Practico 1

Programación II

Equipo Metagross

Nicòlas Rabellino
Sebastian Arbilla
Facundo Nuñez
Jose Maria Gonzalez

21/09/2022

Prof: Gonzalo Duarte

Práctico 1 - Competencia

1 – Introducción

Programación II es una asignatura de la carrera Analista Programador. Esta se dicta durante el primer año en el segundo semestre.
Para aprobar la asignatura hay que tener Programación I.
Para cursar la asignatura se puede deber el Examen Final de Programación I.
Dando una vista por las actividades realizadas se podrá apreciar el diverso trabajo y la búsqueda de soluciones para resolver los problemas planteados en la actividad dada, desde aquí hasta el final mostraremos cómo lo realizamos, sus limitaciones y alcances, sumado a la explicación de cada sección del código.

Equipo Metagross
Nicolas Rabellino, Facundo Nuñez, Sebastian Arbilla y Jose Maria Gonzalez

2 – Ejercicio 1

2.1 Problema planteado por la letra

Realizar un programa que muestre al usuario los detalles de
un auto; el tipo de tapicería y climatizador.

2.2 Solución considerada

Se utilizó la base del programa mostrado en clases,se modificaron ciertas partes.
Otra de estas es que se agregó el color del auto al código.
Se implementó una validación para cambiar el tipo de valor Verdadero/falso a “SI/No” para una fácil visualización.

```
private int ruedas;
private double largo;
private double ancho;
private bool climatizador;
private string tapecieria;
private string clima = "no";
private string color;
1 referencia
```

Figura 2.1: Atributo: color del Auto

```
public void setThisExtras(bool climatizador, string tapecieria)
{
    if (climatizador == true)
    {
        clima = "si";
    }
    this.tapecieria = tapecieria;
}
```

Figura 2.2: Método Setter

```
El auto verde  mide de largo 3 y de 2 ancho.
no tiene climatizador y tapecieria de tela

El auto rojo mide de largo 3,2 y de 2,6 ancho.
si tiene climatizador y tapecieria de cuero

El auto azul mide de largo 3 y de 2 ancho.
no tiene climatizador y tapecieria de lana
```

Figura 2.3: Resultado en consola

3 – Ejercicio 2

3.1 Problema planteado por la letra

Formar un algoritmo para mostrar en pantalla un mensaje del nombre de un alumno o nombre y la edad del mismo. Pidiendo también en esta que se incluya una clase alumno que contenga sus constructores adecuados, dos metodos especificos, y opcional (agregar getters y setters)

3.2 Solución considerada

Se implementó una clase alumno que tenga los atributos nombre y edad.
Los getter que muestran en pantalla los metodos Nombre, Nombre y Edad,

```
public void getNombre()
{
    Console.WriteLine("Nombre: " + nombreAlumno);
}
```

Figura 3.1: Método Getter que muestra solo el nombre.

```
public void getNombreEdad()
{
    Console.WriteLine("Nombre: " + nombreAlumno + " - Edad: " +edadAlumno);
}
```

Figura 3.2: Método Getter que muestra solo el Nombre y Edad.

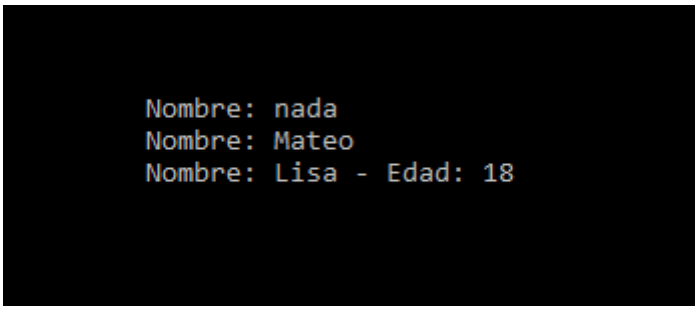

Figura 3.3: Resultado en consola

4 – Ejercicio 3 y 4

4.1 Problema planteado por la letra

Realizar un programa que contenga el algoritmo para imprimir la resta y la suma de dos vectores

4.1.2

Siguiendo la primera parte de la actividad se pide ampliar el ejercicio para poder calcular la distancia entre los puntos asociados a los vectores creados.

