



Practica para la materia:

## ***Programación orientada a objetos***

Presentada por:

***Arteaga Castañeda Luis Eduardo***

Carrera:

***Ingeniería en Sistemas  
Computacionales***



Unidad Sabatina

# ÍNDICE

*Capítulo 1. Introducción*

*Capítulo 2. Descripción general del código*

*Capítulo 3. Explicación del código*

- 3.1 Creación del arreglo*
- 3.2 Modificación del arreglo*
- 3.3 Suma de elementos*
- 3.4 Búsqueda en un arreglo*

*Capítulo 4. Pruebas del programa*

*Capítulo 5. Conclusión*

## 1. Introducción

Este documento tiene como objetivo proporcionar una explicación técnica y detallada sobre el funcionamiento de un programa en C#, desarrollado como parte de la práctica correspondiente a la materia de Programación Orientada a Objetos (POO). El programa está diseñado para manipular un arreglo de números enteros y realizar diversas operaciones sobre él. Las principales funcionalidades del código incluyen:

- **Creación e inicialización de un arreglo:** Se crea un arreglo de enteros de tamaño fijo (10 elementos), inicializándolo con los valores enteros del 1 al 10.
- **Modificación de un elemento específico:** Se permite al usuario ingresar un nuevo valor para reemplazar el tercer elemento del arreglo, que corresponde al índice 2.
- **Cálculo de la suma de los elementos:** Se implementa un ciclo que recorre el arreglo y suma todos sus elementos, lo que permite al programa realizar un procesamiento básico de los datos y obtener el resultado de dicha operación.
- **Búsqueda de un número en el arreglo:** A través de la funcionalidad de búsqueda, el programa permite al usuario ingresar un número que desea localizar dentro del arreglo. Si el número está presente, el programa devuelve el índice correspondiente del arreglo; en caso contrario, informa que el elemento no fue encontrado.

El propósito del código es no solo permitir la interacción del usuario con un arreglo, sino también proporcionar una comprensión práctica de cómo manejar estructuras de datos en C#, interactuar con el usuario, realizar validaciones de entrada y efectuar operaciones matemáticas o de búsqueda básicas.

## 2. Descripción General del Código

El programa está desarrollado en **C** este código en particular está enfocado en trabajar con estructuras de datos simples, específicamente un arreglo unidimensional de enteros, y demuestra cómo interactuar con dichos arreglos en tiempo de ejecución.

A través de la interacción directa con el usuario, el código permite realizar diversas operaciones fundamentales que se requieren en muchos algoritmos y aplicaciones prácticas:

- **Modificación dinámica de datos:** El usuario tiene la capacidad de cambiar el valor de un elemento específico en el arreglo, lo que muestra cómo gestionar la entrada y salida de datos dentro de un programa. Esta

modificación permite demostrar cómo funcionan las referencias a índices dentro de un arreglo.

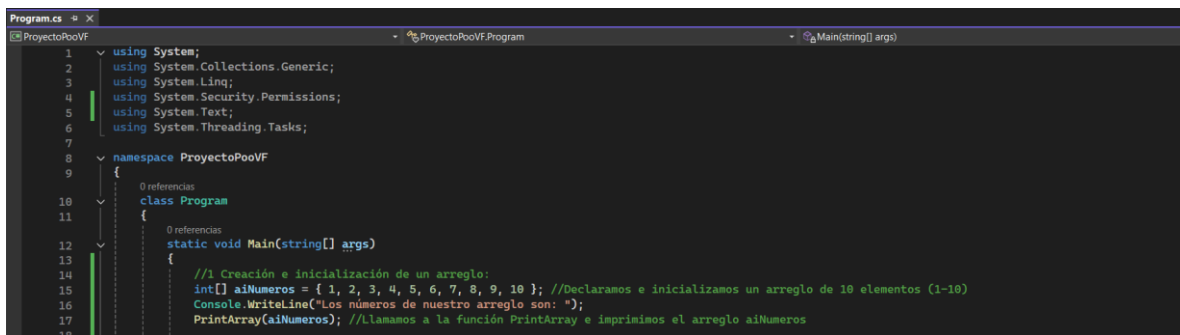
- **Operaciones aritméticas:** El cálculo de la suma de los elementos del arreglo ilustra el uso de ciclos y acumuladores en programación. Este tipo de operaciones es uno de los aspectos más comunes en el procesamiento de datos.
- **Búsqueda de elementos en una estructura de datos:** Mediante el uso del método `Array.IndexOf()`, el código permite realizar una búsqueda de un número dentro del arreglo. Este método de búsqueda secuencial es un ejemplo básico de cómo encontrar elementos en colecciones no ordenadas.

