1. **DATOS GENERALES**

| **RESPONSABLES EQUIPO DE PRODUCCIÓN** |  | Registro revisión |
| --- | --- | --- |
| **Líder de producción** | Olga Constanza Bermudez Jaimes | |
| **Nombre Asignatura** | Pensamiento Algorítmico | |
| **Experto Disciplinar** | Luis Fernando Henao Rodriguez | Octubre 2024 |
| **Asesor Tecnopedagógico** | Elizabeth Bermúdez Díez | Octubre 2024 |
| **Diseñador Instruccional** | Ana Catalina Córdoba Sus | Octubre 2024 |
| **Validador de contenidos** | Luis Fernando Henao | Octubre 2024 |
| **Diseñador gráfico** | Daniel Mutis | Octubre 2024 |
| **Maquetador web** | Francisco Lizcano |  |

| **U1** | **Unidad 1. Fundamentos del Pensamiento Algorítmico** |
| --- | --- |

* + - 1. **INTRODUCCIÓN UNIDAD**

A través de esta unidad, se reconocerán los fundamentos teóricos del pensamiento algorítmico; además, se utilizarán estructuras lógicas en la creación de algoritmos simples, para resolver problemas específicos.

A lo largo de la unidad, se abordarán los conceptos de algoritmo y lógica, así como sus formas de representación mediante pseudocódigo y diagramas de flujo, lo cual permite analizar cada problema desde un enfoque sistémico, desarrollar soluciones adecuadas y llevarlas a la práctica mediante la implementación de algoritmos.

En resumen, se destacará la relevancia de diseñar algoritmos antes de iniciar el proceso de programación, resaltando su utilidad en la resolución de problemas y su aplicación práctica no solo en sistemas de información, sino también en campos como la administración y la educación.

**INTRODUCCIÓN PARA LA PORTADA**

**FUNDAMENTOS DEL PENSAMIENTO ALGORÍTMICO**

En esta primera unidad, se estudiarán los fundamentos esenciales del pensamiento algorítmico, destacando su rol en la resolución de problemas mediante el diseño de algoritmos estructurados. Además, se explorarán conceptos clave como los algoritmos, la lógica detrás de su construcción, y sus formas de representación a través de pseudocódigo y diagramas de flujo.

Prepárate para aplicar estos principios en diversas áreas, desde la informática hasta la administración y educación. ¡Inicia tu viaje para dominar la creación de soluciones eficientes y prácticas!

* + - 1. **SÍNTESIS UNIDAD**

| **Vincular la síntesis de la unidad1** |
| --- |

**Tabla de contenido**

[***Tema 1: Definición de algoritmo. 2***](#_heading=h.1fob9te)

[**Tema 1.1: Sistema 5**](#_heading=h.3znysh7)

[**Tema 1.2: Pensamiento Sistémico o lógico 7**](#_heading=h.2et92p0)

[**Tema 1.3: Eficiencia de los algoritmos 8**](#_heading=h.tyjcwt)

[***Tema 2: Lógica algorítmica: secuencias, condiciones, ciclos. 9***](#_heading=h.3dy6vkm)

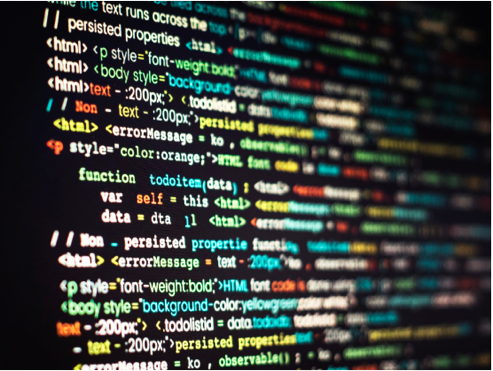
[**Tema 2.1: Metodología para la solución de algoritmos. 12**](#_heading=h.1t3h5sf)

[**Tema 2.2: Secuencias, condiciones, ciclos. 13**](#_heading=h.4d34og8)

[***Tema 3: Representación de algoritmos: pseudocódigo y diagramas de flujo. 16***](#_heading=h.2s8eyo1)

[**Tema 3.1: Representación de algoritmos: pseudocódigo y diagramas de flujo. 17**](#_heading=h.17dp8vu)

# Tema 1: Definición de algoritmo

La palabra algoritmo proviene del matemático Muhammad ibn Musa al-Khwarizmi, quien vivió en el siglo IX en Persia, e implementó métodos sistémicos que resolvían problemas matemáticos. Turing en 1936, inspiró el concepto moderno de algoritmo, a través de su Máquina de Turing y creó un modelo teórico para describir el procesamiento de los algorítmicos, sentando las bases de la ciencia computacional.

Al implementar una secuencia de pasos lógicos finitos que resuelven un problema particular y que permiten visibilizarlo de manera gráfica, se está definiendo el concepto de un algoritmo. “*Un algoritmo se espera resuelva un problema*” (Joyanes, 2008, p. 13). Si bien, entender que para cada proceso debe existir una secuencia lógica de instrucciones hace parte importante en el desarrollo de un algoritmo, también se debe tener en cuenta el hecho que este deberá resolver una situación puntual, así; el algoritmo estará diseñado con un propósito, siguiendo un procedimiento y llegando a la solución, de la manera más eficiente posible.

| **Para conocer los múltiples usos de los algoritmos en la actualidad, lo invitamos a ver los siguientes videos:** | |
| --- | --- |
|  | Magic Markers. (2021). *¿Qué son los algoritmos?* [video]. YouTube. <https://youtu.be/U3CGMyjzlvM> |
| BBC News Mundo. (2020). *Qué son los algoritmos y cómo aprenden de nosotros | BBC Mundo* [video]. YouTube. <https://youtu.be/RSJrBEhdZxw> | |

En nuestro día a día, nos enfrentamos “sin querer” o muchas veces “sin percatarnos”, a la utilización de algoritmos para resolver situaciones cotidianas como levantarse de la cama, ir a la ducha, preparar el desayuno, tomar el autobús, etc.; se está llevando a cabo una secuencia de pasos lógicos que permite tomar decisiones, seleccionar la mejor manera de llevar a cabo la actividad y, por último, solucionarla. Aquí se ha construido en cada ejemplo, un algoritmo que, aunque no se visualiza como tal por tratarse de una actividad que se ejecuta de manera casi que “automática”, el cerebro tuvo la capacidad de hacerlo y a continuación se explicará el porqué.

