TECNOLOGÍA SUPERIOR
EN DESARROLLO DE SOFTWARE

LÓGICA DE PROGRAMACIÓN

Carlos Salazar Guaña

Instituto Superior Tecnológico CEMLAD

TECNOLOGÍA SUPERIOR
EN DESARROLLO DE SOFTWARE

LÓGICA DE PROGRAMACIÓN

Carlos Eduardo Salazar Guaña

tecnológico cemlad

Quito-Ecuador

2022

LÓGICA DE PROGRAMACIÓN
Primera edición, febrero 2022

Autor: Carlos Eduardo Salazar Guaña
Director de la Carrera de Tecnología Superior en Desarrollo de Software
e-mail: carlos.salazar@cemlad.edu.ec

Edición y diseño gráfico: IBCB

Dirección: Alberto Enríquez S8-225 y José Mendoza
Sector Dos Puentes
Quito - Ecuador
e-mail.: info@cemlad.edu.ec

Nº Registro Institucional: PUB.2019.1.118
ISBN: 978-9942-7006-2-9

Contenido

LISTA DE FIGURAS

Dedico esta obra a los estudiantes de la carrera de
Desarrollo de Desarrollo de Software del IST CEMLAD.
Confío les sirva de ayuda para potencializar
el pensamiento lógico requerido para
construir grandes cosas.

"Primero resuelve el problema, luego escribes el código".

John Johnson

Agradecimiento

A todos los estudiantes que, durante su formación, con dudas, inquietudes y sus ganas de aprender, inspiraron la creación de este libro.

También me gustaría agradecer a mis mentoras Lina Zapata y Glenda Toala quienes lograron avivar en mí la pasión por la tecnología y el software en general.

Igualmente quiero agradecer a mi esposa e hijo por su paciencia y apoyo incondicional, a mis padres por el esfuerzo que realizaron por mi educación y a mis queridos suegros por su voz de aliento y ánimo para culminar este libro.

Introducción

Este trabajo está dirigido a todas las personas que empiezan el fantástico mundo del desarrollo de software. De esta manera, se ha tratado de explicar los diferentes conceptos de manera didáctica para que sea fácil de leer e interpretar.

Es necesario mencionar que se ha diseñado el libro para que el lector lo revise desde el inicio, e ir analizando los ejercicios resueltos y resolver los ejercicios propuestos, ya que cada lección utiliza los conceptos de las lecciones anteriores. Sin embargo, si el lector ya conoce algunos de los tópicos tratados, puede estudiar un tema específico sin la necesidad de seguir el desarrollo completo del texto.

Para desarrollar la lógica de programación, más allá de la creación de algoritmos en una hoja de papel, a lo largo de este libro se utilizan dos simuladores: DFD para representar los diagramas de flujo y PseInt para representar el pseudocódigo. Cabe mencionar que el software utilizado se encuentra disponible en Internet para su descarga.

El libro se encuentra divido en tres capítulos:

El capítulo I estudia los conceptos y definiciones generales para empezar a desarrollar la lógica de programación. Se busca introducir al estudiante en conceptos simples pero muy necesarios a la hora de buscar soluciones lógicas a un problema.

El capítulo II aborda la técnica y los diferentes conceptos a la hora de resolver algoritmos. Además, se utilizan diagramas de flujo y pseudocódigo para su representación.

Finalmente, el capítulo III explica el uso de vectores y matrices, los cuales serán de gran ayuda a la hora de resolver algoritmos mucho más complejos.

Se recomienda leer y estudiar a profundidad todos los temas tratados en este libro, los cuales servirán de base para aplicarlos con cualquier lenguaje de programación y así disminuir considerablemente la curva de aprendizaje de lenguajes como, por ejemplo, Java, C#, PHP, Python, etc.

1. Capítulo I

1.2. Lógica

Para empezar a estudiar la lógica de programación, es necesario responder la siguiente pregunta: ¿Qué es la lógica? Para ello, he buscado esta información en Internet que es el medio por el cual se puede acceder a la información en poco tiempo, y he encontrado la siguiente definición de Wikipedia (2019): "La lógica es la ciencia formal y rama tanto de la filosofía como de las matemáticas que estudia los principios de la demostración y la inferencia válida, las falacias, las paradojas y la noción de verdad". No contento con dicha definición, he decidido buscar en otra fuente, el Diccionario de la Lengua Española (2019): "Modo de pensar y de actuar sensato, de sentido común".

Ambas definiciones ayudan a tener indicios sobre el significado de la palabra lógica. No obstante, he indagado un poco más y me he encontrado con una definición bastante explícita (Buriticá, 1999, p 12): "Es la forma más OBVIA y más FÁCIL de hacer algo". Después de una búsqueda exhaustiva en Internet y en libros, pude encontrar esta definición lo suficientemente clara y fácil de entender para cualquier persona que se está iniciando en el fabuloso mundo de la programación. Recomiendo tener siempre en mente esta última definición.

1.3. Programación

La programación es un concepto mucho más simple de entender (Buriticá, 1999, p 10): "La programación involucra el conocimiento de técnicas e instrucciones de un determinado lenguaje a través de los cuales se nos hace sencillo lograr que el computador obtenga unos resultados mucho más rápido que nosotros". Actualmente a la programación se la relaciona con la creación de aplicaciones o programas informáticos que permitan resolver un problema en particular, valiéndose de lenguajes de programación como, por ejemplo, Java, C#, PHP, etc., a través de un Integrated Development Environment (IDE, en español: Entorno de Desarrollo Integrado) como, por ejemplo, Netbeans, Eclipse, Visual Studio, etc.

1.4. Lógica de programación

Entendiendo por separado la definición de lógica y programación, resulta sencillo asimilar la siguiente definición (Buriticá, 1999, p 10): "La lógica de programación

involucra de una manera técnica y organizada, los conceptos que nos permiten diseñar en términos generales la solución a problemas que pueden llegar a ser implementados a través de un computador". Por lo tanto, la lógica de programación es el paso previo a la construcción de un programa informático; únicamente busca una solución en términos generales para resolver un problema, a diferencia de la programación, la cual busca la construcción de un programa informático partiendo de la lógica de programación.

1.5. Estudio de la lógica de programación

Antes de empezar a profundizar en el estudio de la lógica de programación, me gustaría mencionar lo que no se requiere.

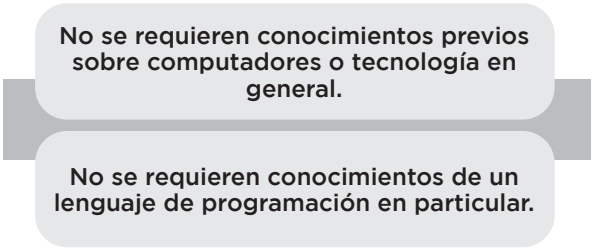


Figura 1. Qué no se requiere para aprender lógica de programación

Elaborado por Carlos Salazar

Como se puede observar en la Figura 1, para empezar a estudiar lógica de programación no hacen falta de conocimientos previos sobre tecnología, lenguajes de programación ni mucho menos de base de datos. Sin embargo, sí se requiere de tiempo para estudiar, practicar y muchas, pero muchas ganas de aprender.

1.6. Metodología para resolver problemas

Por lo general y casi siempre, un programa informático busca dar solución a un problema en particular, por lo tanto, los programadores o desarrolladores de software antes de empezar a programar debieron haber encontrado la solución lógica a dicho problema.

Para continuar con el estudio de este libro, es importante que, a partir de este instante, al problema se lo conozca como objetivo, por lo tanto, el primer paso que se debe realizar a la hora de resolver un problema es identificar el objetivo: esto permitirá visualizar hacia dónde vamos y hasta dónde se debe llegar.

Cuando se busca resolver un problema, siempre se debe tener muy claro el objetivo y nunca perderlo de vista. Pueden ocurrir mil percances o adversidades que nos alejen de dicho objetivo; lo importante como desarrolladores de software es nunca perder de vista el objetivo por alcanzar.

El siguiente paso a la hora de resolver problemas (Buriticá, 1999, p 18) es: "Cuando el objetivo está suficientemente claro podemos vislumbrar un camino lógico para llegar hasta él. Ese camino lógico va a tener un nombre dada la orientación de este libro y ese nombre es algoritmo".

Para finalizar, todo algoritmo debe ser probado antes de ser ejecutado con el propósito de tener la certeza en cuanto al logro del objetivo.

1.7. Algoritmo

Un algoritmo no es más que un conjunto de pasos secuenciales y ordenados que permiten lograr un objetivo. Todos los algoritmos tienen un inicio y un fin.

Si al momento de crear un algoritmo usted siente que no sabe qué hacer o se siente perdido, no busque más, eso simplemente significa que realmente no tenía tan claro el objetivo como había creído.

Ejemplo:

Desarrollar un algoritmo que permita adquirir el libro Don Quijote de la Mancha, escrito por el español Miguel de Cervantes Saavedra.

Objetivo:

Adquirir el libro Don Quijote de la Mancha de Miguel de Cervantes Saavedra. Mucha atención al objetivo: es solamente adquirirlo. En ningún momento el objetivo es leerlo o resumirlo ni nada; solamente adquirirlo.

Algoritmo:

Salimos del lugar en donde nos encontramos y nos dirigimos hacia una librería. En caso de que ya estemos en una, pues sencillamente preguntamos si tienen el libro; si lo tienen lo adquirimos y si no lo tienen vamos a otra librería en donde repetimos el proceso.

Explicado de esta manera, el algoritmo no va a pasar de ser un breve texto explicativo que nos va a permitir lograr el objetivo. Pero podríamos organizar este algoritmo de manera que fuese un poco más entendible y estético, comenzando con que esta vez le vamos a poner un nombre y lo vamos a generalizar para conseguir cualquier libro, siempre y cuando esté completamente definido.

Algoritmo: Adquisicion_libro

Inicio

1. Saber cuál es el libro que se quiere adquirir
2. Desplazarnos hacia una librería
3. Preguntar si tienen el libro que necesitamos
4. Si lo tienen
   Adquirirlo y parar
   Si no lo tienen
   Ir al paso 2

Fin

Note las siguientes observaciones sobre este algoritmo:

1. Los pasos están numerados.
2. Se debe cumplir la orden 1 para seguir con la orden 2.
3. Se puede adquirir cualquier libro.

4. Si probamos este ejemplo con el libro en mención, tendremos un alto porcentaje de seguridad de que lo conseguimos, a menos que esté agotado.

Si se observa con más detalle, cada línea numerada del algoritmo puede considerarse a su vez como otro algoritmo, ya que el solo hecho de saber cuál es el libro que se quiere adquirir nos obliga a realizar una serie de pasos ordenados y secuenciales para poderlo saber. Entonces surge una inquietud: ¿Cuán detallado puede ser un algoritmo? Un algoritmo debe tener el nivel de detalle suficiente como para que no exista ninguna duda en su puesta en marcha; es decir, cada línea debe ser realizada sin el más mínimo asomo de inquietud. Por lo tanto, algunos algoritmos pueden ser más entendibles para unas personas que para otras, dada su misma definición racional.

1.7.1. Tipos de algoritmos

Hay dos tipos de algoritmos: informales y computacionales. Los informales son aquellos algoritmos en donde el ejecutor real es el ser humano como, por ejemplo: algoritmo para dar un beso, algoritmo para cocinar, algoritmo para conseguir un libro. Sin embargo, en la actualidad no solo el ser humano puede realizar este tipo de algoritmos; también lo pueden realizar las máquinas, así que se considerarán como informales aquellos algoritmos que son preferiblemente realizables por el ser humano.

Los algoritmos computacionales son aquellos preferiblemente implementados por un computador con el fin de aprovechar su velocidad de procesamiento como, por ejemplo: generar los primeros 100 números primos. A lo largo de este libro, profundizaremos un poco más en este tipo de algoritmos.

1.7.2. Características de los algoritmos

Las características de un algoritmo son las siguientes:

- Es preciso e indica el orden de realización de cada paso.
- Es definido: si se sigue un algoritmo varias veces, se debe obtener el mismo resultado cada vez.
- Es finito: si se sigue un algoritmo se debe terminar en algún momento. Debe tener un número finito de pasos.

El algoritmo debe ser planteado como un sistema de información.

Figura 2. Representación de un algoritmo como un sistema de información.

Elaborado por Carlos Salazar

Entrada. Como menciona (Regino, p 45):

> La entrada hace referencia a la información proporcionada al algoritmo, la cual debe sufrir un proceso para obtener los resultados. Un algoritmo tiene cero o más datos de entrada. Estos valores le son dados por medio de una instrucción o mandato que se debe cumplir al ejecutarse el algoritmo. Si no existen datos de entrada es porque una o más instrucciones generan los valores de partida, de los que hará uso el algoritmo para producir los datos o valores de salida.

Es necesario recalcar lo siguiente: datos de entrada es toda la información que proporciona el algoritmo la cual debe ser procesada para lograr el objetivo.

Proceso. Citando a (Regino, p 46), el proceso son los "cálculos necesarios para que a partir de un dato de entrada se llegue a los resultados". Por lo tanto, se entiende como proceso al conjunto de acciones u operaciones que se realizan con o sin los datos de entrada para cumplir el objetivo y obtener los datos de salida esperados.

Salida. Haciendo referencia a (Regino, p 46), la salida son los "resultados finales o transformaciones que ha sufrido la información de entrada a través del proceso". Es decir, es el resultado u objetivo logrado gracias a la ejecución del proceso.

1.7.3. Prueba de escritorio

Para efectos técnicos, la prueba de escritorio no es más que la simulación de la puesta en marcha de un algoritmo con el fin de conocer si el algoritmo que se ha diseñado logra el objetivo. De no ser así, podremos concluir que se debe corregir el algoritmo hasta lograr que satisfaga el objetivo propuesto.

1.7.4. Ejercicios sobre algoritmos informales

Como todo en esta vida, se aprende haciendo, y crear algoritmos no es la excepción. Por ello se han propuesto los siguientes ejercicios para resolver:

1. Desarrollar un algoritmo que permita adquirir una revista.
2. Desarrollar un algoritmo que permita entrar en una casa que está con llave.
3. Desarrollar un algoritmo que permita dar un beso.
4. Desarrollar un algoritmo que permita empacar un regalo.
5. Desarrollar un algoritmo que permita encender un vehículo.
6. Desarrollar un algoritmo que permita freír un huevo.
7. Desarrollar un algoritmo que permita mirar por un telescopio.
8. Desarrollar un algoritmo que permita botar la basura.
9. Desarrollar un algoritmo que permita tomar un baño.
10. Desarrollar un algoritmo que permita estudiar para un examen.
11. Desarrollar un algoritmo que permita tocar determinada canción con un instrumento musical.
12. Desarrollar un algoritmo que permita viajar en avión.
13. Desarrollar un algoritmo que permita encender un bombillo.
14. Desarrollar un algoritmo que permita encender una vela.
15. Desarrollar un algoritmo que permita apagar una vela.

16. Desarrollar un algoritmo que permita apagar un bombillo.
17. Desarrollar un algoritmo que permita parquear un vehículo.
18. Desarrollar un algoritmo que permita almorzar.
19. Desarrollar un algoritmo que permita ir de la casa al trabajo.
20. Desarrollar un algoritmo que permita ponerse una camisa.
21. Desarrollar un algoritmo que permita quitarse la camisa.
22. Desarrollar un algoritmo que permita escuchar un determinado disco.
23. Desarrollar un algoritmo que permita abrir una ventana.
24. Desarrollar un algoritmo que permita ir a la tienda a comprar algo.
25. Desarrollar un algoritmo que permita tomar una fotografía.
26. Desarrollar un algoritmo que permita hacer deporte.
27. Desarrollar un algoritmo que permita cortarse el cabello.
28. Desarrollar un algoritmo que permita hacer un avión con una hoja de papel.
29. Desarrollar un algoritmo que permita manejar una bicicleta.
30. Desarrollar un algoritmo que permita manejar una motocicleta.
31. Desarrollar un algoritmo que permita manejar un monociclo.
32. Desarrollar un algoritmo que permita maquillarse.
33. Desarrollar un algoritmo que permita hacer un pastel.
34. Desarrollar un algoritmo que permita hacer un almuerzo.
35. Desarrollar un algoritmo que permita adquirir un pantalón.
36. Desarrollar un algoritmo que permita hacer un mercado pequeño.
37. Desarrollar un algoritmo que permita leer el periódico.
38. Desarrollar un algoritmo que permita saludar a un amigo.
39. Desarrollar un algoritmo que permita arrullar a un bebé hasta que se duerma.
40. Desarrollar un algoritmo que permita hacer un gol en fútbol.
41. Desarrollar un algoritmo que permita jugar ping pong.
42. Desarrollar un algoritmo que permita nadar.
43. Desarrollar un algoritmo que permita tirarse desde un avión con un paracaídas.
44. Desarrollar un algoritmo que permita tirarse desde un avión sin un paracaídas.
45. Desarrollar un algoritmo que permita descifrar un jeroglífico.
46. Desarrollar un algoritmo que permita amarrarse un zapato.
47. Desarrollar un algoritmo que permita quitarse los zapatos.
48. Desarrollar un algoritmo que permita silbar.
49. Desarrollar un algoritmo que permita elevar una cometa.
50. Desarrollar un algoritmo que permita desarrollar algoritmos.

1.7.5. Consideraciones algorítmicas sobre el pensamiento humano

Es importante conocer cuáles son las estructuras básicas que rigen el pensamiento humano. Según (Buriticá, 1999, p 46), concluyó que hay tres estructuras generales que rigen dicho pensamiento, de las cuales se hablará a continuación.

Cuando usted está planeando un paseo familiar para el próximo feriado, su mente va proyectando un conjunto de acciones por realizar que le permitan disfrutar de sus vacaciones tan anheladas. A este conjunto de acciones se lo conoce como secuencia.

Constantemente nuestro cerebro está inmerso en esta estructura: generalmente primero se planea cada secuencia de acciones de forma consciente o inconscientemente antes de ejecutarlas. Cada acción que se hace un día cualquiera, no es más que una secuencia de acciones que se han planeado para cumplir con nuestros objetivos en la sociedad.

Ahora, imagínese que debe elegir el sitio donde se hospedará en sus vacaciones: buscará las mejores habitaciones, un buen desayuno, servicio de parqueadero, piscina, tv cable, costo asequible, etc. Usted deberá escoger en qué hotel pasará la noche, y aquí entra la estructura de decisión, gracias a la cual puede escoger la mejor alternativa de entre varias a disposición. Se debe mencionar que esta estructura siempre se da cuando usted tiene que escoger de entre, por lo menos, dos caminos lógicos, ya que si solo tiene una alternativa simplemente no tiene alternativas que escoger.

Para finalizar, digamos que usted acostumbra a tomar el té todos los días a las 5:00 pm de lunes a domingo, los 365 días del año. En este escenario, usted vive practicando la tercera estructura que son los ciclos, la cual consiste en repetir una o varias acciones una cantidad definida de veces.

Más adelante, se profundizará en mayor detalle en cada una de las estructuras que se han explicado en esta sección. Por ahora, lo más importante es identificar las tres estructuras generales que rigen el pensamiento humano:

a. Secuencia de acciones
b. Decisión de acción
c. Ciclos de acciones

1.8. Diagrama de flujo

Un diagrama de flujo es la representación gráfica de un algoritmo, en donde cada símbolo representa diferentes acciones las cuales se unen entre sí mediante líneas que indican el orden en que se deben ejecutar los procesos.

1.8.1. Características

Toda representación gráfica debe cumplir con las siguientes cualidades:

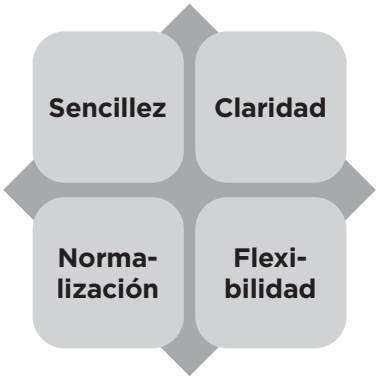


Figura 3. Características del diagrama de flujo

Elaborado por Carlos Salazar

- **Sencillez.** Un método gráfico de diseño de algoritmo se debe implementar fácilmente.
- **Claridad.** Cualquier persona puede interpretar fácilmente el algoritmo, reconociendo todos los elementos que lo conforman.
- **Normalización.** Tanto los diseñadores de programas como los usuarios necesitan utilizar la misma norma de documentación.
- **Flexibilidad.** Es de fácil modificación.

1.8.2. Simbología

Los símbolos utilizados en los diagramas de flujo que se exponen a continuación hacen referencia al simulador DFD que se utilizará en los ejemplos posteriores.

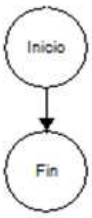
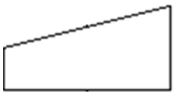
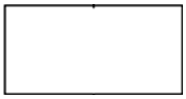
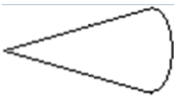
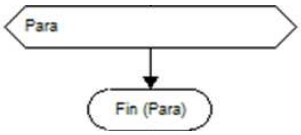
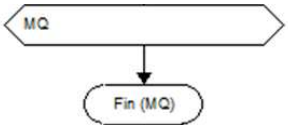
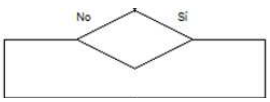
Símbolo	Descripción
	Indica el inicio y el fin de un algoritmo. Los procesos deben estar dentro de esta estructura.
	Representa la entrada de datos. Se debe escribir en su interior el nombre de la variable en donde se desee almacenar el dato que entra por el teclado.
	Este símbolo representa una acción o una orden por ejecutar de manera clara y concreta. Un ejemplo típico es la asignación de un dato a una variable.
	Permite escribir una salida por pantalla. Puede ser el resultado de una operación o un mensaje.
	Permite representar el ciclo repetitivo Para.
	Permite representar el ciclo repetitivo Mientras.
	Permite escoger el camino por seguir en base de una pregunta que devuelve un valor de verdadero o falso.

Figura 4. Simbología del diagrama de flujo

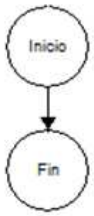
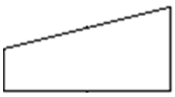
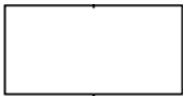
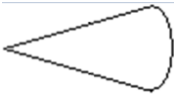
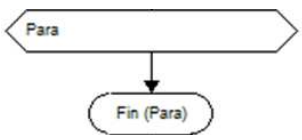
Elaborado por Carlos Salazar

1.9. Pseudocódigo

Para lograr implementar un algoritmo que permita alcanzar un objetivo, necesariamente se tuvo que analizar el problema, en donde se identificó la entrada, proceso y salida. Posteriormente se realiza la implementación gráfica utilizando diagramas de flujo y finalmente se debe proceder a la etapa de codificación, y es precisamente aquí donde se utiliza el pseudocódigo.

Como menciona (M.C. Edgar E. García Cano, p 1): "Un pseudocódigo es la representación escrita de un algoritmo, es decir, muestra en forma de textos los pasos a seguir para solucionar un problema. El pseudocódigo posee una sintaxis propia para poder realizar la representación del algoritmo (solución de un problema)". Aterrizando algunas ideas, los diagramas de flujo permiten representar de manera gráfica el algoritmo, mientras que el pseudocódigo permite la representación textual del mismo, con la característica de usar una sintaxis propia la cual se debe utilizar para simular las acciones del algoritmo.

En los ejercicios propuestos en este libro se utilizará el simulador PseInt para crear el pseudocódigo de los algoritmos utilizados de ejemplo. A continuación, se presenta la representación textual de los objetos en su representación gráfica.

Símbolo	Descripción
	**Proceso** <Nombre de proceso> **FinProceso**
	**Proceso** <Nombre de proceso> **Leer** <variable> **FinProceso**
	La asignación de datos en PseInt se la realiza con el símbolo "=". Ejemplo: x = 1 + 2
	**Proceso** <Nombre de proceso> **Escribir** <variable / texto> **FinProceso**
	**Para**<variable_numerica> <valor_inicial> **Hasta** <valor_final> **Con** **Paso <**paso> **Hacer** secuencia_de_acciones **Fin Para**

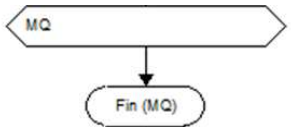
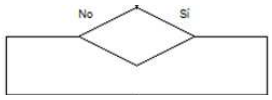
	**Mientras** <expresion> **Hacer** <instrucciones si el resultado es verdadero> **FinMientras**
	**Si** <expresion> **Entonces** <instrucciones si el resultado es verdadero> **Sino** <instrucciones si el resultado es falso> **FinSi**

Figura 5. Pseudocódigo vs Diagrama de flujo

Elaborado por Carlos Salazar

1.9.1. Empezando a programar

A lo largo de mi carrera profesional, tanto como Ingeniero de Sistemas y como Docente de Educación Superior, compañeros de trabajo y estudiantes de las cátedras que he impartido me han preguntado: ¿Cuál es la mejor manera de aprender a programar? Pues bien, antes de responder a esta pregunta hay que mencionar que hay personas que adquieren este conocimiento de diferentes maneras: lectura de libros, videotutoriales, cursos en línea, clases presenciales, etc. Hay estudiantes que haciendo uso de estos medios logran desarrollar la habilidad de programar; sin embargo, hay un número considerable de estudiantes que no logran entender los principios de programación y terminan frustrados y decepcionados.

Aprender a programar desde cero, no es recomendable debido a que los lenguajes de programación están compuestos de instrucciones poco intuitivas para los recién llegados, aparte de que en su gran mayoría tienen una sintaxis en lengua inglesa lo que complica un poco más a las personas que están empezando.

Considero que la mejor manera de aprender a programar es aquella en donde las instrucciones se pueden entender de forma natural, en donde se pueden aprender las bases en la lengua materna, y el lenguaje perfecto para hacerlo se llama pseudocódigo.

A lo largo de este libro se busca desarrollar el pensamiento lógico orientado a la programación, estudiando sus puntos críticos, los mismos que se aplican en todos los lenguajes de programación sin importar su sintaxis. Recuerde que atrás de cualquier código fuente hay una lógica que permite su correcto funcionamiento y es precisamente esa lógica la que se pretende desarrollar a continuación.

2. Capítulo II

2.1. Variables y constantes

Al escuchar la parable variable, seguramente viene la idea de algo que puede cambiar de un momento a otro; es decir, algo que no es fijo, que varía en el tiempo. Esta idea no está muy lejos de su conceptualización técnica, donde una variable es un campo de memoria al que se le puede cambiar su contenido cuantas veces sea necesario.

Para poder utilizar variables primero se debe definir qué tipo de dato van a almacenar. Imagine que las variables son pequeñas cajas de diferentes tamaños, lo que se debe aclarar previamente para que el computador asigne las dimensiones de memoria de acuerdo con las necesidades.

Por otro lado, las constantes son valores que no deben ser alterados o modificados durante la ejecución de un programa, es decir, siempre tendrán el mismo valor y será necesario definir qué tipo de dato van a almacenar. A continuación, se explican los principales tipos de datos que se utilizarán a lo largo de este libro.

2.1.1. Tipos de datos

Entre los principales tipos de datos, se tiene:

- **Entero.** Maneja números que no tienen punto decimal, por lo tanto en sus operaciones jamás va a generar decimales.
- **Real.** Trabaja con números que tienen punto decimal, por lo tanto en sus operaciones puede generar punto decimal.
- **Carácter.** Este tipo de dato puede almacenar letras, signos de puntuación, números, operadores, y se delimitan con comillas simples.
- **Lógico.** Puede tomar valores booleanos (verdadero y falso).

Observe y analice el tipo al que pertenece cada dato en la siguiente figura:

Dato	Tipo
25	Entero
'A'	Carácter
-4	Entero
2/5	Real
'.'	Carácter

Figura 6. Clasificación tipo de dato

Elaborado por Carlos Salazar

Nótese que en la Figura 6 hay datos entre comillas simples, por ejemplo: 'A'; '.'. También puede darse el caso de tener números '6'; '1,5'. Cuando el dato se encuentra entre comillas será almacenado como tipo carácter, sin importar que su contenido sea un número o símbolo. Una vez identificados los tipos de datos, es necesario conocer qué operaciones se puede realizar sobre ellos. En la siguiente sección encuentra más información sobre este tema.

2.2. Operadores

Un operador es una entidad que actúa sobre los datos para modificarlos. A continuación se exponen los diferentes tipos de operadores que se utilizarán en este trabajo.

2.2.1. Operador de asignación

Está representado por la expresión: "=", la cual permite asignar los valores resultantes de las operaciones del lado derecho en la variable o identificador de la parte izquierda. Observe los siguientes ejemplos:

Operación	Descripción
A = 3	Guarda el valor 3 en la variable "A".
X = y	Guarda el valor que tenga la variable "y" en la variable "X".
Suma = a + b	Guarda en la variable "Suma" el resultado de sumar "a" + "b".
Área = longitud * ancho	Guarda la multiplicación de los valores de las variables "longitud" y "ancho" en la variable "Área".

Figura 7. Operador de asignación

Elaborado por Carlos Salazar

Como se observa en la figura 7, el operador de asignación funciona para todos los tipos de datos. Fíjese que al lado izquierdo se ubican las variables y en el derecho las operaciones.

2.2.2. Operadores aritméticos

Permiten realizar operaciones matemáticas devolviendo como resultado un valor numérico. A continuación se muestra el operador y la operación que representa.

Operador	Operación
+	Suma
-	Resta
*	Multiplicación
/	División

Figura 8. Operadores aritméticos

Elaborado por Omar Iván Trejos

Preste atención a los siguientes ejemplos:

- **Si A = 10 y B = 3**

Expresión	Resultado
A + B	13
A – B	7
A * B	30
A / B	3
A % B	1

- **Si A = 12.5 y B = 2.0**

Expresión	Resultado
A + B	14.5
A – B	10.5
A * B	25.0
A / B	6.25
A % B	TIPOS INCORRECTOS

2.2.3. Operadores relacionales

Son símbolos utilizados para comparar dos valores. Al utilizar este tipo de operadores, el resultado siempre será un valor lógico: Verdadero (1) o Falso (0), también conocidos como valores booleanos. Observe en la siguiente figura los tipos de operadores relacionales que se pueden utilizar.

Operador relacional	Función
<	Menor que
< =	Menor o igual que
>	Mayor que
> =	Mayor o igual que
Operador de igualdad	Función
= =	Igual que
!=	No igual que

Figura 9. Operadores relacionales

Elaborado por Omar Iván Trejos

Observe los siguientes ejemplos:

- **Si I = 1, J = 2, K = 3**

Expresión	Interpretación	Valor
I < J	VERDADERO	1
(I + J) > = K	VERDADERO	1
(J + K) > (I + 5)	FALSO	0
K != 3	FALSO	0
I = = J	FALSO	0
K < = I	FALSO	0

2.2.4. Operadores lógicos

Pueden o no trabajar en conjunto con los operadores relacionales permitiendo combinar expresiones lógicas sencillas para generar expresiones lógicas complejas. Observe la siguiente figura:

Operador	Descripción
Y	Conjunción
O	Disyunción
NO	Negación

Figura 10. Operadores lógicos

Elaborado por Omar Iván Trejos

A continuación, se muestra una tabla de verdad para visualizar los valores que devolvería la expresión según su combinación.

p	q	p Y q	p O q	!p
FALSO	FALSO	FALSO	FALSO	VERDADERO
FALSO	VERDADERO	FALSO	VERDADERO	VERDADERO
VERDADERO	FALSO	FALSO	VERDADERO	FALSO
VERDADERO	VERDADERO	VERDADERO	VERDADERO	FALSO

Observe el siguiente ejemplo:

- **Si I = 7, F = 5.5 y C = 10**

EXPRESIÓN	INTERPRETACIÓN	VALOR
(I > = 6) Y (C == 10)	VERDADERO	1
(I > = 6) O (C == 9)	VERDADERO	1
(F < 11) Y (I > 10)	FALSO	0
(C != 15) O ((I + F)< = 10)	VERDADERO	1
(I > = 6) O (C == 65)	VERDADERO	1

2.2.5. Operadores de entrada

Consiste en capturar datos desde un periférico de entrada como el teclado. Los datos capturados van a la memoria RAM.

En PseInt se utiliza la operación “Leer”.

2.2.6. Operadores de salida

Consiste en enviar datos desde un área de la memoria RAM hacia un periférico de salida como el monitor.

En PseInt se utiliza la operación “Escribir”.

2.3. Reglas de prioridades

Cuando se trabaja con expresiones matemáticas hay reglas que indican qué operaciones se deben realizar primero. Por ejemplo, en una operación que contenga suma, resta, multiplicación y división, primero hay que resolver la multiplicación y división para luego resolver la suma y la resta.

A continuación, se muestra una tabla de jerarquías que será de utilidad a la hora de trabajar con expresiones matemáticas.

