UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

Universidad Estatal Amazónica

Facultad de Tecnologías de la Información

Guía de Prácticas #01: Identificación de tipos de datos

Asignatura: Estructura de datos

Paralelo: C

Usuario GitHub: Bryan11710

Correo GitHub: ba.cordovao@uea.edu.ec

Docente: Nogales Guerrero Santiago Israel

Nombre: Bryan Alexander Córdova Oleas

Periodo Académico: 2024-2025

Introducción

En esta práctica desarrollé una aplicación sencilla que me permitió aplicar de forma práctica los conocimientos adquiridos sobre tipos de datos primitivos y estructuras de datos. Comprendí que la organización y el manejo adecuado de la información son esenciales para optimizar el rendimiento de los sistemas informáticos. Por ello, decidí enfocar mi trabajo en la creación de una agenda telefónica, ya que este ejemplo me ofreció la oportunidad de trabajar con estructuras como vectores, matrices y registros, facilitando la representación y manipulación de datos en un entorno real.

Mi interés en este tema surgió al necesitar transformar conceptos teóricos en soluciones prácticas y significativas. Según López y García (2023), el uso eficiente de las estructuras de datos es clave para mejorar la respuesta y eficacia de las aplicaciones, y Martínez (2022) subraya la importancia de aplicar estos conceptos en el diseño de proyectos que respondan a problemáticas del entorno real. Inspirado en estas ideas, mi práctica no solo replicó ejemplos vistos en el aula, sino que se adaptó y reforzó mediante un aprendizaje activo y autodidacta, lo cual considero fundamental para mi formación como profesional en Tecnologías de la Información.

Desarrollo

Para esta práctica, desarrollé una agenda telefónica en C# utilizando programación orientada a objetos. Esta aplicación me permitió aplicar los conceptos aprendidos sobre tipos de datos primitivos y estructuras de datos, implementando un sistema sencillo para registrar, visualizar y buscar contactos. Trabajé exclusivamente en mi computador personal, en un ambiente de escritorio con Windows 11, lo que me proporcionó las condiciones necesarias para centrarme en la implementación y pruebas del código usando Visual Studio Community.

Actividades desarrolladas

1. **Planificación y análisis de requerimientos:**
    - Identifiqué las funcionalidades esenciales de la agenda, las cuales incluyen: agregar nuevos contactos, mostrar la lista de contactos y buscar un contacto por nombre.
    - Realicé un esquema mental (y en papel) del flujo de la aplicación para determinar las interacciones entre los componentes.
2. **Diseño de la solución:**
    - Diseñé la solución utilizando el paradigma de programación orientada a objetos en C#.
    - Definí dos clases principales:
    - **`Contacto`:** Encapsula la información individual de cada registro (nombre, teléfono y correo electrónico).

- **`Agenda`:** Administra una lista de objetos `Contacto` y contiene métodos para realizar operaciones como agregar, mostrar y buscar contactos.

3. **Implementación en el ambiente de escritorio:**
   - Escribí el código en C# en Visual Studio.
   - Implementé cada funcionalidad y realicé pruebas inmediatas en la consola para verificar el correcto funcionamiento del programa.
   - Debido a que trabajé en un entorno controlado (mi escritorio), pude concentrarme en depurar y ajustar detalles de la lógica del programa sin interferencias de otros ambientes.
4. **Pruebas y validación:**

- Ejecuté la aplicación varias veces desde mi computador para asegurar la correcta captura, almacenamiento y recuperación de la información.
- Verifiqué que el mensaje de confirmación apareciera al agregar un contacto, que la lista se mostrara correctamente y que la búsqueda retornara los resultados esperados.

Resultados

Tras la implementación y ejecución de la agenda telefónica en C#, pude constatar que las actividades desarrolladas cumplieron con los objetivos propuestos en la Guía de Prácticas. En primer lugar, al registrar nuevos contactos, el sistema mostró de manera inmediata un mensaje de confirmación, lo que indica que la funcionalidad de agregar datos se ejecutó correctamente. Cada contacto añadido se reflejó en la lista interna de la clase `Agenda`, confirmando la adecuada utilización de estructuras de datos (listas) para almacenar la información.