**Ejemplo 1:**

| **Situación problema:** ducharse.  **Solución:** terminar limpio después de la ducha.  **Procedimiento:**  Paso 1. Inicio.  Paso 2. Ingreso a la ducha.  Paso 3. Abrir la llave.  Paso 4. Regular la temperatura.  Paso 5. Remojarse.  Paso 6. Cerrar la llave.  Paso 7. Aplicar jabón.  Paso 8. Abrir nuevamente la ducha.  Paso 9. Regular nuevamente la temperatura.  Paso 10. Ducharse.  Paso 11. Cerrar la llave.  Paso 12. Fin. |
| --- |

En el **ejemplo 1**, se puede aclarar de una manera mucho más sencilla, el concepto de algoritmo, sus componentes integradores y su propósito; también se puede observar como un proceso sencillo, al llevarlo a un paso a paso, deberá determinar de manera clara y lo más precisa posible, la secuencia de instrucciones para que se resuelva en un feliz término.

Adicionalmente, se puede aumentar la complejidad del mismo, incluyendo subprocesos que permitan ejecutar acciones concretas a la hora de llevar a cabo el objetivo principal del algoritmo; por ejemplo, se citó puntualmente “aplicar jabón” pero se debió ubicar el jabón dentro de la ducha, tomarlo, aplicar el jabón y por último, volverlo a poner en su lugar, para continuar con el proceso inicial; este tipo de subrutinas, aumentan la complejidad del algoritmo y lo convierten en una secuencia detallada que permitirá no “tener pierde” a la hora de ejecutarse.

|  | Los **algoritmos cualitativos** realizan procedimientos sin la utilización de matemáticas, solamente procesos y está ligado directamente al tipo de pensamiento de la persona que lo realiza. Ejemplo: retirar dinero de un cajero automático. **Los algoritmos cuantitativos** requieren de cálculos matemáticos para su elaboración. Ejemplo: obtener los números primos del 1 al 100. |
| --- | --- |

El problema a resolver, permite establecer el objetivo específico a la hora de construir un algoritmo; si se resuelve o no, podrá considerarse que:

| **ACORDEÓN** | |
| --- | --- |
| El algoritmo es **correcto** | Si se finaliza correctamente al ingresar los datos adecuados dentro del procedimiento. |
| El algoritmo es **incorrecto** | Si la solución no finaliza como se espera al ingresar los datos correctos para llevar a cabo el procedimiento, o se obtiene como resultado una salida diferente a la esperada. |

Si el algoritmo **soluciona** el problema, se considera que es correcto.

Un algoritmo es un método para resolver problemas y para lograrlo, se puede aplicar el siguiente procedimiento sugerido:

| **ACORDEÓN** | |
| --- | --- |
| **Definición del problema** | En esta definición es relevante determinar con claridad el objetivo final del algoritmo, qué se desea resolver y qué se espera del sistema. |
| **Definición de variables de entrada y salida** | En la etapa de **definición de variables de entrada y salida** se tendrá en cuenta el resultado que se espera al finalizar el proceso e incluir los datos de entrada necesarios para que dicho proceso entregue el final esperado. |
| **Establecer la estructura de procesamiento de datos** | Al establecer la **estructura de procesamiento de datos,** se utilizará la lógica para definir el procedimiento y la manera como el algoritmo logrará llevar a cabo el procesamiento de los parámetros iniciales, para presentar, como resultado, la variable o variables de salida, esperadas. |

Las características que un algoritmo debe contemplar, son:

| **TARJETAS** | |
| --- | --- |
| Siempre debe tener un inicio. |  |
| Deberá ser preciso, contemplar el orden estructural y secuencial del proceso y establecer prioridades. |  |
| Debe estar definido, lo que a su vez traduce en que cada vez que se ejecute el algoritmo, se obtienen los mismos resultados. |  |
| Siempre debe tener un fin, deberá terminar el algún momento; es decir, un número de pasos establecido. |  |

**Ejemplo 2:**

| Un estudiante llega a la biblioteca universitaria para solicitar un libro, el administrativo del lugar consulta en el sistema el nombre del libro, la ficha técnica y determina si es posible realizar el préstamo o no; a continuación, se redacta un algoritmo que puede resolver la situación.  **Situación problema:** préstamo del libro.  **Solución**: entrega del libro al estudiante según disponibilidad.  **Procedimiento:**   1. Inicio. 2. Recibe solicitud de préstamo por parte del estudiante. 3. El administrativo examina el repositorio y consulta la disponibilidad del libro. 4. Si la biblioteca cuenta con el libro para el préstamo, el administrativo lo hace efectivo; de lo contrario, rechaza la solicitud. 5. Fin. |
| --- |

|  | Si desea conocer más ejemplos de algoritmos, consulte el libro: Joyanes Aguilar, L. (2006). *Programación en C++: algoritmos, estructuras de datos y objetos* (2 ed.). McGraw-Hill.  <https://elibro.net/es/ereader/tecnologicadeloriente/50088?page=54> |
| --- | --- |

No solo se puede hablar de algoritmo a la hora de comprender su relación directa con los sistemas computacionales, también es importante tener claridad con los siguientes conceptos:

| **ACORDEÓN** | |
| --- | --- |
| **Proceso** | Se puede entender como un conjunto de instrucciones que “manipulan” los datos ingresados al sistema, para entregar como resultado, la solución esperada al problema. |
| **Procedimiento o subprograma** | Permite ejecutar cuantas veces sea necesario, un proceso dentro de un programa, ahorrando tiempo y mejorando la eficiencia en el sistema; no es lo mismo realizar 10 veces una multiplicación, a llamar un subprograma que realiza la multiplicación y entrega el resultado. |
| **Programa** | Concepto elaborado por Von Neumann en 1946 y lo define como “conjunto de instrucciones que sigue la computadora, para alcanzar un resultado específico”. |

En conclusión, un algoritmo se puede resumir en tres partes:

1. Entrada.
2. Proceso.
3. Salida.

## 

## Tema 1.1: Sistema



Para entender un algoritmo se debe hablar de un sistema, entendiéndose este último como un conjunto de elementos que tienen una relación directa entre sí y que interactúan de manera coordinada, para obtener un objetivo específico.