Categoría	Operador	Prioridad
Aritmético	*, /, %	1
Aritmético	+, -	2
Relacionales	<, <=, >, >=	3
Igualdad	==, !=	4
And lógico	Y	5
Or lógico	O	6
Asignación	=	7

Figura 11. Tabla de prioridad o jerarquía

Elaborado por Omar Iván Trejos

Observe los siguientes ejemplos:

- **2 + 4 x 2**

  Como primer paso, se debe resolver la multiplicación: 4 x 2 = 8

  Como segundo paso, se resuelve la suma: 2 + 8 = 10

- **9 – 4 / 2**

  Como primer paso, se resuelve la división: 4 /2 = 2

  Como segundo paso, se resuelve la resta: 9 – 2 = 7

Si en la expresión matemática hay agrupamiento por paréntesis, se debe resolver desde los paréntesis interiores hacia los exteriores. Observe los siguientes ejemplos:

- **(5 + 1) x 2 = (6) x 2** = 12

- **(9 – 4) + (2 x 3) = (5) + (6)** = 11
- **(5 - 1) x (7 - 2) / 5 = (4) x (5) / 5** = 4
- **((15 - 3) x 4) – 1 x ((5 + 3) x 4)**
  = ((12) x 4) – 1 x ((8) x 4)
  = (48) -1 x (32)
  = 48 – 32
  = 16

2.4. La secuencia

En secciones anteriores, se abordó el tema de algoritmos informales, en donde, siguiendo una serie de pasos ordenados de manera lógica, se logra alcanzar un objetivo (resolver un problema), pues bien, esta estructura se trata precisamente de eso, ejecutar un conjunto de acciones una tras otra hasta alcanzar el objetivo.

Se puede representar a esta estructura de la siguiente manera:

Figura 12. Estructura de secuencia

Elaborado por Carlos Salazar

A partir de este instante, se trabajará con algoritmos computaciones, aplicando el principio de secuencia. Observe el siguiente ejemplo:

Entero: A, B, C	**Declara que las variables A, B y C son de tipo entero, de manera que solo podrán almacenar datos enteros.**
A = 10	Almacena el dato 10 en la variable A.
B = 15	Almacena el dato 15 en la variable B.
C = 20	Almacena el dato 20 en la variable C.
A = A + B	Almacena en la variable A el resultado de sumar el contenido de A más el contenido de B, en otras palabras, 10 + 15 es igual a 25.
B = B + 8	Almacena en la variable B el resultado de sumar el contenido de B con el valor de 8, en otras palabras, 15 + 8 que es igual a 23.
C = C + A	Almacena en la variable C el resultado de sumar el contenido de la variable C más el contenido de la variable A. En otras palabras, 20 + 25 que es igual a 45. Fíjese que en esta línea se utiliza el último valor de almacenado en la variable A.

Al finalizar, las variables A, B y C tendrían los siguientes resultados:

A = 25 B = 23 C = 45

A continuación, se utilizará el ejercicio anterior pero representado a través de un algoritmo. Para ello es necesario analizar el problema para su posterior implementación. Como primer paso, en todo algoritmo se debe visualizar el objetivo, para ello es necesario identificar entrada, proceso y salida.

■ ANÁLISIS

ENTRADA

Como datos de entrada se puede observar lo siguiente:

- Se tienen tres variables: A, B y C.
- Se asigna a la variable A el valor de 10.
- Se asigna a la variable B el valor de 15.
- Se asigna a la variable C el valor de 20.

PROCESO

Sobre las variables A, B y C, se deben realizar las siguientes operaciones matemáticas:

- A = A + B
- B = B + 8
- C = C + A

Es importante mencionar que dichas operaciones deben ser ejecutadas en un orden determinado, pues en caso contrario el resultado podría ser diferente del esperado.

SALIDA

Mostrar el valor final que contiene la variable A, B y C.

Una vez que se ha identificado la entrada, proceso y salida del algoritmo, se podría decir que se tiene claro el objetivo, por lo tanto, se conoce qué se debe lograr y hasta dónde se debe llegar. A continuación se procede a crear el algoritmo en base del análisis realizado.

ALGORITMO

Algoritmo: Calculo_matematico

Inicio

1. Asignar a la variable A el valor de 10.
2. Asignar a la variable B el valor de 15.
3. Asignar a la variable C el valor de 20.
4. Asignar a la variable A la suma de A y B.
5. Asignar a la variable B la suma entre B y 8.
6. Asignar a la variable C la suma entre C y A.
7 Mostrar en pantalla el valor de la variable A.
8. Mostrar en pantalla el valor de la variable B.
9. Mostrar en pantalla el valor de la variable C.

Fin

Una vez identificado el algoritmo, es necesario probar su funcionamiento. Para ello, se utiliza la prueba de escritorio, que permite ejecutar paso a paso las instrucciones del algoritmo para evaluar si se obtienen o no los resultados deseados.

Operaciones	A	B	C
A = 10	10		
B = 15	10	15	
C = 20	~~10~~	15	20
A = A + B	25	~~15~~	20
B = B + 8	25	23	~~20~~
C = C + A	**25**	**23**	**45**

Como se puede observar, se logra el resultado esperado en las variables A, B y C.

A = 25 B = 23 C = 45

Nótese que cuando se identifica correctamente la entrada, proceso y salida, el desarrollo del algoritmo resulta bastante fácil de lograr. Una vez que se tiene diseñado el algoritmo, se procede a representarlo de manera gráfica (diagrama de flujo). Para ello, se recomienda utilizar el simulador DFD. Observe su implementación en la siguiente figura:

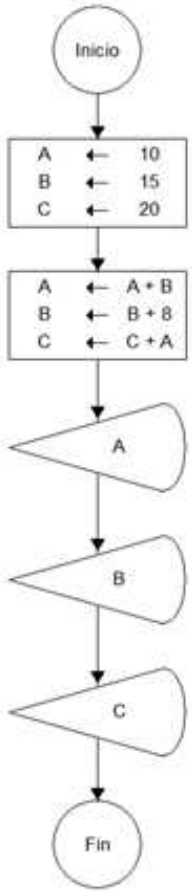


Figura 13. Diagrama de flujo secuencia

Elaborado por Carlos Salazar

Al ejecutar el diagrama de flujo en el simulador, se obtendrán los siguientes resultados:

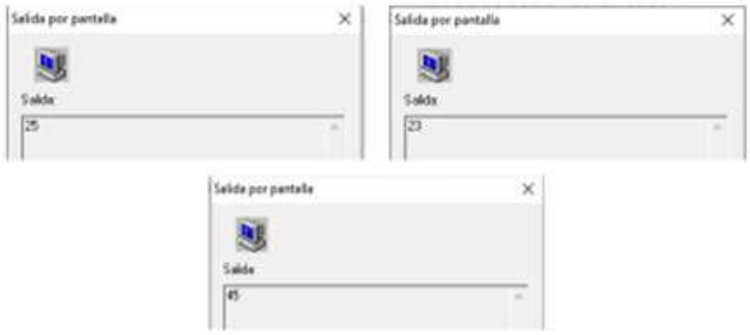


Figura 14. Diagrama de flujo resultado secuencia

Elaborado por Carlos Salazar

Como se puede observar, los resultados esperados para las variables A, B y C se han cumplido, por lo tanto se puede considerar que se ha resuelto el algoritmo. Finalmente se procederá a representar el algoritmo en Pseudocódigo.

Para ello, se utilizará el programa PseInt. Observe su implementación en la siguiente figura:

```
Proceso calculo_matematico
    Definir A Como Entero;
    Definir B Como Entero;
    Definir C Como Entero;

    A = 10;
    B = 15;
    C = 20;

    A = A + B;
    B = B + 8;
    C = C + A;

    Escribir A;
    Escribir B;
    Escribir C;
FinProceso
```

Figura 15. Pseudocódigo secuencia

Elaborado por Carlos Salazar

En relación a la figura 15, note las siguientes observaciones:

- El bloque “Inicio” y “Fin” de un algoritmo, se reemplaza por el bloque “Proceso” y “FinProceso” en pseudocódigo.
- El nombre del algoritmo en pseudocódigo se coloca junto a la sección “Proceso”.
- Antes de hacer uso de las variables, primero se las debe definir, además de indicar el tipo de dato que almacenará.

Al ejecutar el pseudocódigo, se obtiene el siguiente resultado:

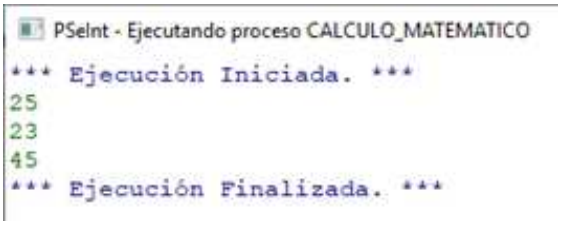


Figura 16. Ejecución secuencia

Elaborado por Carlos Salazar

En el transcurso de este libro se representarán los algoritmos utilizando diagramas de flujo con el simulador DFD y en pseudocódigo con PseInt. Como se mencionó en capítulos anteriores, todo el software que se utiliza en el texto se encuentra disponible en Internet para su descarga.

2.4.1. Ejercicios etapa I

Se recomienda resolver los siguientes ejercicios identificando el algoritmo con su respectivo análisis (entrada, proceso, salida), diagrama de flujo y pseudocódigo.

1. Determinar el valor de A, B y C al finalizar la siguiente serie de instrucciones:
   - A = 10
   - B = 15
   - C = 20
   - A = A + B
   - B = B + 8
   - C = C + A
   - A = A + 5
   - B = B + 3
   - C = C + 2
   - A = A – B
   - B = A – B
   - C = A – B
   - A = A – C

2. Determinar el valor de A, B y C al finalizar la siguiente serie de instrucciones:
   - A = 10
   - B = 20
   - C = 10
   - A = A + 15
   - B = B + 12
   - C = A * C
   - A = B + 1

3. Determinar el valor de A, B y C al finalizar la siguiente serie de instrucciones:
   - A = 3
   - B = 8
   - C = 1
   - A = 5

- B = 9
- C = 7
- A = A + 1
- B = B + 2
- C = C + 3
- A = A + C

4. Determinar el valor de A, B y C al finalizar la siguiente serie de instrucciones:
   - A = 1
   - B = 2
   - C = 3
   - A = A + B
   - B = A – B
   - C = A * B
   - A = A – B
   - B = A + B
   - C = A / B

5. Determinar el valor de A, B y C al finalizar la siguiente serie de instrucciones:
   - A = 1
   - B = 2
   - C = 3
   - A = A + 2
   - B = A + 2 + B
   - C = A + 2 + C
   - A = A / 2
   - B = B / 2
   - C = C / 2

Linealizar

Algo que se debe tener en cuenta al momento de escribir una expresión es que el computador solo entiende expresiones en formato linealizado, es decir, escritas en una sola línea.

Expresión: $$var = \frac{a+b}{c+d}$$

Transformación: $$var = a + b/c + d$$

Agrupación

La agrupación es una técnica que permite alterar la jerarquía de operadores utilizando la agrupación con paréntesis, lo que permite llegar a fórmulas específicas. Es necesario mencionar que un paréntesis mal colocado finalmente hace que los resultados que calcule el computador sean diferentes a los esperados.

2.4.2. Ejercicios etapa II

Se recomienda resolver los siguientes ejercicios identificando el algoritmo con su respectivo análisis (entrada, proceso, salida), diagrama de flujo y pseudocódigo.

1. Determinar el valor de X si:
   - A = 5, B = 4 y C = 2

$$X = \frac{A + \frac{B}{C}}{\frac{A}{B} + C}$$

2. Determinar el valor de X si:
   - A = 10, B = 5 y C = 4

$$X = \frac{A + B + \frac{A}{B}}{C}$$

3. Determinar el valor de X si:
   - A = 2, B = 4, C = 2 y D = 1

$$X = \frac{A + B + \frac{C}{D * A}}{A + B * \frac{C}{D}}$$

4. Determinar el valor de X si:
   - A = 1, B = 2, C = 4 y D = 1

$$X = \frac{A + \frac{B}{C} + D}{A}$$

5. Determinar el valor de X si:
   - A = 4, B = 2, C = 3 y D = 1

$$X = A + \frac{A + \frac{A + B}{C + D}}{A + \frac{A}{B}}$$

6. Determinar el valor de X si:
   - A = 2, B = 3, C = 4 y D = 6

$$X = A + B + \frac{C}{D} + \frac{\frac{A}{B - C}}{\frac{A}{B + C}}$$

7. Determinar el valor de X si:
   - A = 5, B = 6, C = 1 y D = 2

$$X = B + \frac{A}{D} + \frac{\frac{A}{B - C}}{\frac{D * A}{B + C}} + \frac{D}{A}$$

8. Determinar el valor de X si:
   - A = 6, B = 2, C = 3

$$X = C + A + \frac{C}{A} + \frac{\frac{A}{B * A}}{\frac{A}{B * C}}$$

Una vez familiarizados con la creación de algoritmos y su representación tanto en diagramas de flujo como en pseudocódigo, ha llegado la hora de implementar algoritmos un poco más adaptados a un entorno real. Preste atención al siguiente ejemplo:

■ ENUNCIADO

Calcule la paga neta de un trabajador. Para ello se debe solicitar el ingreso, por teclado, de las horas trabajadas, la tarifa horaria y la tasa de impuestos.

ANÁLISIS

El primer paso consiste en identificar y entender el objetivo. Para ello es necesario analizar el enunciado para determinar la entrada, proceso y salida.

ENTRADA

- Horas trabajadas
- Tarifa horaria
- Tasa de impuestos

PROCESO

- Leer por teclado las horas_trabajadas
- Leer por teclado la tarifa_horaria
- Leer por teclado la tasa_impuesto
- Calcular paga = horas_trabajadas * tarifa_horaria
- Calcular impuesto = (tasa_impuesto * paga) / 100
- Calcular paga_neta = paga – impuesto

SALIDA

- Mostrar en pantalla el valor de paga_neta

ALGORITMO

Algoritmo: Calcular_paga

Inicio

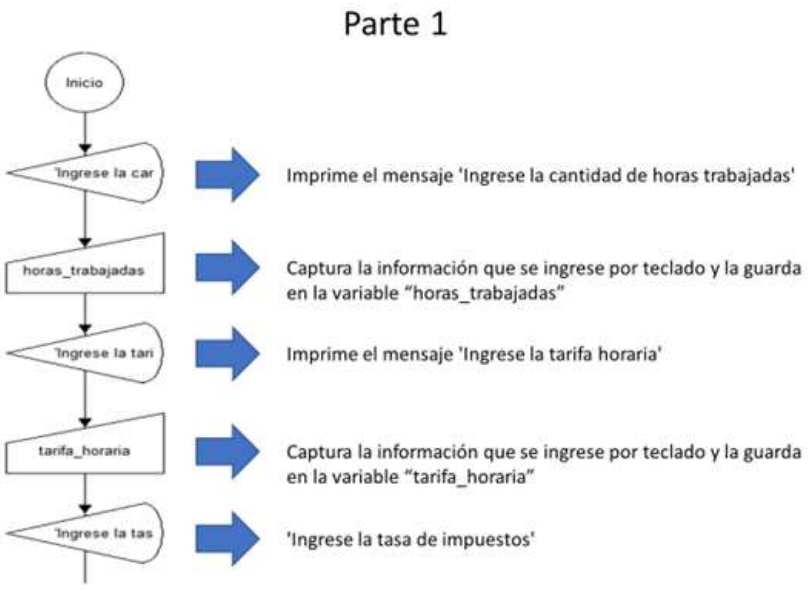
1. Solicitar al usuario que ingrese la cantidad de horas trabajadas.
2. Almacenar el valor en una variable llamada “horas_trabajadas”.
3. Solicitar al usuario que ingrese la tarifa horaria.
4. Almacenar el valor en una variable llamada “tarifa_horaria”.
5. Solicitar al usuario que ingrese la tasa de impuesto.
6. Almacenar el valor en una variable llamada “tasa_impuesto”.
7. En la variable “paga” almacenar el resultado de multiplicar “horas_trabajadas” por “tarifa_horaria”.
8. En la variable “impuesto” almacenar el resultado de multiplicar “tasa_impuesto” por “paga” y dividir para 100.
9. En la variable “paga_neta” almacenar el resultado de restar la “paga” menos el “impuesto”.
10. Mostrar el valor de la variable “paga_neta”.

Fin

Prueba de escritorio

Operaciones	Horas_ trabajadas	Tarifa_ horaria	Tasa_ impuesto	Impuesto	Paga	Paga_ neta
Ingresar la cantidad de horas trabajadas.						
Almacenar el valor en una variable llamada “horas_ trabajadas”.	100					
Solicitar al usuario que ingrese la tarifa horaria.						
Almacenar el valor en una variable llamada “tarifa_ horaria”.		10				
Solicitar al usuario que ingrese la tasa de impuesto.						
Almacenar el valor en una variable llamada “tasa_ impuesto”.			12			
En la variable “paga” almacenar el resultado de multiplicar “horas_ trabajadas” por “tarifa_ horaria”.					1000	
En la variable “impuesto” almacenar el resultado de multiplicar “tasa_impuesto” por “paga” y dividir para 100.				120		
En la variable “paga_neta” almacenar el resultado de restar la “paga” menos el “impuesto”.						880
Mostrar el valor de la variable “paga_neta”.						880

Diagrama de flujo



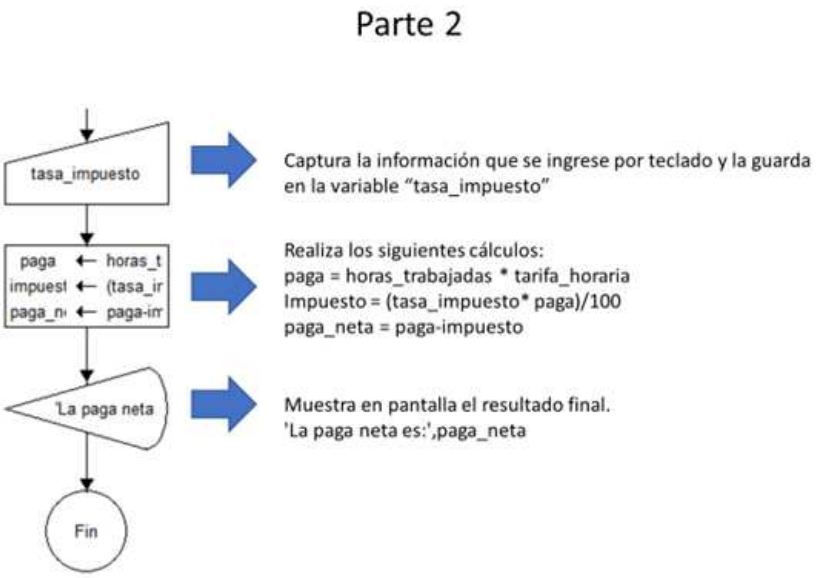


Figura 17. Diagrama de flujo ejercicio paga neta

Elaborado por Carlos Salazar

En base del ejercicio anterior, nótese los siguientes puntos:

- Los textos se escriben entre comillas simples ‘ ’.
- El símbolo coma (,) permite concatenar textos con variables.
- En algunos casos es importante utilizar los paréntesis para priorizar el orden de las operaciones.

Al ejecutar el algoritmo se obtendrá el siguiente resultado:

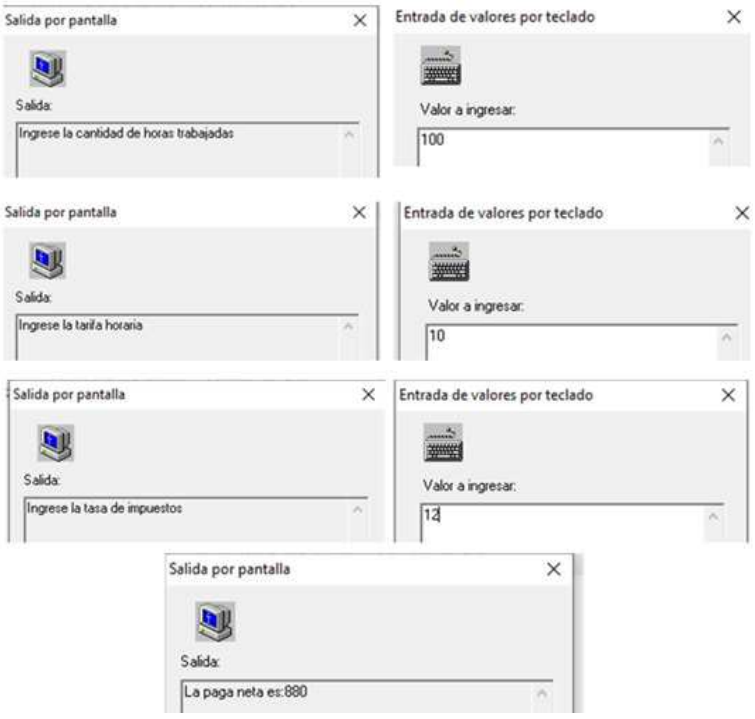


Figura 18. Ejecución paga neta

Elaborado por Carlos Salazar

Como se puede observar al momento de ejecutar el diagrama de flujo, el resultado obtenido coincide con el obtenido en la prueba de escritorio. A continuación, se muestra la implementación del algoritmo utilizando pseudocódigo:

Pseudocódigo

```
Proceso Calcular_paga
    Definir horas_trabajadas Como Entero;
    Definir tarifa_horaria Como Real;
    Definir tasa_impuesto Como Real;
    Definir impuesto Como Real;
    Definir paga Como Real;
    Definir paga_neta Como Real;

    Escribir "Ingrese la cantidad de horas trabajadas";
    Leer horas_trabajadas;
    Escribir "Ingrese la tarifa horaria";
    Leer tarifa_horaria;
    Escribir "Ingrese la tasa de impuestos"
    Leer tasa_impuesto;

    paga = horas_trabajadas * tarifa_horaria;
    impuesto = (tasa_impuesto * paga)/100;
    paga_neta = paga - impuesto;

    Escribir "La paga neta es: ",paga_neta;
FinProceso
```

Figura 19. Pseudocódigo paga neta

Elaborado por Carlos Salazar

Al ejecutar el pseudocódigo de la figura anterior e ingresando los datos que se han tomado como ejemplo, se podrá visualizar lo siguiente:

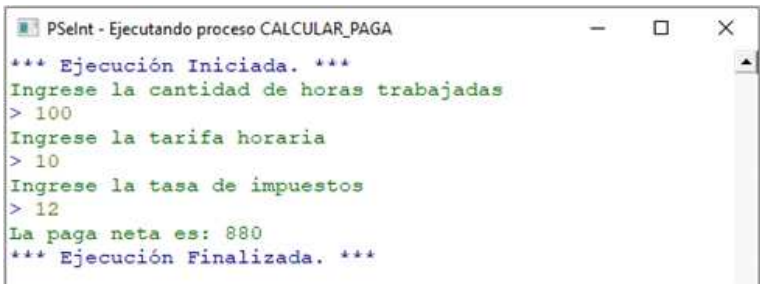


Figura 20 Ejecución pseudocódigo paga neta

Elaborado por Carlos Salazar

Como se puede notar, con un correcto análisis del problema es posible desarrollar un algoritmo eficiente que permite llegar al objetivo. Utilizando dicho algoritmo, es posible crear un diagrama de flujo sin mayor complicación y finalmente, en base del diagrama de flujo, se puede implementar la solución en pseudocódigo obteniendo en todos los métodos el mismo resultado.

A continuación, se muestra una serie de ejercicios propuestos para que el estudiante fortalezca y desarrolle la habilidad de resolver y representar algoritmos utilizando una estructura secuencial.

2.4.3. Ejercicios etapa III

Se recomienda resolver los siguientes ejercicios identificando el algoritmo con su respectivo análisis (entrada, proceso, salida), diagrama de flujo y pseudocódigo.

1. Elabore un algoritmo que permita leer dos números por teclado y los muestre por pantalla.
2. Escriba un algoritmo que permita leer dos números enteros y muestre su suma, resta, multiplicación y división.
3. Elabore un algoritmo que lea un número por teclado y obtenga su valor en negativo.
4. Elabore un algoritmo que lea un número por teclado y obtenga su cuadrado y su cubo.
5. Elabore un algoritmo que pregunte al usuario su nombre y luego muestre un mensaje de bienvenida con su nombre, por ejemplo: "Bienvenido Luis".
6. Escriba un algoritmo que encuentre el salario semanal de un trabajador, dadas la tarifa horaria y el número de horas trabajadas diariamente.
7. Escriba un algoritmo que permita calcular el área de un rectángulo.
8. Elabore un algoritmo que permita determinar el área de un cuadrado.
9. Escriba un algoritmo que permita conocer el área de un triángulo a partir de la base y la altura.
10. Dadas dos variables numéricas A y B, que el usuario debe teclear, elabore un algoritmo que intercambie los valores de ambas variables e imprima el contenido de cada una.
11. Elabore un algoritmo que permita ingresar por teclado cuatro calificaciones de un estudiante y muestre su promedio final.
12. Elabore un algoritmo que solicite el número de respuestas correctas, incorrectas y en blanco, correspondientes a un postulante para ingresar en la carrera de Ingeniería. Al finalizar, debe mostrar el puntaje final, considerando que por cada respuesta correcta tendrá 4 puntos, por cada respuesta incorrecta tendrá -1 y por cada respuesta en blanco tendrá 0.
13. Desarrolle un algoritmo que permita calcular el precio de un boleto de viaje, tomando en cuenta el número de kilómetros que se van a recorrer, con el precio de $2,50 por km recorrido.
14. Elabore un algoritmo que permita calcular el cambio de moneda de dólares a euros y pesos mexicanos.
15. Elabore un algoritmo que lea una distancia en metros y la transforme a pies y pulgadas.
16. Elabore un algoritmo que permita transformar el peso en kilos de una persona a gramos, libras y toneladas.
17. Elabore un algoritmo que permita calcular el descuento y el monto por pagar por un medicamento cualquiera en una farmacia si todos los medicamentos tienen un descuento del 25%.

18. Elabore un algoritmo que permita calcular el nuevo salario de un empleado si obtuvo un incremento del 10% sobre su salario actual y un descuento de 2,5% por servicios.
19. Elabore un algoritmo que permita obtener la edad de una persona en meses, si se ingresa su edad en años y meses. Ejemplo: Ingresando 3 años 4 meses debe mostrar 40 meses.
20. Suponga que un individuo desea invertir su capital en un banco y quiere saber cuánto dinero ganará después de un año si el banco paga 1,5% de interés mensual. Elabore un algoritmo que permita conocer lo solicitado.
21. Elabore un algoritmo que permita calcular el total de una factura considerando el precio unitario, cantidad, precio total y el IVA del 12%. Al finalizar debe mostrar el subtotal, IVA y total.
22. Dada la temperatura en grados Celsius escribir un algoritmo que la transforme a grados Fahrenheit

$$F = (C * \left(\frac{9}{5}\right) + 32)$$

23. Escriba un algoritmo que dada la edad de una persona en años calcule su edad en segundos.
24. Suponga que se guarda en el banco una cantidad de "C" dólares por un tiempo determinado de "N" años a una tasa de interés denominada "I". La cantidad que se va a tener luego de ese tiempo viene dada por la siguiente ecuación:

$$C" = C * (1 + \frac{i}{100})$$

    Al finalizar, el algoritmo debe indicar el valor de "C".
25. Escriba un algoritmo que pida al usuario una cantidad de segundos y muestre su equivalente en horas y minutos.
26. Elabore un algoritmo que permita calcular el número de CD necesarios para hacer una copia de seguridad de la información almacenada en un disco duro cuya capacidad se ingresa por teclado. Considere que el disco duro está lleno de información, expresada en gigabytes. Un CD virgen tiene 700 Mb de capacidad y un Gibagyte es igual a 1024 Mb.
27. Una persona recibe un préstamo de N dólares de un banco a X años y se desea saber cuánto pagará de interés, si el banco le cobra una tasa del 30% anual. Elaborar un algoritmo que dé solución al problema.
28. En un hospital hay 3 áreas: Urgencias, Pediatría y Traumatología. El presupuesto anual del hospital se reparte de la siguiente manera:
    - Urgencias: 30%
    - Pediatría: 40%
    - Traumatología: 30%
    - Elaborar un algoritmo que permita obtener la cantidad de dinero que recibirá cada área para cualquier monto presupuestal.

29. Un vendedor recibe, a más de su sueldo, el 12 % extra por comisión de ventas que realice al mes. Se requiere un algoritmo que lea el sueldo, total de ventas y muestre el total de comisión y el total neto por cobrar.
30. Elabore un algoritmo que permita conocer el porcentaje de hombres y el porcentaje de mujeres que hay en un determinado curso.

2.5. Las decisiones

Como se estudió en secciones anteriores, la decisión es la segunda estructura sobre la cual se basa el pensamiento humano. Gracias a ella, se puede escoger la mejor alternativa según se crea conveniente. Hay que considerar que siempre se requiere de entrada por lo menos dos caminos lógicos para escoger, porque en caso contrario no se estaría trabajando con esta estructura, sino más bien con una estructura secuencial.

Por ejemplo, si usted desea viajar a la ciudad de Quito desde Guayaquil debe escoger una de dos alternativas: vía aérea o vía terrestre. ¿Cuál escoge? Sin duda, el resultado va a depender de algunos criterios por considerar, como: si el pasaje es más barato vía terrestre, entonces elijo esta alternativa. Como tengo prisa en llegar y tengo el dinero suficiente, decido ir en avión. Lo mismo sucede con los algoritmos: cuando es necesario tomar una decisión o camino por seguir, fíjese que en este ejemplo se tuvo dos escenarios u opciones para escoger.

Citando a (Regino, p 52):

> La estructura decisión lógica o selectiva está formada por una condición de tipo lógico que puede ser simple o compuesta, de la que salen dos posibles caminos: un conjunto de acciones o secuencias a ejecutar, si el resultado de la condición es verdadera; u otro conjunto de acciones o secuencias a realizar, si el resultado de la condición es falsa.

Cuando se menciona que la condición puede ser simple, eso quiere decir que en la condición únicamente se utilizan los operadores relacionales, pero cuando se hace referencia a una condición compuesta, se combinan los operadores relacionales con los operadores lógicos, y se obtendrá como resultado dos posibles valores (VERDADERO o FALSO).

A continuación, se describen las variaciones que se pueden encontrar dentro de esta estructura.

2.2.1. Si-Entonces simple

Se utiliza cuando se necesita realizar preguntas lógicas para identificar cuál es el camino que debe tomar el flujo del algoritmo. Esta estructura está formada por dos partes: una expresión lógica y un conjunto o secuencia de instrucciones o acciones por ejecutar. Su estructura es la siguiente:

Primera Forma

Si <expresion> **Entonces**

<instrucciones si el resultado es verdadero>

FinSi

Segunda Forma

Si <expresion> **Entonces**

<instrucciones si el resultado es verdadero>

Sino

<instrucciones si el resultado es falso>

FinSi

Figura 21. Estructura decisión Si-Entonces simple

Elaborado por Carlos Salazar

Como puede observar en la figura anterior, su estructura se utilizará en la creación de algoritmos y para implementar el pseudocódigo. Su representación en diagrama de flujo es la siguiente:

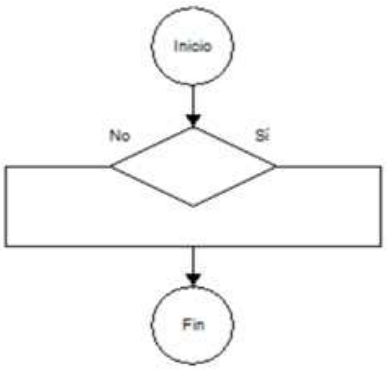


Figura 22. Diagrama de flujo Si-Entonces simple

Elaborado por Carlos Salazar

A continuación, observe el siguiente ejemplo utilizando la estructura SI-ENTONCES SIMPLE.

■ ENUNCIADO

Leer un número entero y determinar si es igual a 10.

ANÁLISIS

El primer paso consiste en identificar y entender el objetivo. Para ello es necesario analizar el enunciado para determinar la entrada, proceso y salida.

ENTRADA

- Número entero.

PROCESO

- Leer por teclado un número entero.
- Comparar si el número leído es igual a 10.
- Si es igual a 10, mostrar que el número es igual a 10.
- Si no, mostrar que el número no es igual a 10.

SALIDA

- Mostrar en pantalla si el número es o no igual a 10.

ALGORITMO

Algoritmo: Compara_numero

Inicio

1. Solicitar el ingreso por teclado de un número entero.
2. Guardar el número ingresado en la variable “num”.
3. Preguntar o comparar si “num” es igual a 10.
   Mostrar el mensaje: “El número ingresado es igual a 10.”
   Caso contrario
   Mostrar el mensaje: “El número ingresado no es igual a 10.”

Fin

Prueba de escritorio

Operaciones	num	pantalla
Solicitar el ingreso por teclado de un número entero.		
Guardar el número ingresado en la variable “num”.	10	
Preguntar o comparar si “num” es igual a 10. Mostrar el mensaje: “El número ingresado es igual a 10.” Caso contrario Mostrar el mensaje: “El número ingresado no es igual a 10.”		El número ingresado es igual a 10.