4.2 Solución considerada

Para dar solución a los problemas planteados consideramos crear una clase que se componga de los elementos necesarios para realizar las operaciones matemáticas al igual que los métodos a emplear para realizar los cálculos matemáticos. Por un lado un método que que carga los datos ingresados para determinar los parámetros y los componentes de los vectores, para después realizar las operaciones de suma y resta de los mismos; seguido de eso desarrollamos un método para que el usuario pueda visualizar en pantalla el resultado de las operaciones. Y para finalizar con la solución planteada realizamos un método que calcula y a la vez muestra al usuario la distancia entre los los puntos asociados a los vectores.

4.3 Resultado de la solución considerada

Como se puede ver en las siguientes imágenes (imagen 4.1 y imagen 4.2) al ejecutar el programa con la solución considerada se realizan correctamente el ingreso de los datos y los cálculos necesarios planteados por el problema de la letra

```
Ingrese la longitud de los vectores a sumar: 3
Ingresando valores al vector A
Ingrese componente [1]: 5
Ingrese componente [2]: 6
Ingrese componente [3]: 8
Ingresando valores al vector B
Ingrese componente [1]: 2
Ingrese componente [2]: 7
Ingrese componente [3]: 5
```

imagen 4.1: ingresos de los datos del ejercicio 3 - 4

```
La suma de los vecores es:
[7]
[13]
[13]

La resta de los vectores es:
[3]
[-1]
[3]


La distancia entre los puntos de los vectores asociados es: 3
La distancia entre los puntos de los vectores asociados es: 1
La distancia entre los puntos de los vectores asociados es: 3
```

imagen 4.2: resultados de los cálculos realizados

5 – Ejercicio 5

5.1 Problema planteado por la letra

Desarrollar un algoritmo para sumar dos matrices

5.2 Soluciones consideradas

Dado la ambigüedad de la letra del ejercicio 5, dimos con la conclusión de crear dos códigos con diferentes enfoques al problema original, uno de ellos, realiza la suma usando dos arrays singulares cargados con valores enteros aleatorios (Ejercicio_5) y por otro lado usamos dos matrices de dos por dos, el cual también usa valores enteros aleatorios (Ejercicio_5.1)

5.3 Primera solución considerada

Como acercamiento al ejercicio, se decidió usar una solución muy simplificada, para esto se utilizaron todos los conocimientos que ya se teníamos acerca de arrays, y usarlo para “simular” estas matrices de enteros aleatorios, para esto cargamos valores aleatorios dentro de ambos arrays usando un simple método “*for*” como se muestra en la siguiente imagen:
El segundo bucle “for” funciona como suma, mostrando el resultado de la agregación de los valores de cada array.

```
for (int i = 0; i <= 4; i++)
{
    numero = rnd.Next(0, 100);
    primerarray[i] = numero;

    numero = rnd.Next(0, 100);
    segundoarray[i] = numero;
}
```

Figura 5.1: Carga de valores enteros aleatorios

```
for (int i = 0; i <= 4; i++)
{
    Console.Write("La suma de " + primerarray[i] + " y " + segundoarray[i] + " es: ");
    numero = primerarray[i] + segundoarray[i];
    Console.WriteLine(numero);
}
```

Figura 5.2: Suma de “matrices”

```
----- Resultados -----
La suma de 54 y 79 es: 133
La suma de 36 y 53 es: 89
La suma de 82 y 6 es: 88
La suma de 12 y 93 es: 105
La suma de 26 y 29 es: 55
```

Figura 5.3: Resultados en consola

5.4 Segunda solución considerada

En primer lugar, se consideró crear una clase que contenga los elementos necesarios para la realización de las operaciones y a su vez visualizar al usuario los resultados de dichos cálculos. Se crearon tres matrices, las dos primeras contendrán en primer lugar los datos ingresados de forma aleatoria y por otro lado el tercero será utilizado para guardar los datos de la suma de los dos primeros matrices. A continuación del ingreso de datos se realizará la suma de las matrices, sumando el lugar correspondiente de una matriz con el lugar correspondiente de otra matriz, un ejemplo sería la suma del lugar 1.1 de la matriz A con el lugar 1.1 de la matriz B. y para finalizar con la solución se le muestra al usuario el resultado de las operaciones.