El código también implementa una **validación robusta de entrada**, asegurándose de que las entradas proporcionadas por el usuario sean correctas antes de intentar utilizarlas, lo que mejora la estabilidad y fiabilidad del programa. Esta validación es un componente esencial en cualquier programa interactivo, ya que evita errores y asegura que los datos procesados sean válidos.

### 3. Explicación del código

#### 3.1 Creación del arreglo

A continuación, se explicará el funcionamiento de cada sección del código, detallando su estructura y propósito.



```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Security.Permissions;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace ProyectoPooVF
9 {
10     0 referencias
11     class Program
12     {
13         0 referencias
14         static void Main(string[] args)
15         {
16             // Creación e inicialización de un arreglo:
17             int[] aiNumeros = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }; //Declaramos e inicializamos un arreglo de 10 elementos (1-10)
18             Console.WriteLine("Los números de nuestro arreglo son: ");
19             PrintArray(aiNumeros); //Llamamos a la función PrintArray e imprimimos el arreglo aiNumeros
20         }
21     }
22 }
```

Iniciamos el código declarando nuestro arreglo de tipo entero con el nombre **aiNumeros**, en el cual almacenaremos 10 elementos (1-10), posteriormente pediremos que se imprima el arreglo en consola utilizando como apoyo un método auxiliar al que definiremos como **PrintArray**.

### Declaración del método auxiliar

¿Para qué nos sirve PrintArray? Nuestro método auxiliar nos va a ayudar a recibir nuestro arreglo aiNumeros y posteriormente lo imprimirá en consola, mostrando cada elemento, se utiliza un ciclo For para recorrer el arreglo.

A pesar de que podríamos simplemente un ciclo For y ahorrarnos la declaración del método, en términos de eficiencia resulta ser mejor, debido a que no será la única vez que necesitemos recorrer el arreglo, por ende, si no tuviéramos el PrintArray, tendríamos que colocar un For cada vez que necesitemos imprimir el Array, con el método, solo necesitaremos mandarlo a llamar y definir que arreglo debe recorrer.

```
2 referencias
static void PrintArray(int[] arr) ///Función auxiliar que imprime nuestro arreglo
{
    for (int i = 0; i < arr.Length; i++)//Recorremos el arreglo y lo imprimimos
    {
        Console.Write(arr[i] + " "); //Imprimimos el elemento en la posición i
    }
    Console.WriteLine();//Imprimimos un salto de línea
}
```

### 3.2 Modificación del arreglo

Una vez que tenemos creado nuestro arreglo, se le solicitara al usuario que ingrese un número que remplace el tercer elemento (índice 2) del arreglo.

```
//2 Modificación de un elemento
Console.WriteLine("Ingresa un número para modificar el tercer elemento:");//Pedimos al usuario que ingrese un número que modifique el índice 2
int iValor;
while (!int.TryParse(Console.ReadLine(), out iValor))
{
    //Validamos que el valor ingresado sea un número
    Console.WriteLine("Por favor, ingresa un número válido: ");
}
aiNumeros[2] = iValor; //Modificamos el tercer elemento del arreglo con el valor ingresado por el usuario
Console.WriteLine("Arreglo actualizado");
PrintArray(aiNumeros); //Imprimimos el arreglo actualizado
```

Se colocó un bucle While que nos permita crear una restricción en caso de que el usuario coloque un elemento que no sea válido, (elemento con el tipo Int), en caso de que el elemento sea invalido se le indicara al usuario y solicitara nuevamente un elemento de tipo Int.

Una vez que se coloque un elemento correcto, el Array será modificado y posteriormente se imprimirá gracias al método PrintArray.

