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** | |
| Control de ciclos: centinelas, contador, acumulador | |

# Objetivo

Desarrollar competencias en el manejo de técnicas de control de bucles, comprendiendo el uso de centinelas, contadores y acumuladores, mediante ejemplos prácticos que faciliten la resolución eficiente de problemas de iteración en programación.

# Introducción

En la programación, los bucles permiten ejecutar una serie de instrucciones repetidamente, pero en muchos casos es necesario controlar o dar seguimiento a lo que ocurre dentro de esas repeticiones. Para ello se emplean tres mecanismos fundamentales: el contador, el acumulador y el centinela.

Estos elementos permiten determinar cuántas veces se ejecuta un ciclo, cuándo debe finalizar y cómo almacenar resultados parciales durante el proceso. Su aplicación es esencial en tareas como el cálculo de promedios, el registro de datos, la validación de entradas o la creación de menús interactivos, siendo herramientas indispensables para el desarrollo de algoritmos eficientes y estructurados.

# Desarrollo del contenido

### Control de ciclos con contador

El contador es una variable que registra el número de iteraciones realizadas dentro de un ciclo. Se inicializa antes de entrar al bucle y se actualiza en cada repetición (generalmente sumando o restando una unidad) para limitar o monitorear la ejecución. Su uso es esencial cuando el número de repeticiones es conocido o puede acotarse razonablemente (Trejos Buriticá, 2023). En el diseño de algoritmos, el patrón inicializar → verificar → actualizar evita bucles infinitos y facilita tareas como conteos, recorridos y control de posiciones en arreglos (García Santillán, 2014).

Figura 1. Estructura lógica (componentes)

Imagen que contiene Diagrama

El contenido generado por IA puede ser incorrecto.

El funcionamiento del contador dentro de un ciclo sigue una secuencia lógica que garantiza el control adecuado del número de repeticiones. Este proceso incluye la inicialización de la variable, la definición de un límite, la evaluación de la condición, la ejecución del bloque de instrucciones y la actualización del contador.

Figura 2. Ciclo controlado por contador

Diagrama, Esquemático

El contenido generado por IA puede ser incorrecto.

En la siguiente anterior se presenta de manera visual el flujo de pasos que componen un ciclo controlado por contador, mostrando cómo cada etapa contribuye al desarrollo ordenado y finito del algoritmo. A continuación, se presenta un ejemplo en PSeInt que demuestra la aplicación práctica de este proceso dentro de un ciclo controlado por contador.

Figura 3. Ejemplo práctico del uso de un contador en PSeInt y su salida en consola

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

A continuación, se muestra el recorrido a las líneas de código desde donde se inicializa el contador, detallando la respuesta o salida en cada línea de iteración.

Figura 4. Ejemplo práctico prueba de escritorio

Tabla

El contenido generado por IA puede ser incorrecto.

El programa ejecuta cinco iteraciones (para los valores 0, 1, 2, 3 y 4). En cada repetición, el valor de contador se incrementa en 1 y se muestra en pantalla hasta que deja de cumplirse la condición contador < 5. Luego el ciclo finaliza.

### Control de ciclos con acumulador

El acumulador es una variable que permite sumar, restar o combinar valores dentro de un ciclo, almacenando un resultado parcial que se actualiza en cada iteración. Su función es conservar el total de una operación que se repite varias veces, como el cálculo de promedios, la suma de números o la acumulación de montos (García Santillán, 2014).

A diferencia del contador, que solo registra el número de repeticiones, el acumulador guarda un valor que depende de los datos procesados dentro del ciclo. Trejos Buriticá (2023) señala que esta técnica es esencial para realizar operaciones matemáticas repetitivas, pues permite construir resultados progresivos mientras se ejecuta el algoritmo.

Figura 5. Estructura lógica del acumulador

Diagrama

El contenido generado por IA puede ser incorrecto.