Un sistema se caracteriza por:

| **ACORDEÓN** | |
| --- | --- |
| **Entrada (*Input*)** | Datos que requiere el algoritmo para funcionar. |
| **(Procesamiento** | Aquí se lleva a cabo la “transformación” de los datos de entrada y se logra el objetivo del algoritmo. |
| **Salida (*Output*):** | Resultado final del sistema. |
| **Retroalimentación (*Feedback*)** | Una vez terminado el algoritmo, se pueden hacer ajustes para mejorar su eficiencia y calidad, mejorando su desempeño y funcionamiento. |

**Ejemplo 3:**

| Si se desea redactar una carta en un procesador de texto, deberá utilizar el teclado del ordenador para ingresar los caracteres (entrada); el sistema las incluirá dentro del procesador de texto (procesamiento) y por último tendrá la carta lista (salida).  El proceso suena sencillo debido a que se ha convertido en una tarea habitual para todos, pero al describir los pasos, se puede caracterizar un algoritmo de la siguiente manera:   1. Entrada: cadenas de caracteres del teclado. 2. Procesamiento: utilización de la CPU para recibir las variables de entrada en código binario, procesarlas y presentarlas en la pantalla, como cadenas de texto entendibles para el usuario. 3. Salida: mensaje de la carta. |
| --- |

Del **ejemplo 3**, cabe resaltar que el funcionamiento del teclado está ligado al *hardware* y *software* del computador; no se podría establecer una conexión directa entre el teclado y el monitor, sin haber tenido un tratamiento desde la unidad central de procesamiento; todo el equipo de cómputo en conjunto, permite que el usuario pueda “operar” sin contratiempos, cada dispositivo; estos, a su vez, suministran los datos requeridos para que la CPU pueda establecer prioridades y dar un orden a cada petición del usuario y, por último, representar las peticiones del usuario a través del monitor.

**Figura 1**

*Interacción usuario-computador*







Se establece una interacción directa Usuario – Computador – Teclado - CPU – Mensaje; es decir, se requiere que cada elemento que integra el sistema, funcione correctamente, para obtener los resultados esperados. Para clasificar los sistemas, según su nivel de interacción con el entorno, es importante conocer los tipos.

Los tipos de sistemas son:

| **TARJETAS** | | |
| --- | --- | --- |
| **Sistemas abiertos** | Interactúan con el entorno y los cambios los afectan directamente y deben adaptarse para mejorar o sobrevivir.  **Ejemplos:** empresas, hospitales. |  |
| **Sistemas cerrados** | Operan de manera autónoma y tienen poca o nula interacción con el entorno; generalmente se ejecutan en entornos controlados.  **Ejemplos:** reloj mecánico, invernadero. |  |
| **Sistemas naturales** | Son los que hacen referencia a la naturaleza y operan sin intervención humana.  **Ejemplos:** bosque, sistema nervioso. |  |
| **Sistemas artificiales** | Creados por humanos con una finalidad específica, facilitan tareas o proporcionan servicios.  **Ejemplos:** sistemas operativos, banca. |  |

Se puede inferir que los sistemas se diferencian, según su grado de interacción con el entorno que los rodea y su propósito; los **sistemas abiertos** deben adaptarse a los cambios constantes y los **sistemas cerrados** se llevan a cabo en ambientes mucho más controlados; los **sistemas naturales** respetan las leyes de la naturaleza y permanecen en un equilibrio constante con su entorno y los **sistemas artificiales** fueron diseñados para cumplir una función específica y apoyar tareas cotidianas. Cada sistema tiene una finalidad; entender su funcionamiento y conocer sus características, permite administrarlos de manera más eficiente.

Para el desarrollo de los algoritmos, es esencial el pensamiento sistémico y el lógico, porque así se podrán establecer, no solo el paso a paso para encontrar una solución (**sistémico**), sino también el procedimiento que se deberá llevar a cabo (**lógico**), para que dicha solución esté acorde con lo esperado como parámetro de salida en el algoritmo.

## Tema 1.2: Pensamiento sistémico o pensamiento lógico

Al abordar los algoritmos, es indispensable hablar de dos tipos de pensamientos: el pensamiento sistémico y el pensamiento lógico; ambos se requieren para comprender el sistema y cómo se deberá realizar el procesamiento de los datos, para encontrar una solución eficiente y acorde con las necesidades de su entorno.

Si bien, ambos tipos de pensamiento tienen corrientes diferentes, en los sistemas de información se relacionan y coexisten de manera natural, entendiéndose sus diferencias.

**Tabla 1**

*Pensamiento sistémico y pensamiento lógico*

| Aspecto | Pensamiento Sistémico | Pensamiento Lógico |
| --- | --- | --- |
| Definición | Se analiza el todo y la interacción del sistema con cada una de sus partes. | Método secuencial estructurado que sigue unas reglas definidas y cuyo objetivo es encontrar una conclusión válida. |
| Enfoque | General, holístico: se debe concentrar la atención en el todo, en cada detalle, en el grado de interacción de cada elemento. | Lineal, secuencial: sigue pasos lógicos estructurados y secuenciales, para resolver el problema planteado. |
| Ejemplos | Gestiones administrativas.  Análisis de ecosistemas. | Programación de algoritmos.  Demostraciones matemáticas. |
| Naturaleza | No lineal, admite retroalimentación, múltiples procesos para una misma solución. | Lineal, sigue una estructura clara desde el inicio hasta el fin. |
| Retroalimentación | En busca de la optimización constante del sistema. | Se llega a la solución y no se requiere ajuste del sistema o alguna optimización. |



El **pensamiento lógico** se relaciona con matemática estructural y permite, a partir de la aplicación de secuencias funcionales, encontrar la solución a un problema; el **pensamiento sistémico** requiere de la lógica matemática, en muchos casos para contemplar una parte de la solución del problema inicial, pero también contempla la interacción de cada parte del sistema, para no solo resolver el sistema, sino también optimizarlo, mejorarlo y hacerlo cada vez más eficiente.

