



UTPL

La Universidad Católica de Loja

Vicerrectorado de Modalidad Abierta y a Distancia

Algoritmos y Resolución de Problemas

Guía didáctica





Facultad Ingenierías y Arquitectura

Algoritmos y Resolución de Problemas

Guía didáctica

| Carrera | PAO Nivel |
|-------------------------------|-----------|
| Tecnologías de la información | I |

Autora:

María del Carmen Cabrera Loayza



Universidad Técnica Particular de Loja

Algoritmos y Resolución de Problemas

Guía didáctica

María del Carmen Cabrera Loayza

Diagramación y diseño digital

Ediloja Cía. Ltda.

Marcelino Champagnat s/n y París

edilocialtda@ediloja.com.ec

www.ediloja.com.ec

ISBN digital -978-9942-25-711-6

Año de edición: abril, 2020

Edición: primera edición reestructurada en agosto 2025 (con un cambio del 50%)

Loja-Ecuador



Los contenidos de este trabajo están sujetos a una licencia internacional Creative Commons **Reconocimiento-NoComercial-CompartirIgual** 4.0 (CC BY-NC-SA 4.0). Usted es libre de **Compartir** — copiar y redistribuir el material en cualquier medio o formato. Adaptar — remezclar, transformar y construir a partir del material citando la fuente, bajo los siguientes términos: Reconocimiento- debe dar crédito de manera adecuada, brindar un enlace a la licencia, e indicar si se han realizado cambios. Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que usted o su uso tienen el apoyo de la licenciante. No Comercial-no puede hacer uso del material con propósitos comerciales. Compartir igual-Si remezcla, transforma o crea a partir del material, debe distribuir su contribución bajo la misma licencia del original. No puede aplicar términos legales ni medidas tecnológicas que restrinjan legalmente a otras a hacer cualquier uso permitido por la licencia. <https://creativecommons.org/licenses/by-nc-sa/4.0/>



Índice

| | |
|--|-----------|
| 1. Datos de información | 8 |
| 1.1 Presentación de la asignatura..... | 8 |
| 1.2 Competencias genéricas de la UTPL..... | 8 |
| 1.3 Competencias específicas de la carrera | 8 |
| 1.4 Problemática que aborda la asignatura en el marco del proyecto | 8 |
| 2. Metodología de aprendizaje | 10 |
| 3. Orientaciones didácticas por resultados de aprendizaje..... | 11 |
| Primer bimestre | 11 |
| Resultado de aprendizaje 1: | 11 |
| Contenidos, recursos y actividades de aprendizaje recomendadas..... | 11 |
| Semana 1 | 11 |
| Unidad 1. Fundamentos de algoritmos y su representación..... | 11 |
| 1.1. Concepto de lógica | 12 |
| 1.2. Algoritmo | 13 |
| 1.3. Sistema | 14 |
| 1.4. Concepto de algoritmo en el marco de la lógica | 14 |
| 1.5. Lógica de la programación..... | 15 |
| 1.6. Lenguajes de programación..... | 15 |
| Actividad de aprendizaje recomendada | 17 |
| Contenidos, recursos y actividades de aprendizaje recomendadas..... | 18 |
| Semana 2..... | 18 |
| Unidad 1. Fundamentos de algoritmos y su representación..... | 18 |
| 1.7. Diagramas de flujo y su representación | 18 |
| Actividades de aprendizaje recomendadas | 20 |
| Autoevaluación 1 | 22 |
| Resultado de aprendizaje 2: | 26 |
| Contenidos, recursos y actividades de aprendizaje recomendadas..... | 26 |
| Semana 3..... | 26 |



| | |
|--|-----------|
| Unidad 2. Datos e información | 26 |
| 2.1. Datos e información..... | 26 |
| 2.2. Bit y byte | 27 |
| 2.3. Múltiplos y submúltiplos..... | 27 |
| 2.4. Operadores | 28 |
| 2.5. Expresiones | 28 |
| Actividades de aprendizaje recomendadas | 31 |
| Contenidos, recursos y actividades de aprendizaje recomendadas..... | 32 |
| Semana 4 | 32 |
| Unidad 2. Datos e información | 32 |
| 2.6. Variables | 32 |
| Actividades de aprendizaje recomendadas | 34 |
| Autoevaluación 2..... | 35 |
| Resultados de aprendizaje 3 y 4: | 38 |
| Contenidos, recursos y actividades de aprendizaje recomendadas..... | 38 |
| Semana 5 | 38 |
| Unidad 3. Ciclo de desarrollo y primitivas algorítmicas | 38 |
| 3.1. Ciclo de desarrollo de un programa | 39 |
| Actividades de aprendizaje recomendadas | 41 |
| Contenidos, recursos y actividades de aprendizaje recomendadas..... | 42 |
| Semana 6 | 42 |
| Unidad 3. Ciclo de desarrollo y primitivas algorítmicas | 42 |
| 3.2. Primitivas algorítmicas..... | 42 |
| 3.3. Diseño de miniespecificaciones | 43 |
| 3.4. Pruebas de escritorio | 44 |
| Actividades de aprendizaje recomendadas | 46 |
| Autoevaluación 3..... | 47 |
| Resultados de aprendizaje 1 a 4:..... | 51 |
| Contenidos, recursos y actividades de aprendizaje recomendadas..... | 51 |



| | |
|--|-----------|
| Semana 7 | 51 |
| Actividades finales del bimestre | 51 |
| Contenidos, recursos y actividades de aprendizaje recomendadas..... | 51 |
| Semana 8..... | 51 |
| Actividades finales del bimestre | 51 |
| Segundo bimestre..... | 53 |
| Resultados de aprendizaje 3 y 4: | 53 |
| Contenidos, recursos y actividades de aprendizaje recomendadas..... | 53 |
| Semana 9 | 53 |
| Unidad 4. Estructuras lógicas condicionales | 53 |
| 4.1. Estructura condicional simple..... | 54 |
| 4.2. Estructura condicional compuesta | 55 |
| Actividades de aprendizaje recomendadas | 57 |
| Contenidos, recursos y actividades de aprendizaje recomendadas..... | 58 |
| Semana 10 | 58 |
| Unidad 4. Estructuras lógicas condicionales | 58 |
| 4.3. Estructura lógica dependiendo de | 58 |
| Actividades de aprendizaje recomendadas | 61 |
| Autoevaluación 4..... | 62 |
| Contenidos, recursos y actividades de aprendizaje recomendadas..... | 68 |
| Semana 11 | 68 |
| Unidad 5. Estructuras lógicas repetitivas | 68 |
| 5.1. Estructura lógica repetitiva: mientras-que – hacer | 69 |
| 5.2. Estructura lógica repetitiva: hacer–hasta..... | 70 |
| Actividades de aprendizaje recomendadas | 74 |
| Contenidos, recursos y actividades de aprendizaje recomendadas..... | 75 |
| Semana 12..... | 75 |
| Unidad 5. Estructuras lógicas repetitivas | 75 |
| 5.3. Estructura lógica repetitiva: para–hacer | 75 |



5.4. Estructuras lógicas repetitivas anidadas 76

Actividades de aprendizaje recomendadas 78

Autoevaluación 5..... 79

Contenidos, recursos y actividades de aprendizaje recomendadas..... 82

Semana 13..... 82

Unidad 6. Estructuras de datos 82

6.1. Arreglos unidimensionales..... 82

Actividades de aprendizaje recomendadas 86

Contenidos, recursos y actividades de aprendizaje recomendadas..... 87

Semana 14..... 87

Unidad 6. Estructuras de datos 87

6.2. Arreglos bidimensionales..... 87

Actividades de aprendizaje recomendadas 89

Autoevaluación 6..... 91

Contenidos, recursos y actividades de aprendizaje recomendadas..... 94

Semana 15..... 94

Actividades finales del bimestre 94

Contenidos, recursos y actividades de aprendizaje recomendadas..... 94

Semana 16..... 94

Actividades finales del bimestre 94

4. Solucionario 96

5. Referencias bibliográficas 120





1. Datos de información

1.1 Presentación de la asignatura



1.2 Competencias genéricas de la UTPL

Organización y planificación del tiempo.

1.3 Competencias específicas de la carrera

- Diseñar aplicaciones de *software* que permitan, mediante técnicas avanzadas de modelado, dar solución a los requerimientos del cliente, utilizando estándares de la industria.
- Implementar aplicaciones de baja, mediana y alta complejidad, integrando diferentes herramientas y plataformas, para dar solución a requerimientos de la organización.

1.4 Problemática que aborda la asignatura en el marco del proyecto

Dentro de la carrera Tecnologías de la Información, la asignatura Algoritmos y resolución de problemas, aporta al desarrollo de habilidades para resolver un problema computacional, mediante la adquisición de una forma de pensamiento que tiene la capacidad de abstracción, de encontrar patrones, de ordenar de manera operativa, de identificar los componentes de un problema y

representarlo en una notación formal, que permita establecer una solución. Todas las personas nos enfrentamos, diariamente, a un sinnúmero de problemas y somos capaces de resolverlos de diferentes maneras, pero existen muchas dificultades al momento de traducir este problema a un lenguaje algorítmico; ese es el objetivo principal de la presente asignatura.





2. Metodología de aprendizaje

Estimado estudiante, la metodología que se utiliza para el desarrollo de la asignatura se denomina Blended Learning. Con esta metodología, el proceso de enseñanza-aprendizaje se divide en trabajo autónomo y actividades síncronas, lo que le permitirá organizar su tiempo y actividades para cumplir con las tareas propuestas en la planificación de la asignatura. Asimismo, permite que el docente acompañe al estudiante por el *chat* de tutoría permanente, foro y video colaboración académicos, que le permiten al estudiante aclarar sus dudas y recibir un trato personalizado que le ayuda a avanzar en la adquisición de las competencias de esta asignatura.





3. Orientaciones didácticas por resultados de aprendizaje



Primer bimestre

Resultado de aprendizaje 1:

Discute la importancia de los algoritmos en el proceso de solución de problemas.

El resultado de aprendizaje está orientado a que los estudiantes comprendan los fundamentos de algoritmos y su representación, un conocimiento inicial necesario para abordar los temas primordiales en el desarrollo de algoritmos.

Contenidos, recursos y actividades de aprendizaje recomendadas

Recuerde revisar de manera paralela los contenidos con las actividades de aprendizaje recomendadas y actividades de aprendizaje evaluadas.



Semana 1

Unidad 1. Fundamentos de algoritmos y su representación

Iniciamos el estudio de la asignatura con los conceptos básicos de la lógica y algoritmos. Imagine que puede aprender a utilizar de forma más rápida y precisa sus conocimientos y que las ideas generadas siempre serán correctas y bien entendidas por los demás. Esto se conoce como lógica y vamos a utilizar algoritmos para plasmar esas ideas y usar la computadora como un medio para poder ejecutarlas.



Para iniciar con el desarrollo de algoritmos, es necesario conocer los fundamentos computacionales como sistema, lógica de la programación y lenguaje de la programación. Una vez conocidos estos conceptos, podemos entender la importancia del uso de algoritmos para la resolución de problemas.

La presente unidad se enfoca en los **Fundamentos de algoritmos y su representación**. A medida que vaya avanzando en su aprendizaje, irá alcanzando el siguiente resultado de aprendizaje: identifica los conceptos básicos de algoritmos y su relación en la resolución de problemas. Además, estará en la capacidad de contestar las siguientes interrogantes:

- ¿Qué son los algoritmos?
- ¿Cómo pueden ayudar los algoritmos en el desarrollo de habilidades de pensamiento computacional?

Para la presente semana se encuentra planificado el estudio de los fundamentos de algoritmos. Considere que, un algoritmo es un conjunto de pasos ordenados lógicamente que permiten realizar una tarea. El desarrollo de algoritmos ayuda a los estudiantes a la capacidad de identificar partes en las que podemos dividir un problema para trabajar sobre ellas en la búsqueda de la solución.



A continuación, resaltamos los aspectos más importantes de las temáticas planificadas para la presente semana. Sin embargo, para ampliar cada una de ellas, revise los apartados 1.1 al 1.6 de la unidad 1 de la [guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#).

1.1. Concepto de lógica

La lógica es una rama de la filosofía que estudia de manera formal las deducciones válidas que se derivan de un sistema de razonamiento, fundamentado en un conjunto de reglas. Si el sistema de razonamiento mencionado se expresa en un lenguaje matemático, recibe el nombre de



“lógica matemática”; en el caso de que el sistema de razonamiento utilice un lenguaje simbólico y un conjunto de reglas de inferencia, recibe el nombre de “lógica simbólica”.

Es, precisamente, la lógica simbólica que, mediante los algoritmos conformados por estructuras lógicas, ha permitido el desarrollo de la informática. Revise la sección 1.1 de la unidad 1 de la [guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#), para reforzar el concepto de lógica.

1.2. Algoritmo

Un algoritmo es la secuencia de pasos necesarios para resolver cualquier problema.

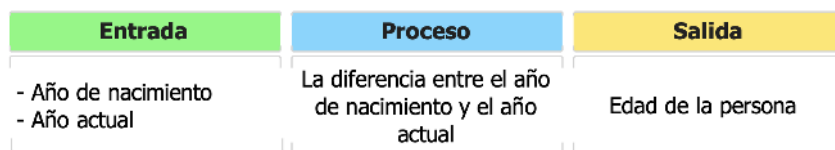
Un algoritmo se caracteriza fundamentalmente por ser:

- **Preciso:** indicar el orden de realización de cada paso dentro del algoritmo.
- **Definido:** obtiene el mismo resultado cada vez que se ejecute el algoritmo.
- **Finito:** debe tener un número finito de pasos.

El desarrollo de algoritmos se enmarca en tres partes: entrada, proceso y salida. Por ejemplo, en la figura 1, se presenta la secuencia de un algoritmo para calcular la edad de una persona, conociendo su año de nacimiento.

Figura 1

Pasos para calcular la edad de una persona



Nota. Cabrera, M., 2025.

Recuerde que, para la elaboración de un algoritmo, se emplea un lenguaje natural para la descripción de cada paso. En la figura 2 se realiza el algoritmo para calcular la edad de una persona:

Figura 2

Algoritmo para calcular la edad de una persona

Algoritmo

1. **Inicio**
 2. Lea el año de nacimiento
 3. Lea el año actual
 4. Reste el año actual y el año de nacimiento
 5. Escriba la edad de la persona
 6. **Fin**
- Entrada
- Proceso
- Salida

Nota. Cabrera, M., 2025.

Para reforzar los contenidos de la presente temática, vaya a la sección 1.2 de la unidad 1 de la [guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#).

1.3. Sistema

Un sistema es un conjunto de entidades que interactúan entre sí de una forma organizada, con el objetivo de desempeñar una función y generar información útil al usuario, dentro de los límites de operación y funcionalidad. Para ampliar su aprendizaje, revise la sección 1.3 de la unidad 1 de la [guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#). Al finalizar el estudio de esta sección, usted podrá responder: ¿qué es un sistema en el marco de la informática?

1.4. Concepto de algoritmo en el marco de la lógica

El análisis del concepto de algoritmo en su relación con la lógica, lleva a identificar que la resolución de problemas factibles al ser tratados por máquinas computacionales inicia en el sistema de razonamiento humano. En este contexto, la computadora se convierte en primera instancia en una herramienta de cálculo de los algoritmos, pero la naturaleza de la resolución de problemas es en sí la lógica humana. En la sección 1.4 de la unidad 1 de la [guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#), puede reforzar esta temática.



1.5. Lógica de la programación

La lógica de la programación es la capacidad de resolver un problema mediante la definición de un conjunto de pasos coherentes con el uso de estructuras, fundamentos y expresiones del conocimiento humano. Para adquirir una lógica de la programación no es necesario tener un conocimiento previo de computadora ni de tecnología en general, tampoco exige la presencia de algún lenguaje de programación específico. Para reforzar el tema, realizar una lectura comprensiva de la sección 1.5 de la unidad 1 de la [guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#).

1.6. Lenguajes de programación

Los lenguajes de programación sirven para escribir programas que permitan transformar los algoritmos a un lenguaje entendible por la máquina. Los denominados traductores (compiladores o intérpretes) convierten las instrucciones escritas en lenguajes de programación en instrucciones escritas en lenguaje máquina (0 y 1, *bits*) que esta pueda entender. Los pasos generales para convertir un algoritmo en programa son:

1. Transcribir el algoritmo con un lenguaje de programación mediante un editor.
2. Introducir el programa fuente en memoria.
3. Compilar el programa.
4. Verificar y corregir los errores de compilación.
5. Se ejecuta el programa; si no presenta errores, se obtendrá la salida del programa que es el resultado.

Una vez culminadas las temáticas de la presente semana, podemos dar respuesta a la siguiente pregunta: ¿cuáles son los conceptos fundamentales de los algoritmos en la resolución de problemas? Si está en la capacidad de responder esta pregunta, avance con el estudio de la siguiente sección; caso contrario, es necesario que realice nuevamente las lecturas recomendadas.



Para continuar con el estudio de los contenidos de la presente semana, es importante apoyarse en los materiales principales de la asignatura. Por este motivo, le invito a realizar una lectura comprensiva de los siguientes documentos complementarios:

- Cabrera, M. y Tenesaca, G. (2018). [Guía didáctica de algoritmos y resolución de problemas](#). Loja, Ecuador: Editorial Universidad Técnica Particular de Loja. Leer la unidad 1. Fundamentos de algoritmos y su representación, sección 1.1–1.6.
- Trejos Buriticá, O. (2017). Lógica de programación: (ed.). Ediciones de la U. Leer: [capítulo 1. La lógica](#).

Además, para reforzar los contenidos, le invito a visualizar los siguientes videos:



- [¿Qué es un algoritmo?](#)

En este video encontrará la conceptualización de un algoritmo, describiendo cómo determinar las entradas, procesos y salidas para resolver un problema utilizando algoritmos.

- [¿Qué es un algoritmo y por qué debería importarte?](#)

En este segundo video se presenta una descripción con ejemplos sobre la importancia de los algoritmos y cómo se utilizan en la resolución de problemas.

¿Le pareció interesante los videos? Ahora ya podemos responder con más seguridad la pregunta: ¿qué es un algoritmo?

En los videos de apoyo se define a un algoritmo como una serie de pasos o instrucciones finitas y ordenadas que sirven para dar solución a un problema. Un algoritmo debe ser preciso y sin ambigüedades. Un algoritmo debe tener siempre inicio y fin, entre el inicio y el fin deben estar las instrucciones o pasos.





Actividad de aprendizaje recomendada

Una vez revisados los materiales complementarios, es importante medir los conocimientos adquiridos; por esta razón, se ha propuesto la siguiente actividad de aprendizaje. Recuerde que estas actividades no son calificadas.

Realice un estudio de los materiales complementarios, siguiendo el siguiente procedimiento:

1. Realice una lectura comprensiva de la [unidad 1 – sección 1.1 – 1.6 de la guía didáctica](#) de Cabrera, M. y Tenesaca, G. (2018), así como del [capítulo 1 de Trejos, O. \(2017\)](#).
2. Revise los videos presentados previamente. Tome nota de lo más importante y las dudas que se le presenten, para resolverlas con su tutor.

El desarrollo de las actividades recomendadas le ayudará a comprender de mejor manera el siguiente tema de estudio: *Diagramas de flujo y su representación*. Para ello, le invito a revisar la guía didáctica y los materiales complementarios.

Nota. Por favor, conteste la actividad en su cuaderno de apuntes o en un documento Word.





