INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
ASIGNATURA:	PROGRAMACIÓN DE SISTEMAS				
TÍTULO DE LA PRÁCTICA:	Ficheros en C++				
NÚMERO DE PRÁCTICA:	06	**AÑO LECTIVO:**	2024-A	**NRO. SEMESTRE:**	V
FECHA DE PRESENTACIÓN	20-06-2024	**HORA DE PRESENTACIÓN**	20:00		
INTEGRANTE (s): ● Chino Pari Joel Antonio ● Cusilayme García José Luis ● Mamani Mamani Alexis Baltazar				**NOTA:**	
DOCENTE(s): ● ING. RONALD MANCINI TICONA					

SOLUCIÓN Y RESULTADOS

I. SOLUCIÓN DE EJERCICIOS/PROBLEMAS

EJERCICIO 02:

Dadas las siguientes instrucciones:

```
struct Tdato
{
int b;
char s[100];
};
int x, n, a[10]={1,2,3,4,5,6,7,8,9,0};
double f;
char nombre[]="Ejercicios ficheros binarios";
Tdato p;
ifstream f1;
ofstream f2;
f1.open("entrada.dat", ios::binary);
f2.open("salida.dat", ios::binary);
```

Escribe las instrucciones para realizar cada una de las siguientes operaciones:

a.Escribir el dato entero x en el fichero.
b.Escribir el dato double f en el fichero.
c.Escribir los 5 primeros elementos del array a en el fichero.
d.Escribir los 10 primeros caracteres de la cadena nombre en el fichero.
e.Escribir el dato de tipo Tdato p en el fichero.
f.Leer un dato entero del fichero y almacenarlo en la variable x.
g. Leer un dato double del fichero y almacenarlo en la variable f.
h.Leer 5 enteros y almacenarlos en el array a.
i.Leer 10 caracteres y almacenarlos en la cadena nombre.
j.Leer un dato de tipo Tdato y almacenarlo en la variable p.

```cpp
#include <iostream>
#include <fstream>
#include <cstdlib> // Para std::system
using namespace std;

struct Tdato {
    int b;
    char s[100];
};

int main() {
    int x = 0; // Inicializamos x
    double f = 0.0; // Inicialimos f
    int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0}; // Agregamos valores al arreglo
    char nombre[] = "Ejercicios ficheros binarios"; // Definimos una frase en un
 arreglo de caracteres
    Tdato p = {0, ""}; // Inicializamos la estructura (Tdato)

    ifstream f1;
    ofstream f2;

    // Imprimir el directorio de trabajo actual para depuración
    cout << "Directorio de trabajo actual: ";
    system("pwd");

    // Verificar si entrada.dat existe y sus permisos
    cout << "Verificando existencia y permisos del archivo entrada.dat" << endl;
    ifstream testFile("entrada.dat");
```

```
    if (!testFile) {
        cerr << "El fichero entrada.dat no existe o no se puede leer." << endl;
        return 1;
    } else {
        cout << "El fichero entrada.dat existe y es accesible." << endl;
    }
    testFile.close();

    // Intentar abrir los archivos
    cout << "Intentando abrir el archivo entrada.dat para lectura." << endl;
    f1.open("entrada.dat", ios::binary);
    if (!f1.is_open()) {
        cerr << "Error al abrir el fichero entrada.dat" << endl;
        return 1;
    } else {
        cout << "Archivo entrada.dat abierto exitosamente para lectura." << endl;
    }

    cout << "Intentando abrir el archivo salida.dat para escritura." << endl;
    f2.open("salida.dat", ios::binary);
    if (!f2.is_open()) {
        cerr << "Error al abrir el fichero salida.dat" << endl;
        return 1;
    } else {
        cout << "Archivo salida.dat abierto exitosamente para escritura." << endl;
    }

    // Escribimos datos en el fichero salida.dat
    f2.write(reinterpret_cast<const char*>(&x), sizeof(x));
    f2.write(reinterpret_cast<const char*>(&f), sizeof(f));
    f2.write(reinterpret_cast<const char*>(a), 5 * sizeof(a[0])); // Solo las
primeras 5 posiciones de a
    f2.write(nombre, 10); // Solo los primeros 10 caracteres de nombre
    f2.write(reinterpret_cast<const char*>(&p), sizeof(p));

    f2.close();
    cout << "Datos escritos en el archivo salida.dat." << endl;
```

```cpp
    // Leer datos del fichero entrada.dat
    f1.read(reinterpret_cast<char*>(&x), sizeof(x));
    if (f1.gcount() != sizeof(x)) cerr << "Error al leer x" << endl;

    f1.read(reinterpret_cast<char*>(&f), sizeof(f));
    if (f1.gcount() != sizeof(f)) cerr << "Error al leer f" << endl;

    f1.read(reinterpret_cast<char*>(a), 5 * sizeof(a[0]));
    if (f1.gcount() != 5 * sizeof(a[0])) cerr << "Error al leer a" << endl;

    f1.read(nombre, 10);
    if (f1.gcount() != 10) cerr << "Error al leer nombre" << endl;
    nombre[10] = '\0';  // Añadir el carácter nulo al final para terminar la cadena

    f1.read(reinterpret_cast<char*>(&p), sizeof(p));
    if (f1.gcount() != sizeof(p)) cerr << "Error al leer p" << endl;

    f1.close();
    cout << "Lectura de datos desde el archivo entrada.dat completada." << endl;

    // Imprimir los datos leídos
    cout << "\nDatos leídos desde entrada.dat:" << endl;
    cout << "x: " << x << endl;
    cout << "f: " << f << endl;
    cout << "a: ";
    for (int i = 0; i < 5; ++i) { // Solo las primeras 5 posiciones de a
        cout << a[i] << " ";
    }
    cout << endl;
    cout << "nombre: " << nombre << endl;
    cout << "p.b: " << p.b << endl;
    cout << "p.s: " << p.s << endl;

    return 0;
}
```