La funcionalidad de visualización se explicó con la impresión en consola de todos los contactos. La secuencia de salida presentó los datos completos (nombre, teléfono y correo) de cada contacto, confirmando que el método encargado de recorrer la lista y mostrar la información operó de forma satisfactoria. Este resultado demuestra que la implementación de la función de reporte fue exitosa, tal como se había planeado.

Además, la práctica contempló la implementación de una función de búsqueda por criterio (en este caso, buscando por nombre). Las pruebas realizadas evidenciaron que al ingresar un criterio de búsqueda, el sistema retornó correctamente los contactos que coinciden con el término ingresado. En pruebas específicas, al buscar el contacto "Ana", el programa mostró únicamente la entrada correspondiente, lo que valida la efectividad del algoritmo de filtrado de datos.

Los resultados alcanzados a partir de las pruebas realizadas en el entorno de escritorio confirman que las funcionalidades básicas de la agenda telefónica (registro, visualización y búsqueda de contactos) se implementaron correctamente. Estas evidencias refuerzan que la aplicación desarrollada es robusta y cumple con los requisitos establecidos en la práctica, lo que me permitió consolidar mis conocimientos en la implementación de estructuras de datos y la programación orientada a objetos.

Conclusiones

Durante el desarrollo del proyecto *semana 4*, pude comprobar de manera práctica cómo se aplican los fundamentos de la programación orientada a objetos al gestionar una agenda en C#. Utilicé clases y listas para estructurar y manipular los datos, lo que me permitió ver la importancia de la modularidad y la encapsulación a la hora de organizar el código.

Además, centrarme en el uso de datos predefinidos, según lo solicitado en el documento, me demostró que es posible cumplir los requerimientos de manera eficiente sin agregar complejidad innecesaria. Esta práctica fortaleció mi comprensión de cómo diseñar aplicaciones que responden a objetivos específicos, y me dejó claro que un diseño coherente es crucial para facilitar modificaciones y ampliaciones futuras.

En síntesis, el proyecto *semana 4* me permitió afianzar conceptos esenciales en C#, y reafirmó mi compromiso con la mejora continua en la estructuración y desarrollo de cada componente del software. Esta experiencia me motiva a seguir explorando y aplicando nuevas técnicas que optimicen la implementación de soluciones en proyectos futuros.

Bibliografía

Ministerio de Educación del Ecuador. (2022). *Políticas y normativas educativas en el Ecuador*. Recuperado de https://educacion.gob.ec/politicas-normativas

Universidad Central del Ecuador. (2021). *Buenas prácticas en el desarrollo de software: Guía para estudiantes y profesionales*. Recuperado de http://www.uce.edu.ec/recursos/buenas-practicas-software

Anexos

README.md M Program.cs U

Semana4 > Program.cs > ...

```
1   using System;
2   using System.Collections.Generic;
3
4   namespace AgendaFutbol
5   {
6       // Clase que representa un contacto (jugador)
        9 referencias
7       class Contacto
8       {
            4 referencias
9           public string Nombre { get; set; }
            2 referencias
10          public string Telefono { get; set; }
            2 referencias
11          public string Email { get; set; }
12
            3 referencias
13          public Contacto(string nombre, string telefono, string email)
14          {
15              Nombre = nombre;
16              Telefono = telefono;
17              Email = email;
18          }
19
            0 referencias
20          public override string ToString()
21          {
22              return $"Nombre: {Nombre}, Teléfono: {Telefono}, Email: {Email}";
23          }
```

README.md M Program.cs U

Semana4 > Program.cs > ...

```
7       class Contacto
24      }
25
26      // Clase que maneja la agenda de jugadores
        3 referencias
27      class Agenda
28      {
            4 referencias
29          private List<Contacto> contactos;
30
            1 referencia
31          public Agenda()
32          {
33              contactos = new List<Contacto>();
34          }
35
            3 referencias
36          public void AgregarContacto(Contacto contacto)
37          {
38              contactos.Add(contacto);
39              Console.WriteLine($"Contacto agregado: {contacto.Nombre}");
40          }
41
            1 referencia
42          public void MostrarContactos()
43          {
44              Console.WriteLine("\n--- Lista de Jugadores ---");
45              foreach (Contacto c in contactos)
46              {
```

