Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.



**Guía de Aprendizaje – Técnico Profesional en Programación Web**

|  |  |
| --- | --- |
| **Programa Académico** | **Curso** |
|
| Técnico Profesional en Programación Web | Fundamentos de Programación |
| **Unidad de aprendizaje** | **Resultado de Aprendizaje de la Unidad** |
| Unidad 2. Estructuras de control iterativas | RA2: Aplicar estructuras de control iterativas para resolver problemas de complejidad básica en programación. |
| **Tema** | |
| Bucles (Para, Mientras, Repetir Mientras Que) | |

# Objetivo

Comprender el concepto de iteración y las diferencias entre ejecución secuencial, condicional e iterativa, reconociendo el papel de las estructuras de control iterativas como herramientas para automatizar procesos repetitivos y aplicarlas en la construcción de algoritmos que permitan resolver problemas prácticos.

# Introducción

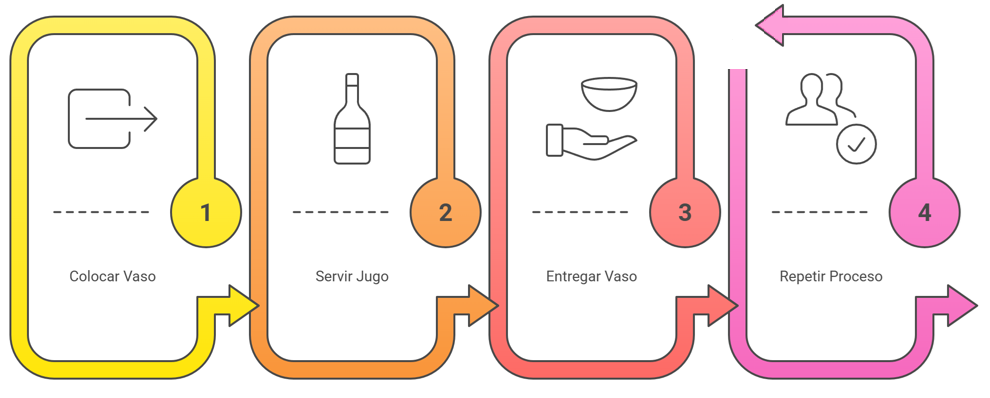
Las estructuras de control iterativas permiten repetir un conjunto de instrucciones de manera automática hasta cumplir una condición, lo que resulta esencial para optimizar procesos y reducir errores en programación. Así como en la vida cotidiana repetimos acciones como practicar una rutina de ejercicios, cocinar varias porciones de un plato o revisar correos uno tras otro, en programación utilizamos bucles como Mientras, Repetir Mientras Que y Para para ejecutar instrucciones múltiples veces sin necesidad de escribirlas repetidamente. Su aplicación facilita desde tareas sencillas, como mostrar tablas de multiplicar o calcular promedios, hasta procesos relacionados con el desarrollo web, como validar datos de forma continua o simular la adición de productos en un carrito de compras, convirtiéndose en una herramienta indispensable para resolver problemas de manera eficiente.

# Desarrollo del contenido

### Concepto de iteración

La iteración es el proceso de ejecutar repetidamente un conjunto de instrucciones de forma controlada hasta que se cumpla una condición o se alcance un número específico de repeticiones. En programación, esta técnica permite automatizar tareas repetitivas, optimizar el código y disminuir la probabilidad de errores al no tener que duplicar instrucciones innecesariamente (GeeksforGeeks, 2025).

Figura 1. Ejemplo cotidiano (con pasos): Preparar jugo para 5 personas



Este ejemplo permite comprender cómo las iteraciones están presentes en la vida cotidiana, en el caso de preparar jugo para cinco personas, las acciones se repiten una y otra vez siguiendo la misma secuencia de pasos hasta completar la cantidad deseada. De manera análoga, en programación, las estructuras iterativas como “mientras”, “para” o “repetir” permiten ejecutar un bloque de instrucciones varias veces, lo que facilita la automatización de tareas y el manejo eficiente de grandes volúmenes de datos sin necesidad de escribir código repetitivo.

