**CARRERA TECNOLOGÍAS DE LA INFORMACIÓN**

**Modalidad de estudios En Línea**

**Unidad de Organización Curricular:** Básica

|  |  |  |
| --- | --- | --- |
| **INFORME DE PRÁCTICAS DEL COMPONENTE PRÁCTICO-EXPERIMENTAL** | | |
| **Nombre de los estudiantes:**  Wio Arnaud Gualinga Bouchat | | |
| **Asignatura:** ESTRUCTURA DE DATOS | | **Calificación:** |
| **Fecha del informe: 04/10/2025** | |
| **N° Práctica: 4** | |
| **Título de la Práctica:** Guía de Prácticas #04: Implementación y representación de árboles y grafos | |
| **Introducción** | La asignatura de **Estructura de Datos** tiene como objetivo enseñar a organizar, almacenar y manipular información de manera eficiente mediante estructuras que faciliten la programación y el análisis de datos. Entre estas estructuras, los **árboles** son especialmente relevantes, ya que permiten representar datos de forma jerárquica y ejecutar búsquedas o recorridos de manera eficiente (Cormen, Leiserson, Rivest & Stein, 2009).  **Conceptualización:** Un **árbol** es una estructura de datos jerárquica compuesta por nodos conectados mediante enlaces, donde un nodo raíz contiene referencias a nodos hijos, permitiendo representar relaciones de dependencia o jerarquía. Los **árboles binarios** son un caso particular donde cada nodo puede tener como máximo dos hijos, aunque existen estructuras más generales donde un nodo puede tener múltiples hijos (Knuth, 1998).  **Fundamentación Teórica:** La representación de información mediante árboles permite organizar datos de forma clara y ordenada. En el ámbito educativo, los árboles son útiles para visualizar calificaciones de estudiantes, organizar materias o representar jerarquías de información, permitiendo un análisis rápido de la información (AppMaster, 2021).  **Antecedentes:** Diversos trabajos han aplicado árboles para representar datos académicos. Por ejemplo, en estudios recientes de algoritmos de gestión escolar, se ha utilizado la estructura de árbol para organizar calificaciones y generar reportes jerárquicos por estudiantes y materias, facilitando tanto la visualización como el procesamiento de información (Lee & Hubbard, 2024).  **Justificación:** La práctica de construir árboles a partir de un bloque de notas permite aplicar conocimientos de estructuras de datos a situaciones reales, promoviendo la comprensión de cómo representar información jerárquica y cómo manipularla eficientemente mediante programación. Esto desarrolla habilidades de pensamiento lógico y algorítmico, fundamentales en Tecnologías de la Información. | |
| **Desarrollo o procedimiento** | La actividad se desarrolló en un **entorno de escritorio** utilizando **Visual Studio Code** y el **SDK de .NET 9.0**, lo que permitió ejecutar, probar y depurar el código de manera eficiente. La información utilizada provino de un **archivo de texto (notas.txt)**, que contenía las calificaciones de tres estudiantes en tres materias diferentes. Esto permitió que los árboles se construyeran a partir de un bloque de notas, cumpliendo con el requerimiento de la práctica.  Se implementaron **dos árboles distintos**, cada uno con un enfoque diferente: **por estudiante** y **por materia**, permitiendo observar la información desde distintos ángulos y facilitando el análisis de los datos.  **Estructuras de datos utilizadas**   * **Clase Nodo**: representa cada elemento del árbol, conteniendo un **nombre** y una **lista de hijos (List<Nodo>)**. Esta estructura permitió almacenar jerárquicamente tanto estudiantes como materias y calificaciones. * **List<Nodo>**: se utilizó para contener los nodos hijos de cada nodo, facilitando la adición dinámica de elementos y permitiendo recorrer la jerarquía de manera flexible. * **Recursividad**: el método Imprimir recorre los árboles de manera automática, mostrando cada nodo con sangría según su nivel, lo que refleja la jerarquía sin necesidad de realizar bucles manuales para cada nivel. * **Diccionarios (Dictionary<string, Nodo>)**: organizaron de manera rápida los nodos raíz por estudiante o por materia, asegurando acceso eficiente y evitando duplicados.   **Procedimientos realizados**   1. **Lectura del archivo notas.txt**    * Se leyó todo el archivo línea por línea, separando cada línea en **Nombre, Materia y Nota**.    * Esto permitió que los árboles se construyeran de manera dinámica y escalable, sin depender de datos codificados manualmente. 2. **Creación de nodos raíz**    * **Árbol por estudiante**: cada estudiante se definió como nodo raíz y sus materias con calificaciones como nodos hijos.    * **Árbol por materia**: cada materia se definió como nodo raíz y los estudiantes con sus calificaciones como nodos hijos. 3. **Agregación de nodos hijos**    * Se utilizó el método AgregarHijo para asociar cada calificación a su nodo correspondiente, ya sea estudiante o materia. 4. **Recorrido e impresión de los árboles**    * Se implementó el método Imprimir de manera recursiva, mostrando cada nodo con sangría, reflejando la jerarquía y facilitando la comprensión visual de las relaciones entre nodos. 5. **Pruebas y verificación**    * Se ejecutaron ambos árboles en consola para verificar que los datos se mostraran correctamente, respetando la jerarquía y asegurando que todos los nodos estuvieran presentes. 6. **Manejo de cambios o ampliaciones**    * La estructura permite agregar nuevos estudiantes o materias sin modificar la lógica del árbol ni el método de impresión, gracias al diseño modular de la clase Nodo y la lista de hijos.   **Bloque de notas (ficticio)**   * Ana → Matemáticas: 18, Lengua: 15, Ciencias: 17 * Luis → Matemáticas: 14, Lengua: 16, Ciencias: 15 * María → Matemáticas: 19, Lengua: 18, Ciencias: 20   **Ejemplo 1: Árbol de notas por estudiante**  Cada estudiante es un nodo raíz y sus materias con calificaciones son nodos hijos. Esto permite observar **el desempeño individual de cada estudiante**. Por ejemplo:   * Ana   + Matemáticas: 18   + Lengua: 15   + Ciencias: 17 * Luis   + Matemáticas: 14   + Lengua: 16   + Ciencias: 15 * María   + Matemáticas: 19   + Lengua: 18   + Ciencias: 20   **Ejemplo 2: Árbol de notas por materia**  Cada materia es un nodo raíz y los estudiantes con sus calificaciones son nodos hijos, permitiendo observar **el desempeño de todos los estudiantes en una materia específica**. Por ejemplo:   * Matemáticas   + Ana: 18   + Luis: 14   + María: 19 * Lengua   + Ana: 15   + Luis: 16   + María: 18 * Ciencias   + Ana: 17   + Luis: 15   + María: 20   **Agente de IA utilizado:**  Durante el desarrollo del código se utilizó **ChatGPT** como herramienta de asistencia. Aproximadamente un **50% del código** fue sugerido por la IA, principalmente en la construcción del esqueleto de la clase Nodo, los métodos de lectura de archivo y el método recursivo Imprimir. La lógica de negocio, la adaptación a la estructura específica de notas y la integración de ambos árboles fueron realizadas manualmente por el estudiante. | |
| **Resultados** | **Resultados generales**   * Se implementaron correctamente dos árboles a partir de un bloque de notas (notas.txt). * El árbol por estudiante permitió analizar el desempeño de cada estudiante individualmente, mostrando sus calificaciones en todas las materias. * El árbol por materia facilitó la comparación de calificaciones entre estudiantes en cada materia específica, permitiendo una visión clara del rendimiento por asignatura. * Reportería: La aplicación mostró los datos de forma jerárquica en **consola**, permitiendo visualizar claramente los nodos raíz y sus hijos. Se pudieron consultar todas las calificaciones por estudiante y por materia, mostrando la estructura completa de los árboles y facilitando la interpretación de la información. * La ejecución del código fue rápida, con un tiempo promedio de respuesta menor a 2 segundos, dado que todos los datos se manejaron en memoria. * La reportería en consola mostró los datos de forma organizada y jerárquica, cumpliendo con la finalidad de la práctica y permitiendo verificar visualmente la correcta construcción de los árboles.   **Resultados específicos**   |  |  |  | | --- | --- | --- | | Aspecto | Ventajas | Desventajas | | Árbol por estudiante | Permite analizar el rendimiento individual; estructura clara de jerarquía | No permite comparar calificaciones por materia rápidamente | | Árbol por materia | Facilita comparación por materia; jerarquía clara de información | No permite ver historial completo por estudiante fácilmente | | Código en C# | Modularidad, recursividad y reutilización de métodos | Requiere conocimiento básico de programación orientada a objetos |   Los resultados evidencian que la implementación de árboles jerárquicos es una herramienta eficiente para organizar y visualizar datos académicos, permitiendo distintos enfoques de análisis según el objetivo deseado, ya sea evaluar el desempeño individual de los estudiantes o comparar rendimientos por materia. | |
| **Conclusiones** | Los árboles son estructuras eficientes para representar información jerárquica, facilitando tanto el análisis individual como comparativo de los datos académicos.  La práctica permitió aplicar conocimientos teóricos a un problema real, desarrollando habilidades de **programación orientada a objetos**, **recursividad** y **organización de datos**.  La implementación demostró que la elección de la estructura adecuada depende del tipo de análisis que se desea realizar: por estudiante o por materia.  El manejo de datos en memoria asegura rapidez, claridad y evita duplicación de esfuerzos en la visualización de información. | |
| **Bibliografía** | Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.  Knuth, D. (1998). *The Art of Computer Programming, Volume 1: Fundamental Algorithms* (3rd ed.). Addison-Wesley.  AppMaster. (2021). ¿Qué son las estructuras de datos? Recuperado de <https://appmaster.io/es/blog/que-son-las-definiciones-y-tipos-de-estructuras-de-datos>  Lee, K. D., & Hubbard, S. (2024). Sets and Maps. En *Data Structures and Algorithms with Python* (pp. 141–163). Springer, Cham. <https://doi.org/10.1007/978-3-031-42209-6_5> | |
| **Anexos** | **Anexo 1 (enlace al repositorio):**  **Anexo 2 (*Reporteria visual*)**  *funcionamiento del código:*    *Arboles de nota por estudiantes:*    *Arboles de nota por materias:*    **Anexo 3 (Código):**  **class Nodo**  **{**  **public string Nombre;**  **public List<Nodo> Hijos = new List<Nodo>();**  **public Nodo(string nombre)**  **{**  **Nombre = nombre;**  **}**  **public void AgregarHijo(Nodo hijo)**  **{**  **Hijos.Add(hijo);**  **}**  **public void Imprimir(string espacio = "")**  **{**  **Console.WriteLine(espacio + Nombre);**  **foreach (var hijo in Hijos)**  **hijo.Imprimir(espacio + " ");**  **}**  **}**  **class Programa**  **{**  **static void Main()**  **{**  **string rutaArchivo = Path.Combine(Directory.GetParent(Directory.GetCurrentDirectory()).Parent.Parent.FullName, "notas.txt");**  **if (!File.Exists(rutaArchivo))**  **{**  **Console.WriteLine("Archivo notas.txt no encontrado en la carpeta del proyecto.");**  **return;**  **}**  **string[] lineas = File.ReadAllLines(rutaArchivo);**  **Dictionary<string, Nodo> estudiantes = new Dictionary<string, Nodo>();**  **Dictionary<string, Nodo> materias = new Dictionary<string, Nodo>();**  **foreach (string linea in lineas)**  **{**  **string[] partes = linea.Split(',');**  **if (partes.Length != 3) continue;**  **string nombre = partes[0].Trim();**  **string materia = partes[1].Trim();**  **string nota = partes[2].Trim();**  **if (!estudiantes.ContainsKey(nombre))**  **estudiantes[nombre] = new Nodo(nombre);**  **estudiantes[nombre].AgregarHijo(new Nodo($"{materia}: {nota}"));**  **if (!materias.ContainsKey(materia))**  **materias[materia] = new Nodo(materia);**  **materias[materia].AgregarHijo(new Nodo($"{nombre}: {nota}"));**  **}**  **Console.WriteLine("Árbol de notas por estudiante:\n");**  **foreach (var est in estudiantes.Values)**  **est.Imprimir();**  **Console.WriteLine("\n-----------------------------\n");**  **Console.WriteLine("Árbol de notas por materia:\n");**  **foreach (var mat in materias.Values)**  **mat.Imprimir();**  **}**  **}** | |

**Firmas de los estudiantes:**

|  |
| --- |
| **Texto  Descripción generada automáticamenteFirma:** |
| **Nombre estudiante: Wio Arnaud Gualinga Bouchat** |

**Responsable UEA**

**Docente:** Ing. ORTEGA TENEZACA DELFIN BERNABE