Antes de implementar un acumulador en un algoritmo, es fundamental inicializar la variable que guardará los resultados, normalmente con el valor 0, ya que servirá como base para las operaciones que se realicen dentro del ciclo. En cada iteración, el acumulador se actualiza sumando un nuevo valor mediante la expresión:

**acumulador = acumulador + nuevo\_valor**

De esta manera, el acumulador conserva el total parcial de las operaciones a medida que el ciclo avanza. El siguiente ejemplo en PSeInt muestra cómo este proceso se aplica de forma práctica: el acumulador inicia en cero, se modifica en cada repetición sumando los datos ingresados por el usuario y finalmente muestra el resultado total al finalizar el ciclo. Esta estructura es esencial en programas que realizan sumas, cálculos promedio o consolidación de datos.

Figura 6. Ejemplo uso de acumulador en PSeInt.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

En este programa, la variable suma actúa como acumulador y se inicializa en 0, en cada iteración del ciclo Para, se solicita un número y se agrega a la suma total. Al finalizar el ciclo, el programa muestra la suma total de los cinco valores ingresados. A continuación, una prueba de escritorio:

Figura 7. Prueba de escritorio uso de acumulador.

Tabla

El contenido generado por IA puede ser incorrecto.

### Control de ciclos con centinela

El centinela es una técnica utilizada para controlar la finalización de un ciclo mediante un valor especial que indica cuándo detener la repetición. A diferencia del contador y del acumulador, en este caso no se conoce de antemano cuántas veces se ejecutará el bucle; en su lugar, el ciclo continúa hasta que el usuario o el programa introduzca un valor determinado que actúa como señal de parada (Trejos Buriticá, 2021).

De acuerdo con García Santillán (2014), el uso de un centinela permite procesar cantidades variables de datos sin necesidad de fijar un límite previo, lo que lo hace muy útil en tareas como la lectura de registros, ingreso de información por teclado o cálculos que dependen de la decisión del usuario.

Figura 8. Estructura lógica del centinela

Diagrama

El contenido generado por IA puede ser incorrecto.

El siguiente ejemplo en PSeInt permite observar cómo se aplica la técnica del centinela dentro de un ciclo. En el programa que se muestra en la figura siguiente, el número -1 actúa como valor centinela, indicando el fin del proceso repetitivo. La variable suma cumple la función de acumulador, registrando los valores ingresados mientras no se digite el centinela. Una vez el usuario introduce -1, la condición **numero <> -1** deja de cumplirse y el programa finaliza mostrando el resultado total acumulado.

Figura 9. Ejemplo uso de centinela en PSeInt.

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

El contenido generado por IA puede ser incorrecto.

Durante la ejecución del programa, el usuario ingresa diferentes valores numéricos que se van sumando progresivamente. El ciclo continúa activo mientras el número ingresado sea distinto de -1, permitiendo observar cómo el acumulador va almacenando los resultados parciales en cada iteración. Una vez se introduce el valor centinela, el ciclo se detiene y el programa muestra en pantalla la suma total. En la siguiente tabla se ilustra el comportamiento del algoritmo paso a paso mediante una prueba de escritorio.

Figura 10. Prueba de escritorio uso de acumulador.

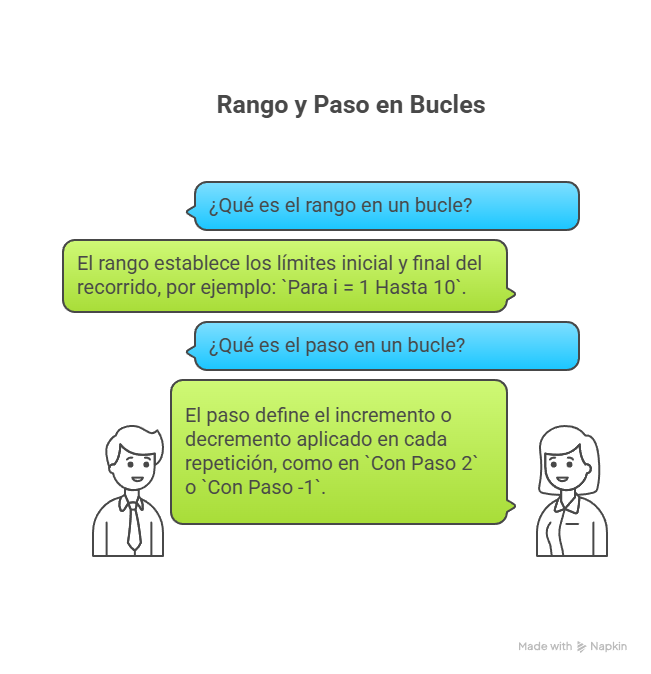
Tabla

El contenido generado por IA puede ser incorrecto.