Diagrama de flujo

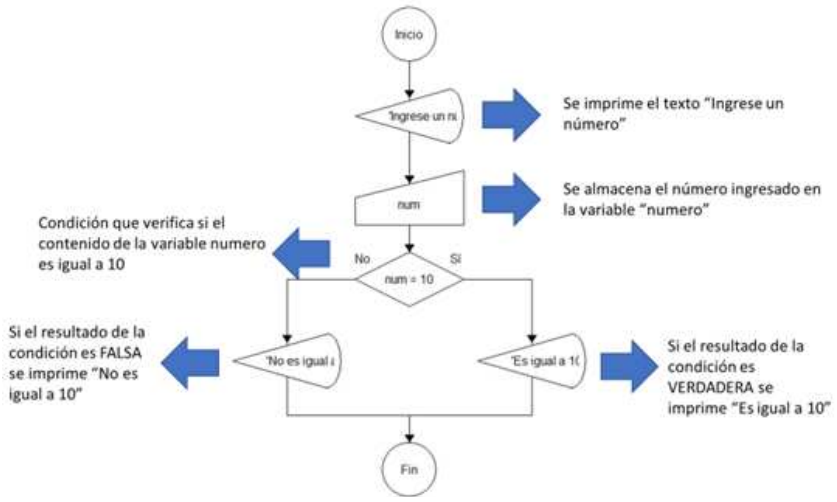


Figura 23. Ejemplo diagrama de flujo Si-Entonces simple

Elaborado por Carlos Salazar

Como se observa en la figura 23, se tienen dos caminos por seguir, según el resultado de la condición. Esto permite controlar el flujo del algoritmo de acuerdo con los resultados obtenidos. Al ejecutar el diagrama de flujo, la ejecución del algoritmo sería de la siguiente manera:

Figura 24. Resultado diagrama de flujo Si-Entonces simple

Elaborado por Carlos Salazar

En la figura 24, se puede observar el resultado del algoritmo en los dos escenarios posibles. Una vez que se ha verificado su funcionamiento de manera gráfica, se procede a realizar su implementación en pseudocódigo.

Pseudocódigo

```
Proceso Compara_numero
    Definir num Como Entero;

    Escribir "Ingrese un número";
    Leer num;
    Si num = 10 Entonces
        Escribir "Es igual a 10";
    Sino
        Escribir "No es igual a 10";
    FinSi
FinProceso
```

Figura 25. Pseudocódigo Si-Entonces simple

Elaborado por Carlos Salazar

Observe la figura 25: el condicional Si-Entonces simple trabaja en forma de bloque, en donde todas las acciones que contiene se encuentran dentro de la estructura con una tabulación. Esto permite mantener código limpio, fácil de entender, leer y mantener, criterios de suma importancia al escribir código.

Por esta razón es importante que se vaya familiarizando con estos pequeños pero útiles consejos durante su formación.

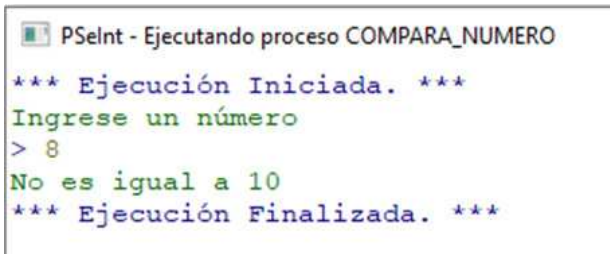


Figura 26. Ejecución de pseudocódigo Si-Entonces simple

Elaborado por Carlos Salazar

Similar al diagrama de flujo, se ha presentado el resultado de los dos escenarios (si es igual a 10 o si no es igual a 10), pero utilizando pseudocódigo. La estructura Si-Entonces simple es una de las más utilizadas a la hora de programar; sin embargo, se tienen otras estructuras que permiten tomar decisiones. El uso entre una u otra estructura dependerá del problema por resolver, las necesidades del algoritmo y sobre todo del proceso que se seguirá para alcanzar el objetivo. A continuación se exponen las diferentes estructuras de decisión disponibles.

2.2.2. Si-Entonces cascada

No es más que una variante a la estructura de decisión Si-Entonces, en donde, el Sino de cada Si condicional da inicio a un nuevo Si condicional N veces. Observe su estructura:

```
Si <expresion> Entonces
   <instrucciones si el resultado es verdadero>
Sino
        Si <expresion> Entonces
                <instrucciones si el resultado es verdadero>
        Sino
                Si <expresion> Entonces
                <instrucciones si el resultado es verdadero>
                FinSi
        FinSi
FinSi
```

Figura 27. Estructura Si-Entonces cascada

Elaborado por Carlos Salazar

En la estructura Si-Entonces cascada, a diferencia de la simple, es posible generar varios condicionales "Si" dentro de una condición "Sino". Generalmente esta estructura se utiliza cuando se desea reevaluar una expresión en caso de que no cumpla la primera.

Hay que tener mucho cuidado con la apertura "Si" y el cierre "FinSi" de cada condición. A continuación se muestra un ejemplo de su uso e implementación en un algoritmo y su representación en diagrama de flujo y pseudocódigo.

■ ENUNCIADO

Elabore un algoritmo que permita leer dos números enteros e indique cuál es el mayor o si son iguales.

ANÁLISIS

El primer paso consiste en identificar y entender el objetivo. Para ello es necesario analizar el enunciado para determinar la entrada, proceso y salida.

ENTRADA

- Dos números enteros.

PROCESO

- Leer por teclado un número entero.
- Almacenar en la variable "num1".
- Leer por teclado el segundo número.
- Almacenar en la variable "num2".
- Si num1 > num2
  Escribir "El primer número es mayor"
  Caso contrario
  Si num1 < num2
  Escribir "El segundo número es mayor"
  Caso contrario
  Escribir "Son iguales"

SALIDA

- Mostrar en pantalla cual número es el mayor o si son iguales.

ALGORITMO

Algoritmo: Numero_mayor

Inicio

1. Solicitar el ingreso de dos números enteros.
2. Almacenar el primer número en la variable "num1".
3. Almacenar el segundo número en la variable "num2".
4. Si num1 > num2
   Escribir "El primer número es mayor"
   Caso contrario
   Si num1 < num2
   Escribir "El segundo número es mayor"
   Caso contrario
   Escribir "Son iguales"

Fin

DIAGRAMA DE FLUJO

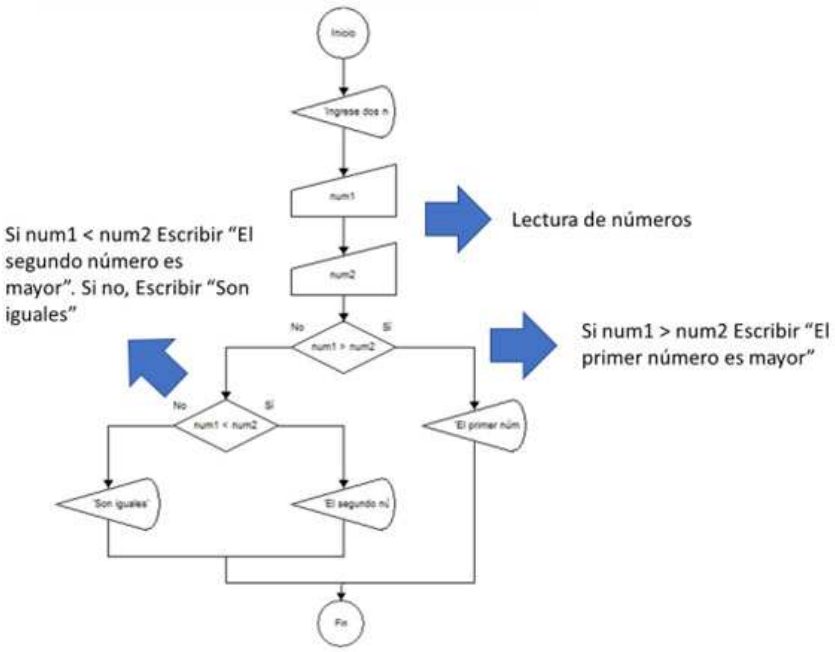


Figura 28. Diagrama de flujo Si-Entonces cascada

Elaborado por Carlos Salazar

Nótese en la figura 28 cómo entra una estructura condicional dentro de otra y esta acción se la puede repetir las veces que sea necesario. De esta manera los posibles caminos que puede tomar un proceso aumentan considerablemente dentro del flujo de un algoritmo.

Si en algún momento la condición no funciona como se espera, se debe verificar el contenido de las variables que se están comparando para asegurar que se está trabajando con los valores correctos.

Una vez diseñado el diagrama de flujo, se procede a ejecutarlo y se obtendrán los siguientes resultados en los tres posibles escenarios (el primer número es el mayor, el segundo número es el mayor o los dos números son iguales).

Figura 29. Resultado diagrama de flujo Si-Entonces cascada

Elaborado por Carlos Salazar

Cuando ya se tenga diseñado el algoritmo en diagrama de flujo y se ha verificado su funcionamiento, se procede a implementarlo en pseudocódigo.

PSEUDOCÓDIGO

```
1   Proceso Numero_mayor
2       Definir num1 Como Entero;
3       Definir num2 Como Entero;
4
5       Escribir "Ingrese dos números enteros";
6       Leer num1;
7       Leer num2;
8
9       Si num1 > num2 Entonces
10          Escribir "El primer número es mayor"
11      Sino
12          Si num1 < num2 Entonces
13              Escribir "El segundo número es mayor"
14          Sino
15              Escribir "Son iguales"
16          FinSi
17      FinSi
18
19  FinProceso
```

Figura 30. Pseudocódigo Si-Entonces cascada

Elaborado por Carlos Salazar

La figura 30 muestra cómo se utilizan dos condicionales en una sola estructura, dependiendo de los valores de las variables, en este caso "num1" y "num2". El flujo del algoritmo tomará un camino u otro. A continuación se muestra el resultado de su ejecución:

```
PSeInt - Ejecutando proceso NUMERO_MAYOR
*** Ejecución Iniciada. ***
Ingrese dos números enteros
> 10
> 8
El primer número es mayor
*** Ejecución Finalizada. ***
```

```
PSeInt - Ejecutando proceso NUMERO_MAYOR
*** Ejecución Iniciada. ***
Ingrese dos números enteros
> 8
> 10
El segundo número es mayor
*** Ejecución Finalizada. ***
```

```
PSeInt - Ejecutando proceso NUMERO_MAYOR
*** Ejecución Iniciada. ***
Ingrese dos números enteros
> 10
> 10
Son iguales
*** Ejecución Finalizada. ***
```

Figura 31. Resultado pseudocódigo Si-Entonces cascada

Elaborado por Carlos Salazar

2.2.3. Si-Entonces secuencia

Esta estructura generalmente se utiliza cuando se deben realizar varias preguntas en donde no es importante el Sino de cada decisión. Se puede o no incluir un Sino para cada Si. Su estructura es la siguiente:

```
Si <expresion> Entonces
   <instrucciones si el resultado es verdadero>
FinSi
Si <expresion> Entonces
   <instrucciones si el resultado es verdadero>
FinSi
Si <expresion> Entonces
   <instrucciones si el resultado es verdadero>
FinSi
```

Figura 32. Estructura si-entonces secuencia

Elaborado por Carlos Salazar

Como puede apreciar en la figura 32, la estructura Si-Entonces secuencia, a diferencia de la estructura en cascada, da fin a una condición "Si" y simultáneamente empieza otra condición. Se utiliza esta estructura cuando se requiere realizar varias condiciones sin importar si se cumple o no la anterior. Observe su uso en el siguiente ejemplo:

■ ENUNCIADO

Crear un algoritmo que permita ingresar un número entero por teclado, y si el número es menor a 20, que imprima un mensaje indicando que es menor a 20; si es mayor a 5, que se imprima un mensaje indicando que es mayor a 5, y si es mayor a cero que imprima un mensaje indicando que el número es positivo.

ANÁLISIS

El primer paso consiste en identificar y entender el objetivo. Para ello es necesario analizar el enunciado para determinar la entrada, proceso y salida.

ENTRADA

- Número entero

PROCESO

- Leer por teclado un número entero.
- Almacenar el número ingresado en la variable "num"
- Si num < 20
  Escribir "El número ingresado es menor a 20"
  Si num > 5
  Escribir "El número ingresado es mayor a 5"
  Si num > 0
  Escribir "El número ingresado es positivo"

SALIDA

- Mostrar en pantalla si el número es menor a 20, mayor a 5 o si es positivo.

Algoritmo

Algoritmo: Numero_comparar

Inicio

1. Solicitar el ingreso de un número entero.
2. Almacenar el número en la variable “num”
3. Si num <20
   Escribir “El número ingresado es menor a 20”
   Si num > 5
   Escribir “El número ingresado es mayor a 5”
   Si num > 0
   Escribir “El número ingresado es positivo”

Fin

DIAGRAMA DE FLUJO

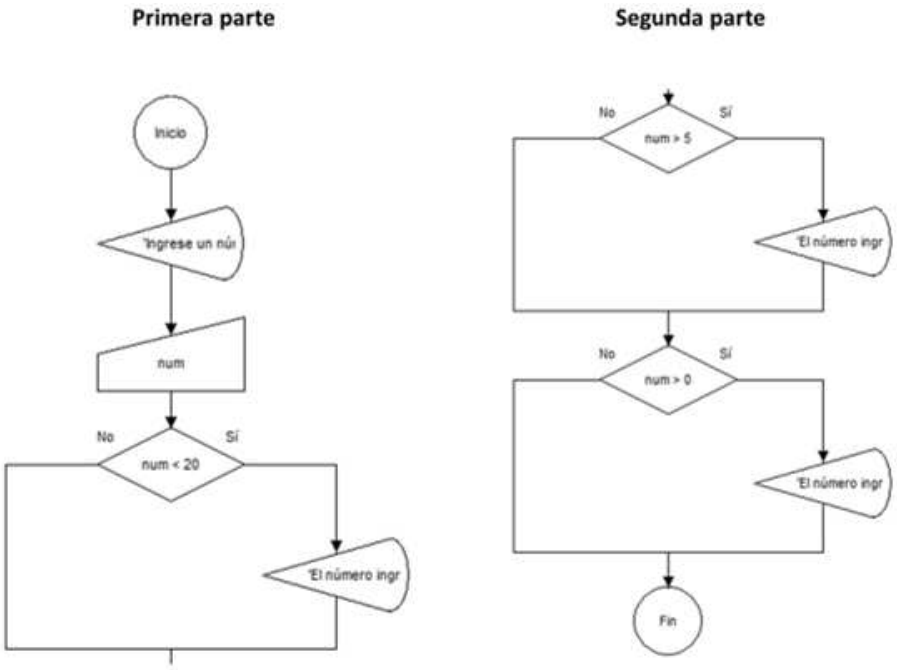


Figura 33. Diagrama de flujo Si-Entonces secuencia

Elaborado por Carlos Salazar

Nótese que, en esta estructura de decisión, el condicional “Si” va a continuación de otro. De esta manera, el proceso tiene que pasar por las tres condiciones establecidas (comparar si es menor a 20; comparar si es mayor a 5, y comprobar si es mayor a 0). Recuerde que en esta estructura se puede tener los condicionales “Si” que sean necesarios para logar un objetivo. Al ejecutar el algoritmo obtendrá un resultado como el que se muestra a continuación:

Figura 34. Resultado diagrama de flujo Si-Entonces secuencia

Elaborado por Carlos Salazar

PSEUDOCÓDIGO

```
Proceso Numero_comparar
    Definir num Como Entero;

    Escribir "Ingrese un número entero";
    Leer num;

    Si num < 20 Entonces
        Escribir "El número ingresado es menor a 20"
    FinSi
    Si num > 5 Entonces
        Escribir "El número ingresado es mayor a 5"
    FinSi
    Si num > 0 Entonces
        Escribir "El número ingresado es positivo"
    FinSi

FinProceso
```

Figura 35. Resultado pseudocódigo Si-Entonces secuencia

Elaborado por Carlos Salazar

Al ejecutar el algoritmo se obtendrá el siguiente resultado:

```
PSeInt - Ejecutando proceso NUMERO_COMPARAR
*** Ejecución Iniciada. ***
Ingrese un número entero
> 10
El número ingresado es menor a 20
El número ingresado es mayor a 5
El número ingresado es positivo
*** Ejecución Finalizada. ***
```

Figura 36. Resultado pseudocódigo Si-Entonces secuencia

Elaborado por Carlos Salazar

Hay que mencionar que en la estructura Si-Entonces secuencia, la condición "Si", puede o no tener un "Sino". Para finalizar, se estudiará la estructura Si-Entonces anidado.

2.2.4. Si-Entonces anidado

Su principal característica es que un condicional "Si" se encuentra dentro de otro condicional "Si", provocando que se deba cumplir más de una condición en el flujo de un algoritmo. Su estructura es la siguiente:

```
Si <expresion> Entonces
    Si <expresion> Entonces
        <instrucciones si el resultado es verdadero>
    Sino
        <instrucciones si el resultado es falso>
    FinSI
Sino <expresion> Entonces
    <instrucciones si el resultado es falso>
FinSi
```

Figura 37. Estructura Si-Entonces anidado
Elaborado por Carlos Salazar

Analice el siguiente ejemplo:

■ ENUNCIADO

Crear un algoritmo que permita ingresar un número entero por teclado. Se debe verificar si el número es positivo, y si es menor a 100, si cumple las condiciones, se debe mostrar un mensaje en pantalla indicando que el número cumple los requerimientos.

ANÁLISIS

El primer paso consiste en identificar y entender el objetivo. Para ello es necesario analizar el enunciado para determinar la entrada, proceso y salida.

ENTRADA

- Número entero.

PROCESO

- Leer por teclado un número entero.
- Almacenar el número ingresado en la variable "num"
- Si num > 0
- Si num < 100
  Escribir "El número cumple los requerimientos".

SALIDA

- Mostrar en pantalla si el número cumple o no los requerimientos.

ALGORITMO

Algoritmo: Numero_requerimientos

Inicio

1. Solicitar el ingreso de un número entero.
2. Almacenar el número en la variable “num”.
3. Si num > 0
   Si num < 100
   Escribir “El número cumple los requerimientos”.
   Sino
   Escribir “El número no cumple los requerimientos”.
   Sino
   Escribir “El número no cumple los requerimientos”.

Fin

DIAGRAMA DE FLUJO

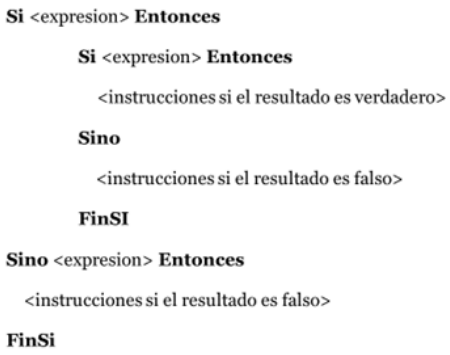


Figura 38. Diagrama de flujo Si-Entonces anidado

Elaborado por Carlos Salazar

Como puede observar en la figura 38, esta estructura es muy similar al condicional Si-Entonces cascada. La diferencia radica en el condicional “Si”, ya que este se encuentra dentro de otro condicional “Si”, valga la redundancia. Al ejecutar el diagrama de flujo obtendrá el siguiente resultado:

Figura 39. Resultado diagrama de flujo Si-Entonces anidado

Elaborado por Carlos Salazar

Para que un número cumpla los requerimientos, debe cumplir con las dos condiciones que se han planteado en el diagrama de flujo. A continuación observe su implementación en pseudocódigo:

PSEUDOCÓDIGO

```
Proceso Numero_requerimientos
    Definir num Como Entero;

    Escribir "Ingrese un número entero";
    Leer num;
    Si num > 0 Entonces
        Si num < 100 Entonces
            Escribir "Cumple los requerimientos";
        Sino
            Escribir "No cumple los requerimientos"
        FinSi
    Sino
        Escribir "No cumple los requerimientos"
    FinSi
FinProceso
```

Figura 40. Pseudocódigo Si-Entonces anidado

Elaborado por Carlos Salazar

Al ejecutar el algoritmo, se obtiene el siguiente resultado:

Figura 41. Resultado pseudocódigo Si-Entonces anidado

```
PSeInt - Ejecutando proceso NUMERO_REQUERIMIENTOS
*** Ejecución Iniciada. ***
Ingrese un número entero
> 10
Cumple los requerimientos
*** Ejecución Finalizada. ***
```

Elaborado por Carlos Salazar

Como puede ver, todas las variantes de la estructura de decisión tienen algo en común: se debe cumplir una condición para realizar una serie de acciones; en caso contrario, se ejecutarán acciones determinadas para controlar el flujo del algoritmo y lograr el objetivo.

A continuación se muestra una serie de ejercicios propuestos para que el estudiante fortalezca y desarrolle la habilidad de resolver y representar algoritmos utilizando estructura secuencial y decisión.

2.2.5. Ejercicios

Se recomienda resolver los siguientes ejercicios identificando el algoritmo con su respectivo análisis (entrada, proceso, salida), diagrama de flujo y pseudocódigo.

1. Lea un número entero y determine si es igual a 10.
2. Lea un número entero y determine si es negativo.
3. Lea dos números enteros y determine, ¿cuál es el mayor?
4. Lea dos números enteros y determine, ¿cuál es el menor?
5. Lea dos números enteros; súmelos y determine si el resultado es mayor a 10.
6. Lea dos números enteros, si el resultado de sumarlos es menor a 5 imprimir la suma, caso contrario imprimir la multiplicación.
7. Lea un número entero y determine si es número negativo o positivo, si es mayor a 10 y si es menor a 100.
8. Lea dos números enteros y realice la división garantizando que los valores ingresados sean mayores a cero.
9. Elabore un algoritmo en el que entre el nombre de un empleado, su salario básico por hora y el número de horas trabajadas en el mes; escriba su nombre y salario mensual si éste es mayor a $400; de lo contrario escriba solo el nombre.
10. Elabore un algoritmo que solicite el ingreso de la edad de una persona y determine si es mayor o menor de edad. Es mayor de edad cuando pasa los 17 años.
11. Elabore un algoritmo que permita leer el peso de tres cajas (A, B y C) y muestre en pantalla cuál es la caja de mayor peso.
12. Una librería hace los siguientes descuentos: si el cliente compra menos de 5 libros se le da un descuento del 10% sobre la compra; si el número de libros es mayor o igual a cinco, pero menos de 10, se le otorga un 20%, y si son 10 o más se le da un 40%. Haga un algoritmo que determine cuánto debe pagar un cliente si el valor de cada libro es de $80.
13. Desarrolle un algoritmo que solicite el ingreso de tres calificaciones, obtenga el promedio final y si el resultado es mayor o igual a 7, que aparezca un mensaje de Aprobado; en caso contrario, que aparezca un mensaje de Reprobado.
14. Elabore un algoritmo que nos diga el porcentaje de descuento aplicado a los comerciales de una empresa, dependiendo de la categoría profesional:
    - Categoría A: 2%
    - Categoría B: 8%
    - Categoría C: 12%
    - Se debe ingresar la categoría y se debe mostrar el porcentaje que le corresponde.

15. Una frutería ofrece las manzanas con descuento según la siguiente tabla:

Número de manzanas compradas	% Descuento
0 - 4	0%
5 - 8	10%
9 - 12	15%
13 en adelante	20%

    - Determinar cuánto pagará una persona que compre X manzanas en esa frutería.

16. Cierta universidad tiene un programa para estimular a los estudiantes con buen rendimiento académico:

    - Si el promedio es de 9,5 o más y el alumno es de pregrado, entonces cursará 30 créditos y se le hará un 25% de descuento.
    - Si el promedio es mayor o igual a 7,0 pero menor que 9,5 y el alumno es de pregrado, entonces cursará 25 créditos y se le hará un 10% de descuento.
    - Si el promedio es mayor que 6,0 y menor que 7,0 y es de pregrado, cursará 20 créditos y no tendrá ningún descuento.
    - Si el promedio es mayor o igual a 5,0 y menor que 6,0 y es de pregrado, cursará 15 créditos y no tendrá descuento.
    - Si el promedio es menor de 5,0 y es de pregrado, no podrá matricularse.
    - Si el promedio es mayor o igual a 8,5 y es de posgrado, cursará 20 créditos y se le hará un 20 % de descuento.
    - Si el promedio es menor de 8,5 y es de posgrado cursará 15 créditos y no tendrá descuento.
    - Haga un algoritmo que determine cuánto debe pagar un estudiante y cuántos créditos registra si el valor de cada crédito es de $500 para pregrado y $300 para posgrado.

17. Un proveedor de repuestos de vehículos ofrece un descuento del 10% si la compra es de $1.000 o más. Además, independientemente, ofrece el 5% de descuento si la marca es Toyota. Determine cuánto pagará, con IVA incluido, un cliente por la compra de repuestos para su vehículo.

18. Determine el precio de un pasaje de ida y vuelta por avión, conociendo la distancia por recorrer, el número de días de estadía y sabiendo que:

    - Si la distancia es superior a 500 km, y el número de días de estadía es superior a 5, la línea aérea le hace un descuento del 15 %. El precio por kilómetro es de $100.

19. En una escuela se requiere implementar un algoritmo que permita convertir calificaciones numéricas, según la siguiente descripción:

    - A = 19 y 20, B = 16, 17 y 18, C = 13, 14 y 15, D = 10, 11 y 12, E = 1 hasta 9. Se asume que la calificación está comprendida entre 1 y 20.

20. Escriba un algoritmo que lea la temperatura y muestre por pantalla la actividad más apropiada para dicha temperatura teniendo en cuenta los siguientes criterios:

- Temperatura > 30 (Natación)
- Temperatura < 20 hasta < = 30 (Fútbol)
- Temperatura < 10 hasta < = 20 (Ciclismo)
- Temperatura < 5 hasta < = 10 (Trote)
- Temperatura < = 5 (Descanso)

2.6. Selección múltiple

También conocida como estructura de caso, se caracteriza por contener un grupo de instrucciones pertenecientes a un conjunto de alternativas posibles. A diferencia de las estructuras de decisión Si-Entonces, no utiliza dentro de sus estructuras expresiones de condición, sino el valor almacenado en una variable, logrando así ejecutar aquellas instrucciones que coinciden en sus casos con el valor almacenado en dicha variable. Observe en la siguiente figura su estructura:

```
Segun variable Hacer
        opcion_1:
                secuencia_de_acciones_1
        opcion_2:
                secuencia_de_acciones_2
        opcion_3:
                secuencia_de_acciones_3
        De Otro Modo:
                secuencia_de_acciones
Fin Segun
```

Figura 42. Estructura selección múltiple

Elaborado por Carlos Salazar

La estructura de selección múltiple es muy utilizada a la hora de programar, muchos lenguajes de programación la utilizan. Sin embargo, el simulador DFD no dispone de una simbología para representar esta estructura, y por esta razón únicamente se la va a representar en pseudocódigo. A continuación, observe el siguiente ejercicio:

■ ENUNCIADO

Realizar un algoritmo que permita ingresar el número de día del 1 al 7 e imprima a qué día de la semana pertenece.

ANÁLISIS

El primer paso consiste en identificar y entender el objetivo. Para ello es necesario analizar el enunciado para determinar la entrada, proceso y salida.

ENTRADA

- Número entero.

PROCESO

- Leer por teclado un número entero.
- Verificar si coindice con el número 1, si es así, imprimir lunes.
- Verificar si coindice con el número 2, si es así, imprimir martes.

- Verificar si coindice con el número 3, si es así, imprimir miércoles.
- Verificar si coindice con el número 4, si es así, imprimir jueves.
- Verificar si coindice con el número 5, si es así, imprimir viernes.
- Verificar si coindice con el número 6, si es así, imprimir sábado.
- Verificar si coindice con el número 7, si es así, imprimir domingo.

SALIDA

- Mostrar en pantalla el día de la semana al que corresponde el número ingresado.

ALGORITMO

Algoritmo: dia-semana

Inicio

1. Leer por teclado un número entero
2. Verificar si coindice con el número 1, si es así, imprimir lunes
3. Verificar si coindice con el número 2, si es así, imprimir martes
4. Verificar si coindice con el número 3, si es así, imprimir miércoles
5. Verificar si coindice con el número 4, si es así, imprimir jueves
6. Verificar si coindice con el número 5, si es así, imprimir viernes
7. Verificar si coindice con el número 6, si es así, imprimir sábado
8. Verificar si coindice con el número 7, si es así, imprimir domingo

Fin

Fíjese en el desarrollo del algoritmo: se podría resolver este ejercicio haciendo uso de 7 estructuras de decisión Si-Entonces. Cuando se tiene este escenario, en donde se desea comparar una variable con un determinado valor, resulta muy eficiente hacer uso de la estructura selectiva. Observe su implementación en pseudocódigo:

PSEUDOCÓDIGO

```
Proceso dia_de_la_semana
    Definir dia Como Entero
    Leer dia
    Segun dia Hacer
        1:
            Escribir "Lunes"
        2:
            Escribir "Martes"
        3:
            Escribir "Miercoles"
        4:
            Escribir "Jueves"
        5:
            Escribir "Viernes"
        6:
            Escribir "Sábado"
        7:
            Escribir "Domingo"
        De Otro Modo:
            Escribir "No pertenece a un dia de la semana"
        Fin Segun
FinProceso
```

Figura 43. Pseudocódigo selección múltiple

Elaborado por Carlos Salazar

Como puede notar en la figura 43, la estructura selectiva realiza una comparación de igualdad en base de una variable, en este caso "dia". Si el contenido de dicha variable coincide con algún valor por comparar, se ejecuta la secuencia de acciones que contiene dicho valor; esta estructura funciona con datos numéricos y caracteres. En el caso de no coincidir con algún valor dentro de la estructura, por defecto se ejecutarán las instrucciones que se encuentran en la sección "De otro modo".

Cuando se tiene escenarios de comparación por igualdad, es mucho mejor trabajar con una estructura de selección múltiple, no solo por la simplicidad en su implementación sino también para mantener un código limpio y de fácil mantenimiento.

A continuación se muestra una serie de ejercicios propuestos para que el estudiante fortalezca y desarrolle la habilidad de resolver y representar algoritmos utilizando estructuras de selección múltiple.

2.2.1. Ejercicios

Se recomienda resolver los siguientes ejercicios identificando el algoritmo con su respectivo análisis (entrada, proceso, salida) y pseudocódigo.

1. Escriba un algoritmo que lea una calificación de la A hasta la letra E y emita un mensaje correspondiente a la nota ingresada:
   - A – Excelente
   - B – Buena
   - C – Regular
   - D – Suficiente
   - E – No Suficiente
   - Si la calificación ingresada no corresponde a la clasificación anterior, se debe imprimir el siguiente mensaje: "Calificación errónea".
2. Escriba un algoritmo que lea un número entero del 1 al 12 y determine a qué mes hace referencia
   - 1 – Enero
   - 2 – Febrero
   - 3 – Marzo …
   - Se debe validar que el número ingresado no sea menor a 0 y tampoco mayor a 12.
3. Una familia ecuatoriana desea viajar en el verano, decide dejar a la suerte su destino turístico, de modo que preparan cinco papeles en una bolsa, numerados del 1 al 5. Si sacan de la bolsa el 1, visitan Atacames; si sacan el 2, van a Baños; si sacan el 3, visitan Cuenca; si sacan 4, van a Galápagos; y si sacan 5, visitan Manta. Realice un algoritmo en pseudocódigo que permita representar el sorteo.
4. Elabore un algoritmo que permita calcular e imprimir el promedio final de un estudiante cuyas calificaciones se realizan con letras en donde: A vale 4 puntos, B vale 3 puntos y C equivale a 2 puntos. De esta manera, si el estudiante obtuvo una A en un curso de cinco créditos, es 4*5 = 20; si obtuvo una C en un curso de

cuatro créditos, es 2*4 = 8; y si obtuvo una B en un curso de tres créditos sería 3 *3 = 9. Se debe ingresar la calificación en letras y determinar el puntaje obtenido. Además, si la calificación ingresada no corresponde a A, B o C, se debe imprimir un mensaje que diga: "No se reconoce la calificación".

5. Elabore un algoritmo que lea dos números y un operador aritmético (+, -, *, /). Se debe realizar la operación ingresada con los números leídos y mostrar en pantalla el resultado.

2.7. Los ciclos

En secciones anteriores se habló sobre las estructuras que rigen el pensamiento humano, entre ellas: secuencia, decisión y ciclos. En este apartado se abordará la última estructura también conocida como estructura repetitiva.

Hasta el momento se ha trabajado con algoritmos en donde sus instrucciones o acciones se ejecutan una, y sólo una vez de manera secuencial. Los ciclos permiten ejecutar estas acciones las veces que sea necesario siempre y cuando se cumpla una condición.

En los ciclos, iteraciones o bucles, lo principal es identificar las acciones que se van a repetir y hasta cuándo o cuántas veces se van a repetir dichas instrucciones; esto dependerá del tipo de bucle que se utilizará. A continuación se abordan las diferentes estructuras de repetición.

2.7.1. Variable Contador

Los contadores se definen como variables de tipo entero y se usan en estructuras repetitivas, generalmente para conocer el número de veces que se repite un procedimiento.

El conteo se hace con una variable que se incrementa o disminuye en un valor constante. Su formato es el siguiente:

```
Contador = Contador + ó – valor_constante
```

Figura 44. Estructura variable contador

Elaborado por Carlos Salazar

2.7.2. Variable Acumulador

Un acumulador o totalizador es una variable cuya misión es almacenar cantidades resultantes de procesos sucesivos. Su formato es el siguiente:

```
Acumulador = Acumulador + ó - expresión
```

Figura 45. Formato variable acumulador

Elaborado por Carlos Salazar

2.7.3. Estructura – Mientras

Esta estructura permite controlar la ejecución de una o más instrucciones (secuencia), en base del valor de verdad que arroje la evaluación de una expresión de tipo lógico. Dicha evaluación determinará el número de veces que se ejecutarán las acciones que se encuentren dentro de esta estructura. Su estructura es la siguiente:

```
Mientras <expresión lógica> Hacer
   <instrucciones si el resultado es verdadero>
FinMientras
```

Figura 46. Estructura Mientras

Elaborado por Carlos Salazar

Como puede observar en la figura 46, la estructura Mientras está formada por dos partes:

- Expresión de tipo lógico.
- Conjunto de instrucciones.

