Ejercicio de Librería en C++: Gestión de Empleados (Versión de 2 Archivos)

Este ejercicio sigue simulando un sistema básico de gestión de empleados, pero la librería se consolidará en un solo archivo.

Archivo de Librería (libreria_empleados.cpp)

Este archivo actuará como la librería completa, conteniendo tanto las declaraciones de la estructura y las funciones como sus implementaciones.

C++

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

// Estructura para representar a un empleado
struct Empleado {
    int id;
    std::string nombre;
    std::string apellido;
    double salario;
    std::string departamento;
};

// --- Procedimientos (funciones void) ---

// Procedimiento para inicializar una lista de empleados
void inicializarEmpleados(std::vector<Empleado>& empleados) {
    empleados.clear(); // Limpia cualquier dato existente
    std::cout << "Lista de empleados inicializada vacia." << std::endl;
}
```

```cpp
// Procedimiento para mostrar los detalles de un solo empleado
void mostrarEmpleado(const Empleado& emp) {
    std::cout << "ID: " << emp.id
              << ", Nombre: " << emp.nombre
              << ", Apellido: " << emp.apellido
              << ", Salario: " << emp.salario
              << ", Departamento: " << emp.departamento << std::endl;
}

// Procedimiento para listar todos los empleados
void listarEmpleados(const std::vector<Empleado>& empleados) {
    if (empleados.empty()) {
        std::cout << "No hay empleados registrados." << std::endl;
        return;
    }
    std::cout << "\n--- Lista de Empleados ---" << std::endl;
    for (size_t i = 0; i < empleados.size(); ++i) {
        mostrarEmpleado(empleados[i]);
    }
    std::cout << "--------------------------" << std::endl;
}

// --- Funciones (retornan un valor) ---

// Función para buscar un empleado por ID (declarada antes para que agregarEmpleado pueda usarla)
// Retorna el índice del empleado en el vector, o -1 si no se encuentra
int buscarEmpleadoPorId(const std::vector<Empleado>& empleados, int idBuscado) {
    for (size_t i = 0; i < empleados.size(); ++i) {
        if (empleados[i].id == idBuscado) {
            return static_cast<int>(i); // Retorna el índice si lo encuentra
        }
    }
    return -1; // Retorna -1 si no lo encuentra
}

// Función para agregar un nuevo empleado
// Retorna true si el empleado fue agregado exitosamente, false en caso contrario
bool agregarEmpleado(std::vector<Empleado>& empleados, int id, const std::string& nombre, const
std::string& apellido, double salario, const std::string& departamento) {
    // Verificar si el ID ya existe usando la función buscarEmpleadoPorId
    if (buscarEmpleadoPorId(empleados, id) != -1) {
```

```
        std::cout << "Error: Ya existe un empleado con el ID " << id << std::endl;
        return false;
    }

    Empleado nuevoEmpleado;
    nuevoEmpleado.id = id;
    nuevoEmpleado.nombre = nombre;
    nuevoEmpleado.apellido = apellido;
    nuevoEmpleado.salario = salario;
    nuevoEmpleado.departamento = departamento;

    empleados.push_back(nuevoEmpleado);
    std::cout << "Empleado " << nombre << " " << apellido << " agregado exitosamente." << std::endl;
    return true;
}
```

Archivo Principal (main.cpp)

Este archivo contendrá la lógica principal del programa e incluirá directamente el archivo de librería.

C++

```
#include <iostream>
#include <vector>
#include <string> // Necesario si se usan strings directamente aquí
#include <limits> // Para std::numeric_limits

// Incluir el archivo de la librería directamente
#include "libreria_empleados.cpp"

// Función auxiliar para limpiar el buffer de entrada
void limpiarBufferEntrada() {
    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
}
```

```cpp
int main() {
    std::vector<Empleado> misEmpleados; // Vector para almacenar a los empleados

    inicializarEmpleados(misEmpleados);

    // Agregando algunos empleados
    std::cout << "\n--- Agregando empleados ---" << std::endl;
    agregarEmpleado(misEmpleados, 101, "Ana", "Gomez", 35000.0, "Ventas");
    agregarEmpleado(misEmpleados, 102, "Luis", "Perez", 42000.0, "Marketing");
    agregarEmpleado(misEmpleados, 103, "Maria", "Lopez", 50000.0, "Recursos Humanos");
    agregarEmpleado(misEmpleados, 101, "Pedro", "Diaz", 30000.0, "IT"); // Intento de agregar ID duplicado

