



### **Actividad 21**

Docente: Jimmy Nataniel Requena Llorentty

Materia: Programación III

Carrera: Ingeniería En Sistemas

Estudiantes: Joaquin Marcos Maita Flores

Santa Cruz – Bolivia

2025

### Encapsulando al Estudiante

```
#include <iostream>    // Librería para entrada/salida estándar (cout,
endl, etc.)

#include <string>       // Librería para poder usar std::string

// Definición de la clase Estudiante

class Estudiante {

private: // Sección de atributos privados, no accesibles directamente
desde fuera de la clase

    std::string nombre;    // Nombre del estudiante

    int edad;              // Edad del estudiante

    std::string matricula; // Matrícula del estudiante (código único)

    double promedio;      // Promedio del estudiante (en una escala de
0.0 a 10.0)

public:

    // Constructor: se ejecuta automáticamente al crear un objeto
Estudiante

    Estudiante(std::string nom, int ed, std::string matr) {

        nombre = nom;

        setEdad(ed);      // Usamos el setter para aplicar validación
al asignar edad

        matricula = matr;

        promedio = 0.0;    // Inicializamos el promedio en 0.0 por
defecto

        std::cout << "Estudiante '" << nombre << "' creado." << std::endl;
```

```
    }

    // Métodos Getter: permiten obtener (leer) los valores de los
    atributos

    std::string getNombre() const { return nombre; }           // Devuelve
    el nombre

    int getEdad() const { return edad; }                       // Devuelve
    la edad

    std::string getMatricula() const { return matricula; }     // Devuelve
    la matrícula

    double getPromedio() const { return promedio; }           // Devuelve
    el promedio

    // Métodos Setter: permiten modificar (escribir) los valores de los
    atributos

    void setNombre(const std::string& nuevoNombre) {

        if (!nuevoNombre.empty()) {                           // Verificamos que el nombre no
        esté vacío

            nombre = nuevoNombre;

        } else {

            std::cout << "Error: El nombre no puede estar vacío." <<
            std::endl;

        }

    }

    void setEdad(int nuevaEdad) {

        if (nuevaEdad >= 5 && nuevaEdad <= 100) { // Validamos un rango
```

```
    aceptable para edad

        edad = nuevaEdad;

    } else {

        std::cout << "Error: Edad '" << nuevaEdad << "' invalida para
el estudiante " << nombre << ". Edad no modificada." << std::endl;

    }

}

// No hay setter para matrícula porque se asume que no debe cambiar
una vez asignada.

// Si fuera necesario cambiarla en el futuro, podrías añadirlo con
validación adecuada.

void setPromedio(double nuevoPromedio) {

    if (nuevoPromedio >= 0.0 && nuevoPromedio <= 10.0) { // Validamos
rango del promedio

        promedio = nuevoPromedio;

    } else {

        std::cout << "Error: Promedio '" << nuevoPromedio << "'
invalido para " << nombre << ". Promedio no modificado." << std::endl;

    }

}

// Método para mostrar toda la información del estudiante de manera
organizada

void mostrarInformacion() const {
```

```
        std::cout << "-----" << std::endl;

        std::cout << "Nombre: " << nombre << std::endl;

        std::cout << "Edad: " << edad << " años" << std::endl;

        std::cout << "Matricula: " << matricula << std::endl;

        std::cout << "Promedio: " << promedio << std::endl;

        std::cout << "-----" << std::endl;

    }

}; // Fin de la clase Estudiante

// Función principal del programa

int main() {

    // Creamos un estudiante con nombre, edad y matrícula

    Estudiante estudiante1("Juaquin Soliz", 20, "A123");

    // Mostramos su información

    estudiante1.mostrarInformacion();

    // Intentamos modificar la edad y promedio del estudiante

    std::cout << "\nIntentando actualizar edad y promedio..." <<
std::endl;

    estudiante1.setEdad(21);          // Actualización válida

    estudiante1.setPromedio(8.5);     // Actualización válida

    estudiante1.setEdad(150);         // Edad inválida: fuera del rango
permitido
```

```
    estudiante1.setPromedio(-2.0); // Promedio inválido: fuera del rango
    permitido

    // Mostramos la información actualizada

    std::cout << "\nInformacion actualizada de " <<
estudiante1.getNombre() << ":" << std::endl;

    estudiante1.mostrarInformacion();

    // Creamos otro estudiante con edad inválida para probar la validación
    desde el constructor

    Estudiante estudiante2("Priscila Vaca", -10, "B456");

    // Mostramos la información del segundo estudiante

    estudiante2.mostrarInformacion(); // Mostrará la edad original si fue
rechazada la inválida

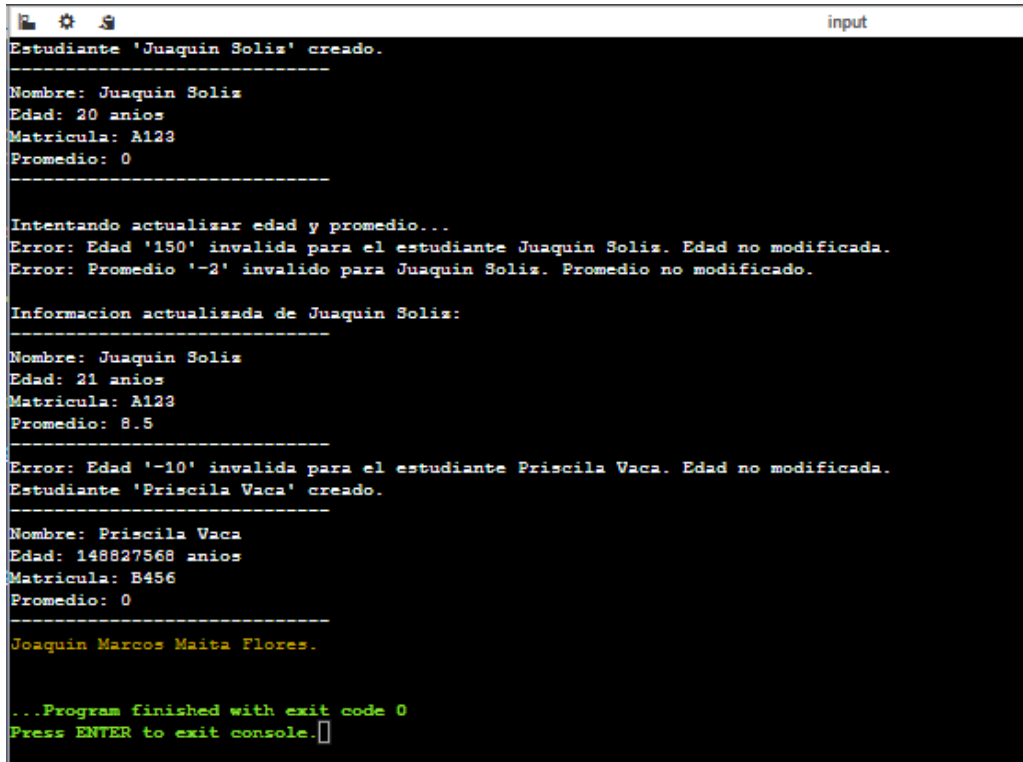
    std::cout << "\033[33mJoaquin Marcos Maita Flores.\033[0m" <<
std::endl;

    return 0;
}
```

