Logotipo

El contenido generado por IA puede ser incorrecto.

**UNIVERSIDAD TECNOLÓGICA DE HONDURAS**

**CAMPUS SANTA BÁRBARA**

**ASIGNATURA:** ANÁLISIS Y DISEÑO DE ALGORITMOS

**INFORME DEL GRUPO 4: CICLO MIENTRAS**

**II PARCIAL**

**ALUMNOS:**

ALLAN ISAAC GUZMÁN MEJÍA (202420050028)

JOSUÉ ALEJANDRO MONTÚFAR ZÚNIGA(202510050077)

NAYDELIN DANELIA MORALES RODRÍGUEZ (202510050055)

SAMIA LIZETH CARDONA TRÓCHEZ (202430050036)

SUSAN ANAHÍ SABILLÓN TRÓCHEZ (202510050028)

**PRESENTADO A:**

ING/ABOG. JAIR JOSUÉ HERNÁNDEZ HERNÁNDEZ

**PERÍODO 2025-1**

**FECHA DE ENTREGA:**

LUNES 10 DE MARZO DEL 2025

**ÍNDICE**

[INTRODUCCIÓN 1](#_Toc192431439)

[OBJETIVOS 2](#_Toc192431440)

[CONCEPTOS GENERALES 3](#_Toc192431441)

[CICLO WHILE (MIENTRAS) 5](#_Toc192431442)

[EJEMPLOS PRÁCTICOS 6](#_Toc192431443)

[ERRORES COMUNES AL USAR EL CICLO WHILE (MIENTRAS) 8](#_Toc192431444)

[COMPARACIÓN CON LA VIDA COTIDIANA 11](#_Toc192431445)

[VENTAJAS DEL CICLO WHILE (MIENTRAS) 13](#_Toc192431446)

[DESVENTAJAS DEL CICLO WHILE (MIENTRAS) 15](#_Toc192431447)

[COMPARACIÓN ENTRE EL CICLO MIENTRAS Y EL CICLO PARA 17](#_Toc192431448)

[CONCLUSIONES 18](#_Toc192431449)

[BIBLIOGRAFÍA 19](#_Toc192431450)

# INTRODUCCIÓN

El presente informe tiene como propósito presentar la investigación de los estudiantes de la clase de Análisis y Diseño de Algoritmos sobre el ciclo mientras en pseudocódigo.

En algoritmos, un bucle mientras es una estructura de control que permite repetir una acción mientras se cumpla una condición determinada. Esto es útil cuando no se conoce de antemano cuántas veces se ejecutará el bloque de instrucciones.

A continuación, se presentarán más detalles sobre el tema.

# OBJETIVOS

* Definir los conceptos de decisiones y bucles en algoritmos.
* Explicar el ciclo mientras, su funcionamiento y sintaxis.
* Presentar ejemplos prácticos, errores comunes y compararlo con el ciclo para.
* Hacer la comparación con la vida cotidiana y presentar ventajas y desventajas.

# CONCEPTOS GENERALES

En el análisis y diseño de algoritmos, las decisiones y los bucles son estructuras de control fundamentales que determinan el flujo de ejecución de un programa.

**Decisiones:**

Las decisiones permiten que un algoritmo seleccione diferentes caminos de ejecución basados en condiciones específicas. Estas estructuras condicionales evalúan expresiones lógicas y, según su resultado (verdadero o falso), ejecutan bloques de código correspondientes. Las formas más comunes de estructuras de decisión son:

* **Condicional simple:** ejecuta un bloque de código si la condición es verdadera.
* **Condicional doble:** ofrece una alternativa; si la condición es verdadera, se ejecuta un bloque, de lo contrario, se ejecuta otro.
* **Condicional múltiple:** permite seleccionar entre múltiples bloques de código según el valor de una expresión.

Estas estructuras son esenciales para que los algoritmos puedan tomar decisiones y adaptarse a diferentes situaciones durante su ejecución.

**Bucles:**

Los bucles permiten la repetición de un conjunto de instrucciones mientras se cumpla una condición determinada. Son especialmente útiles para procesar colecciones de datos o realizar operaciones repetitivas. Los tipos más comunes de bucles son:

* **Bucle "mientras" (while):** repite un bloque de código mientras la condición sea verdadera.
* **Bucle "hacer-mientras" (do-while):** similar al anterior, pero garantiza que el bloque de código se ejecute al menos una vez antes de evaluar la condición.
* **Bucle "para" (for):** se utiliza cuando se conoce de antemano el número de iteraciones que se deben realizar.