EJECUCIÓN:

```
PS D:\PSTeo-A\GUIA_06\Ex02\output> & .\'main.exe'
Directorio de trabajo actual: "pwd" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.
Verificando existencia y permisos del archivo entrada.dat
El fichero entrada.dat existe y es accesible.
Intentando abrir el archivo entrada.dat para lectura.
Archivo entrada.dat abierto exitosamente para lectura.
Intentando abrir el archivo salida.dat para escritura.
Archivo salida.dat abierto exitosamente para escritura.
Datos escritos en el archivo salida.dat.
Lectura de datos desde el archivo entrada.dat completada.

Datos le├¡dos desde entrada.dat:
x: 42
f: 3.14
a: 1 2 3 4 5
nombre: Ejercicios
p.b: 123
p.s: Estructura de datos
PS D:\PSTeo-A\GUIA_06\Ex02\output>
```

EJERCICIO 03:

Crear un programa que pida al usuario que teclee frases, y las almacene en el fichero “frases.txt”. Acabará cuando el usuario pulse Intro sin teclear nada. Después deberá mostrar el contenido del fichero.

```cpp
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

int main() {
    string frase;
    ofstream outfile("frases.txt", ios::app); // Abrir en modo de añadir al final del
 archivo

    if (!outfile) {
        cerr << "No se pudo abrir el archivo para escritura." << endl;
```

```
        return 1;
    }

    cout << "Teclee frases (pulse Enter sin teclear nada para finalizar):" << endl;
    while (true) {
        getline(cin, frase);
        if (frase.empty()) {
            break; // Finaliza si el usuario pulsa Enter sin teclear nada
        }
        outfile << frase << endl;
    }

    outfile.close();

    ifstream infile("frases.txt");
    if (!infile) {
        cerr << "No se pudo abrir el archivo para lectura." << endl;
        return 1;
    }

    cout << "\nContenido del fichero 'frases.txt':" << endl;
    while (getline(infile, frase)) {
        cout << frase << endl;
    }

    infile.close();
    return 0;
}
```

