

UNIVERSIDAD MARIANO GÁLVEZ DE GUATEMALA
CAMPUS JUTIAPA

Catedrático: Ing. Ruldin Ayala

Programación I

Laboratorio 3



Equipo Integrado por:

José Mario Rosales Palma - 0905-24-17488

Jhony Abraham de León Pérez - 0905-24-22282

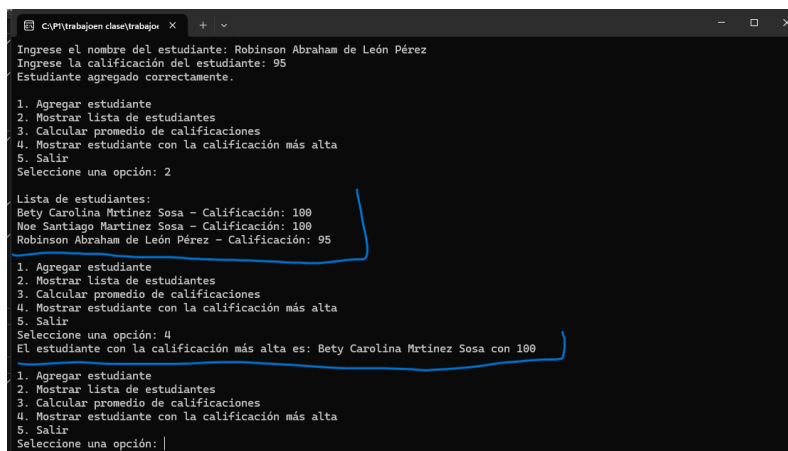
Jutiapa 20 de Febrero de 2025

Análisis Programa Estudiantes

Este es un programa de registro de estudiantes que nos permite ingresar el nombre completo del estudiante, la calificación obtenida, nos muestra la lista de estudiantes ingresados, calcula el promedio de calificaciones general y muestra el estudiante con la calificación más alta.

Se realizó utilizando la sentencia While que sirve para ejecutar en bucle un conjunto de instrucciones hasta que se cumpla una condición determinada en este caso el ingreso de los datos y los procedimientos matemáticos para mostrar el promedio y la calificación más alta.

El programa muestra una sintaxis no tan compleja pero eficaz al momento de realizar lo que el usuario le pide, a mi parecer los aspectos que se deben modificar para un mejor muestreo al usuario es la representación de el estudiante con la calificación más alta ya que si dos tiene las misma calificación solo muestra al primero ingresado en lugar de los dos.



```
C:\PI\trabajos en clase\trabajos >
Ingrese el nombre del estudiante: Robinson Abraham de León Pérez
Ingrese la calificación del estudiante: 95
Estudiante agregado correctamente.

1. Agregar estudiante
2. Mostrar lista de estudiantes
3. Calcular promedio de calificaciones
4. Mostrar estudiante con la calificación más alta
5. Salir
Seleccione una opción: 2

Lista de estudiantes:
Bety Carolina Martinez Sosa - Calificación: 100
Noe Santiago Martinez Sosa - Calificación: 100
Robinson Abraham de León Pérez - Calificación: 95

1. Agregar estudiante
2. Mostrar lista de estudiantes
3. Calcular promedio de calificaciones
4. Mostrar estudiante con la calificación más alta
5. Salir
Seleccione una opción: 4
El estudiante con la calificación más alta es: Bety Carolina Martinez Sosa con 100

1. Agregar estudiante
2. Mostrar lista de estudiantes
3. Calcular promedio de calificaciones
4. Mostrar estudiante con la calificación más alta
5. Salir
Seleccione una opción: |
```

Como sugerencia puedo proponer que en el programa exista un límite sobre las calificaciones, por ejemplo que aparezca de 60 a 0 desaprobados y de 61 a 100 aprobados, asignando esto podemos decir que la calificación del estudiante sea igual o menor que 60 aparezca desaprobado y si es igual o mayor que 61 aparezca aprobado, el objetivo es verificar la calificación de cada estudiante al momento de mostrar la lista de estudiantes o al agregar un nuevo estudiante y que luego muestre si el estudiante aprobó o desaprobó según su calificación.