La expresión lógica será evaluada cada vez que se intente repetir un proceso; esto determinará si se da paso o no a su repetición. En cuanto al conjunto de instrucciones, se debe procurar que en algún momento cambie el valor de la expresión lógica; en caso contrario, se obtendrá como resultado un bucle infinito.

Su representación en DFD se la realiza de la siguiente manera:

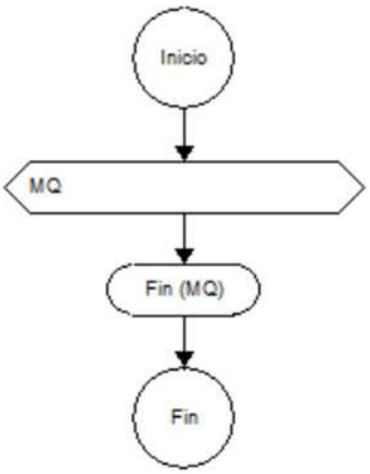


Figura 47. Diagrama de flujo estructura Mientras

Elaborado por Carlos Salazar

A continuación, observe el siguiente ejemplo utilizando la estructura de repetición Mientras.

■ ENUNCIADO

Realizar un algoritmo que permita mostrar en pantalla los números del 1 al 10.

ANÁLISIS

El primer paso consiste en identificar y entender el objetivo. Para ello es necesario analizar el enunciado para determinar la entrada, proceso y salida.

ENTRADA

- Sin datos de entrada

PROCESO

- Utilizar una variable "num" con el valor de 1
- Mientras "num" sea menor o igual a 10
  Imprimir "num"
  Se hace un incremento de uno a la variable "num"

SALIDA

- Mostrar en pantalla los números del 1 al 10

ALGORITMO

Algoritmo: imprimir_numeros

Inicio

1. Definir la variable "num" con el valor de 1
2. Mientras "num" < = 10
   Imprimir "num"
   "num" = "num" + 1

Fin

Como puede observar en la etapa de análisis y en la creación del algoritmo, se utiliza la estructura Mientras para representar las acciones que se van a repetir. Además, nótese en la línea cuatro el uso de una variable que funciona como contador debido a que realiza el incremento de un valor constante, en este caso de 1 en cada iteración.

En la siguiente figura puede observar su implementación utilizando la representación de la estructura Mientras en diagrama de flujo en el programa DFD.

DIAGRAMA DE FLUJO

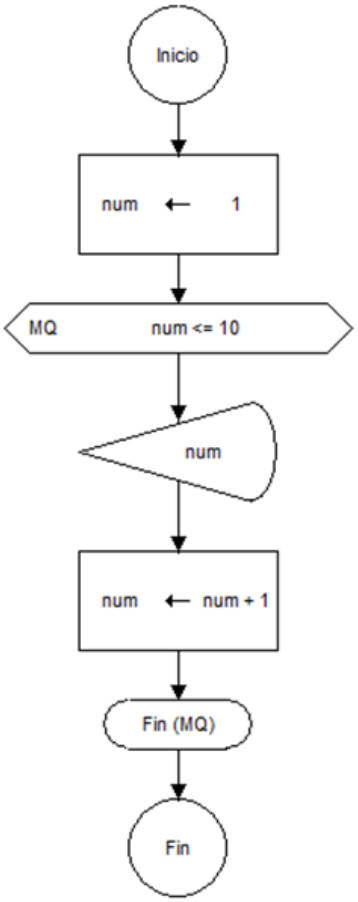


Figura 48. Diagrama de flujo estructura Mientras

Elaborado por Carlos Salazar

Como puede notar en la figura 48, las instrucciones de impresión de la variable "numero" y el incremento se encuentran dentro de la estructura Mientras. De esta manera, en un determinado momento, la condición emitirá como resultado el valor de FALSO y terminará el ciclo repetitivo.

PSEUDOCÓDIGO

```
Proceso imprimir_numeros
    Definir num Como Entero;
    num = 1;
    Mientras num <= 10 Hacer
        Imprimir num;
        num = num + 1;
    Fin Mientras
FinProceso
```

Figura 49. Pseudocódigo estructura Mientras

Elaborado por Carlos Salazar

Nótese en la figura 49 que la estructura Mientras tiene un inicio y un final. Además, todas las instrucciones se encuentran dentro de este bloque. Al ejecutar el algoritmo, se obtendrá el siguiente resultado:

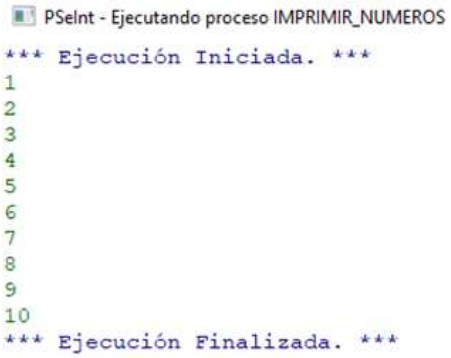


Figura 50. Resultado estructura Mientras

Elaborado por Carlos Salazar

■ ENUNCIADO

Crear un algoritmo que capture un número y realice la tabla de multiplicar del 1 al 10 de dicho número.

ALGORITMO

Algoritmo: tabla_multiplicar

Inicio

1. Leer un número entero y guardar en la variable "num"
2. Crear un contador que inicie en 1
3. Mientras el contador sea menor o igual a 10
   Escribir num "*" contador = num * contador
   contador = contador +1

Fin

DIAGRAMA DE FLUJO

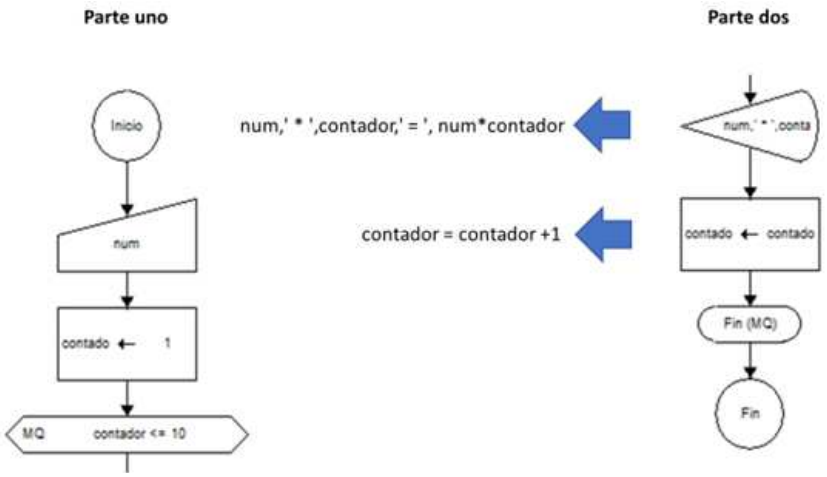


Figura 51. Diagrama de flujo tabla de multiplicar estructura Mientras

Elaborado por Carlos Salazar

PSEUDOCÓDIGO

```
Proceso tabla_multiplicar
    Definir num,contador Como Entero;
    contador = 1;
    Escribir "Ingrese un número";
    Leer num;
    Mientras contador <= 10
        Escribir num," * ",contador," = ",num*contador;
        contador = contador + 1;
    FinMientras
FinProceso
```

Figura 52. Pseudocódigo tabla de multiplicar estructura Mientras

Elaborado por Carlos Salazar

Al ejecutar el algoritmo de la figura 52, obtendrá el siguiente resultado:

```
PSeInt - Ejecutando proceso TABLA_MULTIPLICAR
*** Ejecución Iniciada. ***
Ingrese un número
> 10
10 * 1 = 10
10 * 2 = 20
10 * 3 = 30
10 * 4 = 40
10 * 5 = 50
10 * 6 = 60
10 * 7 = 70
10 * 8 = 80
10 * 9 = 90
10 * 10 = 100
*** Ejecución Finalizada. ***
```

Figura 53. Resultado pseudocódigo tabla de multiplicar

Elaborado por Carlos Salazar

2.7.4. Estructura Repetir

A diferencia de la estructura Mientras, esta condición primero realiza las acciones y después se comprueba la expresión lógica. Por lo tanto, las acciones se ejecutan al menos una vez. Su estructura es la siguiente:

```
Repetir
        <secuencia_de_acciones>
Hasta Que <expresión_lógica>
```

Figura 54. Estructura Repetir

Elaborado por Carlos Salazar

Como puede observar en la figura 54, la estructura está formada por dos partes:

- Conjunto de instrucciones.
- Expresión lógica.

Al igual que la estructura de selección múltiple, no es posible representarla en DFD, por esta razón solo se implementará en PseInt. La mayoría de lenguajes de programación sí la incorporan en sus estructuras de repetición. Observe su uso en el siguiente ejemplo:

■ ENUNCIADO

Realizar un algoritmo que permita mostrar en pantalla los números del 1 al 10.

ANÁLISIS

El primer paso consiste en identificar y entender el objetivo. Para ello es necesario analizar el enunciado para determinar la entrada, proceso y salida.

ENTRADA

- Sin datos de entrada

PROCESO

- Utilizar una variable “num” con el valor de 1
- Imprimir “num”
- Se hace un incremento de uno a la variable “num”
- Repetir hasta que “num” sea menor a 10

SALIDA

- Mostrar en pantalla los números del 1 al 10

ALGORITMO

Algoritmo: imprimir_numeros

Inicio

1. Definir la variable “num” con el valor de 1
2. Hacer
   Imprimir “num”
   “num” = “num” + 1
   Repetir hasta que “num” > 10

Fin

PSEUDOCÓDIGO

```
Proceso imprimir_numeros
    Definir num Como Entero;
    num = 1;
    Repetir
        Imprimir num;
        num = num + 1;
    Hasta Que num > 10
FinProceso
```

Figura 55. Ejercicio estructura Repetir

Elaborado por Carlos Salazar

Al ejecutar el algoritmo de la figura 55, obtendrá el siguiente resultado:

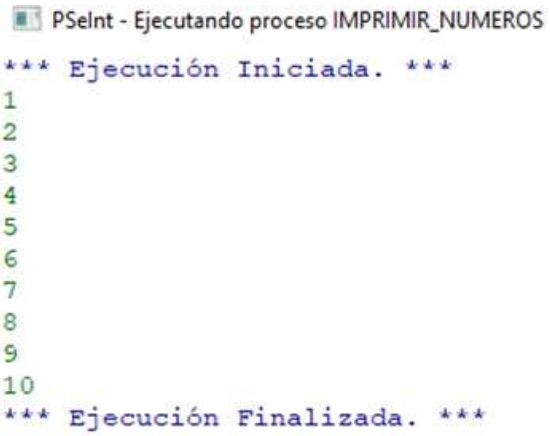


Figura 56. Ejercicio estructura Repetir

Elaborado por Carlos Salazar

■ ENUNCIADO

Crear un algoritmo que capture un número y realice la tabla de multiplicar del 1 al 10 de dicho número.

ALGORITMO

Algoritmo: tabla_multiplicar

Inicio

1. Leer un número entero y guardar en la variable "num"
2. Crear un contador que inicie en 1
3. Hacer
   Escribir num "*" contador = num * contador
   contador = contador +1
   Repetir hasta que contador sea mayor a 10

Fin

PSEUDOCÓDIGO

```
Proceso tabla_multiplicar
    Definir num,contador Como Entero;
    Leer num;
    contador = 1;
    Repetir
        Escribir num," * ",contador," = ",num*contador;
        contador = contador + 1;
    Hasta Que contador > 10
FinProceso
```

Figura 57. Pseudocódigo tabla de multiplicar estructura Repetir

Elaborado por Carlos Salazar

Al ejecutar el algoritmo de la figura 57, obtendrá el siguiente resultado:

```
PSeInt - Ejecutando proceso TABLA_MULTIPLICAR
*** Ejecución Iniciada. ***
> 10
10 * 1 = 10
10 * 2 = 20
10 * 3 = 30
10 * 4 = 40
10 * 5 = 50
10 * 6 = 60
10 * 7 = 70
10 * 8 = 80
10 * 9 = 90
10 * 10 = 100
*** Ejecución Finalizada. ***
```

Figura 58. Resultado pseudocódigo tabla de multiplicar estructura Repetir

Elaborado por Carlos Salazar

2.7.5. Estructura Para

La estructura de repetición Para permite implementar un cierto conjunto de instrucciones un número predeterminado de veces. Para ello, utiliza una variable de control definiendo el valor de inicio, valor final y el incremento o decremento según sea el caso. Observe a continuación su estructura:

```
Para <variable_numérica = valor_inicial> Hasta <valor_final> Con Paso <paso> Hacer
        secuencia_de_acciones
Fin Para
```

Figura 59. Estructura Para

Elaborado por Carlos Salazar

En donde:

- **Para.** Indica el nombre de la variable de control y su valor inicial.
- **Hasta.** Indica el valor que tomará la variable de control para terminar el ciclo repetitivo.
- **Con Paso.** Se establece el incremento que tendrá la variable de control y cada iteración.

Observe el siguiente ejemplo:

■ ENUNCIADO

Realizar un algoritmo que permita mostrar en pantalla los números del 1 al 10.

ANÁLISIS

El primer paso consiste en identificar y entender el objetivo. Para ello es necesario analizar el enunciado para determinar la entrada, proceso y salida.

ENTRADA

- Sin datos de entrada

PROCESO

- Utilizar una variable “num” con el valor de 1
- Repetir hasta que num sea igual a 10 con un incremento de uno en cada iteración
- Imprimir “num”

SALIDA

- Mostrar en pantalla los números del 1 al 10

ALGORITMO

Algoritmo: imprimir_numeros

Inicio

1. Definir la variable “num” con el valor de 1
2. Repetir hasta que “num” = 10 con un incremento de 1 en cada iteración
3. Imprimir “num”

Fin

DIAGRAMA DE FLUJO

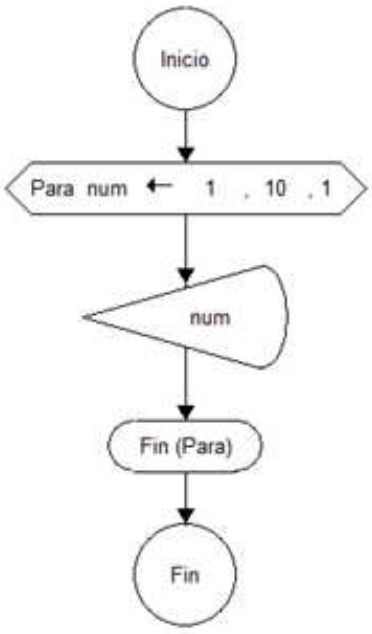


Figura 60. Diagrama de flujo estructura Para

Elaborado por Carlos Salazar

Nótese en la figura 60 las condiciones: el valor inicial, fin e incremento se encuentran explícitas en la estructura y no en las instrucciones. A continuación, observe su implementación en pseudocódigo:

PSEUDOCÓDIGO

```
Proceso imprimir_numeros
    Definir num Como Entero;
    Para num = 1 Hasta 10 Con Paso 1 Hacer
        Escribir num
    Fin Para
FinProceso
```

Figura 61. Pseudocódigo estructura Para

Elaborado por Carlos Salazar

Al ejecutar el algoritmo, obtendrá el siguiente resultado:

```
PSeInt - Ejecutando proceso IMPRIMIR_NUMEROS
*** Ejecución Iniciada. ***
1
2
3
4
5
6
7
8
9
10
*** Ejecución Finalizada. ***
```

Figura 62. Resultado pseudocódigo estructura Para

Elaborado por Carlos Salazar

■ ENUNCIADO

Crear un algoritmo que capture un número y realice la tabla de multiplicar del 1 al 10 de dicho número.

ALGORITMO

Algoritmo: tabla_multiplicar

Inicio

1. Leer un número entero y guardar en la variable “num”
2. Crear un contador que inicie en 1
3. Repetir hasta que “contador” sea igual a 10 y cada iteración se aumenta su valor en uno
   Escribir num “*” contador = num * contador

Fin

DIAGRAMA DE FLUJO

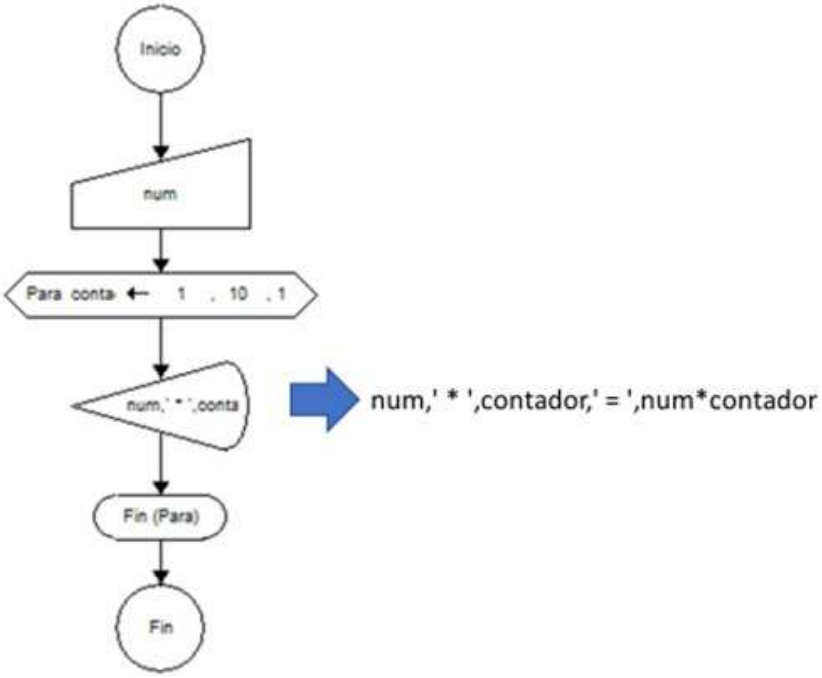


Figura 63. Diagrama de flujo tabla de multiplicar estructura Para

Elaborado por Carlos Salazar

PSEUDOCÓDIGO

```
Proceso tabla_multiplicar
    Definir num,contador Como Entero;
    Leer num;
    Para contador = 1 Hasta 10 Con Paso 1 Hacer
        Escribir num," * ",contador," = ",num*contador;
    FinPara
FinProceso
```

Figura 64. Pseudocódigo tabla de multiplicar estructura Para

Elaborado por Carlos Salazar

Al ejecutar el algoritmo de la figura 64, se obtiene el siguiente resultado en pantalla:

```
PSeInt - Ejecutando proceso TABLA_MULTIPLICAR
*** Ejecución Iniciada. ***
> 10
10 * 1 = 10
10 * 2 = 20
10 * 3 = 30
10 * 4 = 40
10 * 5 = 50
10 * 6 = 60
10 * 7 = 70
10 * 8 = 80
10 * 9 = 90
10 * 10 = 100
*** Ejecución Finalizada. ***
```

Figura 65. Resultado pseudocódigo tabla de multiplicar estructura Para

Elaborado por Carlos Salazar

A continuación, se propone una serie de ejercicios para que el estudiante fortalezca y desarrolle la habilidad de resolver y representar algoritmos utilizando las estructuras de ciclos y decisión.

2.7.6. Ejercicios

Se recomienda resolver los siguientes ejercicios identificando el algoritmo y representarlo en diagrama de flujo y pseudocódigo.

1. Elabore un algoritmo que imprima los números enteros del 0 al 100.
2. Elabore un algoritmo que imprima los números enteros del 100 al 0.
3. Elabore un algoritmo que solo permita ingresar N veces la letra "S" y se detenga cuando se ingrese por teclado la letra "N".
4. Imprima 10 veces la serie del 1 al 10.
5. Realice un algoritmo que encuentre la suma de los primeros cinco números naturales.
6. Realice un algoritmo que encuentre la suma de los primeros N números naturales.
7. Haga un algoritmo que encuentre el valor mayor y menor dentro de N números y encuentre el promedio entre todos los números ingresados.
8. Elabore un algoritmo que permita encontrar la factorial de un número.
9. Elabore un algoritmo que encuentre la factorial de cada uno de N números enteros positivos.
10. Haga un algoritmo que lea un valor positivo y muestre desde 1 hasta el valor ingresado de uno en uno.
11. Haga un algoritmo que pregunte si se desea ingresar calificaciones S/N de un alumno e informe:
    - Cuántas calificaciones son mayores o igual a 7.
    - Cuántas calificaciones son menores a 7.
    - Se debe ingresar calificaciones hasta que el usuario ingrese la letra "N" por teclado.
12. Haga un algoritmo que lea nombres de personas hasta que se ingrese un valor en blanco o nulo.
13. Haga un algoritmo que permita ingresar N números hasta que se ingrese el número 0 (cero) y obtenga el promedio de los números ingresados.
14. Haga un algoritmo que permita imprimir el promedio de 10 números enteros leídos por teclado.
15. Elabore un algoritmo que permita leer 5 números enteros y calcule el promedio final.
16. Haga un algoritmo que permita imprimir la tabla de multiplicar del 0 al 12 de un número ingresado por teclado.
17. Haga un algoritmo que encuentre la suma de los números pares comprendidos entre 1 y N.
18. Realice la suma de N números hasta que se ingrese como dato 0.
19. Cree un algoritmo que encuentre:
    - La suma de los valores de un conjunto de números enteros.

- La cantidad de valores negativos, positivos, iguales a cero.
- El total de números en el conjunto.

20. Un local de comida rápida ofrece hamburguesas sencillas (S), dobles (D) y triples (T), las cuales tienen un costo de $3, $5 y $8 respectivamente. La empresa acepta tarjetas de crédito con un cargo de 5% sobre la compra. Suponiendo que los clientes adquieren N hamburguesas, las cuales pueden ser de diferente tipo, realice un algoritmo para determinar:
    - El número de hamburguesas por cada tipo.
    - Cuánto debe pagar al finalizar la compra.
21. Se requiere un algoritmo para conocer de N cantidades:
    - ¿Cuántas son igual a cero?
    - ¿Cuántas son menores a cero?
    - ¿Cuántas son mayores a cero?
22. Realice un algoritmo para determinar cuánto ahorrará en pesos una persona diariamente, y en un año, si ahorra 3 centavos el primero de enero, 9 centavos el dos de enero, 27 centavos el tres de enero y así sucesivamente todo el año.
23. Se desea saber el total de una caja registradora de un almacén, se conoce el número de billetes y monedas, así como su valor. Realice un algoritmo para determinar el total.
24. Un vendedor ha realizado N ventas y desea saber:
    - ¿Cuántas fueron de $1 a $10.000?
    - ¿Cuántas fueron por más de $10.000 pero por menos de $20.000?
    - ¿Cuánto fue el monto de las ventas de cada una y el monto global?
    - Realice un algoritmo para determinar los totales.
25. Realice un algoritmo para leer las calificaciones de N alumnos y determine el número de alumnos aprobados y reprobados. Un alumno está aprobado cuando su calificación es mayor o igual a 7/10.
26. En 1940, una persona vendió las tierras de su abuelo al gobierno por la cantidad de $2.000. Suponga que esta persona ha colocado el dinero en una cuenta de ahorros que paga 13% anual. ¿Cuánto vale ahora su inversión? Realice un algoritmo para obtener este valor.
27. Realice un algoritmo para determinar cuánto pagará una persona que adquiere N artículos, los cuales están de promoción. Considere que si su precio es mayor o igual a $200 se le aplica un descuento de 15%, y si su precio es mayor a $100 pero menor a $200, el descuento es de 12%; de lo contrario, sólo se le aplica 10% de descuento. Se debe saber cuál es el costo y el descuento que tendrá cada uno de los artículos y finalmente cuánto se pagará por todos los artículos vendidos.
28. El gerente de una distribuidora de bicicletas desea determinar el impuesto que va a pagar por cada bicicleta que posee, además del total que va a pagar por cada categoría y por todas las bicicletas que tiene a disposición, basándose en la siguiente clasificación:
    - Bicicletas con clave 1 pagan 10% de su valor.
    - Bicicletas con clave 2 pagan 7% de su valor.

- Bicicletas con clave 3 pagan 5% de su valor.
- Realice un algoritmo para obtener dicha información.

29. Los directivos de una escuela quieren determinar cuál es la edad promedio de cada uno de las X salones y cuál es la edad promedio de toda la escuela. Cree un algoritmo para determinar dichos promedios.
30. Un profesor tiene un salario inicial de $500, y recibe un incremento de 10% anual durante 6 años, se quiere conocer:
    - ¿Cuál es su salario al cabo de 6 años?
    - ¿Qué salario ha recibido en cada uno de los 6 años?
31. Calcule y escriba las tablas de multiplicar del 10 al 20 en los siguientes rangos: 25 al 35, 45 al 50 y 60 al 85 lo mismo que el promedio de cada rango.
32. Dado un grupo de 10 números enteros diferentes de cero, realice un algoritmo que permita:
    - Encontrar el número mayor y menor.
    - Cantidad de número mayores a 15.
    - Cantidad de números negativos.
    - Promedio de números positivos
33. Una empresa tiene un número N de empleados y de cada uno de ellos se solicita la siguiente información:
    - Código, nombres, número de hijos, salario por hora y número de horas trabajadas al mes.
    - La retención para cada empleado se determina de la siguiente manera:
    - Para salarios menores de $500:
    - Si el número de hijos es mayor de 6 no hay retención; si el número de hijos es menor o igual a 6, se le retiene un porcentaje igual a 6% menos el número de hijos dividido por 2.
    - Para salarios iguales o mayores a $500:
    - Si el número de hijos es menor de 3, se le retiene un 3%; si el número de hijos es mayor o igual a 3 se le retiene un porcentaje igual a 10% dividido por el número de hijos.
    - Elabore un algoritmo que muestre: código, nombres, retención y total por pagar.
34. Una oficina de seguros ha reunido datos concernientes a todos los accidentes de tránsito ocurridos en el área metropolitana de Quito en el último año. Por cada conductor involucrado en un accidente se toman los siguientes datos: año de nacimiento, sexo (1: femenino; 2: masculino), registro del carro (1: Quito; 2: Otras ciudades). Haga un algoritmo que muestre:
    - El porcentaje de conductores menores de 25 años.
    - Porcentaje de conductores del sexo femenino.
    - Porcentaje de conductores masculinos con edades entre 12 y 30 años.
    - Porcentaje de conductores cuyos carros están registrados fuera de Quito.

35. Una empresa de aviación fumiga cosechas contra una gran variedad de plagas. Los valores cobrados a los propietarios de las granjas dependen de lo que se desee fumigar y de cuántas hectáreas se fumigan, de acuerdo con la siguiente distribución:
    - Caso 1: fumigación contra malas hierbas, 10 dólares por hectárea.
    - Caso 2: fumigación contra langostas, 15 dólares por hectárea.
    - Caso 3: fumigación contra gusanos, 20 dólares por hectárea.
    - Caso 4: fumigación contra todo lo anterior, 30 dólares por hectárea.
    - Si el área por fumigar es mayor de 1.000 hectáreas, el granjero goza de un 10% de descuento. Además, cualquier granjero cuya cuenta sobrepase los 3.000 dólares se le descuenta un 15% sobre la cantidad que exceda dicho precio. Si se aplica ambos conceptos, el correspondiente a la superficie se considera primero. Por cada pedido se tiene la siguiente información: nombre del granjero, tipo de fumigación (1, 2, 3, 4) y el número de hectáreas por fumigar. Por cada solicitud se debe suministrar: nombre del granjero y valor por pagar. Además, el algoritmo debe permitir realizar N cotizaciones de servicio hasta que el usuario lo desee.
36. En un curso se generan 4 evaluaciones con los siguientes porcentajes: 25%, 20%, 25% y 30%, respectivamente. Por cada estudiante se ingresa el nombre y los cuatro aportes. Haga un algoritmo que calcule la nota definitiva de cada estudiante, el promedio de notas definitivas del curso y el porcentaje de reprobados (menor a 7).
37. Un restaurante paga a sus meseros dos clases de comisiones:
    - Una comisión del 7% sobre toda venta.
    - Otra comisión que depende del tipo de venta: 15% si la venta es de contado, 10%si la venta se hizo en cheque, y 5% si se hizo con tarjeta de crédito.

    El restaurante tiene por cada venta:

    - Identificación del vendedor (1, 2, 3).
    - Tipos de ventas (1: contado; 2: cheque; 3: tarjeta).
    - Total de la venta.

    Elabore un algoritmo que obtenga el total por pagar a cada uno de los empleados.
38. Se tiene la siguiente información de los N estudiantes de una universidad:
    - Edad.
    - Sexo (1: masculino; 2: femenino).
    - Carrera (1: Tecnología; 2: otra carrera).
    - Haga un algoritmo que obtenga:
        - Promedio de edad de los estudiantes de Tecnología.
        - Porcentaje de hombres en la universidad.
        - Porcentaje de mujeres que estudian tecnología.
39. Elabore un algoritmo que haga el siguiente censo para una empresa de transporte:

  - Número de vehículos cuyo modelo sea anterior a 1990.
  - Número de vehículos cuyo modelo sea de 1990 o posterior y cuya capacidad sea menor de 40 pasajeros.
  - Número de buses cuyo modelo sea posterior a 1990 con capacidad mayor de 40 pasajeros.
  - Número de buses cuyo modelo sea anterior a 1990 con capacidad mayor de 40 pasajeros.
  - Número de busetas con capacidad menor de 40 pasajeros.
  - El total de vehículos de la empresa.
- Por cada vehículo la empresa tiene la siguiente información:
  - Tipo de vehículo (1: buseta; 2: bus)
  - Modelo del vehículo
  - Capacidad del vehículo

40. Un hotel presta cuatro clases de servicios. Por cada servicio que presta se tienen los siguientes datos: clase de servicio prestado (valores del 1 al 4), jornada en la que se prestó el servicio (M: mañana; T: tarde) y valor del servicio. Al final del día se quiere determinar el valor producido por cada clase de servicio, el número de veces que se prestó cada servicio, el servicio que más veces se prestó y si este se prestó más en la mañana o en la tarde.

3. Capítulo III

3.1. Arreglos

En los capítulos anteriores, únicamente se trabajó con variables definidas de tipo entero, real, carácter y booleano. En este capítulo, la definición de variable se expande a una colección de ellas, agrupadas bajo un solo nombre y diferenciadas por un índice.

Imagine el siguiente escenario: se desea almacenar 2.000 nombres de personas. En primera instancia, se debería utilizar 2.000 variables de tipo carácter para almacenar dicha información. Sin embargo, esto conllevaría tener un programa demasiado complicado de mantener; precisamente es en este tipo de casos en donde se utilizan los arreglos. En este texto se estudiarán los arreglos de una dimensión o vectores y los arreglos de dos dimensiones o matrices.

3.2. Arreglos de una dimensión o vectores

Un arreglo es una estructura formada por un grupo de componentes finitos. Dichos componentes deben ser del mismo tipo de dato y son representados a través de un solo nombre y diferenciados entre sí por un índice o subíndice.

La cantidad de elementos del arreglo dependerá del total de datos que se va a manejar. Si la cantidad de datos es N, entonces se define el arreglo con N elementos. Se debe considerar que el índice siempre empieza desde el número 0; es así que si un vector tiene N elementos también tendrá N-1 de índices. Observe la siguiente figura:

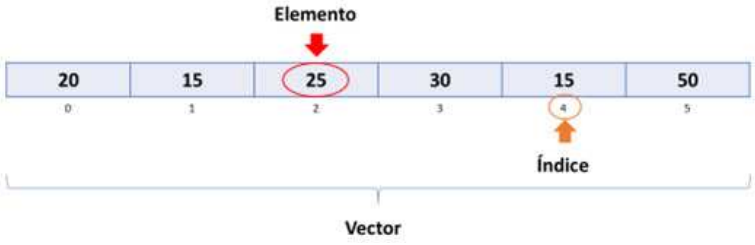


Figura 66. Estructura de un Vector

Elaborado por Carlos Salazar

Como puede notar en la figura 66, se presenta un arreglo con 6 elementos de tipo entero. Nótese que tiene 6 índices que van del número 0 al 5, esto permitirá manipular de forma independiente cada componente del arreglo.

En base del vector de la figura 66, suponga que el vector se llama "Numeros". Cada elemento pertenecería a un índice de dicho vector, por ejemplo:

El vector "Numeros" en su índice "[0]" tiene el valor de 20.
El vector "Numeros" en su índice "[3]" tiene el valor de 30.
El vector "Numeros" en su índice "[5]" tiene el valor 50.

3.2.1. Asignación manual de elementos

En esta sección se indicará cómo asignar elementos a un vector de manera manual, lo cual resulta muy útil en algunos casos. Para empezar, en un algoritmo la asignación se realizaría de la siguiente manera:

Numeros[0] = 20
Numeros[1] = 15
Numeros[2] = 25
Numeros[3] = 30
Numeros[4] = 15
Numeros[5] = 50

La asignación manual de elementos a un vector en diagramas de flujo utilizando DFD se realizaría de la siguiente manera. Observe la siguiente figura:

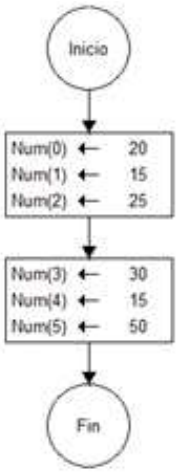


Figura 67. Asignación manual Vector unidimensional

Elaborado por Carlos Salazar

Como puede observar en la figura 67, los índices se encuentran entre paréntesis. A continuación, observe la asignación manual utilizando pseudocódigo:

```
1  Proceso asignacion_manual
2      Definir ArregloNumeros Como Entero;
3      Dimension ArregloNumeros[5];
4
5      ArregloNumeros[0] = 20;
6      ArregloNumeros[1] = 15;
7      ArregloNumeros[2] = 25;
8      ArregloNumeros[3] = 30;
9      ArregloNumeros[4] = 15;
10     ArregloNumeros[5] = 50;
11
12 FinProceso
```

Figura 68. Asignación manual Vector unidimensional pseudocódigo

Elaborado por Carlos Salazar

De la figura 68, nótese lo siguiente:

- Antes de utilizar un arreglo es necesario definirlo, observe la línea 2.
- Una vez que se ha definido el nombre del arreglo es necesario indicar sus dimensiones, observe la línea 3.
- A diferencia de los diagramas de flujo, el índice va entre corchetes junto al nombre del arreglo o vector.