EJECUCIÓN:

```
PS D:\PSTeo-A\GUIA_06\Ex03\output> & .\'main.exe'
Teclee frases (pulse Enter sin teclear nada para finalizar):
La vida es bella, no te suicides.
No es tarde, nunca es tarde.


Contenido del fichero 'frases.txt':
La esperanza es lo último que se pierde
La vida es bella, no te suicides.
No es tarde, nunca es tarde.
PS D:\PSTeo-A\GUIA_06\Ex03\output>
```

EJERCICIO 4

Un programa que pregunte un nombre de fichero y muestre en pantalla el contenido de ese fichero, haciendo una pausa después de cada 25 líneas, para que dé tiempo a leerlo. Cuando el usuario pulse intro, se mostrarán las siguientes 25 líneas, y así hasta que termine el fichero.

```cpp
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

void mostrarArchivoPorPantalla(const string& nombreArchivo) {
    ifstream archivo(nombreArchivo);  // Abrir el archivo

    if (!archivo.is_open()) {
        cerr << "No se pudo abrir el archivo " << nombreArchivo << endl;
        return;
    }
    string linea;
    int contadorLineas = 0;

    // Leer y mostrar el archivo línea por línea
    while (getline(archivo, linea)) {
        cout << linea << endl;
        ++contadorLineas;

        // Pausa cada 25 líneas
        if (contadorLineas % 25 == 0) {
```

```
            cout << "Presiona Enter para mostrar las siguientes 25 líneas..." << endl;
            cin.get();  // Esperar a que el usuario presione Enter
        }
    }
    archivo.close();  // Cerrar el archivo al finalizar
}

int main() {
    string nombreArchivo;

    // Pedir al usuario el nombre del archivo
    cout << "Ingrese el nombre del archivo: ";
    getline(cin, nombreArchivo);

    // Mostrar el contenido del archivo
    mostrarArchivoPorPantalla(nombreArchivo);

    return 0;
}
```

EJECUCIÓN:

```
PS D:\PSTeo-A\GUIA_06\Ex04> g++ .\main.cpp
PS D:\PSTeo-A\GUIA_06\Ex04> .\a.exe
Ingrese el nombre del archivo: ejemplo.txt
1. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
2. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
3. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
4. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
5. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
6. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus.
7. Nullam quis risus eget urna mollis ornare vel eu leo.
8. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.
9. Donec ullamcorper nulla non metus auctor fringilla.
10. Cras justo odio, dapibus ac facilisis in, egestas eget quam.
11. Maecenas sed diam eget risus varius blandit sit amet non magna.
12. Cras mattis consectetur purus sit amet fermentum.
13. Integer posuere erat a ante venenatis dapibus posuere velit aliquet.
14. Donec ullamcorper nulla non metus auctor fringilla.
15. Morbi leo risus, porta ac consectetur ac, vestibulum at eros.
16. Morbi leo risus, porta ac consectetur ac, vestibulum at eros.
17. Aenean eu leo quam. Pellentesque ornare sem lacinia quam venenatis vestibulum.
18. Nullam id dolor id nibh ultricies vehicula ut id elit.
19. Maecenas faucibus mollis interdum.
20. Maecenas sed diam eget risus varius blandit sit amet non magna.
21. Maecenas faucibus mollis interdum.
22. Cras justo odio, dapibus ac facilisis in, egestas eget quam.
23. Integer posuere erat a ante venenatis dapibus posuere velit aliquet.
24. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.
25. Curabitur blandit tempus porttitor. Maecenas sed diam eget risus varius blandit sit amet non magna.
```

```
26. Nullam quis risus eget urna mollis ornare vel eu leo.
27. Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor.
28. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
29. Cras justo odio, dapibus ac facilisis in, egestas eget quam.
30. Maecenas faucibus mollis interdum.
31. Aenean lacinia bibendum nulla sed consectetur.
32. Nullam quis risus eget urna mollis ornare vel eu leo.
33. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.
34. Nullam quis risus eget urna mollis ornare vel eu leo.
35. Vestibulum id ligula porta felis euismod semper.
36. Integer posuere erat a ante venenatis dapibus posuere velit aliquet.
37. Nullam quis risus eget urna mollis ornare vel eu leo.
38. Nulla vitae elit libero, a pharetra augue.
39. Donec ullamcorper nulla non metus auctor fringilla.
40. Sed posuere consectetur est at lobortis.
41. Donec ullamcorper nulla non metus auctor fringilla.
42. Etiam porta sem malesuada magna mollis euismod.
43. Sed posuere consectetur est at lobortis.
44. Donec id elit non mi porta gravida at eget metus.
45. Aenean lacinia bibendum nulla sed consectetur.
46. Nulla vitae elit libero, a pharetra augue.
47. Donec ullamcorper nulla non metus auctor fringilla.
48. Donec ullamcorper nulla non metus auctor fringilla.
49. Sed posuere consectetur est at lobortis.
50. Nullam id dolor id nibh ultricies vehicula ut id elit.
Presiona Enter para mostrar las siguientes 25 l├¡neas...
```

