

TAREAS DE LA MATERIA DE PROGRAMACIÓN 3



Mateo Perez Nomey

Programación 3

Programacion orientada a objetos

Automovil

```
== Proyecto Automovil - Código creado por Mateo Perez Nomey ==  
  
--- Creando un Automovil en el Stack ---  
CONSTRUCTOR Motor: Creado motor de 200 HP.  
CONSTRUCTOR Automovil: Ensamblado un SuperMarca ModeloX con un motor.  
Diagnostico del ModeloX:  
Estado del Motor: Apagado, HP: 200  
ModeloX: Intentando encender...  
Motor: ¡BRUM! Encendido.  
Diagnostico del ModeloX:  
Estado del Motor: Encendido, HP: 200  
ModeloX: Intentando apagar...  
Motor: ...silencio. Apagado.  
  
--- Saliendo de main (miAuto se destrui) ---  
  
¿Deseas ver el autor del proyecto? (s/n): s  
  
Este proyecto fue realizado por: Mateo Perez Nomey  
¡Gracias por revisar este código! 🧠  
DESTRUCTOR Automovil: Desguazando el SuperMarca ModeloX.  
DESTRUCTOR Motor: Destruido motor de 200 HP.  
  
...Program finished with exit code 0  
Press ENTER to exit console.[]
```

```
#include <iostream>
```

```
#include <string>
```

```
// =====  
// PROYECTO DE EJEMPLO - by Mateo Perez Nomey 🧠  
// =====
```

```
// Clase 'Parte': Motor
```

```
class Motor {
```

```
private:
```

```
    int caballosDeFuerza;
```

```
    bool encendido;
```

```
public:
```

```
    Motor(int hp = 150) : caballosDeFuerza(hp), encendido(false) {
```

```
        std::cout << " CONSTRUCTOR Motor: Creado motor de " << caballosDeFuerza << " HP." << std::endl;
```

```
    }
```

```
    ~Motor() {
```

```
        std::cout << " DESTRUCTOR Motor: Destruido motor de " << caballosDeFuerza << " HP." << std::endl;
```

```
    }
```

```
    void arrancar() {
```

```
        if (!encendido) { encendido = true; std::cout << " Motor: ¡BRUM! Encendido." << std::endl; }
```

```
        else { std::cout << " Motor: Ya estaba encendido." << std::endl; }
```

```
    }
```

```
    void detener() {
```

```

        if (encendido) { encendido = false; std::cout << " Motor: ...silencio. Apagado." <<
std::endl;}
        else { std::cout << " Motor: Ya estaba apagado." << std::endl; }
    }
    void mostrarEstado() const {
        std::cout << " Estado del Motor: " << (encendido ? "Encendido" : "Apagado")
        << ", HP: " << caballosDeFuerza << std::endl;
    }
};

```

// Clase 'Todo' o 'Contenedora': Automovil

```

class Automovil {
private:
    std::string marca;
    std::string modelo;
    Motor motorInterno;
public:
    Automovil(std::string ma, std::string mo, int hpDelMotor)
        : marca(ma), modelo(mo), motorInterno(hpDelMotor) {
        std::cout << "CONSTRUCTOR Automovil: Ensamblado un " << marca << " " << modelo
        << " con un motor." << std::endl;
    }
    ~Automovil() {
        std::cout << "DESTRUCTOR Automovil: Desguazando el " << marca << " " << modelo
        << "." << std::endl;
    }
    void encender() {
        std::cout << modelo << ": Intentando encender..." << std::endl;
        motorInterno.arrancar();
    }
    void apagar() {
        std::cout << modelo << ": Intentando apagar..." << std::endl;
        motorInterno.detener();
    }
    void verDiagnostico() const {
        std::cout << "Diagnostico del " << modelo << ":" << std::endl;
        motorInterno.mostrarEstado();
    }
};

```

// Función interactiva para mostrar el autor del proyecto

```

void mostrarAutor() {
    std::cout << "\n👤 Este proyecto fue realizado por: Mateo Perez Nomey\n";
    std::cout << "¡Gracias por revisar este código! 🚗🔧 \n";
}

```

```

int main() {

```

```

std::cout << "=== Proyecto Automovil - Código creado por Mateo Perez Nomey ===\n" <<
std::endl;

std::cout << "--- Creando un Automovil en el Stack ---" << std::endl;
Automovil miAuto("SuperMarca", "ModeloX", 200);
miAuto.verDiagnostico();
miAuto.encender();
miAuto.verDiagnostico();
miAuto.apagar();

std::cout << "\n--- Saliendo de main (miAuto se destruirá) ---" << std::endl;

// Pregunta interactiva
char respuesta;
std::cout << "\n¿Deseas ver el autor del proyecto? (s/n): ";
std::cin >> respuesta;

if (respuesta == 's' || respuesta == 'S') {
    mostrarAutor();
} else {
    std::cout << "Autor oculto. ¡Hasta la próxima! 🙌\n";
}

return 0;
}

```

Automovil - partes

```

=== PROYECTO AUTOMOVIL - Realizado por Mateo Perez Nomey ===

--- Creando un Automovil en el Stack ---
CONSTRUCTOR Motor: Creado motor de 200 HP.
CONSTRUCTOR Rueda: Tipo = Deportiva
CONSTRUCTOR Automovil: Ensamblado un SuperMarca ModeloX
Diagnóstico del ModeloX:
    Estado del Motor: Apagado, HP: 200
ModeloX: Intentando encender...
    Motor: ¡BRUM! Encendido.
Diagnóstico del ModeloX:
    Estado del Motor: Encendido, HP: 200
ModeloX: Intentando apagar...
    Motor: ...silencio. Apagado.

--- Saliendo de main (miAuto se destruirá) ---

¿Deseas conocer al autor del programa? (s/n): s

👤 Autor del proyecto: Mateo Perez Nomey
Gracias por revisar este código 🙌
DESTRUCTOR Automovil: Desguazando el SuperMarca ModeloX
    DESTRUCTOR Rueda: Tipo = Deportiva
    DESTRUCTOR Motor: Destruído motor de 200 HP.

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

```

```

#include <iostream>
#include <string>

// =====
// PROYECTO AUTOMÓVIL - by Mateo Perez Nomey 🧠
// =====

// Clase 'Parte': Motor
class Motor {
private:
    int caballosDeFuerza;
    bool encendido;

public:
    Motor(int hp = 150) : caballosDeFuerza(hp), encendido(false) {
        std::cout << " CONSTRUCTOR Motor: Creado motor de " << caballosDeFuerza << "
HP." << std::endl;
    }

    ~Motor() {
        std::cout << " DESTRUCTOR Motor: Destruido motor de " << caballosDeFuerza << "
HP." << std::endl;
    }

    void arrancar() {
        if (!encendido) {
            encendido = true;
            std::cout << " Motor: ¡BRUM! Encendido." << std::endl;
        } else {
            std::cout << " Motor: Ya estaba encendido." << std::endl;
        }
    }

    void detener() {
        if (encendido) {
            encendido = false;
            std::cout << " Motor: ...silencio. Apagado." << std::endl;
        } else {
            std::cout << " Motor: Ya estaba apagado." << std::endl;
        }
    }

    void mostrarEstado() const {
        std::cout << " Estado del Motor: " << (encendido ? "Encendido" : "Apagado")
<< ", HP: " << caballosDeFuerza << std::endl;
    }
};

```

// Clase 'Parte': Rueda

```
class Rueda {
private:
    std::string tipo;

public:
    Rueda(std::string t = "Normal") : tipo(t) {
        std::cout << " CONSTRUCTOR Rueda: Tipo = " << tipo << std::endl;
    }

    ~Rueda() {
        std::cout << " DESTRUCTOR Rueda: Tipo = " << tipo << std::endl;
    }
};
```

