

PRACTICA NUMERO 1.

Etudiante: Ruben Dario Higuaita.

Nombre del proyecto: Sistema Básico de Verificación de Telemetría del Vehículo.

He diseñado una pequeña aplicación de consola para simular la recepción de los datos de dos sensores muy críticos dentro de la inyección electrónica del motor:

- Temperatura del refrigerante del motor.
- Presión del aceite del motor.

He establecido unos parámetros seguros y debo comprobar los datos o valores que recibo de las señales que llegan del motor al sistema creado y compararlos con mi tabla predefinida con los valores seguros de que deben estar actuando dentro del motor. Una vez recibido los datos el programa emitirá una señal de alerta si los valores recibidos están fuera de rango, ó enviaremos un mensaje de “OK” si ambos están dentro de los límites seguros.

Rangos seguros:

- Temperatura refrigerante:
Mínimo 80°C --- Máximo 105°C
- Presión de aceite:
Mínimo 25psi --- Máximo 65psi

Diseño del programa:

Tipos de datos utilizados:

- **double**: los uso para valores con decimales (temperatura y presión).
- **const double**: este tipo de dato lo utilicé para valores fijos.
- **bool**: este tipo de dato booleano lo uso como flag del estado de alerta.
- **string**: lo uso para los mensajes de salida.

Estructuras de Control:

- **if-else** : Las he utilizado para evaluar las condiciones de cada sensor.
- **while** : Para realizar un bucle y simular múltiples lecturas (Bucle de telemetría).
- **Operadores lógicos** : **&&**, **|**.

Flujo del Programa:

1. Definición rangos seguros como constantes
2. Iniciar bucle de simulación
3. Pedir al vehículo o usuario los valores de sensores.
4. Validar cada valor.
5. Mostrar estado general: OK ó ALERTA.
6. Preguntar si desea Continuar.

Lenguaje : C#

IDE: Visual Studio Code

- Los retos que tuve para la programación en el IDE, ya que como sistema operativo estoy utilizando Ubuntu,. Utilicé el visual studio code, necesité el SDK .NET, el paquete de desarrollo que contiene el compilador, las librerías y el comando **dotnet**. Esto construye y ejecuta el código.

- El C# Dev kit para la sintaxis del código, también me ayuda para el auto completado.

- Estructura general del programa:

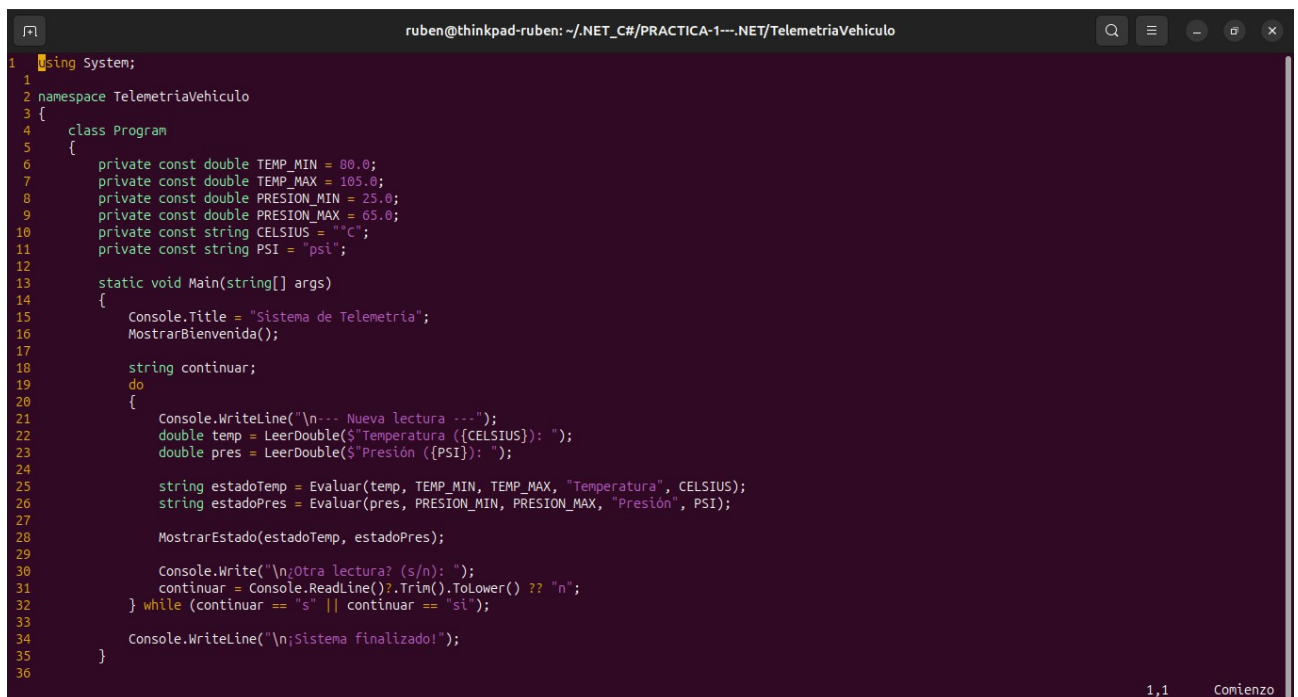
using System : Para Console, librería.
namespace TelemetriaVehiculo: Organizacion del codigo.
Class program: Modulo o plantilla principal del programa.

- Campos de clase:

const: Valores constantes que no cambian. Valores que utilizo para los limites de los rangos correctos.
private: Es encapsulamiento, solo el programa interno necesita saber los limites que he puesto anteriormente.
Double: usamos valores decimales, que son los valores que tomaremos de los sensores del motor.