EJERCICIO 5

Crear una agenda que maneje los siguientes datos: nombre, dirección, tlf móvil, email, y día, mes y año de nacimiento (estos tres últimos datos deberán ser números enteros cortos). Deberá tener capacidad para 100 fichas. Se deberá poder añadir un dato nuevo, visualizar los nombres de las fichas existentes, o mostrar todos los datos de una persona (se preguntará al usuario cual es el nombre de esa persona que quiere visualizar). Al empezar el programa, leerá los datos de un fichero llamado “agenda.dat” (si existe). Al terminar, guardará todos los datos en ese fichero.

```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <sstream>
#include <algorithm>
using namespace std;

struct Contacto {
    string nombre;
    string direccion;
    string telefono;
    string email;
    short dia;
    short mes;
    short anio;
};
```

```
class Agenda {
private:
    vector<Contacto> contactos;
    const string NOMBRE_ARCHIVO = "agenda.dat";

public:
    Agenda() {
        cargarDatos();
    }

    ~Agenda() {
        guardarDatos();
    }

    void agregarContacto() {
    Contacto nuevo;

    cout << "Nombre: ";
    cin.ignore(); // Limpiamos el buffer de entrada
    getline(cin, nuevo.nombre);

    cout << "Dirección: ";
    getline(cin, nuevo.direccion);

    cout << "Teléfono móvil: ";
    getline(cin, nuevo.telefono);

    cout << "Email: ";
    getline(cin, nuevo.email);

    cout << "Día de nacimiento: ";
    cin >> nuevo.dia;

    cout << "Mes de nacimiento: ";
    cin >> nuevo.mes;

    cout << "Año de nacimiento: ";
```

```
    cin >> nuevo.anio;

    contactos.push_back(nuevo);
    cout << "Contacto agregado exitosamente." << endl;
}



    void mostrarNombres() {
        if (contactos.empty()) {
            cout << "No hay contactos en la agenda." << endl;
            return;
        }

        cout << "Nombres en la agenda:" << endl;
        for (const auto& contacto : contactos) {
            cout << contacto.nombre << endl;
        }
    }

    void mostrarContacto() {
        if (contactos.empty()) {
            cout << "No hay contactos en la agenda." << endl;
            return;
        }

        string nombre;
        cout << "Ingrese el nombre del contacto a buscar: ";
        cin.ignore();
        getline(cin, nombre);

        auto it = find_if(contactos.begin(), contactos.end(), [&nombre](const
Contacto& c) {
            return c.nombre == nombre;
        });

        if (it != contactos.end()) {
            mostrarDetalleContacto(*it);
```

```
        } else {
            cout << "Contacto no encontrado." << endl;
        }
    }

    void eliminarContacto() {
        if (contactos.empty()) {
            cout << "No hay contactos en la agenda." << endl;
            return;
        }

        string nombre;
        cout << "Ingrese el nombre del contacto a eliminar: ";
        cin.ignore();
        getline(cin, nombre);

        auto it = remove_if(contactos.begin(), contactos.end(), [&nombre](const
 Contacto& c) {
            return c.nombre == nombre;
        });

        if (it != contactos.end()) {
            contactos.erase(it, contactos.end());