## Tema 1.3: Eficiencia de los algoritmos

Al realizar un algoritmo, se pueden encontrar diferentes recursos que llegan a la misma solución; el concepto de “Eficiencia” hace referencia a tomar el algoritmo para que de manera más óptima, llegue a la solución; tendría la desventaja que para encontrar el algoritmo eficiente, se verá en la necesidad de construir diferentes prototipos del mismo algoritmo, aunque el hecho de realizar un algoritmo, puede implicar un esfuerzo: hacerlo para resolver la misma situación desde diferentes perspectivas algorítmicas, debe considerarse si se desea seleccionar el más eficiente.

La calidad de un algoritmo se mide según su eficiencia; esta se puede traducir como el uso de los recursos requeridos para llevar a cabo la tarea que se ejecuta. Así, en términos computacionales, un algoritmo de calidad deberá consumir en la medida de lo posible, la menor cantidad de recursos para su ejecución (memoria, procesador, etc.).

Otro aspecto clave con respecto a la eficiencia, es el tiempo; este factor se verá afectado por la cantidad de datos de entrada, la calidad del código, equipo donde se ejecuta el programa y su complejidad. Diseñar un algoritmo debe contemplar siempre la complejidad; a menor complejidad, mayor será la eficiencia, puesto que se traduce en menor tiempo de ejecución y menor consumo de recursos en el equipo que se ejecute.

# Tema 2: Lógica algorítmica: secuencias, condiciones, ciclos.

Al desarrollar un algoritmo, no se podrá simplemente escribir el mismo y esperar que el resultado sea el esperado; se debe realizar un análisis del problema, definir las variables, estructurar el procedimiento y tener claro cuál será el resultado a obtener, una vez que el algoritmo termine; para lograrlo, se requiere del uso de la lógica algorítmica. El análisis del problema es fundamental y se convierte en el insumo necesario a la hora de construir un algoritmo, no solo eficiente, sino también eficaz.

Antes de comenzar con la estructuración de un algoritmo, se deben contemplar algunos aspectos claves como:

| **ACORDEÓN** | |
| --- | --- |
| **Tipos de datos numéricos** | Números enteros y reales, positivos o negativos y no llevan punto decimal.  +200 –10 125 483 119 –550 |
| **Tipos de datos alfanuméricos** | Datos de tipo carácter o cadena de caracteres, contienen letras, signos, símbolos especiales, etc. Se deben escribir dentro de comillas simples (‘’) o dobles (“”), pueden contener números, pero no se pueden realizar operaciones matemáticas con ellos. |
| **Tipo carácter** | ‘h’ ‘d’ ‘l’ |
| **Tipo cadena** | ‘universidad’ ‘$Clave\_24’ ‘José13’ |
| **Memoria** | Espacio reservado dentro del sistema, que contiene un nombre y una variable. Al establecer como parámetro año 2024, se puede inferir que la variable es año y su valor es 2024. |
| **Constante** | Es una variable almacenada en memoria y cuyo contenido no cambiará durante todo el proceso de ejecución del algoritmo. |
| **Variable** | Diferente a la constante, la variable si cambia “varía” su valor, según los procedimientos que se lleven a cabo dentro del algoritmo. |
| **Identificador** | Corresponde al nombre asignado a una variable en la memoria y deberá respetar las dos reglas siguientes:   1. Iniciar con una letra. 2. Seguido de caracteres, como letras o números, sin acentuación.   **Ejemplo:** *edad, num\_cedula, direccion*.  Se deberá utilizar la nemotecnia para llamar cada identificador, de manera que su nombre sea representativo dentro del algoritmo. Por ejemplo, si se habla de una operación matemática como la multiplicación, se podrá utilizar *num1, num2, multiplicacion*. |
| **Expresiones** | Los operadores y operandos componen una expresión. El operador determina el tipo de operación a realizar y los operandos serán los valores a procesar. |

Los tipos de operadores que existen son:

**Tabla 2**

*Tipos de operadores*

| Tipo de operador | Operador | Operación |
| --- | --- | --- |
| Asociativos | () |  |
| Aritméticos | ^  \*  /  + - | Potencia  Multiplicación  División  Suma  Resta |
| Relacionales | =  <> o ><  <  >  <=  >= | Igual  Diferente  Menor que  Mayor que  Menor o igual que  Mayor o igual que |
| Lógicos | NOT  AND  OR | Negación  Conjunción  Disyunción |

|  | Importante recordar que, al **negar** un valor, este cambiará de verdadero a falso y viceversa; en la **conjunción** si ambos valores son verdaderos, se obtendrá un valor verdadero; en cualquier otro caso, el resultado será falso y, en la **disyunción,** si ambos valores son falsos, el resultado será falso; en cualquier otro caso, será verdadero. |
| --- | --- |

**Tabla 3**

*Tipos de expresiones*

| Tipo de expresión | Ejemplo | Resultado |
| --- | --- | --- |
| Aritméticas | 2 \* (4-3) + 4 | 6 |
| Relacionales | 7=5  5<3  9>8 | Falso  Falso  Verdadero |
| Lógicas | (4-1) > (5-3)  (3-2) = (5-4) | Verdadero  Verdadero |
| Lógicos | NOT  AND  OR | Negación  Conjunción  Disyunción |

Las reglas a seguir a la hora de resolver expresiones relacionales, son:

1. Primero resolver los paréntesis.
2. Encontrar valores de verdad para cada expresión.
3. Aplicar jerarquía de operadores lógicos.