Semana 2

Unidad 1. Fundamentos de algoritmos y su representación

1.7. Diagramas de flujo y su representación

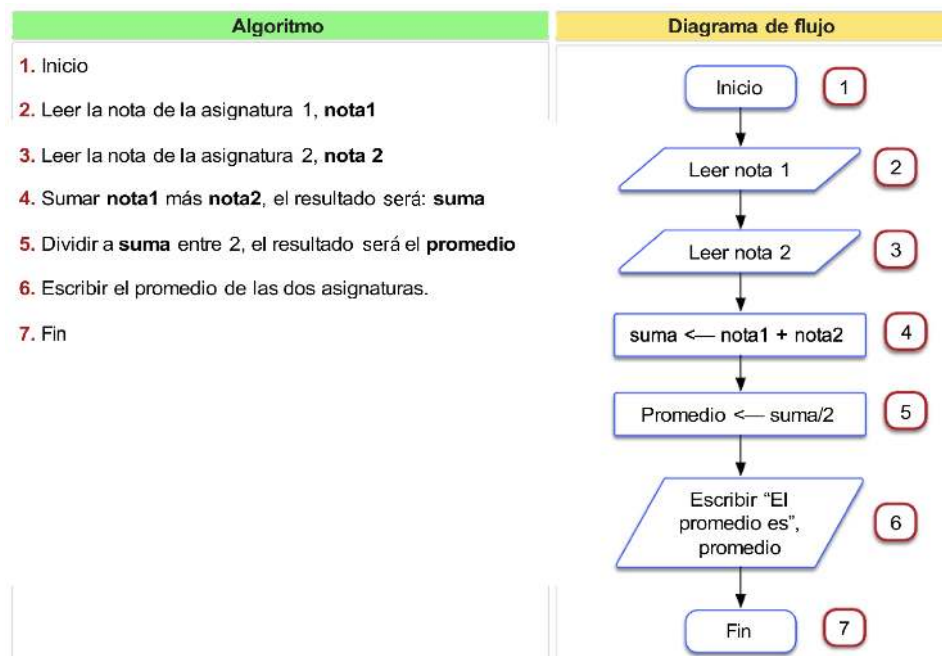
Los diagramas de flujo son una técnica de representación de algoritmos mediante símbolos unidos por flechas, denominadas líneas de flujo, que indican la secuencia de los pasos dentro de un algoritmo. Para conocer la simbología sobre diagramas de flujo, revise la sección 1.7 de la unidad 1 de la [guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#).

A continuación, para la comprensión de la forma de representar un algoritmo mediante un diagrama de flujo, se realiza el siguiente ejercicio: en la figura 3 se presenta un algoritmo y su diagrama de flujo que permite calcular el promedio de dos asignaturas. Observe que la numeración relaciona cada uno de los pasos del algoritmo con su representación en el diagrama de flujo.



Figura 3

Representación de un algoritmo en diagrama de flujo



Nota. Cabrera, M., 2025.

¿Qué le pareció la temática?, ¿aún tiene dudas? A continuación, revise los siguientes materiales de apoyo que le ayudarán a solventar sus dudas.

Para la presente semana se propone el estudio del contenido de los siguientes documentos complementarios:

- Cabrera, M. y Tenesaca, G. (2018). [Guía didáctica de Algoritmos y resolución de problemas](#). Loja, Ecuador: Editorial Universidad Técnica Particular de Loja. Realice la lectura de la Unidad 1. Fundamentos de algoritmos y su representación, sección 1.7.
- Trejos Buriticá, O. (2017). Lógica de programación: (ed.). Ediciones de la U. Lectura: [Capítulo 4. Estructuras básicas y técnicas para representar algoritmos](#), sección 4.4.1.
- Lectura: Capítulo 1 – Marco Conceptual de la lógica de la programación.

Además, para reforzar la temática de diagramas de flujo, le invito a visualizar el siguiente video: [LP # 11| Diagramas de flujo](#).

En el video recomendado encontrará una descripción a detalle sobre los diagramas de flujo y cómo utilizar cada uno de sus símbolos. Además, se presentan algunos ejemplos que le serán de mucha utilidad.

¿Le parecieron interesantes los materiales complementarios? Ahora bien, ya podemos dar respuesta a las siguientes interrogantes: ¿qué es un diagrama de flujo? y ¿para qué sirven los diagramas de flujo?



Un diagrama de flujo es una herramienta que ayuda a representar los algoritmos de forma gráfica con una simbología específica. Los diagramas de flujo son considerados como un entregable en la fase de diseño, y ayudan a esquematizar de mejor manera un algoritmo. Son de mucha ayuda para que los programadores puedan entender el problema a codificar.

Estimado estudiante, esperamos haya quedado claro los contenidos de la semana. Sin embargo, si aún tiene inquietudes, puede consultar a su docente mediante el *chat* de tutoría permanente. ¡Continuamos con la asignatura!



Actividades de aprendizaje recomendadas

A continuación, es importante medir los conocimientos adquiridos. Le recomendamos realizar las siguientes actividades de aprendizaje. Recuerde que estas actividades no son calificadas.

1. Realice una revisión de materiales complementarios, tomando en cuenta el siguiente procedimiento:
 - a. Realice una lectura comprensiva de la [unidad 1 – sección 1.7 de la guía didáctica](#) de Cabrera, M. y Tenesaca, G. (2018), así como de la sección 4.1.1, del [capítulo 4](#) de Trejos, O. (2017).



b. Revise el video presentado previamente. Tome nota de lo más importante y las dudas que se le presenten, para resolverlas con su tutor.

2. Elabore un diagrama de flujo, siguiendo el siguiente procedimiento:

a. Elabore un diagrama de flujo con base en las indicaciones dadas en la guía didáctica del siguiente problema:

- **Tarea:** representación gráfica de un problema mediante un diagrama de flujo.
- **Problema:** se requiere elaborar un diagrama de flujo que permita ingresar la edad de una persona e identificar si es un niño, joven, adulto o de tercera edad.
- **Datos adicionales:** considere los siguientes rangos de edad:
 - Edad $> 1 \leq 12$ --> Niño.
 - Edad $> 12 \leq 18$ --> Joven.
 - Edad $> 18 \leq 65$ --> Adulto.
 - Edad > 65 --> Tercera e

b. Puede compartir sus respuestas en el EVA para una retroalimentación general con todos sus compañeros.

Nota. Por favor, conteste las actividades en su cuaderno de apuntes o en un documento Word.

3. Desarrolle la autoevaluación 1, que se encuentra al final de la unidad

1. La misma consiste en un cuestionario de preguntas objetivas relacionadas con las temáticas de la primera y segunda





Autoevaluación 1

Seleccione la respuesta según corresponda:

1. El desarrollo de la informática como ciencia, estudia el tratamiento de:
 - a. Ficheros.
 - b. Información.
 - c. Programas.
2. Traslada el control del programa a otra parte dentro del mismo diagrama:
 - a. Símbolo conector.
 - b. Símbolo de línea de flujo.
 - c. Proceso.
3. Realiza un conjunto de pasos cuya ejecución para dar la solución del problema puede ser ejecutada manualmente, mecánicamente o utiliza una máquina de procesamiento electrónico de datos:
 - a. Sistema.
 - b. Lenguaje de programación.
 - c. Algoritmo.
4. Es una ventaja ser generalmente conocidos e interpretados:
 - a. Diagramas de flujo.
 - b. Mini especificación.
 - c. Sistema.
5. Es utilizado para procesar información y obtener resultados, capaz de ejecutar cálculos y tomar decisiones a velocidades millones o cientos



de millones más rápidas de lo que puedan hacerlo los seres humanos:

- a. Sistema computacional.
- b. Computadora.
- c. Lenguaje de programación.

6. Son los caracteres (generalmente no más de dos), que provocan un comportamiento predecible dentro de un programa Ítem:

- a. Variables.
- b. Palabras reservadas.
- c. Símbolos especiales.

7. Generalmente, son elaborados en las fases iniciales del ciclo de vida de desarrollo (análisis y diseño):






- a. Diagramas de flujo.
- b. Lenguajes de programación.
- c. Símbolo de datos.

8. En un diagrama de flujo, el paralelogramo es el símbolo que representa:

- a. Petición de datos.
- b. Proceso definido.
- c. Operación de decisión.

9. Relacione el término con la forma apropiada (ver figura):



| | | |
|--------------------|---|---|
| 1. Conector | a |  |
| 2. Límite de bucle | b |  |
| 3. Decisión | c |  |
| 4. Terminador | d |  |
| 5. Proceso | e |  |

a. 1a, 2b, 3d, 4e, 5c.

b. 1c, 2e, 3b, 4d, 5a.

c. 1e, 2c, 3a, 4d, 5b.

10. Complete el siguiente diagrama de flujo, para que permita pedir un mensaje al usuario y luego presente el mismo en pantalla.



Diagrama de flujo a



Diagrama de flujo b

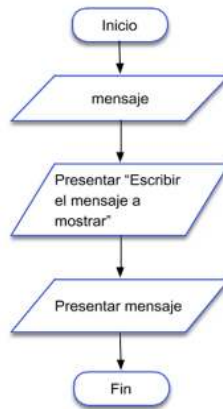


Diagrama de flujo c



- a. Diagrama de flujo a.
- b. Diagrama de flujo b.
- c. Diagrama de flujo c.

[Ir al solucionario](#)



Resultado de aprendizaje 2:

Identifica las propiedades necesarias para un buen algoritmo.

El resultado de aprendizaje está orientado a identificar las particularidades de los datos, así como sus elementos procesamiento y almacenamiento. Además, aprenderá a representar los datos y sus dominios según el contexto del problema.

Contenidos, recursos y actividades de aprendizaje recomendadas

Recuerde revisar de manera paralela los contenidos con las actividades de aprendizaje recomendadas y actividades de aprendizaje evaluadas.



Semana 3

Unidad 2. Datos e información

¿Cuál es la naturaleza de los datos para la computadora?, ¿qué son las expresiones y variables?

Para el desarrollo de algoritmos, es necesario conocer la naturaleza de los datos a representar: tipos de datos y sus dominios. De esta forma, se puede representar correctamente dentro de un algoritmo, el dominio de valores a utilizar en el desarrollo de la solución. Así también, se debe tener claro, qué expresiones y operaciones podemos utilizar en el contexto de lógica de la programación, su representación y uso. Finalmente, aquí aprendemos el concepto de variable, como una unidad de almacenamiento de una expresión o el resultado de una operación.

2.1. Datos e información

La información es la asociación coherente y con significado dentro de un contexto definido de datos individuales o de un conjunto de datos. Por lo tanto, la información siempre tendrá significado para quien la recibe. Para una



empresa, la información representa el “activo más valioso” que posee y constituye la materia prima para los procesos en la toma de decisiones. Cuando ingresan datos a una computadora, un intérprete se encarga de transformarlos a un lenguaje codificado como el sistema numérico binario (0 y 1).

2.2. Bit y byte

Un *bit* (Dígito binario – *Binary digit*), es considerado la unidad más pequeña de información; es utilizado fundamentalmente en operaciones booleanas conocidas como: AND, OR, NOT y XOR. Sin embargo, se necesita más de un *bit* para la representación de datos (números, letras, símbolos). Revisemos el siguiente concepto.

Se denomina byte u octeto a un conjunto de ocho *bits*, en los que se almacenan números binarios de ocho dígitos. El *byte* es la unidad básica de medida de la capacidad de la memoria o cualquier dispositivo de almacenamiento de una computadora.



¿Qué le pareció la temática? Ahora bien, para reforzar su aprendizaje sobre datos e información, le invito a revisar la sección 2.2 de la unidad 2 de la [guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#). Esto le ayudará a comprender de mejor manera el siguiente tema de estudio: Múltiplos y submúltiplos del *byte*. ¡Avancemos!

2.3. Múltiplos y submúltiplos

Los múltiplos del *byte* son *Word* (Palabra), *Half Word* (Media Palabra) y *Double Word* (Doble Palabra). Un *Word* es el número máximo de *bits* con los que trabaja un procesador de manera simultánea.

Los únicos submúltiplos del *byte* son el *nibble* y el *bit*; un *nibble* es un conjunto de cuatro *bits*, conocido también como “cuarteto o semiocteto”, que permite representar un número binario de cuatro dígitos.



Para revisar la tabla de conversión de múltiplos y submúltiplos, vaya a la sección 2.3 de la unidad 2 de la [guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#).

2.4. Operadores

Un operador es un símbolo que tiene una función predefinida (suma, resta, multiplicación, mayor que, etc.), que nos permite construir expresiones compuestas. Los operadores se clasifican en aritméticos, relacionales y lógicos. Para ver el detalle y ejemplos, enfocados a cada uno de los tipos de operadores, revise la sección 2.4 de la unidad 2 de la [guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#).

Ahora, vamos a continuar con el estudio de las expresiones. ¡¡Adelante, vamos por más conocimientos!!

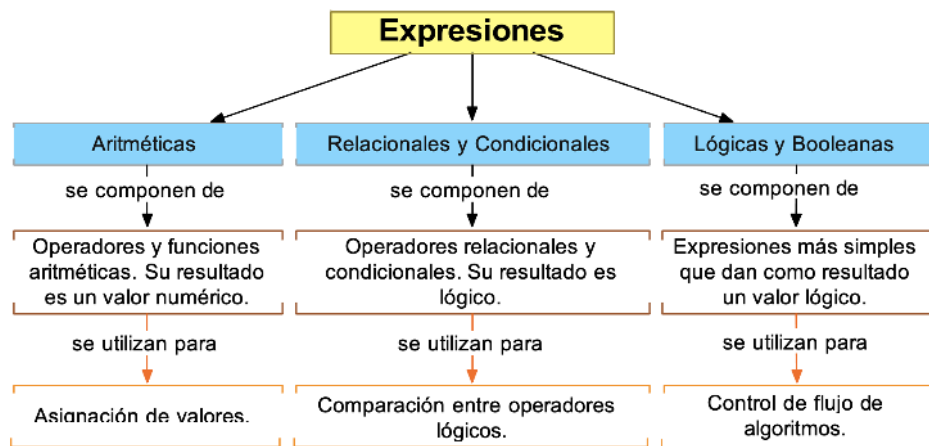
2.5. Expresiones

Una expresión es una relación entre variables y operadores relacionales, aritméticos y/o lógicos. Así también, las expresiones están clasificadas en aritméticas, relacionales y lógicas, como se detalla en la figura 4. Para profundizar sobre la temática, revise la sección 2.5 de la unidad 2 de la [guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#).



Figura 4

Tipos de expresiones



Nota. Cabrera, M., 2025.

¿Cómo le fue con las temáticas? ¿aún hay dudas? Si es así, revise los siguientes materiales de aprendizaje que le ayudaran a responder sus preguntas y a reforzar lo aprendido.

Para iniciar el estudio de los contenidos de la presente semana es importante apoyarse en los materiales principales de la asignatura. Por este motivo, le invito a realizar una lectura comprensiva de los siguientes documentos complementarios:

- Cabrera, M. y Tenesaca, G. (2018). [Guía didáctica de Algoritmos y resolución de problemas](#). Loja, Ecuador: Editorial Universidad Técnica Particular de Loja. Realice la lectura de la Unidad 2. Datos e información.
- Ramírez Marín, J. H. (2019). Fundamentos iniciales de lógica de programación I. Algoritmos en PSeInt y Python: (1 ed.). D - Institución Universitaria de Envigado.

Lectura: [Capítulo 1: Conceptos básicos – Sección 1.1 a 1.4.](#)

Además, para reforzar los contenidos, revise los siguientes videos, donde aprenderá cómo escribir expresiones matemáticas en un programa y conocerá las cuatro reglas fundamentales, así como la definición y el uso de los operadores aritméticos, lógicos y relacionales:

- [LP #7| Jerarquía de operaciones.](#)
- [LP #10| Operadores aritméticos, lógicos y relacionales.](#)

Luego de revisar los materiales de apoyo, hemos aclarado las temáticas de operadores y reglas de precedencia. Con el primer video nos queda más claro que existen 3 tipos de operadores: aritméticos, relacionales y lógicos son muy importantes para la programación.

El segundo video refuerza la importancia de la relación que existen entre la programación basada en gran parte en las matemáticas puras. Es primordial conocer que existen 4 reglas al programar expresiones matemáticas que son enlistados:

- i. Una expresión va en una sola línea,
- ii. La computadora resuelve las expresiones en orden jerárquico.

¡Recuerde!, que primero se resuelve la potencia, luego multiplicación y división, tercero suma y resta.
- iii. En caso de establecer otro orden se utiliza paréntesis.
- iv. Sí hay varios paréntesis se resuelve primero lo más interno. En los materiales de apoyo se plantean algunos ejercicios, si desea reforzar lo aprendido puede resolverlos.





Actividades de aprendizaje recomendadas

Ahora, mediremos los conocimientos adquiridos. Para ello, realice las siguientes actividades de aprendizaje. Recuerde que estas actividades no son calificadas.

1. Realice una revisión de los materiales complementarios, siguiendo el siguiente procedimiento:

- a. Realice una lectura comprensiva de la unidad 2 – sección 2.1 – 2.5 de la [guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#), así como del [capítulo 1: Conceptos básicos – sección 1.1 a 1.4 de Ramírez, J. \(2019\)](#).
- b. Revise los videos presentados previamente. Tome nota de lo más importante y las dudas que se le presenten, para resolverlas con su tutor.

2. Desarrolle los siguientes ejercicios para reforzar lo aprendido:

- Realice las conversiones necesarias aplicando las equivalencias de múltiplos y submúltiplos del *byte*, según se indique:
 - a. ¿Cuántos *bytes* hay en 2378 *gigabytes*?
 - b. ¿Cuántas palabras dobles hay en 48 *terabytes*?
- Evalúe paso a paso la siguiente expresión considerando los siguientes valores: $x \leftarrow 2$; $y \leftarrow$ Tenga en cuenta la precedencia de los operadores y establezca diferencias donde se indique:
 $sw \leftarrow ((x \neq y) \wedge (x \leq y))$.
- Puede compartir sus respuestas en el EVA para una retroalimentación colaborativa con todos sus compañeros.

Nota. Por favor, conteste las actividades en su cuaderno de apuntes o en un documento Word.





Unidad 2. Datos e información

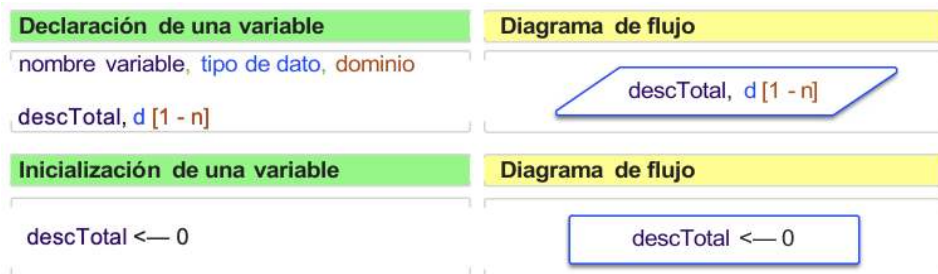
2.6. Variables

Una variable es un espacio en memoria que almacena un valor que puede cambiar durante la ejecución de un algoritmo.

Al momento de declarar una variable, se debe proporcionar un nombre que será su identificador, recuerde que el nombre siempre tiene que ser representativo del valor que se va a almacenar. Además, debe tener un tipo de dato y un dominio. Por ejemplo, en la figura 5, se observa la declaración de la variable, que representa el descuento total en una compra, y su declaración inicial.

Figura 5

Declaración e inicialización de una variable



Nota. Cabrera, M., 2025.

Como puede observar en la figura 5, el nombre seleccionado para identificar el descuento total de una compra es descTotal. El tipo de dato es decimal (d), debido a que el valor puede contener valores decimales. Y finalmente, el dominio ([1 - n]) que son todos los números positivos, ya que un descuento no puede ser negativo.



La inicialización de las variables se realiza al inicio del algoritmo, por lo general toma el valor inicial de 1, 0 o depende del problema a resolver. Sin embargo, en muchos casos se requiere un valor constante de inicialización. Por ejemplo, para realizar conversiones (transformar de kilos a libras), hay valores fijos que no cambiarán en la ejecución del algoritmo y a este tipo de variables se les denomina constantes.

Para ampliar las temáticas sobre variables, tipos de datos y dominio, revise la sección 2.6 de la unidad 2 de la [guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#). Esta sección le proporcionará las pautas necesarias para declarar una variable correctamente, así como la selección del tipo de dato y el rango de dominio que requiere para almacenar el valor que necesitamos.

Para iniciar el estudio de los contenidos de la presente semana es importante apoyarse en los materiales principales de la asignatura. Por este motivo, le invito a realizar una lectura comprensiva de los siguientes documentos complementarios:

- Cabrera, M. y Tenesaca, G. (2018). [Guía didáctica de Algoritmos y resolución de problemas](#). Loja, Ecuador: Editorial Universidad Técnica Particular de Loja. Realice la lectura de la Unidad 2. Datos e información, sección 2.6.
- Trejos Buriticá, O. (2017). Lógica de programación: (ed.). Ediciones de la U.

Lectura: [Capítulo 3. Variables, constantes y operadores](#).



Además, para reforzar los contenidos, se recomienda realizar la visualización del siguiente video: [LP #5| ¿Variables y constantes?](#)

En este video recomendado, reforzará el concepto de variable, constante, su declaración y cómo utilizarlas en la resolución de problemas, mediante algoritmos y programación.