### Diferencia entre ejecución secuencial, condicional e iterativa

En programación, existen tres formas principales de ejecutar instrucciones: secuencial, condicional e iterativa. La ejecución secuencial consiste en realizar las instrucciones una tras otra en el mismo orden en que aparecen, sin saltos ni repeticiones, garantizando un flujo lineal de operaciones. Por ejemplo, al preparar un café siguiendo pasos fijos —calentar agua, agregar café y servir— se sigue una secuencia establecida.

Por su parte, la ejecución condicional permite que el programa tome decisiones según una condición o criterio lógico. Esto significa que ciertas instrucciones se ejecutan solo si una condición es verdadera; de lo contrario, el flujo del programa cambia. Un ejemplo cotidiano sería decidir si agregar azúcar al café: solo se hace si la persona lo prefiere dulce.

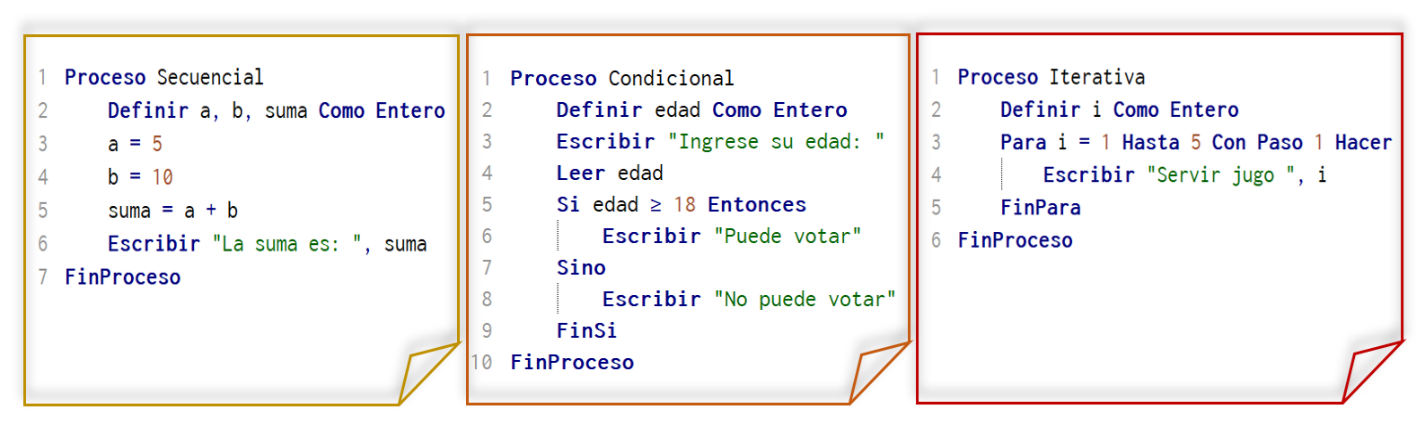
Finalmente, la ejecución iterativa implica repetir un conjunto de instrucciones varias veces hasta que se cumpla una condición o se alcance un número determinado de repeticiones. En este tipo de ejecución, el proceso se repite de manera controlada, como en el ejemplo anterior de preparar jugo para varias personas.

Figura 2. Características de los algoritmos

**Imagen que contiene Diagrama

El contenido generado por IA puede ser incorrecto.**

Para comprender mejor cómo se representan en la práctica los diferentes tipos de ejecución dentro de un algoritmo, podemos observar su aplicación en PSeInt, una herramienta que facilita la escritura y visualización de pseudocódigo. En los siguientes ejemplos se muestra cómo se verían en este entorno las estructuras secuencial, condicional e iterativa, permitiendo identificar con claridad el orden de ejecución, las decisiones lógicas y los ciclos repetitivos que caracterizan a cada una.