// Clase 'Todo' o 'Contenedora': Automovil

```
class Automovil {
private:
    std::string marca;
    std::string modelo;
    Motor motorInterno;
    Rueda ruedaDelantera;

public:
    Automovil(std::string ma, std::string mo, int hpDelMotor)
        : marca(ma), modelo(mo), motorInterno(hpDelMotor), ruedaDelantera("Deportiva") {
        std::cout << "CONSTRUCTOR Automovil: Ensamblado un " << marca << " " << modelo
        << std::endl;
    }

    ~Automovil() {
        std::cout << "DESTRUCTOR Automovil: Desguazando el " << marca << " " << modelo
        << std::endl;
    }

    void encender() {
        std::cout << modelo << ": Intentando encender..." << std::endl;
        motorInterno.arrancar();
    }

    void apagar() {
        std::cout << modelo << ": Intentando apagar..." << std::endl;
        motorInterno.detener();
    }

    void verDiagnostico() const {
        std::cout << "Diagnóstico del " << modelo << ":" << std::endl;
        motorInterno.mostrarEstado();
    }
};
```

```

    }
};

// Función para mostrar el autor
void mostrarAutor() {
    std::cout << "\n👤 Autor del proyecto: Mateo Perez Nomey" << std::endl;
    std::cout << "Gracias por revisar este código 🚗🔧\n";
}

int main() {
    std::cout << "=== PROYECTO AUTOMÓVIL - Realizado por Mateo Perez Nomey ===\n"
    << std::endl;

    std::cout << "--- Creando un Automovil en el Stack ---" << std::endl;
    Automovil miAuto("SuperMarca", "ModeloX", 200);

    miAuto.verDiagnostico();
    miAuto.encender();
    miAuto.verDiagnostico();
    miAuto.apagar();

    std::cout << "\n--- Saliendo de main (miAuto se destruirá) ---" << std::endl;

    // Interacción final
    char opcion;
    std::cout << "\n¿Deseas conocer al autor del programa? (s/n): ";
    std::cin >> opcion;

    if (opcion == 's' || opcion == 'S') {
        mostrarAutor();
    } else {
        std::cout << "Autor oculto. ¡Hasta la próxima! 🙌\n";
    }

    return 0;
}

```

Proyecto vehículo

```
=== PROYECTO VEHÍCULO - Código creado por Mateo Perez Nomey ===
Vehículo creado: Toyota (2022)
Automóvil con combustible Gasolina creado.
Toyota está arrancando...
Toyota: ¡Piiii!

¿Deseas conocer al autor del proyecto? (s/n): s
👤 Autor del proyecto: Mateo Perez Nomey
Gracias por revisar este código 🙌
Automóvil destruido: Toyota
Vehículo destruido: Toyota

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

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
// =====
```

```
// PROYECTO VEHÍCULO - by Mateo Perez Nomey 🚗
```

```
// =====
```

```
class Vehiculo {
```

```
protected:
```

```
    string marca;
```

```
    int año;
```

```
public:
```

```
    Vehiculo(string m, int a) : marca(m), año(a) {
```

```
        cout << "Vehículo creado: " << marca << " (" << año << ")" << endl;
```

```
    }
```

```
    ~Vehiculo() {
```

```
        cout << "Vehículo destruido: " << marca << endl;
```

```
    }
```

```
    void arrancar() const {
```

```
        cout << marca << " está arrancando..." << endl;
```

```
    }
```

```
};
```

```
class Automovil : public Vehiculo {
```

```
private:
```

```
    string tipoCombustible;
```

```
public:
```

```
    Automovil(string m, int a, string tc) : Vehiculo(m, a), tipoCombustible(tc) {
```

```
        cout << "Automóvil con combustible " << tipoCombustible << " creado." << endl;
```



```

    }

    ~Automovil() {
        cout << "Automóvil destruido: " << marca << endl;
    }

    void tocarBocina() const {
        cout << marca << ": ¡Piiii!" << endl;
    }
};

// Función interactiva para mostrar el autor
void mostrarAutor() {
    cout << "\n👤 Autor del proyecto: Mateo Perez Nomey\n";
    cout << "Gracias por revisar este código 🚗✨\n";
}

int main() {
    cout << "=== PROYECTO VEHÍCULO - Código creado por Mateo Perez Nomey ===\n"
    << endl;

    Automovil a("Toyota", 2022, "Gasolina");
    a.arrancar();
    a.tocarBocina();

    // Interacción final
    char respuesta;
    cout << "\n¿Deseas conocer al autor del proyecto? (s/n): ";
    cin >> respuesta;

    if (respuesta == 's' || respuesta == 'S') {
        mostrarAutor();
    } else {
        cout << "Autor oculto. ¡Hasta la próxima! 🙌\n";
    }

    return 0;
}

```

Formas geometricas

```

=== PROYECTO FORMAS GEOMETRICAS - Código creado por Mateo Perez Nomey ===

--- Descripción de Forma ---
Mi Círculo (Rojo): Dibujando CIRCULO con radio 7.
    Área: 153.938
    Color: Rojo

--- Descripción de Forma ---
Mi Rectángulo (Azul): Dibujando RECTANGULO base 4, altura 5.
    Área: 20
    Color: Azul

--- Procesando todas las formas del vector ---
Mi Círculo (Rojo): Dibujando CIRCULO con radio 7.
Mi Rectángulo (Azul): Dibujando RECTANGULO base 4, altura 5.
Otro Círculo (Verde): Dibujando CIRCULO con radio 3.

--- Liberando memoria ---
DESTRUCTOR Círculo: Otro Círculo
DESTRUCTOR FormaGeometrica: Otro Círculo
DESTRUCTOR Rectángulo: Mi Rectángulo
DESTRUCTOR FormaGeometrica: Mi Rectángulo
DESTRUCTOR Círculo: Mi Círculo
DESTRUCTOR FormaGeometrica: Mi Círculo

¿Deseas conocer al autor del proyecto? (s/n): s

👤 Autor del proyecto: Mateo Perez Nomey
Gracias por revisar este código de formas geométricas 📄

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

```

```

#include <iostream>
#include <string>
#include <vector>
#include <cmath> // Para M_PI

// =====
//  PROYECTO FORMAS GEOMÉTRICAS - Mateo Perez Nomey 🧠
//  =====

// Clase Base ABSTRACTA
class FormaGeometrica {
protected:
    std::string nombreForma;
    std::string color;
public:
    FormaGeometrica(std::string nf, std::string c) : nombreForma(nf), color(c) {}

    // FUNCIONES VIRTUALES PURAS
    virtual void dibujar() const = 0;
    virtual double calcularArea() const = 0;

    // Métodos comunes
    std::string getColor() const { return color; }
    std::string getNombre() const { return nombreForma; }

```

```

    virtual ~FormaGeometrica() {
        std::cout << "DESTRUCTOR FormaGeometrica: " << nombreForma << std::endl;
    }
};

// Clase Derivada: Circulo
class Circulo : public FormaGeometrica {
private:
    double radio;
public:
    Circulo(std::string nf, std::string c, double r) : FormaGeometrica(nf, c), radio(r) {}

    void dibujar() const override {
        std::cout << getNombre() << " (" << getColor() << "): Dibujando CIRCULO con radio "
<< radio << "." << std::endl;
    }
    double calcularArea() const override {
        return M_PI * radio * radio;
    }
    ~Circulo() override {
        std::cout << " DESTRUCTOR Circulo: " << getNombre() << std::endl;
    }
};

// Clase Derivada: Rectangulo
class Rectangulo : public FormaGeometrica {
private:
    double base, altura;
public:
    Rectangulo(std::string nf, std::string c, double b, double h) : FormaGeometrica(nf, c),
base(b), altura(h) {}




    void dibujar() const override {
        std::cout << getNombre() << " (" << getColor() << "): Dibujando RECTANGULO base "
<< base << ", altura " << altura << "." << std::endl;
    }
    double calcularArea() const override {
        return base * altura;
    }
    ~Rectangulo() override {
        std::cout << " DESTRUCTOR Rectangulo: " << getNombre() << std::endl;
    }
};

// Función para mostrar descripción de forma
void describirForma(const FormaGeometrica* forma) {
    std::cout << "\n--- Descripción de Forma ---" << std::endl;
    forma->dibujar();
}