Una vez revisado los materiales de apoyo, hemos aclarado el concepto de variable y podemos dar respuesta al siguiente interrogante: ¿Para qué me sirve una variable? recuerde que las variables sirven para identificar un dato específico que ha sido o será almacenado en la memoria del computador. De aquí en adelante el concepto de variables será utilizado para el desarrollo de algoritmos y programas.



Actividades de aprendizaje recomendadas

Ahora, es importante medir los conocimientos adquiridos. Para ello, le recomendamos realizar las siguientes actividades. Recuerde que estas actividades de aprendizaje no son calificadas.

1. Revise los siguientes materiales complementarios:

- a. Realice una lectura comprensiva de la [unidad 2 – sección 2.6](#) de la guía didáctica de Cabrera, M. y Tenesaca, G. (2018), así como también del [capítulo 3. Variables, constantes y operadores de Trejos, O. \(2017\)](#).
- b. Revise el video presentado previamente. Tome nota de lo más importante y de las dudas que se le presenten, para resolverlas con su tutor.

2. Desarrolle el siguiente ejercicio para reforzar lo aprendido:

Si costoProd y precioProd son variables numéricas y nombreProd es una variable de cadena, ¿cuáles de las siguientes declaraciones son asignaciones válidas? Si una declaración no lo es, explique por qué no.

- precioProd ← "35.76"
- costoProd ← costoProd + 20
- precioProd ← costoProd + nombreProd
- nombreProd ← "televisor"



Puede compartir sus respuestas en el EVA para una retroalimentación colaborativa con todos sus compañeros.

Nota. Por favor, conteste las actividades en su cuaderno de apuntes o en un documento Word.

3. Desarrolle la autoevaluación 2, la misma consiste en un cuestionario de preguntas objetivas relacionadas con las temáticas de la tercera y cuarta semana.



Autoevaluación 2

Responda las siguientes preguntas de opción múltiple y seleccione la respuesta correcta para cada una de ellas:

1. Corresponden a un carácter del sistema hexadecimal, van desde (0_{16}) al (F_{16}) :
 - a. *Bit*.
 - b. *Double Word*.
 - c. *Nibble*.
2. Si desea almacenar el costo de un producto, ¿qué tipo de dato se debe usar, teniendo en cuenta el contexto que se presenta?
 - a. *Byte*.
 - b. *Decimal*.
 - c. *Entero*.
3. ¿Cuál de las siguientes expresiones retorna un valor igual a True?
 - a. $"C" = "c"$.
 - b. $7 > = 7$.
 - c. $4 + 5 = 9 - 1$.



4. ¿Qué dominio es el correcto para almacenar los datos referentes a la edad de una persona?

- a. $b [0-110]$.
- b. $b [0-10]$.
- c. $b [18-100]$.

5. La precedencia que se provoca mediante el uso de paréntesis se conoce como:

- a. Posicional.
- b. Implícita.
- c. Explícita.

6. Tenemos la siguiente expresión: $(a - b < 3 - c)$ and $(c * 1 == a - b)$.
Donde $a = 2$, $b = 4$, $c = 6$. Determine el valor resultante:

- a. Verdadero.
- b. Falso.
- c. Nulo.

7. Los nombres de variables deben cumplir algunas reglas; del siguiente listado, indique cuáles pertenecen a estas reglas:

- 1. Siempre iniciar con letra.
- 2. Puede contener caracteres especiales.
- 3. Contener espacio y caracteres especiales opcionalmente.
- 4. Puede contener vocales tildadas.
- 5. Máximo 32 caracteres de longitud.
- 6. Representativo al valor que guarda.

- a. 1,5,6.
- b. 1,2,5,6.
- c. 1,4,5.





8. Seleccione la representación abstracta de dominio para una clave formada por una letra "Z" y 5 números; ningún número puede ser 0:
- a. $X(6) [1 \{Z\}, 5 \{0-9\}]$.
 - b. $X(6) [1 \{A-Z\}, 5 \{1-9\}]$.
 - c. $X(6) [1 \{Z\}, 5 \{1-9\}]$.
9. ¿Cuál de las siguientes opciones no describe el tipo de dato de una variable?
- a. Los valores que puede contener esta.
 - b. Qué operaciones pueden realizarse con ella.
 - c. El ámbito de la misma.
10. Si costoProducto y precioProducto son variables numéricas y nombreProducto, es una variable cadena, ¿cuál de las siguientes declaraciones no es válida?
- a. `costoProducto ← 100`.
 - b. `precioProducto ← "24.95"`.
 - c. `costoProducto ← precioProducto - 10`.

[Ir al solucionario](#)



Hemos concluido la unidad 2; avancemos en nuestros conocimientos con la unidad 3, Ciclo de desarrollo y primitivas algorítmicas.

Resultados de aprendizaje 3 y 4:

- Crea algoritmos para solucionar problemas simples.
- Aplica estrategias efectivas de depuración.

Por medio de los resultados de aprendizaje se analiza, diseña, crea y evalúa soluciones algorítmicas para dar solución a problemas planteados. Identificando claramente las variables, estructuras lógicas y los pasos a seguir plasmados en primitivas algorítmicas y evaluados mediante pruebas de escritorio.

Contenidos, recursos y actividades de aprendizaje recomendadas

Recuerde revisar de manera paralela los contenidos con las actividades de aprendizaje recomendadas y actividades de aprendizaje evaluadas.



Semana 5

Unidad 3. Ciclo de desarrollo y primitivas algorítmicas

¿Cuál es el procedimiento por seguir para el desarrollo de algoritmos?, ¿cuáles son las primitivas algorítmicas para la creación de algoritmos?, ¿en qué nos contribuye el uso de miniespecificaciones y pruebas de escritorio?

Con los contenidos de la presente unidad, el estudiante será capaz de analizar los casos del mundo real para dar solución a un problema.

En la semana 5 se analizará el Ciclo de vida de un programa y en la semana 6 se abordará el Uso de primitivas algorítmicas y el desarrollo de miniespecificaciones y pruebas de escritorio.



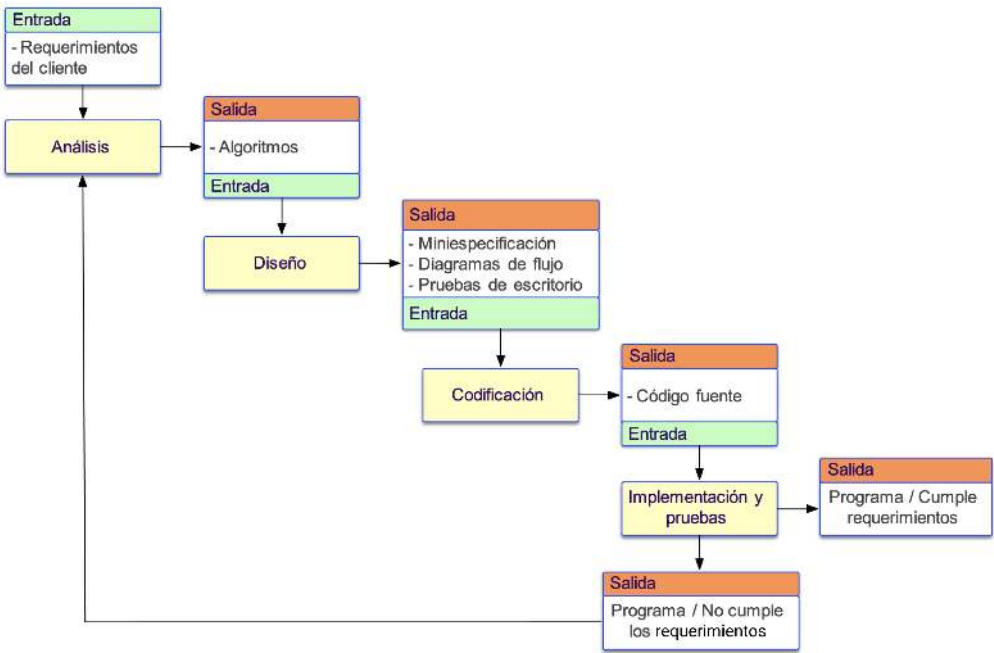
3.1. Ciclo de desarrollo de un programa

El ciclo de desarrollo de un programa permite plantear la solución a un problema dado, desde su análisis hasta la programación y validación. Esto, permite llevar de forma organizada cada una de las fases del ciclo, considerando que cada una de ellas debe dar respuesta a una pregunta objetiva y proporcionar un entregable resultado de la fase.

Dentro del ciclo de desarrollo de programas, se involucran cuatro fases: análisis, diseño, codificación y pruebas de implementación. El estudio de la asignatura abarca las dos primeras fases con el Desarrollo de algoritmos, Diagramas de flujo, Miniespecificaciones y pruebas de escritorio.

En la figura 6 se puede observar cada una de las fases del ciclo de desarrollo de un programa con la entrada y salida correspondientes.

Figura 6
Ciclo de vida del desarrollo de un programa



Nota. Cabrera, M., 2025.

Como se puede observar en la figura 6, la salida de cada una de las fases es la entrada de la siguiente, hasta culminar con el ciclo, el mismo que puede llevar nuevamente a la fase inicial si no cumple el programa los requerimientos del cliente.

En la sección 3.1 de la Unidad 3 de la guía didáctica de Cabrera, M. y Tenesaca, G. (2018), usted puede ampliar los contenidos de cada una de las fases del ciclo de desarrollo de un programa.

Para la presente semana se debe realizar una lectura comprensiva del siguiente documento complementario de la asignatura:

- Cabrera, M. y Tenesaca, G. (2018). Guía didáctica de Algoritmos y resolución de problemas. Loja, Ecuador: Editorial Universidad Técnica Particular de Loja.

Lectura: [Unidad 3. Ciclo de desarrollo y primitivas algorítmicas, sección 3.1.](#)

Para comprender mejor el tema, le invito a visualizar los siguientes videos, en el que aprenderá cómo analizar y resolver un problema antes de comenzar a escribir el código fuente, mediante un ejemplo práctico. Además, descubrirá qué es un algoritmo y la dinámica que está detrás de todos los videojuegos y programas que utiliza cotidianamente, lo que le permitirá reforzar la comprensión sobre el ciclo de desarrollo de un programa:

- [Curso de programación desde cero.](#)
- [Tutorial de algoritmos de programación.](#)





Actividades de aprendizaje recomendadas

Una vez comprendida cada una de las fases, resulta importante medir los conocimientos adquiridos. Le recomendamos realizar las siguientes actividades. Recuerde que estas actividades no son calificadas.

1. Revise los siguientes materiales complementarios:

- a. Realice una lectura comprensiva de la unidad 3 – sección 3.1 de la [guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#).
- b. Revise los videos presentados previamente. Tome nota de lo más importante y de las dudas que se le presenten, para resolverlas con su tutor.

2. Analice las fases del ciclo de desarrollo de un programa, considerando el siguiente procedimiento:

- a. Luego de revisar los apartados conceptuales de la unidad, analice cada una de las fases del ciclo de desarrollo de un programa. Responda a cada una de las siguientes preguntas planteadas:
 - ¿Qué quiere automatizar el cliente?
 - ¿Cómo se puede automatizar lo que quiere el cliente?
 - ¿Cómo se representan las especificaciones de diseño en un lenguaje de programación?
 - ¿Los programas cumplen a detalle las especificaciones de diseño?
- b. Finalmente, comparta sus valoraciones en el espacio no calificado que se habilitará en el EVA para el efecto. Recuerde que, a través de este tipo de actividad, podrá conseguir la retroalimentación tanto de su tutor como de sus compañeros.

Nota. Por favor, conteste las actividades en su cuaderno de apuntes o en un documento Word.





Semana 6

Unidad 3. Ciclo de desarrollo y primitivas algorítmicas

En esta semana identificará las primitivas para la creación de algoritmos y el uso de miniespecificaciones, herramientas que facilitan la representación de un problema real, de forma descriptiva y entendible por la computadora.

A través de estas miniespecificaciones, se darán los primeros pasos para trasladar algo entendible por los humanos, en algo entendible por la computadora. Finalmente, se validarán los algoritmos y miniespecificaciones, realizadas mediante pruebas de escritorio.

3.2. Primitivas algorítmicas

Las primitivas algorítmicas se utilizan en la solución de un problema en forma de algoritmo para posteriormente ser codificado de acuerdo con la sintaxis de un lenguaje de programación. En los algoritmos se usan palabras y frases del lenguaje natural sujetas a unas determinadas reglas. Todo algoritmo consta básicamente de un conjunto de primitivas algorítmicas.

En la figura 7 se presentan los tipos de primitivas algorítmicas y su representación dentro de un algoritmo y diagrama de flujo.



Figura 7
Tipos de primitivas algorítmicas

| Tipo primitiva | Algoritmo | Diagrama de flujo |
|--------------------------|-------------------------------------|-------------------------------------|
| Primitivas de inicio | Inicio | Inicio |
| Primitivas de entrada | Leer nota 1 | Leer nota 1 |
| | Leer nota 2 | Leer nota 2 |
| Primitivas de asignación | suma ← nota1 + nota2 | suma ← nota1 + nota2 |
| Primitivas de salida | Escribir "El promedio es", promedio | Escribir "El promedio es", promedio |
| Primitivas de fin | Fin | Fin |

Nota. Cabrera, M., 2025.

Revise la sección 3.2 de la [Unidad 3 de la guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#), para distinguir detalladamente los tipos de primitivas algorítmicas.

3.3. Diseño de miniespecificaciones

Una miniespecificación es la representación abstracta y simbólica que muestra de mejor manera las operaciones que un programa debe hacer con los datos. La miniespecificación es el paso siguiente al desarrollo de algoritmos. Se les considera además como una herramienta para la especificación sistemática de procesos computacionales, sin dejar de ser entendible por humanos.

Para conocer las principales características de una miniespecificación, le invito a revisar la sección 3.3 de la Unidad 3 de la guía didáctica de Cabrera, M. y Tenesaca, G. (2018).

Para ilustrar de mejor manera los cambios que se realizan al convertir un algoritmo a miniespecificación, en la figura 8 se presenta como ejemplo el cálculo de la longitud de una circunferencia de radio r.

Figura 8

Miniespecificación para calcular la longitud de una circunferencia

| | | Algoritmo | Miniespecificación |
|-----------------------------|----|---|---|
| Inicio | 1. | Inicio | Inicio |
| Declaración de variables | 2. | Decimal: r, π, L | $r, d[1 - n]$ $\pi, d[1 - n]$ $L, d[1 - n]$ |
| Inicialización de variables | 3. | | $r \leftarrow 0$ $\pi \leftarrow 3.1416$ $L \leftarrow 0$ |
| Lectura de datos | 4. | Leer L | $\gg r$ |
| Proceso | 5. | Calcular: $L \leftarrow 2 \times \pi \times r$ | $L \leftarrow 2 \times \pi \times r$ |
| Salida de datos | 6. | Escriba "La longitud de la circunferencia es igual a", L | \ll "La longitud de la circunferencia es igual a", $+L$ |
| Fin | 7. | Fin | Fin |

Nota. Cabrera, M., 2025.

Como se puede observar en el ejemplo de la figura 8, la miniespecificación es una versión más sintetizada del algoritmo y se orienta a un paso previo en la codificación de programas.

3.4. Pruebas de escritorio

Las pruebas de escritorio son pruebas manuales que se encargan de visualizar el comportamiento de los estados de las variables en el transcurso de la ejecución de un algoritmo o miniespecificación.

Continuando con el ejemplo anterior, la figura 9 presenta la prueba de escritorio obtenida.

Figura 9

Prueba de escritorio

| | Variables | | |
|-----------------------|-----------|-------|--------|
| | r | L | Pi |
| Valores de entrada | 0 | 0 | 3.1416 |
| Valores en el proceso | 8 | 50,26 | 3.1416 |
| Valore de salida | | 50,26 | |

Nota. Cabrera, M., 2025.

Para la presente semana se debe realizar una lectura comprensiva del siguiente documento complementario de la asignatura:

- Cabrera, M. y Tenesaca, G. (2018). Guía didáctica de Algoritmos y resolución de problemas. Loja, Ecuador: Editorial Universidad Técnica Particular de Loja.

Lectura: [Unidad 3: Ciclo de desarrollo y primitivas algorítmicas – Sección 3.1.](#)

Además, para reforzar los contenidos, se recomienda realizar la revisión de los siguientes videos:

- [¿Cómo hacer una miniespecificación?](#)

En este video se presenta cómo se puede transformar un algoritmo a miniespecificación, mediante el ejemplo de Fibonacci.

- [UTPL PRUEBAS DE ESCRITORIO \[\(INFORMÁTICA\) \(LÓGICA DE LA PROGRAMACIÓN\)\]](#)

En este video se explica la comprobación de un algoritmo, mediante pruebas de escritorio, para determinar si la solución resuelve el problema dado.

Luego de revisar los materiales de apoyo, ahora con mayor seguridad estamos en la capacidad de responder preguntas como: ¿qué es una miniespecificación?, ¿para qué me sirven las miniespecificaciones?, ¿es lo mismo la miniespecificación que un algoritmo?

Le invito a responder estas preguntas, sin olvidar que una miniespecificación, también conocida como **Pseudocódigo**, es una representación abstracta y simbólica, que sin llegar a la rigidez de la sintaxis de un lenguaje de programación ni a la fluidez del lenguaje natural, permite codificar un programa o algoritmo con mayor agilidad.



Actividades de aprendizaje recomendadas

Ahora que tiene los conceptos más claros, le recomendamos realizar las siguientes actividades para medir los conocimientos adquiridos. Recuerde que estas actividades de aprendizaje no son calificadas.

1. Revise los siguientes materiales complementarios:
 - a. Realice una lectura comprensiva de la [unidad 3 de la guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#).
 - b. Revise los videos presentados previamente. Tome nota de lo más importante y las dudas que se le presenten, para resolverlas con su tutor.
2. Realizar un algoritmo, la miniespecificación y el diagrama de flujo para calcular el número de días transcurridos de una persona desde su año de nacimiento.

Nota. Por favor, conteste las actividades en un cuaderno de apuntes o en un documento Word.

3. Desarrolle la autoevaluación 3, que se encuentra al final de la unidad 3. La misma consiste en un cuestionario de preguntas objetivas relacionadas con las temáticas de la quinta y sexta semana.





Autoevaluación 3

Analice las siguientes preguntas de opción múltiple y seleccione la respuesta correcta:

1. ¿Cuál es el orden correcto de las fases de desarrollo de un programa?
 - a. Análisis, codificación, diseño, pruebas e implementación.
 - b. Codificación, análisis, diseño, pruebas e implementación.
 - c. Análisis, diseño, codificación, pruebas e implementación.
2. En el ciclo de desarrollo de un programa, ¿cuál es la fase en la que se analiza una situación del mundo real?
 - a. Diseño.
 - b. Pruebas e implementación.
 - c. Análisis.
3. En el ciclo de desarrollo de un programa, ¿cuál es la fase que se caracteriza por ser importante para el cliente, más que para el desarrollador?
 - a. Pruebas e implementación.
 - b. Codificación.
 - c. Diseño.
4. En el ciclo de desarrollo de un programa, ¿cuál de las fases, da respuesta a la siguiente pregunta?, ¿cómo se puede automatizar lo que quiere el cliente?
 - a. Diseño.
 - b. Codificación.
 - c. Análisis.





5. ¿Cuál es el concepto apropiado para definir un algoritmo de programación?
- Conjunto ordenado y finito de asignaciones, procesos, cálculos y decisiones que permiten a un programa satisfacer una unidad de funcionalidad dada.
 - Conjunto ordenado e infinito de asignaciones, procesos, cálculos y decisiones que permiten a un programa satisfacer una unidad de funcionalidad dada.
 - Conjunto sin ordenar y finito de asignaciones, procesos, cálculos y decisiones que permiten a un programa satisfacer una unidad de funcionalidad dada.
6. Si tenemos como entregable, la aceptación del programa que hemos realizado, ¿a qué fase del ciclo de desarrollo de un programa estamos haciendo referencia?
- Análisis.
 - Diseño.
 - Pruebas e implementación.
7. Se desea desarrollar un algoritmo para calcular el costo de un terreno rectangular, cuyo valor se calcula multiplicando el área del terreno, por el valor del metro cuadrado, más el 5 % de este valor, que corresponde al pago de impuestos. El algoritmo es el siguiente:
- Inicio.
 - Se solicitan las dimensiones del terreno (largo y ancho).
 - Se calcula el área del terreno.
 - Se calcula el valor del terreno.
 - Se calcula el valor del impuesto.
 - Se calcula el total, sumando el valor del terreno, más el impuesto calculado.
 - Fin.