Identificar las tareas repetitivas o que pueden encapsularse en funciones. Por ejemplo:

Agregar estudiantes.

Esta función pide el nombre y la calificación del estudiante luego agrega estos datos a las listas estudiantes y calificaciones.

Mostrar la lista de estudiantes.

Esta función nos permite ver cada uno de los alumnos que se ingresaron asimismo la calificación obtenida, al momento de programar se le asigna una variable que va a ir incrementando de valor dependiendo del número de alumnos ingresados por ejemplo el valor: `i++`.

Calcular promedios.

En esta función se realizan los cálculos matemáticos necesarios como sumar el total de calificaciones ingresadas asimismo dividirlos por el total dando un dato general.

Encontrar al estudiante con la calificación más alta.

Esta función busca la calificación más alta y el estudiante que la obtuvo, supongamos que tenemos a Juan y Pablo con calificación de Juan 95 y Pablo 70, nos aparecerá en la lista y nos dirá que Juan obtuvo la calificación más alta con 95.

Variables locales vs Variables globales

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

Jhony: Las variables locales se utilizan cuando el código no es muy complejo y cada una es de la función a la que están asignadas en cambio las variables globales se utilizan cuando se debe compartir datos entre variables.

Jmario: Usar las variables locales cuando no necesitamos que los valores se compartan entre funciones y las variables globales utilizarlas cuando los valores necesitan ser accesibles por partes del programa.

- **¿Qué datos deben ser accesibles en todo el programa?**
Los nombres de los estudiantes y las calificaciones ya que son operadas en otras funciones parte de las propias
- **¿Qué datos solo son necesarios dentro de una función específica?**
El promedio general y la calificación más alta registrada.

Modularización

Convertir el programa en funciones

- **Modulariza el programa creando funciones claras y específicas. Por ejemplo:**
 - **AgregarEstudiante()**
 - **MostrarEstudiantes()**
 - **CalcularPromedio()**
 - **MostrarEstudianteConMaxCalificacion()**
 - *(No necesariamente tienen que ser estas funciones, puedes asignarles tus propios nombres).*

Definir variables locales y globales:

- **Defina qué variables deben ser locales y cuáles deben ser globales.**
 - ¿Qué datos necesitan ser compartidos entre múltiples funciones?
el nombre, la calificación del estudiante y el promedio, para llevar un control ordenado de los datos y su posible modificación como la posibilidad de que sean utilizadas en otras funciones como la asignación de cursos una función que puede ser añadida para mayor control en las calificaciones donde necesitaremos el nombre de los alumnos ingresados.
 - ¿Qué datos sólo son relevantes dentro de una función específica?
la suma de calificaciones, la cantidad de estudiantes, calificación actual.

Preguntas Guía:

Responde las siguientes preguntas en tu análisis:

1. **¿Qué ventajas tiene dividir el código en funciones?**
 - o Al dividir el código en funciones se pueden separar el trabajo complejo en subtarear que puedan ser más comprensibles asimismo se mejora la estructura del código.
2. **¿Por qué es importante limitar el uso de variables globales?**
 - o porque puede afectar el rendimiento del código y no los puede dar una buena claridad y seguridad.
3. **¿Cómo se puede mejorar la legibilidad del código?**
 - o Declarando bien la estructura y variables del código para esto es necesario tener un flujo claro de ejecución siendo simple, conciso y expresivo.

Mejoras adicionales:

- **Validación de Entradas del Usuario:**
 - o **La validación de datos:** se debe modificar la parte donde el usuario ingresa las notas ya que si por algún error este escribe la cifra en letras el programa se romperá y tocará iniciar de nuevo esto se puede dejar en un rango numérico de 0 a 100.

Ejemplo:

```
Console.WriteLine("Ingrese la calificación del estudiante: ");  
double calificación = double.Parse(Console.ReadLine());  
if (double.TryParse(Console.ReadLine() out Calificación))  
else  
(!double.TryParse(Console.ReadLine(), out calificación) || calificación < 0 ||  
calificacion >100 )
```