Es crucial diseñar bucles que aseguren la finalización de las iteraciones para evitar bucles infinitos, que pueden llevar a que el programa no termine su ejecución.

**Importancia en el diseño de algoritmos:**

Las decisiones y los bucles son componentes esenciales en la construcción de algoritmos eficientes y efectivos. Permiten que un programa responda de manera adecuada a diferentes entradas y situaciones, y que realice tareas repetitivas de manera controlada. Un diseño adecuado de estas estructuras contribuye a la claridad, eficiencia y mantenibilidad del código.

# CICLO WHILE (MIENTRAS)

**¿Qué es un ciclo While?**

La instrucción Mientras (While) ejecuta una secuencia de instrucciones mientras una condición sea verdadera.

\*Mientras <condición> Hacer

<instrucciones>

FinMientras\*

**¿Cómo funciona?**

Al ejecutarse esta instrucción, la condición es evaluada. Si la condición resulta verdadera, se ejecuta una vez la secuencia de instrucciones que forman el cuerpo del ciclo. Al finalizar la ejecución del cuerpo del ciclo se vuelve a evaluar la condición y, si es verdadera, la ejecución se repite. Estos pasos se repiten mientras la condición sea verdadera.

Note que las instrucciones del cuerpo del ciclo pueden no ejecutarse nunca, si al evaluar por primera vez la condición resulta ser falsa.

Si la condición siempre es verdadera, al ejecutar esta instrucción se produce un ciclo infinito. A fin de evitarlo, las instrucciones del cuerpo del ciclo deben contener alguna instrucción que modifique la o las variables involucradas en la condición, de modo que ésta sea falsificada en algún momento y así finalice la ejecución del ciclo.

# EJEMPLOS PRÁCTICOS

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

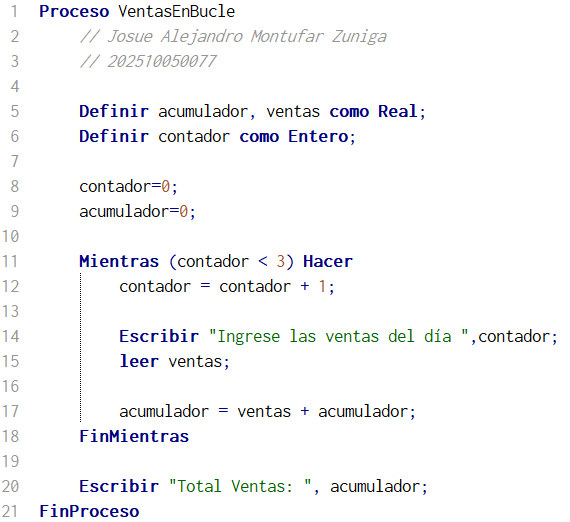
El contenido generado por IA puede ser incorrecto.**Ejemplo 1:** Sacar fotocopias

Contamos con una variable de entrada llamada “copiasNecesarias” y se define como tipo entero.

Se solicita al usuario ingresar la cantidad de hojas que desea imprimir.

El bucle mientras simula la impresión de una hoja y se actualiza la cantidad de hojas restantes. Finaliza cuando se hayan impreso todas las hojas.

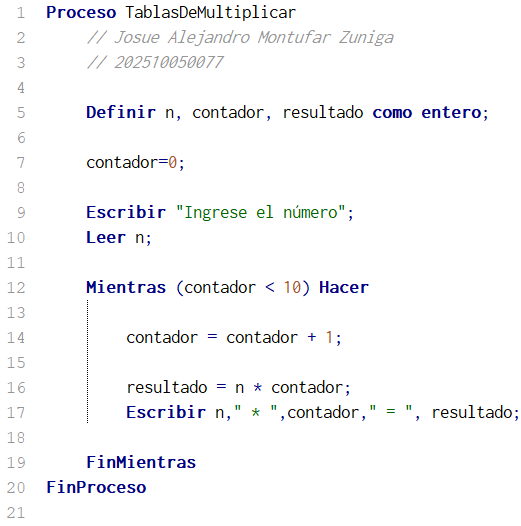
* **Contador:** Es una variable que va incrementando o decrementando un valor fijo dentro de un bucle. Su función principal es contar la cantidad de veces que ocurre una acción determinada. En este caso el contador es la variable “copiasNecesarias”.