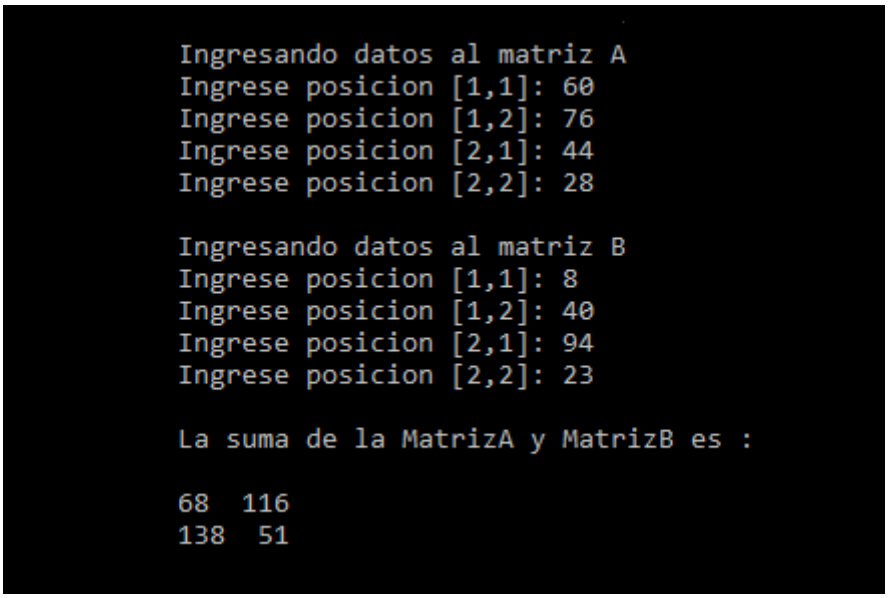

Imagen 5.1: Resultados en consola

6 – Ejercicio 6

6.1 Problema planteado por la letra

Crear un programa capaz de almacenar la hora con precisión de milisegundos que contenga un método que pueda incrementar la hora de diferente manera, que a su vez pueda calcular la diferencia entre diferente tiempos y la suma de horas.

6.2 Solución considerada

Para este problema decidimos utilizar el método DateTime el cual posee la capacidad de sumar, almacenar y comparar (entre otras utilidades) desde años hasta milisegundos, brindándonos una manera mucho más simple de utilizar estos horarios además de darnos una mejora mayor: poder usar días, meses y años, con la pequeña complicación de no poder mostrar en pantalla esos milisegundos utilizados, quitando esto, el programa es capaz de sumar, restar, comparar, almacenar y cambiar horas y días de varias maneras.

En la siguiente imagen se puede ver el resultado dado por ingresar las horas 21:13:0 y 10:20:20, contando con el día, mes y año en el que se ejecute el programa