```

```

std::cout << " Área: " << forma->calcularArea() << std::endl;
std::cout << " Color: " << forma->getColor() << std::endl;
}

// Función interactiva para mostrar autor
void mostrarAutor() {
    std::cout << "\n👤 Autor del proyecto: Mateo Perez Nomey\n";
    std::cout << "Gracias por revisar este código de formas geométricas    \n";
}

int main() {
    std::cout << "=== PROYECTO FORMAS GEOMÉTRICAS - Código creado por Mateo Perez Nomey ===\n" << std::endl;

    FormaGeometrica* ptrCirculo = new Circulo("Mi Círculo", "Rojo", 7.0);
    FormaGeometrica* ptrRectangulo = new Rectangulo("Mi Rectángulo", "Azul", 4.0, 5.0);

    describirForma(ptrCirculo);
    describirForma(ptrRectangulo);

    std::vector<FormaGeometrica*> misFormas;
    misFormas.push_back(ptrCirculo);
    misFormas.push_back(ptrRectangulo);
    misFormas.push_back(new Circulo("Otro Círculo", "Verde", 3.0));

    std::cout << "\n--- Procesando todas las formas del vector ---" << std::endl;
    for (const FormaGeometrica* f : misFormas) {
        f->dibujar();
    }

    std::cout << "\n--- Liberando memoria ---" << std::endl;
    delete misFormas[2]; // Borra "Otro Círculo"
    delete ptrRectangulo; // Ya estaba en el vector
    delete ptrCirculo; // Ya estaba en el vector

    // Interacción final
    char opcion;
    std::cout << "\n¿Deseas conocer al autor del proyecto? (s/n): ";
    std::cin >> opcion;

    if (opcion == 's' || opcion == 'S') {
        mostrarAutor();
    } else {
        std::cout << "Autor oculto. ¡Hasta la próxima! 🙌\n";
    }

    return 0;
}

```

Boton Herencia

```
=== PROYECTO BOTON - Código creado por Mateo Perez Nomey ===
Botón creado: [OK]
Botón OK personalizado listo.
Mostrando botón: [OK]
;Botón OK presionado!

;Deseas conocer al autor del proyecto? (s/n): s
👤 Autor del proyecto: Mateo Perez Nomey
Gracias por probar este botón interactivo 🟢
Botón OK destruido.
Botón destruido: [OK]

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

```
#include <iostream>
#include <string>
using namespace std;

// =====
// PROYECTO BOTÓN - by Mateo Perez Nomey 🖱️
// =====

class Boton {
protected:
    string etiqueta;

public:
    Boton(string e) : etiqueta(e) {
        cout << "Botón creado: [" << etiqueta << "]" << endl;
    }

    ~Boton() {
        cout << "Botón destruido: [" << etiqueta << "]" << endl;
    }

    void mostrar() const {
        cout << "Mostrando botón: [" << etiqueta << "]" << endl;
    }
};

class BotonOk : public Boton {
public:
    BotonOk() : Boton("OK") {
        cout << "Botón OK personalizado listo." << endl;
    }

    ~BotonOk() {
        cout << "Botón OK destruido." << endl;
    }
}
```

```

void presionar() const {
    cout << "¡Botón OK presionado!" << endl;
}
};

// Función para mostrar el autor del programa
void mostrarAutor() {
    cout << "\n👤 Autor del proyecto: Mateo Perez Nomey\n";
    cout << "Gracias por probar este botón interactivo 🖱️✅\n";
}

int main() {
    cout << "=== PROYECTO BOTÓN - Código creado por Mateo Perez Nomey ===\n" <<
endl;

    BotonOk b;
    b.mostrar();
    b.presionar();

    // Opción interactiva
    char respuesta;
    cout << "\n¿Deseas conocer al autor del proyecto? (s/n): ";
    cin >> respuesta;

    if (respuesta == 's' || respuesta == 'S') {
        mostrarAutor();
    } else {
        cout << "Autor oculto. ¡Hasta la próxima! 🙌\n";
    }

    return 0;
}

```

Clase Perro

```
👋;Hola! Soy Mateo Perez Nomey, creador de este mini programa de perros.

--- Conozcamos a nuestras mascotas ---
Hola, soy Firulais, un Mestizo Juguetón de 3 años.
Firulais dice: ¡Guau! ¡Guau!

Hola, soy Rex, un Pastor Alemán de 5 años.
Rex dice: ¡Guau! ¡Guau!

👋;Gracias por usar este código! Espero que te haya gustado. 🍷🍷🍷

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

```
#include <iostream>
#include <string>
using namespace std;

// =====
// CLASE 'Perro'
// =====
class Perro {
public:
    string nombre;
    string raza;
    int edad;

    void ladrar() {
        cout << nombre << " dice: ¡Guau! ¡Guau!" << endl;
    }

    void presentar() {
        cout << "Hola, soy " << nombre << ", un " << raza
            << " de " << edad << " años." << endl;
    }
};

// =====
// CLASE PERSONALIZADA: MateoPerezNomey
// =====
class MateoPerezNomey {
public:
    void presentarse() const {
        cout << "\n👋 ¡Hola! Soy Mateo Perez Nomey, creador de este mini programa de
perros." << endl;
    }

    void despedirse() const {
        cout << "\n👋 ¡Gracias por usar este código! Espero que te haya gustado. 🍷" << endl;
    }
};
```

```

    }
};

// =====
// FUNCIÓN PRINCIPAL
// =====
int main() {
    MateoPerezNomey autor;
    autor.presentarse();

    Perro perro1;
    Perro perro2;

    perro1.nombre = "Firulais";
    perro1.raza = "Mestizo Juguetón";
    perro1.edad = 3;

    perro2.nombre = "Rex";
    perro2.raza = "Pastor Alemán";
    perro2.edad = 5;

    cout << "\n--- Conozcamos a nuestras mascotas ---" << endl;
    perro1.presentar();
    perro1.ladrar();

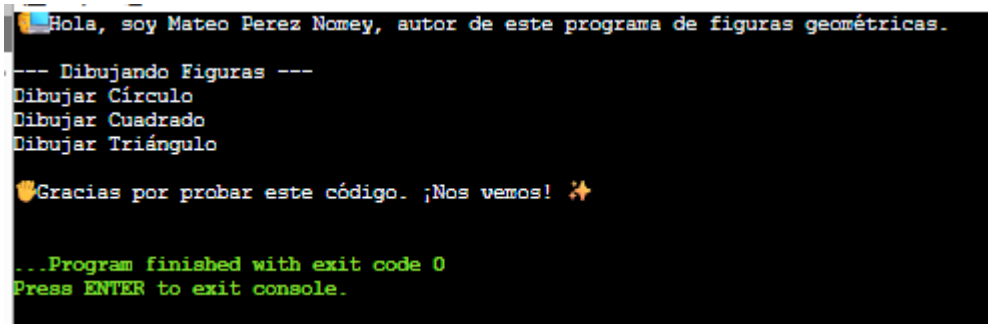
    cout << endl;
    perro2.presentar();
    perro2.ladrar();

    autor.despedirse();

    return 0;
}

```

Coleccion Polimórfica



```

Hola, soy Mateo Perez Nomey, autor de este programa de figuras geométricas.

--- Dibujando Figuras ---
Dibujar Círculo
Dibujar Cuadrado
Dibujar Triángulo

👍 Gracias por probar este código. ¡Nos vemos! ✨

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