3.2.2. Lectura y escritura manual de elementos

En esta sección se explicará cómo representar la lectura (almacenar elementos desde teclado) en un arreglo o vector a través de un algoritmo:

Ingresar en Numeros[0]
Ingresar en Numeros[1]

Una vez que se han guardado elementos a un vector, es muy probable que sea necesario mostrar en pantalla dicha información. Esto se puede hacer de la siguiente manera:

Imprimir Numeros[0]
Imprimir Numeros[1]

Es posible realizar en diagramas de flujo de la siguiente manera:

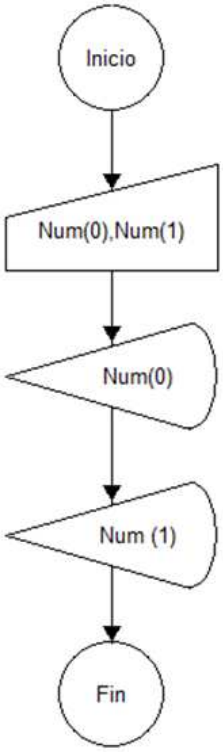


Figura 69. Lectura y escritura manual de vectores diagramas de flujo

Elaborado por Carlos Salazar

A continuación, se muestra la lectura y escritura manual de elementos utilizando pseudocódigo:

```
Proceso asignacion_manual
    Definir Num Como Entero;
    Dimension Num[2];

    Leer Num[0];
    Leer Num[1];

    Escribir Num[0];
    Escribir Num[1];

FinProceso
```

Figura 70. Lectura y escritura manual de elementos en un Vector pseudocódigo

Elaborado por Carlos Salazar

3.2.3. Lectura y escritura con ciclo repetitivo

Realizar la asignación, lectura y escritura de elementos en un vector de manera manual es de utilidad para casos puntuales. Sin embargo, en la mayoría de casos dichas acciones se las realiza dentro de un ciclo repetitivo, logrando así optimizar el algoritmo. Observe el siguiente ejemplo:

■ ENUNCIADO

Elaborar un algoritmo que permita ingresar 5 números enteros en un arreglo y después se muestre su contenido.

ALGORITMO

Algoritmo: Elementos_vector

Inicio

1. Definir un vector llamado “num” de 5 dimensiones
2. Definir una variable “indice” que inicie en 0
3. Repetir hasta que índice sea < a 5 con un incremento de uno en cada iteración
   Leer num[indice]
4. Repetir hasta que índice sea < a 5 con un incremento de uno en cada iteración
   Escribir num[indice]

Fin

DIAGRAMA DE FLUJO

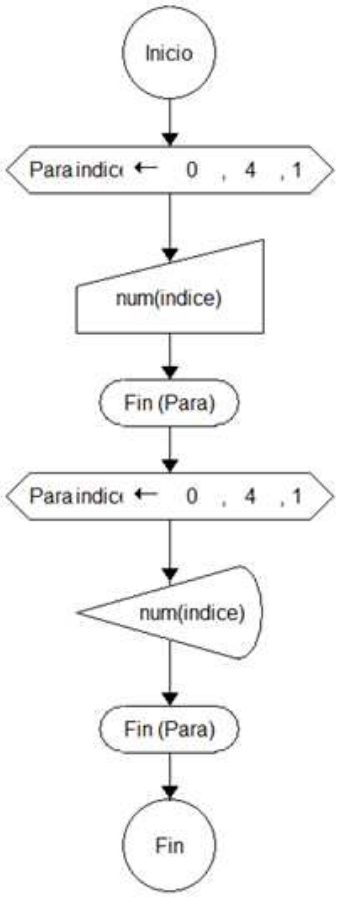


Figura 71. Lectura y escritura de elementos de un Vector diagrama de flujo

Elaborado por Carlos Salazar

Como se puede observar en la figura 71, se tiene dos ciclos "Para": el primero se utiliza para guardar los datos en un vector llamado "num" y se utiliza la variable "indice" para indicar la posición en la que se va a almacenar el valor ingresado; el segundo ciclo "Para" se utiliza para recorrer e imprimir el contenido del vector "num". Nótese que también se hace uso de la variable "indice" para identificar la posición del vector que se desea imprimir en pantalla. Observe su implementación en pseudocódigo:

PSEUDOCÓDIGO

```
Proceso asignacion_manual
    Definir Num Como Entero;
    Dimension Num[5];

    Para indice = 0 Hasta 4 Con Paso 1 Hacer
        Leer Num[indice];
    Fin Para
    Para indice = 0 Hasta 4 Con Paso 1 Hacer
        Escribir Num[indice];
    Fin Para
FinProceso
```

Figura 72. Lectura y escritura de datos en un Vector en pseudocódigo

Elaborado por Carlos Salazar

Observe la figura 72: la implementación del algoritmo en pseudocódigo es muy similar al diagrama de flujo. Nótese las siguientes diferencias:

- Es necesario definir el vector antes de utilizarlo.
- El ciclo "para" tiene diferente estructura pero tiene el mismo funcionamiento.

Es posible leer y escribir vectores utilizando otro tipo de estructuras repetitivas. Sin embargo, la estructura "para" es la más común para este fin.

A continuación se propone una serie de ejercicios para que el estudiante fortalezca y desarrolle la habilidad de resolver y representar algoritmos.

3.2.4. Ejercicios

Se recomienda resolver los siguientes ejercicios identificando el algoritmo y representarlo en diagrama de flujo y pseudocódigo.

1. Realice un algoritmo que permita ingresar en un vector 5 valores numéricos y encuentre el número mayor y su índice.
2. Desarrolle un algoritmo que permita ingresar 10 números enteros en un vector e imprima los números que sean múltiplos de tres.
3. Haga un algoritmo que almacene 8 números enteros y los almacene en un vector. Al finalizar se debe mostrar la suma de los elementos que se encuentren en índices pares e impares por separado.
4. Cargue en un vector de 10 elementos, los 10 primeros números de la serie Fibonacci y muéstrelos en pantalla.
5. Lea los nombres y la edad de 5 personas y después imprima los datos de la persona más joven.
6. Realice un algoritmo que permita leer 5 números en un vector, y a partir de este genere otro vector con el resultado de multiplicar el vector ingresado por 5.
7. Elabore un algoritmo que permita ingresar la edad y los nombres de 5 estudiantes y muestre cuántos estudiantes son mayores a 20 años.
8. Lea 5 palabras, almacénelas en un arreglo y a continuación visualícelas separadas por un espacio en blanco.
9. Cree un algoritmo que permita almacenar en un vector los números del 1 al 10 y en otro vector el número escrito en inglés, respectivamente. Se debe solicitar ingresar un número del 1 al 10 y se mostrará el nombre del número ingresado en inglés.
10. Realice un algoritmo que permita ingresar en un vector únicamente 5 números mayores a 10. Si se ingresa un número menor, no debe ser almacenado en el vector. Al finalizar se debe imprimir el vector.
11. Cree un algoritmo que permita almacenar 3 números en un vector y obtenga el promedio de los valores ingresados.
12. Realice un algoritmo que permita ingresar 10 números enteros en un vector y obtenga el promedio. Además, debe almacenar en otro vector los elementos que se encuentren sobre o sean iguales al promedio y en otro vector los elementos que sean inferiores al promedio.
13. Elabore un algoritmo que lea dos vectores de N tamaño (N se ingresa por te-

clado) y forme un tercer arreglo con el producto de los elementos de los dos arreglos. Al finalizar se deben imprimir los tres vectores.

14. Cree un algoritmo que lea dos vectores de N tamaño (N se ingresa por teclado) y forme un vector para la suma y otro para la resta de los vectores ingresados. Al finalizar se deben imprimir todos los vectores.
15. Realice un algoritmo que forme dos arreglos de N dimensiones (N se ingresa por teclado). El primero contiene los números de cédula de estudiantes de una universidad que reprobaron la asignatura de "MATEMATICA" y el segundo contiene los números de cédula de estudiantes que reprobaron la asignatura de "INGLES". Partiendo de los dos vectores, se debe formar un tercero que contenga el código de estudiantes que reprobaron las dos materias. Al finalizar imprimir dicho vector.
16. Se tiene un vector de 20 elementos con valores numéricos enteros. Haga lo siguiente:
    - Lea un valor x y busque en qué posición del vector se encuentra.
    - Llene otro vector con los elementos de las posiciones impares del vector dado.
    - En este último vector, busque cuántos elementos son mayores a 20.
17. Elabore un algoritmo que permita ingresar los códigos y nombres de N empleados de una empresa (N se ingresa por teclado). Al finalizar, debe solicitar ingresar un código y se debe imprimir el nombre del empleado que corresponda.
18. Elabore un algoritmo que ingrese 10 valores numéricos en un vector y calcule e imprima:
    - El número de datos repetidos en el vector.
    - El número de valores impares.
    - El número de valores pares.
19. Cree un algoritmo que forme dos vectores relacionados que almacenen los códigos de los N artículos que se venden en un almacén y el stock de cada uno de los artículos, por ejemplo:

Código	Existencia
1	30
2	40
N...	...

    - Por cada proveedor o cliente que llega al almacén, se genera un registro con los siguientes datos:
    - Tipo de transacción (1 - recibido; 2 - venta).
    - Código del artículo.
    - Números de unidades (recibidas o vendidas).

    Se requiere calcular lo siguiente para cada transacción:
    - Si se reciben productos, se suma a la existencia actual de ese artículo el número de unidades de la transacción.
    - Si es venta de productos, se suma a las unidades en existencia de ese artículo las unidades vendidas.

Al final se deben mostrar los códigos de los artículos y las existencias de cada uno de ellos, es decir, el arreglo de códigos y de existencias actualizado.

Para el ejemplo, se debe asumir que en ningún escenario la cantidad vendida es mayor que la cantidad en existencia y por lo tanto no se presentarán inconsistencias.

20. En una universidad se efectúa un examen de admisión que consta de dos pruebas: aptitud matemática y aptitud verbal. Cada pregunta tiene 5 opciones numeradas del 1 al 5. Se prepara un registro con 20 campos de una sola posición que contiene, cada uno, la respuesta correcta a la pregunta correspondiente. Las 10 primeras posiciones corresponden al examen de aptitud matemática y las restantes a las de aptitud verbal. Al finalizar se debe indicar el número de respuestas correctas e incorrectas de las dos pruebas.

3.3. Matrices

A diferencia de los vectores, las matrices son un conjunto de datos organizados en forma de filas y columnas en donde, para referenciar cada dato, se necesita establecer claramente en qué fila y en qué columna se encuentra.

Como menciona (Regino, p 175):

> Las matrices son estructuras de datos que organizan su información en forma de tablas; es decir, los elementos que la conforman están dispuestos bajo dos conceptos de clasificación (fila, columna). Para poder indicar el lugar donde se encuentra un determinado elemento, es necesario utilizar dos índices: uno para indicar el renglón o fila y otro para indicar la columna.

También, a una matriz se la conoce como un vector de vectores de dimensión N * M, en donde, N representa las filas y M las columnas.

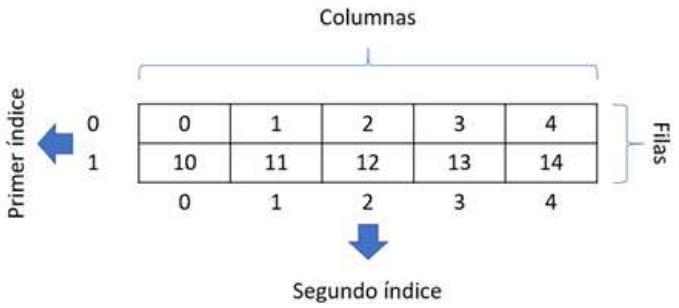


Figura 73. Estructura de una Matriz

Elaborado por Carlos Salazar

Como se observa en la figura 73, la matriz de ejemplo consta de dos filas y cinco columnas, en donde todos sus elementos son de tipo entero; por lo tanto, la matriz también será de tipo entero.

En el siguiente ejemplo, observe como se manipulan los elementos de una matriz de manera manual y a través de un ciclo repetitivo.

3.3.1. Manipulación de elementos en una Matriz

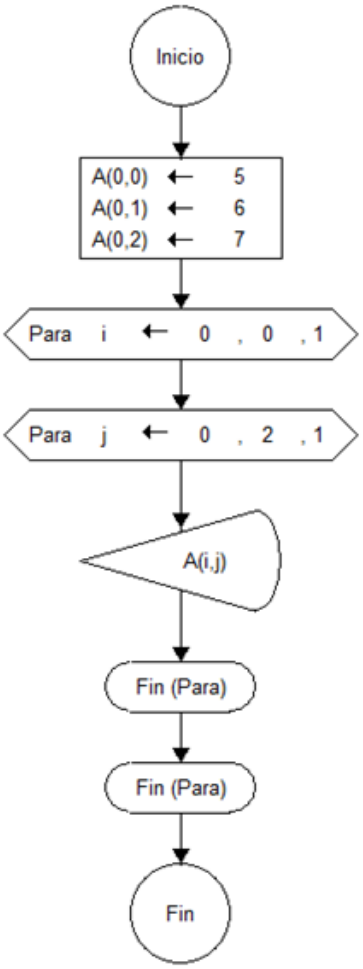


Figura 74. Manejo de elementos Matriz diagrama de flujo

Elaborado por Carlos Salazar

En la figura 74, nótense las siguientes observaciones:

- Los índices de las matrices van entre paréntesis.
- Primero se coloca el índice de las filas y después el de las columnas, separados entre sí por una coma.
- Para recorrer una matriz se requiere de dos ciclos repetitivos Para: el primero para recorrer las filas y el segundo para recorrer las columnas.
- Cuando se desea imprimir un elemento de la matriz en un bucle repetitivo, se debe colocar la variable de tipo contador que representa las filas y la que representa las columnas. En este ejercicio (i, j) respectivamente.

```
Proceso Matriz

    Definir Matrices Como Entero;
    Dimension Matrices[2,4]

    Matrices[0,0]=1
    Matrices[0,1]=2
    Matrices[0,2]=3
    Matrices[0,3]=4

    Matrices[1,0]=4
    Matrices[1,1]=3
    Matrices[1,2]=2
    Matrices[1,3]=1

    Para i=0 Hasta 1 Con Paso 1 Hacer
        Para j=0 Hasta 3 Con Paso 1 Hacer
            Escribir Matrices[i,j]
        Fin Para
    Fin Para

FinProceso
```

Figura 75. Manipulación de elementos Matriz pseudocódigo

Elaborado por Carlos Salazar

De la figura 75, nótense los siguientes puntos:

- En la definición de la dimensión primero se indica el número de filas y después el número de columnas entre corchetes.
- Para recorrer la matriz, se hace uso de dos ciclos repetitivos para, uno se utiliza para recorrer las filas y el otro para recorrer las columnas, además, de indicar el respectivo incremento de dichas variables en cada ciclo.

A continuación, observe el siguiente ejemplo:

■ ENUNCIADO

Ingrese 8 números enteros en una matriz de 4 x 2 y posteriormente imprima el contenido de la matriz.

ALGORITMO

Algoritmo: recorrer_matriz

Inicio

- Definir una matriz de 4 x 2 de tipo entero
- Repetir 4 veces
  Repetir 2 veces
  Ingrese un número entero y almacenar en la matriz
- Repetir 4 veces
  Repetir 2 veces
  Imprimir elemento de matriz

Fin

DIAGRAMA DE FLUJO

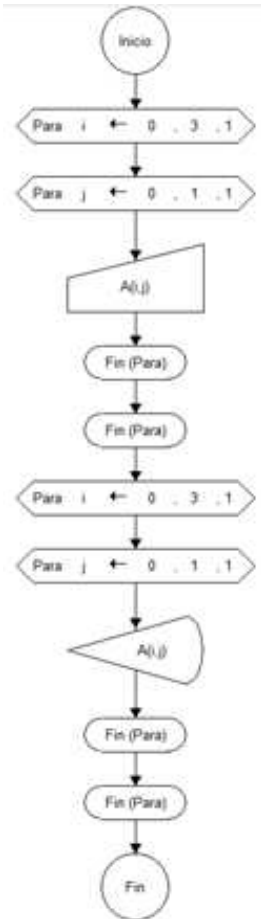


Figura 76. Manejo de matrices diagrama de flujo

Elaborado por Carlos Salazar

PSEUDOCÓDIGO

```
Proceso recorrer_matriz
    Definir A Como Entero;
    Dimension A[4,2];

    Para i = 0 Hasta 3 Con Paso 1 Hacer
        Para j = 0 Hasta 1 Con Paso 1 Hacer
            Leer A[i,j];
        Fin Para
    Fin Para

    Para i = 0 Hasta 3 Con Paso 1 Hacer
        Para j = 0 Hasta 1 Con Paso 1 Hacer
            Escribir  A[i,j];
        Fin Para
    Fin Para
FinProceso
```

Figura 77. Manejo de matrices en pseudocódigo

Elaborado por Carlos Salazar

A continuación, se propone una serie de ejercicios para que el estudiante fortalezca y desarrolle la habilidad de resolver y representar algoritmos.

3.3.2. Ejercicios

Se recomienda resolver los siguientes ejercicios identificando el algoritmo y representarlo en diagrama de flujo y pseudocódigo.

1. Leer dos matrices de orden Mx N y generar una tercera matriz con la suma de las dos.
2. Leer una matriz A de orden M x N y un número K. Multiplicar todos los elementos de la matriz por el número K. Mostrar la matriz resultante.
3. Crear un algoritmo que permita ingresar números enteros en una matriz de 4 x 3 y al finalizar debe mostrar:
   - El promedio por columna.
   - El promedio por fila.
   - El promedio de toda la matriz.
4. Desarrolle un algoritmo que permita leer una matriz de 4x4 y determine cuántas veces se repite en ella el número mayor.
5. Realice un algoritmo que permita leer una matriz de orden 4 x 3, se debe calcular la suma de los elementos de cada fila y determinar cuál es la fila que tiene la mayor suma.
6. Cree un algoritmo que permita leer una matriz de orden 4 x 2 y permita calcular el promedio de los números mayores de cada fila.
7. Leer una matriz de 3 x 2 y determinar cuántos números almacenados en ella tienen más de tres dígitos.
8. Elaborar un algoritmo que lea una matriz de 3 x 2 por filas y la imprima por columnas.
9. Elaborar un algoritmo que lea la dimensión de una matriz y sus elementos, finalmente debe mostrar el número mayor y menor con sus respectivas posiciones.
10. Implementa un algoritmo que permita realizar operaciones sobre matrices de M x N. El programa debe permitir al usuario la selección de alguna de las siguientes operaciones: sumar, restar, multiplicas o trasponer dos matrices.
11. Una fábrica produce N artículos diferentes y utiliza M sucursales para su distribución, variando el precio de venta de cada artículo según la sucursal que lo distribuye. Para esto la fábrica tiene un cuadro que muestra el precio de cada artículo según la sucursal que lo comercia. Al final de cada periodo cada sucursal envía a la fábrica la cantidad vendida de cada artículo, formándose un nuevo cuadro. Elaborar un algoritmo que encuentre las ventas totales de la fábrica para cada uno de los artículos y para cada una de las sucursales.
12. Un avión dispone de 180 plazas, de las cuales 60 son de no fumador numeradas del 1 a las 60 y 120 plazas numeradas del 61 al 180. Diseñar un algoritmo que permita reservar plazas del avión y diga cuántos pasajeros de cada clase ocupan asientos en el vuelo.
13. Estudiantes provenientes de 5 colegios, han decidido presentar examen de ingreso a la universidad. Se requiere de una matriz que indique los resultados de

las personas de cada colegio que han presentado el examen. La entrada está compuesta por el código del estudiante, el código del colegio (de 1 a 5) y el resultado del examen. La salida debe tener los puntos de los estudiantes procedentes de cada colegio y un total de 10 estudiantes por cada uno.

14. Elaborar un algoritmo que forme una matriz de N * M elementos. Cada elemento del arreglo representa las ventas atribuibles a cada uno de los N vendedores de una empresa, para cada uno de los M años de operaciones que ha tenido la misma y, luego calcular:
    - El total de ventas de cada vendedor de los M años.
    - El total de ventas en cada año.
    - El gran total de ventas de la empresa.
15. En una panadería se elaboran diariamente los siguientes tipos de productos: galletas (0,15), pan (0,65), biscocho (0,5), palanqueta (0,9), queso de ½ kg (1,5), queso de 1kg (2,5) y queso de 3kg (4,5). Los fines de semana se elaboran tortas (1,45) y empanadas (1,8). Los precios de cada producto se indican entre paréntesis. El panadero desea mantener actualizada la información de una semana sobre la cantidad diaria de productos elaborados, productos vendidos en horario de mañana y productos vendidos en horario de la tarde. Considera que el horario de mañana es de 8 a 13h, y el de la tarde es de 16 a 19 h. Realiza las declaraciones de tipos apropiadas para representar la citada información y escribe un algoritmo que muestre en pantalla el dinero obtenido por la venta de todos los productos el último día de la semana.
16. Se desea obtener el grado de eficiencia de N operarios de una fábrica productora de pupitres, de acuerdo con las siguientes condiciones que se les impone para un periodo determinado. Condiciones:
    - Ausencia del trabajo < = 3,5 horas.
    - Pupitres defectuosos < 300.
    - Pupitres producidos > 10000.

    Los grados de eficiencia del trabajador son asignados de la siguiente forma:
    - Si no cumple ninguna condición: Grado = 5
    - Si sólo cumple la primera condición: Grado = 7
    - Si sólo cumple la segunda condición: Grado = 8
    - Si sólo cumple la tercera condición: Grado = 9
    - Si cumple la primera y la segunda: Grado = 12
    - Si cumple la primera y la tercera: Grado = 13
    - Si cumple la segunda y la tercera: Grado = 15
    - Si cumple las tres condiciones: Grado = 20

    Por cada trabajador se tiene un registro con los siguientes datos:
    - Código del trabajador.
    - Horas de ausencia.
    - Pupitres defectuosos.
    - Pupitres producidos.

    Elaborar un algoritmo que calcule para cada trabajador su correspondiente grado de eficiencia e imprima toda su información.

17. Un campo de golf consta de 18 hoyos, en ellos debe introducirse, sucesivamente, una pelota a base de golpes con un bastón. En una tarjeta van anotándose el número de golpes requeridos para llegar a cada uno de los hoyos. En la misma tarjeta van anotándose el número de golpes requeridos para llegar a cada uno de los hoyos. En la misma tarjeta pueden anotarse los golpes de varios jugadores, ya que ésta tiene la forma de una tabla (matriz): los renglones corresponden a los jugadores y las columnas a cada hoyo del campo. Por ejemplo, si en un juego participan 4 jugadores la tarjeta tendrá 4 renglones y 18 columnas. El juego lo gana el participante que llegue al hoyo 18 con el menor número de golpes.

    Suponga que después de concluido el partido se tiene la tarjeta con todos los golpes de los N jugadores. Elaborar un algoritmo que obtenga:

    - La formación de un vector con los nombres de los participantes.
    - El nombre de la persona que ocupó el primer lugar, el segundo, etc., hasta el enésimo.

18. En una universidad, se tiene un grupo de N estudiantes; cada uno de ellos toma 8 materias; semestralmente el profesor de cada materia reporta a secretaría general el código del estudiante, el número de la materia y la calificación final. Elaborar un algoritmo que obtenga:

    - El promedio de las calificaciones obtenidas por cada estudiante.
    - El promedio de la calificación por cada materia.
    - El estudiante con mayor promedio.
    - El número de estudiantes aprobados (nota final sea mayor o igual a 7) por cada materia.
    - El número de estudiantes reprobados (nota final sea menor a 7) por cada materia.

19. Un instituto tiene 10 carreras y cada carrera tiene 5 niveles. Semestralmente cada carrera elabora un registro que contiene: código de la carrera y, a continuación, los costos de cada nivel. Elaborar un algoritmo que calcule e imprima:

    - Costo total y promedio por nivel de todas las carreras.
    - El nivel menos cuantioso por carrera.
    - El nivel más costoso por carrera.
    - La carrera más costosa.

20. En un hospital de la ciudad de Quito, se tiene la siguiente clasificación de tipo de urgencia:

NIVEL DE URGENCIA	TIPO DE URGENCIA
1	RESUCITACIÓN
2	EMERGENCIA
3	URGENCIA
4	URGENCIA MENOR
5	SIN URGENCIA

    El hospital tiene la capacidad de recibir a 100 pacientes por día, de los cuales se obtiene la siguiente información:

- Número de cédula.
- Nombres.
- Nivel de urgencia.
- Edad.

Elaborar un algoritmo que permita obtener la siguiente información:

- El tipo de urgencia que más pacientes tuvo.
- El tipo de urgencia que menos pacientes tuvo.
- Un listado de urgencias con su respectivo total de pacientes atendidos.
- El número de pacientes menores de edad.
- El número de pacientes mayores de edad.
- Un listado de pacientes con la siguiente información (número de cédula, nombres, edad y tipo de urgencia).

3.4. Subprocesos

Un subproceso o también conocido como subprograma, es una técnica dentro de la programación estructurada que permite dar más entendibilidad y facilidad en la construcción y corrección de posibles errores que se presentan a la hora de dar solución a un determinado problema.

A lo largo de este libro, se han entendido los problemas como si fuesen un todo, sin embargo, es necesario aplicar la conocida frase "Divide y vencerás", pues bien, precisamente de eso se tratan los subprocesos, tomar un problema macro, y dividirlo en pequeños subproblemas los cuales, trabajando en conjunto logran alcanzar el objetivo planteado.

Citando a (Regino, p 133), "Un subprograma es un algoritmo diseñado para efectuar una tarea particular, bajo la dependencia de un algoritmo u otro subprograma que lo utiliza." En otras palabras, un subproceso está diseñado para realizar ciertas acciones dentro del flujo de un algoritmo principal.

El uso de subprocesos en la creación de algoritmos provoca trabajar de manera modular, esto evita la creación de acciones redundantes y en vez de ello, se convoca a un módulo o subprograma para que ejecute dichas acciones las veces que sea necesario, por lo tanto, su uso hace que los algoritmos sean mas cortos en su implementación. Como lo menciona (Regino, p 133) "La fragmentación en módulos individuales proporciona claridad, facilidad de distribución de trabajo en equipo y, por ser algoritmos cortos, son más fáciles de escribir, corregir y su estructura lógica es más clara que la de los programas que no están divididos en módulos."

Se debe visualizar al subproceso como un algoritmo, el cual tiene un inicio, fin, datos de entrada, proceso y datos de salida; sin embargo, es utilizado para un propósito específico. Por lo general un subproceso recibe datos de un algoritmo o de otro subproceso que lo invoca y este devuelve un resultado. Imagínese el siguiente ejemplo: En un almacén se tiene un jefe encargado (algoritmo) que le da instrucciones a un empleado (subproceso), para que realice determinada labor y así poder continuar con su trabajo; en este lapso de tiempo, el jefe encargado (algoritmo) se detiene a la espera de que el empleado (subproceso) termine su labor para poder continuar con el suyo.

Cuando se trabaja de manera modular, es necesario pasar cierta información desde un algoritmo al subproceso, para ello, se hace uso de parámetros, las cuales funcionan como si se tratasen de variables.

Cuando se trabaja con subprocesos, se debe tener en cuenta lo siguiente:

- Los subprocesos deben ser muy sencillos, deben realizar acciones muy puntuales.
- Un programa se simplifica significativamente cuando se trabaja de manera modular, lo importante es identificar cómo y qué acciones realiza para cumplir con su objetivo.
- Un subproceso como representación de un módulo, tiene todas las características de un algoritmo.
- Los módulos o subprocesos siempre realizan una determinada tarea y constan de un grupo de acciones y un nombre por el cual son invocados desde un algoritmo u otro subproceso.
- Los subprocesos deben ser definidos antes del algoritmo que los invoca.
- Los argumentos que se envían desde el algoritmo deben ser igual en número y en tipo de dato de los que recibe el subproceso.

3.4.1. Representación de Subprocesos

Para representar los diagramas de flujo a lo largo de este libro, se ha utilizado el software DFD, dentro de este simulador, se utilizará la herramienta "Llamada" y "Nuevo Subprograma". El objeto "llamada" es utilizado para convocar a un subproceso o subprograma desde el algoritmo y el objeto "Nuevo Subprograma" es utilizado para definir el subproceso al que se está haciendo referencia, en esta sección se indica las acciones que realizará dicho subproceso al ser convocado.

En pseudocódigo, un subproceso inicia con la palabra "SubProceso" y finaliza con "FinSubProceso" las acciones que realizará se colocan dentro de este bloque. Observe el siguiente ejemplo:

■ ENUNCIADO

Crear un algoritmo que permita convocar tres veces un subproceso, el cual, imprimirá en pantalla un mensaje de "Hola mundo".

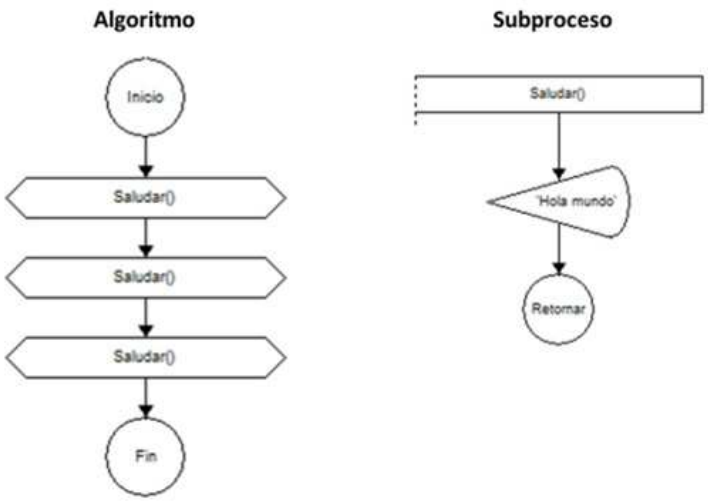


Figura 78. Subproceso imprimir diagrama de flujo

Elaborado por Carlos Salazar

```
SubProceso Saludar()
    Escribir 'Hola mundo';
FinSubProceso

Proceso ProcesoPrincipal
    Saludar();
    Saludar();
    Saludar();
FinProceso
```

Figura 79. Subproceso imprimir pseudocódigo

Elaborado por Carlos Salazar

Como puede observar en la figura 78 y 79, desde el algoritmo principal se realiza la invocación al subproceso las veces que se requiera.

Ahora bien, cambiando un poco el enunciado del ejemplo anterior, se va a realizar lo mismo, pero se debe enviar el mensaje que se desea imprimir a través de un parámetro. Observe el siguiente ejemplo:

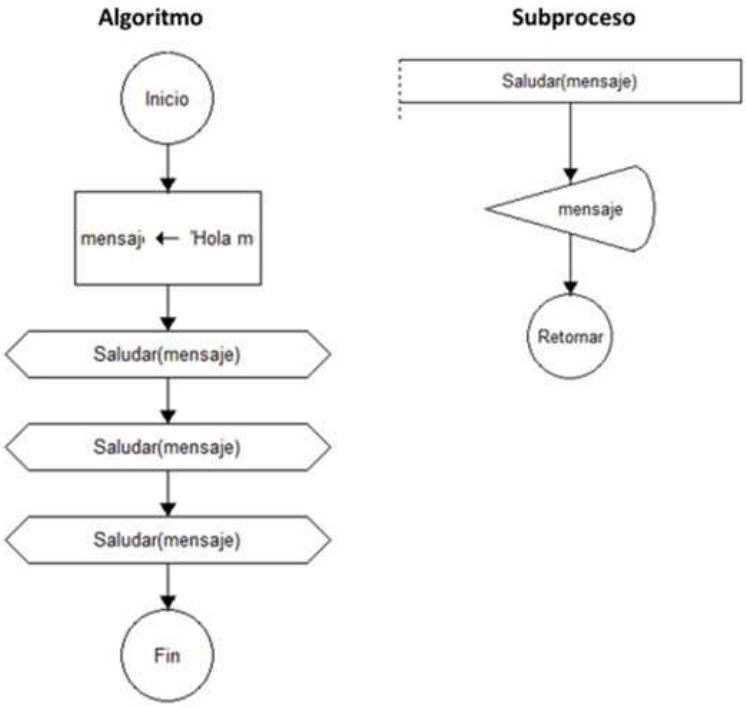


Figura 80 Subproceso con parámetros diagrama de flujo

Elaborado por Carlos Salazar

Observe la figura 80: nótese que en el algoritmo se define la variable mensaje con el texto "Hola mundo". Además, cuando se convoca al subproceso "Saludar" se envía dicha variable. En el subproceso se recibe la variable "mensaje" y se imprime su contenido.