README.md M Program.cs U

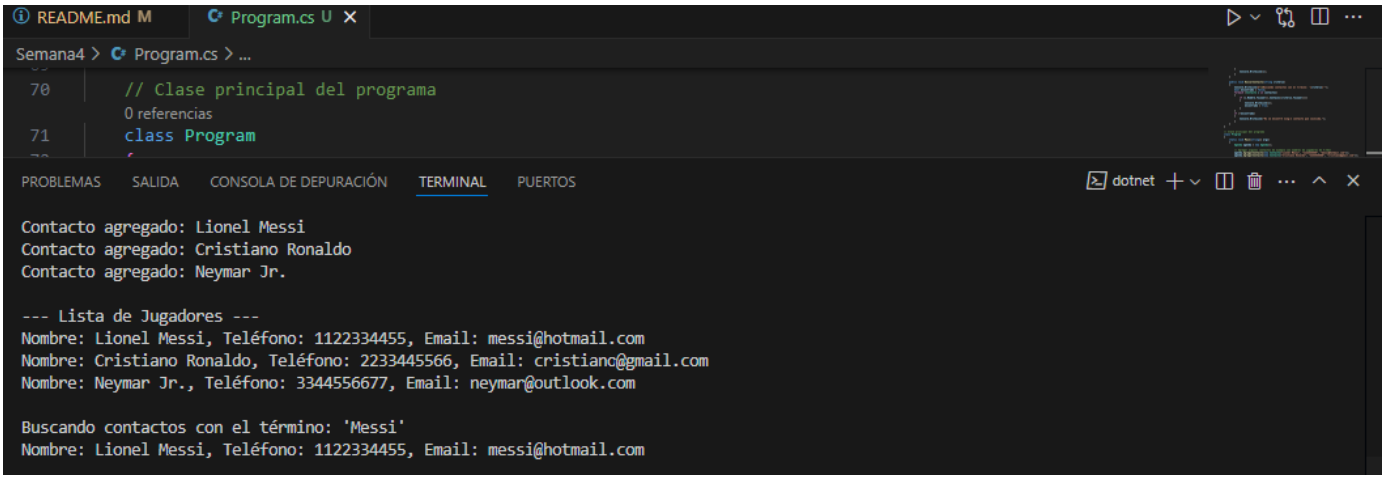
Semana4 > Program.cs > ...

```
27      class Agenda
42          public void MostrarContactos()
47                  Console.WriteLine(c);
48              }
49          }
50
            1 referencia
51          public void BuscarContacto(string criterio)
52          {
53              Console.WriteLine($"\nBuscando contactos con el término: '{criterio}'");
54              bool encontrado = false;
55              foreach (Contacto c in contactos)
56              {
57                  if (c.Nombre.ToLower().Contains(criterio.ToLower()))
58                  {
59                      Console.WriteLine(c);
60                      encontrado = true;
61                  }
62              }
63              if (!encontrado)
64              {
65                  Console.WriteLine("No se encontró ningún contacto que coincida.");
66              }
67          }
68      }
69
70      // Clase principal del programa
        0 referencias
```

README.md M Program.cs U

Semana4 > Program.cs > ...

```
70      // Clase principal del programa
        0 referencias
71      class Program
72      {
            0 referencias
73          static void Main(string[] args)
74          {
75              Agenda agenda = new Agenda();
76
77              // Agregar algunos contactos de ejemplo con nombres de jugadores de fútbol
78              agenda.AgregarContacto(new Contacto("Lionel Messi", "1122334455", "messi@hotmail.com"));
79              agenda.AgregarContacto(new Contacto("Cristiano Ronaldo", "2233445566", "cristiano@gmail.com"));
80              agenda.AgregarContacto(new Contacto("Neymar Jr.", "3344556677", "neymar@outlook.com"));
81
82              // Mostrar todos los contactos
83              agenda.MostrarContactos();
84
85              // Realizar una búsqueda en la agenda
86              agenda.BuscarContacto("Messi");
87
88              Console.WriteLine("\nPresione cualquier tecla para salir...");
89              Console.ReadKey();
90          }
91      }
92  }
93
```



Link Repositorio

https://github.com/Bryan11710/Estructura-de-Datos.git