**Ejemplo:**

(5\*2)>(2+3) AND (1+4)<(2^2)





|  | Se puede poner en práctica lo aprendido, desarrollando los ejercicios propuestos “***Aplicación de la jerarquía de operadores***” del libro: Gaxiola Pacheco, C. G. & Flores Gutiérrez, D. L. (2008). *Metodología de la programación con pseudocódigo enfocado al lenguaje C.* Plaza y Valdés.  <https://elibro.net/es/ereader/tecnologicadeloriente/75935?page=24> |
| --- | --- |

## Tema 2.1: Metodología para la solución de algoritmos



Recordar que una computadora es un equipo y no va a funcionar correctamente si no se le dan las instrucciones precisas para ello, es importante a la hora de desarrollar un algoritmo; se debe tener presente el paso a paso secuencial, lógico y ordenado, que le asegurará el éxito en su solución.

Existe una metodología sugerida para la solución de los algoritmos y se puede establecer de la siguiente manera:

| **TARJETAS** | |
| --- | --- |
| Comprender el problema. |  |
| Generar un algoritmo inicial. |  |
| Seleccionar una estrategia de solución para el algoritmo. |  |
| Implementar un lenguaje de programación. |  |
| Verificar la eficiencia del algoritmo. |  |
| Realizar pruebas y verificar el funcionamiento del algoritmo. |  |
| Optimización final. |  |

|  | Para ampliar el concepto de metodología para la solución de algoritmos, puede consultar el libro: Gaxiola Pacheco, C. G. & Flores Gutiérrez, D. L. (2008). *Metodología de la programación con pseudocódigo enfocado al lenguaje C*. Plaza y Valdés.  <https://elibro.net/es/ereader/tecnologicadeloriente/75935?page=27> |
| --- | --- |

## Tema 2.2: Secuencias, condiciones, ciclos

| documento icono | ¿Cuáles son los componentes claves en la lógica algorítmica? Lo invitamos a consultar el PDF **Secuencias-condiciones-ciclos**, para conocerlas. |
| --- | --- |

# Tema 3: Representación de algoritmos: pseudocódigo y diagramas de flujo.

Como se mencionó anteriormente, un algoritmo es un conjunto de instrucciones lógicas ordenadas que cuentan con un inicio, un fin y que resuelven un problema. Encontrar estos pasos requiere de estructurarlos de manera tal que permitan ser eficientes y eficaces, resolviendo el problema de la manera más práctica posible. Una manera práctica de comprender la estructura funcional de un algoritmo, es la siguiente:

**Tabla 4**

*Estructura de un algoritmo*

| Estructura | Descripción | Ejemplo |
| --- | --- | --- |
| Solicitar datos | Un algoritmo debe estar en la capacidad de solicitar “datos” que pueden ser valores o cadenas de caracteres. | Solicitar edad;  Solicitar año;  Solicitar dirección; |
| Imprimir datos | Un algoritmo debe estar en la capacidad de “presentar” los datos a través de impresión por pantalla.  Es importante recordar que una variable contiene un valor y una cadena de texto contiene un conjunto de valores o caracteres. Por ejemplo, Valor 10, cadena “Henry”. | Imprimir mensaje  Imprimir “Error en el valor ingresado”  Imprimir variable  Imprimir “Su edad es” +edad |
| Evaluar condiciones | Proceso que le permite al algoritmo obtener una respuesta (F o V) y de acuerdo con ella realizar un procedimiento.  La evaluación de condiciones se lleva a cabo a través de la verificación del orden, longitud, tamaño, etc. | If (edad>18)  Desplegar “Usted es mayor de edad”  else  Desplegar “Usted es menor de edad” |
| Ejecutar operaciones | El algoritmo debe estar en la capacidad de realizar operaciones aritméticas y guardar el resultado en una variable. | edad\_actual = fecha\_actual - fecha\_nacimiento  descuento = vlr\_articulo\*10% |

Una vez se ha logrado consolidar la estructura del algoritmo, en donde se tienen presente las variables de entrada, el procedimiento que se debe realizar y el resultado esperado, se puede plasmar a través del pseudocódigo y un diagrama de flujo el proceso que debe ejecutar el algoritmo para su solución; verificar a través de la comprobación y ajustar si es necesario.

## 

## Tema 3.1: Representación de algoritmos: pseudocódigo y diagramas de flujo.

**3.1.1 Pseudocódigo**

El pseudocódigo permite escribir de manera ordenada, la secuencia de pasos lógicos que el algoritmo debe ejecutar; es universal, lo que implica su comprensión y lectura sin requerir un lenguaje de programación formal para su escritura. Se considera natural o semiformal, se utiliza para dar lógica al algoritmo, antes de su construcción.

|  | El pseudocódigo debe ser comprensible para personas sin conocimientos técnicos en programación, dado que debe priorizar la claridad sobre la sintaxis. |
| --- | --- |

Las características del pseudocódigo son:

| **ACORDEÓN** | |
| --- | --- |
| **Simple y claro** | Utilice palabras simples y claras, inicio, fin, mientras. |
| **Libre de rigidez sintáctica** | Libre de rigidez sintáctica propia de los lenguajes de programación, es decir; no sigue las reglas propias de un lenguaje. |
| **Estructura lógica** | Utiliza secuencias, condiciones y ciclos dentro de su estructura. |
| **Facilidad de traducción** | Una vez se ha terminado de redactar el algoritmo, se podría traducir fácilmente a cualquier lenguaje de programación, gracias a su fácil lectura y comprensión. |
| **Universal** | Se debe comprender fácilmente debido a su simplicidad y claridad. |

La estructura del pseudocódigo está conformada por:

**Figura 2**

*Estructura del pseudocódigo*





Para comprender mejor la estructura del pseudocódigo, es necesario definir cada paso:

|  | 1. Todo algoritmo debe tener un **inicio** y un **fin**. 2. Las variables deben ser **declaradas** (se deben asignar los espacios en memoria, para cada variable, porque serán utilizadas por el código a la hora de realizar los procedimientos propios del algoritmo). Este paso es importante pues requiere de atención para evitar errores en la ejecución o validación del algoritmo. 3. La **secuencia** determina el orden lógico que debe seguir el algoritmo. 4. Las **condiciones** permiten la toma de decisiones dentro del ciclo normal del algoritmo; así, a través de la evaluación de expresiones lógicas, se determina el paso a seguir dentro del código. 5. Los **ciclos** permiten crear estructuras repetitivas que ahorran líneas de código y agilizan el procesamiento de las operaciones dentro del código. |
| --- | --- |

| **Para ampliar los conceptos sobre pseudocódigo y conocer algunos ejemplos, lo invitamos a ver los siguientes videos:** | |
| --- | --- |
|  | Aprende y Verifica. (2022). *¿Qué es un pseudocódigo? | Explicación rápida y sencilla* [video]. YouTube. <https://youtu.be/Q1z34G3fTUM> |
| Aprende y Verifica. (2022). *Ejemplo de un pseudocódigo y diagrama de flujo ⏐ Explicación RÁPIDA Y SENCILLA* [video]. YouTube. <https://youtu.be/lByEQkmKRis> | |

**Ejemplo - Operador matemático. Suma de dos números**

Para realizar la suma de dos números, se requiere el número 1, número 2, un operador matemático (+) y el proceso resultante deberá ser almacenado dentro de una variable suma.

Siempre se debe dar inicio y fin al algoritmo, además de declarar las variables; continuando con el ejemplo, el pseudocódigo resultante sería:

**Tabla 5**

*Pseudocódico operador matemático*

| Pseudocódigo | Descripción |
| --- | --- |
| Inicio  Leer numero1  Leer numero2  suma = numero1 + numero2  Imprimir “El resultado de la suma es”, suma  Fin | Inicio  Se lee el valor del número 1  Se lee el valor del número 2  Se realiza la operación suma  Se muestra en pantalla el resultado de la suma  Fin |

El ejemplo anterior cumple con las características del pseudocódigo; es simple, claro, lógico, cuenta con una sintaxis libre de lenguajes de programación y es universal; fácilmente se comprende el objetivo del código escrito, sin ser programador.

Si se desea aumentar el nivel de complejidad del ejemplo 1, se puede incluir una validación que indique cuál de los dos números es mayor.

**Ejemplo - Operador relacional. Verificación del mayor de dos números**

Para realizar la verificación del mayor de dos números, se requiere el número 1, número 2, un operador relacional (>) y el proceso resultante deberá ser presentado en pantalla.

**Tabla 6**

*Pseudocódigo operador relacional*

| Pseudocódigo | Descripción |
| --- | --- |
| Inicio  Leer numero1  Leer numero2  si (numero1 > número2)  Imprimir “El número 1 es mayor”  SiNo  Imprimir “El número 2 es mayor”  Fin | Inicio  Se lee el valor del número 1  Se lee el valor del número 2  Se realiza la comparación entre los dos números  Si (n1>n2) Se muestra en pantalla “el n1 es mayor”  SiNo  Se muestra en pantalla “el n2 es mayor”.  Fin |

En el ejemplo anterior, queda clara la forma como funciona, el operador relacional.

Al verificar si se cumple que el número 1 es mayor al número 2, se presenta en pantalla el mensaje “el número 1 es mayor”; de lo contrario, el número 2 sería mayor al número 1 y el mensaje sería “el número 2 es mayor”.

|  | Como ejercicio práctico, puede asignar valores a cada número, seguir la secuencia de pasos y utilizar el operador relacional. La impresión en pantalla dependerá de los números que utilice. |
| --- | --- |

**Ejemplo 3 - Utilización de ciclo**

Obtener el resultado de la multiplicación de un número **x** hasta el 10.

**Tabla 7**

*Pseudocódigo ciclo mientras*

| Pseudocódigo | Descripción |
| --- | --- |
| Inicio  contador = 1;  Leer numerox;  Mientras contador <=10 haga  Imprimir numerox \* contador  contador = contador +1  Fin | Inicio  Se declara e inicia la variable contador  Se lee el valor del número x  Se realiza la verificación (contador<=10)  Se muestra en pantalla el resultado de (numerox \* contador)  Se incrementa la variable contador en una unidad  Fin |

Del ejemplo anterior, pueden surgir algunos interrogantes:

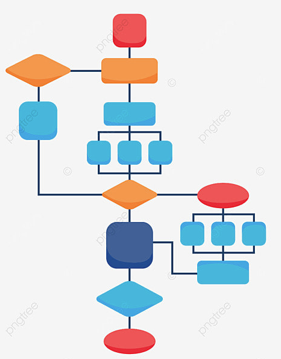
| **TARJETAS** | |
| --- | --- |
| ¿Qué puede pasar si el contador no se inicializa? | sondeo icono |
| ¿Qué pasaría si no se incrementa el contador? | sondeo icono |
| ¿Qué pasa en el momento que el contador sea igual a 10? | sondeo icono |

Las ventajas del pseudocódigo son:

| **ACORDEÓN** | |
| --- | --- |
| **Claridad** | Al momento de desarrollar el pseudocódigo, se concentra en expresar la lógica del programa sin especificaciones propias de lenguajes técnicos. |
| **Versatilidad** | Se puede adaptar a cualquier lenguaje de programación. |
| **Facilita la comunicación** | Cualquier persona puede entender la finalidad del programa. |
| **Ahorra tiempo** | Permite encontrar errores antes de pasar a escribir el código. |

En conclusión, el pseudocódigo, permite escribir un programa de manera semiformal, enfocándose en la lógica del algoritmo y cuyo propósito es facilitar la implementación de un lenguaje de programación.

|  | Se puede ampliar el concepto sobre pseudocódigo en el libro: Gaxiola Pacheco, C. G. & Flores Gutiérrez, D. L. (2008). Metodología de la programación con pseudocódigo enfocado al lenguaje C: (ed.). Plaza y Valdés (México).  <https://elibro.net/es/ereader/tecnologicadeloriente/75935?page=90> |
| --- | --- |

**3.1.2 Diagrama de flujo**

Un diagrama de flujo corresponde a la representación gráfica de un proceso y se utiliza en programación, para crear algoritmos en los cuales se emplean representaciones gráficas o símbolos estandarizados del proceso; ilustran paso a paso y de manera sencilla la lógica que debe seguir el programa; su uso es amplio como por ejemplo, la toma de decisiones, gestión de procesos, entre otros.