A continuación, observe la implementación del ejercicio anterior utilizando pseudocódigo:

```
SubProceso ImprimirMensaje(message)
    Escribir message;
FinSubProceso

Proceso Mensaje
    Definir message Como Caracter;
    message = 'Hola mundo';
    ImprimirMensaje(message);
    ImprimirMensaje(message);
    ImprimirMensaje(message);

FinProceso
```

Figura 81. Subproceso con parámetros pseudocódigo

Elaborado por Carlos Salazar

Ahora observe el siguiente ejemplo de subprocesos que devuelven un valor:

```
Subproceso res = Operacion(num)
    res = num *2;
FinSubProceso

Proceso Multiplicacion
    Escribir "El resultado es: ",Operacion(5);

FinProceso
```

Figura 82. Subproceso con retorno

Elaborado por Carlos Salazar

A continuación, se propone una serie de ejercicios para que el estudiante fortalezca y desarrolle la habilidad de resolver y representar algoritmos.

3.3.2. Ejercicios

Se recomienda resolver los siguientes ejercicios identificando el algoritmo y representándolo en diagrama de flujo y pseudocódigo. Recuerde utilizar subprocesos en su implementación.

1. Cree un algoritmo que permita obtener el interés generado por X dólares que se invierten al 1% mensual en un periodo de N meses.
2. Haga un algoritmo que guarde 5 valores en un vector, luego imprima dicho vector.
3. Cree un algoritmo que permita leer un vector de N elementos y encuentre el número mayor.
4. Elabore un algoritmo que permita encontrar la posición del número mayor y menor en un vector de N elementos.
5. Sume los elementos correspondientes de dos vectores para dar como resultado un nuevo vector.
6. Elabore un algoritmo que emule el funcionamiento de una calculadora.
7. Elabore un algoritmo que busque un número dado un vector de N elementos y muestre su posición.
8. Elabore un algoritmo que permita leer N palabras y muestre un resumen de las palabras con el número de veces que se repite.

9. Elabore un algoritmo que emule las ventas en una tienda de ropa y muestre la prenda más vendida.
10. Elabore un algoritmo que emule la asignación de cupos de estudiantes a la universidad. Se debe mostrar el número de estudiantes Aceptados, Rechazados, de un total de N estudiantes.
11. Elabore un algoritmo que permita consultar el estado del clima para cualquier día de la semana.
12. Elabore un algoritmo que permita conocer a qué signo del horóscopo pertenecen N personas recibiendo como dato la fecha de nacimiento.
13. Elabore un algoritmo que emule el censo de una ciudad y determine:
    - Cuántos ciudadanos hombres y mujeres existen.
    - Cuántos ciudadanos son mayores de edad.
    - El porcentaje de la población que se encuentra trabajando.
    - El porcentaje de la población que tiene estudios superiores.
14. Cree un algoritmo que permita emular las multas realizadas por un radar de velocidad en la carretera. Se debe mostrar la siguiente información:
    - El número de vehículos que tuvieron una multa por exceso de velocidad.
    - El número de vehículos que viajaban a la velocidad máxima permitida.
    - El número de vehículos que circulaba a la velocidad mínima permitida.
15. Desarrolle un algoritmo que permita asignar conductores a una flota de buses interprovinciales por cada día de la semana. Al finalizar, se debe mostrar el calendario de asignación semanal entre buses y conductores.
16. Elabore un algoritmo que permita identificar el medicamento más vendido de la semana. Además, se debe mostrar el resumen diario de la venta de medicamentos que se tienen en stock.
17. Cree un algoritmo que permita conocer qué juego generó mayor ganancia en el día en un parque de atracciones.
18. Elabore un algoritmo que permita conocer el número de horas promedio que ve televisión una familia al día.
19. Establezca un algoritmo que permita identificar el curso con más estudiantes en una escuela de primero a sexto grado.
20. Desarrolle un algoritmo que permita conocer la cantidad promedio de integrantes de una familia en un determinado barrio.

3.5. Consejos y reflexiones sobre programación

He dedicado esta sección a la recopilación de consejos y reflexiones que he aprendido durante mi carrera profesional. Espero que los próximos párrafos le ahorren mucho tiempo en la curva de aprendizaje en su formación como desarrollador de software.

3.5.1. Sobre la lógica

No olvide la definición de lógica: "Es la manera más fácil y obvia de hacer algo". Esto resulta muy sencillo de aplicar a la hora de resolver problemas. Si en algún punto se observa que la solución que se planea implementar no es sencilla, por simple

definición, usted podría concluir que no es lo suficientemente obvia; por lo tanto, recomiendo que se tome el tiempo para replantear el algoritmo que está diseñando.

Una manera fácil de plantear soluciones es resolverlo utilizando nuestras propias reglas, las cuales, posteriormente se deben ajustar a las reglas de la lógica de programación (secuencias, decisiones, ciclos, etc.) para su futura codificación.

Es posible desarrollar nuestra lógica. Para ello, le sugiero destinar parte de su tiempo a resolver cualquier tipo de problema, no tiene necesariamente que ser sobre programación. Pueden ser acertijos matemáticos, cartas, juegos de mesa, ajedrez e incluso adivinanzas. Esto le permitirá desarrollar soluciones espontáneas, originales y creativas.

Durante su formación, lo más común es trabajar de manera individual asimilando los conceptos y poniéndolos en práctica. Sin embargo, cuando usted trabaje en equipo, podrá notar que sus compañeros encuentran formas diferentes de resolver problemas; usted debe tomarse el tiempo para escuchar sus propuestas y en conjunto definir qué algoritmo utilizar. Recuerde que no todas las personas pensamos de la misma manera: hay experiencias, conocimientos, convivencias, entorno personal, familiar y muchas más razones que hacen que una persona vea la solución desde una óptica diferente a la nuestra.

3.5.2. Sobre la metodología para resolver problemas

A estas alturas, podrá notar que la metodología que se ha propuesto le permitirá no solo resolver problemas computaciones, también podrá aplicarla para resolver cualquier tipo de problema.

Debe tener claro que antes de buscar una solución, debe tener muy pero muy claro cuál es el objetivo: esto permitirá conocer qué se debe hacer y hasta dónde se debe llegar. Recuerde que una vez identificado el objetivo no debe perderlo nunca de vista porque eso le permitirá visualizar el camino por seguir para crear el algoritmo.

Cuando usted diseñe algoritmos computacionales, estos siempre deben estar enfocados en facilitar su posterior codificación en cualquier lenguaje de programación. Siempre tenga en mente que detrás de una buena aplicación hay buenos algoritmos que permiten lograr un objetivo.

Una vez creado el algoritmo es necesario poder verificar su efectividad. Para ello es muy importante hacer uso de pruebas de escritorio con los escenarios posibles que se pueden presentar y evaluar el resultado que se obtiene; de esta manera, se puede mejorar y corregir cualquier desperfecto.

Si usted no realiza la prueba de escritorio, el algoritmo no es más que un grupo de pasos secuenciales que posiblemente den solución a un problema, y digo posiblemente porque se espera que lo resuelva, pero no se ha demostrado. Si se omite la prueba de escritorio, su verificación se realizará una vez que ya esté implementado en algún lenguaje de programación y su corrección en este punto puede conllevar dificultades y demora en la construcción del aplicativo.

El desarrollo de software parte de la metodología utilizada para resolver un problema. Me gustaría recalcar que, identificando el objetivo y su posterior implementación en un algoritmo con su respectiva prueba de escritorio, permitirá obtener una buena base lógica en la construcción de un producto de software. Pero su mala

implementación puede comprometer gravemente el correcto funcionamiento del mismo.

3.5.3. Sobre las variables y operadores

Cuando usted empiece a crear un algoritmo, es importante definir las variables que puede visualizar en ese instante. No pierda tiempo analizando todas las variables que necesitará, empiece por las que puede identificar de manera inmediata y según vaya diseñando el algoritmo podrá identificar las faltantes y definirlas en la marcha.

En cuanto a los operadores lo primordial es dominar el uso de la agrupación de términos, además de tener claro que primero se hacen las multiplicaciones y divisiones; después, las sumas y restas. Este principio utiliza la mayoría de lenguajes de programación.

3.5.4. Sobre la representación de algoritmos

Una vez que se tiene implementado el algoritmo con su respectiva prueba de escritorio, es importante poder visualizar lo mismo, pero de manera gráfica, ya que es posible que se encuentre una solución aún más eficiente a la que se realizó por el simple hecho de verlo reflejado desde otro punto de vista.

La representación gráfica de un algoritmo se traduce en un diagrama de flujo, lo cual permite tener una óptica diferente del flujo de acciones que se está diseñando. Se ha demostrado que hay personas que entienden o captan mejor las ideas de manera gráfica; por lo tanto, si usted no está convencido del algoritmo que ha creado, a pesar de lograr el objetivo, es muy probable que pueda encontrar la manera de optimizarlo utilizando su representación gráfica.

Otro punto a favor de los diagramas de flujo es que la secuencia de acciones resulta simple, sencilla de leer y entender para la mayoría de programadores ya que se utilizan simbologías con acciones y significados ya establecidos.

Además de representar un algoritmo de manera gráfica, también es posible representarlo de manera textual. Pero ojo: no es cualquier texto; bajo mi óptica diría que es un texto técnico con estructuras y principios generales de programación. Su principal ventaja a futuro es disminuir la curva de aprendizaje de cualquier lenguaje de programación ya que se centra en desarrollar la lógica algorítmica y no en comprender los pormenores de un lenguaje de programación como tal.

Para finalizar, considero que es bueno conocer estas técnicas de representación de algoritmos no solo por ser de gran utilidad para entender los fundamentos de programación, sino también porque existen muchos autores de manuales y libros que los utilizan para representar conceptos técnicos relacionados con el desarrollo de software.

3.5.5. Sobre las decisiones

Es importante identificar cuándo y qué estructuras de decisión utilizar. Recuerde que se puede combinar varias estructuras para lograr el objetivo; además, recomiendo utilizar siempre que pueda la estructura de selección múltiple ya que ahorra código y permite entender el algoritmo de mejor manera.

3.5.6. Sobre los ciclos

De manera general, debe asegurarse de crear acciones que permitan terminar con un ciclo repetitivo; en caso contrario, se estaría enfrentando a un bucle infinito. Las diferentes estructuras que ha estudiado en este libro podrá encontrarlas en la mayoría de lenguajes de programación. La lógica se mantiene; lo único que cambiará es la sintaxis. Es cuestión de práctica para perfeccionar su uso y explotar su potencial.

4. Anexos

A continuación, se presenta una serie de ejercicios realizados por los estudiantes de primer nivel de la carrera de Desarrollo de Software. Espero sirva como ejemplo para estudiar y analizar el uso de las diferentes estructuras abordadas en este libro.

Debo mencionar que en el CD del libro se encuentran los archivos en DFD y PseInt para que pueda cargarlos en su máquina y ejecutarlos o modificarlos si así lo requiere.

1. Dadas las variables A, B y C, realizar un algoritmo que imprima en pantalla la suma de las tres notas y el promedio del estudiante.

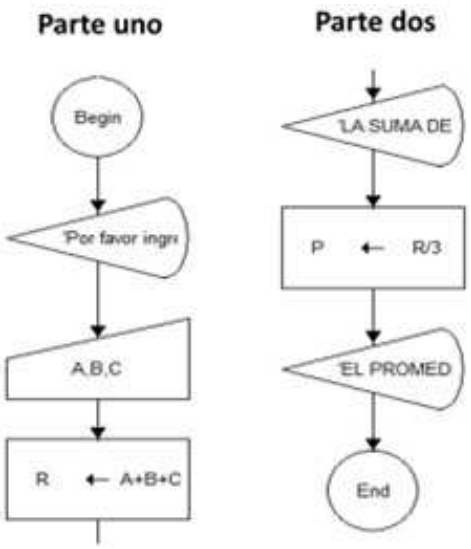


Figura 83. DFD Promedio ejemplo

Elaborado por Rafael Carpio (estudiante)

```
Proceso PROMEDIO
    Definir A,B,C,R,P Como Real;
    Escribir "ingrese 3 numeros";
    Leer A,B,C;

    R=A+B+C;
    Escribir "la suma de las notas es:", R;
    P=R/3;
    Escribir "El promedio del estudiantes es:", P;
FinProceso
```

Figura 84. PseInt Promedio ejemplo

Elaborado por Rafael Carpio (estudiante)

2. Dadas A = 10, B = 20 y C = 30, se desea saber el cuadrado de A y el cubo de B y C.

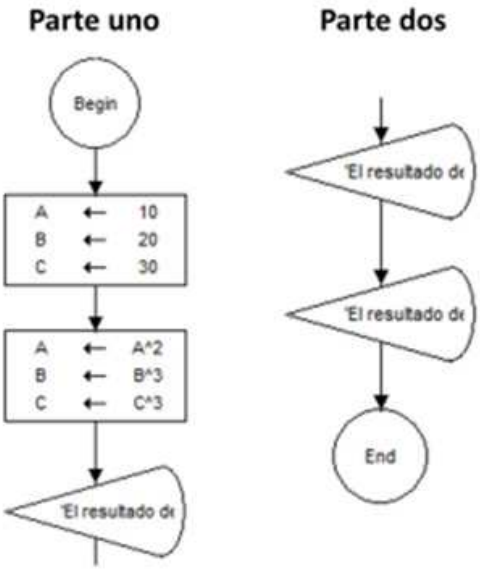


Figura 85. DFD Cuadrado y cubo ejemplo

Elaborado por Rafael Carpio (estudiante)

```
Proceso CUDRADOS_CUBOS
    Definir a,b,c como real;
    A = 10;
    B = 20;
    C = 30;
    A = A^2;
    Escribir "El cuadrado de A es:", A;
    B = B^3;
    Escribir "El cubo de B es:", B;
    C = C^3;
    Escribir "El cubo de C es:", C;
FinProceso
```

Figura 86. PseInt Cuadrado y cubo ejemplo

Elaborado por Rafael Carpio (estudiante)

3. Determinar la hipotenusa de un triángulo rectángulo aplicando el teorema de Pitágoras.

$$c = \sqrt{a^2 - b^2}$$

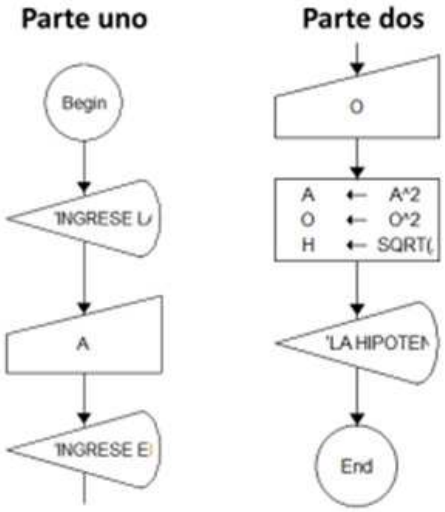


Figura 87. DFD Hipotenusa ejemplo

Elaborado por Rafael Carpio (estudiante)

```
Proceso Hipotenusa

    Definir A,B,C como Real;
    Escribir "Ingrese la adyacente";
    Leer A;
    Escribir "Ingrese el opuesto";
    Leer B;
    A = A^2;
    B = B^2;
    H =rc(A+B);
    Escribir "La hipotenusa es:", H;
FinProceso
```

Figura 88. PseInt Hipotenusa ejemplo

Elaborado por Rafael Carpio (estudiante)

4. Realizar un algoritmo que sume el cuadrado de:
   - A=A2+B2
   - B=B2+C2
   - C=A2+C2

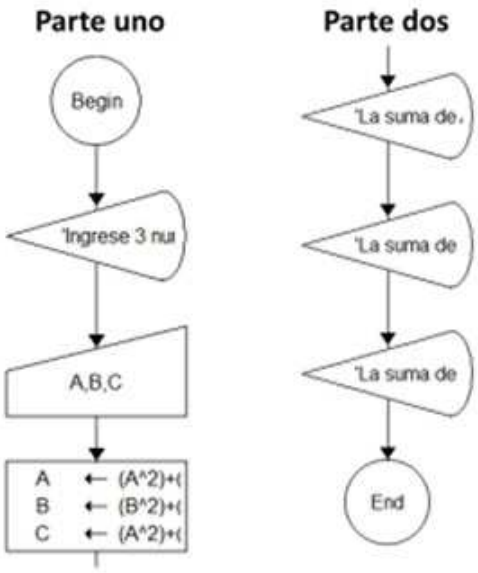


Figura 89. DFD Suma de cuadrados ejemplo

Elaborado por Rafael Carpio (estudiante)

```
Proceso SUMA_CUADRADOS

    Definir A,B,C como real;
    Escribir "Ingrese 3 numeros";
    Leer A,B,C;

    A=(A^2)+(B^2);
    Escribir "La suma de A es:",A;
    B=(B^2)+(C^2);
    Escribir "La suma de B es:",B;
    C=(A^2)+(C^2);
    Escribir "La suma de C es:",C;

FinProceso
```

Figura 90. DFD Suma de cuadrados ejemplo

Elaborado por Rafael Carpio (estudiante)

5. Realizar un algoritmo que sume, reste y multiplique el cuadrado de:
   - A=A2+B2
   - B=B2-C2
   - C=A2*C2

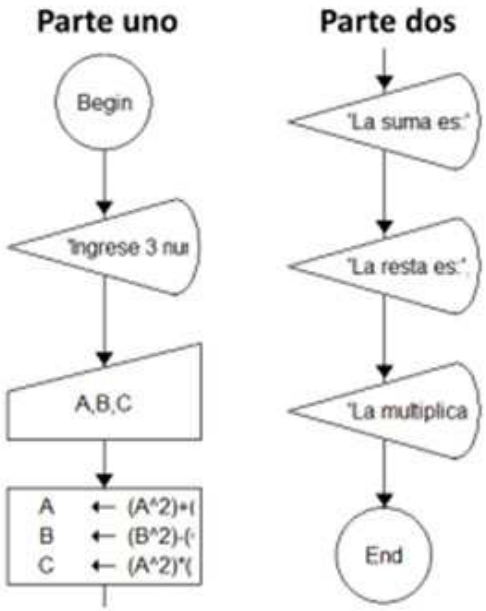


Figura 91. DFD Operaciones ejemplo

Elaborado por Rafael Carpio (estudiante)

```
Proceso OPERACIONES

    Definir A,B,C como real;
    Escribir "Ingrese 3 numeros";
    Leer A,B,C;

    A=(A^2)+(B^2);
    Escribir "La suma es: b",A;
    B=(B^2)-(C^2);
    Escribir "La resta es: ",B;
    C=(A^2)*(C^2);
    Escribir "La multiplicacion es: ",C;

FinProceso
```

Figura 92. PseInt Operaciones ejemplo

Elaborado por Rafael Carpio (estudiante)

6. Linealizar la siguiente expresión, el algoritmo debe solicitar el ingreso por teclado de las variables.

$$x = \frac{a}{b} * \frac{c}{d} - a$$

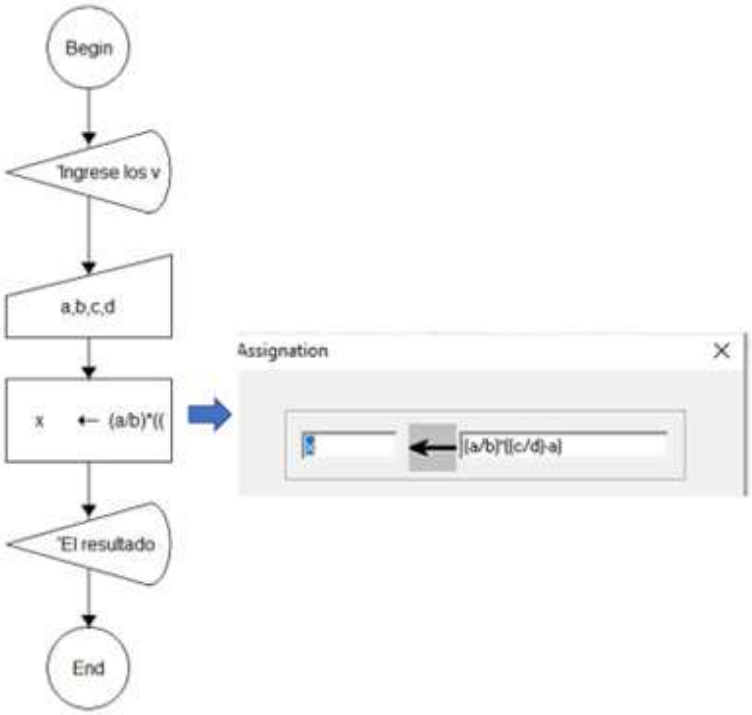


Figura 93. DFD Linealizar1 ejemplo

Elaborado por Kevin Simba (estudiante)

```
Proceso Ejercicio6

    definir a,b,c,d como entero;
    Escribir "ingrese laa variablea";
    leer a,b,c,d
    X=(a/b)*((c/d)-a)
    escribir "El resultado es ",X

FinProceso
```

Figura 94. PseInt Linealizar1 ejemplo

Elaborado por Kevin Simba (estudiante)

7. Linealizar la siguiente expresión, el algoritmo debe solicitar el ingreso por teclado de las variables.

$$x = \frac{\frac{a}{b} - \frac{c}{d}}{\frac{c}{d} + \frac{a}{b}}$$

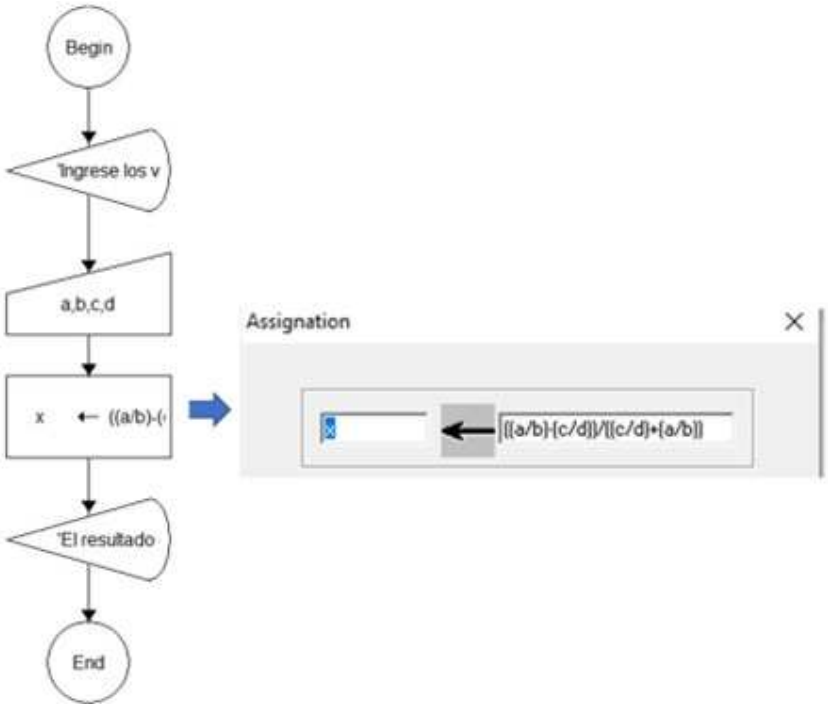


Figura 95. DFD Linealizar2 ejemplo

Elaborado por Kevin Simba (estudiante)

```
Proceso Ejercicio7

    Definir a,b,c,d como real;
    Leer a,b,c,d;
    escribir "Ingrese las variables";
    X=((a/b)-(c/d))/((c/d)+(a/b));
    escribir "El resultado es ",X;

FinProceso
```

Figura 96. PseInt Linealizar2 ejemplo

Elaborado por Kevin Simba (estudiante)

8. Linealizar la siguiente expresión, el algoritmo debe solicitar el ingreso por teclado de las variables.

$$x = \frac{a^b - \frac{c}{d} + \frac{a}{c}}{\frac{b-a}{(a-c)+(a^c)}}$$

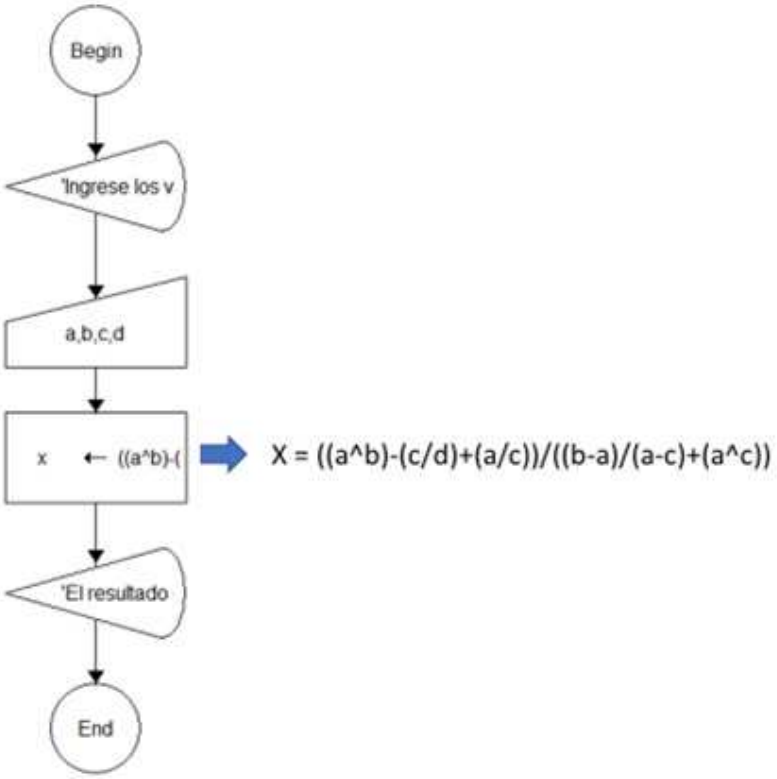


Figura 97. DFD Linealizar3 ejemplo

Elaborado por Kevin Simba (estudiante)

```
Proceso Ejercicio8

    Definir a,b,c,d como real
    Escribir "Ingrese las variables"
    leer a,b,c,d
    X=((a^b)-(c/d)+(a/c))/((b-a)/(a-c)+(a^c))
    Escribir "El resultado es ",X

FinProceso
```

Figura 98. PseInt Linealizar3 ejemplo

Elaborado por Kevin Simba (estudiante)

9. Linealizar la siguiente expresión, el algoritmo debe solicitar el ingreso por teclado de las variables.

$$x = \frac{a-c}{c*d} * \frac{a+c}{b-d}$$

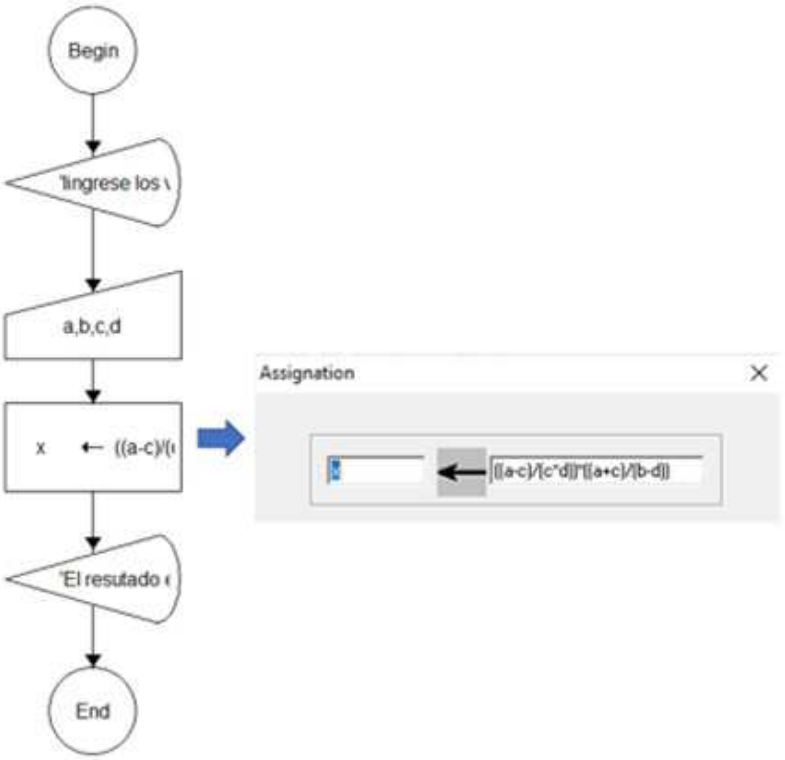


Figura 99. DFD Linealizar4 ejemplo

Elaborado por Kevin Simba (estudiante)

```
Proceso Ejercicio9

    definir a,b,c,d como real;
    leer a,b,c,d;
    x=((a-c)/(c*d))*((a+c)/(b-d));
    escribir "El resultado es ",x;

FinProceso
```

Figura 100. PseInt Linealizar 4 ejemplo

Elaborado por Kevin Simba (estudiante)

10. Linealizar la siguiente expresión, el algoritmo debe solicitar el ingreso por teclado de las variables.

$$x = \frac{(a+b+c)*(d-e-f)}{(a+e*f)*(\frac{a+e}{c})}$$

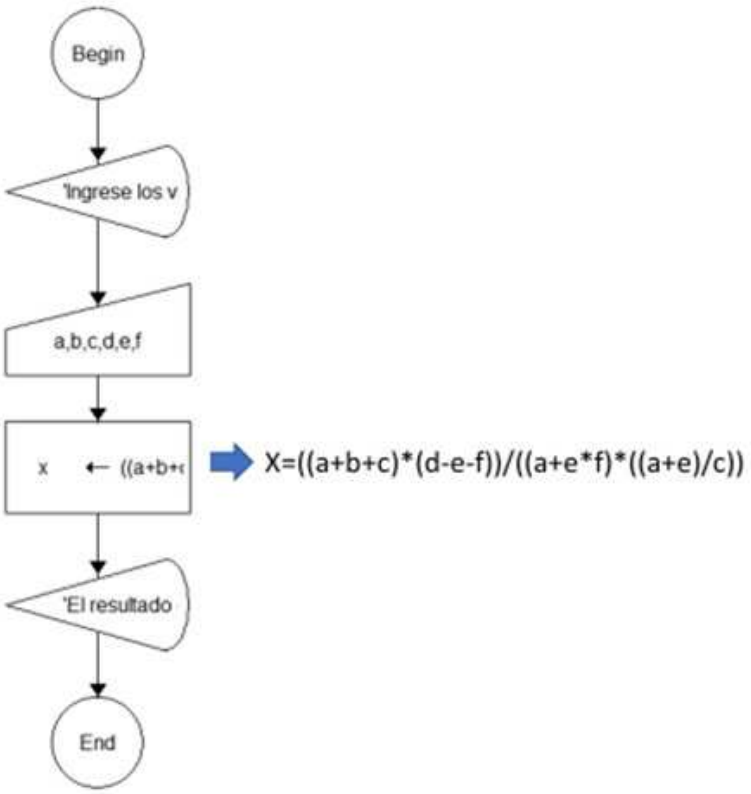


Figura 101. DFD Linealizar5 ejemplo

Elaborado por Kevin Simba (estudiante)

```
Proceso Ejercicio10

    Definir a,b,c,d,e,f como real;
    escribir "Ingrese los valores";
    Leer a,b,c,d,e,f;
    x=((a+b+c)*(d-e-f))/((a+e*f)*((a+e)/c));
    escribir "El resutado es ",x;

FinProceso
```

Figura 102. PseInt Linealizar5 ejemplo

Elaborado por Kevin Simba (Estudiante)

11. En un transcurrido hospital de la ciudad se requiere un algoritmo que permita agendar citas médicas. Para ello el paciente debe ingresar la siguiente información:
    - Nombre del paciente
    - Edad
    - Desea agendar cita (Sí: 1; No: 2)
    - Si el paciente ingresa la opción 1, se lo debe asignar automáticamente al departamento médico correspondiente según la edad del paciente.
    - Pediatría: 1-13 años
    - Medicina general: 14-50 años
    - Geriatría: 51-110 años

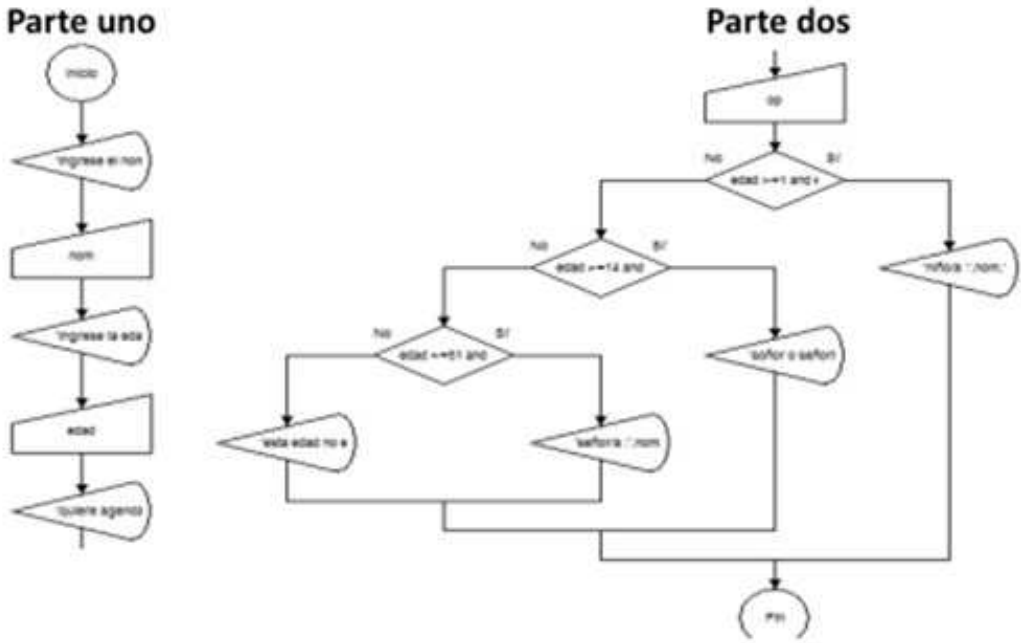


Figura 103. DFD Decisión1 ejemplo

Elaborado por Javier Armijos y Dayana Medina (estudiantes)

```
Proceso Ejercicio11
    Definir nom Como Caracter
    Definir edad,op Como Entero
    Escribir "ingrese el nombre del paciente";
    Leer nom
    Escribir "ingrese la edad";
    Leer edad
    Escribir "quiere agendar la cita 1 Si o 2 No"
    leer op;
    si op=1 entonces

    si edad >=1 & edad <=13 Entonces
        Escribir "niño/a :",nom, " de ",edad
        Escribir "su cita es para pediatria";

    Sino
        si edad >=14 & edad <=50 Entonces
            Escribir "señor/a :",nom, " de ",edad
            Escribir "su cita es para medicina general";
        Sino
            si edad >=51 & edad <=110 Entonces
                Escribir "señor/a :",nom, " de ",edad
                Escribir "su cita es para geriatria";
            sino
                Escribir "esta edad no existe"
        FinSi
    FinSi

    FinSi
Sino
    Escribir "intente nuevamente ";
FinSi

FinProceso
```

Figura 104. PseInt Decisión1 ejemplo

Elaborado por Javier Armijos y Dayana Medina (estudiantes)

12. Realice un algoritmo que solicite el ingreso de estos datos: año de nacimiento, mes de nacimiento, año actual y mes actual; y devuelva el total de años y meses que tiene actualmente.