    // Listar todos los empleados
    listarEmpleados(misEmpleados);

    // Buscar un empleado por ID
    std::cout << "\n--- Buscando empleado por ID ---" << std::endl;
    int idABuscar = 102;
    int indiceEncontrado = buscarEmpleadoPorId(misEmpleados, idABuscar);
    if (indiceEncontrado != -1) {
        std::cout << "Empleado con ID " << idABuscar << " encontrado: ";
        mostrarEmpleado(misEmpleados[indiceEncontrado]);
    } else {
        std::cout << "Empleado con ID " << idABuscar << " no encontrado." << std::endl;
    }

    idABuscar = 999; // ID que no existe
    indiceEncontrado = buscarEmpleadoPorId(misEmpleados, idABuscar);
    if (indiceEncontrado != -1) {
        std::cout << "Empleado con ID " << idABuscar << " encontrado: ";
        mostrarEmpleado(misEmpleados[indiceEncontrado]);
    } else {
        std::cout << "Empleado con ID " << idABuscar << " no encontrado." << std::endl;
    }

    // Ejemplo de interacción con el usuario
    std::cout << "\n--- Interaccion con el usuario ---" << std::endl;
    int nuevoId;
    std::string nuevoNombre, nuevoApellido, nuevoDepartamento;
```

```
    double nuevoSalario;

    std::cout << "Ingrese ID del nuevo empleado: ";
    std::cin >> nuevoId;
    limpiarBufferEntrada(); // Limpiar el buffer después de leer int

    std::cout << "Ingrese Nombre del nuevo empleado: ";
    std::getline(std::cin, nuevoNombre);

    std::cout << "Ingrese Apellido del nuevo empleado: ";
    std::getline(std::cin, nuevoApellido);

    std::cout << "Ingrese Salario del nuevo empleado: ";
    std::cin >> nuevoSalario;
    limpiarBufferEntrada(); // Limpiar el buffer después de leer double

    std::cout << "Ingrese Departamento del nuevo empleado: ";
    std::getline(std::cin, nuevoDepartamento);

    agregarEmpleado(misEmpleados, nuevoId, nuevoNombre, nuevoApellido, nuevoSalario,
nuevoDepartamento);

    listarEmpleados(misEmpleados); // Listar nuevamente para ver el nuevo empleado

    return 0;
}
```

Salida por Consola Esperada

La salida por consola será **exactamente la misma** que en el ejemplo anterior, ya que la funcionalidad no ha cambiado, solo la organización de los archivos.

```
Lista de empleados inicializada vacia.

--- Agregando empleados ---
Empleado Ana Gomez agregado exitosamente.
Empleado Luis Perez agregado exitosamente.
Empleado Maria Lopez agregado exitosamente.
Error: Ya existe un empleado con el ID 101
```

--- Lista de Empleados ---
ID: 101, Nombre: Ana, Apellido: Gomez, Salario: 35000, Departamento: Ventas
ID: 102, Nombre: Luis, Apellido: Perez, Salario: 42000, Departamento: Marketing
ID: 103, Nombre: Maria, Apellido: Lopez, Salario: 50000, Departamento: Recursos Humanos
----

--- Buscando empleado por ID ---
Empleado con ID 102 encontrado: ID: 102, Nombre: Luis, Apellido: Perez, Salario: 42000, Departamento: Marketing
Empleado con ID 999 no encontrado.

--- Interaccion con el usuario ---
Ingrese ID del nuevo empleado: 104
Ingrese Nombre del nuevo empleado: Carlos
Ingrese Apellido del nuevo empleado: Garcia
Ingrese Salario del nuevo empleado: 40000
Ingrese Departamento del nuevo empleado: IT
Empleado Carlos Garcia agregado exitosamente.

--- Lista de Empleados ---
ID: 101, Nombre: Ana, Apellido: Gomez, Salario: 35000, Departamento: Ventas
ID: 102, Nombre: Luis, Apellido: Perez, Salario: 42000, Departamento: Marketing
ID: 103, Nombre: Maria, Apellido: Lopez, Salario: 50000, Departamento: Recursos Humanos
ID: 104, Nombre: Carlos, Apellido: Garcia, Salario: 40000, Departamento: IT