            cout << "Contacto eliminado exitosamente." << endl;
        } else {
            cout << "Contacto no encontrado." << endl;
        }
    }

private:
    void mostrarDetalleContacto(const Contacto& contacto) {
        cout << "Nombre: " << contacto.nombre << endl;
        cout << "Dirección: " << contacto.direccion << endl;
        cout << "Teléfono móvil: " << contacto.telefono << endl;
        cout << "Email: " << contacto.email << endl;
        cout << "Fecha de nacimiento: " << contacto.dia << "/" << contacto.mes << "/"
 << contacto.anio << endl;
```

```
    }

    void cargarDatos() {
        ifstream archivo(NOMBRE_ARCHIVO);
        if (archivo.is_open()) {
            string linea;
            while (getline(archivo, linea)) {
                contactos.push_back(parseContacto(linea));
            }
            archivo.close();
            cout << "Datos cargados desde " << NOMBRE_ARCHIVO << endl;
        }
    }

    void guardarDatos() {
        ofstream archivo(NOMBRE_ARCHIVO);
        if (archivo.is_open()) {
            for (const auto& contacto : contactos) {
                archivo << contacto.nombre << "|" << contacto.direccion << "|" <<
contacto.telefono << "|" << contacto.email << "|" << contacto.dia << "|" <<
contacto.mes << "|" << contacto.anio << endl;
            }
            archivo.close();
            cout << "Datos guardados en " << NOMBRE_ARCHIVO << endl;
        } else {
            cout << "No se pudo guardar los datos en " << NOMBRE_ARCHIVO << endl;
        }
    }

    Contacto parseContacto(const string& linea) {
        Contacto contacto;
        stringstream ss(linea);
        getline(ss, contacto.nombre, '|');
        getline(ss, contacto.direccion, '|');
        getline(ss, contacto.telefono, '|');
        getline(ss, contacto.email, '|');
        ss >> contacto.dia;
```

```
        ss.ignore();
        ss >> contacto.mes;
        ss.ignore();
        ss >> contacto.anio;
        return contacto;
    }
};

int main() {
    Agenda agenda;
    int opcion;

    do {
        cout << "\n--- Menú ---" << endl;
        cout << "1. Agregar contacto" << endl;
        cout << "2. Mostrar nombres" << endl;
        cout << "3. Mostrar datos de un contacto" << endl;
        cout << "4. Eliminar contacto" << endl;
        cout << "5. Salir" << endl;
        cout << "Ingrese su opción: ";

        while (!(cin >> opcion)) {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Entrada inválida. Por favor, ingrese un número: ";
        }

        switch (opcion) {
            case 1:
                agenda.agregarContacto();
                break;
            case 2:
                agenda.mostrarNombres();
                break;
            case 3:
                agenda.mostrarContacto();
                break;
```

```
            case 4:
                agenda.eliminarContacto();
                break;
            case 5:
                cout << "Saliendo del programa..." << endl;
                break;
            default:
                cout << "Opción inválida. Intente de nuevo." << endl;
        }
    } while (opcion != 5);

    return 0;
}
```