Título: Actividad 21

Estudiante/s: Joaquín Marcos Maita Flores

## Corriendo Código



```
input
Estudiante 'Joaquin Soliz' creado.
-----
Nombre: Joaquin Soliz
Edad: 20 años
Matricula: A123
Promedio: 0
-----

Intentando actualizar edad y promedio...
Error: Edad '150' invalida para el estudiante Joaquin Soliz. Edad no modificada.
Error: Promedio '-2' invalido para Joaquin Soliz. Promedio no modificado.

Informacion actualizada de Joaquin Soliz:
-----
Nombre: Joaquin Soliz
Edad: 21 años
Matricula: A123
Promedio: 8.5
-----

Error: Edad '-10' invalida para el estudiante Priscila Vaca. Edad no modificada.
Estudiante 'Priscila Vaca' creado.
-----
Nombre: Priscila Vaca
Edad: 148827568 años
Matricula: B456
Promedio: 0
-----

Joaquín Marcos Maita Flores.

...Program finished with exit code 0
Press ENTER to exit console.
```

## ¿Qué aprendimos?

- Cómo encapsular datos usando `private` y `public`.
- Uso de setters con validación para proteger los atributos.
- Cómo evitar errores al modificar datos sensibles.
- Buenas prácticas como inicialización en el constructor y separación clara de responsabilidades.

anexo

<https://onlinegdb.com/8cONJdqcl>

Fecha y hora actual: 2025-06-17 21:01:06

Carrera: Ingeniería En Sistemas

Materia: Programación III