



Universidad
del Valle de México
LAUREATE INTERNATIONAL UNIVERSITIES®

Actividad 4

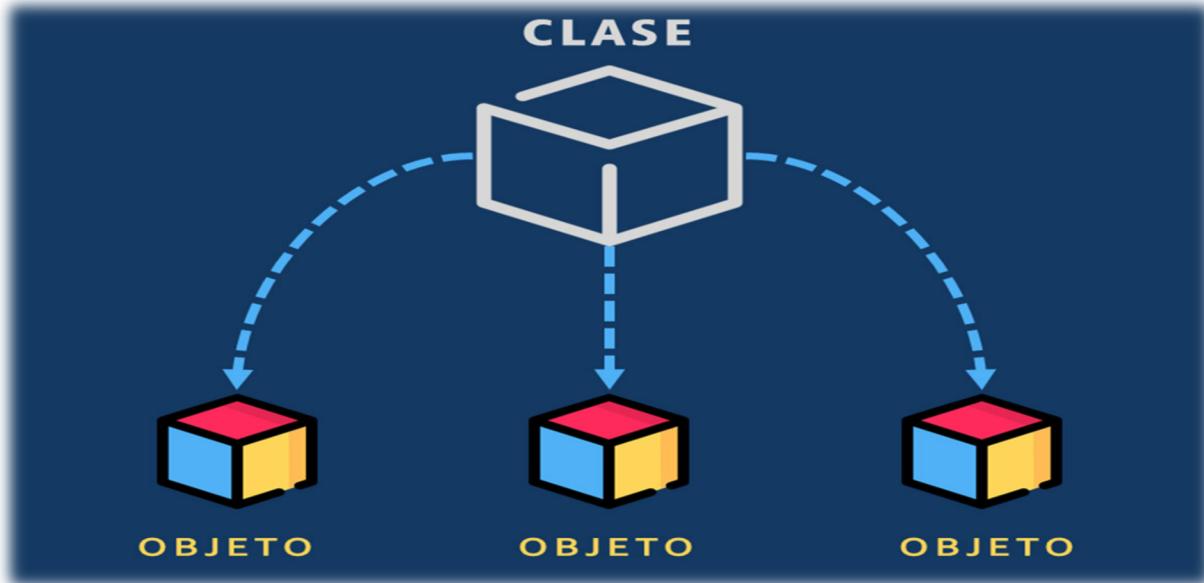
Proyecto integrador etapa 1

Programación orientada a objetos

Docente. Ing. Saúl Santiago Rivera

José Emiliano Jauregui Guzmán

Por siempre responsable de lo que se ha cultivado



Análisis y modelado de sistema

Introducción

Esta actividad consiste en aplicar los conocimientos adquiridos a lo largo del curso y retomar lo aprendido en cada una de las actividades realizadas, lo que garantiza la transversalidad de los contenidos revisados para fortalecer el desarrollo de competencias y lograr el fin de formación planteado.

Objetivo

El objetivo del Proyecto integrador es desarrollar el prototipo de un sistema de control escolar básico haciendo uso de las diferentes características de la programación orientada a objetos con el fin de evidenciar las ventajas del paradigma.

Planteamiento

Cierta institución educativa ha solicitado el desarrollo de un prototipo de sistema de control escolar para la gestión de alumnos, profesores y materias. El sistema debe permitir registrar y eliminar información.

- Para cada alumno se requiere almacenar Nombre, Apellido, No. de Control Escolar, Edad, Sexo, Semestre y el conjunto de materias que cursará.
- Para cada materia debemos saber Nombre, Número de créditos y profesor que la imparte.
- De los profesores necesitamos Nombre, Apellido, Sexo, Edad, Título o profesión y No. de Cédula profesional.

I. Análisis y modelado de sistemas

1.1 Identificación de clases y objetos

- Analiza con detenimiento lo que se indica en el planteamiento e identifica las clases y objetos, así como sus atributos y métodos necesarios para cumplir con los requerimientos del sistema control escolar que se solicita.

En este caso crearemos un interfaz en el cual nos permitirá acceder al sistema de control escolar para la gestión de alumnos, profesores y materias, los cuales identificaremos como las clases para dar registro a sus atributos para cada objeto independiente.

Me apoyare en un catálogo para que sea mas fácil identificar en un gráfico el escenario en el cual van a conectarse sus funciones que derivan a cada clases por un método que van a compartir las tres clases.



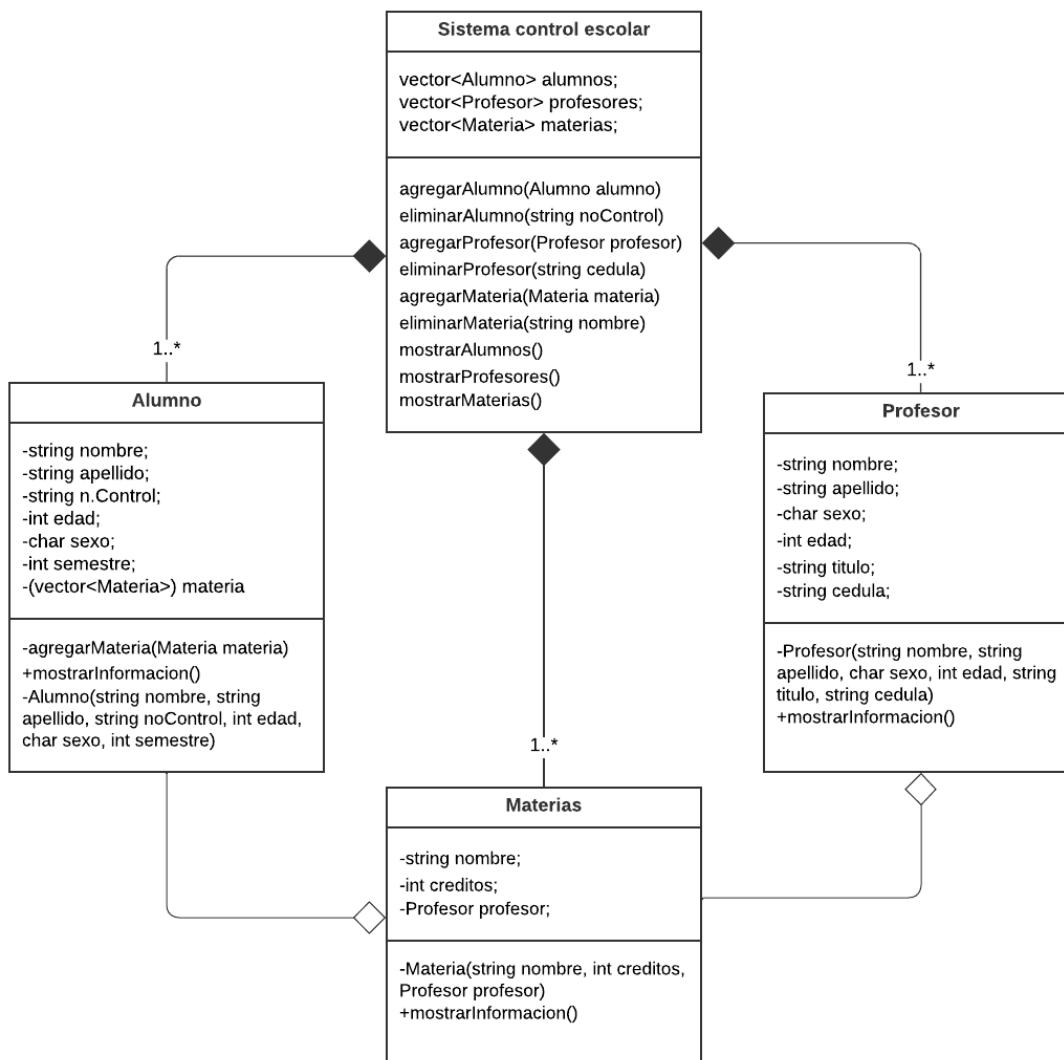
<p>Pantalla 1</p> <p>Sistema de control escolar:</p> <ol style="list-style-type: none"> 1. Registrar alumno 2. Eliminar alumno 3. Registrar Profesor 4. Eliminar Profesor 5. Registrar Materia 6. Eliminar Materia 7. Mostrar Alumnos 8. Mostrar Profesores 9. Mostrar Materias 10. Salir <p>Seleccione una opción (1..10):</p>	<p>Pantalla 2</p> <p>Ingrese el nombre del alumno: Ingrese el apellido del alumno Ingrese el No. de Control: Ingrese la edad del alumno: Ingrese el semestre: Ingrese el sexo (M/F):</p>
<p>Pantalla 3</p> <p>Ingrese el No. de Control del alumno a eliminar:</p>	<p>Pantalla 4</p> <p>Ingrese el nombre del profesor: Ingrese el apellido del profesor: Ingrese la edad del profesor: Ingrese el título del profesor: Ingrese la cédula profesional del profesor: Ingrese el sexo (M/F):</p>
<p>Pantalla 5</p> <p>Ingrese la cédula profesional del profesor a eliminar:</p>	<p>Pantalla 6</p> <p>Ingrese el nombre de la materia: Ingrese los créditos de la materia: Ingrese el nombre del profesor que imparte la materia: Ingrese el apellido del profesor: Ingrese la cédula del profesor:</p>
<p>Pantalla 7</p> <p>Ingrese el nombre de la materia a eliminar:</p>	<p>Pantalla 8</p> <p>Mostrar Alumnos</p>



Pantalla 9	Pantalla 10
Mostrar Profesores	Mostrar Materias

1.2 Diagrama de clases y establecimiento de relaciones

Realiza el **diagrama de clases** de las entidades identificadas y establece las relaciones de las mismas. Incluye una explicación de cada entidad y de sus propiedades y métodos, así como, de la relación entre las mismas.