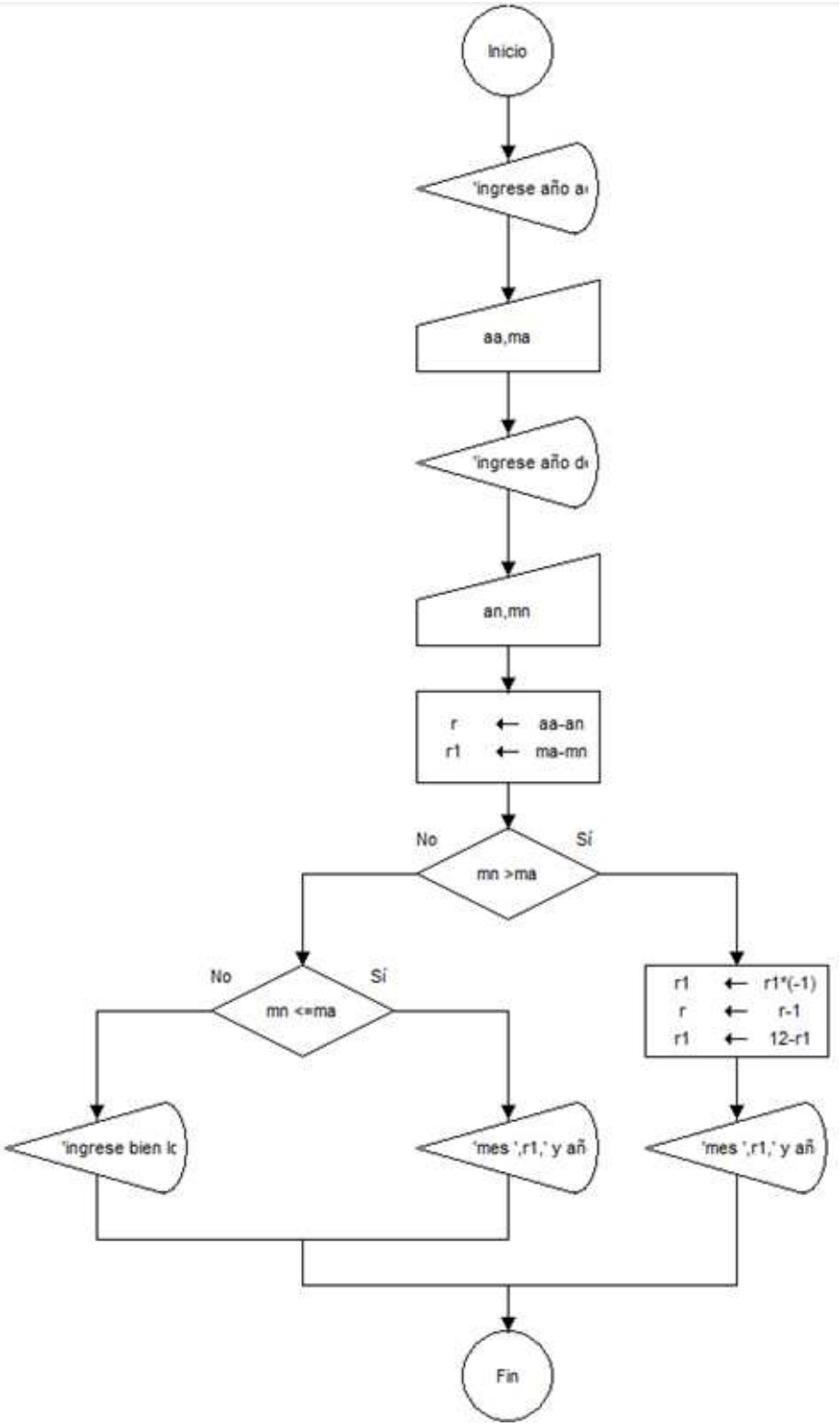


Figura 105. DFD Decisión2 ejemplo

Elaborado por Javier Armijos y Dayana Medina (estudiantes)

```
Proceso Ejercicio12
    Definir nom,ape Como Caracter
    definir aa,ma,an,mn,r,r1,r2 como entero
    ESCRIBIR "ingrese el año actual";
    leer aa;
    ESCRIBIR "ingrese el mes actual";
    leer ma;
    ESCRIBIR "ingrese el año naciemiento";
    leer an;
    ESCRIBIR "ingrese el mes nacimiento";
    leer mn;
    r= aa-an;
    r1 = ma-mn;

    si(mn > ma) Entonces
        r1 = r1*(-1);
        r=r-1;
        r1=12-r1;

        escribir "mes",r1,"y año",r
    Sino
        si (mn <= ma) entonces
            escribir  "años",r,"y mese",r1;
        sino
            escribir "ingrese bien los datos";
        FinSi

    FinSi
FinProceso
```

Figura 106. PseInt Decisión2 ejemplo

Elaborado por Javier Armijos y Dayana Medina (estudiantes)

13. En una conocida escuela de la ciudad se requiere elaborar un algoritmo que permita ingresar los 4 aportes de un estudiante y se elimine la calificación más baja. Se debe mostrar como resultado el promedio final y la calificación que fue eliminada.

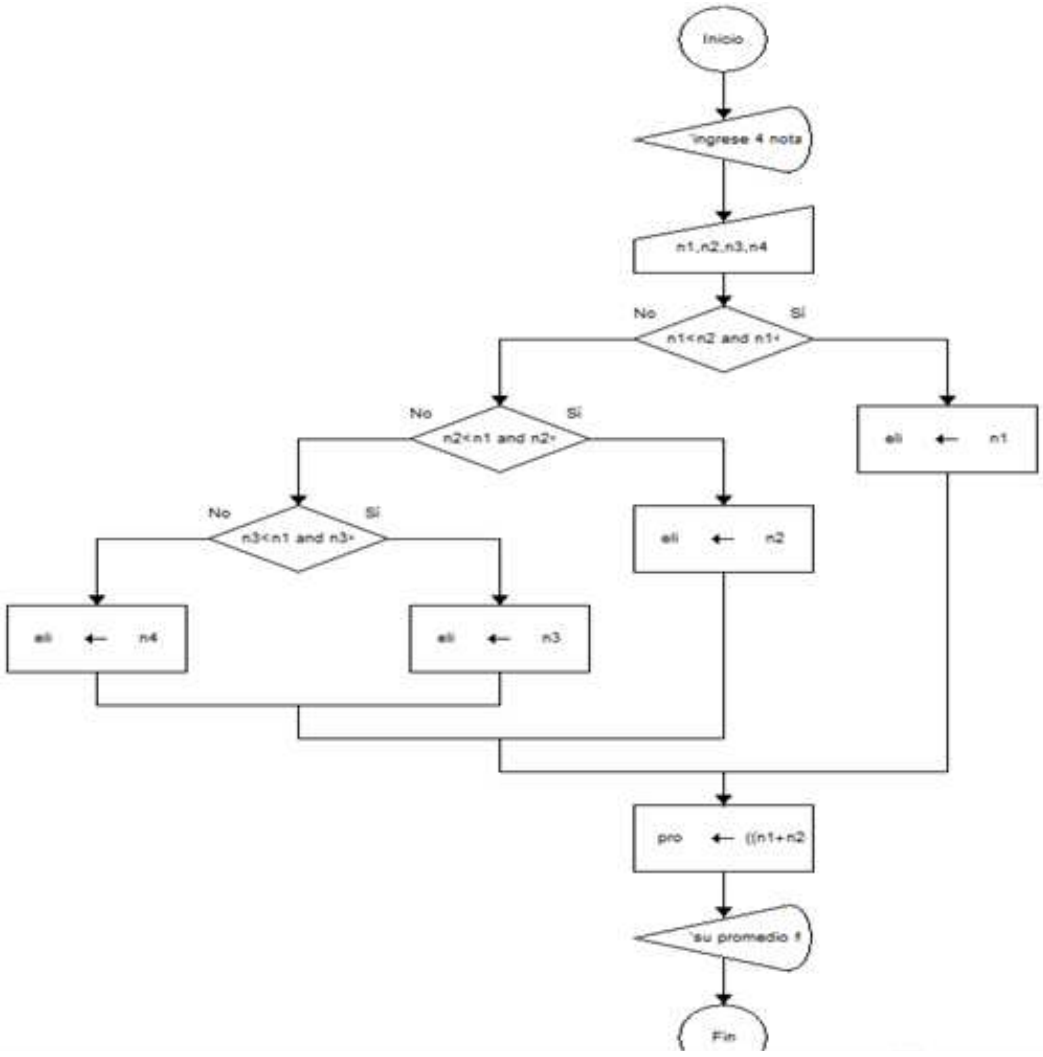


Figura 107. DFD Decisión3 ejemplo

Elaborado por Javier Armijos y Dayana Medina (estudiantes)

```
Proceso Ejercicio13
    Definir n1,n2,n3,n4,pro,eli Como Real
    escribir "diguite las 4 notas "
    Leer n1,n2,n3,n4;
    si n1<n2 & n1<n3 & n1< n4 Entonces
    |   eli=n1
    Sino
    |   si n2<n1 & n2<n3 & n2< n4 Entonces
    |   |   eli=n2
    |   Sino
    |   |   si n3<n1 & n3<n2 & n3< n4 Entonces
    |   |   |   eli=n3
    |   |   Sino
    |   |   |   eli=n4
    |   |   FinSi
    |   FinSi
    FinSi
    pro=((n1+n2+n3+n4)-eli)/3
    escribir "su promedio fue",pro "y la nota eliminada fue ",eli;
FinProceso
```

Figura 108. PseInt Decisión3 ejemplo

Elaborado por Javier Armijos y Dayana Medina (estudiantes)

14. Desarrollar un algoritmo que permita calcular la raíz cuadrada de un número ingresado por teclado. Se debe tener en cuenta que no se puede sacar raíces de números negativos.

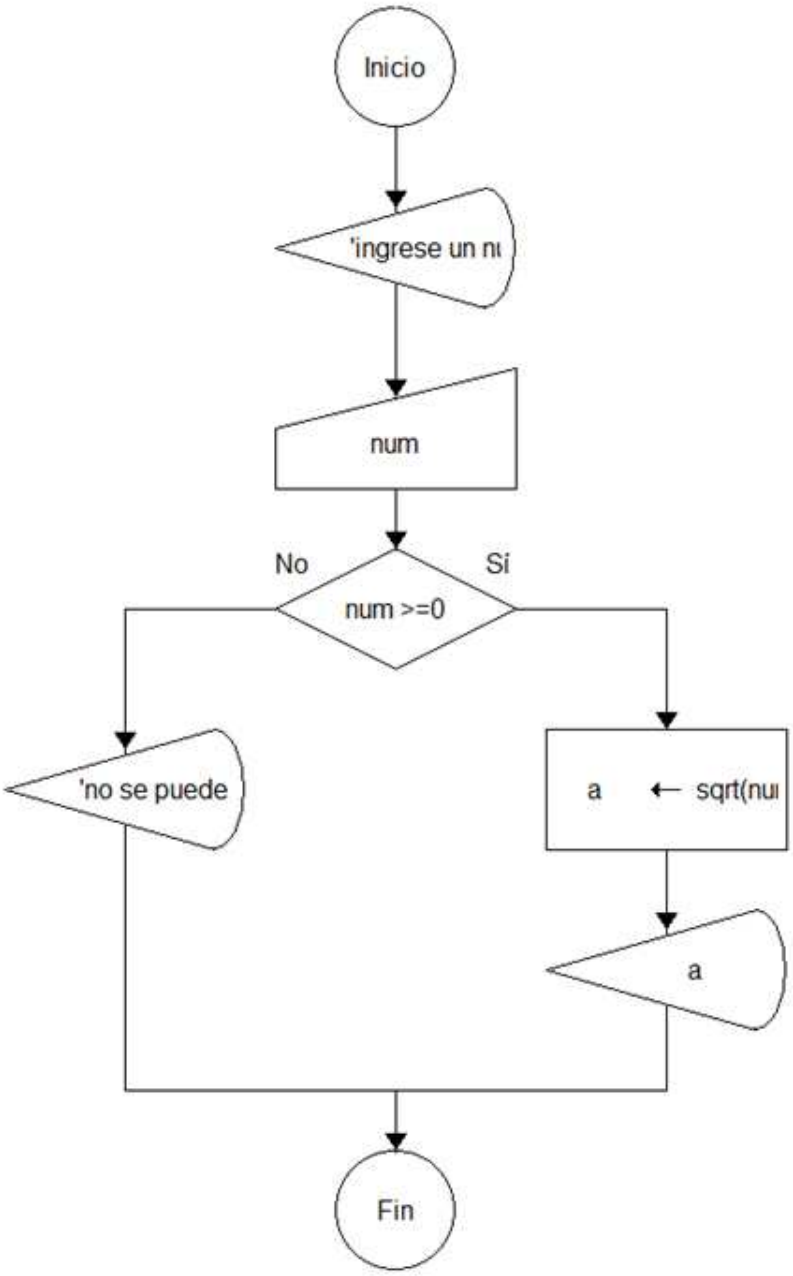


Figura 109. DFD Decisión4 ejemplo

Elaborado por Javier Armijos y Dayana Medina (estudiantes)

```
Proceso Ejercicio14
    Definir a, num Como Real
    Escribir "ingrese un numero";
    Leer num;
    si num >=0 entonces
        a= raiz(num);
        escribir a;
    Sino
        escribir "no se puede raiz negativa";
    finsi

FinProceso
```

Figura 110. PseInt Decisión4 ejemplo

Elaborado por Javier Armijos y Dayana Medina (estudiantes)

15. Elaborar un algoritmo que solicite el ingreso de una marca de vehículo y muestre su información.

```
Proceso Ejercicio15

    DEFINIR A como caracter


    Escribir 'ingrese la marca del vehiculo'
    leer A


    si (A='chevrolet')entonces
    |   escribir "chevrolet tiene Transformamos un icono para crear el Chevr
    Sino
    |
    |   si(A='toyota') Entonces
    |   |   escribir "toyota tiene un diseño proactivo y versátil, carroceri
    |   Sino
    |   |
    |   |   si (A='nissan') Entonces
    |   |   |   escribir "nissan tiene La tecnologia del Nuevo Nissan Versa
    |   |   Sino
    |   |   |   escribir 'no existe esa marca'
    |   |   FinSi
    |   |
    |   FinSi
    FinSi

FinProceso
```

Figura 111. PseInt Decisión5 ejemplo

Elaborado por Javier Armijos y Dayana Medina (estudiantes)

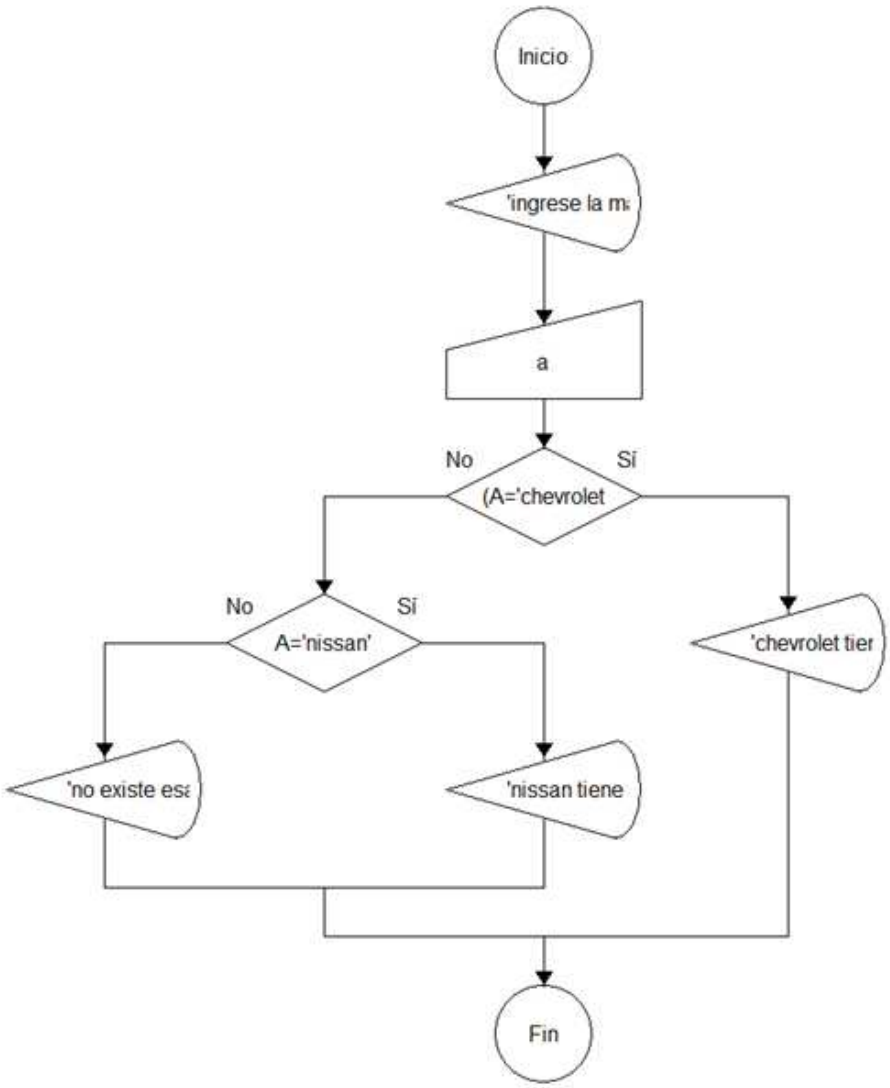


Figura 112. PseInt Decisión5 ejemplo

Elaborado por Javier Armijos y Dayana Medina (estudiantes)

16. Realizar un algoritmo que permita calcular: área del rombo, área del trapecio, perímetro del triángulo. En cada caso se deben ingresar los valores necesarios para dicho cálculo.

```
Proceso Ejercicio16
    Definir a Como Entero
    Definir d1,d2,r,h Como Real

    Escribir "ingrese una opcion del 1 al 3";
    escribir "1 : AREA DEL ROMBO";
    escribir "2 : AREA DEL TRAPECIO";
    escribir "3 : PERIMETRO DEL TRIANGULO";
    leer a;

    segun a hacer
        1:
            escribir "ingrese las diagonales de el rombo";
            leer d1,d2;
            Limpiar Pantalla
            r=(d1*d2)/2;
            escribir r;
        2:
            escribir "ingrese las bases y la altura de el trapecio";
            leer d1,d2,h;
            Limpiar Pantalla
            si(d1<>d2)entonces
                r=(d1+d2)/2;
                r=r*h;
                escribir r;
            Sino
                escribir "las bases deben ser diferentes";

            FinSi
        3: Escribir "ingrese los lados del triangulo"
            Leer d1,d2,h;
            Limpiar Pantalla
            r=d1+d2+h;
            Escribir r;
    FinSegun
FinProceso
```

Figura 113. DFD Selección Múltiple1 ejemplo

Elaborado por Javier Armijos y Dayana Medina (estudiantes)

17. Elaborar un algoritmo que permita realizar reservaciones para las vacaciones según la siguiente información:
    - Costa - (Esmeraldas y Tonsupa)
    - Sierra - (Quito y Ambato)
    - Oriente - (Tena)
    - Región Insular - (Islas Isabela y Santa Cruz)

```
Proceso Ejercicio17
    Definir a,a1,p Como Entero

    Escribir "ingrese opciones para ide de vacaciones ";
    Escribir "1: COSTA";
    Escribir "2: SIERRA";
    Escribir "3: ORIENTE";
    Escribir "4: REGION INSULAR";
    leer a;

    segun a hacer
        1:
            Escribir "desea realizar la reservacion 1 Si o 2 No";
            leer p;

            si(p = 1) entonces;
               escribir "1: ESMERALDAS";
               Escribir "2: TONSUPA";
               leer a1
               segun a1 hacer
                   1:
                      Escribir "SU RESERVACION DE ESMERALDAS FUE UN EXITO LO ESPERAMOS "
                   2:
                      Escribir " SU RESERVACION DE TONSUPA FUE UN EXITO LO ESPERAMOS "
               FinSegun
               escribir "su reserva esta echa para la costa ";
            Sino
               escribir "intente nuevamente ";

            FinSi
        2:
            Escribir "desea realizar la reservacion 1 Si o 2 No";
            leer p;

            si(p = 1) entonces;
               escribir "1: QUITO";
               Escribir "2: AMBATO";
               leer a1
               segun a1 hacer
                   1:
                      Escribir "SU RESERVACION DE QUITO FUE UN EXITO LO ESPERAMOS "
                   2:
                      Escribir " SU RESERVACION DE AMBATO FUE UN EXITO LO ESPERAMOS "
               FinSegun
               escribir "su reserva esta echa para la CIERRA ";
            Sino
               escribir "intente nuevamente ";

            FinSi
        3:
            Escribir "desea realizar la reservacion 1 Si o 2 No";
            leer p;

            si(p = 1) entonces;
               escribir "1: TENA";
               Escribir "2: BAÑOS";
               leer a1
               segun a1 hacer
                   1:
                      Escribir "SU RESERVACION DEL TENA FUE UN EXITO LO ESPERAMOS "
                   2:
                      Escribir " SU RESERVACION DE BAÑOS FUE UN EXITO LO ESPERAMOS "
               FinSegun
               escribir "su reserva esta echa para el oriete ";
            Sino
               escribir "intente nuevamente ";

            FinSi
        4:
            Escribir "desea realizar la reservacion 1 Si o 2 No";
            leer p;

            si(p = 1) entonces;
               escribir "1: ISLA ISABELA";
               Escribir "2: ISLA SANTA CRUZ";
               leer a1
               segun a1 hacer
                   1:
                      Escribir "SU RESERVACION PARA LA ISLA ISABELLA FUE UN EXITO LO ESPERAMOS "
                   2:
                      Escribir " SU RESERVACION DE LA ISLA SANTA CRUZ FUE UN EXITO LO ESPERAMOS "
               FinSegun
               escribir "su reserva esta echa para la region insular ";
            Sino
               escribir "intente nuevamente ";

            FinSi

    FinSegun
FinProceso
```

Figura 114. PseInt Selección Múltiple2 ejemplo

Elaborado por Javier Armijos y Dayana Medina (estudiantes)

18. Un profesor del IST CEMLAD desea saber el nombre del estudiante ingresando su respectivo código, de esta manera, se tiene el siguiente registro:
    1 - Alejandro
    2 - Anita
    3 - Patricio
    4 - Guillermo
    5 - Laia
    Dado un código por parte del docente, se debe mostrar el nombre del estudiante al que corresponde.

```
Proceso Ejercicio18
    Definir codigo Como Entero;
    Escribir "Ingrese el código del estudiante";
    leer codigo;

    Segun codigo Hacer
        1:
            Escribir "ALEJANDRO";
        2:
            Escribir "ANITA";
        3:
            Escribir "PATRICIO";
        4:
            Escribir "GUILLERMO";
        5:
            Escribir "LAIA";
        De Otro Modo:
            Escribir "NO SE ENCUENTRA EN LA LISTA DE ESTUDIANTES"
    FinSegun
```

Figura 115. PseInt Selección Múltiple3 ejemplo

Elaborado por Alejandro Herrera (estudiante)

19. Realizar un algoritmo que permita conocer si un estudiante pasa el año, tiene supletorio, tiene un examen de gracia o pierde el año. Sabiendo que: con notas de 0 hasta 4 el estudiante pierde el año; con 5 el estudiante tiene la opción de un examen de gracia; con 6 el estudiante tiene la opción de examen remedial; con 7 el estudiante tiene la opción de un examen supletorio; y con 8, 9 y 10 el estudiante pasa el año.

```
Proceso Ejercicio19
    Definir N Como Entero;
    Escribir "ingrese una calificacion del 1 al 10";
    Leer N;
    Segun N Hacer
        0,1,2,3,4:
            Escribir "El estudiante pierde el año";
        5:
            Escribir "El estudiante se queda a examen de gracia";
        6:
            Escribir "El estudiante se queda a examen remedial";
        7:
            Escribir "El estudiante se queda a examen supletorio";
        8,9,10:
            Escribir "El estudiante aprueba el año";;
        De otro modo:
            Escribir "la nota no existe";
    FinSegun
FinProceso
```

Figura 116. PseInt Selección Múltiple4 ejemplo

Elaborado por Rafael Carpio (estudiante)

20. En la ciudad de Lima una agencia de seguros para automóviles asigna costos basados en el sexo y la edad del conductor: los varones menores de 25 años pagan los precios más altos, 1.000 dólares; los hombres de 25 años o más pagan 700 dólares; las mujeres de menos de 21 años o más pagan 500 dólares. Escribir el algoritmo del programa que imprima la edad del conductor, sexo y el pago correspondiente para los clientes de la aseguradora.

```
Proceso Ejercicio20

    Definir sexo Como Caracter;
    Definir edad, interruptor Como Entero;
    definir costo como real;


    escribir 'ingrese el sexo (H/M)';
    leer sexo;
    escribir 'ingrese la edad';
    leer edad;

    Si (sexo = 'H') entonces
    |   interruptor <- 1;
    Sino
    |   interruptor <-2;
    FinSi

    Segun interruptor hacer
    |   1:
    |   |   si edad < 25 entonces
    |   |   |   costo =1000;
    |   |   Sino
    |   |   |   costo =700;
    |   |   FinSi
    |   |
    |   2:
    |   |
    |   |   si (edad <> 21 )&(edad=21) entonces
    |   |   |   costo =500;
    |   |
    |   |   FinSi
    |   |
    FinSegun

    Escribir 'EDAD : ',EDAD;
    Escribir 'SEXO : ',SEXO;
    Escribir 'PAGO : ',COSTO;
FinProceso
```

Figura 117. PseInt Selección Múltiple5 ejemplo

Elaborado por Dayana Medina

21. Realice un algoritmo que permita calcular la media de gastos realizados de una cantidad N de meses ingresada por el usuario.

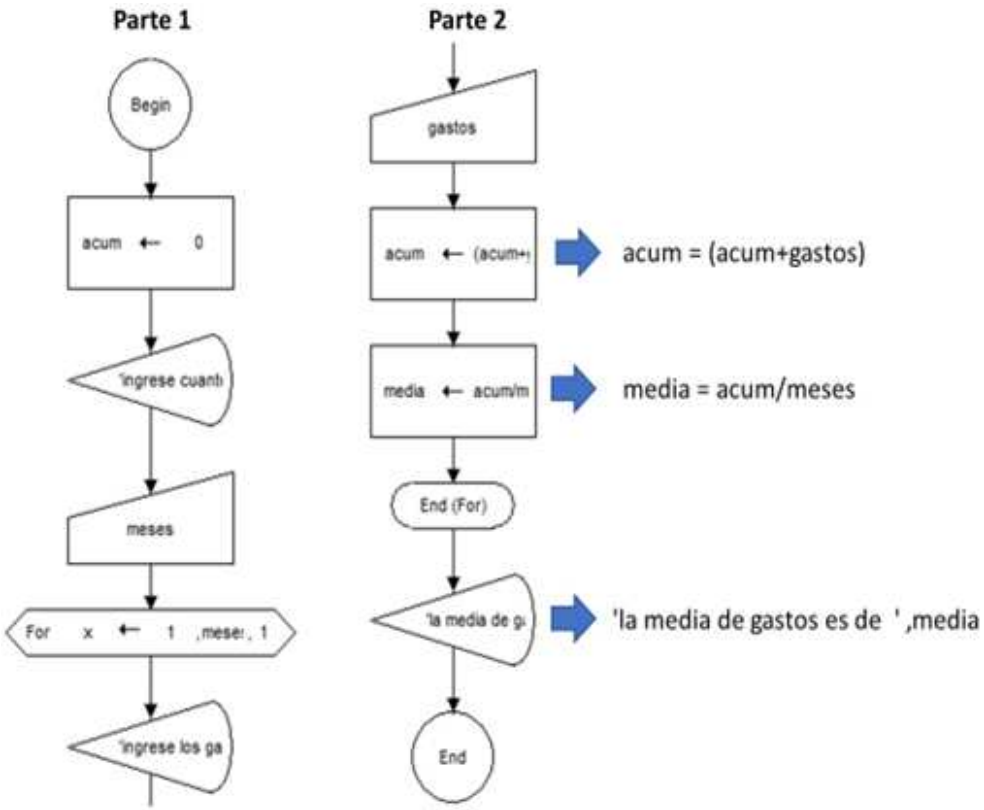


Figura 118. DFD Ciclo 1 ejemplo

Elaborado por Dayana Medina (estudiante)

```
Proceso mese_gastos
    definir meses,gasto,acum,x,media como real
    acum=0
    escribir 'ingrese cuantos meses quere calcular la media ';
    Leer meses
    Para x<-1 hasta meses con paso 1 hacer
        escribir 'ingrese los gastos para el mes ',x
        leer gasto
        acum=acum+gasto
    FinPara
    media =acum/meses;
    escribir 'la media de gastos es de ', media;

FinProceso
```

Figura 119. PseInt DFD Ciclo 1 ejemplo

Elaborado por Dayana Medina (estudiante)

22. Elaborar un algoritmo que permita ingresar e imprimir el nombre y apellido de N personas. El valor de N debe ser ingresado por el usuario.

