Tarea

Laboratorio 3

Estudiantes	**Carné**
Mario David Tereta Sapalun	0905-15-14297
Lester David Payes Méndez	0905-24-22750

Facultad de Ingeniería En Sistemas de Información y Ciencias de la Computación.

Universidad Mariano Gálvez De Guatemala

Programación 1

21 de febrero de 2025

Debate y Análisis del Programa

Tareas Repetitivas o que pueden encapsularse en funciones

Uso de la función line1()

Al inicio del programa debatimos con mi compañero, que en este se tendría que utilizar una función llamada linea1(), ya que es una acción que se repite muchas veces. Al usar esta función, evitamos repetir el mismo código cada vez que necesitamos leer un dato numérico, utilizando el tipo de dato double. En las funciones donde se recibe un dato entero, solo se convierte ese retorno de dato en int. Además, la función incluye la validación para que los datos sean correctos es decir numéricos. Si el usuario ingresa algo incorrecto, la función le pedirá que lo ingrese de nuevo, asegurando que solo se usen datos válidos en el programa.

Variables Globales vs Locales

En el programa, las únicas variables que se utilizan son las listas estudiantes y calificaciones. Estas variables están declaradas dentro de la clase Program, pero fuera de cualquier función, lo que les permite ser accesibles desde cualquier función estática dentro de la clase, como el método Main(). Dado que las usamos en varias funciones estáticas, se consideran variables globales dentro de la clase.

```
0 referencias
class Program
{
    static List<string> estudiantes = new List<string>();
    static List<double> calificaciones = new List<double>();
    3 referencias
    static double Line1(String num1)
    {
        Console.WriteLine(num1);
        if (!double.TryParse(Console.ReadLine(), out double Valido))
        {
            Console.WriteLine("Número no válido");
            return Line1(num1);
        }
        return Valido;
    }
    1 referencia
```

Por otro lado, las funciones AddStudent(), ListStudent(), y otras, manejan variables locales que solo existen dentro de esas funciones. Esto es clave para mantener el orden y la claridad en el código. Al mantener las variables dentro de su ámbito local, evitamos posibles errores al modificar variables globales de manera no controlada. Además, tener funciones locales ayuda a mantener un mejor control de los datos, mejora la legibilidad y hace que el código sea más fácil de mantener.

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

> **Lester Payes:** Las variables globales se utilizan cuando varias funciones necesitan modificar o usar la misma variable para funcionar sin pasárselas como parámetro, mientras que las variables locales se usan cuando solo son necesarias para hacer funcionar la función en la que se declararon y no se necesitarán en ninguna otra.
>
> **Mario Tereta:** Las globales son las que se declaran fuera de la función principal, permitiendo usarlas en cualquier parte del programa y las locales cuando queremos que estas no puedan ser modificadas fuera de la función donde fueron declaradas .

Debate de la Modularización

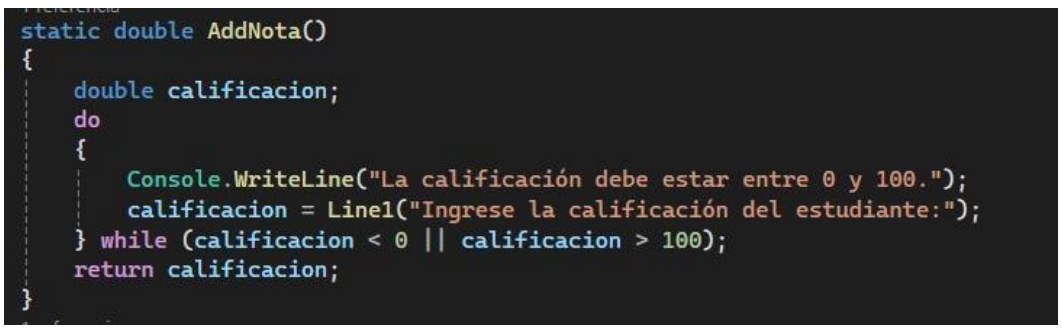
División del código en funciones

Concordamos que, para mejorar la visibilidad y claridad del código, realizaríamos una modularización adecuada. Se crearon funciones específicas para manejar diferentes operaciones del programa, ya que el código original se veía desordenado al utilizarlas. Las funciones creadas para el menú son:

- AddStudent()
- ListStudent()
- PromCal()
- MaxCal()

Estas funciones permiten que el código sea más organizado y fácil de entender, evitando la redundancia de código y facilitando su mantenimiento en el menú principal.

Así mismo en el programa se implementa una función para agregar una calificación para cada estudiante. Si más adelante se desea agregar una nota para cada curso, esta función ya está implementada y puede ser modificada para realizar la nueva acción requerida.



```
static double AddNota()
{
    double calificacion;
    do
    {
        Console.WriteLine("La calificación debe estar entre 0 y 100.");
        calificacion = Line1("Ingrese la calificación del estudiante:");
    } while (calificacion < 0 || calificacion > 100);
    return calificacion;
}
```

Uso de switch en lugar del menú sugerido en el código original

En lugar de un menú que no llama funciones y aparte de eso se realiza con las condicional if, se utilizó un switch para gestionar las diferentes opciones del programa es decir llamar las funciones creadas por nosotros y administrar la variable opción de mejor manera. Esto permite una mejor organización del flujo de ejecución y facilita la adición de nuevas opciones en el futuro.