1.3 Integración de clases

En tu entorno de desarrollo realiza un programa en modo consola y en el integra cada una de las clases con sus respectivas propiedades. Los métodos se integrarán en la segunda etapa.

```

main.cpp
1 #include <iostream>
2 #include <vector>
3 #include <string>
4
5 using namespace std;
6
7 // Clase Profesor
8 class Profesor {
9 public://Atributos
10   string nombre;
11   string apellido;
12   char sexo;
13   int edad;
14   string titulo;
15   string cedula;
16 //Constructor, nos sirve para inicializar los atributos
17   Profesor(string nombre, string apellido, char sexo, int edad, string titulo, string cedula)
18     : nombre(nombre), apellido(apellido), sexo(sexo), edad(edad), titulo(titulo), cedula(cedula) {}
19 };
20
21 // Clase Materia
22 class Materia {
23 public://Atributos
24   string nombre;
25   int creditos;
26   Profesor profesor;
27 //Constructor, nos sirve para inicializar los atributos
28   Materia(string nombre, int creditos, Profesor profesor)
29     : nombre(nombre), creditos(creditos), profesor(profesor) {}
30 };
31
32 // Clase Alumno
33 class Alumno {
34 public://Atributos
35   string nombre;
36   string apellido;
37   string noControl;
38   int edad;
39   char sexo;
40   int semestre;
41   vector<Materia> materias;
42
43 //Constructor, nos sirve para inicializar los atributos
44   Alumno(string nombre, string apellido, string noControl, int edad, char sexo, int semestre)
45     : nombre(nombre), apellido(apellido), noControl(noControl), edad(edad), sexo(sexo), semestre(semestre) {}

// Clase SistemaControlEscolar
class SistemaControlEscolar {
private://Objetos
  vector<Alumno> alumnos;
  vector<Profesor> profesores;
  vector<Materia> materias;
}

```



```

#include <iostream>
#include <vector>
#include <string>

using namespace std;

// Clase Profesor
class Profesor {
public://Atributos
    string nombre;
    string apellido;
    char sexo;
    int edad;
    string titulo;
    string cedula;
//Constructor, nos sirve para inicializar los atributos
    Profesor(string nombre, string apellido, char sexo, int edad, string titulo, string
cedula)
        : nombre(nombre), apellido(apellido), sexo(sexo), edad(edad), titulo(titulo),
cedula(cedula) {}
};

// Clase Materia
class Materia {
public://Atributos
    string nombre;
    int creditos;
    Profesor profesor;
//Constructor, nos sirve para inicializar los atributos
    Materia(string nombre, int creditos, Profesor profesor)
        : nombre(nombre), creditos(creditos), profesor(profesor) {}
};

// Clase Alumno
class Alumno {
public://Atributos
    string nombre;
    string apellido;
    string noControl;
    int edad;
    char sexo;
    int semestre;
    vector<Materia> materias;

//Constructor, nos sirve para inicializar los atributos
    Alumno(string nombre, string apellido, string noControl, int edad, char sexo, int
semestre)

```



```
: nombre(nombre), apellido(apellido), noControl(noControl), edad(edad),
sexo(sexo), semestre(semestre) {}
```

```
// Clase SistemaControlEscolar
class SistemaControlEscolar {
private://Objetos
vector<Alumno> alumnos;
vector<Profesor> profesores;
vector<Materia> materias;
```

Conclusión

Para este primer paso del proyecto integrador me resultó interesante ver la manera de generar un interfaz de menú de igual manera me basé un poco de la materia de lógica y programación estructurada en la interfaz de un menú para el registro, eliminación o como en este proyecto mostrar los datos ya registrados por ello fue un poco más fácil implementar las funciones ya que hubo un parecido similar, de igual manera tuve mucha ayuda tanto de páginas y PDF para comprender un poquito ciertas líneas de código que no sabía exactamente cómo eran para solicitar ciertas acciones. Igualmente cuando llegué al último punto del de la actividad me di cuenta que únicamente sería insertar las clases y sus atributos, no había llegado yo a esa parte porque en el inciso uno me desarrollé la idea del interfaz completo y me expandí un poco pero únicamente muestro el avance que solicita la actividad de igual manera para la siguiente etapa puedo tener en cuenta el desarrollo que ya tuve previamente en este proyecto para mejorarlo y tener una mejor comprensión del tema.

Referencia

Lucichart en español (Productor). Tutorial - Diagrama de Clases UML. Recuperado el 25 de noviembre del 2024 de, <https://www.youtube.com/watch?v=Z0yLerU0g-Q>

Programación ATS. (Productor). (09 de enero de 2017) 127. Programación en C++ || POO || Clases y objetos en C++ Recuperado el 25 de noviembre del 2024 de, <https://www.youtube.com/watch?v=tbVHbfIVxs4&t=25s>

del Rio, F. M. (n.d.). *FUNDAMENTOS BASICOS DE PROGRAMACION EN C++*. Ujaen.Es. Recuperado el 25 de noviembre del 2024 de, <https://www4.ujaen.es/~fmartin/apuntesC++.pdf>

Ceballos, J. (2007). Programación orientada a objetos con C++. Haga clic para ver más opciones [Recuperado el 25 de noviembre del 2024 de, https://kupdf.net/download/ceballos-programaci-oacute-n-orientada-a-objetos-con-c-4ed_58a8b3476454a7596fb1e8d9_.pdf



ChatGPT. (n.d.). Chatgpt.com. Recuperado el 25 de noviembre del 2024 de,
<https://chatgpt.com>

García, F., Pardo, C. (s.f.). Introducción al Análisis y Diseño Orientado a Objetos
++ Haga clic para ver más opciones. Recuperado el 25 de noviembre del 2024 de,
<https://repositorio.grial.eu/bitstream/grial/265/1/AOO.pdf>





Universidad
del Valle de México
LAUREATE INTERNATIONAL UNIVERSITIES®

Actividad 6

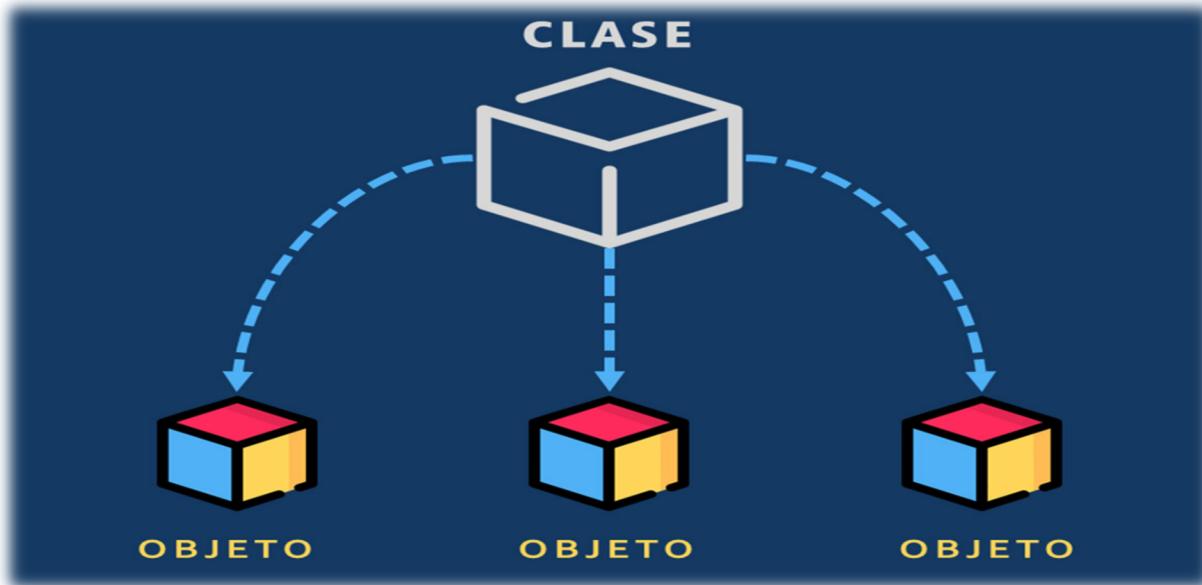
Proyecto integrador etapa 2

Programación orientada a objetos

Docente. Ing. Saúl Santiago Rivera

José Emiliano Jauregui Guzmán

Por siempre responsable de lo que se ha cultivado



Implementación de métodos e integración de relaciones

Introducción

Esta actividad consiste en aplicar los conocimientos adquiridos a lo largo del curso y retomar lo aprendido en cada una de las actividades realizadas, lo que garantiza la transversalidad de los contenidos revisados para fortalecer el desarrollo de competencias y lograr el fin de formación planteado.

Objetivo

El objetivo del Proyecto integrador es desarrollar el prototipo de un sistema de control escolar básico haciendo uso de las diferentes características de la programación orientada a objetos con el fin de evidenciar las ventajas del paradigma.

