Ejercicio en Clase 3

1. Análisis del Programa

Tareas Repetitivas o que pueden encapsularse en funciones

- Agregar estudiantes.
- Mostrar la lista de estudiantes.
- Calcular promedios.
- Encontrar al estudiante con la calificación más alta.

Algunas de las funciones que se podrían agregar al código serian:

1. Agregar Estudiante ():
   Esta función ayudaría en el proceso de ingresar estudiante y su respectiva calificación y se remplazaría la lógica.

2. Mostrar Estudiantes ():
   Esta función ayudaría en la lógica para mostrar la lista de los estudiantes ingresados junto con su calificación.

3. Calcular Promedio():
   Esta función ayudaría en capsular la lógica para calcular y mostrar el promedio de cada estudiante respecto a su calificación ingresas.

4. MostrarEstudianteCalificacionAlta():
   Esta función ayudaría en la lógica para encontrar y mostrar al estudiante con la nota mas alta de las que hayan sido ingresadas.

   Cada tarea esta encapsulada en una función, lo cual hará que el código sea más claro y más fácil de mantener.

Variables Locales vs Variables Globales

- Discutir cuándo es apropiado usar variables locales y cuándo usar variables globales.

El uso correcto de variables locales y globales mejora la claridad y legibilidad del proyecto, por tal manera seria mejor ingresas mas variables locales que globales.

- **¿Qué datos deben ser accesibles en todo el programa?**

Se necita el ingreso de variables globales ya que serian datos que se comparten entre varias funciones y métodos, como listas y calificaciones ingresadas.

- **¿Qué datos solo son necesarios dentro de una función específica?**

Se necesita el ingreso solo variables locales ya que estas sirven para cálculos temporales o iterar en bucles.

2. Modularización

Convertir el Programa en Funciones

- Modulariza el programa creando funciones claras y específicas. Por ejemplo:
  - Agregar Estudiante() Permite ingresar un estudiante y su calificación
  - MostrarEstudiantes() Muestra la lista de estudiantes con sus calificaciones
  - CalcularPromedio() Calcula y muestra el promedio de las **calificaciones**
  - MostrarEstudianteConMaxCalificacion() (extra) Encuentra muestra con la mejor calificación

Definir Variables Locales y Globales

- Define qué variables deben ser locales y cuáles deben ser globales.
  - **¿Qué datos necesitan ser compartidos entre múltiples funciones?**

Las listas de estudiantes y calificaciones deben ser globales Varias opciones del menú las utilizan para agregar, mostrar y procesar Información de estudiantes

- **¿Qué datos solo son relevantes dentro de una función específica?**

Las variables nombre y calificación en la opción 1 solo son necesarias al agregar un estudiante.
Se suma y promedio en la opción 3 solo se usan para calcular el promedio de calificaciones.

Max calificación y estudiante Max en la opción 4 solo son útiles para determinar la calificación más alta.

Opción solo es relevante para procesar la selección del usuario en cada iteración del while.

3. Preguntas Guía

Responde las siguientes preguntas en tu análisis:

1. **¿Qué ventajas tiene dividir el código en funciones?**
    - Reflexiona sobre cómo el modularidad mejora la organización, reutilización y mantenimiento del código.

- ✓ Facilita la reutilización del código.
- ✓ Reduce el riesgo de errores inesperados ya que los errores en variables locales solo existen mientras la función se ejecuta, liberando memoria una vez terminan.
- ✓ Fomenta el encapsulamiento y la seguridad , evitando que otras partes del código modifiquen datos de manera no intencionada.
- ✓ Hace el código más legible y estructurado.

2. **¿Por qué es importante limitar el uso de variables globales?**
    - Considera los riesgos de tener demasiadas variables globales, como la posibilidad de errores inesperados o dificultades para depurar el código.

- ✓ Reduce el Riesgo de errores inesperados
- ✓ Ya que las variables globales pueden ser modificadas en cualquier parte del código, lo que puede causar errores
- ✓ Optimiza el uso de memoria, porque las variables locales solo existen mientras la función se ejecuta y luego se eliminan automáticamente

3. **¿Cómo se puede mejorar la legibilidad del código?**
    - Piensa en cómo nombrar funciones y variables de manera descriptiva, así como en la importancia de comentarios y estructura clara.

- ✓ Para mejorar la legibilidad del código se pueden seguir prácticas.
- ✓ Aplicar Principios de programación estructurado y modular por separado la lógica

Mejoras Adicionales

- **Validación de Entradas del Usuario:**
    - Implementa validaciones para evitar errores si el usuario ingresa texto en lugar de números.

Sirve para mejorar el código en cuanto a validación de entradas, lo cual se podría hacer de la siguiente manera: modificar la entrada de datos para asegurarse de que el usuario ingrese números válidos cuando se le pida una calificación.

Console.Write("Ingrese la calificación del estudiante (0-100): ");

    - Ejemplo: Si se pide una calificación, asegúrate de que el valor ingresado sea numérico y esté dentro de un rango válido.