UNIVERSIDAD NACIONAL DE LOJA

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Asignatura:

Teoría de la Programación

Unidad: 2

Tema:

Cuadro comparativo entre las estructuras repetitivas.

Docente:

I ng. Lissette Geoconda López Faicán

Estudiante:

Valeria Idaly Agila Gomez

1. Estructuras repetitivas

Las estructuras repetitivas permiten ejecutar un bloque de código mientras se cumpla una condición. Los ciclos for, while y do...while se usan según la necesidad de repetir el código un número determinado de veces o hasta que cambie la condición.

Tabla comparativa

Tipo	Estructura	Uso

For	**for**(inicialización; condición; incremento/decremento) {
	// instrucciones
	}
		La estructura for es un **bucle que repite un bloque de código mientras la condición sea verdadera**, utilizado principalmente cuando se conoce **de antemano** cuántas veces se repetirá. Primero se inicializa un contador, luego se evalúa la condición antes de cada iteración, y finalmente se aplica el incremento o decremento. Si la condición es falsa desde el inicio, el bloque no se ejecuta. Se usa comúnmente para recorrer arreglos, listas o ejecutar un proceso un número específico de veces [1].

While	**while**(condición) {
	// instrucciones
	}
		La estructura while es un bucle que repite un bloque de código mientras la condición sea verdadera, evaluando la condición antes de cada ejecución. Si la condición es falsa desde el inicio, el bloque no se ejecuta ninguna vez. Se utiliza cuando no se conoce el número exacto de repeticiones y la continuidad depende de una condición dinámica, como leer datos hasta que el usuario ingrese un valor específico [2].

Do...While	**do**
	{
	// instrucciones
	// instrucciones
	} **while(**condición);
		La estructura do...while es un **bucle que repite un bloque de código mientras la condición sea verdadera**, pero la condición se evalúa **después** de ejecutar el bloque, garantizando que el bloque se ejecute **al menos una vez** incluso si la condición inicial es falsa. Se utiliza comúnmente para menús interactivos o procesos que deben ejecutarse primero y luego verificar si continúan [3].

2. Ejercicio

- **Planteamiento del problema**

Problema:

Se desea calcular el promedio de un grupo de alumnos a partir de sus calificaciones. Esto permitirá conocer el rendimiento general del grupo de manera rápida y confiable.

Datos de entrada:

1. Un número entero n ($1 \leq n \leq 1000$), que representa la cantidad de alumnos del grupo.
    - **Nota:** n debe estar dentro del rango permitido; si no, el programa debe manejar el error.

2. **n** números reales, cada uno con un solo decimal, que representan las calificaciones de cada alumno (valores entre 0 y 10).
    - Cada calificación debe ser válida; si se ingresa un valor fuera del rango, debe solicitarse nuevamente.

Datos de salida:

- Un número real con **2 decimales**, que representa el promedio del grupo.

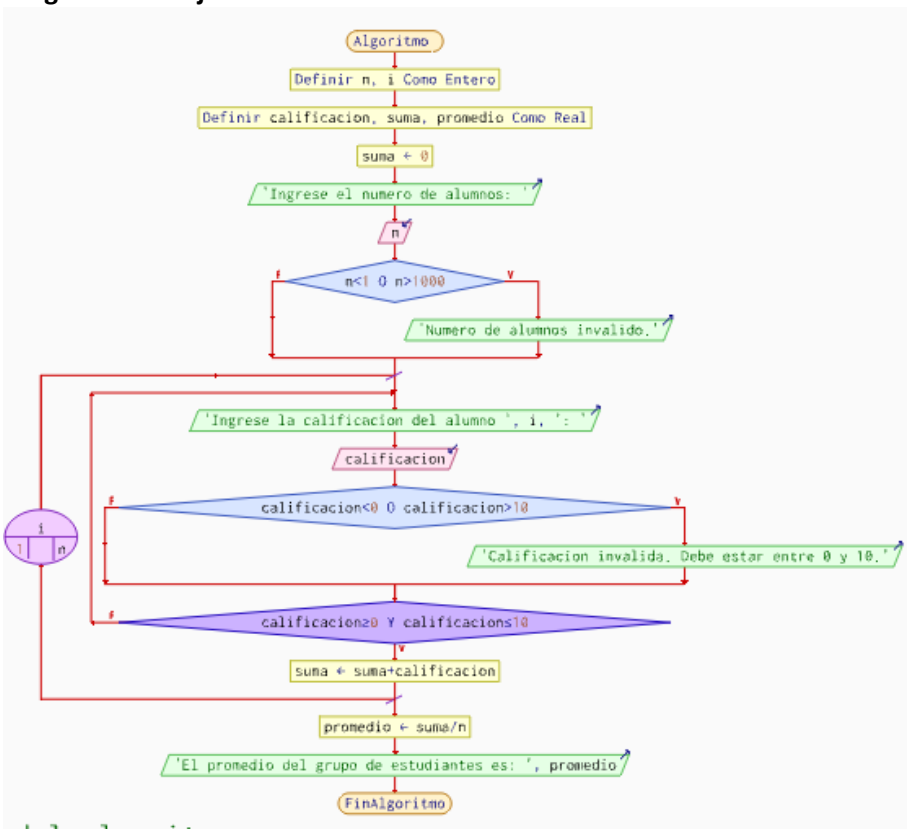
Restricciones:

- $1 \leq n \leq 1000$
- $0 \leq$ calificación ≤ 10
- Cada calificación tiene un solo decimal

Procedimiento a seguir:

1. Leer el número de alumnos n.
2. Validar que n esté dentro del rango permitido.
3. Inicializar una variable acumuladora para la suma de calificaciones.
4. Para cada alumno:
    - Leer su calificación.
    - Validar que esté entre 0 y 10.
    - Sumar la calificación.
5. Calcular el promedio dividiendo la suma total entre n.
6. Imprimir el promedio con **exactamente 2 decimales.**

- **Diagrama de flujo**



- **Código en C**

```c
#include <stdio.h>

int main() {
    //Definir variables
    int n;
    double calificacion, promedio;
    double suma = 0;
    // Entrada: número de alumnos
    printf("Ingrese el numero de alumnos: ");
    scanf("%d", &n);

    // Validar que n esté entre 1 y 1000
    if (n < 1 || n > 1000) {
        printf("Numero de alumnos invalido.\n");
        return 1;
    }
    // Entrada: calificaciones de los alumnos
    for (int i = 0; i < n; i++) {
        do {
            printf("Ingrese la calificacion del alumno %d : ", i + 1);
            scanf("%lf", &calificacion);
            // validar calificación
            if (calificacion < 0 || calificacion > 10) {
                printf("Calificacion invalida. Debe estar entre 0 y 10.\n");
            }
        } while (calificacion < 0 || calificacion > 10); // Repetir hasta que sea válida

        suma = suma + calificacion;
    }
    // Calcular promedio
    promedio = suma / n;
    // Imprimir promedio con 2 decimales
    printf("El promedio del grupo de estudiantes es: %.2f\n", promedio);
    return 0;
}
```

- **Compilación**

```
PS D:\Desktop\lenguaje C> gcc promedioup.c -o promedioup
PS D:\Desktop\lenguaje C> .\promedioup.exe
Ingrese el numero de alumnos: 5
Ingrese la calificacion del alumno 1 : 9.2
Ingrese la calificacion del alumno 2 : 7.3
Ingrese la calificacion del alumno 3 : 8.5
Ingrese la calificacion del alumno 4 : 10.0
Ingrese la calificacion del alumno 5 : 11
Calificacion invalida. Debe estar entre 0 y 10.
Ingrese la calificacion del alumno 5 : 4.9
El promedio del grupo de estudiantes es: 7.98
PS D:\Desktop\lenguaje C>
```

3. Conclusiones

Se concluye lo siguiente:

- Las estructuras repetitivas son fundamentales en programación, ya que permiten ejecutar tareas de manera eficiente y ordenada, evitando la duplicación de código y facilitando la automatización de procesos, como el cálculo de promedios de un grupo de alumnos. Esto demuestra cómo los ciclos simplifican el trabajo del programador al manejar grandes cantidades de datos de forma sistemática.
- Su uso adecuado asegura que las operaciones se realicen la cantidad correcta de veces y permite validar datos de forma sistemática, garantizando resultados precisos y confiables en cualquier programa. Además, evidencia la importancia de planificar correctamente la lógica del programa antes de su implementación.
- Comprender y aplicar correctamente los ciclos fortalece la capacidad de diseñar algoritmos claros y lógicos, evidenciando cómo las estructuras repetitivas son esenciales para la resolución de problemas complejos, el desarrollo del pensamiento computacional y la adquisición de habilidades analíticas necesarias en la programación moderna.

4. Bibliografía

[1] M. Goin, Caminando junto al Lenguaje C, Editorial UNRN, 2022. Disponible en: https://editorial.unrn.edu.ar/index.php/catalogo/346/view_bl/62/lecturas-de-catedra/26/caminando-junto-al-lenguaje-c?tab=getmybooksTab&is_show_data=1

[2] J. E. Guerra Salazar, M. V. Ramos Valencia, y G. E. Vallejo Vallejo, Programando en C desde la práctica problemas resueltos, Puerto Madero Editorial, 2023. Disponible en: https://dialnet.unirioja.es/servlet/libro?codigo=933288

[3] E. N. Figueroa Piscoya, I. Maldonado Ramirez, y R. C. Santa Cruz Acosta, *Fundamentos de programación, Un enfoque práctico*, Biblioteca Nacional del Perú, 2021. Disponible en: https://www.academia.edu/103779068/FUNDAMENTOS_DE_PROGRAMACI%C3%93N_Un_enfoque_pr%C3%A1ctico

Declaración de uso de IA

Se declara que, para la elaboración de este informe, se utilizó la inteligencia artificial únicamente como apoyo en la redacción y organización del contenido.