2. En la etapa 1 del Proyecto identificaste las clases necesarias para el requerimiento solicitado, las clases que debes haber programado son: materia, alumno y profesor cada una con sus respectivas propiedades. Se sugiere que entre las propiedades incluyas un ID, para identificar cada elemento. Realiza los ajustes necesarios si tus Clases no corresponden o justifica la diferencia.

II. Implementación de métodos e integración de relaciones

2.1 Definición e implementación de métodos

- Define e implementa los métodos para:

a. Adicionar registros de cada uno de los objetos

```
// Método para agregar una materia
void agregarMateria(Materia materia) {
    materias.push_back(materia);
}
```

```
// Método para agregar una materia
void agregarMateria(Materia materia) {
    materias.push_back(materia);
}
```

```
72 public:
73     // Funciones para agregar, eliminar y mostrar
74     void agregarAlumno(Alumno alumno) {
75         alumnos.push_back(alumno);
76     }
```

```
// Funciones para agregar, eliminar y mostrar
void agregarAlumno(Alumno alumno) {
    alumnos.push_back(alumno);
```



```

}
89 void agregarProfesor(Profesor profesor) {
90     profesores.push_back(profesor);
91 }
92
void agregarProfesor(Profesor profesor) {
    profesores.push_back(profesor);
}

```

b. Visualizar registros ingresando su ID

```

// Mostrar información de un Alumno por ID
void mostrarAlumnoPorID(string ID) const {
    for (const auto& alumno : alumnos) {
        if (alumno.ID == ID) {
            alumno.mostrarInformacion();
            return;
        }
    }
    cout << "Alumno con ID " << ID << " no encontrado." << endl;
}

```

```

// Mostrar información de un Alumno por ID
void mostrarAlumnoPorID(string ID) const {
    for (const auto& alumno : alumnos) {
        if (alumno.ID == ID) {
            alumno.mostrarInformacion();
            return;
        }
    }
    cout << "Alumno con ID " << ID << " no encontrado." << endl;
}

```

```

// Mostrar información de un Profesor por ID
void mostrarProfesorPorID(string ID) const {
    for (const auto& profesor : profesores) {
        if (profesor.ID == ID) {
            cout << "Profesor: " << profesor.nombre << " " << profesor.apellido << ", ID: "
                << profesor.ID << ", Cédula: " << profesor.cedula << ", Título: " << profesor.titulo << endl;
            return;
        }
    }
    cout << "Profesor con ID " << ID << " no encontrado." << endl;
}

```

```

// Mostrar información de un Profesor por ID
void mostrarProfesorPorID(string ID) const {
    for (const auto& profesor : profesores) {
        if (profesor.ID == ID) {
            cout << "Profesor: " << profesor.nombre << " " << profesor.apellido << ",
ID: "
                << profesor.ID << ", Cédula: " << profesor.cedula << ", Título: " <<
profesor.titulo << endl;
            return;
        }
    }
}

```



```

        }
        cout << "Profesor con ID " << ID << " no encontrado." << endl;
    }

// Mostrar información de una Materia por ID
void mostrarMateriaPorID(string ID) const {
    for (const auto& materia : materias) {
        if (materia.ID == ID) {
            cout << "Materia: " << materia.nombre << ", ID: " << materia.ID << ", Créditos: " << materia.creditos << ", Profesor: "
                << materia.profesor.nombre << " " << materia.profesor.apellido << endl;
            return;
        }
    }
    cout << "Materia con ID " << ID << " no encontrada." << endl;
}

// Mostrar información de una Materia por ID
void mostrarMateriaPorID(string ID) const {
    for (const auto& materia : materias) {
        if (materia.ID == ID) {
            cout << "Materia: " << materia.nombre << ", ID: " << materia.ID << ", Créditos: " << materia.creditos << ", Profesor: "
                << materia.profesor.nombre << " " << materia.profesor.apellido << endl;
            return;
        }
    }
    cout << "Materia con ID " << ID << " no encontrada." << endl;
}

```

Se recomienda la revisión de la siguiente referencia para apoyarte:

Universidad Nacional de Colombia (s.f.) Recuperado de
<https://dis.unal.edu.co/~fgonza/courses/2003/poo/c++.htm>

- En tu función principal *main()* solicita la captura de 2 profesores, 3 materias y 2 alumnos. Define el orden en que la información debe ser capturada para poder establecer la relación entre clases.



```

switch (opcion) {
    case 1: { // Registrar Alumno
        string nombre, apellido, noControl, ID;
        int edad, semestre;
        char sexo;
        cout << "Ingrese el nombre del alumno: ";
        nombre = leerString();
        cout << "Ingrese el apellido del alumno: ";
        apellido = leerString();
        cout << "Ingrese el No. de Control: ";
        noControl = leerString();
        cout << "Ingrese la edad del alumno: ";
        cin >> edad;
        cout << "Ingrese el semestre: ";
        cin >> semestre;
        cout << "Ingrese el sexo (M/F): ";
        cin >> sexo;
        cin.ignore();
        cout << "Ingrese el ID del alumno: ";
        ID = leerString();

        Alumno nuevoAlumno(nombre, apellido, noControl, edad, sexo, semestre, ID);
        sistema.agregarAlumno(nuevoAlumno);
        break;
    }
}

```

```

switch (opcion) {
    case 1: { // Registrar Alumno
        string nombre, apellido, noControl, ID;
        int edad, semestre;
        char sexo;
        cout << "Ingrese el nombre del alumno: ";
        nombre = leerString();
        cout << "Ingrese el apellido del alumno: ";
        apellido = leerString();
        cout << "Ingrese el No. de Control: ";
        noControl = leerString();
        cout << "Ingrese la edad del alumno: ";
        cin >> edad;
        cout << "Ingrese el semestre: ";
        cin >> semestre;
        cout << "Ingrese el sexo (M/F): ";
        cin >> sexo;
        cin.ignore();
        cout << "Ingrese el ID del alumno: ";
        ID = leerString();

        Alumno nuevoAlumno(nombre, apellido, noControl, edad, sexo,
semestre, ID);
        sistema.agregarAlumno(nuevoAlumno);
        break;
    }
}

```



```

case 3: { // Registrar Profesor
    string nombre, apellido, titulo, cedula, ID;
    int edad;
    char sexo;
    cout << "Ingrese el nombre del profesor: ";
    nombre = leerString();
    cout << "Ingrese el apellido del profesor: ";
    apellido = leerString();
    cout << "Ingrese la edad del profesor: ";
    cin >> edad;
    cout << "Ingrese el título del profesor: ";
    cin.ignore();
    titulo = leerString();
    cout << "Ingrese la cédula profesional del profesor: ";
    cedula = leerString();
    cout << "Ingrese el sexo (M/F): ";
    cin >> sexo;
    cin.ignore();
    cout << "Ingrese el ID del profesor: ";
    ID = leerString();

    Profesor nuevoProfesor(nombre, apellido, sexo, edad, titulo, cedula, ID);
    sistema.agregarProfesor(nuevoProfesor);
    break;
}

```

```

case 3: { // Registrar Profesor
    string nombre, apellido, titulo, cedula, ID;
    int edad;
    char sexo;
    cout << "Ingrese el nombre del profesor: ";
    nombre = leerString();
    cout << "Ingrese el apellido del profesor: ";
    apellido = leerString();
    cout << "Ingrese la edad del profesor: ";
    cin >> edad;
    cout << "Ingrese el título del profesor: ";
    cin.ignore();
    titulo = leerString();
    cout << "Ingrese la cédula profesional del profesor: ";
    cedula = leerString();
    cout << "Ingrese el sexo (M/F): ";
    cin >> sexo;
    cin.ignore();
    cout << "Ingrese el ID del profesor: ";
    ID = leerString();

    Profesor nuevoProfesor(nombre, apellido, sexo, edad, titulo, cedula, ID);
    sistema.agregarProfesor(nuevoProfesor);
    break;
}

```



```

case 5: { // Agregar Materia
    string nombreMateria, nombreProfesor, apellidoProfesor, IDMateria;
    int creditos;
    cout << "Ingrese el nombre de la materia: ";
    nombreMateria = leerString();
    cout << "Ingrese los créditos de la materia: ";
    cin >> creditos;
    cin.ignore();
    cout << "Ingrese el nombre del profesor que imparte la materia: ";
    nombreProfesor = leerString();
    cout << "Ingrese el apellido del profesor: ";
    apellidoProfesor = leerString();
    cout << "Ingrese el ID de la materia: ";
    IDMateria = leerString();
    cout << "Ingrese la cédula del profesor: ";
    string cedulaProfesor;
    cin >> cedulaProfesor;
    cin.ignore();
}

```

```

case 5: { // Agregar Materia
    string nombreMateria, nombreProfesor, apellidoProfesor, IDMateria;
    int creditos;
    cout << "Ingrese el nombre de la materia: ";
    nombreMateria = leerString();
    cout << "Ingrese los créditos de la materia: ";
    cin >> creditos;
    cin.ignore();
    cout << "Ingrese el nombre del profesor que imparte la materia: ";
    nombreProfesor = leerString();
    cout << "Ingrese el apellido del profesor: ";
    apellidoProfesor = leerString();
    cout << "Ingrese el ID de la materia: ";
    IDMateria = leerString();
    cout << "Ingrese la cédula del profesor: ";
    string cedulaProfesor;
    cin >> cedulaProfesor;
    cin.ignore();
}