```

```
#include <iostream>
```



```

#include <vector>
using namespace std;

// =====
// Clase base abstracta: Figura
// =====
class Figura {
public:
    virtual void dibujar() const = 0; // Método virtual puro
    virtual ~Figura() {} // Destructor virtual
};

// =====
// Clases derivadas
// =====
class Circulo : public Figura {
public:
    void dibujar() const override {
        cout << "Dibujar Círculo" << endl;
    }
};

class Cuadrado : public Figura {
public:
    void dibujar() const override {
        cout << "Dibujar Cuadrado" << endl;
    }
};

class Triangulo : public Figura {
public:
    void dibujar() const override {
        cout << "Dibujar Triángulo" << endl;
    }
};

// =====
// Clase personalizada: MateoPerezNomey
// =====
class MateoPerezNomey {
public:
    void presentarse() const {
        cout << "👤 Hola, soy Mateo Perez Nomey, autor de este programa de figuras
geométricas.\n" << endl;
    }

    void despedirse() const {
        cout << "\n👋 Gracias por probar este código. ¡Nos vemos! ✨" << endl;
    }
};

```

```

    }
};

// =====
// Función principal
// =====
int main() {
    MateoPerezNomey autor;
    autor.presentarse();

    vector<Figura*> figuras;
    figuras.push_back(new Circulo());
    figuras.push_back(new Cuadrado());
    figuras.push_back(new Triangulo());

    cout << "--- Dibujando Figuras ---" << endl;
    for (const Figura* f : figuras) {
        f->dibujar();
    }

    for (Figura* f : figuras) {
        delete f;
    }

    autor.despedirse();
    return 0;
}

```

Constructor animal

```

Hola, soy Mateo Perez Nomey, autor de este programa de animales.

--- Creando un Animal generico ---
CONSTRUCTOR Animal: Nace un animal llamado 'Alex el Leon' de 10 anio(s).
Alex el Leon esta comiendo.

--- Creando un Perro ---
CONSTRUCTOR Animal: Nace un animal llamado 'Buddy' de 3 anio(s).
CONSTRUCTOR Perro: Nace un perro de raza 'Golden Retriever', que tambien es un animal llamado 'Buddy'.
Buddy esta comiendo.
Buddy esta durmiendo.
El nombre de mi mascota es: Buddy
Buddy (un Golden Retriever) dice: ¡Guau! ¡Guau!

--- Fin de main (los objetos se destruirán) ---

Gracias por usar el programa. ¡Hasta la próxima!
DESTRUCTOR Perro: Muere el perro 'Buddy' de raza 'Golden Retriever'.
DESTRUCTOR Animal: Muere el animal 'Buddy'.
DESTRUCTOR Animal: Muere el animal 'Alex el Leon'.

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

```

#include <iostream>

```

#include <string>
using namespace std;

// =====
// Clase Base
// =====
class Animal {
protected:
    string nombre;
    int edad;

public:
    Animal(const string& n, int e) : nombre(n), edad(e) {
        cout << " CONSTRUCTOR Animal: Nace un animal llamado " << nombre
            << " de " << edad << " anio(s)." << endl;
    }
    ~Animal() {
        cout << " DESTRUCTOR Animal: Muere el animal " << nombre << "." << endl;
    }
    void comer() const {
        cout << nombre << " esta comiendo." << endl;
    }
    void dormir() const {
        cout << nombre << " esta durmiendo." << endl;
    }
    string getNombre() const { return nombre; }
};

// =====
// Clase Derivada: Perro
// =====
class Perro : public Animal {
private:
    string raza;

public:
    Perro(const string& n, int e, const string& r)
        : Animal(n, e), raza(r) {
        cout << " CONSTRUCTOR Perro: Nace un perro de raza " << raza
            << ", que tambien es un animal llamado " << nombre << "." << endl;
    }

    ~Perro() {
        cout << " DESTRUCTOR Perro: Muere el perro " << nombre
            << " de raza " << raza << "." << endl;
    }

    void ladrar() const {

```

```

        cout << getNombre() << " (un " << raza << ") dice: ¡Guau! ¡Guau!" << endl;
    }
};

// =====
// Clase Personalizada: MateoPerezNomey
// =====
class MateoPerezNomey {
public:
    void presentarse() const {
        cout << "👋 Hola, soy Mateo Perez Nomey, autor de este programa de animales.\n" <<
endl;
    }

    void despedirse() const {
        cout << "\n👋 Gracias por usar el programa. ¡Hasta la próxima! 🐾" << endl;
    }
};

// =====
// Función Principal
// =====
int main() {
    MateoPerezNomey autor;
    autor.presentarse();

    cout << "--- Creando un Animal generico ---" << endl;
    Animal animalGenerico("Alex el Leon", 10);
    animalGenerico.comer();

    cout << "\n--- Creando un Perro ---" << endl;
    Perro miMascota("Buddy", 3, "Golden Retriever");

    miMascota.comer();
    miMascota.dormir();
    cout << "El nombre de mi mascota es: " << miMascota.getNombre() << endl;
    miMascota.ladRAR();

    cout << "\n--- Fin de main (los objetos se destruirán) ---" << endl;

    autor.despedirse();
    return 0;
}

```

Describir forma geométrica

```
👋 Bienvenido al programa de figuras geométricas.
Autor: Mateo Perez Nomey

--- Descripción de Forma ---
Círculo Principal (Rojo): Dibujando CIRCULO con radio 5.
    Área: 78.5398
    Color: Rojo
DESTRUCTOR Círculo: Círculo Principal
DESTRUCTOR FormaGeometrica: Círculo Principal

👋 Gracias por usar este programa. ¡Hasta la próxima! 🍷

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

```
#include <iostream>
#include <string>
#include <vector>
#include <cmath> // Para operaciones matemáticas

#ifndef M_PI
#define M_PI 3.14159265358979323846
#endif

// =====
// Clase Base ABSTRACTA
// =====
class FormaGeometrica {
protected:
    std::string nombreForma;
    std::string color;

public:
    FormaGeometrica(std::string nf, std::string c) : nombreForma(nf), color(c) {}

    virtual void dibujar() const = 0;
    virtual double calcularArea() const = 0;

    std::string getColor() const { return color; }
    std::string getNombre() const { return nombreForma; }

    virtual ~FormaGeometrica() {
        std::cout << "DESTRUCTOR FormaGeometrica: " << nombreForma << std::endl;
    }
};

// =====
// Clase Derivada: Circulo
// =====
class Circulo : public FormaGeometrica {
```

```

private:
    double radio;

public:
    Circulo(std::string nf, std::string c, double r) : FormaGeometrica(nf, c), radio(r) {}

    void dibujar() const override {
        std::cout << getNombre() << " (" << getColor() << "): Dibujando CIRCULO con radio "
            << radio << "." << std::endl;
    }

    double calcularArea() const override {
        return M_PI * radio * radio;
    }

    ~Circulo() override {
        std::cout << " DESTRUCTOR Circulo: " << getNombre() << std::endl;
    }
};

// =====
// Clase Personalizada: MateoPerezNomey
// =====
class MateoPerezNomey {
public:
    void presentarse() const {
        std::cout << "👋 Bienvenido al programa de figuras geométricas." << std::endl;
        std::cout << "Autor: Mateo Perez Nomey\n" << std::endl;
    }

    void despedirse() const {
        std::cout << "\n👋 Gracias por usar este programa. ¡Hasta la próxima! 🧠📐" <<
std::endl;
    }
};

// =====
// Función auxiliar
// =====
void describirForma(const FormaGeometrica* forma) {
    std::cout << "\n--- Descripción de Forma ---" << std::endl;
    forma->dibujar();
    std::cout << " Área: " << forma->calcularArea() << std::endl;
    std::cout << " Color: " << forma->getColor() << std::endl;
}