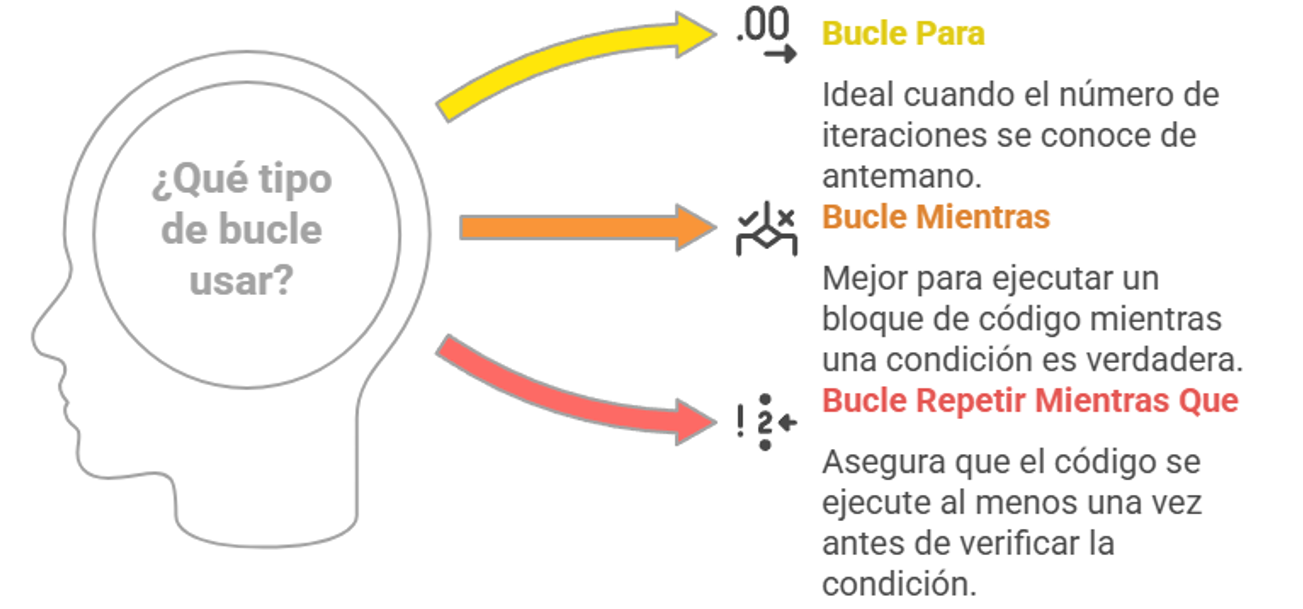
Figura 3. Ejemplos de ejecución secuencial, condicional e iterativa en PSeInt.

Como se aprecia en los ejemplos, cada estructura cumple un propósito específico dentro del diseño de algoritmos: la secuencia organiza las instrucciones de manera lineal, la condición permite tomar decisiones lógicas, y la iteración facilita la repetición controlada de procesos. Comprender y aplicar correctamente estas tres formas de ejecución es fundamental para desarrollar programas eficientes y estructurados. A partir de esta base, en el siguiente apartado se profundizará en el uso práctico de las estructuras iterativas, sus tipos y la forma en que contribuyen a la automatización de tareas dentro de un programa.

### Tipos de ciclos repetitivos

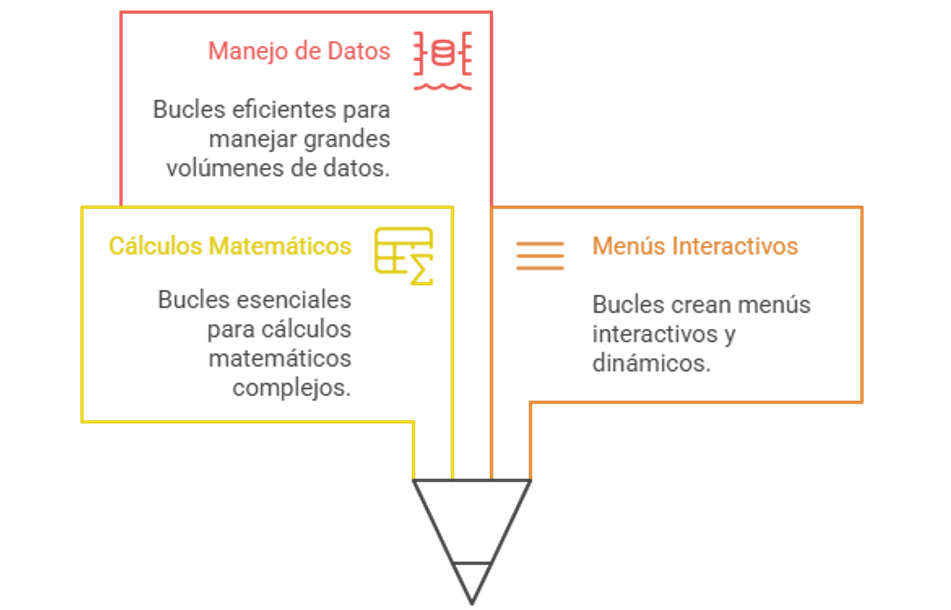
Los ciclos repetitivos son estructuras que permiten ejecutar un conjunto de instrucciones varias veces, ya sea mientras se cumpla una condición o durante un número determinado de iteraciones.