```

2.2 Visualización de información (registros)

- Una vez capturada la información el sistema debe permitir visualizar la información capturada, para ello establece un menú:

Desea visualizar:

1. Profesor
2. Materia
3. Alumno

Después de la respuesta se debe solicitar el ID del registro y a partir de este ID se debe mostrar toda la información asociada al registro.



```
Ingrese el ID del alumno a mostrar: 123
Alumno: Emilio Jauregui
ID: 123, No. de Control: 1
Edad: 22
Sexo: m
Semestre: 5
Materias:

Sistema de Control Escolar
1. Registrar Alumno
2. Eliminar Alumno
3. Registrar Profesor
4. Eliminar Profesor
5. Registrar Materia
6. Eliminar Materia
7. Alumno
8. Profesor
9. Materia
10. Salir
Seleccione una opción (1..10): 8
Ingrese el ID del profesor a mostrar: 123
Profesor: Pedro Gomez, ID: 123, Cédula: 123, Título: Ingeniero

Sistema de Control Escolar
1. Registrar Alumno
2. Eliminar Alumno
3. Registrar Profesor
4. Eliminar Profesor
5. Registrar Materia
6. Eliminar Materia
7. Alumno
8. Profesor
9. Materia
10. Salir
Seleccione una opción (1..10): 9
Ingrese el ID de la materia a mostrar: 123
Materia: Matematicas, ID: 123, Créditos: 8, Profesor: Peedr Gomez
```

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

using namespace std;

// Clase Profesor
class Profesor {
public:
    string nombre;
    string apellido;
    char sexo;
    int edad;
    string titulo;
    string cedula;
    string ID;
```



```

// Constructor para inicializar los atributos
Profesor(string nombre, string apellido, char sexo, int edad, string titulo, string
cedula, string ID)
    : nombre(nombre), apellido(apellido), sexo(sexo), edad(edad), titulo(titulo),
cedula(cedula), ID(ID) {}
};

// Clase Materia
class Materia {
public:
    string nombre;
    int creditos;
    Profesor profesor;
    string ID;

// Constructor para inicializar los atributos
Materia(string nombre, int creditos, Profesor profesor, string ID)
    : nombre(nombre), creditos(creditos), profesor(profesor), ID(ID) {}
};

// Clase Alumno
class Alumno {
public:
    string nombre;
    string apellido;
    string noControl;
    int edad;
    char sexo;
    int semestre;
    vector<Materia> materias;
    string ID;

// Constructor para inicializar los atributos
Alumno(string nombre, string apellido, string noControl, int edad, char sexo, int
semestre, string ID)
    : nombre(nombre), apellido(apellido), noControl(noControl), edad(edad),
sexo(sexo), semestre(semestre), ID(ID) {}

// Método para agregar una materia
void agregarMateria(Materia materia) {
    materias.push_back(materia);
}

// Método para mostrar información del alumno
void mostrarInformacion() const {
    cout << "\nAlumno: " << nombre << " " << apellido << endl;
    cout << "ID: " << ID << ", No. de Control: " << noControl << endl;
}

```



```

cout << "Edad: " << edad << endl;
cout << "Sexo: " << sexo << endl;
cout << "Semestre: " << semestre << endl;
cout << "Materias: " << endl;
for (const auto& materia : materias) {
    cout << "- " << materia.nombre << "(" << materia.creditos << " créditos),
Profesor: "
    << materia.profesor.nombre << " " << materia.profesor.apellido << endl;
}
};

// Clase SistemaControlEscolar
class SistemaControlEscolar {
private:
    vector<Alumno> alumnos;
    vector<Profesor> profesores;
    vector<Materia> materias;

public:
    // Funciones para agregar, eliminar y mostrar
    void agregarAlumno(Alumno alumno) {
        alumnos.push_back(alumno);
    }

    void eliminarAlumno(string ID) {
        for (auto it = alumnos.begin(); it != alumnos.end(); ++it) {
            if (it->ID == ID) {
                alumnos.erase(it);
                cout << "Alumno con ID " << ID << " eliminado." << endl;
                return;
            }
        }
        cout << "Alumno no encontrado." << endl;
    }

    void agregarProfesor(Profesor profesor) {
        profesores.push_back(profesor);
    }

    void eliminarProfesor(string ID) {
        for (auto it = profesores.begin(); it != profesores.end(); ++it) {
            if (it->ID == ID) {
                profesores.erase(it);
                cout << "Profesor con ID " << ID << " eliminado." << endl;
                return;
            }
        }
    }
};

```



```

    }
    cout << "Profesor no encontrado." << endl;
}

void agregarMateria(Materia materia) {
    materias.push_back(materia);
}

void eliminarMateria(string ID) {
    for (auto it = materias.begin(); it != materias.end(); ++it) {
        if (it->ID == ID) {
            materias.erase(it);
            cout << "Materia con ID " << ID << " eliminada." << endl;
            return;
        }
    }
    cout << "Materia no encontrada." << endl;
}

// Mostrar información de un Alumno por ID
void mostrarAlumnoPorID(string ID) const {
    for (const auto& alumno : alumnos) {
        if (alumno.ID == ID) {
            alumno.mostrarInformacion();
            return;
        }
    }
    cout << "Alumno con ID " << ID << " no encontrado." << endl;
}

// Mostrar información de un Profesor por ID
void mostrarProfesorPorID(string ID) const {
    for (const auto& profesor : profesores) {
        if (profesor.ID == ID) {
            cout << "Profesor: " << profesor.nombre << " " << profesor.apellido << ", "
ID: "
            << profesor.ID << ", Cédula: " << profesor.cedula << ", Título: " <<
profesor.titulo << endl;
            return;
        }
    }
    cout << "Profesor con ID " << ID << " no encontrado." << endl;
}

// Mostrar información de una Materia por ID
void mostrarMateriaPorID(string ID) const {
    for (const auto& materia : materias) {

```



```

        if (materia.ID == ID) {
            cout << "Materia: " << materia.nombre << ", ID: " << materia.ID << ","
            Créditos: " << materia.creditos << ", Profesor: "
                << materia.profesor.nombre << " " << materia.profesor.apellido <<
            endl;
            return;
        }
    }
    cout << "Materia con ID " << ID << " no encontrada." << endl;
}
};

// Menú de opciones
void mostrarMenu() {
    cout << "\nSistema de Control Escolar" << endl;
    cout << "1. Registrar Alumno" << endl;
    cout << "2. Eliminar Alumno" << endl;
    cout << "3. Registrar Profesor" << endl;
    cout << "4. Eliminar Profesor" << endl;
    cout << "5. Registrar Materia" << endl;
    cout << "6. Eliminar Materia" << endl;
    cout << "7. Alumno" << endl;
    cout << "8. Profesor" << endl;
    cout << "9. Materia" << endl;
    cout << "10. Salir" << endl;
    cout << "Seleccione una opción (1..10): ";
}

// Función para leer una cadena de texto
string leerString() {
    string input;
    getline(cin, input);
    return input;
}

// Función principal que ejecuta el sistema
int main() {
    SistemaControlEscolar sistema;
    int opcion;

    do {
        mostrarMenu();
        cin >> opcion;
        cin.ignore(); // Limpiar el buffer de entrada

        switch (opcion) {
            case 1: { // Registrar Alumno

```



```

string nombre, apellido, noControl, ID;
int edad, semestre;
char sexo;
cout << "Ingrese el nombre del alumno: ";
nombre = leerString();
cout << "Ingrese el apellido del alumno: ";
apellido = leerString();
cout << "Ingrese el No. de Control: ";
noControl = leerString();
cout << "Ingrese la edad del alumno: ";
cin >> edad;
cout << "Ingrese el semestre: ";
cin >> semestre;
cout << "Ingrese el sexo (M/F): ";
cin >> sexo;
cin.ignore();
cout << "Ingrese el ID del alumno: ";
ID = leerString();

Alumno nuevoAlumno(nombre, apellido, noControl, edad, sexo,
semestre, ID);
sistema.agregarAlumno(nuevoAlumno);
break;
}

case 2: { // Eliminar Alumno
string ID;
cout << "Ingrese el ID del alumno a eliminar: ";
ID = leerString();
sistema.eliminarAlumno(ID);
break;
}

case 3: { // Registrar Profesor
string nombre, apellido, titulo, cedula, ID;
int edad;
char sexo;
cout << "Ingrese el nombre del profesor: ";
nombre = leerString();
cout << "Ingrese el apellido del profesor: ";
apellido = leerString();
cout << "Ingrese la edad del profesor: ";
cin >> edad;
cout << "Ingrese el título del profesor: ";
cin.ignore();
titulo = leerString();
cout << "Ingrese la cédula profesional del profesor: ";

```



```

cedula = leerString();
cout << "Ingrese el sexo (M/F): ";
cin >> sexo;
cin.ignore();
cout << "Ingrese el ID del profesor: ";
ID = leerString();