### Rango y paso de la iteración

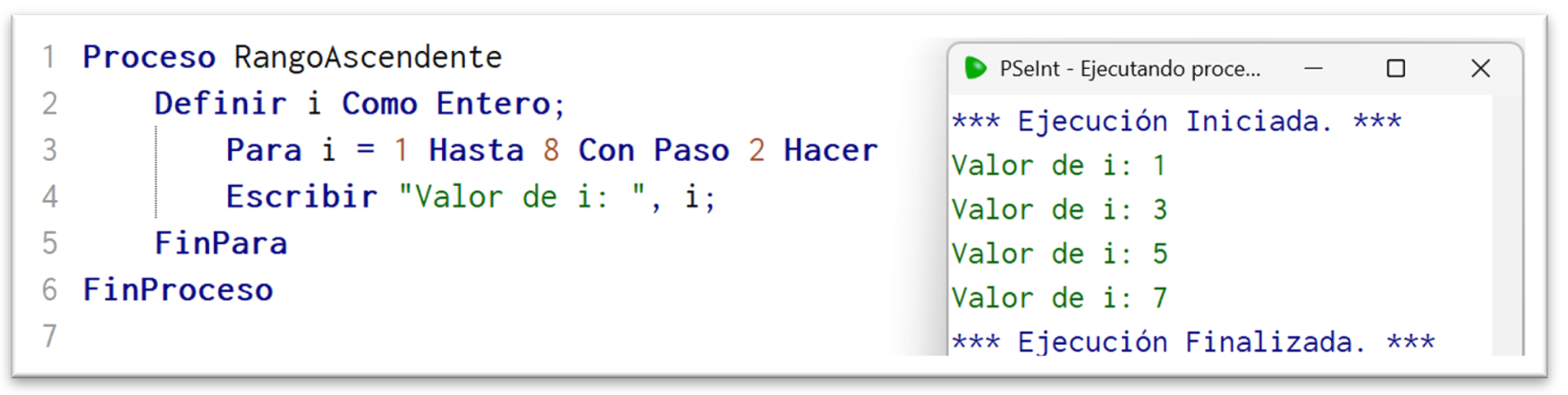
En los ciclos controlados, como el Para (for), es posible definir explícitamente un rango de valores que la variable de control recorrerá, así como el paso con el que avanzará en cada iteración. Estos dos parámetros determinan cuántas veces se ejecutará el bloque de instrucciones y con qué variación numérica lo hará (Trejos Buriticá, 2023).

Figura 11. Rango y paso de la iteración



De esta manera, el programador puede controlar la frecuencia de ejecución del ciclo y adaptarlo a distintas necesidades, ya sea avanzar en secuencias crecientes, decrecientes o con intervalos específicos.

Figura 12. Ejemplo rango ascendente.



En el ejemplo anterior, el rango va del 1 al 8 y el paso es 2, por lo que el ciclo imprime solo los valores impares. El inicio es i = 1, la salida esperada son los números:1, 3, 5, 7, 9.

De forma similar al rango ascendente, es posible definir ciclos que recorran los valores en sentido descendente, es decir, desde un número mayor hacia uno menor. Para lograrlo, se establece un paso negativo, que indica que la variable de control debe disminuir en cada iteración. En el ejemplo que se muestra a continuación, el ciclo inicia en 10 y termina en 2, disminuyendo su valor en 2 unidades por repetición, lo que permite generar una secuencia decreciente controlada.

Figura 13. Ejemplo rango descendente.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

### Bucles anidados

Un bucle anidado es una estructura en la que un ciclo se encuentra dentro de otro, permitiendo realizar repeticiones múltiples y combinadas. En este tipo de estructuras, el bucle externo controla el número total de iteraciones principales, mientras que el bucle interno se ejecuta completamente en cada una de las repeticiones del externo.

De acuerdo con Trejos Buriticá (2023), los bucles anidados son muy útiles cuando se requiere recorrer estructuras bidimensionales, como matrices, tablas o gráficos, y cuando es necesario combinar procesos repetitivos dependientes entre sí. Sin embargo, su uso debe planificarse cuidadosamente, ya que el número total de ejecuciones aumenta proporcionalmente con la cantidad de repeticiones internas.