### 3.3 Suma de elementos

En este punto el programa se encargará de recorrer el arreglo y sumar todos los elementos que se encuentren dentro de él, como se puede apreciar en la siguiente imagen.

```
//3 Suma de los elementos del arreglo:
int iSuma = 0; //Declaramos una variable para almacenar la suma de los elementos del arreglo
for (int i = 0; i < aiNumeros.Length; i++) //Recorremos el arreglo y sumamos cada uno de sus elementos
{
    iSuma += aiNumeros[i];
}
Console.WriteLine("La suma de los elementos del arreglo es: " + iSuma); //Imprimimos la suma de los elementos del arreglo
```

Declararemos una variable iSuma que será inicializada en 0, esta misma nos servirá como contador, en el ciclo For pediremos que recorra cada elemento del arreglo y los vaya sumando, estos mismos serán almacenados en la variable iSuma que al final terminara guardando el resultado y posteriormente lo mostrara en consola.

### 3.4 Búsqueda en un arreglo

Para el 4to punto se le solicitara al usuario que ingrese un numero a elección para buscar su posición en el array.

```
//4 Búsqueda de un número en el arreglo:
Console.WriteLine("Ingresa un número para que lo busquemos en el Array: ");
int iBusqueda;
while (!int.TryParse(Console.ReadLine(), out iBusqueda)) { //Validamos que el valor ingresado sea un número
    Console.WriteLine("Por favor, ingresa un número válido: ");
}
int iPosicion = Array.IndexOf(aiNumeros, iBusqueda); //Buscaremos el número ingresado por el usuario en el arreglo
if (iPosicion != -1)
{
    Console.WriteLine("El número " + iBusqueda + " se encuentra en la posición " + iPosicion + " del arreglo.");
}
else{
    Console.WriteLine("El número " + iBusqueda + " no se encuentra en el arreglo.");
}
//Si el número se encuentra en el arreglo, imprimimos su posición, pero si no esta en el arreglo, imprimimos un mensaje que nos lo indique
```

Mediante un bucle While verificaremos que el dato ingresado sea válido(elemento de tipo Int), en caso de que el usuario digite un elemento que no lo sea, el programa le solicitara que intente de nuevo, cuando el usuario digite un entero, se buscara el índice, mediante una condicional If-Else, se indicara la posición en el arreglo, de lo contrario, si el numero no está, la consola indicara que dicho numero no existe dentro de aiNumeros.

```
0 referencias
static void Main(string[] args)
{
    //1 Creación e inicialización de un arreglo:
    int[] aiNumeros = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }; //Declaramos e inicializamos un arreglo de 10 elementos (1-10)
    Console.WriteLine("Los números de nuestro arreglo son: ");
    PrintArray(aiNumeros); //Llamamos a la función PrintArray e imprimimos el arreglo aiNumeros

    //2 Modificación de un elemento
    Console.WriteLine("Ingresa un número para modificar el tercer elemento: "); //Pedimos al usuario que ingrese un número que modifique el índice 2
    int iValor;
    while (!int.TryParse(Console.ReadLine(), out iValor))
    { //Validamos que el valor ingresado sea un número
        Console.WriteLine("Por favor, ingresa un número válido: ");
    }
    aiNumeros[2] = iValor; //Modificamos el tercer elemento del arreglo con el valor ingresado por el usuario
    Console.WriteLine("Arreglo actualizado");
    PrintArray(aiNumeros); //Imprimimos el arreglo actualizado

    //3 Suma de los elementos del arreglo:
    int iSuma = 0; //Declaramos una variable para almacenar la suma de los elementos del arreglo
    for (int i = 0; i < aiNumeros.Length; i++) //Recorremos el arreglo y sumamos cada uno de sus elementos
    {
        iSuma += aiNumeros[i];
    }
    Console.WriteLine("La suma de los elementos del arreglo es: " + iSuma); //Imprimimos la suma de los elementos del arreglo

    //4 Búsqueda de un número en el arreglo:
    Console.WriteLine("Ingresa un número para que lo busquemos en el Array: ");
    int iBusqueda;
    while (!int.TryParse(Console.ReadLine(), out iBusqueda)) { //Validamos que el valor ingresado sea un número
        Console.WriteLine("Por favor, ingresa un número válido: ");
    }
    int iPosicion = Array.IndexOf(aiNumeros, iBusqueda); //Buscaremos el número ingresado por el usuario en el arreglo
    if (iPosicion != -1)
    {
        Console.WriteLine("El número " + iBusqueda + " se encuentra en la posición " + iPosicion + " del arreglo.");
    }
    else{
        Console.WriteLine("El número " + iBusqueda + " no se encuentra en el arreglo.");
    }
    //Si el número se encuentra en el arreglo, imprimimos su posición, pero si no esta en el arreglo, imprimimos un mensaje que nos lo indique
}

2 referencias
static void PrintArray(int[] arr) //Función auxiliar que imprime nuestro arreglo
{
    for (int i = 0; i < arr.Length; i++) //Recorremos el arreglo y lo imprimimos
    {
        Console.Write(arr[i] + " "); //Imprimimos el elemento en la posición i
    }
    Console.WriteLine(); //Imprimimos un salto de línea
}
```