Profesor nuevoProfesor(nombre, apellido, sexo, edad, titulo, cedula, ID);
sistema.agregarProfesor(nuevoProfesor);
break;
}

case 4: { // Eliminar Profesor
string ID;
cout << "Ingrese el ID del profesor a eliminar: ";
ID = leerString();
sistema.eliminarProfesor(ID);
break;
}

case 5: { // Agregar Materia
string nombreMateria, nombreProfesor, apellidoProfesor, IDMateria;
int creditos;
cout << "Ingrese el nombre de la materia: ";
nombreMateria = leerString();
cout << "Ingrese los créditos de la materia: ";
cin >> creditos;
cin.ignore();
cout << "Ingrese el nombre del profesor que imparte la materia: ";
nombreProfesor = leerString();
cout << "Ingrese el apellido del profesor: ";
apellidoProfesor = leerString();
cout << "Ingrese el ID de la materia: ";
IDMateria = leerString();
cout << "Ingrese la cédula del profesor: ";
string cedulaProfesor;
cin >> cedulaProfesor;
cin.ignore();

// Crear profesor
Profesor profesor(nombreProfesor, apellidoProfesor, 'M', 40,
"Licenciado", cedulaProfesor, "PROF123");
Materia nuevaMateria(nombreMateria, creditos, profesor, IDMateria);
sistema.agregarMateria(nuevaMateria);
break;
}

```



```

        case 6: { // Eliminar Materia
            string IDMateria;
            cout << "Ingrese el ID de la materia a eliminar: ";
            IDMateria = leerString();
            sistema.eliminarMateria(IDMateria);
            break;
        }

        case 7: { // Mostrar Alumno por ID
            string ID;
            cout << "Ingrese el ID del alumno a mostrar: ";
            ID = leerString();
            sistema.mostrarAlumnoPorID(ID);
            break;
        }

        case 8: { // Mostrar Profesor por ID
            string ID;
            cout << "Ingrese el ID del profesor a mostrar: ";
            ID = leerString();
            sistema.mostrarProfesorPorID(ID);
            break;
        }

        case 9: { // Mostrar Materia por ID
            string ID;
            cout << "Ingrese el ID de la materia a mostrar: ";
            ID = leerString();
            sistema.mostrarMateriaPorID(ID);
            break;
        }

        case 10: { // Salir
            cout << "Saliendo del sistema..." << endl;
            break;
        }

        default: {
            cout << "Opción no válida. Intente nuevamente." << endl;
            break;
        }
    }

} while (opcion != 10);

return 0;
}

```



```

main.cpp 184 // Función principal que ejecuta el sistema
185 int main() {
186     SistemaControlEscolar sistema;
187     int opcion;
188
189     do {
190         mostrarMenu();
191         cin >> opcion;
192         cin.ignore(); // Limpiar el buffer de entrada
193
194         switch (opcion) {
195             case 1: { // Registrar Alumno
196                 string nombre, apellido, noControl, ID;
197                 int edad, semestre;
198                 char sexo;
199                 cout << "Ingrese el nombre del alumno: ";
200                 nombre = leerString();
201                 cout << "Ingrese el apellido del alumno: ";
202                 apellido = leerString();
203                 cout << "Ingrese el No. de Control: ";
204                 noControl = leerString();
205                 cout << "Ingrese la edad del alumno: ";
206                 cin >> edad;
207                 cout << "Ingrese el semestre: ";
208                 cin >> semestre;
209                 cout << "Ingrese el sexo (M/F): ";
210                 cin >> sexo;
211                 cin.ignore();
212                 cout << "Ingrese el ID del alumno: ";
213                 ID = leerString();
214
215                 Alumno nuevoAlumno(nombre, apellido, noControl, edad, sexo, s
216                 sistema.agregarAlumno(nuevoAlumno);
217
218             break;
219         }
220
221         case 2: { // Eliminar Alumno
222             string ID;
223             cout << "Ingrese el ID del alumno a eliminar: ";
224             ID = leerString();
225             sistema.eliminarAlumno(ID);
226             break;
227
228         case 3: { // Registrar Profesor
229             string nombre, apellido, titulo, cedula, ID;
230             int edad;
231             char sexo;
232             cout << "Ingrese el nombre del profesor: ";
233             nombre = leerString();
234             cout << "Ingrese el apellido del profesor: ";
235             apellido = leerString();
236             cout << "Ingrese la edad del profesor: ";
237             cin >> edad;
238             cout << "Ingrese el titulo del profesor: ";
239             cin.ignore();
240             titulo = leerString();
241             cout << "Ingrese la cédula profesional del profesor: ";
242             cedula = leerString();
243             cout << "Ingrese el sexo (M/F): ";
244             cin >> sexo;
245             cin.ignore();
246             cout << "Ingrese el ID del profesor: ";
247             ID = leerString();
248
249             Profesor nuevoProfesor(nombre, apellido, sexo, edad, titulo, c
250             sistema.agregarProfesor(nuevoProfesor);
251             break;
252
253         case 4: { // Eliminar Profesor
254             string ID;
255
256             cout << "Ingrese el ID de la materia a eliminar: ";
257             IDMateria = leerString();
258             sistema.eliminarMateria(IDMateria);
259             break;
260
261         case 5: { // Agregar Materia
262             string nombreMateria, nombreProfesor, apellidoProfesor, IDMateria;
263             int creditos;
264             cout << "Ingrese el nombre de la materia: ";
265             nombreMateria = leerString();
266             cout << "Ingrese los créditos de la materia: ";
267             cin >> creditos;
268             cin.ignore();
269             cout << "Ingrese el nombre del profesor que imparte la materia: ";
270             nombreProfesor = leerString();
271             cout << "Ingrese el apellido del profesor: ";
272             apellidoProfesor = leerString();
273             cout << "Ingrese el ID de la materia: ";
274             IDMateria = leerString();
275             cout << "Ingrese la cédula del profesor: ";
276             string cedulaProfesor;
277             cin >> cedulaProfesor;
278             cin.ignore();
279
280             // Crear profesor
281             Profesor profesor(nombreProfesor, apellidoProfesor, 'M', 40, "Licencia");
282             Materia nuevaMateria(nombreMateria, creditos, profesor, IDMateria);
283             sistema.agregarMateria(nuevaMateria);
284             break;
285
286         case 6: { // Eliminar Materia
287             string IDMateria;
288             cout << "Ingrese el ID de la materia a eliminar: ";
289             IDMateria = leerString();
290             sistema.eliminarMateria(IDMateria);
291             break;
292
293         case 7: { // Mostrar Alumno por ID
294             string ID;
295             cout << "Ingrese el ID del alumno a mostrar: ";
296             ID = leerString();
297             sistema.mostrarAlumnoPorID(ID);
298             break;
299
300         case 8: { // Mostrar Profesor por ID
301             string ID;
302             cout << "Ingrese el ID del profesor a mostrar: ";
303             ID = leerString();
304             sistema.mostrarProfesorPorID(ID);
305             break;
306
307         case 9: { // Mostrar Materia por ID
308             string ID;
309             cout << "Ingrese el ID de la materia a mostrar: ";
310             ID = leerString();
311             sistema.mostrarMateriaPorID(ID);
312             break;
313
314         case 10: { // Salir
315             cout << "Saliendo del sistema..." << endl;
316             break;
317
318         default: {
319             cout << "Opción no válida. Intente nuevamente." << endl;
320             break;
321
322         }
323
324     }
325
326 }

```

Conclusión

Al estar realizando esta segunda etapa me fui guiando con el modelado de sistemas de la etapa 1 que realice ya que mediante fui avanzando en este proyecto tenia la idea de ir desarrollando un programa de registro escolar en la cual podamos ingresar a un interfaz que nos permita manipular nuestros propios registros individualmente por ello realice métodos aprendidos de la materia de lógica y programación estructura para guiarne en el interfaz del menú y me tome el atrevimiento de utilizar lineas de código que nos permita almacenar información que le vayamos a ingresar ya que me gusto la idea de poder realizar una calculadora como usted nos enseño a realizar al principio de la materia y no tener que capturar los datos desde la función principal main ya que considero que es más eficiente realizar una tarea que nos permitirá el almacenamiento de datos dejando afuera la



modificación del código y así poder prever errores. Claro que considero que era aún más sencillo capturar únicamente los datos en la función principal ya que me tomo bastante tiempo investigar las líneas de código que me apoyaran a realizar el almacenamiento y la eliminación de la información pero con apoyo de igual manera del chat pude comprender con gran precisión las líneas y poderlas implementar en esta segunda etapa del proyecto integrador.

Referencias

S.A. (s.f.). Funciones en C Haga clic para ver más opciones [Archivo PDF]. Recuperado de <https://www.mheducation.es/bcv/guide/capitulo/8448148681.pdf>

- YouTube. (n.d.-f). Youtu.Be. Recuperado el 30 de noviembre del 2024, de <https://youtu.be/fxFJtP3gvZs?si=CHFXT9Y7UFt7EsW>

Márquez Frausto, T. G., Osorio Ángle, S., & Olvera Pérez, E. N. (2011). Introducción a la programación estructurada en C. Recuperado el 30 de noviembre del 2024 de, <https://pdfcoffee.com/introduccion-a-la-programacion-11-pdf-free.html>

- YouTube. (n.d.-g). Youtu.Be. Recuperado el 30 de noviembre del 2024, de <https://youtu.be/8Klq2w9tw64?si=fSkXxpQ0R4QqPSFh>

ChatGPT. (n.d.). Chatgpt.com. Recuperado el 30 de noviembre del 2024, de <https://chatgpt.com>

Clases y Objetos en C++ (Práctica 1) Haga clic para ver más opciones (s.f). Recuperado el 30 de noviembre del 2024, de <https://www.codingame.com/playgrounds/50557/clases-y-objetos-en-c-practica-1/miembros-de-clase-en-c-variables-y-metodos>

Ceballos, J. (2007). Programación orientada a objetos con C++ Haga clic para ver más opciones. Recuperado el 30 de noviembre del 2024, de https://kupdf.net/download/ceballos-programaciacute-n-orientada-a-objetos-con-c-4ed_58a8b3476454a7596fb1e8d9.pdf





Universidad
del Valle de México
LAUREATE INTERNATIONAL UNIVERSITIES®

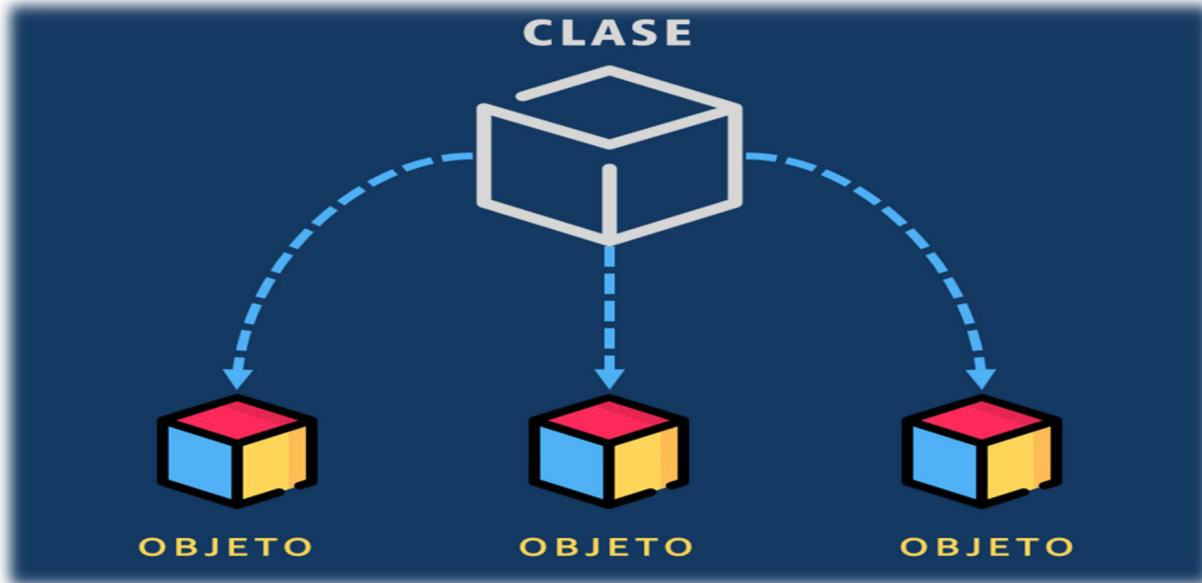
Actividad 9

Proyecto integrador etapa 3 Programación orientada a objetos

Docente. Ing. Saúl Santiago Rivera

José Emiliano Jauregui Guzmán

Por siempre responsable de lo que se ha cultivado



Integración de herencia y colecciones

Introducción

Esta actividad consiste en aplicar los conocimientos adquiridos a lo largo del curso y retomar lo aprendido en cada una de las actividades realizadas, lo que garantiza la transversalidad de los contenidos revisados para fortalecer el desarrollo de competencias y lograr el fin de formación planteado.

Objetivo

El objetivo del Proyecto integrador es desarrollar el prototipo de un sistema de control escolar básico haciendo uso de las diferentes características de la programación orientada a objetos con el fin de evidenciar las ventajas del paradigma.

2. En la etapa 2 del proyecto se solicitó la implementación de métodos para adicionar y visualizar registros a cada una de tus clases, en esta última etapa deberás realizar lo siguiente:

III. Integración de herencia y colecciones

3.1 Modificación de clases

Retoma las *clases* programadas y modificalas para integrar la *herencia* en el diseño de tu sistema. Se recomienda volver a revisar el siguiente material proporcionado con anterioridad:

Para poder hacer una modificación para integrar la herencia tendremos que crear una clase persona para darle atributos que las clases hijas hereden.

3.2 Herencia de clases

- Define una clase base “Persona” con los atributos generales de una persona
- De esta clase base hereda la clase alumno y profesor, en las clases hijas o heredadas coloca los atributos específicos de cada una.

Para ello cree una clase persona con los atributos como: nombre, apellido, sexo, edad e ID para que las clases profesor y alumno las hereden. De igual manera se utilizo el método mostrarInformacion() que sobrescribe en las dos clases profesor y alumno para agregar detalles específicos como el título o materias.