Figura 14. Ejemplo de ciclo Mientras (While) en PSeInt y su ejecución en consola.

Diagrama

El contenido generado por IA puede ser incorrecto.

El siguiente ejemplo en PSeInt muestra cómo funciona un bucle anidado, es decir, un ciclo dentro de otro. En este caso, el bucle externo controla el número de tablas a generar, mientras que el bucle interno se encarga de realizar las multiplicaciones correspondientes a cada tabla. De esta forma, el programa recorre dos niveles de iteración: por cada valor de la variable i, la variable j ejecuta todas sus repeticiones antes de que i avance a la siguiente vuelta. En este ejemplo, el bucle externo controla el número de tablas (de 1 a 2), mientras que el bucle interno genera los productos del 1 al 3 para cada número.

Figura 15. Ejemplo Tabla de multiplicar del 1 y del 2 en PSeInt.

Interfaz de usuario gráfica, Texto, Aplicación, Word

El contenido generado por IA puede ser incorrecto.

### 4. Cuestionario y actividad complementaria

### 4.1. Cuestionario de repaso

1. Ciclo con contador. Observa el siguiente fragmento de código e indica la salida esperada:

Proceso ContadorBasico

Definir i Como Entero;

i = 1;

Mientras i <= 3 Hacer

Escribir "Iteración ", i;

i = i + 1;

FinMientras

FinProceso

A. Iteración 1, Iteración 2, Iteración 3

B. Iteración 0, Iteración 1, Iteración 2

C. Iteración 1, Iteración 2, Iteración 3, Iteración 4

D. Iteración 2, Iteración 3, Iteración 4

2. Acumulador. Analiza el siguiente código. ¿Qué valor se mostrará al final del programa?

Proceso SumaNumeros

Definir i, suma Como Entero;

suma = 0;

Para i = 1 Hasta 4 Con Paso 1 Hacer

suma = suma + i;

FinPara

Escribir "Resultado final: ", suma;

FinProceso

A. 6

B. 10

C. 4

D. 15

3. Ciclo con centinela. Observa el siguiente código:

Proceso SumaCentinela

Definir num, total Como Entero;

total = 0;

Escribir "Ingrese un número (-1 para terminar): ";

Leer num;

Mientras num <> -1 Hacer

total = total + num;

Leer num;

FinMientras

Escribir "Total: ", total;

FinProceso

Si el usuario ingresa: 5, 3, 2, -1, ¿cuál será el resultado final?

A. 8

B. 9

C. 10

D. 11

4. Ciclo con rango y paso. Analiza el siguiente fragmento:

Proceso RangoYpaso

Definir i Como Entero

Para i = 2 Hasta 10 Con Paso 3 Hacer

Escribir i

FinPara

FinProceso

¿Qué valores se imprimirán?

A. 2, 3, 4, 5, 6, 7, 8, 9, 10

B. 2, 5, 8

C. 2, 5, 8, 11

D. 3, 6, 9

5. Ciclo anidado. Observa el siguiente código:

Proceso TablasMultiplicar

Definir i, j Como Entero

Para i = 1 Hasta 2 Con Paso 1 Hacer

Para j = 1 Hasta 3 Con Paso 1 Hacer

Escribir i, " x ", j, " = ", i\*j

FinPara

FinPara

FinProceso

¿Cuántas líneas imprimirá este programa?

A. 3

B. 5

C. 6

D. 9

### 4.2. Actividad complementaria

Enunciado: Diseñe un algoritmo en PSeInt que permita registrar las notas finales de varios estudiantes y calcular el promedio general del grupo. El programa debe funcionar de la siguiente manera:

* Solicitar al usuario ingresar la nota de un estudiante (entre 0 y 5).
* Continuar pidiendo notas hasta que se ingrese una nota negativa.
* Utilizar un acumulador para sumar todas las notas válidas y un contador para registrar cuántas notas se ingresaron.

Al finalizar, el programa debe mostrar:

* El número total de estudiantes evaluados.
* El promedio general del grupo.

### Bibliografía

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.

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