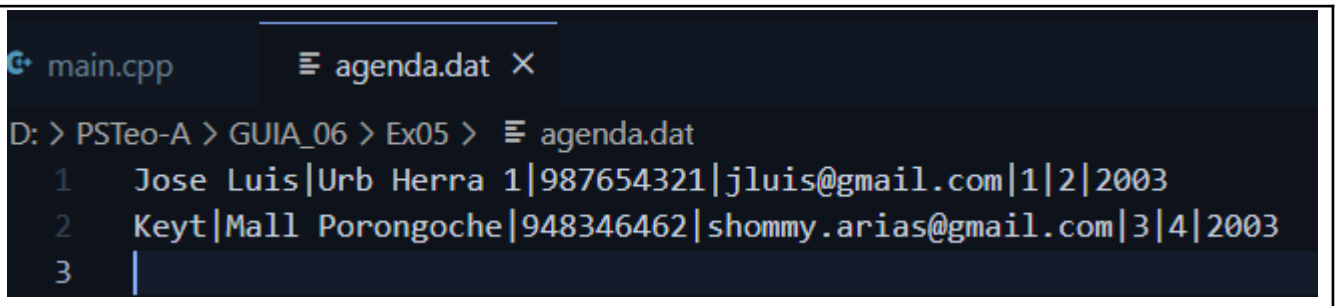
EJECUCIÓN:

```
--- Men├‖ ---
1. Agregar contacto
2. Mostrar nombres
3. Mostrar datos de un contacto
4. Eliminar contacto
5. Salir
Ingrese su opci├│n: 1
Nombre: Keyt
Direcci├│n: Mall Porongoche
Tel├⌐fono m├│vil: 948346462
Email: shommy.arias@gmail.com
D├¡a de nacimiento: 03
Mes de nacimiento: 04
A├▒o de nacimiento: 2003
Contacto agregado exitosamente.
```

```
--- Men├‖ ---
1. Agregar contacto
2. Mostrar nombres
3. Mostrar datos de un contacto
4. Eliminar contacto
5. Salir
Ingrese su opci├│n: 2
Nombres en la agenda:
Jose Luis
Keyt
```

```
--- Men├‖ ---
1. Agregar contacto
2. Mostrar nombres
3. Mostrar datos de un contacto
4. Eliminar contacto
5. Salir
Ingrese su opci├│n: 5
Saliendo del programa...
```

agenda.dat Después de la ejecución:



II. SOLUCIÓN DEL CUESTIONARIO

1. **¿Es necesario tener una función para cerrar un archivo?**
   Sí, es una buena práctica tener una función para cerrar un archivo después de haber terminado de trabajar con él. Aunque muchos sistemas operativos liberarán los recursos asociados con un archivo cuando el programa termine, cerrar explícitamente un archivo cuando ya no se necesita es una práctica recomendada para asegurar que los recursos se liberen de manera oportuna, especialmente en programas más grandes o complejos donde manejar múltiples archivos puede ser común.
2. **¿Cuántos tipos de permisos tienen los archivos en un sistema operativo basado en una distribución Linux?**
   En un sistema operativo basado en una distribución Linux, los archivos pueden tener tres tipos principales de permisos: lectura (read), escritura (write) y ejecución (execute). Estos permisos se aplican para tres tipos de usuarios: propietario (owner), grupo (group) y otros (others). Por lo tanto, hay un total de tres tipos de permisos (lectura, escritura y ejecución) y tres categorías de usuarios (propietario, grupo y otros), lo que suma un total de nueve combinaciones posibles de permisos.

III. CONCLUSIONES

El manejo de ficheros es un aspecto esencial en el desarrollo de software, ya que permite la persistencia de datos más allá del ciclo de vida de una aplicación. La comprensión y la correcta implementación de técnicas de manejo de ficheros aseguran que los datos se almacenen y se recuperen de manera eficiente y segura, lo cual es crucial para la robustez y la fiabilidad de cualquier programa.

REFERENCIAS Y BIBLIOGRAFÍA

[1] P.J. Deitel and H.M. Deitel, "Cómo Programar en C++", México, Ed. Pearson Educación, 2009

[2] B. Stroustrup, "El Lenguaje de Programación C++", Madrid, Adisson Pearson Educación, 2002

[3] B. Eckel, "Thinking in C++", Prentice Hall, 2000