**Ejemplo 2:** Suma de ventas

Contamos con una variable de entrada (ventas de tipo real) y dos variables de salida (acumulador de tipo real y contador de tipo entero). Luego se inicializan el contador y acumulador para evitar errores.

Se solicita al usuario ingresar las ganancias de ventas de días diferentes para luego sumarlos y escribir un total.

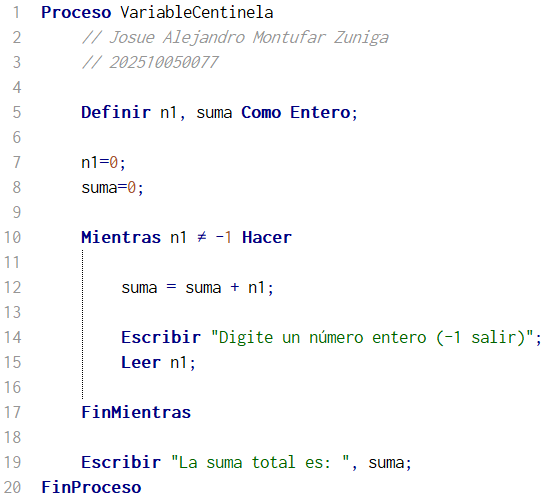
* **Acumulador:** Va aumentando o decrementando un valor variable dentro del bucle. En este caso el acumulador va sumando los valores de las ventas para luego mostrar un total al final del pseudocódigo.

**Ejemplo 3:** Tablas de multiplicar

Contamos con una variable de entrada (n) y dos variables de salida (contador y resultado), todas de tipo entero. Luego se inicializa el contador para evitar errores.

Se solicita al usuario ingresar un número para mostrar su tabla de multiplicar del 1 al 10.

Las instrucciones dentro del bucle mientras permiten realizarlo de forma dinámica y ahorrando muchas líneas de código.

**Ejemplo 4:** Sumas con variable centinela

Contamos con una variable de entrada (n1) y una variable de salida (suma), ambas de tipo entero. Se inicializan para evitar errores.

El bucle mientras permitirá que el usuario pueda ingresar un sinfín de números para luego sumarlos en el acumulador “suma”.

Se detendrá cuando el usuario digite -1 para después mostrar el resultado final.

* **Variable centinela:** Es una variable especial utilizada para darle fin a un bucle o una secuencia de datos cuando el usuario lo decida. Es muy utilizada en las estructuras de datos Mientras y Repetir.

En este caso, n1 es la variable centinela, al escribir -1 el bucle termina y nos imprime el resultado; caso contrario al ingresar otro valor el bucle no se detendrá.

# ERRORES COMUNES AL USAR EL CICLO WHILE (MIENTRAS)

* **Bucle infinito:** Este es el error más frecuente. Ocurre cuando la condición del "Mientras" siempre se evalúa como verdadera, lo que hace que el ciclo se repita indefinidamente.  
  Las causas de este error pueden ser que olvidamos actualizar la variable de control dentro del bucle, otra es una condición que siempre se mantendrá verdadera por un error lógico. Ejemplo:  
  Interfaz de usuario gráfica, Aplicación

  El contenido generado por IA puede ser incorrecto.
* Para solucionar el error es necesario hacer funcionar el contador dentro del bucle.  
  En este caso está comentado.
* **El ciclo nunca inicia:** La condición del "Mientras" se evalúa como falsa desde el principio, por lo que el código dentro del bucle nunca se ejecuta.  
  Esto puede ocurrir cuando inicializamos mal una variable o cuando nuestra condición no se cumple. Ejemplo:   
  Interfaz de usuario gráfica

  El contenido generado por IA puede ser incorrecto.
* Para solucionar el error debemos corregir el valor inicial de la variable “contador”.   
  Según la lógica del bucle hay que pasarlo de 100 a 1.
* **Olvidar inicializar la o las variables a evaluar:** Usar una variable en la condición del "Mientras" sin haberle asignado un valor inicial puede llevar a un error o a un comportamiento inesperado.  
  Cuando hablamos de inicializar nos referimos a darle un valor a la variable, debemos recordar que una variable sin inicializar se le conoce como variable nula. Ejemplo:  
  Interfaz de usuario gráfica, Texto

  El contenido generado por IA puede ser incorrecto.
* Podemos observar que la variable “contador” no está inicializada.  
  Para solucionar este error simplemente debemos darle un valor inicial a “contador”.
* **Confundir la asignación (<-) con la igualdad (=):** Esto se da comúnmente en la programación real. Usar el operador incorrecto en el ciclo mientras podría ocasionar que nunca inicie o llevar a errores de lógica. Ejemplo:  
  Interfaz de usuario gráfica, Texto, Aplicación

  El contenido generado por IA puede ser incorrecto.
* Al escribir <- en la condición de Mientras estamos haciendo una asignación.  
  Se soluciona al cambiar la asignación por la igualdad (= en PSeInt, == en programación).