Las características de un diagrama de flujo son:

| **Vincular la infografía de la Unidad 1** |
| --- |

| documento icono | Para profundizar sobre los elementos asociados a los diagramas de flujo, lo invitamos a consultar el PDF **Simbología, ventajas y desventajas de los diagramas de flujo** |
| --- | --- |

* + - 1. **CRONOGRAMA DE ACTIVIDADES**

| **Actividad de aprendizaje** | **Descripción de la Actividad** | **Criterio de Evaluación** | Porcentaje evaluativo |
| --- | --- | --- | --- |
| Foro Temático | Foro de discusión: “**Reconocer estructuras básicas en los algoritmos**”.  Este foro tiene como objetivo que el estudiante reconozca las estructuras básicas a partir de ejemplos de algoritmos (Secuencias, condiciones, ciclos) a través de ejemplos prácticos. Busca fomentar la participación activa y lograr un primer acercamiento sobre la temática de la unidad, además intercambiar ideas que permitan crear algoritmos sencillos que puedan ser utilizados en diferentes situaciones.  Pregunta orientadora.   * + - * 1. ¿Cómo puede optimizar el algoritmo propuesto?         2. ¿En qué situación del diario vivir podría utilizar el algoritmo propuesto?   Instrucciones para participar:  Publicar un ejemplo de un algoritmo.  Explicar de manera clara cómo funciona el algoritmo y cuál es el resultado esperado.  Comenta de manera clara y respetuosa la participación de uno de sus compañeros.  Evaluación de la participación:   1. Claridad en la respuesta: 40% 2. Interacción con otros compañeros 30% 3. Respeto en el diálogo 20% 4. Puntualidad 10% | Reconoce las estructuras básicas a partir de ejemplos de algoritmos sencillos | 20% |
| AA1 | Taller Individual “**Identificación de estructuras lógicas en pseudocódigo**”.  **Objetivo:** el estudiante realizará la búsqueda en internet de 5 ejemplos de pseudocódigo con el fin de identificar las estructuras básicas: secuencia, condición y ciclo. Con este ejercicio se crea un acercamiento del estudiante con la temática del pseudocódigo.  **Instrucciones:**  Busque en internet 5 ejemplos de pseudocódigos en donde se empleen estructuras de secuencia, condición y ciclos.  Imprima/dibuje sobre cada uno la estructura.  Mejore el pseudocódigo incluyendo nuevos procesos, condiciones, ciclos.  En una sesión sincrónica responda: ¿Cuál es la estructura lógica que predomina en el ejemplo? ¿Cómo se puede mejorar la estructura del pseudocódigo?  **Criterios de evaluación:**  a. Entrega los 5 ejemplos con sus anotaciones personales sobre las estructuras: 50%  b. Entrega las mejoras en los pseudocódigos seleccionados: 30%  c. Argumenta con claridad cada estructura, su funcionalidad dentro del algoritmo y las acciones de mejora que se pueden proponer: 10%  d. organización y claridad en la presentación: 10% | Identifica correctamente las estructuras lógicas presentes en el ejemplo de análisis de pseudocódigo | 40% |
| AA2 | Taller Grupal “**Identificación de estructuras básicas para el diseño de un algoritmo**”.  **Objetivo:** los estudiantes conocerán los conceptos teóricos sobre secuencias, ciclos, condiciones y los aplicarán a través de ejercicios prácticos. Podrán desarrollar la habilidad de generar algoritmos sencillos que encuentren la solución a problemas específicos.  **Instrucciones:**  Conformar equipos de trabajo de 3 estudiantes para trabajar colaborativamente.  Cada grupo deberá generar un algoritmo que permita utilizar las secuencias, ciclos y condiciones según el problema propuesto.  **Actividades del taller:**   1. Secuencias: diseñar un algoritmo que resuelva una actividad cotidiana (cepillarse los dientes, preparar un café, abrir el refrigerador). 2. Condición: diseñar un algoritmo que permita tomar decisiones (al vestirse, al comprar un producto, al seleccionar un teléfono móvil). 3. Ciclo: diseñe un algoritmo que permita utilizar ciclos en su estructura (generar las tablas de multiplicar, obtener los números primos, calcular el factorial).   Se debe entregar el pseudocódigo y diagrama de flujo para cada ejercicio.  Tiempo estimado: 90 min.  Recursos necesarios: papel, lápiz, computadora.  En una sesión sincrónica cada grupo deberá compartir los diagramas y pseudocódigos elaborados comunicando a sus demás compañeros cómo identificaron las estructuras a utilizar y cómo se pueden aplicar en otros contextos.  **Criterios de evaluación:**  a. Identificación de las estructuras: 50 %  b. justificación y argumentación: 30 %  c. organización y claridad en la presentación: 20 % | Identifica las estructuras básicas para el diseño de un algoritmo | 40% |
| **% de Unidad de Aprendizaje** | | | 100 % |