// =====
// Función principal

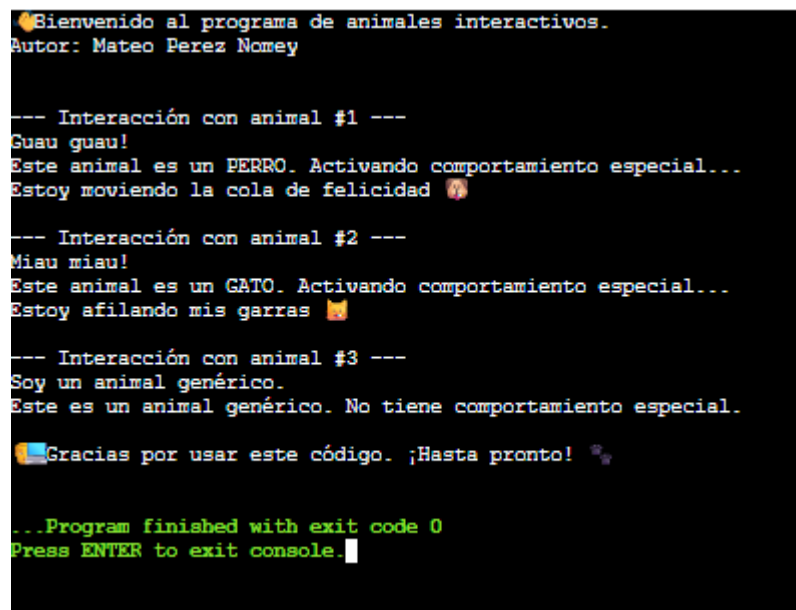
```

```
// =====
int main() {
    MateoPerezNomey autor;
    autor.presentarse();

    FormaGeometrica* miCirculo = new Circulo("Círculo Principal", "Rojo", 5.0);
    describirForma(miCirculo);
    delete miCirculo;

    autor.despedirse();
    return 0;
}
```

DInamyc Cast



```
👋Bienvenido al programa de animales interactivos.
Autor: Mateo Perez Nomey

--- Interacción con animal #1 ---
Guau guau!
Este animal es un PERRO. Activando comportamiento especial...
Estoy moviendo la cola de felicidad 🐶

--- Interacción con animal #2 ---
Miau miau!
Este animal es un GATO. Activando comportamiento especial...
Estoy afilando mis garras 🐱

--- Interacción con animal #3 ---
Soy un animal genérico.
Este es un animal genérico. No tiene comportamiento especial.

👋Gracias por usar este código. ¡Hasta pronto! 🐾

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

```
#include <iostream>
#include <string>
using namespace std;

// =====
// Clase base
// =====
class Animal {
public:
    virtual void hablar() const {
        cout << "Soy un animal genérico." << endl;
    }

    virtual ~Animal() {} // Destructor virtual
};

// =====
```

```

// Clase derivada: Perro
// =====
class Perro : public Animal {
public:
    void hablar() const override {
        cout << "Guau guau!" << endl;
    }

    void moverCola() const {
        cout << "Estoy moviendo la cola de felicidad 🐶." << endl;
    }
};

// =====
// Clase derivada: Gato
// =====
class Gato : public Animal {
public:
    void hablar() const override {
        cout << "Miau miau!" << endl;
    }

    void afilarGarras() const {
        cout << "Estoy afilando mis garras 🐱." << endl;
    }
};

// =====
// Clase personalizada: MateoPerezNomey
// =====
class MateoPerezNomey {
public:
    void presentarse() const {
        cout << "👋 Bienvenido al programa de animales interactivos." << endl;
        cout << "Autor: Mateo Perez Nomey\n" << endl;
    }

    void despedirse() const {
        cout << "\n👋 Gracias por usar este código. ¡Hasta pronto! 🐾" << endl;
    }
};

// =====
// Función que recibe un puntero a Animal
// =====
void interactuarConAnimal(Animal* animal) {
    animal->hablar(); // Comportamiento polimórfico
}

```



```

// Intentamos convertir a Perro
if (Perro* ptrPerro = dynamic_cast<Perro*>(animal)) {
    cout << "Este animal es un PERRO. Activando comportamiento especial..." << endl;
    ptrPerro->moverCola();
}

// Intentamos convertir a Gato
else if (Gato* ptrGato = dynamic_cast<Gato*>(animal)) {
    cout << "Este animal es un GATO. Activando comportamiento especial..." << endl;
    ptrGato->afilarGarras();
}

else {
    cout << "Este es un animal genérico. No tiene comportamiento especial." << endl;
}
}

// =====
// Función principal
// =====
int main() {
    MateoPerezNomey autor;
    autor.presentarse();

    Animal* listaDeAnimales[3];
    listaDeAnimales[0] = new Perro();
    listaDeAnimales[1] = new Gato();
    listaDeAnimales[2] = new Animal();

    for (int i = 0; i < 3; ++i) {
        cout << "\n--- Interacción con animal #" << (i + 1) << " ---" << endl;
        interactuarConAnimal(listaDeAnimales[i]);
    }

    for (int i = 0; i < 3; ++i) {
        delete listaDeAnimales[i];
    }

    autor.despedirse();
    return 0;
}

```

Especialidad Figura

```
👋 Bienvenido al programa de figuras geométricas.
Autor: Mateo Perez Nomey

--- Creando y dibujando una Figura generica ---
CONSTRUCTOR Figura: 'Figura Misteriosa' de color Azul
Figura 'Figura Misteriosa': Dibujando una figura geometrica generica de color Azul.

--- Creando y dibujando un Circulo ---
CONSTRUCTOR Figura: 'Circulo' de color Rojo
CONSTRUCTOR Circulo: Radio 5
Circulo 'Circulo': Dibujando un circulo perfecto de color Rojo y radio 5.
Area: 78.5397

--- Creando y dibujando un Rectangulo ---
CONSTRUCTOR Figura: 'Rectangulo' de color Verde
CONSTRUCTOR Rectangulo: Base 4, Altura 6
Rectangulo 'Rectangulo': Dibujando un rectangulo de color Verde con base 4 y altura 6
Area: 24

--- Fin de main ---

👋 Gracias por usar este programa. ¡Hasta pronto!
DESTRUCTOR Rectangulo: Base 4, Altura 6
DESTRUCTOR Figura: 'Rectangulo'
DESTRUCTOR Circulo: Radio 5
DESTRUCTOR Figura: 'Circulo'
DESTRUCTOR Figura: 'Figura Misteriosa'

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

```
#include <iostream>
#include <string>
#define PI 3.14159

// =====
// Clase Personalizada: Autor
// =====
class MateoPerezNomey {
public:
    void presentarse() const {
        std::cout << "👋 Bienvenido al programa de figuras geométricas." << std::endl;
        std::cout << "Autor: Mateo Perez Nomey\n" << std::endl;
    }

    void despedirse() const {
        std::cout << "\n🧠 Gracias por usar este programa. ¡Hasta pronto!" << std::endl;
    }
};

// =====
// Clase base
// =====
class Figura {
protected:
    std::string color;
    std::string nombreFigura;
```

```

public:
    Figura(std::string c, std::string nf) : color(c), nombreFigura(nf) {
        std::cout << " CONSTRUCTOR Figura: " << nombreFigura << " de color " << color <<
std::endl;
    }

    virtual void dibujar() const {
        std::cout << "Figura " << nombreFigura << ": Dibujando una figura geometrica
generica de color "
            << color << "." << std::endl;
    }

    virtual ~Figura() {
        std::cout << " DESTRUCTOR Figura: " << nombreFigura << "" << std::endl;
    }
};

// =====
// Clase derivada: Circulo
// =====
class Circulo : public Figura {
private:
    double radio;
public:
    Circulo(std::string c, double r) : Figura(c, "Circulo"), radio(r) {
        std::cout << " CONSTRUCTOR Circulo: Radio " << radio << std::endl;
    }

    void dibujar() const override {
        std::cout << "Circulo " << nombreFigura << ": Dibujando un circulo perfecto de color "
<< color
            << " y radio " << radio << "." << std::endl;
        std::cout << " Area: " << (PI * radio * radio) << std::endl;
    }

    ~Circulo() override {
        std::cout << " DESTRUCTOR Circulo: Radio " << radio << std::endl;
    }
};

// =====
// Clase derivada: Rectangulo
// =====
class Rectangulo : public Figura {
private:
    double base, altura;
public:
    Rectangulo(std::string c, double b, double h) : Figura(c, "Rectangulo"), base(b), altura(h) {

```

