTRO ***

Si desea eliminar otro registro ingrese S, de lo contrario ENTER:
=====

Presione [ENTER] para continuar

```

Luego, al listar los datos obtendremos:

No.	DNI	NOMBRE	SUELDO
1)	16565940	ARROYO GORDILLO MARIA HELI	4405.67
2)	12213443	BARZOLA QUISPE NAOMI ALEXANDRA	1725.57
3)	26358230	CABRERA LUNA SHIRLEY ISABEL	899.96
4)	92136229	CASTRO SUAREZ ROLANDO	2577.10
5)	48179147	CERNA PAIRAZAMAN FIORELLA MARGOT	6096.82
6)	70022841	CHUMPITAZ IBEROS RONAL	1176.82
7)	27366789	CHUNG LEE ALVARO	3249.18
8)	57823550	DIAZ ANTEZANO MAGALI SILVANA	6197.99
9)	15523742	DIAZ CORREA IRMA VIRGINIA	399.30
10)	77755511	FARIAS VILCA NAOMI VALENTINA	1736.49
11)	94424972	FORZANI CARO ALONSO	4441.98
12)	22766834	HERNANDEZ BARRIOS EMILIA	5849.14
13)	53219729	MEJIA DIAZ BLANCA MILAGROS	4935.17
14)	22794073	MENDOZA EGUSQUIZA ROSARIO	2123.56
15)	40590570	PEREZ FARFAN RICHARD SANDRO	7896.45
16)	44660461	PINTO CALIXTO SHIRLEY	4032.82
17)	97514234	RAFFO ARIAS SURAMI ANNIE	2978.98
18)	74294791	RAFFO CHUNG ADRIA GLORIA	8838.54
19)	57553055	RAMIREZ CARRILLO WALTER SAUL	3451.18
20)	73232995	REYES TANG EDWARD	3997.58
21)	84018683	ROMERO PALOMARES GLORIA TATIANA	4977.56
22)	77889955	RONCAL NEYRA ANA CECILIA	4585.57
23)	85428483	SALAZAR CORRALES LORENA IRMA	3099.94
24)	88641921	SANTISTEBAN MEZA ROSA	9739.04
26)	87895755	SHIRAKAWA ORTEGA ANGEL	4358.37
27)	90001464	VALVERDE FLORES IRMA YDALI	1261.69
28)	82000163	VEGA GARCIA JANET REBECA	2796.96
29)	45968606	WONG MARTEL MADELEINE	7068.99
30)	99009923	ZURITA YANES PAULA VALENTINA	2875.00

Presione [ENTER] para continuar

Finalmente, presentamos funciones auxiliares que se utilizan en el proyecto:

```

void datosDelArchivo(ifstream &archBin, int tamReg,
                      int &tamArch, int &numReg) {
    archBin.seekg(0, ios::end);
    tamArch = archBin.tellg();
    archBin.seekg(0, ios::beg);
    numReg = tamArch/tamReg;
}
void datosDelArchivo(fstream &archBin, int tamReg,
                      int &tamArch, int &numReg) {
    archBin.seekg(0, ios::end);
    tamArch = archBin.tellg();
    archBin.seekg(0, ios::beg);
    numReg = tamArch/tamReg;
}

void imprimeLinea(char car, int numCar) {
    for(int i=0; i<numCar; i++) cout.put(car);
    cout<<endl;
}

```