Evalúe el algoritmo y determine ¿qué información le falta para poder resolver el problema?

- a. El área del terreno.
- b. El costo por metro cuadrado.
- c. El valor del impuesto.

8. Se desea desarrollar un algoritmo que permita mostrar las tablas de restar de un número, solo en el caso de que el mismo esté entre 5 y 20. El algoritmo debe presentar un mensaje, en caso de que no haya seleccionado un número dentro del rango establecido. Seleccione la opción correcta para el caso presentado:

- a.
 - 1. Inicio.
 - 2. Se pregunta la tabla del número que se desea.
 - 3. Si la tabla está entre 5 y 20, entonces se procede a imprimir la tabla de restar indicada.
 - 4. Si la tabla no está en el rango, entonces se procede a mostrar un mensaje de error en pantalla.
 - 5. Fin.
- b.
 - 1. Inicio.
 - 2. Se pregunta la tabla del número que se desea.
 - 3. Si la tabla está entre 5 y 19, entonces se procede a imprimir la tabla de restar indicada.
 - 4. Si la tabla no está en el rango, entonces se procede a mostrar un mensaje de error en pantalla.
 - 5. Fin.
- c.
 - 1. Inicio.
 - 2. Se pregunta la tabla del número que se desea.
 - 3. Si la tabla está entre 5 y 20, entonces se procede a mostrar un mensaje de error en pantalla.
 - 4. Si la tabla no está en el rango, entonces se procede a imprimir la tabla de restar indicada.
 - 5. Fin.





9. ¿Cuál es la función principal de una miniespecificación dentro del proceso de desarrollo de programas?
- a. Sustituir al algoritmo para simplificar el diseño.
 - b. Representar de forma abstracta las operaciones que el programa realizará con los datos.
 - c. Convertir directamente el algoritmo en código fuente.
10. ¿Qué permiten observar las pruebas de escritorio en un algoritmo o miniespecificación?
- a. La eficiencia en el uso de memoria del programa.
 - b. El comportamiento de las variables durante la ejecución.
 - c. La traducción del algoritmo a un lenguaje de programación.

[Ir al solucionario](#)



Con esto, hemos terminado los contenidos del primer bimestre. Continúe con el mismo entusiasmo de aprender, en el segundo bimestre.

Resultados de aprendizaje 1 a 4:

- Discute la importancia de los algoritmos en el proceso de solución de problemas.
- Identifica las propiedades necesarias para un buen algoritmo.
- Crea algoritmos para solucionar problemas simples.
- Aplica estrategias efectivas de depuración.

Contenidos, recursos y actividades de aprendizaje recomendadas



Semana 7

Actividades finales del bimestre

En la presente semana se recomienda realizar un repaso general de las unidades 1, 2 y 3. Además, se deben realizar las actividades planificadas según las directrices dadas en cada una de ellas.

Contenidos, recursos y actividades de aprendizaje recomendadas



Semana 8

Actividades finales del bimestre

En la presente semana podemos tener un espacio para repasar los contenidos de la unidad 3. Con esto, podrá estar más preparado para la evaluación presencial del primer bimestre.



Al finalizar esta semana, se encuentra planificada la evaluación presencial. Realice todas las actividades de aprendizaje recomendadas y calificadas, esto le ayudará a cumplir con éxito esta actividad.



Para consolidar los conocimientos adquiridos, le invito a explorar el siguiente crucigrama, enfocado en los fundamentos y principios básicos de programación. Esta actividad le permitirá repasar y afianzar los conceptos del primer bimestre mediante un ejercicio dinámico que facilita la memoria y su aplicación práctica en la resolución de problemas.

[Fundamentos y Principios Básicos de Programación.](#)

Con estas bases sólidas, le invito a continuar con el mismo entusiasmo de aprender en el segundo bimestre.





Segundo bimestre

Resultados de aprendizaje 3 y 4:

- Crea algoritmos para solucionar problemas simples.
- Aplica estrategias efectivas de depuración.

Por medio de los resultados de aprendizaje se analiza, diseña, crea y evalúa soluciones algorítmicas para dar solución a problemas planteados. Identificando claramente las variables, estructuras lógicas y los pasos a seguir plasmados en primitivas algorítmicas y evaluados mediante pruebas de escritorio.

Contenidos, recursos y actividades de aprendizaje recomendadas

Recuerde revisar de manera paralela los contenidos con las actividades de aprendizaje recomendadas y actividades de aprendizaje evaluadas.



Semana 9

Unidad 4. Estructuras lógicas condicionales

¿Qué son las estructuras lógicas condicionales y para qué me sirven en el desarrollo de algoritmos?

Aplicar estructuras lógicas condicionales permite resolver problemas más complejos. No todos los problemas pueden resolverse empleando estructuras secuenciales. Cuando hay que tomar una decisión, se deben utilizar las estructuras condicionales. En nuestra vida diaria se nos presentan situaciones como, por ejemplo: si quiero identificar entre un grupo de personas cuáles son mayores de edad, necesitamos establecer una condición, para determinar que la variable edad sea ≥ 18 años.



En otras palabras, con las estructuras de condición se hace una pregunta y, dependiendo de la respuesta, se ejecuta una acción antes de continuar con la siguiente instrucción definida en el algoritmo. Existen dos tipos de estructuras lógicas condicionales: simples y compuestas.

A continuación, en la semana 9 nos centraremos en el estudio de las estructuras condicionales simples y en la semana 10 en el estudio de las estructuras condicionales compuestas.

4.1. Estructura condicional simple

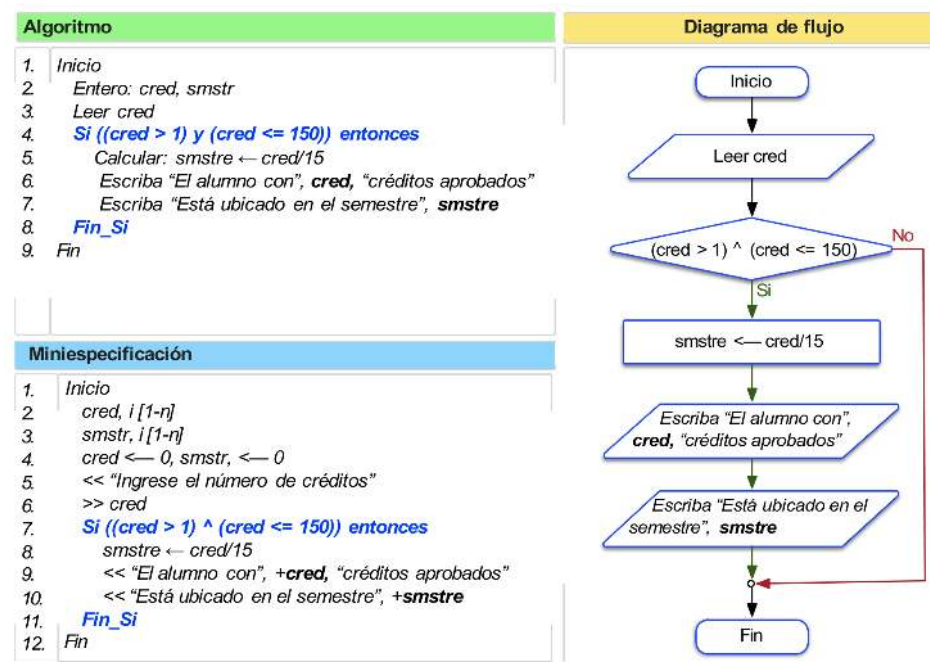
Las estructuras condicionales simples son de alternativa única, en estos casos usted no emprende una acción especial si no cumple la condición. Las condiciones se especifican usando expresiones lógicas. La representación de una estructura condicional se hace con palabras *Si – Entonces – Fin Si*. En la sección 4.1 de la [unidad 4 de la guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#), usted puede encontrar la sintaxis para una estructura condicional simple y su representación mediante un diagrama de flujo.

Para ilustrar de mejor manera el uso de estructuras lógicas condicionales, se detalla el ejercicio, el algoritmo, la miniespecificación y el diagrama de flujo en la figura 10.

Ejercicio 4.1: se requiere ubicar de semestre a un estudiante en función del número de créditos. Un programa académico de pregrado tiene un total de 150 créditos, suponga que por semestre el alumno puede cursar 15 créditos, si la variable *cred* es el número de créditos totales aprobados por el estudiante, diseñe un algoritmo que calcule el semestre de ubicación del estudiante, validando que el número de créditos leídos esté en el intervalo de [1;150] créditos.



Figura 10
Ejercicio. Estructura condicional simple



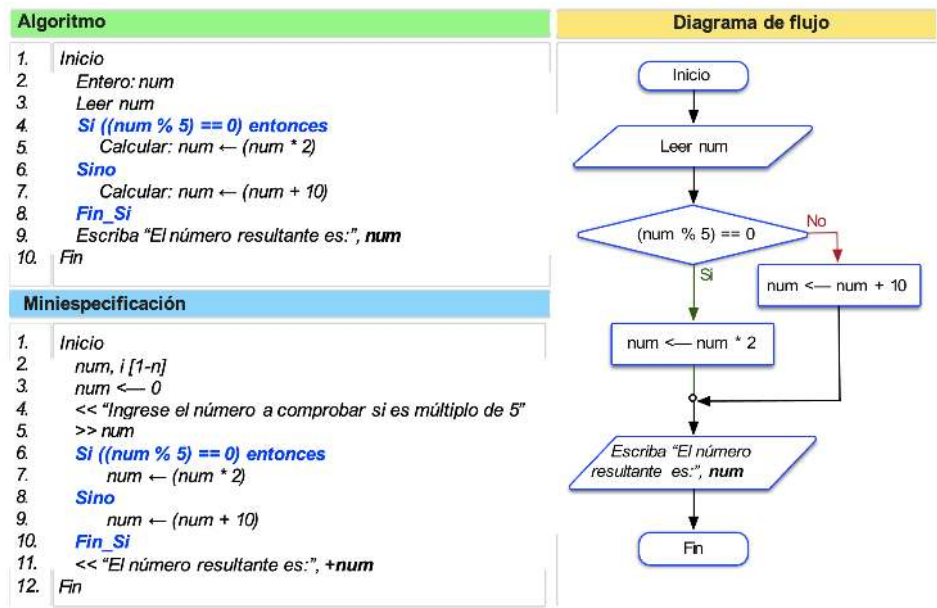
Nota. Cabrera, M., 2025.

4.2. Estructura condicional compuesta

Por otro lado, las condicionales compuestas tienen dos alternativas: la ejecución de instrucciones cuando la condición es comprobada como **verdadera** y la ejecución de instrucciones cuando la condición es comprobada como **falsa**. La representación de una estructura condicional se hace con palabras **SI-ENTONCES- SINO-FINSI**. En la sección 4.1 [Unidad 4 de la guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#), puede encontrar la sintaxis para una estructura condicional compuesta y su representación, mediante un diagrama de flujo.

A continuación, en la figura 11, se presenta el algoritmo, miniespecificación y el diagrama de flujo para determinar si un número es múltiplo de 5. Si el número es múltiplo de 5, se debe multiplicar por 2, caso contrario, se debe sumar el valor de 10. Presentar el valor resultante.

Figura 11
Ejercicio. Estructura condicional compuesta



Nota. Cabrera, M., 2025.

Para continuar con el estudio de los contenidos de la presente semana es importante apoyarse en los materiales principales de la asignatura. Por este motivo, le invito a realizar una lectura comprensiva de los siguientes documentos complementarios:

- Cabrera, M. y Tenesaca, G. (2018). Guía didáctica de Algoritmos y resolución de problemas. Loja, Ecuador: Editorial Universidad Técnica Particular de Loja.

Lectura: [Unidad 4. Estructuras lógicas condicionales, sección 4.1 y 4.2.](#)

- Trejos Buriticá, O. (2017). Lógica de programación: (ed.). Ediciones de la U.

Lectura: [Capítulo 7. Decisiones, sección 7.1.](#)

Además, para reforzar los contenidos, le recomiendo visualizar el siguiente video: [LP #9| Decisión y Algoritmo](#). En este video aprenderá qué es una decisión y cómo aplicarla en un algoritmo; además, se realizan ejemplos para reforzar lo aprendido.

¿Revisó el material de apoyo? ¡Muy bien! Con este video hemos reforzado algunos aspectos muy importantes en el uso de estructuras condicionales, a continuación, hacemos un resumen de estos aspectos:

- El resultado de evaluar una condición dentro de la estructura siempre nos dará valores booleanos, es decir un valor de verdad o un valor falso, de esto depende la ejecución de instrucciones dentro de la estructura. ¿Qué es un algoritmo?
- Se puede utilizar dentro de la condición operadores de comparación relacionados para comparar dos operandos del mismo tipo. Los operandos más utilizados son: $=$, $>$, $<$, $>=$, $<=$ y $<>$.
- Cuando utilizamos el operador AND, dos condiciones deben ser verdaderas para que ocurra una acción resultante.
- Con el operador OR, al menos una de las dos condiciones debe ser verdadera para que se presente una acción resultante.
- Y, cuando combinamos los operadores AND y OR en una expresión, debemos recordar que el operador AND tiene mayor precedencia.



Actividades de aprendizaje recomendadas

Una vez revisados los materiales complementarios, es importante medir los conocimientos adquiridos. Por esta razón se han propuesto las siguientes actividades de aprendizaje. Recuerde que estas actividades no son calificadas.

1. Revise los siguientes materiales complementarios:

- a. Realice una lectura comprensiva de la sección 4.2 y 4.3 de la [unidad 4 de la guía didáctica de Cabrera, M. y Tenesaca, G.](#)



(2018), así como también de la [sección 7.1 del capítulo 7. Decisiones de Trejos, O. \(2017\).](#)

- b. Revise el video presentado previamente. Tome nota de lo más importante y las dudas que se le presenten para luego resolverlas con su tutor.

2. Realizar la miniespecificación y el diagrama de flujo del siguiente ejercicio:

- Ingrese un número entero y determine si el número es positivo, negativo o si es 0.
- El desarrollo del ejercicio le ayudará a comprender de mejor manera el siguiente tema de estudio: Estructura condicional Dependiendo – De.

Nota. Por favor, conteste las actividades en su cuaderno de apuntes o en un documento Word.

Contenidos, recursos y actividades de aprendizaje recomendadas



Semana 10

Unidad 4. Estructuras lógicas condicionales

4.3. Estructura lógica dependiendo de

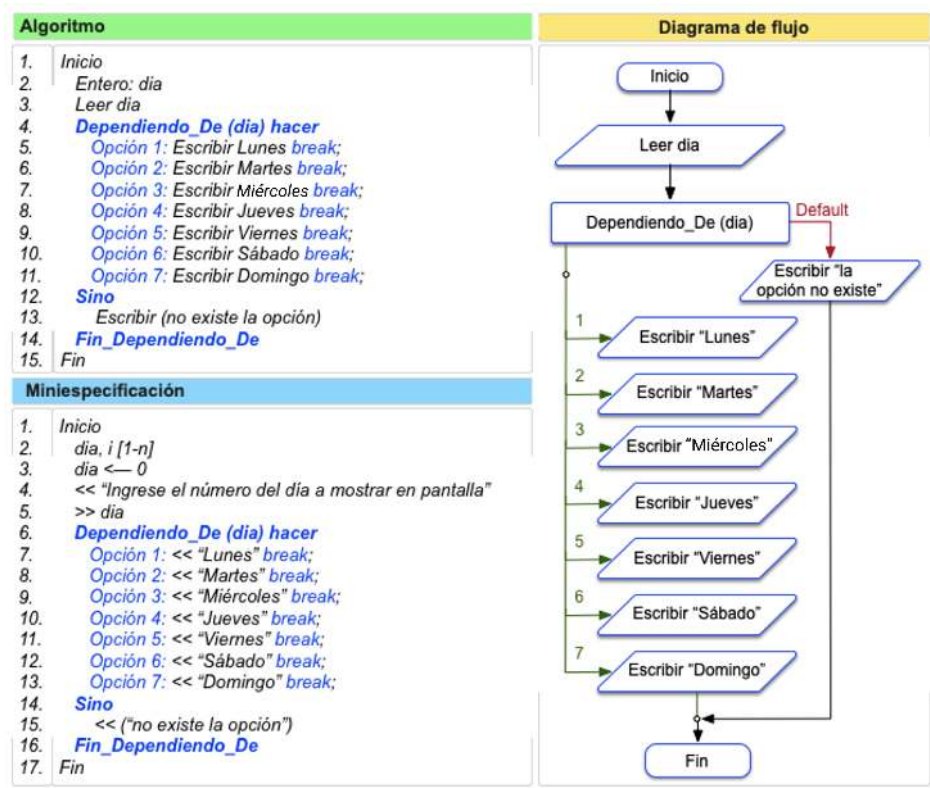
Con frecuencia, para la resolución de muchos problemas, es necesario que existan más de dos elecciones posibles.

La estructura lógica, *Dependiendo De*, es una estructura de decisión múltiple que evalúa una expresión que puede tomar n valores distintos. Con base en la opción que se valide en la condición, se realizará una de las n acciones, o lo que es igual, el flujo del algoritmo seguirá un determinado camino entre los n caminos posibles.



Para ilustrar el uso de la estructura, Dependiendo De, se realiza el siguiente ejercicio: Se desea diseñar una solución que escriba los nombres de los días de la semana, en función del valor de una variable **día** introducida por teclado. A continuación, en la figura 12, se presenta el algoritmo, la miniespecificación y el diagrama de flujo.

Figura 12
Ejercicio. Estructura condicional Dependiendo_De



Nota. Cabrera, M., 2025.

A continuación, lo invitamos a revisar los siguientes documentos de aprendizaje que le ayudarán a reforzar la temática de la presente unidad.

Siguiendo con el estudio de los contenidos de la presente unidad, es importante apoyarse en los materiales principales de la asignatura. Por este motivo, le invito a realizar una lectura comprensiva de los siguientes documentos complementarios:

- Cabrera, M. y Tenesaca, G. (2018). Guía didáctica de algoritmos y resolución de problemas. Loja, Ecuador: Editorial Universidad Técnica Particular de Loja.

Lectura: [unidad 4. Estructuras lógicas condicionales, sección 4.3.](#)

- Trejos Buriticá, O. (2017). Lógica de programación: (ed.). Ediciones de la U.

Lectura: [capítulo 7. Decisiones, sección 7.2.](#)

Además, para reforzar los contenidos, le recomiendo visualizar el siguiente video: "[Ejemplo de condicional múltiple](#)". En este video, se presenta un ejemplo de la estructura condicional múltiple en PSeInt, con sus respectivas fases de desarrollo.

¿Qué le pareció el video? Con este material hemos reforzado algunos aspectos muy importantes en la aplicación de la estructura condicional Dependiendo – De.



Recuerde que esta estructura condicional es una estructura de decisión múltiple que evalúa una expresión que puede tomar n valores distintos. Según se elija uno de estos valores en la condición, se realizará una de las acciones definidas por cada opción de la condición.





Actividades de aprendizaje recomendadas

Para medir los conocimientos de la presente unidad, le invito a desarrollar las siguientes actividades. Recuerde que estas actividades no son calificadas. ¡Ánimo, vamos avanzando muy bien con la asignatura!

1. Revise los siguientes materiales complementarios:

- a. Realice una lectura comprensiva de la [unidad 4, sección 4.3 de la guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#), así como también de la [sección 7.2 del capítulo 7 de Trejos, O. \(2017\)](#).
- b. Revise el video presentado previamente. Tome nota de lo más importante y las dudas que se le presenten para luego resolverlas con su tutor.

2. Realice la miniespecificación y el diagrama de flujo del siguiente problema:

- Elaborar una calculadora de 4 operaciones, con las siguientes opciones: (1) suma, (2) resta, (3) multiplicación y (4) división.
- Posteriormente, debe solicitar dos números, asimismo debe solicitar el número de la operación a realizar sobre los números solicitados previamente. Es decir, si la opción seleccionada es 1, debe realizar la suma de los números y presentar el resultado. Así sucesivamente para las 4 operaciones.

Nota. Por favor, conteste las actividades en su cuaderno de apuntes o en un documento Word.

3. Hemos concluido el estudio de la cuarta unidad; comprobaremos lo aprendido, realizando la autoevaluación 4. Esta consiste en un cuestionario de preguntas objetivas relacionadas con las temáticas de la novena y décima semana.