```

        std::cout << "   CONSTRUCTOR Rectangulo: Base " << b << ", Altura " << h <<
std::endl;
    }

    void dibujar() const override {
        std::cout << "Rectangulo '" << nombreFigura << "': Dibujando un rectangulo de color "
<< color
            << " con base " << base << " y altura " << altura << "." << std::endl;
        std::cout << "       Area: " << (base * altura) << std::endl;
    }

    ~Rectangulo() override {
        std::cout << "   DESTRUCTOR Rectangulo: Base " << base << ", Altura " << altura <<
std::endl;
    }
};

// =====
// Función principal
// =====
int main() {
    MateoPerezNomey autor;
    autor.presentarse();

    std::cout << "--- Creando y dibujando una Figura generica ---" << std::endl;
    Figura fig("Azul", "Figura Misteriosa");
    fig.dibujar();

    std::cout << "\n--- Creando y dibujando un Circulo ---" << std::endl;
    Circulo circ("Rojo", 5.0);
    circ.dibujar();

    std::cout << "\n--- Creando y dibujando un Rectangulo ---" << std::endl;
    Rectangulo rect("Verde", 4.0, 6.0);
    rect.dibujar();

    std::cout << "\n--- Fin de main ---" << std::endl;

    autor.despedirse();
    return 0;
}

```

Estudiante

```
Programa realizado por: Mateo Perez Nomey
Clase: Gestión de Estudiantes

Estudiante 'Juaquin Soliz' creado.
-----
Nombre: Juaquin Soliz
Edad: 20 años
Matrícula: A123
Promedio: 0
-----

Intentando actualizar edad y promedio...
Error: Edad '150' inválida para el estudiante Juaquin Soliz. Edad no modificada.
Error: Promedio '-2' inválido para Juaquin Soliz. Promedio no modificado.

Información actualizada de Juaquin Soliz:
-----
Nombre: Juaquin Soliz
Edad: 21 años
Matrícula: A123
Promedio: 8.5
-----

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

🧠 Fin del programa. ¡Gracias por utilizar este sistema educativo!

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

```
#include <iostream>
```

```
#include <string>
```

```
// Clase de autor
```

```
class MateoPerezNomey {
```

```
public:
```

```
    void presentarse() const {
```

```
        std::cout << "👋 Programa realizado por: Mateo Perez Nomey\n"
```

```
        << "📖 Clase: Gestión de Estudiantes\n" << std::endl;
```

```
    }
```

```
    void despedirse() const {
```

```
        std::cout << "\n🧠 Fin del programa. ¡Gracias por utilizar este sistema educativo!" <<
std::endl;
```

```
    }
```

```
};
```

```
class Estudiante {
```

```
private:
```

```
    std::string nombre;
```

```
    int edad;
```

```
    std::string matricula;
```

```
    double promedio;
```

public:

```
Estudiante(std::string nom, int ed, std::string matr) {  
    nombre = nom;  
    setEdad(ed);  
    matricula = matr;  
    promedio = 0.0;  
    std::cout << "Estudiante " << nombre << " creado." << std::endl;  
}
```

```
std::string getNombre() const { return nombre; }  
int getEdad() const { return edad; }  
std::string getMatricula() const { return matricula; }  
double getPromedio() const { return promedio; }
```

```
void setNombre(const std::string& nuevoNombre) {  
    if (!nuevoNombre.empty()) {  
        nombre = nuevoNombre;  
    } else {  
        std::cout << "Error: El nombre no puede estar vacío." << std::endl;  
    }  
}
```

```
void setEdad(int nuevaEdad) {  
    if (nuevaEdad >= 5 && nuevaEdad <= 100) {  
        edad = nuevaEdad;  
    } else {  
        std::cout << "Error: Edad " << nuevaEdad << " inválida para el estudiante "  
            << nombre << ". Edad no modificada." << std::endl;  
    }  
}
```

```
void setPromedio(double nuevoPromedio) {  
    if (nuevoPromedio >= 0.0 && nuevoPromedio <= 10.0) {  
        promedio = nuevoPromedio;  
    } else {  
        std::cout << "Error: Promedio " << nuevoPromedio << " inválido para "  
            << nombre << ". Promedio no modificado." << std::endl;  
    }  
}
```

```
void mostrarInformacion() const {  
    std::cout << "-----" << std::endl;  
    std::cout << "Nombre: " << nombre << std::endl;  
    std::cout << "Edad: " << edad << " años" << std::endl;  
    std::cout << "Matrícula: " << matricula << std::endl;  
    std::cout << "Promedio: " << promedio << std::endl;  
    std::cout << "-----" << std::endl;  
}
```

```

    }
};

int main() {
    MateoPerezNomey autor;
    autor.presentarse();

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

    std::cout << "\nIntentando actualizar edad y promedio..." << std::endl;
    estudiante1.setEdad(21);
    estudiante1.setPromedio(8.5);
    estudiante1.setEdad(150);
    estudiante1.setPromedio(-2.0);

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

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

    autor.despedirse();
    return 0;
}

```

Estudiante error

```
Programa desarrollado por: Mateo Perez Nomey
Tema: Encapsulamiento con validaciones y atributos adicionales

Estudiante 'Juaquin Soliz' creado.
-----
Nombre: Juaquin Soliz
Edad: 20 años
Matrícula: A123
Promedio: 0
Carrera: Ingeniería
Correo: juaquin@upds.edu
-----

Intentando actualizar edad, promedio y correo...
Error: Correo electrónico inválido.
Error: Edad '150' inválida para el estudiante Juaquin Soliz.
Error: Promedio '-2' inválido para Juaquin Soliz.

Información actualizada de Juaquin Soliz:
-----
Nombre: Juaquin Soliz
Edad: 21 años
Matrícula: A123
Promedio: 8.5
Carrera: Ingeniería
Correo: juaquin@upds.edu
-----

Error: Edad '-10' inválida para el estudiante Priscila Vaca.
Error: Correo electrónico inválido.
Estudiante 'Priscila Vaca' creado.
-----
Nombre: Priscila Vaca
Edad: 1194243552 años
Matrícula: B456
Promedio: 0
Carrera: No especificada
Correo:
-----

✅ Fin del programa. Gracias por revisar el sistema de estudiantes.

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

```
#include <iostream>
#include <string>
#include <stdexcept>
```

```
// Clase de presentación del autor
class MateoPerezNomey {
public:
    void presentarse() const {
        std::cout << "👤 Programa desarrollado por: Mateo Perez Nomey\n"
                    << "📘 Tema: Encapsulamiento con validaciones y atributos adicionales\n" <<
std::endl;
    }

    void despedirse() const {
        std::cout << "\n✅ Fin del programa. Gracias por revisar el sistema de estudiantes." <<
std::endl;
    }
};
```


// Clase Estudiante con encapsulamiento y validaciones

```
class Estudiante {
```

```
private:
```

```
    std::string nombre;  
    int edad;  
    std::string matricula;  
    double promedio;  
    std::string carrera;  
    std::string correo;
```

```
public:
```

```
    Estudiante(std::string nom, int ed, std::string matr, std::string carr = "", std::string mail = "")
```

```
{
```

```
    setNombre(nom);  
    setEdad(ed);  
    matricula = matr;  
    promedio = 0.0;  
    setCarrera(carr);  
    setCorreo(mail);  
    std::cout << "Estudiante " << nombre << " creado." << std::endl;
```

```
}
```

```
// Getters
```

```
std::string getNombre() const { return nombre; }  
int getEdad() const { return edad; }  
std::string getMatricula() const { return matricula; }  
double getPromedio() const { return promedio; }  
std::string getCarrera() const { return carrera; }  
std::string getCorreo() const { return correo; }
```

```
// Setters
```

```
void setNombre(const std::string& nuevoNombre) {  
    if (!nuevoNombre.empty()) {  
        nombre = nuevoNombre;  
    } else {  
        std::cout << "Error: El nombre no puede estar vacío." << std::endl;  
    }  
}
```