Figura 4. ¿Qué tipo de bucle usar?



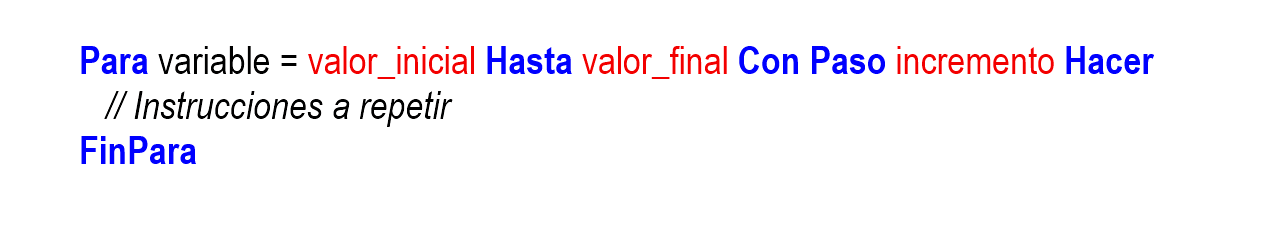
Existen tres tipos fundamentales de ciclos que constituyen la base de toda estructura iterativa: Mientras (While), Repetir Mientras Que (Do–While) y Para (For). Cada uno de ellos controla la repetición de manera diferente, dependiendo del momento en que se evalúa la condición y del tipo de tarea que se desea automatizar. Antes de analizar la estructura y funcionamiento de cada ciclo, es importante reconocer que las iteraciones no solo son un recurso técnico, sino una herramienta poderosa para optimizar distintos procesos en programación. Los bucles permiten automatizar cálculos, crear interactividad y gestionar grandes volúmenes de información de manera eficiente, como se muestra en la siguiente figura.

Figura 5. Aplicaciones prácticas de los bucles en programación.



### Ciclo Para (for)

El ciclo Para se utiliza cuando se conoce de antemano el número exacto de repeticiones que debe ejecutar un conjunto de instrucciones. Es ideal para recorrer un rango de valores, realizar conteos o ejecutar procesos controlados por un contador. Este tipo de bucle incrementa automáticamente una variable en cada iteración, facilitando tareas como generar tablas, recorrer listas o realizar cálculos repetitivos. La sintaxis general del ciclo **Para** es la siguiente:



Este tipo de ciclo se basa en **tres elementos esenciales** que son el inicializador, la finalización y el actualizador, como se indica a continuación.

Figura 6. Elementos esenciales del ciclo for.

Diagrama

El contenido generado por IA puede ser incorrecto.

A continuación, se presenta un ejemplo en PseInt que muestra cómo generar la tabla del 5:

Figura 7. Tabla de multiplicar del número 5.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

En este caso:

* El inicio es i = 1.
* La condición de fin es Hasta 10.
* El actualizador es en incremento Con Paso 1.

Por tanto, el ciclo ejecuta diez repeticiones, multiplicando el número 5 por el valor actual de i y mostrando el resultado en pantalla.