```
// Clase Persona
class Persona {
public:
    string nombre;
    string apellido;
```



```

char sexo;
int edad;
string ID;

// Constructor
Persona(string nombre, string apellido, char sexo, int edad, string ID)
    : nombre(nombre), apellido(apellido), sexo(sexo), edad(edad), ID(ID) {}

// Método para mostrar información
virtual void mostrarInformacion() const {
    cout << "Nombre: " << nombre << " " << apellido << ", Edad: " << edad << ",
Sexo: " << sexo << endl;
}
};


```

```

// Clase Persona
class Persona {
public:
    string nombre;
    string apellido;
    char sexo;
    int edad;
    string ID;

    // Constructor
    Persona(string nombre, string apellido, char sexo, int edad, string ID)
        : nombre(nombre), apellido(apellido), sexo(sexo), edad(edad), ID(ID) {}

    // Método para mostrar información
    virtual void mostrarInformacion() const {
        cout << "Nombre: " << nombre << " " << apellido << ", Edad: " << edad << ", Sexo: " << sexo << endl;
    }
};


```

```

// Clase Profesor
class Profesor : public Persona {
public:
    string titulo;
    string cedula;

    // Constructor
    Profesor(string nombre, string apellido, char sexo, int edad, string titulo, string
cedula, string ID)
        : Persona(nombre, apellido, sexo, edad, ID), titulo(titulo), cedula(cedula) {}

    void mostrarInformacion() const override {
        Persona::mostrarInformacion();
        cout << "Título: " << titulo << ", Cédula: " << cedula << endl;
    }
};


```



```
// Clase Profesor
class Profesor : public Persona {
public:
    string titulo;
    string cedula;

    // Constructor
    Profesor(string nombre, string apellido, char sexo, int edad, string titulo, string cedula, string ID)
        : Persona(nombre, apellido, sexo, edad, ID), titulo(titulo), cedula(cedula) {}

    void mostrarInformacion() const override {
        Persona::mostrarInformacion();
        cout << "Título: " << titulo << ", Cédula: " << cedula << endl;
    }
};
```

```
// Clase Alumno
class Alumno : public Persona {
public:
    string noControl;
    int semestre;

    // Constructor
    Alumno(string nombre, string apellido, string noControl, int edad, char sexo, int semestre, string ID)
        : Persona(nombre, apellido, sexo, edad, ID), noControl(noControl),
          semestre(semestre) {}

    void mostrarInformacion() const override {
        Persona::mostrarInformacion();
        cout << "No. de Control: " << noControl << ", Semestre: " << semestre <<
        endl;
    }
};
```

```
// Clase Alumno
class Alumno : public Persona {
public:
    string noControl;
    int semestre;

    // Constructor
    Alumno(string nombre, string apellido, string noControl, int edad, char sexo, int semestre, string ID)
        : Persona(nombre, apellido, sexo, edad, ID), noControl(noControl),
          semestre(semestre) {}

    void mostrarInformacion() const override {
        Persona::mostrarInformacion();
        cout << "No. de Control: " << noControl << ", Semestre: " << semestre << endl;
    }
};
```



3.3. Utilización de colecciones

- Modifica tu función principal `main()` para utilizar colecciones, define colecciones de tipo `List` para tus objetos (`List<T>`)
- Solicita la captura de 3 profesores, 4 materias y 3 alumnos para cada alumno asigna dos materias de las capturadas, almacena la información en las listas definidas
- Una vez capturada la información el sistema debe permitir visualizar la información capturada, para ello establece un menú:

<pre> Sistema de Control Escolar 1. Registrar Alumno 2. Eliminar Alumno 3. Registrar Profesor 4. Eliminar Profesor 5. Registrar Materia 6. Eliminar Materia 7. Alumno 8. Profesor 9. Materia 10. Salir Seleccione una opción (1..10): 7 Ingrese el ID del alumno a mostrar: 987 Nombre: Valeria Rico, Edad: 22, Sexo: F No. de Control: 2, Semestre: 4 Sistema de Control Escolar 1. Registrar Alumno 2. Eliminar Alumno 3. Registrar Profesor 4. Eliminar Profesor 5. Registrar Materia 6. Eliminar Materia 7. Alumno 8. Profesor 9. Materia 10. Salir Seleccione una opción (1..10): 8 Ingrese el ID del profesor a mostrar: 234 Nombre: Pablo Hernandez , Edad: 44, Sexo: M Título: Ingeniero, Cédula: 789 Sistema de Control Escolar 1. Registrar Alumno 2. Eliminar Alumno 3. Registrar Profesor 4. Eliminar Profesor 5. Registrar Materia 6. Eliminar Materia 7. Alumno 8. Profesor 9. Materia 10. Salir Seleccione una opción (1..10): 9 Ingrese el ID de la materia a mostrar: 2345 Materia: Calculo, ID: 2345, Créditos: 30, Profesor: Pablo Hernandez </pre>	<pre> Sistema de Control Escolar 1. Registrar Alumno 2. Eliminar Alumno 3. Registrar Profesor 4. Eliminar Profesor 5. Registrar Materia 6. Eliminar Materia 7. Alumno 8. Profesor 9. Materia 10. Salir Seleccione una opción (1..10): 7 Ingrese el ID del alumno a mostrar: 098 Nombre: Jose Jauregui, Edad: 22, Sexo: M No. de Control: 1, Semestre: 4 Sistema de Control Escolar 1. Registrar Alumno 2. Eliminar Alumno 3. Registrar Profesor 4. Eliminar Profesor 5. Registrar Materia 6. Eliminar Materia 7. Alumno 8. Profesor 9. Materia 10. Salir Seleccione una opción (1..10): 8 Ingrese el ID del profesor a mostrar: 123 Nombre: Francisco Perez, Edad: 33, Sexo: M Título: Ingeniero, Cédula: 890 Sistema de Control Escolar 1. Registrar Alumno 2. Eliminar Alumno 3. Registrar Profesor 4. Eliminar Profesor 5. Registrar Materia 6. Eliminar Materia 7. Alumno 8. Profesor 9. Materia 10. Salir Seleccione una opción (1..10): 9 Ingrese el ID de la materia a mostrar: 1234 Materia: Algebra, ID: 1234, Créditos: 28, Profesor: Francisco Perez </pre>
---	---