```
void setEdad(int nuevaEdad) {  
    if (nuevaEdad >= 5 && nuevaEdad <= 100) {  
        edad = nuevaEdad;  
    } else {  
        std::cout << "Error: Edad " << nuevaEdad << " inválida para el estudiante " <<  
nombre << "." << std::endl;  
    }  
}
```

```

void setPromedio(double nuevoPromedio) {
    if (nuevoPromedio >= 0.0 && nuevoPromedio <= 10.0) {
        promedio = nuevoPromedio;
    } else {
        std::cout << "Error: Promedio " << nuevoPromedio << " inválido para " << nombre
<< "." << std::endl;
    }
}

void setCarrera(const std::string& nuevaCarrera) {
    carrera = nuevaCarrera.empty() ? "No especificada" : nuevaCarrera;
}

void setCorreo(const std::string& nuevoCorreo) {
    if (nuevoCorreo.find('@') != std::string::npos) {
        correo = nuevoCorreo;
    } else {
        std::cout << "Error: Correo electrónico inválido." << std::endl;
    }
}

void mostrarInformacion() const {
    std::cout << "-----" << std::endl;
    std::cout << "Nombre: " << nombre << std::endl;
    std::cout << "Edad: " << edad << " años" << std::endl;
    std::cout << "Matrícula: " << matricula << std::endl;
    std::cout << "Promedio: " << promedio << std::endl;
    std::cout << "Carrera: " << carrera << std::endl;
    std::cout << "Correo: " << correo << std::endl;
    std::cout << "-----" << std::endl;
}

};

int main() {
    MateoPerezNomey autor;
    autor.presentarse();

    Estudiante estudiante1("Juaquin Soliz", 20, "A123", "Ingeniería", "juaquin@upds.edu");
    estudiante1.mostrarInformacion();

    std::cout << "\nIntentando actualizar edad, promedio y correo..." << std::endl;
    estudiante1.setEdad(21);
    estudiante1.setPromedio(8.5);
    estudiante1.setCorreo("correo-invalido"); // Inválido
    estudiante1.setEdad(150);                // Inválido
    estudiante1.setPromedio(-2.0);           // Inválido
}

```

```

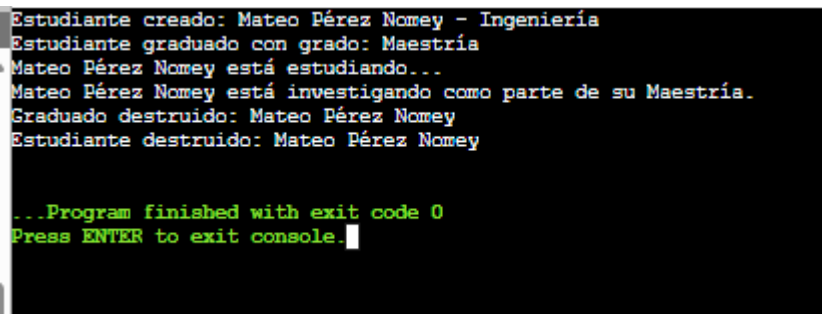
std::cout << "\nInformación actualizada de " << estudiante1.getNombre() << ":\n" <<
std::endl;
estudiante1.mostrarInformacion();

Estudiante estudiante2("Priscila Vaca", -10, "B456", "", "pvaca.com");
estudiante2.mostrarInformacion();

autor.despedirse();
return 0;
}

```

Estudiante herencia



```

Estudiante creado: Mateo Pérez Nomey - Ingeniería
Estudiante graduado con grado: Maestría
Mateo Pérez Nomey está estudiando...
Mateo Pérez Nomey está investigando como parte de su Maestría.
Graduado destruido: Mateo Pérez Nomey
Estudiante destruido: Mateo Pérez Nomey

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

```

```

#include <iostream>
#include <string>
using namespace std;

class Estudiante {
protected:
    string nombre;
    string carrera;

public:
    Estudiante(string n, string c) : nombre(n), carrera(c) {
        cout << "Estudiante creado: " << nombre << " - " << carrera << endl;
    }

    ~Estudiante() {
        cout << "Estudiante destruido: " << nombre << endl;
    }

    void estudiar() const {
        cout << nombre << " está estudiando..." << endl;
    }
};

class EstudianteGraduado : public Estudiante {

```

```

private:
    string grado;

public:
    EstudianteGraduado(string n, string c, string g) : Estudiante(n, c), grado(g) {
        cout << "Estudiante graduado con grado: " << grado << endl;
    }

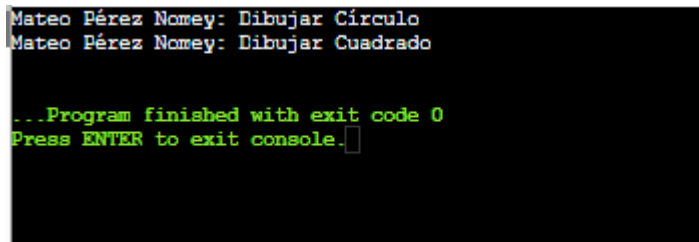
    ~EstudianteGraduado() {
        cout << "Graduado destruido: " << nombre << endl;
    }

    void investigar() const {
        cout << nombre << " está investigando como parte de su " << grado << "." << endl;
    }
};

int main() {
    EstudianteGraduado eg("Mateo Pérez Nomey", "Ingeniería", "Maestría");
    eg.estudiar();
    eg.investigar();
    return 0;
}

```

Figura



```

Mateo Pérez Nomey: Dibujar Círculo
Mateo Pérez Nomey: Dibujar Cuadrado

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

```

```

#include <iostream>
using namespace std;

// Clase base
class Figura {
public:
    // Método virtual para habilitar el despacho dinámico
    virtual void dibujar() const {
        cout << "Mateo Pérez Nomey: Dibujar Figura genérica" << endl;
    }

    // Es buena práctica tener un destructor virtual
    virtual ~Figura() {}
};

// Clase derivada

```

```

class Circulo : public Figura {
public:
    // Redefinición del método virtual
    void dibujar() const override {
        cout << "Mateo Pérez Nomey: Dibujar Círculo" << endl;
    }
};

// Clase derivada adicional para contraste
class Cuadrado : public Figura {
public:
    void dibujar() const override {
        cout << "Mateo Pérez Nomey: Dibujar Cuadrado" << endl;
    }
};

int main() {
    // Puntero de tipo base apuntando a objeto derivado
    Figura* figura1 = new Circulo();
    Figura* figura2 = new Cuadrado();

    // Invocación del método virtual: se ejecuta dinámicamente
    figura1->dibujar(); // Salida: "Mateo Pérez Nomey: Dibujar Círculo"
    figura2->dibujar(); // Salida: "Mateo Pérez Nomey: Dibujar Cuadrado"

    // Liberar memoria
    delete figura1;
    delete figura2;

    return 0;
}

```

Forma Geometrica

```
Mateo Pérez Nomey - CONSTRUCTOR FormaGeometrica: Mi Circulo (Rojo)
Mateo Pérez Nomey - CONSTRUCTOR FormaGeometrica: Mi Rectangulo (Azul)

Mateo Pérez Nomey --- Descripcion de Forma ---
Mateo Pérez Nomey - Mi Circulo (Rojo): Dibujando CIRCULO de radio 7.
Area: 153.938
Color: Rojo

Mateo Pérez Nomey --- Descripcion de Forma ---
Mateo Pérez Nomey - Mi Rectangulo (Azul): Dibujando RECTANGULO base 4, altura 5.
Area: 20
Color: Azul

Mateo Pérez Nomey - CONSTRUCTOR FormaGeometrica: Otro Circulo (Verde)

Mateo Pérez Nomey --- Procesando todas las formas del vector ---
Mateo Pérez Nomey - Mi Circulo (Rojo): Dibujando CIRCULO de radio 7.
Mateo Pérez Nomey - Mi Rectangulo (Azul): Dibujando RECTANGULO base 4, altura 5.
Mateo Pérez Nomey - Otro Circulo (Verde): Dibujando CIRCULO de radio 3.