**Para evitar estos errores, se recomienda:**

* Utilizar la opción de ejecución paso a paso para ver que ocurre en nuestro pseudocódigo.
* Agregar comentarios al pseudocódigo para explicar la lógica del bucle y facilitar su comprensión.
* Ir ejecutando nuestro algoritmo frecuentemente para verificar que todo va conforme a lo establecido.
* Escribir el pseudocódigo de la forma más clara posible, para evitar errores de lectura, y por lo tanto de lógica.

# COMPARACIÓN CON LA VIDA COTIDIANA

**Ejemplo 1:** Estudiar un tema.

Cuando estamos estudiando, estamos intentando comprender un tema determinado mediante la lectura repetida, de preguntas, respuestas o párrafos.

* **Condición:** ¿Me aprendí el tema?
* **Repetición:** Seguir estudiando
* **Finalización:** cuando te sientas listo para pasar al siguiente tema.

**Ejemplo 2:** Barrer un cuarto.

Cuando estamos barriendo un cuarto, seguimos un proceso en el que realizamos la acción de barrer y recoger la tierra una y otra vez hasta que sentimos que el suelo esté limpio.

* **Condición:** ¿Todavía hay suciedad?
* **Repetición:** Seguir barriendo y usando el recogedor para acumular la tierra.
* **Finalización:** Cuando ya ni haya más suciedad que recoger.

**Ejemplo 3:** Lavar la ropa

Cuando lavamos la ropa hacemos un proceso repetitivo, hasta asegurarnos que toda la ropa este limpia.

* **Condición:** ¿Queda más ropa por lavar?
* **Repetición:** Seguir lavando, pasar el jabón de barra, enjuagar y separar.
* **Finalización:** Cuando toda la ropa este limpia y lista para tender.

**Ejemplo 4:** Cocinar.

Cuando cocinamos seguimos una serie de pasos repetitivos hasta que la comida esté lista para servir.

* **Condición:** ¿La comida esta lista?
* **Repetición:** Seguir cocinando, preparar ingredientes, mezclar, cocinar a juego adecuado, probar el sabor.
* **Finalización:** Cuando la comida este bien cocida y lista para servir.

# VENTAJAS DEL CICLO WHILE (MIENTRAS)

1. **Flexible y útil cuando el número de iteraciones es incierto**

El bucle While es ideal cuando no se sabe de antemano cuántas veces se repetirá una acción. A diferencia de Para, que requiere un número fijo de iteraciones, While permite que el programa siga ejecutando un conjunto de instrucciones hasta que se cumpla una condición específica. Esto lo hace muy útil en situaciones donde las repeticiones dependen de eventos dinámicos, como la entrada de datos del usuario o la respuesta de un sistema externo.

1. **Eficiente en cuanto al uso de recursos.**

Como la condición se evalúa antes de ejecutar el bloque de código, si la condición es Falso desde el inicio, el bucle simplemente no se ejecuta. Esto evita realizar cálculos o iteraciones innecesarias, lo que puede hacer que el programa sea más eficiente. Por ejemplo, si el usuario introduce un dato válido en la primera entrada, el bucle nunca entrará en ejecución, ahorrando tiempo y recursos del sistema.

1. **Ofrece un control preciso sobre la condición de salida.**

El bucle permite que la ejecución continúe mientras una condición siga siendo Verdadero, lo que brinda un control más detallado sobre cuándo detener el ciclo. En comparación con Para, donde las repeticiones son fijas, While permite salir del bucle tan pronto como se cumpla una determinada condición. Esto es muy útil en situaciones donde se necesita monitorear continuamente un estado y terminar la ejecución en cuanto cambie.

1. **Útil para validaciones de datos y procesos interactivos**

Una de las aplicaciones más comunes del bucle While es la validación de entradas del usuario. Muchas veces, los programas necesitan asegurarse de que un usuario ingrese datos correctos antes de continuar con la ejecución. Por ejemplo, si se requiere que el usuario introduzca un número positivo, el bucle While puede seguir solicitándolo hasta que se cumpla esta condición. Esto garantiza que el programa reciba datos adecuados sin necesidad de estructuras adicionales.