Autoevaluación 4

1. Cuando la condición es verdadera, solo se ejecutan las primitivas de control diseñadas, en la parte verdadera de la condición ($p_1, p_2, p_3, \dots, p_i, \dots, p_n$) y cuando la condición es falsa, se ejecutan solo las estructuras de control, diseñadas en la parte falsa de cumplimiento de la condición ($q_1, q_2, q_3, \dots, q_i, \dots, q_n$).
 - a. Condicional simple.
 - b. Condicional compuesta.
 - c. Bucle Para.
2. ¿Cuál es la estructura lógica que es útil en el diseño de algoritmos, que requieren el diseño de menús de opciones o diseños que requieren la modularización parcial de la lógica de control del algoritmo?
 - a. Condicional simple.
 - b. Condicional compuesta.
 - c. Dependiendo_De.
3. ¿Cuál de las siguientes instrucciones, le permitirá incrementar el salario de un empleado en un 25 %, siempre y cuando gane 750 dólares o más?

| | |
|---------------|--|
| | Si salario \geq 750 entonces |
| Alternativa 1 | salario \leftarrow (salario * 0.25) |
| | Fin Si |

| | |
|---------------|--|
| | Si salario \geq 750 entonces |
| Alternativa 2 | salario \leftarrow salario + (salario * 0.25) |
| | Fin Si |

| | |
|---------------|---------------------------------------|
| Alternativa 3 | Si (salario > 750) entonces |
|---------------|---------------------------------------|



salario ← salario + (salario * 0.25)

Fin Si

4. Se desea generar un programa que permita determinar, si alguien es mayor de edad, en Ecuador (desde los 18 años). Si es mayor de edad, debe presentar la edad. En caso de que no sea mayor de edad, presentar un mensaje de "incorrecto". ¿Cuál de las siguientes sentencias permite obtener el resultado deseado?

| | |
|---------------|---------------------------------|
| Alternativa 1 | Si edad > 18 entonces |
| | >> edad |
| | Sino |
| | << "incorrecto" |
| | Fin Si |

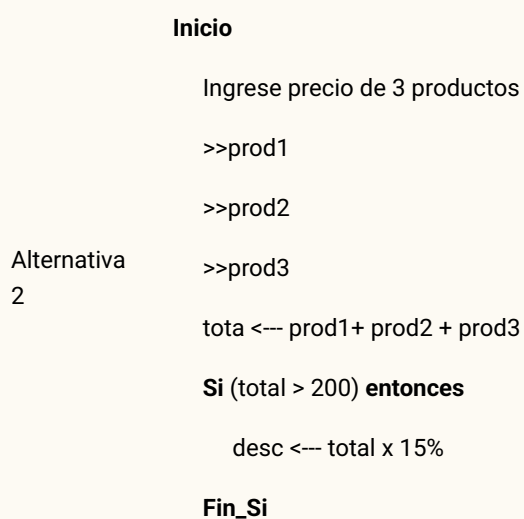
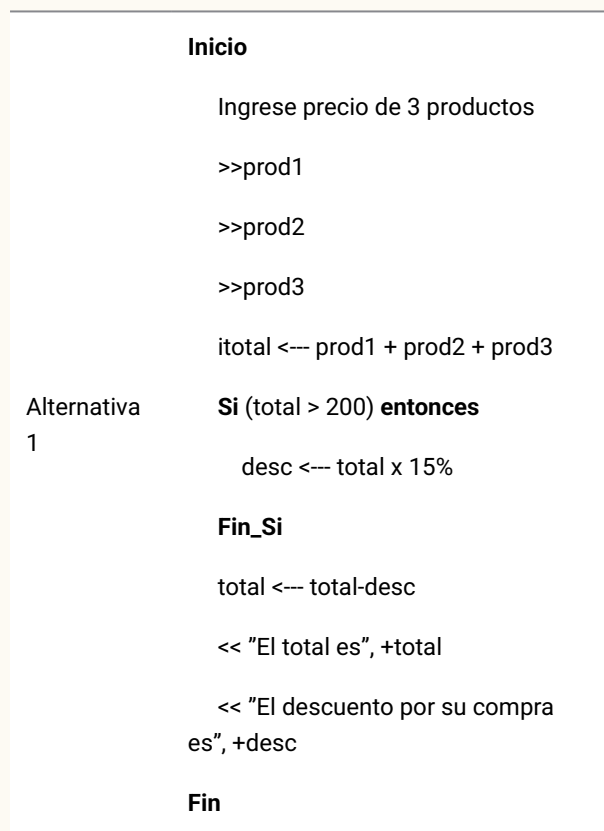
| | |
|---------------|----------------------------------|
| Alternativa 2 | Si edad >= 18 entonces |
| | >> edad |
| | Sino |
| | << "incorrecto" |
| | Fin Si |

| | |
|---------------|----------------------------------|
| Alternativa 3 | Si edad >= 18 entonces |
| | << edad |
| | Sino |
| | <<"incorrecto" |
| | Fin Si |

5. Seleccione el algoritmo que permita determinar si un usuario que adquiere 3 productos, su costo final es mayor a 200 dólares y se



realiza un descuento del 15 %; el algoritmo debe mostrar el total, descuento:



```

total <-- total + desc

<<" El total es", +total

<<" El descuento por su compra
es", +desc

```

Fin

Inicio

Ingrese precio de 3 productos

>>prod1

>>prod2

>>prod3

total \leftarrow prod1 + prod2 + prod3

Alternativa
3

Si (total < 200) **entonces**

desc \leftarrow total x 15%

total \leftarrow total – desc

Fin_Si

<<"El total es", +total

<<"El descuento por su compra
es", +desc

Fin

6. Determine las respuestas de los siguientes enunciados, según correspondan, verdadero (V) o falso (F):

La "condición" se refiere a una decisión que se debe tomar en la lógica de control. ()

La "decisión" implica la solución a una pregunta que se estructura en ciclo de repetición. ()

7. Con base en las siguientes expresiones, determine qué estructura de decisión o control se está usando:

1. **Si** ((área \geq 10) or (área = 50), **entonces**

2. \ll área \leftarrow área + 1000

3. **Fin Si.**

a. Acumulador.

b. Condicional compuesta.

c. Condicional simple.

8. Se desea generar un programa que permite obtener y presentar el cubo de un número, siempre y cuando, el número sea menor a 60, caso contrario, debería obtener y presentar el cuadrado del número indicado. ¿Cuál de las siguientes sentencias permite obtener el resultado deseado?

| | |
|-------------|---------------------------------------|
| | Si número < 60 entonces |
| | cuadrado \leftarrow número 2 |
| | \ll cuadrado |
| Sentencia 1 | Sino |
| | cubo \leftarrow número 3 |
| | \ll cubo |
| | Fin Si |

| | |
|-------------|---------------------------------------|
| | Si número < 60 entonces |
| | cuadrado \leftarrow número 3 |
| | \ll cuadrado |
| Sentencia 2 | Sino |
| | cubo \leftarrow numero 2 |
| | \ll cubo |





Fin Si

Sentencia 3

Si número ≤ 60 entonces

cuadrado \leftarrow número 3

\ll cuadrado

Sino

cubo \leftarrow número 2

\ll cubo

Fin Si

9. Se desea generar un programa que permita determinar, si alguien tiene la edad para entrar a la universidad en un país (el país tiene como ley que los estudiantes pueden ingresar a la universidad solo si tienen 20 años). Si la persona cumple con la edad, presente un mensaje de "ingreso exitoso". En caso de que no tenga la de edad, presente un mensaje de "edad incorrecta":

a.

Si edad > 20 entonces

\ll "ingreso exitoso"

Sino

\ll "edad incorrecta"

Fin Si

b.

Si edad ≥ 20 entonces

\ll "ingreso exitoso"

Sino

\ll "edad incorrecta"

Fin Si

Si edad $== 20$ entonces

<< "ingreso exitoso"

Sino

C.

<< "edad incorrecta"

Fin Si

10. ¿Cuál es la estructura, en la que cada alternativa es disyuntiva y se ejecuta independiente de las otras?

- a. Condicional simple.
- b. Condicional DD.
- c. Condicional compuesta.

[Ir al solucionario](#)

Contenidos, recursos y actividades de aprendizaje recomendadas



Semana 11

Unidad 5. Estructuras lógicas repetitivas

¿Qué son las estructuras lógicas repetitivas y para qué sirven en el desarrollo de algoritmos?

Aplicar estructuras lógicas repetitivas le permite resolver problemas más complejos que requieran la ejecución de operaciones por varias interacciones. Aunque tomar decisiones es lo que hace que las computadoras parezcan inteligentes, la creación de ciclos hace que la programación de computadoras sea más eficiente. Cuando usted usa un ciclo, un conjunto de instrucciones, opera en múltiples conjuntos separados de datos. Las ventajas de utilizar ciclos repetitivos es el uso de menos instrucciones, menos tiempo requerido para el diseño y la codificación, menos errores y un tiempo de compilación más breve.

¿Cómo funciona una estructura lógica repetitiva?



En este tipo de estructuras, o también llamadas ciclos repetitivos o bucles, se repite un mismo conjunto de instrucciones mientras una condición sigue siendo verdadera. Esto continúa hasta que la condición se vuelve falsa y se dé por finalizada la ejecución de la estructura y continúe con la siguiente instrucción. En esta asignatura se revisan tres tipos de estructuras lógicas repetitivas: Mientras-Que – Hacer, Hacer – Hasta y Para – Hacer. Se recomienda hacer el mayor número de ejercicios para comprender de mejor manera esta temática.

En esta unidad también es importante conocer dos conceptos fundamentales: contadores y acumuladores. Un contador es una variable que se utiliza en un ciclo repetitivo y se usa para almacenar valores cuyos incrementos o decrementos son en forma constante por cada iteración. Por otro lado, un acumulador es una variable que, por cada iteración de un ciclo, permite almacenar valores cuyos incrementos o decrementos son en forma variable.

5.1. Estructura lógica repetitiva: mientras-que – hacer

Cuando se ejecuta la instrucción, Mientras, en primera instancia, se evalúa una condición. Si la condición es falsa, no se toma ninguna acción y el algoritmo continúa con la siguiente instrucción. Si la expresión booleana es verdadera, entonces se ejecutan las instrucciones dentro del ciclo repetitivo, después de lo cual se evalúa nuevamente la condición. Este proceso se repite una y otra vez mientras la condición sea verdadera.

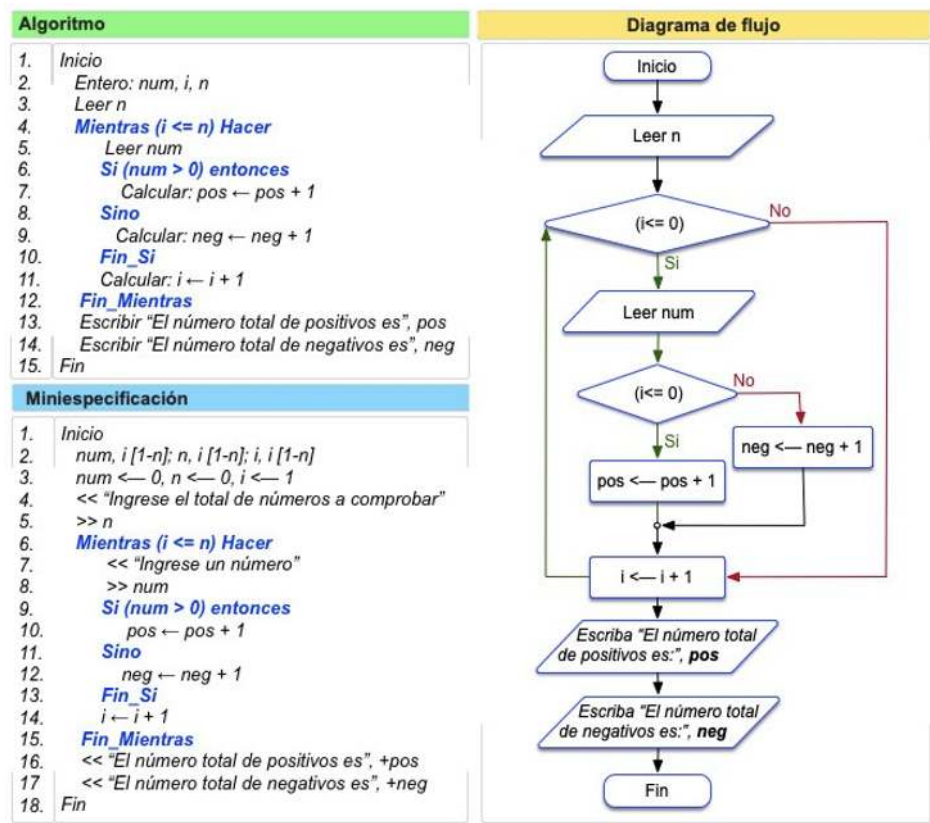
La característica principal en la estructura Mientras es que para ejecutarse por lo menos una vez el cuerpo de instrucciones dentro del ciclo, la condición debe ser verdadera.

Le recomiendo revisar la sección 5.1 de la [unidad 5 de la guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#), ahí puede encontrar la sintaxis para la estructura lógica repetitiva Mientras y su representación mediante diagramas de flujo. A continuación, en la figura 13 se presenta un ejemplo que le ayudará a comprender de mejor manera la temática.



Ejemplo 5.1: se requiere determinar cuántos números positivos y negativos se ingresan en un total de n elementos.

Figura 13
Ejercicio. Estructura lógica repetitiva Mientras–que–Hacer



Nota. Cabrera, M., 2025.

Analizado el ejercicio propuesto, daremos inicio al estudio de la siguiente estructura lógica repetitiva.

5.2. Estructura lógica repetitiva: hacer–hasta

La estructura hacer – hasta es muy parecida a la estructura mientras, las instrucciones dentro del ciclo se ejecuta una y otra vez mientras la condición es verdadera. Sin embargo, existe una gran diferencia que es la ubicación de la

condición a ser evaluada, en el caso de la estructura mientras la condición se encuentra al inicio de la estructura y en el caso del hacer – hasta se ubica al final.

Es decir, que en la estructura hacer – hasta las instrucciones dentro del ciclo se ejecutan al menos una vez, antes de que se evalúe la condición. Por lo tanto, siempre se ejecuta, al menos una vez, incluso aunque la condición sea falsa.

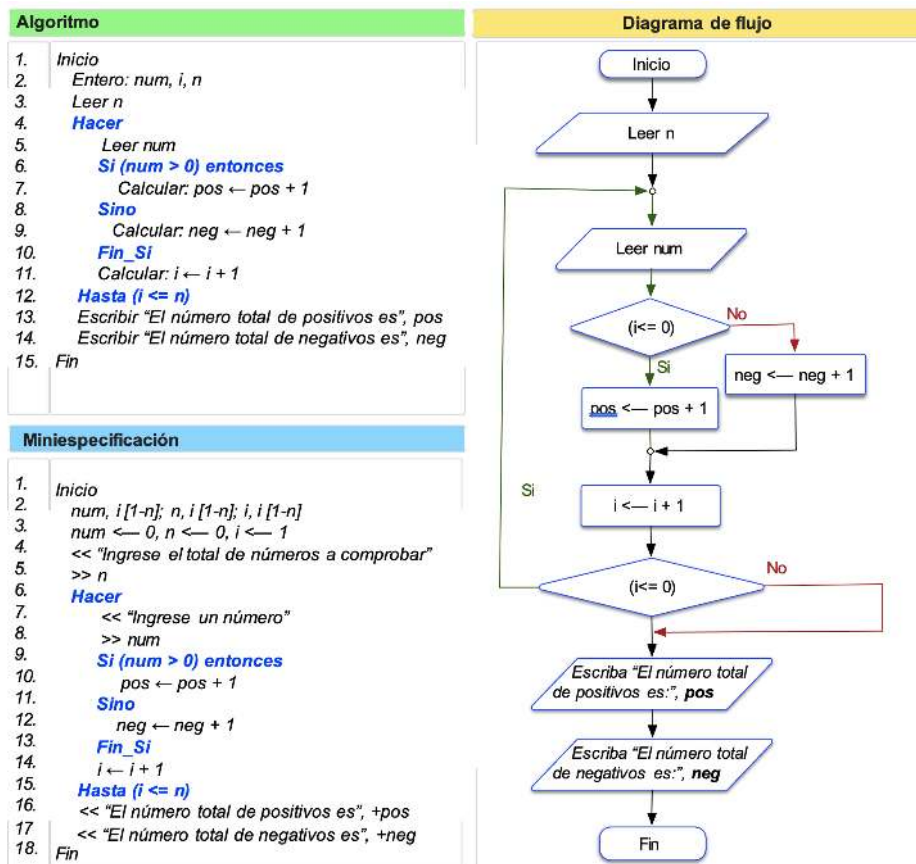
Se recomienda revisar la sección 5.2 de la [Unidad 5 de la guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#), ahí puede encontrar la sintaxis para la estructura lógica repetitivas hacer - hasta y su representación mediante diagramas de flujo.

A continuación, en la figura 14 se presentan las modificaciones del ejercicio anterior utilizando la estructura lógica hacer – hasta.



Figura 14

Ejercicio. Estructura lógica condicional Hacer-Hasta



Nota. Cabrera, M., 2025.

Para iniciar el estudio de los contenidos de la presente semana es importante apoyarse en los materiales principales de la asignatura. Por este motivo, le invito a realizar una lectura comprensiva de los siguientes documentos complementarios:

- Cabrera, M. y Tenesaca, G. (2018). Guía didáctica de algoritmos y resolución de problemas. Loja, Ecuador: Editorial Universidad Técnica Particular de Loja.

Lectura: [unidad 5. Estructuras lógicas repetitivas, sección 5.1 y 5.2.](#)

- Trejos Buriticá, O. (2017). Lógica de programación: (ed.). Ediciones de la U.

Lectura: [capítulo 8. Ciclos, sección 8.2.1 ciclo Mientras-Que y sección 8.2.3 ciclo Hacer-Hasta.](#)

Además, para reforzar los contenidos, le recomiendo visualizar los siguientes videos:

- [Guía Lección 3 - Tipos de estructuras repetitivas.](#)
- [Guía Lección 4 - Estructura Repetitiva Mientras Que \(Parte 1\).](#)
- [Guía Lección 4 - Estructura Repetitiva Mientras Que \(Parte 2\).](#)
- [Guía Lección 5 - Estructura Repetitiva Haga-Hasta.](#)

En estos videos se abordan las estructuras lógicas repetitivas Para, Mientras y Haga-Hasta, explicando su sintaxis, estructura y aplicación en el desarrollo de algoritmos. A través de ejemplos y ejercicios prácticos, se refuerza lo aprendido y se muestra cómo emplear cada tipo de estructura en la resolución de problemas.

Luego de revisar los materiales de apoyo, tenga en cuenta los siguientes puntos, que son primordiales cuando se utilizan estructuras de lógicas repetitivas:

- La diferencia entre un contador y un acumulador es que, mientras el primero va aumentando en una cantidad fija, que por lo general es de uno en uno, el acumulador va aumentando en una cantidad variable.
- Cuando no se conoce el número de veces que tiene que repetirse el ciclo repetitivo, este, se convierte en un dato de entrada y se tiene que pedir al usuario que lo ingrese. Además, se debe considerar que la condición debe llegar en algún momento a ser falsa; si no, se convertiría en un LOOP (ciclo repetitivo infinito, es decir, que nunca termina).
- Cuando el contador es inicializado en 1, la condición en la estructura repetitiva *Mientras* debe ser menor o igual (\leq) que el límite. Y cuando el contador es inicializado en 0, la condición de la estructura repetitiva *Mientras* debe ser únicamente menor ($<$) al límite.



- La estructura lógica repetitiva, *Hacer-Hasta*, se ejecuta al menos una vez, aunque la condición no se cumpla.



Actividades de aprendizaje recomendadas

Una vez comprendida la temática, le recomendamos realizar las siguientes actividades para medir los conocimientos adquiridos. Recuerde que estas actividades no son calificadas.

1. Realice una revisión de los siguientes materiales complementarios:

- a. Realice una lectura comprensiva de la sección 5.1 y 5.2 de la [unidad 5 de la guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#), así como también de la [sección 8.2.1 y 8.2.3 del capítulo 8 de Trejos, O. \(2017\)](#).
- b. Revise los videos previamente presentados. Tome nota de lo más importante y las dudas que se le presenten, para resolverlas con su tutor.

2. Desarrolle el siguiente ejercicio para reforzar lo aprendido:

Realice la miniespecificación y el diagrama de flujo del siguiente ejercicio utilizando la estructura lógica repetitiva Mientras-Que y Hacer-Hasta:

- Ingresar números enteros hasta que digiten 0 y determinar a cuánto es igual el promedio de los números leídos que hayan sido positivos.
- Puede compartir sus respuestas en el EVA para una retroalimentación colaborativa entre su tutor y compañeros.