Mateo Pérez Nomey --- Liberando memoria ---
Mateo Pérez Nomey - DESTRUCTOR Circulo: Otro Circulo
Mateo Pérez Nomey - DESTRUCTOR FormaGeometrica: Otro Circulo
Mateo Pérez Nomey - DESTRUCTOR Rectangulo: Mi Rectangulo
Mateo Pérez Nomey - DESTRUCTOR FormaGeometrica: Mi Rectangulo
Mateo Pérez Nomey - DESTRUCTOR Circulo: Mi Circulo
Mateo Pérez Nomey - DESTRUCTOR FormaGeometrica: Mi Circulo

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

```
#include <iostream>
#include <string>
#include <vector>
#include <cmath>
// Clase base abstracta
class FormaGeometrica {
protected:
    std::string nombreFigura;
    std::string color;
public:
    FormaGeometrica(std::string nf, std::string c) : nombreFigura(nf), color(c) {
        std::cout << "Mateo Pérez Nomey - CONSTRUCTOR FormaGeometrica: " <<
nombreFigura << " (" << color << ")" << std::endl;
    }
    virtual void dibujar() const = 0; // Método puro
    virtual double calcularArea() const = 0; // Método puro
    std::string getColor() const { return color; }
    std::string getNombre() const { return nombreFigura; }
    virtual ~FormaGeometrica() {
        std::cout << "Mateo Pérez Nomey - DESTRUCTOR FormaGeometrica: " <<
nombreFigura << std::endl;
    }
};
// Clase derivada: Circulo
class Circulo : public FormaGeometrica {
private:
    double radio;
```

```

public:
    Circulo(std::string nf, std::string c, double r) : FormaGeometrica(nf, c), radio(r) {}

    void dibujar() const override {
        std::cout << "Mateo Pérez Nomey - " << getNombre() << " (" << getColor() << "):
Dibujando CIRCULO de radio "
        << radio << "." << std::endl;
    }
    double calcularArea() const override {
        return 3.14159 * radio * radio;
    }

    ~Circulo() override {
        std::cout << "Mateo Pérez Nomey - DESTRUCTOR Circulo: " << getNombre() <<
std::endl;
    }
};

// Clase derivada: Rectangulo
class Rectangulo : public FormaGeometrica {
private:
    double base, altura;
public:
    Rectangulo(std::string nf, std::string c, double b, double h) : FormaGeometrica(nf, c),
base(b), altura(h) {}

    void dibujar() const override {
        std::cout << "Mateo Pérez Nomey - " << getNombre() << " (" << getColor() << "):
Dibujando RECTANGULO base "
        << base << ", altura " << altura << "." << std::endl;
    }
    double calcularArea() const override {
        return base * altura;
    }
    ~Rectangulo() override {
        std::cout << "Mateo Pérez Nomey - DESTRUCTOR Rectangulo: " << getNombre() <<
std::endl;
    }
};

// Función para describir cualquier forma
void describirForma(const FormaGeometrica* forma) {
    std::cout << "\nMateo Pérez Nomey --- Descripcion de Forma ---" << std::endl;
    forma->dibujar();
    std::cout << " Area: " << forma->calcularArea() << std::endl;
    std::cout << " Color: " << forma->getColor() << std::endl;
}

int main() {
    FormaGeometrica* ptrCirculo = new Circulo("Mi Circulo", "Rojo", 7.0);
    FormaGeometrica* ptrRectangulo = new Rectangulo("Mi Rectangulo", "Azul", 4.0, 5.0);

```

```

    describirForma(ptrCirculo);
    describirForma(ptrRectangulo);
    std::vector<FormaGeometrica*> misFormas;
    misFormas.push_back(ptrCirculo);
    misFormas.push_back(ptrRectangulo);
    misFormas.push_back(new Circulo("Otro Circulo", "Verde", 3.0));
    std::cout << "\nMateo Pérez Nomey --- Procesando todas las formas del vector ---" <<
std::endl;
    for (const FormaGeometrica* f : misFormas) {
        f->dibujar();
    }
    std::cout << "\nMateo Pérez Nomey --- Liberando memoria ---" << std::endl;
    delete misFormas[2]; // Borra "Otro Circulo"
    delete ptrRectangulo;
    delete ptrCirculo;
    return 0;
}

```

Herencia animal

```

--- Creando un Animal genérico ---
CONSTRUCTOR Animal: Nace un animal llamado 'Alex el León' de 10 anio(s).
Alex el León está comiendo.

--- Creando un Perro llamado Mateo Pérez Nomey ---
CONSTRUCTOR Animal: Nace un animal llamado 'Mateo Pérez Nomey' de 3 anio(s).
CONSTRUCTOR Perro: Nace un perro de raza 'Golden Retriever', que también es un animal
llamado 'Mateo Pérez Nomey'.
Mateo Pérez Nomey está comiendo.
Mateo Pérez Nomey está durmiendo.
El nombre de mi mascota es: Mateo Pérez Nomey
Mateo Pérez Nomey (un Golden Retriever) dice: ¡Guau! ¡Guau!

--- Fin de main (los objetos se destruirán) ---
DESTRUCTOR Perro: Muere el perro 'Mateo Pérez Nomey' de raza 'Golden Retriever'.
DESTRUCTOR Animal: Muere el animal 'Mateo Pérez Nomey'.
DESTRUCTOR Animal: Muere el animal 'Alex el León'.

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

```

```
#include <iostream>
```

```
#include <string>
```

```
// Clase Base
```

```
class Animal {
```

```
protected:
```

```
    std::string nombre;
```

```
    int edad;
```

```
public:
```

```
    Animal(const std::string& n, int e) : nombre(n), edad(e) {
```

```
        std::cout << "CONSTRUCTOR Animal: Nace un animal llamado " << nombre
        << " de " << edad << " anio(s)." << std::endl;
```



```

}
~Animal() {
    std::cout << "DESTRUCTOR Animal: Muere el animal " << nombre << "." << std::endl;
}
void comer() const {
    std::cout << nombre << " está comiendo." << std::endl;
}
void dormir() const {
    std::cout << nombre << " está durmiendo." << std::endl;
}
std::string getNombre() const { return nombre; }
};

```

// Clase Derivada

```
class Perro : public Animal {
```

```
private:
```

```
    std::string raza;
```

```
public:
```

```
    Perro(const std::string& n, int e, const std::string& r)
```

```
        : Animal(n, e), raza(r) {
```

```
            std::cout << "CONSTRUCTOR Perro: Nace un perro de raza " << raza
                << ", que también es un animal llamado " << nombre << "." << std::endl;
        }
```

```
    ~Perro() {
```

```
        std::cout << "DESTRUCTOR Perro: Muere el perro " << nombre << " de raza " <<
            raza << "." << std::endl;
    }
```

```
    void ladrar() const {
```

```
        std::cout << getNombre() << " (un " << raza << ") dice: ¡Guau! ¡Guau!" << std::endl;
    }
```

```
};
```

```
int main() {
```

```
    std::cout << "--- Creando un Animal genérico ---" << std::endl;
```

```
    Animal animalGenerico("Alex el León", 10);
```

```
    animalGenerico.comer();
```

```
    std::cout << "\n--- Creando un Perro llamado Mateo Pérez Nomey ---" << std::endl;
```

```
    Perro miMascota("Mateo Pérez Nomey", 3, "Golden Retriever");
```

```
    miMascota.comer();
```

```
    miMascota.dormir();
```

```
    std::cout << "El nombre de mi mascota es: " << miMascota.getNombre() << std::endl;
```

```
    miMascota.ladrar();
```

```
    std::cout << "\n--- Fin de main (los objetos se destruirán) ---" << std::endl;
```

```
    return 0;  
}
```