### Ciclo Mientras (While)

El ciclo Mientras permite repetir un conjunto de instrucciones mientras una condición sea verdadera. Es ideal para situaciones en las que no se conoce con exactitud cuántas veces debe repetirse el proceso, ya que la cantidad de iteraciones depende del cumplimiento de una condición lógica. A diferencia del ciclo Para, que se controla mediante un contador, el ciclo Mientras se basa en la evaluación de una expresión booleana que determina si el bloque de instrucciones continúa ejecutándose o si finaliza. Según GeeksforGeeks (2025), esta estructura se utiliza ampliamente en algoritmos de control, validación de datos y procesos donde la repetición depende del estado de una variable o entrada del usuario.

La estructura en PSeInt es la siguiente:

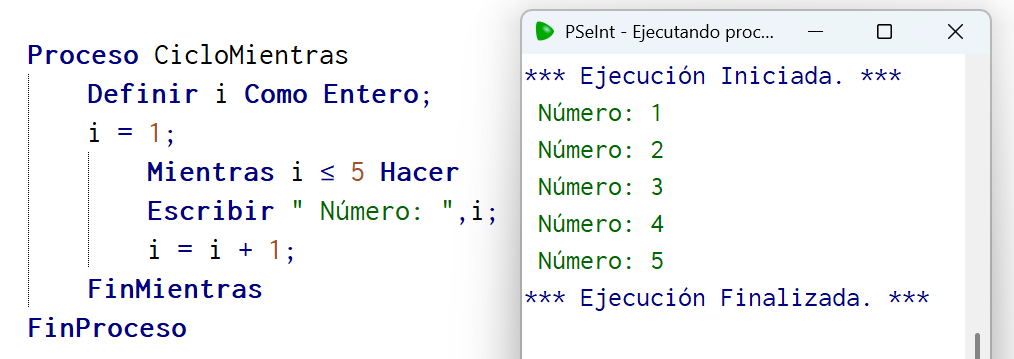
**Mientras (condición) Hacer**

**// Instrucciones a repetir**

**FinMientras**

A continuación, se muestra un ejemplo que imprime los números del 1 al 5 utilizando un ciclo Mientras:

Figura 8. Ejemplo de ciclo Mientras (While) en PSeInt y su ejecución en consola.



En este ejemplo:

* La inicialización se realiza con i = 1.
* La condición i <= 5 determina que el ciclo se repita mientras i sea menor o igual a 5.
* La actualización i = i + 1 incrementa el contador para acercarse al fin del ciclo.

El ciclo finalizará cuando i sea mayor que 5, momento en que la condición dejará de cumplirse.

### Ciclo Repetir mientras que (Do/While)

El ciclo Repetir Mientras Que ejecuta un bloque de instrucciones al menos una vez, independientemente de si la condición es verdadera o falsa en la primera evaluación. Esto se debe a que la condición se verifica al final del ciclo, no al inicio, lo que lo diferencia del ciclo Mientras, donde la evaluación se realiza antes de entrar al bloque. De acuerdo con Programiz (2024), este tipo de bucle es ideal cuando se desea asegurar una primera ejecución antes de validar la condición, como en procesos de validación de entrada de datos o en la simulación de menús interactivos que deben mostrarse al menos una vez.

La estructura general en PSeInt es la siguiente:

**Repetir**

**// Instrucciones a ejecutar**

**Hasta Que (condición)**

A continuación, se presenta un ejemplo que solicita al usuario ingresar un número mayor que cero, y repite el proceso mientras el número digitado sea menor o igual a 0.

Figura 9. Ejemplo de ciclo Repetir (do/while) en PSeInt y su ejecución en consola.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

El contenido generado por IA puede ser incorrecto.

Este tipo de ciclo es especialmente útil para validar entradas del usuario o procesos que deben ejecutarse al menos una vez antes de verificar una condición.

### 4. Cuestionario y actividad complementaria

### 4.1. Cuestionario de repaso

1. (Comprensión). En programación, una iteración se utiliza principalmente para:

A. Garantizar que una instrucción se ejecute solo una vez.

B. Repetir un conjunto de instrucciones hasta cumplir una condición.

C. Declarar variables dentro de un bloque de código.

D. Establecer una relación entre dos valores lógicos.

2. (Aplicación – contexto cotidiano). Si un programa debe preparar café para 10 personas repitiendo las mismas acciones (servir agua, agregar café, entregar taza), ¿qué tipo de ejecución se aplica?