#### 4. Pruebas del programa

A continuación, veremos una serie de casos en las que mostraremos el funcionamiento del programa.

```
D:\ProyectoPooVF\ProyectoPi x + v
Los números de nuestro arreglo son:
1 2 3 4 5 6 7 8 9 10
Ingresa un número para modificar el tercer elemento:
|
```

Caso 1: Introducir valores invalido para comprobar la primera restricción:

```
D:\ProyectoPooVF\ProyectoPi x + v
Los números de nuestro arreglo son:
1 2 3 4 5 6 7 8 9 10
Ingresa un número para modificar el tercer elemento:
q
Por favor, ingresa un número válido:
3.5
Por favor, ingresa un número válido:
.45
Por favor, ingresa un número válido:
45
Arreglo actualizado
```

Al probar con datos de tipo char y decimales no nos deja continuar, al momento que se coloca el tipo de dato entero, nos envía mensaje de que el array fue actualizado.

Caso 2: Impresión del arreglo actualizado

```
Arreglo actualizado
1 2 45 4 5 6 7 8 9 10
```

Como se puede apreciar en este caso, anterior mente modificamos el tercer elemento por el número 45, en el arreglo actualizado ya nos retorna el nuevo número del índice 2.

Caso 3: Suma de todos los elementos del arreglo

```
Arreglo actualizado
1 2 45 4 5 6 7 8 9 10
La suma de los elementos del arreglo es: 97
```

La suma de los números 1, 2, 45, 4, 5, 6, 7, 8, 9, 10 si da un resultado de 97, lo que indica que el ciclo For si este sumando de manera correcta los elementos del arreglo.

Caso 4: ingreso de valor invalido para comprobar la segunda restricción en la búsqueda de datos en el array.

```
Ingresa un número para que lo busquemos en el Array:
a
Por favor, ingresa un número válido:
dd
Por favor, ingresa un número válido:
.56
Por favor, ingresa un número válido:
|
```

Al igual que en el primer caso no nos deja continuar si no colocamos los datos validos que nos indica el programa (elementos del tipo Int)

Caso 5: Elemento que no exista en el arreglo

```
Ingresa un número para que lo busquemos en el Array:
a
Por favor, ingresa un número válido:
dd
Por favor, ingresa un número válido:
.56
Por favor, ingresa un número válido:
32
El número 32 no se encuentra en el arreglo.
```

En este caso introducimos un elemento que no existía, por lo tanto, nos retornó el mensaje de que el número 32 no se encuentra en nuestro arreglo.

Caso 6: Elemento que si existe en el arreglo

```
Ingresa un número para que lo busquemos en el Array:
a
Por favor, ingresa un número válido:
dd
Por favor, ingresa un número válido:
.56
Por favor, ingresa un número válido:
7
El número 7 se encuentra en la posición 6 del arreglo.
```

Como se puede apreciar, en caso de que si exista el numero en el arreglo, este nos retornara la posición que ocupa dentro del array.