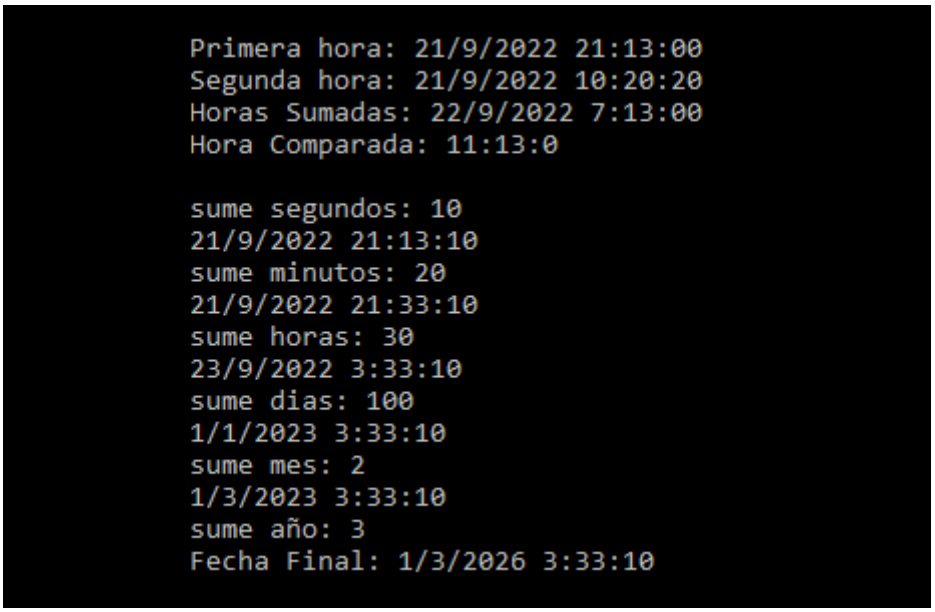

Figura 6.1: Resultado en consola

7 – Ejercicio 7

7.1 Problema planteado por la letra

Una empresa en crecimiento necesita digitalizar una serie de archivos para facilitar el acceso a los mismos, dado que el formato físico de estos está generando dificultades a la empresa

7.2 Solución considerada

Para resolver el problema planteado por la empresa consideramos crear diferentes clases que representen los diferentes archivos que la empresa desea digitalizar y agregar en estas clases los elementos que corresponden a cada una, un ejemplo de clase sería “Factura” y los elementos que esta alberga son “importe” y “fecha”. Seguido de la creación de las clases con sus correspondientes elementos, debíamos crear una forma de poder mostrarle al usuario los diferentes elementos de cada clase, determinando el equipo utilizar una interfaz para esta tarea, dado que la función a realizar (imprimir) es similar, y solo varia en los datos mostrados, pudiendo utilizar esta interfaz para todas las clases.

7.3 Resultado de la solución planteada

Como se puede apreciar en la imagen 7.1 la solución considerada cumple con el problema planteado por la empresa.

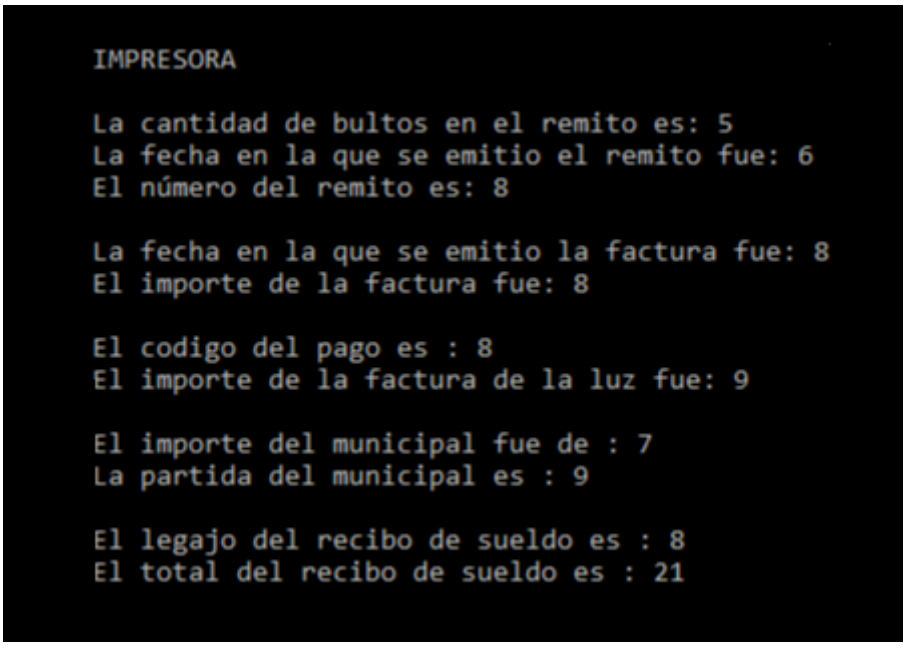

Imagen 7.1: impresión de los datos del ejercicio 7