Nota. Por favor, conteste las actividades en su cuaderno de apuntes o en un documento Word.





Unidad 5. Estructuras lógicas repetitivas

5.3. Estructura lógica repetitiva: para–hacer

Las estructuras lógicas *Mientras-Que* y *Hacer – Hasta* evalúan una condición (expresión lógica) para que se ejecuten las instrucciones dentro del ciclo.

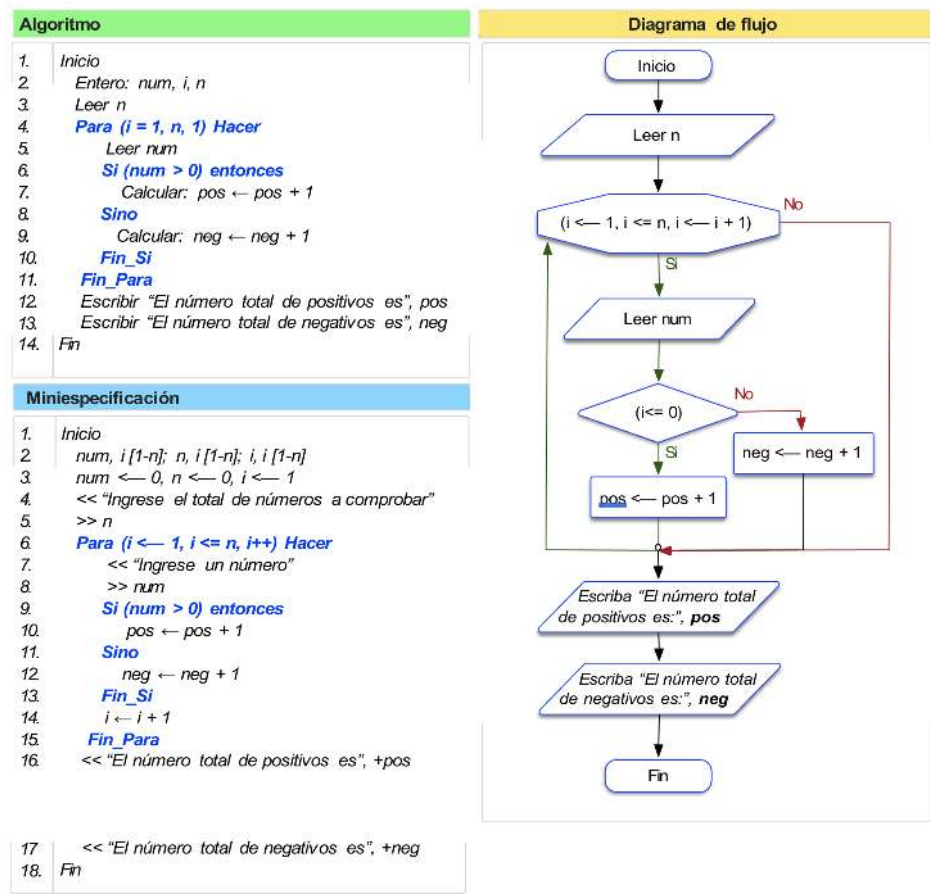
Sin embargo, en ocasiones se conoce como dato de entrada el número de veces que se desea que el ciclo se repita. En estos casos, en los que el número de iteraciones es fijo, se debe usar la estructura lógica repetitiva Para – Hacer. Esta estructura ejecuta las instrucciones dentro del ciclo, un número específico de veces y, de modo automático, se controla el número de iteraciones.

Para reforzar esta temática, le recomiendo revisar la sección 5.3 de la [unidad 5 de la guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#), ahí puede encontrar la sintaxis para la estructura lógica repetitiva Para - Hacer y su representación mediante diagramas de flujo.

¿La temática quedó clara con las lecturas realizadas? Si aún se presentan dudas, a continuación, en la figura 15 realizaremos un ejercicio utilizando la estructura lógica condicional para su mejor comprensión y ver la diferencia con las otras estructuras lógicas repetitivas.



Figura 15
Ejercicio. Estructura lógica repetitiva Para-Hacer



Nota. Cabrera, M., 2025.

5.4. Estructuras lógicas repetitivas anidadas

Se conoce como estructuras anidadas cuando una estructura está dentro de otra estructura. Se pueden anidar estructuras lógicas de decisión, así como estructuras lógicas de repetición.

Se recomienda revisar la sección 5.4 de la de [unidad 5 de la guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#), ahí puede encontrar ejercicios con el uso de estructuras lógicas anidadas. Revise a detalle cada uno de los ejercicios planteados.

A continuación, se describen los siguientes documentos que le ayudarán a ampliar sus conocimientos sobre la temática.

Para iniciar el estudio de los contenidos, es importante apoyarse en los materiales principales de la asignatura. Por este motivo, le invito a realizar la lectura de los siguientes documentos complementarios:

- Cabrera, M. y Tenesaca, G. (2018). Guía didáctica de algoritmos y resolución de problemas. Loja, Ecuador: Editorial Universidad Técnica Particular de Loja.

Lectura: [unidad 5. Estructuras lógicas repetitivas, sección 5.3 y 5.4.](#)

- Trejos Buriticá, O. (2017). Lógica de programación: (ed.). Ediciones de la U.

Lectura: [capítulo 8. Ciclos, sección 8.2.2, ciclo Para-Hacer y sección 8.4. Ciclos anidados](#) .

Además, para reforzar los contenidos, le recomiendo visualizar el siguiente video: "[Guía 4, lección 3, estructura repetitiva Para](#)", en la cual se explica la estructura repetitiva para, su estructura y aplicación en el desarrollo de algoritmos mediante el desarrollo de ejercicios.





Actividades de aprendizaje recomendadas

Una vez comprendida la temática, le recomendamos realizar las siguientes actividades para medir los conocimientos adquiridos. Recuerde que estas actividades de aprendizaje no son calificadas.

1. Revise los siguientes materiales complementarios:

- a. Realice una lectura comprensiva de la sección 5.1 y 5.2 de la [unidad 5 de la guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#), así como también de la [sección 8.2.2 y 8.4 del capítulo 8 de Trejos, O. \(2017\)](#).
- b. Revise el video presentado previamente. Tome nota de lo más importante y las dudas que se le presenten, para resolverlas con su tutor.

2. Desarrolle el siguiente ejercicio para reforzar lo aprendido:

Realice la miniespecificación y el diagrama de flujo:

- Ingrese un número entero y calcule su factorial, utilizando la estructura repetitiva Para-Hacer.
- Puede compartir sus respuestas en el EVA para una retroalimentación colaborativa entre su tutor y compañeros.

Nota. Por favor, conteste las actividades en su cuaderno de apuntes o en un documento Word.

3. Desarrolle la autoevaluación 5, que se encuentra al final de la unidad 5. La misma consiste en un cuestionario de preguntas objetivas relacionadas con las temáticas de la semana once y doce.





Autoevaluación 5

Analice las siguientes preguntas de opción múltiple y seleccione la respuesta correcta:

1. Un contador da un seguimiento al número de:
 - a. Estructuras de ciclo dentro de un programa.
 - b. Veces en que se acumula una variable.
 - c. Ciclos de máquina requeridos para un segmento de un programa.
2. Agregar el valor de 1 a una variable también se llama:
 - a. Reiniciar.
 - b. Incrementar.
 - c. Decrementar.
3. Por lo general, el valor agregado a una variable acumuladora es:
 - a. El mismo para cada iteración.
 - b. El valor de 1.
 - c. Diferente por cada iteración.
4. ¿Cuáles de las siguientes afirmaciones son verdaderas sobre las estructuras *Mientras Que* y *Hacer Hasta*?
 - a. La estructura **Mientras** se ejecuta siempre y cuando se cumpla la condición.
 - b. La estructura **Hacer-Hasta** nunca se ejecuta si la condición no se cumple.
 - c. La estructura **Mientras** se ejecuta, independiente de la condición.
 - d. La estructura **Hacer-Hasta** se ejecuta al menos una vez, así la condición no se cumpla.





5. De las siguientes ideas presentadas, ¿cuál es el concepto apropiado para describir una estructura de control anidada?

- a. Cuando se coloca una estructura de control fuera de otra.
- b. Cuando se coloca una variable dentro de otra.
- c. Cuando se coloca una estructura de control dentro de otra.

6. Con base en la siguiente miniespecificación, determine cuál es el valor resultante de i.

- 1. $i \leftarrow 0$
- 2. $\text{limite} \leftarrow 5$
- 3. Mientras $i \leq \text{limite}$ hacer
- 4. $i \leftarrow i + 1$
- 5. Fin_Mientras
- 6. $\ll i$

- a. 4.
- b. 5.
- c. 6.

7. De la siguiente lista de estructuras de control, ¿cuál es la que posee la característica que permite que el proceso se ejecute al menos una vez?

- a. Bucle de comparación al inicio.
- b. Bucle de comparación al final.
- c. Bucle de comparación ilimitado.

8. Si se ejecuta la siguiente miniespecificación y se desarrollan las pruebas de escritorio, ¿qué secuencia de números se genera?

- 1. $i \leftarrow 0$
- 2. $\text{límite} \leftarrow 24$
- 3. Mientras $i < \text{límite}$ hacer
- 4. Si $(i \% 6 == 0)$ entonces

5. << i
6. Fin Si
7. i = i + 3
8. Fin_Mientras

- a. 0, 6, 12, 18, 24.
- b. 6, 12, 18, 24.
- c. 0, 6, 12, 18.

9. Las siguientes sentencias imprimen los números del 1 al 20. ¿Qué se necesita cambiar en la estructura Mientras, para que se convierta en un ciclo infinito?

1. Inicio
2. numero1, i[1-n]
3. numero1 <- 1
4. Mientras numero1 <= 20 hacer
5. << numero1
6. numero1 <- numero1 + 1
7. Fin_Mientras
8. Fin

- a. En la línea 6, cambiar por: número1 <- número1 + 20.
- b. Eliminar la línea 6.
- c. En la línea 6, cambiar por: número1 <- número1 * 20.

10. ¿Cuál es el concepto apropiado para describir una estructura lógica, repetitiva o bucle?

- a. Permite realizar la evaluación de expresiones lógicas, para controlar la ejecución de una o más instrucciones.
- b. Permite realizar la ejecución de un conjunto de instrucciones de manera repetitiva mediante la evaluación de una o más expresiones lógicas.



c. Permite realizar la evaluación de expresiones lógicas, para controlar la ejecución de una o más instrucciones.

[Ir al solucionario](#)

Contenidos, recursos y actividades de aprendizaje recomendadas



Semana 13

Unidad 6. Estructuras de datos

Una estructura de datos, denominada en esta unidad, arreglos, es una serie o lista de valores en la memoria de la computadora. Por lo general, todos los valores en un arreglo tienen algo en común. Por ejemplo, podrían representar una lista de los números de identificación de los empleados o de los precios para los artículos vendidos en una tienda.

Un arreglo es un conjunto finito y ordenado de elementos homogéneos. La propiedad “ordenado” significa que cada uno de los elementos pueden ser identificados y homogéneos; significa que todos los elementos del arreglo son del mismo tipo de dato. Con el presente resultado de aprendizaje, aprenderá el uso de arreglos unidimensionales, denominados, también vectores, y arreglos bidimensionales, denominados matrices.

6.1. Arreglos unidimensionales

Un arreglo unidimensional, o también denominado vector, es una estructura de datos compuesta por un conjunto de elementos de la misma especie, los cuales son direccionados por un único subíndice que está organizado linealmente, con el fin de almacenar un conjunto de datos de forma secuencial.

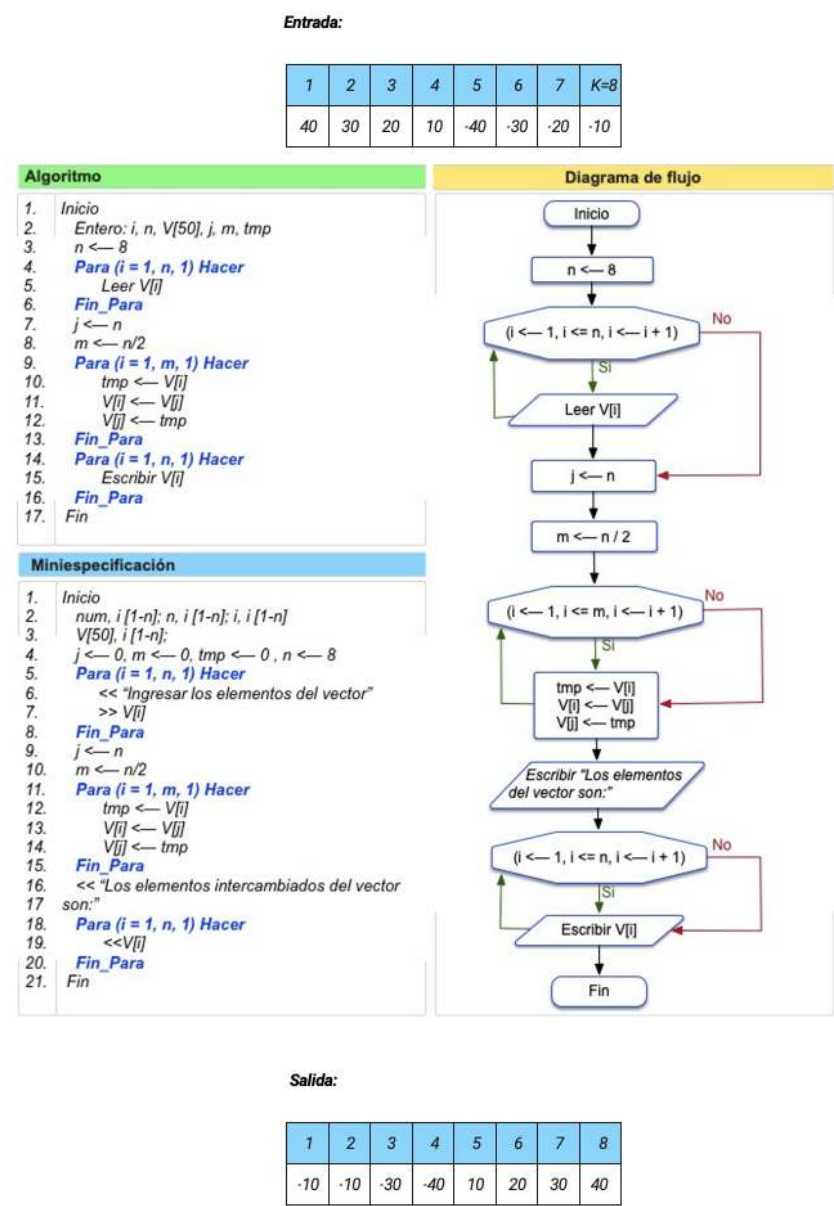
Las operaciones que se pueden realizar con vectores, durante el proceso de resolución de un problema, son: asignación, lectura/escritura, recorrido (acceso secuencial), actualizar (añadir, borrar, insertar), ordenación y búsqueda.



Para conocer los elementos y la sintaxis de un arreglo unidimensional, revise detenidamente la sección 6.1 de la [unidad 6 de la guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#). Luego, analice el siguiente ejercicio que se presenta en la figura 16. Consiste en intercambiar los elementos de un vector de ocho elementos.



Figura 16
Ejercicio. Estructura de datos unidimensionales



Nota. Cabrera, M., 2025.

¿Aún tiene dudas sobre la temática? No se preocupe, los siguientes materiales de aprendizaje le ayudarán a resolver sus inquietudes y a reforzar su aprendizaje.

Para la presente semana, lea detalladamente los siguientes documentos complementarios de la asignatura:

- Cabrera, M. y Tenesaca, G. (2018). Guía didáctica de Algoritmos y resolución de problemas. Loja, Ecuador: Editorial Universidad Técnica Particular de Loja.

Lectura: [Unidad 6. Estructuras de Datos, sección 6.1.](#)

- Trejos Buriticá, O. (2017). Lógica de programación: (ed.). Ediciones de la U.

Lectura: [Capítulo 9. Arreglos.](#)

Además, para reforzar los contenidos, le recomiendo visualizar los siguientes videos:

- [Guía 5 lección 2 Estructura de Datos \(parte 1\).](#)
- [Guía 5 lección 2 \(parte 2\) Estructura de Datos.](#)

En estos videos se explican qué son los datos, qué es la información y las estructuras de datos que pueden emplearse en el desarrollo de algoritmos. Además, se enseña a utilizar las estructuras de datos unidimensionales, o vectores, mediante la resolución de problemas prácticos.





Actividades de aprendizaje recomendadas

Una vez comprendidas las estructuras de datos unidimensionales, a continuación, se proponen algunas actividades recomendadas para medir su aprendizaje. Recuerde que estas actividades no son calificadas.

1. Revise los siguientes materiales complementarios:

- a. Realice una lectura comprensiva de la sección 6.1 de la [unidad 6 de la guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#), así como también del [capítulo 9 - Arreglos de Trejos, O. \(2017\)](#).
- b. Revise los videos presentados previamente. Tome nota de lo más importante y de las dudas que se le presenten, para resolverlas con su tutor.

2. Desarrolle el siguiente ejercicio para reforzar lo aprendido:

Realice la miniespecificación y el diagrama de flujo:

- Elabore una miniespecificación que reciba como datos de entrada un número entero positivo y los elementos de un vector de tamaño n , y que regrese como dato de salida cuántas veces se repite el último elemento del vector.

Entrada: $n = 6$ $A = [4, 2, 5, 3, 7, 5]$.

Salida: el número 5 se encuentra repetido 2 veces.

- Comparta sus valoraciones en el espacio no calificado que se habilitará en el EVA para el efecto. Recuerde que, a través de este tipo de actividad, podrá conseguir la retroalimentación tanto de su tutor como de sus compañeros.

Nota. Por favor, conteste las actividades en su cuaderno de apuntes o en un documento Word.





Unidad 6. Estructuras de datos

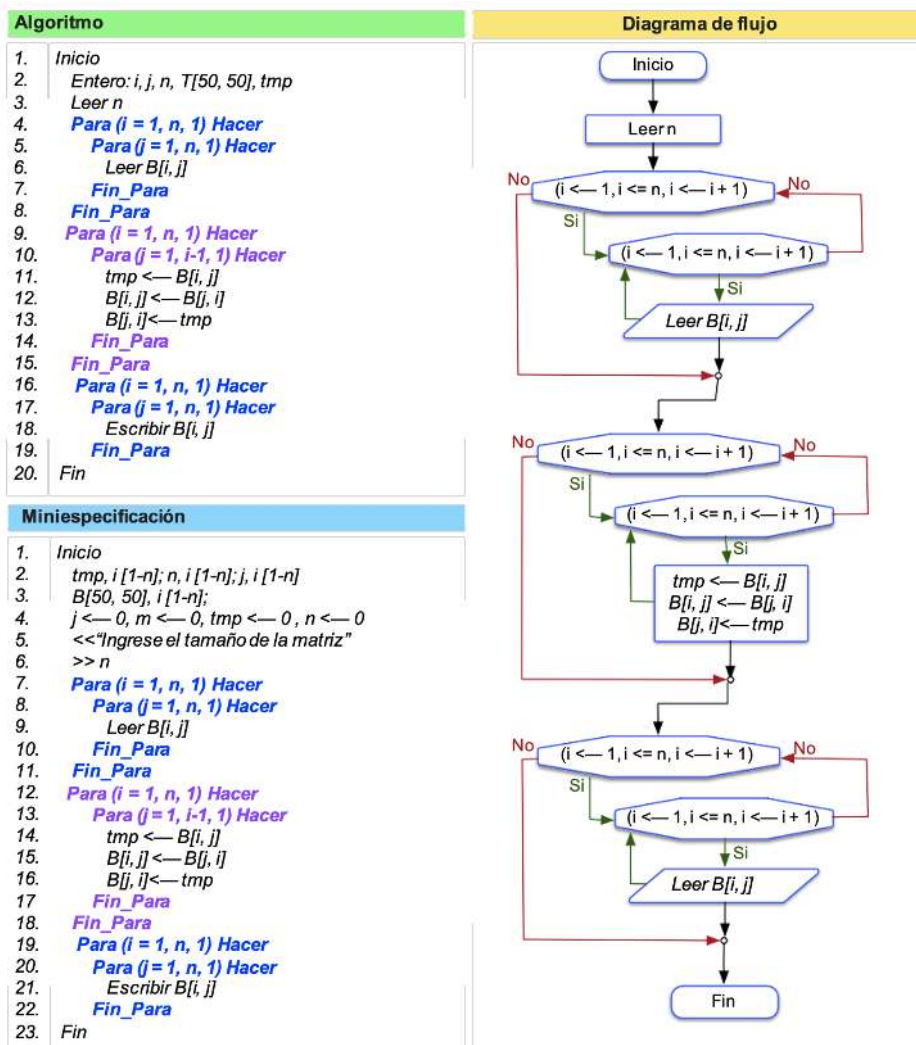
6.2. Arreglos bidimensionales

A diferencia de un arreglo unidimensional, un arreglo bidimensional, es un conjunto de elementos de la misma especie, almacenados en un conjunto de vectores, denominados filas, los cuales se intersecan con otro conjunto de vectores, denominados columnas. El espacio de intersección entre una fila y una columna, generan un espacio de almacenamiento de datos. Este espacio de almacenamiento está direccionado por dos subíndices que identifican la posición de un elemento dentro del arreglo. Los arreglos bidimensionales, también se denominan matrices.