# DESVENTAJAS DEL CICLO WHILE (MIENTRAS)

1. **Puede generar bucles infinitos si la condición nunca cambia**

Una de las mayores desventajas del bucle While es que, si la condición nunca se vuelve Falso, el bucle continuará ejecutándose indefinidamente, lo que puede hacer que el programa se bloquee o quede atrapado en un ciclo sin fin. Esto suele ocurrir cuando el programador olvida actualizar las variables involucradas en la condición del bucle. Por ejemplo, si un programa espera que el usuario introduzca un número mayor a 10 pero no actualiza correctamente la variable de entrada, el bucle se ejecutará para siempre.

1. **Menos controlado en comparación con el bucle Para**

Aunque While es flexible, en algunos casos puede ser menos conveniente que Para, especialmente cuando se necesita un número exacto de iteraciones. El bucle Para permite definir explícitamente cuántas veces se ejecutará un bloque de código, mientras que en While, la finalización depende de una condición, lo que puede hacer que el control sea más complejo. Si el programador no maneja adecuadamente la actualización de variables, el bucle puede ejecutarse más veces de las necesarias o incluso no ejecutarse en absoluto.

1. **Puede reducir el rendimiento si se usa incorrectamente**

Si el bucle While no está bien optimizado, puede llevar a evaluaciones innecesarias de la condición, lo que afecta el rendimiento del programa. En algunos casos, si la condición depende de operaciones costosas, como consultas a una base de datos o cálculos intensivos, evaluarla en cada iteración puede hacer que el código se vuelva ineficiente. Por esta razón, es importante asegurarse de que la condición de salida sea lo más simple y eficiente posible.

1. **Mayor complejidad en ciertos algoritmos**

Aunque el bucle While es útil para muchas situaciones, en algunos algoritmos puede hacer que el código sea más difícil de leer y mantener en comparación con Para. En problemas donde se necesita recorrer una secuencia con un número definido de pasos, Para es más intuitivo y fácil de usar. En cambio, While requiere que el programador controle manualmente las condiciones de entrada y salida, lo que puede aumentar la posibilidad de errores lógicos.

# COMPARACIÓN ENTRE EL CICLO MIENTRAS Y EL CICLO PARA

El siguiente cuadro comparativo muestra las diferencias clave entre los ciclos While y For en Pseint, considerando aspectos fundamentales en su uso y aplicación en la programación estructurada.

|  |  |  |
| --- | --- | --- |
| Aspecto | Ciclo While (Mientras) | Ciclo For (Para) |
| Condición de repetición | Depende de una condición booleana que se evalúa antes de cada iteración. | Depende de un rango predefinido de valores para la variable de control. |
| Uso recomendado | Cuando no se conoce de antemano la cantidad de repeticiones y depende de factores externos. | Cuando se conoce exactamente la cantidad de repeticiones desde el inicio. |
| Control del ciclo | Requiere inicializar, actualizar y verificar la condición manualmente. | La variable de control se maneja automáticamente dentro de la estructura del ciclo. |
| Flexibilidad | Es más flexible y permite condiciones dinámicas que pueden cambiar en tiempo de ejecución. | Es más estructurado y sigue un patrón fijo, lo que lo hace más seguro y eficiente. |

# CONCLUSIONES

* En algoritmos, las decisiones son una acción mientras que los bucles son repeticiones.
* El ciclo mientras repite instrucciones cuando su condición se mantenga verdadera.
* Dependiendo de la situación se usará el bucle Mientras o el bucle Para.
* Nosotros utilizamos el ciclo mientras en todo momento de nuestra vida, al igual que el ciclo para.

# BIBLIOGRAFÍA

* Joyanes Aguilar, L. (2003). Fundamentos de programación. McGraw Hill. España.
* Gueguen, G., & Vallecillo, A. (s.f.). Técnicas de Diseño de Algoritmos.
* Aliat Universidades. (s.f.). Análisis y diseño de algoritmos.
* <https://www.itson.mx/oferta/isw/Documents/guia_pseint_2016.pdf>
* <https://en.cppreference.com/w/c/keyword/while>
* <https://www.tutorialspoint.com/cprogramming/c_loops.htm>
* https://pseintinternet.blogspot.com/p/unidad-3.html