**1.4.1 - Manejo de Memoria Estática.**

**Cuál es la memoria estática**

Es la que no se puede modificar o ejecutar en momento de ejecución. También se debe conocer con anticipo el tamaño de la estructura que está en ejecución por Ej. Un vector, matriz, cubo etc. Estos ejemplos que con anticipación se deben conocer el tamaño de la estructura. Algunos lenguajes de programación utilizan la palabra static para especificar elementos del programa que deben almacenarse en memoria estática.

Los objetos son creados en ese momento y destruidos al finalizar el programa. Mantienen la misma localización en memoria durante todo el transcurso del programa.

**Quienes la utilizan y manejan en C#**

* Código del programa.
* Las variables definidas en la sección principal del programa, las cuales pueden solo cambiar su contenido no su tamaño.
* Todas aquellas variables declaradas como estáticas en otras clases o módulos.

Los objetos administrados de ese modo son:

1. Variables static
2. Variables globales
3. Miembros static de clases
4. Literales de cualquier tipo

**Ventajas y Desventajas del uso de memoria estática.**

Ventajas

* La velocidad de acceso es alta.
* Para retener los datos solo necesita estar energizada.
* Lógica simple.
* Son más fáciles de diseñar.

Desventajas:

* No se puede modificar el tamaño de la estructura en tiempo de ejecución.
* No es óptimo con grandes cantidades de datos.
* Desperdicio de memoria cuando no se utiliza en su totalidad  del tamaño  v[100]  .
* Menor capacidad, debido a que cada celda de almacenamiento requiere más transistores.
* Mayor costo por bit.
* Mayor consumo de Potencia.

**Programa ejemplo en C# que utilice memoria estática.**

using System;

using System.Collections.Generic;

using System.Text;

public static class ConvertirTemperatura

{

public static double CelsiusToFahrenheit(string TemperaturaCelsius)

{

double celsius = Double.Parse(TemperaturaCelsius);

double fahrenheit = (celsius \* 9 / 5) + 32;

return fahrenheit;

}

public static double FahrenheitToCelsius(string TemperaturaFahrenheit)

{

double fahrenheit = Double.Parse(TemperaturaFahrenheit);

double celsius = (fahrenheit - 32) \* 5 / 9;

return celsius;

}

}

class TestTemperatureConverter

{

static void Main()

{

Console.WriteLine("Convertidor de temperaturas");

Console.WriteLine("1. De Celsius a Fahrenheit.");

Console.WriteLine("2. De Fahrenheit a Celsius.");

Console.Write("Seleccione una opcion:");

string selection = Console.ReadLine();

double F, C = 0;

switch (selection)

{

case "1":

Console.Write("Ingrese la temperatura en Celsius: ");

F = ConvertirTemperatura.CelsiusToFahrenheit(Console.ReadLine());

Console.WriteLine("Temperatura en Fahrenheit: {0:F2}", F);

break;

case "2":

Console.Write("Ingrese la temperatura en Fahrenheit: ");

C = ConvertirTemperatura.FahrenheitToCelsius(Console.ReadLine());

Console.WriteLine("Temperatura en Celsius: {0:F2}", C);

break;

default:

Console.WriteLine("Porfavor seleccione el convertidor");

break;

}

Console.WriteLine("Presione cualquier letra para salir.");

Console.ReadKey();

}

}

<https://es.slideshare.net/Tell123Say123/memoria-estatica#:~:text=La%20memoria%20est%C3%A1tica%20es%20la%20que%20no%20se%20puede%20modificar,el%20tama%C3%B1o%20de%20la%20estructura>.

<http://apuntesestructuradedatos.blogspot.com/2009/03/unidad-ii-manejo-de-memoria.html#:~:text=12%20de%20din%C3%A1mica.&text=La%20memoria%20est%C3%A1tica%20es%20la,todo%20el%20transcurso%20del%20programa>.

<https://docs.microsoft.com/es-es/dotnet/csharp/programming-guide/classes-and-structs/static-classes-and-static-class-members>