Para conocer los elementos y la sintaxis de un arreglo unidimensional, revise detenidamente la sección 6.2 de la [unidad 6 de la guía didáctica de Cabrera, M. y Tenesaca, G. \(2018\)](#). Luego, analice el siguiente ejercicio para la generación de una matriz transpuesta que se presenta en la figura 17.



Ejercicio. Estructura de datos bidimensionales



Nota. Cabrera, M., 2025.

Ahora bien, para reforzar los contenidos de la temática lo invito a estudiar los siguientes materiales de aprendizaje. Esto le ayudará a solventar sus inquietudes y a reforzar lo aprendido.

Para la presente semana, se debe realizar una lectura comprensiva de los siguientes documentos complementarios de la asignatura:

- Cabrera, M. y Tenesaca, G. (2018). Guía didáctica de Algoritmos y resolución de problemas. Loja, Ecuador: Editorial Universidad Técnica Particular de Loja.

Lectura: [Unidad 6. Estructuras de Datos, sección 6.2.](#)

- Trejos Buriticá, O. (2017). Lógica de programación: (ed.). Ediciones de la U.

Lectura: [Capítulo 10. Matrices.](#)

Además, para reforzar los contenidos le recomiendo visualizar el siguiente video: "[Ejercicios PSeInt - Matrices #1 - Ventas de departamentos](#)", en el cual se explican las estructuras de datos bidimensionales, denominadas también matrices. Se explica su estructura y su función, en la resolución de problemas, mediante algoritmos, utilizando PSeInt.

Una vez comprendidas las estructuras de datos unidimensionales, a continuación, se proponen algunas actividades para medir su aprendizaje:



Actividades de aprendizaje recomendadas

Tenga presente que estas actividades de aprendizaje no son calificadas.

1. Revise los siguientes materiales de apoyo:
 - a. Realice una lectura comprensiva de la sección 6.2 de la [unidad 6 de la guía didáctica de Cabrera, y Tenesaca, G. \(2018\)](#), así como del [capítulo 10 - Matrices de Trejos, O. \(2017\)](#).
 - b. Revise el video presentado previamente. Tome nota de lo más importante y de las dudas que se le presenten, para resolverlas con su tutor.
2. Desarrolle el siguiente ejercicio para reforzar lo aprendido:



Realice la miniespecificación y el diagrama de flujo:

- Desarrolle una solución que permita sumar el número de elementos positivos y negativos de una matriz de $m \times n$.

Entrada:

| | | | |
|----|----|----|----|
| -1 | 2 | 3 | 4 |
| 5 | -1 | 6 | 7 |
| 8 | 9 | -1 | 1 |
| 2 | 3 | 4 | -1 |

Salida:

La suma de los números positivos es 54.

La suma de los números negativos es -4.

- Finalmente, comparta sus valoraciones en el espacio no calificado que se habilitará en el EVA para el efecto.

Recuerde que, a través de este tipo de actividad, podrá conseguir la retroalimentación tanto de su tutor como de sus compañeros.

Nota. Por favor, conteste las actividades en su cuaderno de apuntes o en un documento Word.

3. Desarrolle la autoevaluación 6, que se encuentra al final de la unidad 6. La misma consiste en un cuestionario de preguntas objetivas relacionadas con las temáticas de la semana trece y catorce.





Autoevaluación 6

Analice las siguientes preguntas de opción múltiple y seleccione la respuesta correcta:

1. Un subíndice es:
 - a. Un elemento del arreglo.
 - b. Nombre alternativo de un arreglo.
 - c. Número que indica la posición de un elemento en un arreglo.
2. Cada valor de datos en un arreglo se llama:
 - a. Tipo de dato.
 - b. Subíndice.
 - c. Elemento.
3. Es un tipo de datos estructurados, compuesto por un conjunto de elementos, de tamaño fijo y elementos homogéneos (del mismo tipo), los cuales son direccionados por un único subíndice:
 - a. Arreglo bidimensional.
 - b. Arreglo unidimensional.
 - c. Multidimensional.
4. ¿Cuál de las siguientes opciones es correcta, para presentar los elementos de un arreglo bidimensional (A)?
 - a. Para ($i \leftarrow -1, n, 1$) entonces
Para ($j \leftarrow 1, n, 1$) entonces
 $\ll A[i, j]$
Fin_Para

Fin_Para
 - b. $i \leftarrow -1$



Para ($j \leftarrow -1, n, 1$) entonces

$A[i, j] \leftarrow 0$

Fin_Para

c. Para ($i \leftarrow -1, n, 1$) entonces

$j \leftarrow -1$

$<< A[i, j]$

Fin_Para

5. ¿Cuál es el resultado de la siguiente miniespecificación?

1. Inicio
2. $A[100, 100], i[1-n]$
3. $n, i, j, i[1-n]$
4. $n \leftarrow 3$
5. Para ($i \leftarrow -1, n, 1$) entonces
6. Para ($j \leftarrow -1, n, 1$) entonces
7. $A[i, j] \leftarrow 0$
8. Fin_Para
9. Fin_Para
10. Fin

- a. Asigna a la primera fila y primera columna el valor de 0.
- b. Asigna a la última fila y última columna el valor de 0.
- c. Asigna a todos los elementos de la matriz el valor de 0.

6. Con la siguiente miniespecificación, identifique a qué elementos de la matriz se les asigna el valor de 1:

1. Inicio
2. $n, i, j, i[1-n], A[100, 100]$
3. $n \leftarrow 3$
4. Para ($i \leftarrow -1, n, 1$) entonces



5. Para ($j \leftarrow -1, n, 1$) entonces
6. Si ($i + j == (n + 1)$) entonces
7. $A[i, j] \leftarrow -1$
8. Fin_Si
9. Fin_Para
10. Fin_Para
11. Fin

- a. Los elementos de la diagonal principal.
- b. Los elementos de la triangular inferior.
- c. Los elementos de la diagonal secundaria.

7. ¿Cuál de las siguientes opciones puede estar compuesta de todos sus elementos de tipo cadena?, ¿otro puede tener sus elementos de tipo entero, etc.?

- a. Arreglo,
- b. Contador,
- c. Ciclo,

8. ¿Cuál es una de las ventajas más importantes de usar un vector?

- a. Almacenar tipos de datos.
- b. Almacenar un conjunto de variables.
- c. Almacenar un conjunto de datos.

9. Suponga que tiene un arreglo llamado Num y dos de sus elementos son Num[1] y Num[4]. Usted sabe que:

- a. Los dos elementos están en la misma ubicación de memoria.
- b. El arreglo contiene exactamente cuatro elementos.
- c. Hay exactamente dos elementos entre estos dos elementos.



10. El tipo de subíndice más útil para manejar los arreglos es un(a):

- a. Nombre de archivo.
- b. Carácter.
- c. Variable.

Ir al solucionario

Contenidos, recursos y actividades de aprendizaje recomendadas



Semana 15

Actividades finales del bimestre

En la presente semana se recomienda realizar un repaso general de las unidades 4, 5 y 6. Además, se deben realizar las actividades planificadas según las directrices dadas en cada una de ellas.

Para reforzar la comprensión de los temas vistos, le invito a participar en la siguiente sopa de letras, diseñada para afianzar los conceptos clave de programación y estructuras lógicas. A través de esta actividad lúdica, podrá repasar definiciones, identificar términos esenciales y fortalecer su capacidad para relacionar la teoría con su aplicación práctica en el desarrollo de algoritmos.

[Conceptos Clave de Programación y Estructuras Lógicas.](#)

Contenidos, recursos y actividades de aprendizaje recomendadas



Semana 16

Actividades finales del bimestre

En la presente semana, repase los contenidos de la unidad 6. Con esto, podrá estar más preparado para la evaluación presencial del primer bimestre.



Actividad: evaluación presencial.

Al final de esta semana, se encuentra planificada la evaluación presencial. Realice todas las actividades de aprendizaje recomendadas y calificadas; esto le ayudará a cumplir con éxito esta actividad.



¡Felicidades! Hemos terminado los contenidos de la asignatura, gracias por su constancia y esfuerzo en cada una de las semanas. Le invito a seguir con el mismo entusiasmo, cultivando más conocimiento en el transcurso de la carrera. ¡Éxitos!





4. Solucionario

Autoevaluación 1

| Pregunta | Respuesta | Retroalimentación |
|----------|-----------|--|
| 1 | b | Dado que la informática, como ciencia, estudia el tratamiento de la información , mediante máquinas de procesamiento electrónico de datos, al servicio de una sociedad digital y global; se basa tanto en la tecnología como en la capacidad racional humana. |
| 2 | a | El símbolo conector. Este es un símbolo especial dentro de los diagramas de flujo y tiene como función, trasladar el control del programa a otra parte, dentro del mismo diagrama. Cada conector deberá contar con un identificador único. |
| 3 | c | Porque un algoritmo es un conjunto finito de reglas bien definidas en su lógica de control, que permiten la solución de un problema en una cantidad finita de tiempo. En la resolución del problema, con las reglas mencionadas, el algoritmo realiza un conjunto de pasos, cuya ejecución para dar la solución del problema, puede ser ejecutada manualmente, mecánicamente o utilizando una máquina de procesamiento electrónico de datos. |
| 4 | a | Esta ventaja corresponde a los diagramas de flujo, debido a que un diagrama de flujo, es uno de los entregables más comunes en la fase de análisis de un programa y sirve como referencia para que sean interpretados por los programadores. Por lo tanto, la opción correcta es la a . |
| 5 | b | Debido a que una computadora es un dispositivo electrónico, utilizado para procesar información y obtener resultados, es capaz de ejecutar cálculos y tomar decisiones a velocidades millones o cientos de millones más rápidas, de lo que pueden hacerlo los seres humanos. |
| 6 | c | Esta definición corresponde a los símbolos especiales, la opción c . A la categoría de símbolos especiales, pertenecen típicamente los operadores aritméticos (+, -, *, /) y de comparación (<, >, <=, =). |
| 7 | a | Los diagramas de flujo, son entregables en las fases de análisis y diseño, en el ciclo de desarrollo de un programa, dado que facilitan la comunicación entre los desarrolladores y los clientes. Por lo tanto, la opción correcta es la a . |

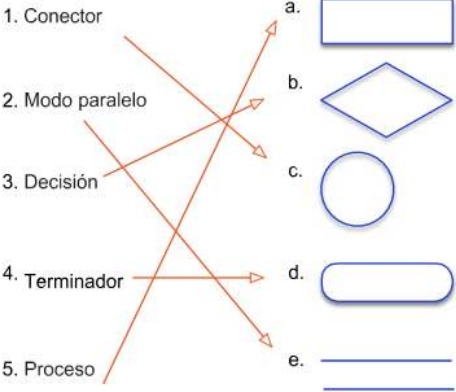


| Pregunta | Respuesta | Retroalimentación |
|----------|-----------|-------------------|
|----------|-----------|-------------------|

| | | |
|---|---|--|
| 8 | a | Dado que un paralelogramo es un símbolo de datos que representa los datos de entrada y salida. Asimismo, soporta las operaciones de petición y muestra de datos. |
|---|---|--|

| | | |
|---|---|--|
| 9 | b | |
|---|---|--|

La relación de los términos con sus símbolos quedaría:



Por consiguiente, la opción b es la correcta.

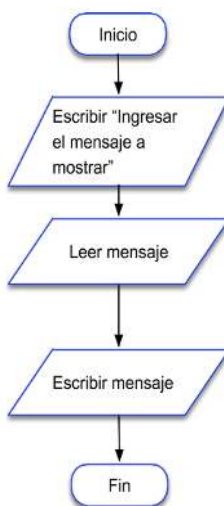
Los pasos por seguir para resolver el problema son:

1. Mostrar en pantalla, un mensaje donde se solicita que el usuario ingrese el mensaje a mostrar.
2. Se lee el mensaje ingresado por el usuario.
3. Se presenta el mensaje en pantalla.

El algoritmo para resolver el problema es:

1. Inicio.
2. Escribir "Ingresar el mensaje a mostrar en pantalla".
3. Leer el mensaje.
4. Escribir mensaje.
5. Fin.

A continuación, se presenta la solución en un diagrama de flujo, por lo tanto, la opción correcta es la **a**.



[Ir a la autoevaluación](#)

Autoevaluación 2

| Pregunta | Respuesta | Retroalimentación |
|----------|-----------|---|
| 1 | c | <p>Porque el <i>nibble</i> es conocido como “cuarteto o semiocteto”, que permite representar un número binario de cuatro dígitos. Estos números van desde el 0000 (0_{16}) al 1111 (F_{16}) y corresponden a un carácter del sistema numérico hexadecimal.</p> <p>Las opciones a y b son incorrectas.</p> |
| 2 | b | <p>Porque el tipo de dato que se puede usar es decimal, porque debe almacenar el valor de un producto y puede contener valores de 0 a 10, la opción a no es correcta, porque byte solo almacena un conjunto de ocho bits; la opción c es un tipo de variable que admite a todos.</p> |
| 3 | b | <p>Debido a que $7 \geq 7$ retorna valor True, porque haciendo razonamiento, sabemos que 7 es mayor o igual a 7; la opción a no es correcta, ya que el carácter C (mayúscula) no es igual a c (minúscula); la opción c no es correcta, ya que, desarrollando las operaciones matemáticas, 9 no es igual a 8.</p> |
| 4 | a | <p>Porque la edad de una persona es de 0 hasta 110 años; la opción b no puede ser, ya que el rango de edad es muy bajo desde 0 a 10. La opción c no es correcta, una persona no puede tener edad desde los 18 hasta los 100 años.</p> |
| 5 | c | <p>Precedencia explícita, se provoca mediante el uso de paréntesis, aquello que se encierra en un paréntesis es obligado a resolverse sin respetar otras reglas de precedencia respecto a lo que está afuera; la opción a no es correcta porque la precedencia posicional se presenta cuando tiene varias expresiones; la opción b no es correcta, porque la precedencia implícita es aquella inherente a los operadores y la categoría a la que pertenecen.</p> |
| 6 | b | <p>La expresión $(a - b < 3 - c)$ and $(c * 1 == a - b)$ es falsa, porque reemplazando los valores $a = 2$, $b = 4$, $c = 6$, genera en la primera parte $(-2 < 4 == -2)$, lo cual genera inconsistencia de valor. Por lo tanto, la opción b es la correcta.</p> |
| 7 | a | <p>Las reglas para definir correctamente el nombre de una variable son las numerales:</p> <ol style="list-style-type: none"> 1. Siempre iniciar con letra, 5) Máximo 32 caracteres de longitud, y 6) Representativo al valor que guarda. <p>Además, se recomienda que debe componerse de letras y números, no contener espacios en blanco ni caracteres generales (incluyendo letras acentuadas y signos regionales) a excepción del guion bajo (_).</p> |



| Pregunta | Respuesta | Retroalimentación |
|----------|-----------|---|
| 8 | c | <p>Porque la clave formada por una letra "Z" y 5 números con regla de ningún número puede ser 0. Es $X(6) [1 \{Z\}, 5 \{1-9\}]$, debido a que solicita una sola letra Z, 5 números no pueden ser 0. Entonces el rango es de 1 a 9.</p> <p>La opción a, no es correcta porque en el rango de números incluye el valor de 0 y la opción b no es correcta, porque el rango de letras incluye más de una letra. Todo el alfabeto de A-Z solo necesita una letra: Z",</p> |
| 9 | c | <p>No describe el tipo de dato de una variable, la opción c, porque indica un contexto ambiguo sin relación al tipo de dato.</p> |
| 10 | b | <p>No es una declaración válida la opción b, porque la variable <i>precioProducto</i> es una variable numérica y se le está asignando el valor "24.95", que es una cadena.</p> <p>Es una declaración válida porque $\text{costoProducto} = 100$ es correcto, y la opción c, también es válida porque a la variable <i>costoProducto</i> le asigna $\text{precioProducto} - 10$, que son del mismo tipo numérico.</p> |

[Ir a la autoevaluación](#)



Autoevaluación 3

| Pregunta | Respuesta | Retroalimentación |
|----------|-----------|---|
| 1 | c | Las fases del desarrollo de un programa permiten organizar el trabajo de forma secuencial y lógica. El análisis asegura comprender el problema; el diseño traduce esa comprensión en una solución estructurada; la codificación convierte el diseño en un lenguaje comprensible para la computadora; las pruebas garantizan que el programa funcione como se espera, y finalmente la implementación pone en marcha la solución. Seguir este orden evita errores y facilita el mantenimiento del <i>software</i> . |
| 2 | c | La fase de análisis consiste en descomponer el todo en sus partes, con el fin de comprender y dar solución a problemas de la vida real. El objetivo es analizar detalladamente el problema planteado, para determinar las posibles soluciones y levantar los requerimientos necesarios para resolver el mismo. Por lo tanto, la opción c es la correcta. |
| 3 | a | Debido a que, en el ciclo de desarrollo de un programa, el cliente se involucra en la fase de pruebas e implementación. En esta fase, el cliente puede medir si el programa cumple con todos los requerimientos solicitados desde el inicio en la fase de análisis. De no ser así, se deben revisar nuevamente las fases previas. |
| 4 | a | Porque en la fase de diseño, se da respuesta a esta pregunta y el objetivo de esta fase es traducir técnicamente los requerimientos del cliente, obtenidos en la fase de análisis a una forma abstracta y estandarizada. |
| 5 | a | Un algoritmo es un conjunto ordenado y finito de asignaciones, procesos, cálculos y decisiones que permiten a un programa satisfacer una unidad de funcionalidad dada. Un algoritmo no puede estar sin orden, ya que para la resolución de problemas se necesita seguir una secuencia lógica de pasos. Y no puede ser infinito, dado que nunca se llegaría a resolver el problema planteado. Por lo tanto, la opción a es la correcta. |
| 6 | c | Dado que la aceptación del programa se realiza en la fase de pruebas e implementación. Es en esta fase donde el cliente valida si el programa cumple con los requerimientos solicitados; de ser así, acepta el programa, caso contrario, se debe evaluar en las fases anteriores porque no se cumplen todos los requerimientos dados. |



| Pregunta | Respuesta | Retroalimentación |
|----------|-----------|--|
| 7 | b | <p>Para resolver el problema planteado, se deben identificar los datos de entrada, los del proceso y los de salida.</p> <p>Los datos de entrada son largos, anchos y costo por metro cuadrado.</p> <p>Los procesos a realizarse son calcular el área del terreno, calcular el valor del terreno y calcular el descuento.</p> <p>La salida a obtener: costo total del terreno.</p> <p>Como se puede observar en los datos de entrada, en el algoritmo dado, el dato faltante para resolver el problema es el <i>costo por metro cuadrado</i>.</p> |
| 8 | a | <p>Con base en el problema planteado, la opción a es correcta, debido a que cumple con las especificaciones dadas al inicio por el cliente.</p> <p>La opción b no cumple con el rango dado. Y la opción c no cumple, debido a que la salida presentada no es la solicitada por el cliente.</p> |
| 9 | b | <p>La miniespecificación es un paso intermedio entre el algoritmo y la codificación. Su propósito es ofrecer una visión clara y estructurada de las operaciones, manteniendo la comprensión para los humanos. Esto permite verificar la lógica antes de traducirla al lenguaje de programación.</p> |
| 10 | b | <p>Las pruebas de escritorio son un recurso manual que facilita seguir paso a paso cómo cambian las variables en el transcurso de la ejecución de un algoritmo. Gracias a ellas se pueden detectar errores lógicos y comprobar que los resultados obtenidos corresponden a lo esperado, antes de implementar el programa en un lenguaje formal.</p> |