```
Sistema de Control Escolar
1. Registrar Alumno
2. Eliminar Alumno
3. Registrar Profesor
4. Eliminar Profesor
5. Registrar Materia
6. Eliminar Materia
7. Alumno
8. Profesor
9. Materia
10. Salir
Seleccione una opción (1..10): 7

Ingrese el ID del alumno a mostrar: 876
Nombre: Emiliano Guzman, Edad: 22, Sexo: M
No. de Control: 3, Semestre: 4

Sistema de Control Escolar
1. Registrar Alumno
2. Eliminar Alumno
3. Registrar Profesor
4. Eliminar Profesor
5. Registrar Materia
6. Eliminar Materia
7. Alumno
8. Profesor
9. Materia
10. Salir
Seleccione una opción (1..10): 8

Ingrese el ID del profesor a mostrar: 345
Nombre: Marco Godin, Edad: 39, Sexo: M
Título: Doctorado, Cédula: 678

Sistema de Control Escolar
1. Registrar Alumno
2. Eliminar Alumno
3. Registrar Profesor
4. Eliminar Profesor
5. Registrar Materia
6. Eliminar Materia
7. Alumno
8. Profesor
9. Materia
10. Salir
Seleccione una opción (1..10): 9

Ingrese el ID de la materia a mostrar: 3456
Materia: Ecuaciones Diferenciales, ID: 3456, Créditos: 38, Profesor: Marco Godin
```

```
#include <iostream>
#include <list>
#include <string>

using namespace std;

// Clase Persona
class Persona {
public:
    string nombre;
    string apellido;
    char sexo;
    int edad;
    string ID;

    // Constructor
    Persona(string nombre, string apellido, char sexo, int edad, string ID)
        : nombre(nombre), apellido(apellido), sexo(sexo), edad(edad), ID(ID) {}

    // Método para mostrar información
    virtual void mostrarInformacion() const {
        cout << "Nombre: " << nombre << " " << apellido << ", Edad: " << edad << ",
        Sexo: " << sexo << endl;
    }
}
```



```

};

// Clase Profesor
class Profesor : public Persona {
public:
    string titulo;
    string cedula;

    // Constructor
    Profesor(string nombre, string apellido, char sexo, int edad, string titulo, string
cedula, string ID)
        : Persona(nombre, apellido, sexo, edad, ID), titulo(titulo), cedula(cedula) {}

    void mostrarInformacion() const override {
        Persona::mostrarInformacion();
        cout << "Título: " << titulo << ", Cédula: " << cedula << endl;
    }
};

// Clase Alumno
class Alumno : public Persona {
public:
    string noControl;
    int semestre;

    // Constructor
    Alumno(string nombre, string apellido, string noControl, int edad, char sexo, int
semestre, string ID)
        : Persona(nombre, apellido, sexo, edad, ID), noControl(noControl),
semestre(semestre) {}

    void mostrarInformacion() const override {
        Persona::mostrarInformacion();
        cout << "No. de Control: " << noControl << ", Semestre: " << semestre <<
endl;
    }
};

// Clase Materia
class Materia {
public:
    string nombre;
    int creditos;
    Profesor profesor;
    string ID;

    // Constructor

```



```

Materia(string nombre, int creditos, Profesor profesor, string ID)
    : nombre(nombre), creditos(creditos), profesor(profesor), ID(ID) {}

};

// Clase Sistema Control Escolar
class SistemaControlEscolar {
private:
    list<Alumno> alumnos;
    list<Profesor> profesores;
    list<Materia> materias;

public:
    // Método para agregar
    void agregarAlumno(const Alumno& alumno) {
        alumnos.push_back(alumno);
    }

    void agregarProfesor(const Profesor& profesor) {
        profesores.push_back(profesor);
    }

    void agregarMateria(const Materia& materia) {
        materias.push_back(materia);
    }

    // Método para mostrar
    void mostrarAlumnoPorID(const string& ID) const {
        for (const auto& alumno : alumnos) {
            if (alumno.ID == ID) {
                alumno.mostrarInformacion();
                return;
            }
        }
        cout << "Alumno con ID " << ID << " no encontrado." << endl;
    }

    void mostrarProfesorPorID(const string& ID) const {
        for (const auto& profesor : profesores) {
            if (profesor.ID == ID) {
                profesor.mostrarInformacion();
                return;
            }
        }
        cout << "Profesor con ID " << ID << " no encontrado." << endl;
    }

    void mostrarMateriaPorID(const string& ID) const {

```



```

        for (const auto& materia : materias) {
            if (materia.ID == ID) {
                cout << "Materia: " << materia.nombre << ", ID: " << materia.ID << ","
                Créditos: " << materia.creditos << ", Profesor: "
                    << materia.profesor.nombre << " " << materia.profesor.apellido <<
                endl;
                return;
            }
        }
        cout << "Materia con ID " << ID << " no encontrada." << endl;
    }

    // Método para eliminar
    void eliminarAlumno(const string& ID) {
        for (auto it = alumnos.begin(); it != alumnos.end(); ++it) {
            if (it->ID == ID) {
                alumnos.erase(it);
                cout << "Alumno con ID " << ID << " eliminado." << endl;
                return;
            }
        }
        cout << "Alumno no encontrado." << endl;
    }

    void eliminarProfesor(const string& ID) {
        for (auto it = profesores.begin(); it != profesores.end(); ++it) {
            if (it->ID == ID) {
                profesores.erase(it);
                cout << "Profesor con ID " << ID << " eliminado." << endl;
                return;
            }
        }
        cout << "Profesor no encontrado." << endl;
    }

    void eliminarMateria(const string& ID) {
        for (auto it = materias.begin(); it != materias.end(); ++it) {
            if (it->ID == ID) {
                materias.erase(it);
                cout << "Materia con ID " << ID << " eliminada." << endl;
                return;
            }
        }
        cout << "Materia no encontrada." << endl;
    }

    list<Profesor>& getProfesores() {

```



```

        return profesores;
    }
};

// Función para mostrar el menú
void mostrarMenu() {
    cout << "\nSistema de Control Escolar" << endl;
    cout << "1. Registrar Alumno" << endl;
    cout << "2. Eliminar Alumno" << endl;
    cout << "3. Registrar Profesor" << endl;
    cout << "4. Eliminar Profesor" << endl;
    cout << "5. Registrar Materia" << endl;
    cout << "6. Eliminar Materia" << endl;
    cout << "7. Alumno" << endl;
    cout << "8. Profesor" << endl;
    cout << "9. Materia" << endl;
    cout << "10. Salir" << endl;
    cout << "Seleccione una opción (1..10): ";
}

// Función principal (main)
int main() {
    SistemaControlEscolar sistema;
    int opcion;

    do {
        mostrarMenu();
        cin >> opcion;
        cin.ignore(); // Limpiar el buffer de entrada

        switch (opcion) {
            case 1: { // Registrar Alumno
                string nombre, apellido, noControl, ID;
                int edad, semestre;
                char sexo;
                cout << "\nIngrese el nombre del alumno: ";
                getline(cin, nombre);
                cout << "Ingrese el apellido del alumno: ";
                getline(cin, apellido);
                cout << "Ingrese el No. de Control: ";
                getline(cin, noControl);
                cout << "Ingrese la edad del alumno: ";
                cin >> edad;
                cout << "Ingrese el semestre: ";
                cin >> semestre;
                cout << "Ingrese el sexo (M/F): ";
                cin >> sexo;
            }
        }
    }
}

```



```

    cin.ignore();
    cout << "Ingrese el ID del alumno: ";
    getline(cin, ID);

