



Universidad
Nacional
de Loja

FACULTAD DE LA ENERGÍA, LAS INDUSTRIAS Y
LOS RECURSOS NATURALES NO RENOVABLES
Carrera de Ingeniería en Sistemas / Carrera Computación

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

Carrera de Ingeniería en Computación

Teoría de la Programación

UNIDAD 2

Aprendizaje Autónomo (AA):
Actividad Individual Nro. 2

Cuadro comparativo entre las estructuras
repetitivas

Docente: Ing. Lissette Geoconda López Faicán, PhD

Alumna: Kiara Salomé Condoy Morocho

LOJA – ECUADOR
2025

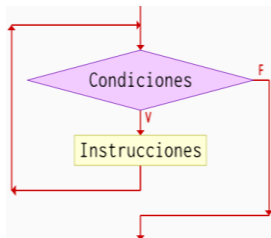
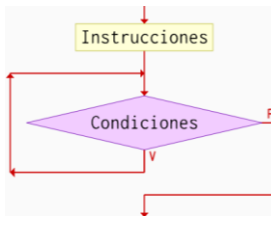
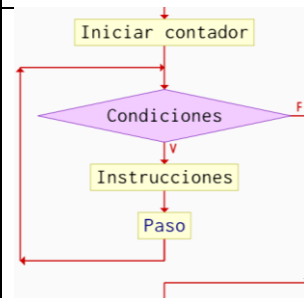
I. INTRODUCCIÓN

El aprendizaje de las estructuras de bucles condicionales repetitivos constituye un pilar fundamental en la formación de todo programador. Estas estructuras permiten automatizar tareas y resolver problemas de manera eficiente, lo que las convierte en herramientas indispensables en prácticamente todas las áreas de la programación.

En este trabajo presentaremos, a modo de resumen, un cuadro comparativo de los tres tipos de estructuras estudiadas en clase. Además, incluiremos un ejemplo práctico que evidencie su aplicación y utilidad en la resolución de problemas reales.

II. DESARROLLO

1. Tabla comparativa de las estructuras repetitivas

Aspectos\ Tipo	While	Do...While	For
Descripción	En español denominada estructura “Mientras”. Es un bucle condicional en el cual, el número de repeticiones depende de las instrucciones y los datos a procesar [1]. La condición aquí se comprueba al inicio. Es comúnmente usada cuando no se conoce la cantidad de veces que se repetirá el ciclo.	En español es más conocida como “Hacer mientras” o “Repetir”. Esta estructura comprueba al final la condición, esta al cumplirse, termina el ciclo [1]. Usada cuando se desea que las instrucciones descritas dentro del bucle se ejecuten por lo menos una vez.	La estructura “Para”, repite las acciones que se encuentran en el cuerpo del ciclo por un número específico de veces, controla la cantidad de iteraciones que se repetirán automáticamente. Requiere de un contador, y se debe especificar: el valor de inicio del contador, el valor final del contador y el valor del paso [2].
Estructura en Diagrama de Flujo			
(Sintaxis en C)	<pre>while (condition) { /* code */ }</pre>	<pre>do{ /*Code*/ }while (condition);</pre>	<pre>for (size_t i = 0; i < count; i++) { /* code */ }</pre>
Ejemplo en C Contar del 1-5	<pre>int i = 1; while (i <= 5) { printf("%d\n", i); i++; }</pre>	<pre>int i = 1; do { printf("%d\n", i); i++; } while (i <= 5);</pre>	<pre>for (int i = 1; i <= 5; i++) { printf("%d\n", i); }</pre>

2. Ejercicio

a. Planteamiento del problema

Nombre del ejercicio: Promedio

El siguiente es un programa que dada las calificaciones de un grupo con alumnos calcula el promedio del grupo.

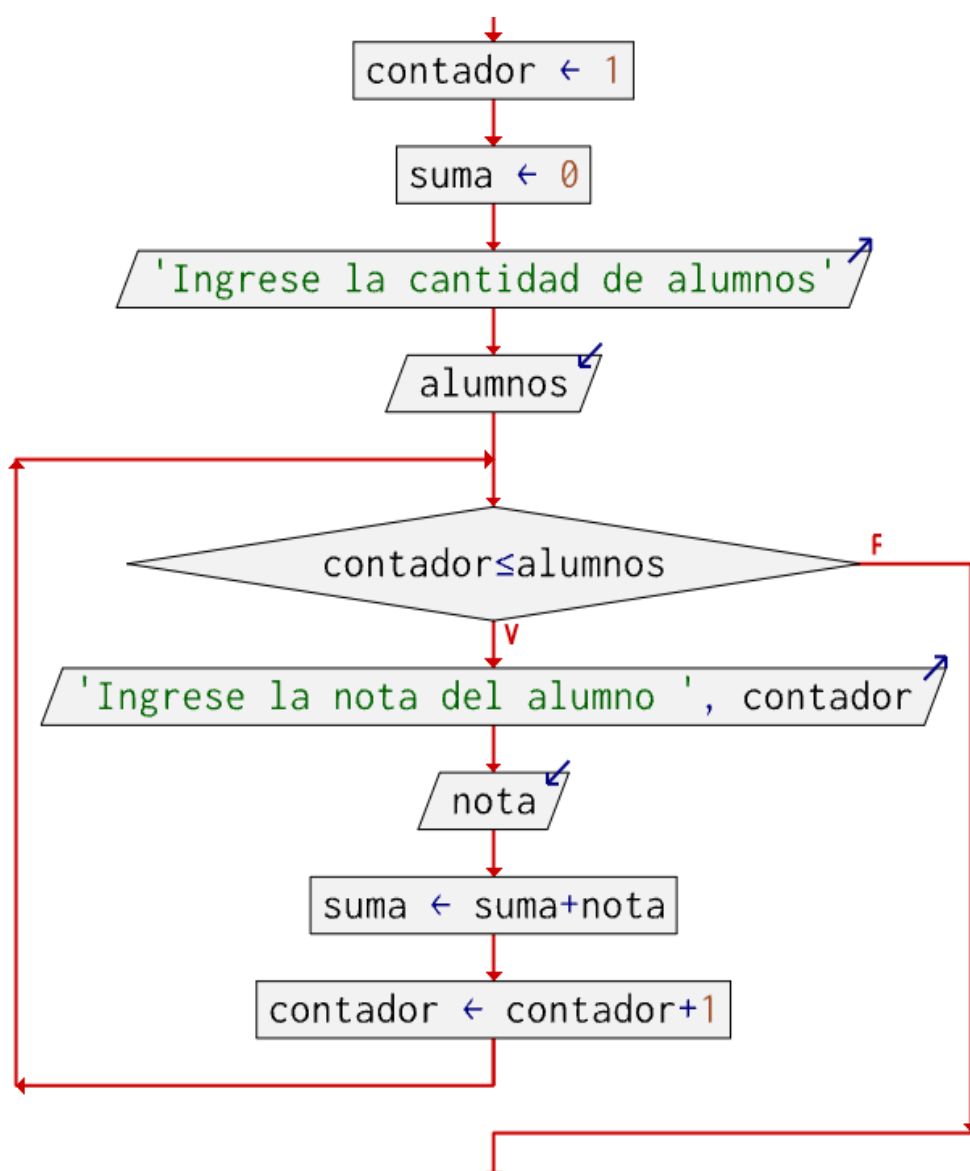
Entrada:

Un número n , representando el número de alumnos en la clase. Le siguen líneas indicando las calificaciones de cada uno de los alumnos del grupo. Las calificaciones son números reales entre 0 y 10 con un sólo decimal.

Salida:

El promedio del grupo con una precisión de 2 decimales.

b. Diagrama de flujo (simplificado)



c. Código en C

```

1  #include <stdio.h>
2
3  int main() {
4      int alumnos, contador = 1;
5      float promedio, nota, suma = 0;
6
7      printf("Ingrese la cantidad de alumnos\n");
8      scanf("%i", &alumnos);
9
10     while (contador<=alumnos) {
11         printf("Ingrese la nota del alumno %i\n", contador);
12         scanf("%f", &nota);
13         suma = nota + suma;
14         contador++;
15     }
16
17     promedio = suma/alumnos;
18     printf("El promedio total de los alumnos es: %.2f", promedio);
19
20     return 0;
21 }
```

d. Validación del código

- Pruebas de escritorio

# Alumnos	Notas	Contador	Proceso	Promedio	Salida
6	5	1	Suma= 0+5	Promedio= Suma/Alumnos Promedio=44.9/6	7.48
	9.2	2	Suma=5+9.2		
	7.3	3	Suma=14.2+7.3		
	8.5	4	Suma=21.5+8.5		
	10.0	5	Suma=30+10.0		
	4.9	6	Suma=40.0+4.9		

- Ejecución

```

PS C:\Users\ASUS\Downloads\UNL\1erCiclo\Teoria de la Programación\Unidad 2\Ejercicios C> gcc U2AA2.c -o U2AA2
PS C:\Users\ASUS\Downloads\UNL\1erCiclo\Teoria de la Programación\Unidad 2\Ejercicios C> ./U2AA2.exe
Ingrese la cantidad de alumnos
6
Ingrese la nota del alumno 1
5
Ingrese la nota del alumno 2
9.2
Ingrese la nota del alumno 3
7.3
Ingrese la nota del alumno 4
8.5
Ingrese la nota del alumno 5
10.0
Ingrese la nota del alumno 6
4.9
El promedio total de los alumnos es: 7.48
```



III. CONCLUSIÓN

Se concluye que el dominio de las estructuras repetitivas en programación resulta esencial para escribir código más eficiente y organizado. El conocimiento de las distintas variantes como “while”, “do-while” y “for” nos permite elegir la más adecuada según el problema que enfrentemos y así optimizar el desarrollo de soluciones. Además, comprender sus diferencias y aplicaciones prácticas nos brinda la capacidad de adaptarnos a diversos escenarios: desde la lectura de datos hasta el cálculo de promedios o la automatización de procesos. Esto evidencia que no solo entendemos su sintaxis, sino también su utilidad en la resolución de problemas reales.

IV. BIBLIOGRAFÍA

- [1] J. Llerena Izquierdo, *Guía de aprendizaje de programación*. 2023. Accedido: 3 de diciembre de 2025. [En línea]. Disponible en: <http://dspace.ups.edu.ec/handle/123456789/24037>
- [2] Universidad Cooperativa de Colombia, F. Y. Patiño Martínez, C. I. Torres Londoño, y P. Chica Sosa, «Elaboración de diagramas de flujo con estructuras repetitivas y arreglos», Ediciones Universidad Cooperativa de Colombia, oct. 2023. doi: 10.16925/gcgp.102.

V. DECLARACIÓN DE IA

Yo, Kiara Condoy, declaro que he utilizado la herramienta ChatGPT de OpenAI únicamente como apoyo para la redacción y revisión gramatical de la introducción del presente trabajo. Todas las ideas y reflexiones son de mi autoría. La herramienta no fue usada para generar respuestas automáticas ni sustituir mi propio razonamiento. A su vez, la bibliografía incluida fue buscada en fuentes confiables.

E. FIRMAS DE RESPONSABILIDAD DE LO ACTUADO	
Estudiante:	Firma
Kiara Salomé Condoy Morocho	