Uso de Console.Clear() y Console.ReadKey();

Se implementó Console.Clear() en el menú principal para borrar procedimientos de las funciones anteriores seleccionadas por el usuario y mejorar la estética del código. Esto ayuda a mantener la salida del programa más estética. Además, se utilizó Console.ReadKey(); para que luego de terminar la ejecución de cada función no se borre de una vez, sino que el procedimiento se elimina hasta que el usuario presione una tecla.

Optimización del WriteLine() para mostrar el menú

En lugar de usar varias llamadas a Console.WriteLine() para mostrar el menú, se utilizó una sola línea con caracteres especiales como \n para los saltos de línea y la diagonal inversa \ para mejorar la estética y claridad del código. Esto reduce la redundancia y hace que la estructura del menú sea más compacta y fácil de leer.

```
0 referencias
static void Main(string[] args)

        while (true)
{
    Console.Clear();
    Console.WriteLine("-------Bienvenido al sistema de gestión de estudiantes------");
    Console.WriteLine("\n1. Agregar estudiante \n2. Mostrar Lista de Estudiantes \n3. Calcular Promedio de Calificacion \n4. Mostrar Estudiante con calificacion más alta \n5. Salir");
    int opcion = (int) Line1("Seleccione una opción: ");

    switch (opcion)
    {
        case 1:
            AddStudent();
            break;
        case 2:
            ListStudent();
            break;
        case 3:
            PromCal();
            break;
        case 4:
            MaxCal();
            break;
        case 5:
            Console.WriteLine("Saliendo del sistema...");
            return;
        default:
            Console.WriteLine("Opción no válida. Intente de nuevo.");
            break;

    }
    Console.WriteLine("Presione una tecla para continuar...");
    Console.ReadKey();
}
```

Definir Variables Locales y Globales

- **Variables Globales**: En este caso, las listas estudiantes y calificaciones deben ser **globales**, ya que se utilizan en múltiples funciones dentro de la clase y deben ser accesibles para todas ellas.
- **Variables Locales**: Las variables que se usan solo dentro de una función específica, como las de las funciones AddStudent() y ListStudent(), deben ser **locales**, ya que solo son relevantes dentro del contexto de esas funciones.

1. **¿Qué datos necesitan ser compartidos entre múltiples funciones?**

Las listas estudiantes y calificaciones necesitan ser **compartidas** entre las diferentes funciones de la clase, por lo que deben ser globales. Estas listas contienen los datos que las funciones requieren para operar (agregar estudiantes, listar calificaciones, etc.).

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

Los datos dentro de funciones como AddStudent() o ListStudent(), que no necesitan ser utilizados fuera de esas funciones, deben ser **locales**. Estos datos no afectan a otras partes del código y su uso está restringido al ámbito de la función.

Preguntas Guía

1. **¿Qué ventajas tiene dividir el código en funciones?**

Mejora organización del código, facilitan la comprensión y seguimiento. También nos permiten reutilizar funciones en diferentes partes del código. Facilidad de mantenimiento. Facilita la comprensión y el trabajo colaborativo.

2. **¿Por qué es importante limitar el uso de variables globales?**

Es muy importante ya que se corre el peligro de ser modificadas en cualquier parte del código, lo que podría causar errores. Si el programa depende de variables globales en donde es mejor utilizar locales, cualquier cambio en ellas puede afectar varias funciones. Al reducir las variables globales, cada función tiene su propio contexto y no se ve afectada por otras partes del programa. El uso excesivo de variables globales puede dificultar la colaboración, ya que múltiples personas podrían estar manipulando las mismas variables sin darse cuenta.

3. **¿Cómo se puede mejorar la legibilidad del código?**

Seguir un solo estilo determinado, como el uso de CamelCase o snake_case en todo el programa. Esto hace que el código sea fácil de leer para otros desarrolladores.

También para mejorar la legibilidad del código, debemos usar nombres claros, comentarios útiles, una estructura lógica y mantener el código simple y consistente.

Conclusión

El programa se realizó de manera modular, para asegurarnos de que se entienda bien para su futuro uso. Se encapsularon tareas repetitivas, como la entrada de datos numéricos, en la función linea1(), que valida las entradas para asegurar que sean correctas. Las variables estudiantes y calificaciones son globales dentro de la clase Program, mientras que las funciones como AddStudent() y ListStudent() manejan variables locales, evitando errores y mejorando la claridad.

El menú se optimizó utilizando un switch en lugar de condicionales if, facilitando la gestión de opciones. Se implementaron Console.Clear() y Console.ReadKey() para mejorar la estética del programa, manteniendo la salida limpia entre funciones. se mejoró la estética utilizando una única línea WriteLine() con “\n” para crear saltos de línea, en lugar de usar múltiples llamadas a WriteLine().