## Capturas de pantalla completas

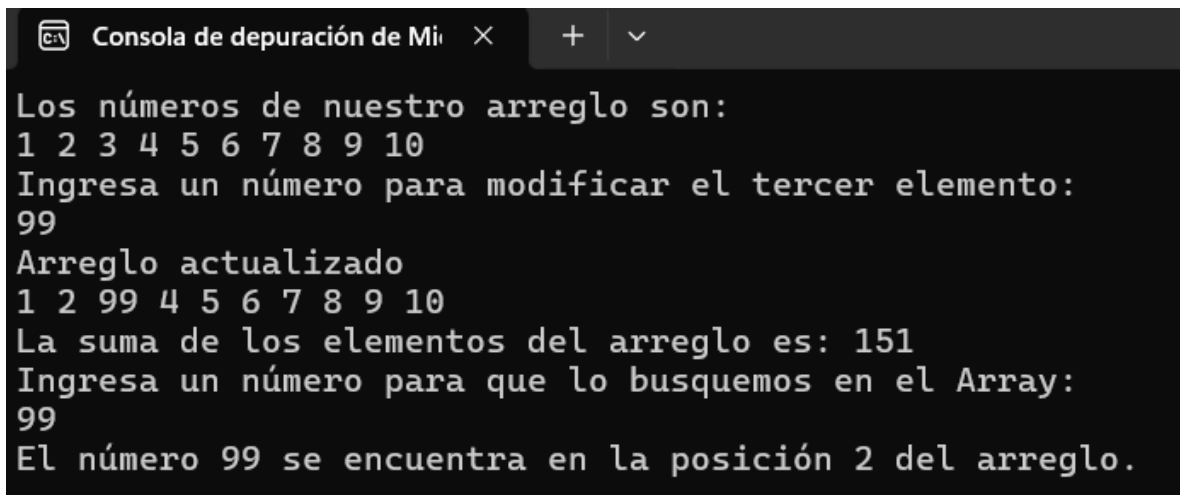
### Ejempló 1:

```
Consola de depuración de Mi × + v
Los números de nuestro arreglo son:
1 2 3 4 5 6 7 8 9 10
Ingresa un número para modificar el tercer elemento:
q
Por favor, ingresa un número válido:
3.5
Por favor, ingresa un número válido:
.45
Por favor, ingresa un número válido:
45
Arreglo actualizado
1 2 45 4 5 6 7 8 9 10
La suma de los elementos del arreglo es: 97
Ingresa un número para que lo busquemos en el Array:
a
Por favor, ingresa un número válido:
dd
Por favor, ingresa un número válido:
.56
Por favor, ingresa un número válido:
32
El número 32 no se encuentra en el arreglo.
```

### Ejemplo 2:

```
Consola de depuración de Mi × + v
Los números de nuestro arreglo son:
1 2 3 4 5 6 7 8 9 10
Ingresa un número para modificar el tercer elemento:
q
Por favor, ingresa un número válido:
3.5
Por favor, ingresa un número válido:
.45
Por favor, ingresa un número válido:
45
Arreglo actualizado
1 2 45 4 5 6 7 8 9 10
La suma de los elementos del arreglo es: 97
Ingresa un número para que lo busquemos en el Array:
a
Por favor, ingresa un número válido:
dd
Por favor, ingresa un número válido:
.56
Por favor, ingresa un número válido:
7
El número 7 se encuentra en la posición 6 del arreglo.
```

Ejemplo 3: Con modificación al número 99



```
Consola de depuración de Mi X + v
Los números de nuestro arreglo son:
1 2 3 4 5 6 7 8 9 10
Ingresa un número para modificar el tercer elemento:
99
Arreglo actualizado
1 2 99 4 5 6 7 8 9 10
La suma de los elementos del arreglo es: 151
Ingresa un número para que lo busquemos en el Array:
99
El número 99 se encuentra en la posición 2 del arreglo.
```

## 5. Conclusión

Este código representa una implementación eficiente para la manipulación de arreglos en **C#**, proporcionando al usuario diversas funcionalidades que incluyen la modificación de elementos, la realización de cálculos aritméticos y la búsqueda de valores dentro de la estructura de datos. A través de su diseño, el programa ejemplifica el uso de bucles, estructuras de control y validación de entradas, asegurando un flujo de ejecución robusto y libre de errores.