[Ir a la autoevaluación](#)



Autoevaluación 4

| Pregunta | Respuesta | Retroalimentación |
|----------|-----------|---|
| 1 | b | Condicional compuesta, porque solo ejecuta las primitivas de control cuando es verdadera y cuando la condición es falsa; la opción a no es correcta, porque solo ejecuta la condición cuando es verdadera; la opción c , es un ciclo repetitivo. |
| 2 | c | Ya que <i>Dependiendo_De</i> es una estructura condicional compuesta que permite la ejecución de un conjunto de primitivas de control(pi), en la que el cumplimiento de la condición de la estructura en mención hace posible la ejecución de las instrucciones entre muchas alternativas, no solo por el cumplimiento verdadero de la condición, punto por el cual se ejecutan las primitivas p1...pn, sino el cumplimiento de la condición cuando es falsa; la opción a no es correcta, porque condicional simple permite ejecuciones de las primitivas p1...pn, si la condición es verdadera o se cumple; la opción b , no es correcta, porque solo ejecuta las primitivas de control cuando es verdadera y cuando la condición es falsa. |
| 3 | b | Alternativa 2, porque el problema requiere solucionar el incremento del salario de un empleado en un 25%, siempre y cuando gane 750 dólares o más. Por ello, se aplica una condición simple en la que controla si el salario es mayor o igual a 750, entonces a la variable salario se le asigna el valor de salario + (salario *0.25). La opción a no es correcta, porque en la condición simple indica que, si el salario es mayor o igual a 750, entonces, a la variable salario le asigna el valor de (salario *0.25). Sin embargo, no almacena el valor de salario; la opción c , no es correcta porque si la condición simple de salario es mayor a 750, entonces, a la variable salario le almacena salario +(salario *0.25) y no la almacenaría cuando la variable salario sea igual a 750. |
| 4 | c | Porque en la condición simple, si la edad es mayor o igual a 18, presenta la edad; caso contrario, presentar incorrecto. La opción a no es correcta, porque la condición indica que, si la edad es mayor a 18 años, presentar la edad; caso contrario, ingresar "incorrecto", lo que aquí no realizaría la solución adecuada, ya que, solo evalúa edad mayor a 18, o sea, puede ser desde 18.1 y no desde igual a 18 y es cuando también es mayor de edad. La opción b , no es correcta, porque la condición indica que, si la edad es mayor o igual a 18 años, presentar la edad, caso contrario, ingresar "incorrecto", lo que aquí no realizaría la solución adecuada porque lo que debe hacer es presentar la palabra incorrecto y no la palabra ingresar. Tomar en cuenta los operadores >>, <<. |



| Pregunta | Respuesta | Retroalimentación |
|----------|-----------|--|
| 5 | b | <p>Porque se solicita resolver el siguiente problema: si un usuario que adquiere tres productos, su costo final es mayor a 200 dólares, se realiza un descuento de 15 %, el algoritmo debe mostrar el total, descuento y el algoritmo, los que hace es primero ingresar los tres productos y los almacena en tres variables diferentes, luego a variable total, le asigna la suma de los tres productos ($\text{total} = \text{prod1} + \text{prod2} + \text{prod3}$), con ello, con la condición simple Si controla ($\text{total es mayor a 200}$), entonces a la variable descuento le asigna $\text{total} * 15\%$ de descuento, finalizado el control a la variable total, asigna la suma de total y descuento $\text{total} = (\text{Total} + \text{desc})$, finalmente presenta los valores de total y descuento.</p> <p>Las opciones a y c no son correctas.</p> |
| 6 | V, F | <p>En programación, la condición es un elemento fundamental porque define el criterio que controla el flujo de un algoritmo, ya sea en una estructura condicional o repetitiva. La decisión, en cambio, corresponde a la acción que se ejecuta cuando la condición se cumple o no, pero no siempre está ligada a un ciclo repetitivo; puede presentarse en estructuras simples de control de flujo. Por eso, distinguir ambos conceptos es clave para construir algoritmos correctos y eficientes.</p> |
| 7 | c | <p>Porque la condición controla el área, según su expresión, presenta el área y sale del ciclo de control. Por tanto, es una condicional simple: la opción a no es correcta; la opción b, no es correcta.</p> |
| 8 | b | <p>Porque se desea resolver un programa que permite obtener y presentar el cubo de un número, siempre y cuando, el número sea menor a 60, En caso contrario, debería obtener y presentar el cuadrado del número indicado. Por ello, la condición simple con la variable número, controla si número es menor a 60, entonces a la variable cuadrado se le asigna número elevado a 3 ($\text{cuadrado} \leftarrow \text{número}^3$), luego, presentar el valor de cuadrado. Si no se cumple la condición a la variable, cubo le asigna número elevado a 2 ($\text{cubo} \leftarrow \text{número}^2$) y presentar el valor de cubo y finalizar; las opciones a y c no son correctas.</p> |
| 9 | c | <p>Porque se desea dar solución a un programa que permita determinar si alguien tiene la edad para entrar a la universidad en un país (el país tiene como ley que los estudiantes pueden ingresar a la universidad solo si tienen 20 años). Si la persona cumple con la edad, presentar un mensaje de "ingreso exitoso". En caso de que no tenga la edad, presentar un mensaje de "edad incorrecta", por ello, la solución correcta es con la condicional simple en la variable edad que controla si la edad es igual a 20 (Si $\text{edad} == 20$, entonces), presentar el texto "ingreso exitoso". Si no se cumple, entonces presentar "edad incorrecta" y finalizar el ciclo de control. La opción a y b es incorrecta.</p> |



| Pregunta | Respuesta | Retroalimentación |
|----------|-----------|-------------------|
|----------|-----------|-------------------|

| | | |
|----|---|---|
| 10 | b | <p>Condicional DD (Dependiendo_De) compuesta, que permite la ejecución de un conjunto de primitivas de control(pi), en la que el cumplimiento de la condición de la estructura en mención, hace posible la ejecución de las instrucciones, entre muchas alternativas, no solo por el cumplimiento verdadero de la condición, punto por el cual se ejecutan las primitivas p1..pn, sino el cumplimiento de la condición cuando es falsa.</p> <p>La opción a y c es incorrecta.</p> |
|----|---|---|

[Ir a la autoevaluación](#)



Autoevaluación 5

| Pregunta | Respuesta | Retroalimentación |
|----------|-----------|--|
| 1 | c | Dado que un contador es una variable cuyo valor se incrementa o decrementa en una cantidad constante, con la finalidad de contar sucesos o acciones internas de un ciclo repetitivo. Por lo general, un contador se inicializa desde 0 o 1 y va incrementando de uno en uno. |
| 2 | b | Se llama incrementar al proceso de agregar el valor de 1 a una variable denominada contador. Generalmente, se utiliza para trabajar con estructuras lógicas repetitivas. Por lo tanto, la opción b es correcta. |
| 3 | c | Debido a que un acumulador es una variable que suma, sobre sí misma, un conjunto de valores, para, de esta forma, almacenar la suma de todos ellos en una sola variable. Los valores por almacenar, varían con base en los procesos y operaciones. |
| 4 | a, d | La diferencia entre la estructura <i>Mientras Que-Hacer</i> y la estructura <i>Hacer-Hasta</i> es que la primera se ejecuta únicamente si se cumple la condición, mientras que la segunda se ejecuta al menos una vez, aunque la condición no se cumpla. |
| 5 | c | Dado que se conoce como estructura anidada, una estructura que está dentro de otra estructura. Se pueden anidar estructuras lógicas de decisión, así como estructuras lógicas de repetición. |



Para obtener el valor de la variable i de la miniespecificación, se debe realizar una prueba de escritorio.

- a. $i \leftarrow 0$
- b. $\text{limite} \leftarrow 5$
- c. Mientras ($i \leq \text{limite}$) hacer
- d. $i \leftarrow i + 1$
- e. Fin_Mientras
- f. $\ll i$



| Pregunta | Respuesta | Retroalimentación |
|----------|-----------|-------------------|
|----------|-----------|-------------------|

| Valores | Variables | |
|---------|-----------|---|
| | Límite | i |
| | 5 | 1 |
| | | 2 |
| | | 3 |
| | | 4 |
| | | 5 |
| | | 6 |

La salida de $i = 6$. Por lo tanto, la opción c, es la correcta.

| | | |
|---|---|--|
| 7 | b | El bucle de comparación al final es la estructura lógica que permite que el proceso se ejecute al menos una vez. La estructura <i>Hacer-Hasta</i> , es un bucle de comparación al final. Por lo tanto, la opción correcta es la b . |
|---|---|--|



Para obtener la serie de la miniespecificación, se realiza la prueba de escritorio:

- a. $i \leftarrow 0$
- b. limite 24
- c. Mientras ($i < 24$) hacer
- d. Si ($i \% 6 == 0$) entonces
- e. $\ll i$
- f. Fin Si
- g. $i \leftarrow i + 3$
- h. Fin mientras

Prueba de escritorio



| Pregunta | Respuesta | Retroalimentación | |
|----------|-----------|--|--------|
| | | Variables | Salida |
| | | Límite | i |
| | | 24 | 0 |
| | | | 3 |
| | | | 6 |
| | Valores | | 9 |
| | | | 12 |
| | | | 15 |
| | | | 18 |
| | | | 21 |
| | | | 24 |
| | | La serie resultante es: 0, 6, 12, 18. Por lo tanto, la opción c, es la correcta. | |



Para que la estructura *Mientras* de la siguiente miniespecificación, se convierta en un bucle infinito, se debe eliminar la línea 6. Para verificar esto, se realiza la prueba de escritorio.

- a. Inicio
- b. $\text{numero1}, i[1-n]$
- c. $\text{numero1} \leftarrow 1$
- d. Mientras $\text{numero1} \leq 20$ hacer
- e. $\ll \text{numero1}$
- f. Fin_Mientras
- g. Fin

Prueba de escritorio



| Pregunta | Respuesta | Retroalimentación | |
|----------|-----------|-------------------|--|
|----------|-----------|-------------------|--|

| Valores | Variables | | Condición | Salida |
|---------|-----------|---------|-------------|-----------|
| | Límite | numero1 | numero1<=20 | <<numero1 |
| | 20 | 1 | 1<=20 → Si | 1 |
| | | | 1<=20 → Si | 1 |
| | | | 1<=20 → Si | 1 |
| | | | 1<=20 → Si | 1 |
| | | | 1<=20 → Si | 1 |
| | | | ... | |
| | | | 1<=20 → Si | 1 |
| | | | ... | |

Como se puede observar en la prueba de escritorio, el valor de la variable **numero1** nunca incrementa, por lo que siempre se va a cumplir la condición de que **numero1** va a ser menor o igual que el **límite(20)**. Es por ello, que la opción correcta es la **b**.

| | | |
|----|---|---|
| 10 | b | Las estructuras repetitivas permiten automatizar procesos al ejecutar un conjunto de instrucciones varias veces, siempre que una condición lógica lo determine. Son útiles cuando se desconoce cuántas repeticiones serán necesarias o cuando el proceso debe realizarse hasta que se cumpla cierto criterio. Diferenciarlas de las estructuras condicionales es esencial, ya que estas últimas ejecutan instrucciones una sola vez en función de una condición, mientras que los bucles mantienen la iteración hasta que se alcance el resultado esperado. |
|----|---|---|

[Ir a la autoevaluación](#)

Autoevaluación 6

| Pregunta | Respuesta | Retroalimentación |
|----------|-----------|---|
| 1 | c | <p>Un subíndice es el número que indica la posición de un elemento dentro de un arreglo.</p> <p>En los arreglos unidimensionales, se utiliza un solo subíndice → $A[i]$, donde i es el subíndice. En los bidimensionales, se utilizan dos $A[i, j]$, donde i y j son los subíndices.</p> |
| 2 | c | <p>En un arreglo, cada posición almacena un valor individual conocido como elemento. Estos se identifican mediante un subíndice o índice, que indica su ubicación dentro de la estructura.</p> |
| 3 | b | <p>La opción b, es la correcta, debido a que un arreglo unidimensional, es un tipo de estructura de datos compuesta por un conjunto de elementos, de tamaño fijo y homogéneos (del mismo tipo), los cuales son direccionados por un único subíndice. Caso contrario, de los arreglos bidimensionales y multidimensionales, que están dirigidos por dos o más índices.</p> |
| 4 | a | <p>La opción correcta para presentar los elementos de una matriz es la opción a.</p> <p>Para $(i \leftarrow 1, n, 1)$ entonces</p> <p> Para $(j \leftarrow 1, n, 1)$ entonces</p> <p> $\ll A[i, j]$</p> <p> Fin_Para</p> <p>Fin_Para</p> <p>La opción b, es incorrecta, debido a que, únicamente, recorre y presenta los elementos de la primera fila. La opción c, únicamente recorre y presenta los elementos de la primera columna.</p> |



Para identificar la opción de respuesta correcta, se debe realizar la prueba de escritorio de la miniespecificación.

1. Inicio
2. $A[100, 100], i[1-n]$
3. $n, i, j, i[1-n]$
4. $n \leftarrow 3$
5. Para ($i \leftarrow 1, n, 1$) entonces
6. Para ($j \leftarrow 1, n, 1$) entonces
7. $A[i, j] \leftarrow 0$
8. Fin_Para
9. Fin_Para
10. Fin

Prueba de escritorio



| Valores | Variables | | | |
|---------|-----------|---|---|----------|
| | n | I | J | A [i, j] |
| | 3 | 1 | 1 | 0 |
| | | | 2 | 0 |
| | | | 3 | 0 |
| | | 2 | 1 | 0 |
| | | | 2 | 0 |
| | | | 3 | 0 |
| | | 3 | 1 | 0 |
| | | | 2 | 0 |
| | | | 3 | |

Como se puede observar, la miniespecificación planteada, asigna el valor de cero a todos los elementos de la matriz A. Por lo tanto, la opción **c**, es la correcta.



| Pregunta | Respuesta | Retroalimentación | | |
|----------|-----------|-------------------|---|---|
| | | 0 | 0 | 0 |
| | | 0 | 0 | 0 |
| | | 0 | 0 | 0 |



Para identificar a qué elementos de la matriz A se está asignando el valor de 1, se debe realizar la prueba de escritorio de la siguiente miniespecificación.

1. *Inicio*
2. $n, i[1-n]$
3. $n \leftarrow 3$
4. *Para* ($i \leftarrow 1, n, 1$) *entonces*
5. *Para* ($j \leftarrow 1, n, 1$) *entonces*
6. *Si* ($i + j \neq (n + 1)$) *entonces*
7. $A[i, j] \leftarrow 1$
8. *Fin_Si*
9. *Fin_Para*
10. *Fin_Para*
11. *Fin*

Prueba de escritorio



| | Variables | | | | | |
|---------|-----------|---|---|---------|---------|----------|
| | N | i | j | (i + j) | (n + 1) | A [i, j] |
| Valores | 3 | 1 | 1 | 2 | 4 | -- |
| | | | 2 | 3 | 4 | -- |
| | | | 3 | 4 | 4 | 1 |
| | | 2 | 1 | 3 | 4 | -- |
| | | | 2 | 4 | 4 | 1 |
| | | | 3 | 5 | 4 | -- |
| | | 3 | 1 | 4 | 4 | 1 |
| | | | 2 | 5 | 4 | -- |
| | | | 3 | 6 | 4 | -- |

Una vez terminada la prueba de escritorio, observamos que se ha asignado el valor de 1, a los elementos que conforman la diagonal secundaria.



| Pregunta | Respuesta | Retroalimentación |
|-----------------------------------|-----------|--|
| | | -- -- 1 |
| | | -- 1 -- |
| | | 1 -- -- |
| 7 | a | Ya que, un arreglo es una estructura de datos que permite almacenar un conjunto de datos del mismo tipo. Para mayor retroalimentación, revise la unidad 6, sección 6.1 de la guía didáctica. |
| 8 | c | Una de las principales ventajas de las estructuras de datos, a diferencia de las variables, es poder almacenar un conjunto de datos del mismo tipo, bajo una misma estructura y posteriormente, poder acceder a ellos, sin ninguna restricción. En cambio, una variable permite almacenar un valor único, perdiendo y sobrescribiendo su valor a medida que avanza el programa. |
| 9 | c | Como indica la pregunta, hay dos elementos Num[1] y Num[4]; con esto, no quiere decir que haya exactamente cuatro elementos en el arreglo y los dos están ubicados en dos posiciones de memoria diferentes, 1 y 4. Pero sí es seguro determinar que, entre estos dos elementos, existen exactamente dos elementos más. Por lo tanto, la opción c , es la correcta. |
| 10 | c | El tipo más adecuado para manejar un subíndice es una variable entera; por lo general, esta variable es denominada i, j, o, k. Los subíndices permiten identificar la posición de cada elemento dentro de un arreglo. Aunque pueden representarse de distintas formas, el uso de variables resulta más práctico porque permite recorrer dinámicamente el arreglo en operaciones como búsquedas, recorridos o cálculos. |
| <div>Ir a la autoevaluación</div> | | |





5. Referencias bibliográficas

- Cabrera, M. y Tenesaca, G. (2018). [Guía didáctica de Algoritmos Resolución de Problemas](#) Loja. Universidad Técnica Particular de Loja.
- Trejos Buriticá, O. I. (2017). [Lógica de programación](#): (ed.). Ediciones de la U.
- Ramírez Marín, J. H. (2019). [Fundamentos iniciales de lógica de programación I. Algoritmos en PSeInt y Python](#): (1 ed.). D - Institución Universitaria de Envigado.
- Arteaga Martínez, M. M. (2023). [Lógica de programación con PSeInt: enfoque práctico](#). (1 ed.). Corporación Universitaria Remington.
- Ramírez, F. (2007). Introducción a la programación: algoritmos y su implementación en VB. Net, C#, Java y C++. México, D. F. Alfaomega Grupo Editor SA.
- Zapata, L. (2012). Desarrollo del pensamiento analítico y sistemático: Guía práctica para aprender a programar por competencias. Medellín, Politécnico Colombiano.
- Magic Marckers (2015). ¿Qué es un algoritmo? Recuperado de: <https://goo.gl/mHnvFA>
- KhanAcademyEspanol (2016). ¿Qué es un algoritmo y por qué debería importarte? Recuperado de: <https://goo.gl/M8buBY>
- Código Compilado (2014). Diagramas de flujo. Recuperado de: <https://goo.gl/M8buBY>



Código Compilado (2014). Jerarquía de operadores Recuperado de: <https://goo.gl/Cw2jhx>

Código Compilado (2014). Operadores aritméticos, lógicos y relacionales. Recuperado de: <https://www.youtube.com/watch?v=au3VqQsKfyw>

Código Compilado (2014). Variables y constantes. Recuperado de: <https://goo.gl/zyEDV3>

VIDEO 7: Geekipedia (2016). Análisis y resolución de problemas Recuperado de: <https://goo.gl/pfQLy5>

Platzi (2014). Algoritmos de programación. Recuperado de: <https://goo.gl/N6Zpyz>

Jaramillo R. (2012). Miniespecificaciones. Recuperado de: <https://goo.gl/yvpTGq>

Videoconferencia UTPL (2012). Pruebas de Escritorio. Recuperado de: <https://goo.gl/xTaCMR>

Código compilado (2014). Decisión y algoritmo. Recuperado de: <https://goo.gl/6RRZQF>

Los programadores (2014). Condición Múltiple. Recuperado de: <https://goo.gl/2Brcqa>

Entrenamiento a la medida (2014). Tipos de estructuras Recuperado de: <https://goo.gl/9b9Hor>

Entrenamiento a la medida (2014). Estructura repetitiva MIENTRAS QUE (Parte 1). Recuperado de: <https://goo.gl/VnXkeX>

Entrenamiento a la medida (2014). Estructura repetitiva MIENTRAS QUE (Parte 2). Recuperado de: <https://goo.gl/kvQzPL>



Entrenamiento a la medida (2014). Estructura repetitiva Haga – Hasta.
Recuperado de: <https://goo.gl/uvFEuf>

Entrenamiento a la medida (2014). Estructura repetitiva PARA.
Recuperado de: <https://goo.gl/d6WoD3>

Entrenamiento a la medida (2014). Estructuras de datos (Parte 1).
Recuperado de: <https://goo.gl/nfJMWQ>

Entrenamiento a la medida (2014). Estructuras de datos (Parte 2).
Recuperado de: <https://goo.gl/nfJMWQ>

Disco Duro (2016). Matrices - PseInt. Recuperado de: <https://goo.gl/UDHFRg>