A. Ejecución secuencial.

B. Ejecución condicional.

C. Ejecución iterativa.

D. Ejecución paralela.

3. (Código – Ciclo Para). Analiza el siguiente fragmento de código:

Proceso TablaDelTres

Definir i, resultado Como Entero;

Para i = 1 Hasta 5 Con Paso 1 Hacer

resultado = 3 \* i;

Escribir "3 x ", i, " = ", resultado;

FinPara

FinProceso

¿Cuántas veces se ejecutará el ciclo y cuál será la última línea mostrada?

A. 4 veces – “3 x 4 = 12”

B. 5 veces – “3 x 5 = 15”

C. 6 veces – “3 x 6 = 18”

D. 3 veces – “3 x 3 = 9”

✅ Respuesta correcta: B

4. (Análisis – Ciclo Mientras). ¿Qué ocurrirá en el siguiente código?

Proceso Ejemplo

Definir x Como Entero;

x = 1;

Mientras x <= 4 Hacer

Escribir x;

x = x + 1;

FinMientras

FinProceso

A. Imprimirá los números del 1 al 3.

B. Imprimirá los números del 1 al 4.

C. Se ejecutará infinitamente.

D. No imprimirá nada.

5. Analiza el siguiente código. ¿Cuál es la característica principal de este ciclo?

Proceso Validacion

Definir clave Como Entero

Repetir

Escribir "Ingrese la clave: ";

Leer clave;

Hasta Que (clave == 1234)

Escribir "Acceso concedido";

FinProceso

A. Se ejecuta solo si la condición inicial es verdadera.

B. Evalúa la condición antes de ejecutar el bloque.

C. Se ejecuta al menos una vez antes de verificar la condición.

D. No puede incluir lecturas de datos.

6. (Análisis de error lógico). ¿Qué problema presenta el siguiente código?

Proceso EjemploError

Definir i Como Entero;

i = 1;

Mientras i <= 3 Hacer

Escribir i;

FinMientras

FinProceso

A. No imprime ningún valor.

B. El bucle es infinito por falta de actualización de la variable.

C. El contador está mal inicializado.

D. El ciclo no tiene condición.

7. (Aplicación – comparación de ciclos). ¿Cuál de los siguientes ciclos es más adecuado si no se sabe cuántas veces se repetirá la acción, pero debe ejecutarse mientras una condición siga siendo verdadera?

A. Ciclo Para.

B. Ciclo Mientras.

C. Ciclo Repetir Mientras Que.

D. Condicional Si–Entonces.

8. (Código – comprensión de salida). ¿Cuál será la salida del siguiente programa?

Proceso ConteoDescendente

Definir i Como Entero;

Para i = 5 Hasta 1 Con Paso -2 Hacer

Escribir i;

FinPara

FinProceso

A. 5, 4, 3, 2, 1

B. 5, 3, 1

C. 1, 3, 5

D. 5, 2

9. (Razonamiento – condición booleana). En un ciclo Mientras, ¿qué sucede si la condición nunca se cumple desde el inicio?

A. El ciclo se ejecuta al menos una vez.

B. El ciclo no se ejecuta en absoluto.

C. El ciclo genera un error de sintaxis.

D. El programa se detiene abruptamente.

### Bibliografía

García Santillán, I. (Ed.). (2014). Fundamentos de programación usando PSeInt. Unidad de Producción y Difusión Científica y Académica.

GeeksforGeeks. (2025, enero 8). Iteration statements in programming. GeeksforGeeks. <https://www.geeksforgeeks.org/dsa/iteration-statements-in-programming/>

Programiz. (2024). Do while loop in programming. Recuperado de <https://www.programiz.com/cpp-programming/do-while-loop>

Trejos Buriticá, O. I. (2023). Lógica de programación. Ediciones de la U.

Trejos Buriticá, O. I. (2021). Lógica de programación: solucionario en pseudocódigo. Ejercicios resueltos. Ediciones de la U.