* + - 1. **GLOSARIO DE LA UNIDAD**

| **Relacione las definiciones de los términos claves, requeridas para comprender adecuadamente los contenidos de esta unidad. Presentarlo en orden alfabético. Máximo 20 palabras.** | |
| --- | --- |
| **PALABRA, TÉRMINO O ABREVIATURA** | **SIGNIFICADO** |
| Algoritmo | Secuencia de pasos lógicos, finitos que resuelven un problema particular. |
| Ciclos | Conjunto de instrucciones de repetición que se llevan a cabo mientras se cumple una condición. |
| Condiciones | Etapa en la que el algoritmo debe tomar una decisión y según la respuesta obtenida continuará su proceso. |
| Constante | Es una variable almacenada en memoria y cuyo contenido no cambiará durante todo el proceso de ejecución del algoritmo. |
| Diagrama de flujo | Representación gráfica de un proceso, ilustran paso a paso la lógica que debe seguir el programa de manera sencilla. |
| Eficiencia | Seleccionar la manera óptima para resolver un problema. |
| Identificador | Corresponde al nombre asignado a una variable en la memoria |
| Memoria | Espacio reservado dentro del sistema que contiene un nombre y una variable |
| Proceso | Conjunto de instrucciones que “manipulan” los datos y entregan la solución al problema. |
| Programa | “Conjunto de instrucciones que sigue la computadora para alcanzar un resultado específico” (Von Neumann 1946) |
| Pseudocódigo | Permite escribir de manera ordenada la secuencia de pasos lógicos que el algoritmo debe ejecutar, es universal lo que implica su comprensión y lectura sin requerirse un lenguaje de programación formal para su escritura |
| Secuencia | Ejecución de instrucciones de manera ordenada y jerárquica, lineal, una tras otra y sin interrupción hasta su finalización |
| Sistema | Conjunto de elementos que tienen una relación directa entre sí y que interactúan de manera coordinada para obtener un objetivo específico |
| Variable | Diferente a la constante, la variable si cambia “varía” su valor según los procedimientos que se lleven a cabo dentro del algoritmo. |

1. **MATERIAL DE APOYO**

| **RECURSOS BIBLIOGRÁFICOS DE REVISIÓN BÁSICA**  Incluye libros impresos, digitales, electrónicos, ebook, artículos de revistas impresas, electrónicas, informes, seminarios, congresos, ponencias, presentaciones o diapositivas online, etc. Desde el 2019 en adelante excepto casos excepcionales. Bases de datos (licenciadas por la institución), bases de datos libres, videos bajo licenciamiento creative commons | | **BASES DE DATOS**  Si las referencias bibliográficas son de las bases de datos institucionales mencione únicamente el nombre de la base de datos donde se encuentra el recurso. |
| --- | --- | --- |
| 1 | Bisbal Riera, J. (2013). *Manual de algorítmica: recursividad, complejidad y diseño de algoritmos:*(ed.). Editorial UOC. https://elibro.net/es/ereader/tecnologicadeloriente/56561?page=50 | Elibro.net |
| 2 | Gaxiola Pacheco, C. G. & Flores Gutiérrez, D. L. (2008). *Metodología de la programación con pseudocódigo enfocado al lenguaje C:*(ed.). Plaza y Valdés (México). https://elibro.net/es/ereader/tecnologicadeloriente/75935?page=15 | Elibro.net |
| 3 | Joyanes Aguilar, L. (2005). *Programación en C: metodología, algoritmos y estructura de datos:*(2 ed.). McGraw-Hill España. https://elibro.net/es/ereader/tecnologicadeloriente/50302?page=87 | Elibro.net |
| 4 | Joyanes Aguilar, L. (2006). *Programación en C++: algoritmos, estructuras de datos y objetos:*(2 ed.). McGraw-Hill España. https://elibro.net/es/ereader/tecnologicadeloriente/50088?page=52 | Elibro.net |
| 5 | Mancilla Herrera, A. (2015). *Diseño y construcción de algoritmos:*(ed.). Universidad del Norte. https://elibro.net/es/ereader/tecnologicadeloriente/69931?page=35 | Elibro.net |
| 6 | Moreno, E. (2012). *Grafos: fundamentos y algoritmos:*(ed.). Editorial ebooks Patagonia - J.C. Sáez Editor. https://elibro.net/es/ereader/tecnologicadeloriente/68438?page=22 | Elibro.net |
| 7 | Oviedo Regino, E. M. (2015). *Lógica de programación orientada a objetos:*(ed.). Ecoe Ediciones. https://elibro.net/es/ereader/tecnologicadeloriente/70431?page=67 | Elibro.net |
| 8 | Pérez Aguila, R. (2012). *Una introducción a las matemáticas para el análisis y diseño de algoritmos:*(ed.). El Cid Editor. https://elibro.net/es/ereader/tecnologicadeloriente/35059?page=106 | Elibro.net |
|  |  |  |
| **RECURSOS AUDIOVISUALES**  Incluye videos, podcast, audiolibros, grabaciones sonoras o musicales, que se encuentran alojados en internet. | | |
| 1 | Magic Markers. (2021, 15 octubre). ¿Qué son los algoritmos? [Video]. YouTube. <https://youtu.be/U3CGMyjzlvM> | |
| 2 | BBC News Mundo. (2020, 2 octubre). Qué son los algoritmos y cómo aprenden de nosotros | BBC Mundo [Video]. YouTube. <https://youtu.be/RSJrBEhdZxw> | |
| 3 | Aprende y Verifica. (2022, 10 agosto). ¿Qué es un pseudocódigo? | Explicación rápida y sencilla [Video]. YouTube. https://youtu.be/Q1z34G3fTUM | |
| 4 | Aprende y Verifica. (2022, 2 noviembre). Ejemplo de un pseudocódigo y diagrama de flujo ⏐ Explicación RÁPIDA Y SENCILLA [Video]. YouTube. https://youtu.be/lByEQkmKRis | |
| **PÁGINAS WEB DE REVISIÓN BÁSICA**  Incluye blogs, hipertextos, animaciones, etc. | | |
|  | https://www.khanacademy.org/computing/computer-science/algorithms | |
|  | [edx.org/learn/algorithms/stanford-university-algorithms-design-and-analysis-part-1?index=product&queryID=6fea1e01232093af619ff7063b7346be&position=4&linked\_from=autocomplete&c=autocomplete](http://edx.org/learn/algorithms/stanford-university-algorithms-design-and-analysis-part-1?index=product&queryID=6fea1e01232093af619ff7063b7346be&position=4&linked_from=autocomplete&c=autocomplete) | |

| **LABORATORIOS, SIMULADORES, SOFTWARE, APP´S Y HERRAMIENTAS DE IA DE APOYO** | | **LINK DE ACCESO** |
| --- | --- | --- |
| 1 | Midjourney (accesible solo a través de Discord) | <https://www.midjourney.com> |
| 2 | GitHub Copilot (Microsoft y GitHub) | <https://github.com/features/copilot> |