

MANUAL DE MALAS PRÁCTICAS Y SUS SOLUCIONES

1. Uso excesivo de variables globales

Solución: Utilizar variables locales siempre que sea posible y pasar datos como parámetros a funciones cuando sea necesario.

```
// Mala práctica
public class MyClass
{
    private int globalVariable = 10;

    public void DoSomething()
    {
        // ... usa globalVariable ...
    }
}

// Buena práctica
public class MyClass
{
    public void DoSomething()
    {
        int localVariable = 10;
        // ... usa localVariable ...
    }
}
```

2. Métodos demasiado largos y complejos


Solución: Dividir métodos largos en métodos más pequeños y específicos.

```
// Mala práctica
public void ProcessData()
{
    // ... muchas líneas de código ...
}

// Buena práctica
public void ProcessData()
{
    StepOne();
    StepTwo();
    // ...
}

private void StepOne()
{
    // ...
}

private void StepTwo()
{
    // ...
}
```



3.No manejar excepciones de manera adecuada

Solución: Utilizar bloques “try-catch” para capturar y manejar excepciones de forma apropiada.

```
// Mala práctica
public void DoSomething()
{
    try
    {
        // ... código propenso a errores ...
    }
    catch
    {
        // No se maneja la excepción de manera adecuada
    }
}

// Buena práctica
public void DoSomething()
{
    try
    {
        // ... código propenso a errores ...
    }
    catch (Exception ex)
    {
        // Manejar la excepción apropiadamente
        Console.WriteLine($"Ocurrió un error: {ex.Message}");
    }
}
```

4.No utilizar comentarios descriptivos o mantenerlos desactualizados

Solución: Escribir comentarios claros y actualizados que expliquen la lógica o el propósito del código.

```
// Mala práctica
// Método para procesar datos
public void ProcessData()
{
    // ... código ...
}

// Buena práctica
// Calcula la suma de los elementos en la lista
public void CalculateSum(List<int> numbers)
{
    // ... código ...
}
```

5.No utilizar constantes o enums para valores fijos

Solución: Utilizar constantes o enumeraciones para valores que no cambian.

```
// Mala práctica
public double CalculateTax(double income)
{
    return income * 0.15; // 0.15 es un valor mágico
}

// Buena práctica
private const double TaxRate = 0.15;

public double CalculateTax(double income)
{
    return income * TaxRate;
}
```

6.No seguir convenciones de nombres

Solución: Utilizar convenciones de nombres claras y significativas para variables, métodos y clases.

```
// Mala práctica
int x = 5;

// Buena práctica
int numberOfStudents = 5;
```

7.No cerrar recursos adecuadamente

Solución: Utilizar bloques “using” para asegurarse de que los recursos se liberen adecuadamente.

```
// Mala práctica
FileStream file = new FileStream("archivo.txt", FileMode.Open);
// ... código ...
file.Close(); // Podría no cerrarse en caso de excepción

// Buena práctica
using (FileStream file = new FileStream("archivo.txt", FileMode.Open))
{
    // ... código ...
} // El recurso se cerrará automáticamente al salir del bloque using
```