    Alumno nuevoAlumno(nombre, apellido, noControl, edad, sexo,
semestre, ID);
    sistema.agregarAlumno(nuevoAlumno);
    break;
}

case 2: { // Eliminar Alumno
    string ID;
    cout << "\nIngrese el ID del alumno a eliminar: ";
    getline(cin, ID);
    sistema.eliminarAlumno(ID);
    break;
}

case 3: { // Registrar Profesor
    string nombre, apellido, titulo, cedula, ID;
    int edad;
    char sexo;
    cout << "\nIngrese el nombre del profesor: ";
    getline(cin, nombre);
    cout << "Ingrese el apellido del profesor: ";
    getline(cin, apellido);
    cout << "Ingrese la edad del profesor: ";
    cin >> edad;
    cout << "Ingrese el título del profesor: ";
    cin.ignore();
    getline(cin, titulo);
    cout << "Ingrese la cédula profesional del profesor: ";
    getline(cin, cedula);
    cout << "Ingrese el sexo (M/F): ";
    cin >> sexo;
    cin.ignore();
    cout << "Ingrese el ID del profesor: ";
    getline(cin, ID);

    Profesor nuevoProfesor(nombre, apellido, sexo, edad, titulo, cedula, ID);
    sistema.agregarProfesor(nuevoProfesor);
    break;
}

case 4: { // Eliminar Profesor
    string ID;
    cout << "\nIngrese el ID del profesor a eliminar: ";
}

```



```

getline(cin, ID);
sistema.eliminarProfesor(ID);
break;
}

case 5: { // Registrar Materia
    string nombre, IDMateria, idProfesor;
    int creditos;
    cout << "\nIngrese el nombre de la materia: ";
    getline(cin, nombre);
    cout << "Ingrese los créditos de la materia: ";
    cin >> creditos;
    cin.ignore();
    cout << "Ingrese el ID del profesor: ";
    getline(cin, idProfesor);

    // Buscar al profesor por ID
    Profesor* profesor = nullptr;
    for (auto& prof : sistema.getProfesores()) {
        if (prof.ID == idProfesor) {
            profesor = &prof;
            break;
        }
    }

    if (profesor) {
        cout << "\nIngrese el ID de la materia: ";
        getline(cin, IDMateria);
        Materia nuevaMateria(nombre, creditos, *profesor, IDMateria);
        sistema.agregarMateria(nuevaMateria);
    } else {
        cout << "Profesor no encontrado." << endl;
    }
    break;
}

case 6: { // Eliminar Materia
    string ID;
    cout << "\nIngrese el ID de la materia a eliminar: ";
    getline(cin, ID);
    sistema.eliminarMateria(ID);
    break;
}

case 7: { // Mostrar Alumno por ID
    string ID;
    cout << "\nIngrese el ID del alumno a mostrar: ";
}

```



```

        getline(cin, ID);
        sistema.mostrarAlumnoPorID(ID);
        break;
    }

    case 8: { // Mostrar Profesor por ID
        string ID;
        cout << "\nIngrese el ID del profesor a mostrar: ";
        getline(cin, ID);
        sistema.mostrarProfesorPorID(ID);
        break;
    }

    case 9: { // Mostrar Materia por ID
        string ID;
        cout << "\nIngrese el ID de la materia a mostrar: ";
        getline(cin, ID);
        sistema.mostrarMateriaPorID(ID);
        break;
    }

    case 10: { // Salir
        cout << "Saliendo del sistema..." << endl;
        break;
    }

    default:
        cout << "Opción no válida. Intente nuevamente." << endl;
    }
}

} while (opcion != 10);

return 0;
}

```





```

217     cout << "\nIngrese el ID del alumno a eliminar: ";
218     getline(cin, ID);
219     sistema.eliminarAlumno(ID);
220     break;
221   }
222
223   case 3: { // Registrar Profesor
224     string nombre, apellido, titulo, cedula, ID;
225     int edad;
226     char sexo;
227     cout << "\nIngrese el nombre del profesor: ";
228     getline(cin, nombre);
229     cout << "Ingrese el apellido del profesor: ";
230     getline(cin, apellido);
231     cout << "Ingrese la edad del profesor: ";
232     cin >> edad;
233     cout << "Ingrese el título del profesor: ";
234     cin.ignore();
235     getline(cin, titulo);
236     cout << "Ingrese la cédula profesional del profesor: ";
237     getline(cin, cedula);
238     cout << "Ingrese el sexo (M/F): ";
239     cin >> sexo;
240     cin.ignore();
241     cout << "Ingrese el ID del profesor: ";
242     getline(cin, ID);
243
244     Profesor nuevoProfesor(nombre, apellido, sexo, edad, titulo, cedula, ID);
245     sistema.agregarProfesor(nuevoProfesor);
246     break;
247   }
248
249   case 4: { // Eliminar Profesor
250     string ID;
251     cout << "\nIngrese el ID del profesor a eliminar: ";
252     getline(cin, ID);
253
254     string ID;
255     cout << "\nIngrese el ID de la materia a eliminar: ";
256     getline(cin, ID);
257     sistema.eliminarMateria(ID);
258     break;
259   }
260
261   case 5: { // Registrar Materia
262     string nombre, IDMateria, idProfesor;
263     int creditos;
264     cout << "\nIngrese el nombre de la materia: ";
265     getline(cin, nombre);
266     cout << "Ingrese los créditos de la materia: ";
267     cin >> creditos;
268     cin.ignore();
269     cout << "Ingrese el ID del profesor: ";
270     getline(cin, idProfesor);
271
272     // Buscar al profesor por ID
273     Profesor* profesor = nullptr;
274     for (auto prof : sistema.getProfesores()) {
275       if (prof.ID == idProfesor) {
276         profesor = &prof;
277         break;
278       }
279
280       if (profesor) {
281         cout << "\nIngrese el ID de la materia: ";
282         getline(cin, IDMateria);
283         Materia nuevaMateria(nombre, creditos, *profesor, IDMateria);
284         sistema.agregarMateria(nuevaMateria);
285       } else {
286         cout << "Profesor no encontrado." << endl;
287       }
288     }
289
290     case 6: { // Eliminar Materia
291       cout << "\nIngrese el ID del alumno a mostrar: ";
292       getline(cin, ID);
293       sistema.mostrarAlumnoPorID(ID);
294       break;
295     }
296
297     case 7: { // Mostrar Alumno por ID
298       string ID;
299       cout << "\nIngrese el ID del alumno a mostrar: ";
300       getline(cin, ID);
301       sistema.mostrarAlumnoPorID(ID);
302       break;
303     }
304
305     case 8: { // Mostrar Profesor por ID
306       string ID;
307       cout << "\nIngrese el ID del profesor a mostrar: ";
308       getline(cin, ID);
309       sistema.mostrarProfesorPorID(ID);
310       break;
311     }
312
313     case 9: { // Mostrar Materia por ID
314       string ID;
315       cout << "\nIngrese el ID de la materia a mostrar: ";
316       getline(cin, ID);
317       sistema.mostrarMateriaPorID(ID);
318       break;
319     }
320
321     case 10: { // Salir
322       cout << "Saliendo del sistema..." << endl;
323       break;
324     }
325
326     default:
327       cout << "Opción no válida. Intente nuevamente." << endl;
328     }
329   }
330
331   return 0;
332 }
333

```

Conclusión

Para esta ultima etapa del proyecto integrador me enfrete a varios problemas en la modificación de mi programa ya que esta parte 3 me solicito puntos a integrar que me resultaron complicados de agregar en el mío ya que como lo había mencionado en las conclusiones de la etapa 2 yo realice este programa basándome en un interfaz en el cual me sería útil y eficiente para poder agregar, eliminar y modificar información desde el menú con apoyo de la materia lógica y programación estructurada que me dio las enseñanzas de poder concluir con este interfaz de un menú de sistema de control escolar funcional para su uso a largo plazo para no solo agregar a la cantidad solicitadas y corriendo el programa solo me arrojara esos resultados, si no poder almacenar una cantidad mas grande. Por ello realice los cambios solicitados ya que mi programa funcionaba con vectores y el cambio fue a la biblioteca `<list>`; el uso de las herencias considero que es sumamente útil para poder agilizar el código y hacerlo ver más limpio sin tanta linea de código pero creo que es aun mas útil cuando implementamos mas clases hijas para que el esfuerzo de realizar este paso sea mas eficiente. Gracias a este proyecto integrador pude



tener un iniciativa propia en la elaboración de un interfaz que me gustaría seguir desarrollando para en algún punto tener un uso personal que me ayude almacenar datos y ejercitarse los conocimientos obtenidos en esta materia.

Referencia

Oviedo, E. (2015). Lógica de programación orientada a objetos Haga clic para ver más opciones [Archivo PDF]. Recuperado el 14 de diciembre del 2024, de <https://www.studocu.com/co/document/pontificia-universidad-javeriana/taller-de-programacion/logica-de-programacion-orientada-a-objetos-cap-7-pg-219-287/17217649?origin=home-recent-3>

Microsoft (s.f.) Colecciones (C ++ / CX) [Sitio web]. Recuperado el 14 de diciembre del 2024, de <https://docs.microsoft.com/en-us/cpp/cppcx/collections-c-cx?view=vs-2019>

ChatGPT. (n.d.). Chatgpt.com. Recuperado el 14 de diciembre del 2024, de <https://chatgpt.com>

Microsoft (s.f.) List T Clase [Sitio web]. Recuperado el 14 de diciembre del 2024, de <https://docs.microsoft.com/es-es/dotnet/api/system.collections.generic.list-1?view=netframework-4.8>

- YouTube. (n.d.). Youtu.Be. Recuperado el 14 de diciembre del 2024, de <https://youtu.be/0kQKH0Y1i-Y?si=4O28xLaWNQFQhCCa>