- Método main:

static void main: Es el primer método que C# busca para iniciar el programa.
Console.Title: Detalle estetico.
MostrarBienvenida: Llama a la funcion externa para limpiar el codigo principal. Solo por buena practica.
Do: Dentro del bucle llamamos herramientas especializadas como (LeerDouble, Evaluar, MostrarEstado).



```
1 using System;
2 namespace TelemetriaVehiculo
3 {
4     class Program
5     {
6         private const double TEMP_MIN = 80.0;
7         private const double TEMP_MAX = 105.0;
8         private const double PRESION_MIN = 25.0;
9         private const double PRESION_MAX = 65.0;
10        private const string CELSIUS = "°C";
11        private const string PSI = "psi";
12
13        static void Main(string[] args)
14        {
15            Console.Title = "Sistema de Telemetria";
16            MostrarBienvenida();
17
18            string continuar;
19            do
20            {
21                Console.WriteLine("\n--- Nueva lectura ---");
22                double temp = LeerDouble($"Temperatura ({CELSIUS}): ");
23                double pres = LeerDouble($"Presión ({PSI}): ");
24
25                string estadoTemp = Evaluar(temp, TEMP_MIN, TEMP_MAX, "Temperatura", CELSIUS);
26                string estadoPres = Evaluar(pres, PRESION_MIN, PRESION_MAX, "Presión", PSI);
27
28                MostrarEstado(estadoTemp, estadoPres);
29
30                Console.WriteLine("\nOtra lectura? (s/n): ");
31                continuar = Console.ReadLine()?.Trim().ToLower() ?? "n";
32            } while (continuar == "s" || continuar == "si");
33
34            Console.WriteLine("\nSistema finalizado!");
35        }
36    }
37 }
```

1,1 Comienzo

- Método de ayuda:

MostrarBienvenida(): Su proposito es saldar e informar al usuario.

LeerDouble(): Recibe el parametro del Usuario.

Evaluar(): Comprueba si los Valores entran dentro de los parametros correctos.

MostrarEstado(): Damos el resultado de la alerta o el OK, del programa.

Console.ResetColor(): Limpia despues de cambiar el color de la salida del mensaje.

```
ruben@thinkpad-ruben: ~/.NET_C#/PRACTICA-1---.NET/TelemetriaVehiculo

36
35 static void MostrarBienvenida()
34 {
33     Console.WriteLine("=== SISTEMA DE TELEMETRÍA DEL VEHÍCULO ===");
32     Console.WriteLine($"Rangos seguros:");
31     Console.WriteLine($"  • Temperatura: {TEMP_MIN}{CELSIUS} - {TEMP_MAX}{CELSIUS}");
30     Console.WriteLine($"  • Presión: {PRESION_MIN} {PSI} - {PRESION_MAX} {PSI}\n");
29 }
28
27 static double LeerDouble(string msg)
26 {
25     while (true)
24     {
23         Console.Write(msg);
22         if (double.TryParse(Console.ReadLine(), out double v)) return v;
21         Console.WriteLine("Error: número inválido.");
20     }
19 }
18
17 static string Evaluar(double valor, double min, double max, string nombre, string unidad)
16 {
15     if (valor < min) return $"[ALERTA] {nombre} BAJA: {valor}{unidad} (mín: {min}{unidad})";
14     if (valor > max) return $"[ALERTA] {nombre} ALTA: {valor}{unidad} (máx: {max}{unidad})";
13     return $"[OK] {nombre}: {valor}{unidad}";
12 }
11
10 static void MostrarEstado(string t, string p)
9 {
8     Console.WriteLine("\n--- ESTADO ---");
7     Console.WriteLine(t);
6     Console.WriteLine(p);
5     bool alerta = t.Contains("ALERTA") || p.Contains("ALERTA");
4     Console.ForegroundColor = alerta ? ConsoleColor.Red : ConsoleColor.Green;
3     Console.WriteLine(alerta ? "\nALERTA - Revisar vehiculo" : "\nOK - Todo en orden");
2     Console.ResetColor();
1 }
73 }
```

- Prueba de Compilacion del Programa:

```
Archivo  Editar  Selección  Ver  Ir  ...  TelemetriaVehiculo

EXPLORADOR
  EDITORES ABIERTOS
    Program.cs
  TELEMETRIAVEHICULO
    bin
    obj
    Program.cs
  TelemetriaVehiculo.csproj
  TelemetriaVehiculo.sln

MAKEFILE: ESQUEMA DEL PROYECTO
  Configuración: [Anular]
  Destino de compilación: [Anular]
  Destino de inicio: [Anular]
  Makefile (no encontrado): [/home/ruben/...
  Make: [/usr/bin/make]
  Registro de la compilación: [Anular]

ESQUEMA
  LINEA DE TIEMPO
  NORMINETTE TREE
  EXPLORADOR DE SOLUCIONES

Program.cs x
5 class Program
12 private const string PSI = "psi";

PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  GITLENS
dotnet

ruben@thinkpad-ruben:~/C#/.NET/PRACTICA-1---.NET/TelemetriaVehiculo$ dotnet run
=== SISTEMA DE TELEMETRÍA DEL VEHÍCULO ===
Rangos seguros:
  • Temperatura: 80°C - 105°C
  • Presión: 25 psi - 65 psi

--- Nueva lectura ---
Temperatura (°C): 110
Presión (psi): 30

--- ESTADO ---
[ALERTA] Temperatura ALTA: 110°C (máx: 105°C)
[OK] Presión: 30psi

ALERTA - Revisar vehiculo

¿otra lectura? (s/n): s

--- Nueva lectura ---
Temperatura (°C): 100
Presión (psi): 60

--- ESTADO ---
[OK] Temperatura: 100°C
[OK] Presión: 60psi

OK - Todo en orden

¿otra lectura? (s/n):
```