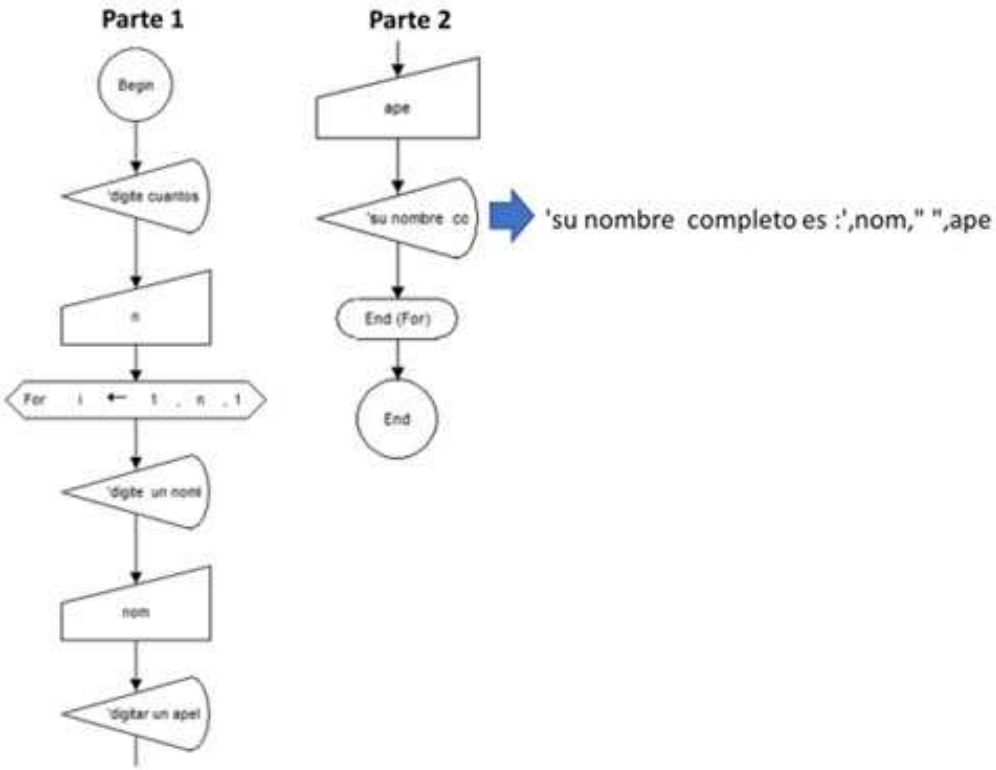


Figura 120. DFD Ciclo 2 ejemplo

Elaborado por Jennifer Molina (estudiante)

```
Proceso NOMBRES_apellidos
    Definir nom,ape Como caracter ;
    definir n , i como entero;
    Escribir 'DIGITE CUANTOS NOMBRES DESEA INGRESAR';
    Leer n;
    Para i<-1 Hasta n con paso 1 Hacer
        Escribir 'digitar un nombre';
        Leer nom;
        Escribir 'digite un apellido';
        Leer ape;
        Escribir 'su nombre completo es : ' , nom ,' ', ape;
    FinPara
FinProceso
```

Figura 121. PseInt Ciclo 2 ejemplo

Elaborado por Jennifer Molina (estudiante)

23. Elaborar un algoritmo que permita ingresar N números enteros. Se debe preguntar al usuario si desea o no continuar con el ingreso. Cuando el usuario ingrese "no", se debe mostrar la suma de todos los números ingresados.

```
Proceso Ejercicio23
    n<-0;
    acum<-0;

    repetir
        escribir "Ingrese un numero";
        leer n;

        acum=acum+n;

        Escribir "Decea continuar SI/NO";
        Leer r;
    Hasta Que r="no"
        escribir "La suma de los numeros ingresados es:  ",acum;
FinProceso
```

Figura 122. PseInt Ciclo 3 ejemplo

Elaborado por Kevin Simba (estudiante)

24. Dadas las siguientes credenciales: usuario = "javier" y clave = 1234, elaborar un algoritmo que solicite el ingreso de usuario y clave con las siguientes condiciones: si el usuario ingresa correctamente las credenciales, debe aparecer un mensaje de bienvenida; el usuario podrá intentar ingresar máximo tres veces; si supera ese valor, debe aparecer el mensaje: "sobrepasó el número de intentos".

```
Proceso Ejercicio24
    Definir nom como caracter
    definir clave,int como entero
    Escribir "ingrese nombre de usuario"
    leer nom
    Escribir "ingrese clave"
    Leer clave
    int=1
    Hacer
        si nom = "javier" y clave = 1234 entonces
            Escribir "biemvenido"
            int = 3
        Sino
            escribir "ingrese de nuevo"
            int =int+1
            Escribir "ingrese nombre de usuario"
            leer nom
            Escribir "ingrese clave"
            Leer clave
            si int =3 Entonces
                escribir "sobrepaso el numero de intentos"
            FinSi
        FinSi
    Hasta Que int = 3
FinProceso
```

Figura 123. PseInt Ciclos 4 ejemplo

Elaborado por Javier Armijos

25. Elaborar un algoritmo que permita imprimir las tablas de multiplicar del 1 al 12.

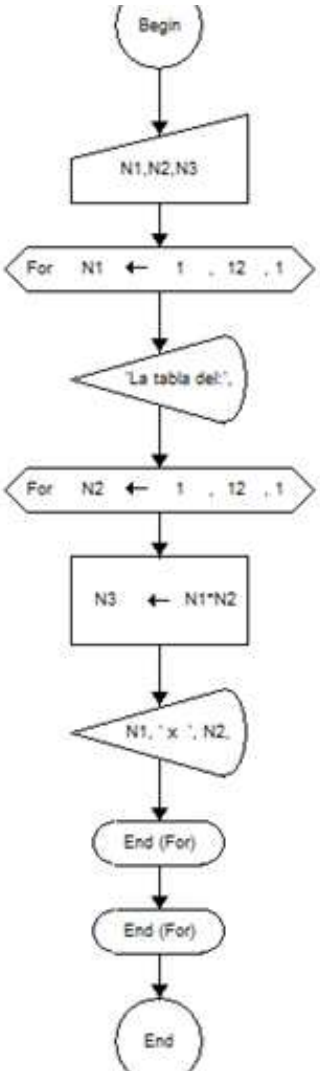


Figura 124. DFD Ciclos 5 ejemplo

Elaborado por Rafael Carpio

```
Proceso Ejercicio25
    Definir N1,N2,N3 Como Entero;
    Para N1 <-1 hasta 12 Con Paso 1 Hacer
        Escribir "la tabla del:" , N1;
        Para N2 <-1 hasta 12 Con Paso 1 Hacer
            N3<-N1*N2;
            Escribir N1, " x ", N2, " = ", N3;
        FinPara
    FinPara
FinProceso
```

Figura 125. PseInt Ciclo 5 ejemplo

Elaborado por Rafael Carpio

GUIA DIDÁCTICA DE LA LÓGICA DE PROGRAMACIÓN

Por Carlos Salazar

Modelo Pedagógico del IST CEMLAD

Según (Parra, 2020) el IST CEMLAD, en su modelo de desarrollo, apuesta por un proceso de emprendimiento en la dimensión humana como centro de aprendizaje y promoción profesional; el desarrollo a escala humana se plantea desde la perspectiva del Dr. Max Neff (premio Nobel en Economía). Este modelo propone un tipo de desarrollo basado en las personas y no en los objetos; en otras palabras, este desarrollo se orienta en la satisfacción de las necesidades humanas fundamentales, por ejemplo: alimentación, vivienda, educación, etc. Además, se propone la división de estas necesidades en ontológicas y axiológicas.

En cuanto a las necesidades ontológicas se puede encontrar: ser, tener, hacer y estar. Por otro lado, las necesidades axiológicas tienen que ver con: subsistencia, protección, afecto, entendimiento, participación, ocio, creación, identidad y libertad. Este conjunto de necesidades básicas, según el Dr. Max Neff, todo ser humano debería poder cumplirlas y buscar la forma de satisfacerlas.

Asimismo, el IST CEMLAD a través de la agenda académica, promueve el desarrollo a escala humana con base en la propuesta de Manfred Max-Neff, en diálogo continuo con los contextos a los cuales sirve en los procesos de emprendimiento, gestión social y empresarial a los que contribuye.

Educación por competencias

De acuerdo con (Parra, 2020), la educación basada en competencias se fundamenta en un currículum que se apoya en las competencias de manera integral y en la resolución de problemas. Utiliza recursos que simulen la vida real: análisis y resolución de problemas, que aborda de manera integral; trabajo cooperativo o por equipos, favorecido por tutorías.

El concepto de competencia es multidimensional e incluye distintos niveles como: saber (datos, conceptos, conocimientos), saber hacer (habilidades, destrezas, métodos de actuación), saber ser (actitudes y valores que guían el comportamiento) y saber estar (capacidades relacionadas con la comunicación interpersonal y el trabajo cooperativo). En otras palabras, la competencia es la capacidad de un buen desempeño en contextos complejos y auténticos; se basa en la integración y activación de conocimientos, habilidades, destrezas, actitudes y valores.

Competencias por niveles de aprendizaje

En base de (Parra, 2020), los participantes de las carreras del IST CEMLAD, logran competencias importantes para su vida profesional. A más de su crecimiento personal, con identidad para el desarrollo, también promovemos con espíritu emprendedor las competencias necesarias para promoverse como sujeto social, y que tienen que ver desde lo concreto hasta lo abstracto en la medida que logran su desempeño. Superando la complejidad de los contextos, asumimos la propuesta de Benjamín Bloom (1913-1999) como dominios del aprendizaje que para nosotros son consideradas competencias de los participantes.

(EDUTEC, 2013) plantea un conjunto de actividades según el nivel de aprendizaje basado en competencias para lograr los objetivos deseados.

A continuación, se listan y ordenan actividades según la habilidad de pensamiento requerido para su ejecución.

1. Recuerdo. Inicio del conocimiento	
Seguir el discurso del profesor. Seguir instrucciones. Recordar. Explorar la herramienta. Hacer lectura de manuales, visionar tutoriales, etc. Explorar ejemplos de aplicación. Hacer búsquedas guiadas de material relacionado con la herramienta o con el uso de la herramienta.	
2. Comprensión. Cualquier grado de comprensión	
Hacer uso inicial de una herramienta u observar su uso. Hacer cualquier práctica inicial en sesiones de aula o en actividades de enseñanza aprendizaje. Encontrar un uso posible a la herramienta. Recopilar y ordenar el material descubierto. Hacerse una primera idea de la herramienta y sus posibles usos. Relacionar la herramienta y su uso con aprendizajes o saberes anteriores.	
3. Uso. Utilización de las herramientas (Aplicación)	
3.1. Guiado	Hacer una aplicación puntual con instrucciones y guía. Resolver ejercicios aplicativos. Elaborar trabajos pautados: trabajos fin de curso, proyectos finales, etc.
3.2. Consciente intencionado	Aplicar de manera consciente, querida, intencionada estructurada y de forma o propuesta autónoma. Hacer una aplicación particular o colectiva Incorporar las herramientas, aplicaciones o programas a proyectos de aula y/o personales. Transferir el conocimiento a otro entorno, situación, proyecto, ejemplo, etc.
4. Análisis. Realización de cualquier tipo de análisis	
4.1. Sencillo	Hacer comparaciones y/o estudios de una herramienta o aplicación. Extraer características y/o diferencias. Hacer comparaciones y/o estudios de experiencias que han usado o incluido la herramienta TIC. Hacer comparaciones y/o estudios de contenidos de área implicados en herramientas o aplicativos concretos. Hacer comparaciones y/o estudios de experiencias que integran el uso de una herramienta TIC. Hacer comparaciones y/o estudios de experiencias TIC relacionadas con contenidos de área.
4.2. Medio	Hacer estudios comparativos entre herramientas similares, combinación de herramientas y/o herramientas complejas. Hacer comparaciones, extraer las características y/o la descripción del potencial pedagógico de las herramientas o aplicaciones. Identificar los requerimientos de uso, ventajas y/o desventajas observados. Identificar los criterios que conducen a la determinación de características selectivas y a la contextualización de posibles aplicaciones concretas de la herramienta. Hacer análisis de experiencias educativas que aborden el estudio de las herramientas TIC combinadas con el análisis del contenido de un área de conocimiento o la combinación de las herramientas TIC con el proceso pedagógico donde han sido incluidas.

4.3. Complejo	Hacer análisis de escenarios o de situaciones de aprendizaje con el uso integrado de TIC y/o 3.0. Hacer estudios detallados de las herramientas TIC o de aplicativos 3.0 de contenidos de área y de las competencias definidas en los objetivos de la propuesta pedagógica. Hacer estudios, en escenarios de enseñanza con integración de TIC, del proceso de secuenciación de las tareas, la construcción de conocimiento; el modelo didáctico usado, el rol de los estudiantes y del profesor; el tipo de materiales facilitados; el tratamiento del grupo, aprendizaje colaborativo y la gestión colectiva de conocimiento. Elaboración de informes descriptivos.
5. Síntesis. Uso con conciencia plena y metacognición	
Evaluar proyectos y elaborar argumentaciones con la concreción e inclusión de criterios personales. Hacer análisis de situaciones personales: ¿Qué puedo hacer? ¿Cómo uso las herramientas? ¿Qué plan de ruta elaboro? Elaboración de informes comparativos.	
6. Evaluación. Cualquier acción de evaluación por criterios	
Realizar actividades que impliquen la definición de criterios y elaboración de juicios razonados. Realizar evaluaciones que conduzcan a la concreción de criterios personales. Elaborar informes comparativos y evaluativos por criterios	
7. Creación. Diseño de itinerarios de aprendizaje con integración de TIC y 3.0	
7.1 Nivel sencillo	Integrar una herramienta en un periodo temporal corto (máximo un mes) de uso individual.
7.2. Nivel medio	Combinar herramientas en escenarios colaborativos en periodos de media duración (un trimestre).
7.3. Nivel complejo	Realizar proyectos de gran abasto.
Colaboración. (Categoría paralela a las anteriores)	
Sencilla	En grupo, puesta en común puntual.
Media	Suma de trabajos individuales y revisión final.
Compleja	Construcción colaborativa, con debate, propuestas, criterios consensuados y argumentos compartidos.

Estrategias de enseñanza – aprendizaje

De acuerdo con (SERVICIO NACIONAL DE APRENDIZAJE (SENA), 2003), se pueden clasificar las estrategias de enseñanza-aprendizaje según el énfasis establecido al interior de cada una de las estrategias en el proceso educativo. De esta manera, se pueden clasificar en:

- Los sujetos (docente y estudiante).
- El proceso o las mediaciones didácticas.
- Los objetos de conocimiento.

Las estrategias con énfasis en los **sujetos** tienen dos categorías, por un lado, las enfocadas en el alumno y las centradas en los docentes. A continuación, se puede observar las principales características sobre estos enfoques:

Centradas en el estudiante	Centrada en los docentes
El docente toma un rol pasivo y el estudiante un rol activo.	El docente toma un rol activo y el estudiante un rol pasivo.
Utiliza como técnicas de enseñanza: el método de problemas, el método de situaciones (o de casos), el método de indagación, la tutoría, la enseñanza por descubrimiento, el método de proyectos.	Utiliza como técnicas de enseñanza: clase expositiva, demostración, lectura dirigida, enseñanza tradicional, etc.
En la educación universitaria estimula la socialización, ayuda a que el estudiante comprenda y respete a los otros, interactúa en clase, permite un aprendizaje profundo y permanente, propicia el desarrollo de habilidades, actitudes, forma el pensamiento crítico, el estudiante aprende a aprender, entre otros.	Frecuentemente utilizada en educación universitaria, en la cual el docente imparte clase y el estudiante escucha, toma nota y en ocasiones intervienen haciendo alguna pregunta.
Exige un trabajo previo del estudiante.	Su gran desventaja es que no favorece al desarrollo de habilidades de razonamiento y pensamiento crítico de los alumnos.

En palabras de (SERVICIO NACIONAL DE APRENDIZAJE (SENA), 2003): "en el ámbito educativo hay factores que limitan la posibilidad de implementación de estrategias interactivas y de largo plazo". De esta manera, se recomienda utilizar estrategias centradas en los docentes siempre y cuando:

- El docente sea un experto en el área que va a ofrecer.
- El tipo de conocimiento que se imparta sea más informativo que constructivo.
- Como el ambiente de trabajo académico es limitado, hay que recurrir a los elementos comprobados, a las tesis admitidas y a los procedimientos estandarizados.

Por otro lado se tienen las estrategias centradas en los **procesos y/o mediaciones didácticas,** que son muy útiles cuando el nivel de complejidad de los contenidos lo requieren. Este enfoque tiene las siguientes características:

- El proceso es determinante en el aprendizaje porque implica una secuencia de acciones conducentes a un propósito.
- El método de enseñanza actúa como un conjunto de estrategias que permiten guiar al alumno en la progresiva comprensión de elementos de conocimiento y en la aplicación de los mismos a circunstancias concretas para verificar el cumplimiento de leyes y principios, para verificar hipótesis, procedimientos o secuencias.
- Busca que el estudiante esté en condiciones de cuestionar o evaluar críticamente la información que recibe y las instrucciones o guías que se le dan, porque ha desarrollado los elementos necesarios para crear nueva información, nuevos procedimientos y métodos alternativos.
- Utiliza como técnicas de enseñanza la simulación, el seminario investigativo, el método de los cuatro pasos, el modelo didáctico operativo, la enseñanza mediante el conflicto cognitivo, la enseñanza mediante la investigación dirigida, el taller educativo.

Finalmente se exponen las principales características de las estrategias **centradas en el objeto de conocimiento:**

- La información que se transmite no puede dejarse inconexa, sino que se debe

estructurar de modo sistemático para lograr coherencia interna.

- El docente es responsable de planear cada una de las actividades y de verificar que el alumno las ejecute para alcanzar altos niveles de dominio de lo conceptual y de lo procedimental.
- Utiliza como técnicas la enseñanza basada en analogías o aprendizajes por transferencia analógica, la enseñanza por explicación y contratación de modelos, la enseñanza basada en la evidencia de desempeño (práctica o pasantía empresarial), la enseñanza para la comprensión.

Estrategias centradas en el alumno

El método de problemas

Se basa en proponer situaciones problemáticas a los estudiantes quienes, para solucionarlas, deberán realizar investigaciones, revisiones o estudio de temas que no estén debidamente asimilados; de esta manera se fortalece el análisis y la síntesis. Al poner al estudiante ante una situación conflictiva o dudosa y se le desafía a encontrar una solución satisfactoria para la misma, se entrena el razonamiento, la reflexión, en donde el estudiante aporta con ideas en lugar de cosas.

Objetivos

1. Desarrollar el raciocinio, sacándolo de la posición de receptividad de datos y de soluciones y obligándolo a buscarlos.
2. Desarrollar aptitudes de planeamiento, dado que el camino para llegar a las soluciones y obligándolo a buscarlos.
3. Desarrollar la iniciativa, dado que el educando se coloca ante una situación problemática ante la cual tiene que hallar una salida.
4. Desarrollar el control emocional, pues tendrá que esforzarse para trabajar con tranquilidad y eficiencia en la resolución del problema que se le proponga.
5. Desarrollar el espíritu de iniciativa, dado que el estudiante tiene que tomar todas las previsiones para la solución.
6. Hacer que el estudiante trabaje con base en hipótesis, cuya verificación exige el ejercicio de la reflexión, capacitándolo mejor para tomar decisiones, juzgar hechos y apreciar valores.
7. Provocar motivación, debido a la satisfacción que produce la solución, el descubrimiento del conocimiento.
8. Lograr un mejor desarrollo del aprendizaje.
9. Facilitar la transferencia del aprendizaje, es decir favorecer la aplicación de lo aprendido en situaciones diferentes.

Fases

I. **Planteamiento del problema.** El docente utiliza los recursos más apropiados según la naturaleza del tema y explica el problema a la clase.

II. **Hipótesis.** La clase elabora una o más hipótesis, que tratan de explicar la situación problemática planteada. Las hipótesis son caminos que guiarán la labor de los alumnos.

III. **Definición.** El trabajo consiste en definir, con la mayor precisión posible, los términos de la hipótesis, a fin de que se sepa exactamente de qué se trata, para encontrar con mayor certeza aquello que se busca.

IV. **Exploración lógica.** A partir de las hipótesis se identifican las conclusiones lógicas que las reforzarán o las debilitarán. Es por lo tanto una fase en la que se procura prever todas las consecuencias posibles de las hipótesis planteadas.

V. **Presentación de pruebas.** Esta fase está incluida en la anterior, dado que a medida que se hacen los razonamientos, se buscan en los hechos pruebas o comprobaciones que corroboren o no las hipótesis.

VI. **Generalización.** Se presenta la solución del problema propuesto o la comprobación de la hipótesis formulada basada en las pruebas disponibles. Como actitud educativa, es interesante advertir al alumno que la solución hallada no es tal vez la verdad absoluta sino tan sólo algo que va aproximándose a ella.

Funciones del docente

- Planificar y preparar el ambiente adecuado.
- Estimular a los alumnos para que organicen sus propias investigaciones.
- Estimular las discusiones, principalmente cuando éstas van perdiendo interés.

Es interesante agregar, a estas tres funciones, una más, que es la de seleccionar problemas desafiantes, actuales y adecuados para los estudiantes.

Fases de ejecución

Fase 1. El docente presenta las partes esenciales de una unidad u orienta el estudio de las mismas por medio de cualquier otro procedimiento didáctico, de la lectura, el estudio dirigido, etc.

Fase 2. Después de esta fase inicial y cuando juzgue que la clase está suficientemente informada con respecto a la unidad, el docente plantea una o más situaciones problemáticas relacionadas con los estudios realizados.

Fase 3. La solución de las cuestiones propuestas, que pueden buscarse individualmente o en grupo, requerirá la revisión y la profundización de los estudios realizados.

Fase 4. Las soluciones halladas individualmente o en grupo se presentan y se discuten en clase, a fin de enseñar la mejor (o las mejores).

Fase 5. El docente aprecia los trabajos de los estudiantes aceptando soluciones o recomendando nuevos estudios.

Fase 6. Prueba de verificación del aprendizaje con respecto a la unidad estudiada.

Fase 7. Rectificación del aprendizaje y asistencia especial a los educandos más atrasados.

Ficha de resolución de problemas

Grupo: Fecha:

Programa o especialidad:

1. Problema.
2. Datos del problema.

3. Mejor o mejores soluciones encontradas.
4. Verificación o sugerencias para la verificación de la mejor (o de las mejores) soluciones.

Método de situaciones (o de casos)

Describen una situación o problema similar a la realidad (ya sea tomado de una organización real o ficticia) que contiene acciones para ser valoradas y llevar a vía de hecho un proceso de toma de decisiones.

El docente no se convierte en transmisor de conocimientos; por el contrario, en el proceso de enseñanza él conduce la actividad de los participantes, su interrelación y la búsqueda de soluciones acertadas. Y lo más importante: enfatiza en el proceso de toma de decisiones, mediante lo cual se logra el aprendizaje.

Objetivos

1. Desarrollar las habilidades necesarias para el trabajo en colectivo.
2. Intercambiar las capacidades para tomar decisiones en forma colectiva.
3. Intercambiar criterios, ideas y experiencias en la solución del problema planteado.
4. Comprender cómo se aplican en la práctica los elementos teóricos que poseen.

Metodología de elaboración de ejercicios para la aplicación de los métodos de situaciones

a. Definir los objetivos que se persiguen, identificar el tipo de estudiante, tipo de curso, etc.
b. Selección del tipo de problema por describir, de qué rama, empresa o entidad económica se va a plantear la situación o hecho.
c. Observar las condiciones, sucesos que ocurren y recopilar todos los datos necesarios. Para esto pueden hacerse entrevistas, visitas a las entidades, lecturas de documentos, informes, análisis realizados, utilizar experiencias anteriores, etc.
d. Descripción de la situación. Para esto debe seleccionarse adecuadamente la información, poner en el material sólo la necesaria y prescindir de los juicios subjetivos del problema y exponerlos de forma lógica, precisa, sin palabras rebuscadas.
e. Aplicar el material de modo experimental para ver si se obtienen los resultados deseados, el debate previsto, si cumple los objetivos, etc.
f. Reelaboración del material de acuerdo con los resultados obtenidos y acompañarlos para uso de los docentes de orientaciones metodológicas para su utilización la cual debe obtener:
    a. Objetivos
    b. Material docente
    c. Información complementaria
    d. Solución real si fuese basado en un hecho real
    e. Tipo de estudiante con que se puede aplicar
    f. Tiempo aproximado para la aplicación y distribución del mismo

Es importante al concluir precisar que la aplicación de un solo ejercicio no desarrolla destrezas; por lo tanto, deben utilizarse en forma sistemática y además combi-

narse con otros métodos que conduzcan al verdadero cambio que se espera que el estudiante efectúe en sus conocimientos, actitudes, capacidades y destrezas.

Método de indagación

Con palabras del (SERVICIO NACIONAL DE APRENDIZAJE (SENA), 2003), la indagación puede ser entendida como la habilidad para hacer preguntas. Esta habilidad tiene origen en las necesidades del niño, y se convierte en un medio o instrumento para comprender. Proporciona una posible respuesta acerca del papel de la interrogante, la curiosidad, en cuánto a la actitud exploratoria. Es la que da origen al pensamiento. Inicialmente en el niño la curiosidad es como un instinto natural. Con el crecimiento y su participación en las relaciones sociales, el niño se vale del lenguaje interrogativo para continuar explorando, por medio de los adultos, el mundo. La pregunta sustituye a las manos. En este sentido la pregunta viene a ser algo así como las manos con las que el pensamiento explora el mundo.

Se deben considerar los siguientes pasos metodológicos:

1. Asegurar las estructuras mentales previas a la temática que se va a trabajar.
2. Orientación hacia los objetivos de aprendizaje.
3. Realización del proceso de indagación sobre el material objeto de estudio.
4. Elaboración por parte del estudiante, de preguntas para responder el material.
5. Discusión y respuesta a las preguntas planteadas.
6. Evaluación de las preguntas a partir de los criterios de una apropiada indagación.
7. Transformación creativa de las preguntas para que puedan servir como complemento al texto.
8. Cierre y evaluación final. Este último punto es de carácter metacognitivo, es decir se analizan los procesos mentales que desarrolla el estudiante durante el proceso.

Funciones del profesor indagador

- Arbitrar el proceso de discusión.
- Facilitar y estimular el razonamiento de los alumnos sobre sus propios temas, surgidos de ellos a través de la discusión en el aula.
- Mostrar interés en diferentes puntos de vista, aunque algunos de ellos no estén correctamente enfocados.
- Respetar y hacer respetar cada punto de vista. Tomarlos seriamente y con imparcialidad.
- Enfatizar la calidad del proceso de discusión más que el logro rápido de una conclusión específica.
- Crear una atmósfera de dar y recibir.
- Propiciar que cada estudiante desarrolle razones para sustentar sus opiniones.
- Mantener la discusión en una dirección constructiva y productiva.
- Propiciar la mayor participación posible de la clase en busca de mejorar la calidad de las mismas, y sin forzar a los estudiantes que mantienen un silencio productivo.
- Estimular las líneas de discusión divergentes (abanicos de ideas) y convergentes (conclusiones y cierres parciales).

- El facilitador debe ser capaz de agrupar ideas diferentes y sugerencias de los estudiantes para realizar, cuando sea necesario, un sumario, sobre todo si distintas porciones del grupo no se dan cuenta de las diferencias de sus opiniones.
- Sugerir posibles líneas de amplitud de discusión (divergencia) pero destacar los argumentos compatibles y sin contradicción.
- Mostrar las conexiones entre los argumentos de los estudiantes, que ellos sin embargo no han notado, así como las posiciones que van en la misma dirección (convergencia).

Es muy importante considerar que a diferencia del facilitador de grupo que tiende a estimular simplemente la producción de opiniones, el profesor indagador debe promover el razonamiento y el argumento que respalda la posición sumida, así como la creatividad para generar diversos ángulos de enfoque.

La tutoría

La tutoría es una actividad pedagógica que tiene como propósito orientar y apoyar a los alumnos durante su proceso de formación. Esta actividad no sustituye las tareas del docente, a través de las cuales se presentan a los alumnos contenidos diversos para que los asimilen, dominen o recreen mediante síntesis innovadoras. La tutoría es una acción complementaria, cuya importancia radica en orientar a los alumnos a partir del conocimiento de sus problemas y necesidades académicas, así como de sus inquietudes, y aspiraciones profesionales.

Objetivo

La tutoría tiene dos propósitos generales, favorecer el desempeño académico de los alumnos a través de acciones personalizadas o grupales, y contribuir a su formación integral.

Funciones del tutor

Las funciones del tutor suelen definirse y agruparse de acuerdo con el contenido de la tutoría, es decir, de acuerdo con el tipo de orientaciones y apoyos que se brindarán a los alumnos. En este sentido pueden reconocerse tres grupos de funciones básicas:

a. Desarrollo personal del estudiante
    - Descubre sus intereses.
    - Identifica sus dificultades.
    - Asume las consecuencias de sus actos.
    - Define su plan de vida.
    - Fortalece su autoestima.
    - Desarrolla habilidades para relacionarse con otros.
b. Desarrollo académico (profesional) del estudiante
    - Establece metas académicas claras y factibles.
    - Identifica sus dificultades de aprendizaje.
    - Realiza actividades pertinentes para resolver sus problemas educativos.
    - Selecciona adecuadamente sus actividades académicas formales y complementarias de acuerdo con sus intereses.

- Evalúa objetivamente su rendimiento escolar.
- Fortalece sus habilidades de estudio y de trabajo académico.

c. Orientación profesional del estudiante

- Visualiza con certidumbre su formación y sus posibilidades profesionales.
- Obtiene información precisa del campo laboral donde posiblemente se desempeñará.
- Identifica los retos actuales de su profesión.
- Transita sin conflicto del centro educativo al centro de trabajo.

La enseñanza por descubrimiento

La mejor manera de aprender algo es descubrirlo o crearlo por uno mismo, en lugar de que otra persona haga de intermediario entre uno y el conocimiento. Como dijo Piaget en una frase que se ha hecho célebre: "cada vez que se le enseña prematuramente a un niño algo que hubiera podido descubrir solo, se le impide a ese niño inventarlo y en consecuencia entenderlo completamente". Desde ese punto de vista, la enseñanza de la ciencia debe estar dirigida a facilitar ese descubrimiento. Pero ese descubrimiento no tiene por qué ser necesariamente autónomo, sino que puede y debe ser guiado por el profesor a través de la planificación de las experiencias y actividades didácticas.

Fases

1. Presentación de una situación problemática.
2. Observación, identificación de variables y recogida de datos.
3. Experimentación, para comprobar las hipótesis formuladas sobre las variables y los datos.
4. Organización e interpretación de los resultados.
5. Reflexión sobre el proceso seguido y los resultados obtenidos.

Dificultades

El dilema planteado ya hace algunos años cuando se trataba de aplicar la teoría de Piaget a la educación desde el enfoque del descubrimiento es: o se lo enseñamos muy pronto y no pueden entenderlo o se lo enseñamos demasiado tarde y ya lo saben.

Ideas básicas	Limitaciones
Todo el conocimiento real es descubierto por uno mismo.	La mayor parte de lo que uno sabe consiste en ideas que han sido descubiertas por otros y posteriormente comunicadas significativamente.
El significado es un producto exclusivo del descubrimiento creativo, no verbal.	El descubrimiento no es la única alternativa a la memorización.
El método de descubrimiento constituye el principal método para la transmisión del contenido de las materias de estudio.	El método de descubrimiento es muy lento y, sobre todo, se apoya en un inductivismo ingenuo.
La capacidad de resolver problemas constituye la meta primaria de la educación.	La capacidad de resolver problemas científicos nuevos de un modo autónomo no está al alcance de la mayor parte de los estudiantes.

El adiestramiento de la heurística del descubrimiento es más importante que el entretenimiento en la materia de estudio.	No se pueden resolver problemas científicos a menos que se disponga de un amplio bagaje de conocimientos con respecto al área temática de la que se trate.
El descubrimiento es un generador singular de motivación y confianza en uno mismo.	La motivación y la confianza en uno mismo se alcanzará sólo si el descubrimiento concluye en éxito, cosa que no debe esperarse de un modo generalizado.
El descubrimiento asegura la conservación de la memoria.	No hay pruebas de que el método por descubrimiento produzca un aprendizaje más eficaz y duradero que la enseñanza receptiva significativa.

El método de proyectos

El desarrollo de proyectos, así como el desarrollo de solución de problemas, se derivaron de la filosofía pragmática que establece que los conceptos son entendidos a través de las consecuencias observables y que el aprendizaje implica el contacto directo con las cosas.

Pasos para planear un proyecto

a. **Antes de la planeación de un proyecto.** Planear un proyecto toma tiempo y organización. Implementar el proyecto puede ser difícil las primeras veces. Por esta razón se sugiere empezar con proyectos cortos y conforme se vaya ganando experiencia se podrán hacer proyectos amplios.
b. **Metas.** El primer paso en la planeación de un proyecto es definir las metas u objetivos que se espera que los alumnos logren al finalizarlo, así como los aprendizajes que se desea que aprendan. Las metas pueden ser tan amplias como para ser cubiertas en un proyecto semestral o tan específicas que cubran un solo tema o unidad.
c. **Resultados esperados en los alumnos.** Conocimiento y desarrollo de habilidades: se refiere a aquellos que los alumnos sabrán y lo que serán capaces de hacer al finalizar el proyecto. Resultados del proceso de trabajo: se refiere a las competencias, estrategias, actitudes y disposición que los alumnos aprenderán durante su participación en el proyecto.
d. **Preguntas guía.** Una pregunta guía permite dar coherencia a la poca o ninguna estructura de los problemas o actividades a las que se enfrentan los alumnos que realizan un proyecto. Las preguntas guía conducen a los alumnos hacia el logro de los objetivos del proyecto. La cantidad de preguntas guía es proporcional a la complejidad del proyecto.
e. **Subpreguntas y actividades potenciales.** Una vez definidas las preguntas guía es necesario hacer una lista con todas las subpreguntas y actividades potenciales derivadas de ella. Las subpreguntas deben guiar a los estudiantes a responder las preguntas guía, se puede generar controversias y debates que les permita desarrollar la investigación y la capacidad de análisis.
f. **Productos.** Los productos son construcciones, presentaciones y exhibiciones realizadas durante el proyecto. Si bien no es posible identificar por adelantado todos los productos que resultarán del proyecto, es necesario tomar un tiempo para pensar qué podrían los estudiantes presentar, construir, diseñar, etc. Estos productos deben ser seleccionados con mucho cuidado. Los productos pueden ser presentados a lo largo del proyecto.

g. **Actividades de aprendizaje.** Las actividades de aprendizaje deben ser construidas en bloques, de manera que lleven a los estudiantes a alcanzar contenidos de conocimiento, de desarrollo de habilidades y de resultados de procesos.

   Estas actividades llevan a los alumnos a profundizar en los contenidos de conocimiento y a desarrollar habilidades de frente a las necesidades del proyecto, ya que requieren del alumno la transformación, análisis y evaluación de la información y las ideas para buscar la solución a una situación. A continuación se presenta algunos ejemplos de actividades de aprendizaje que pueden construirse dentro del proyecto: creación de prospectos, propuestas a considerar, desarrollo del plan de trabajo, cronograma, presupuestos, anteproyecto, diagrama de Gantt, observar, buscar información, realizar experimentos, contactar expertos, trabajar con asesores, discutir información recabada, buscar soporte técnico, construir, diseñar, fabricar, componer, presentar prototipo, pedir retroalimentación, hacer pruebas, evaluar, revisión/corrección de detalles, reconstruir a partir de la retroalimentación, adaptar, preparar, incorporar producción profesional, seguir estándares, presentar, exhibir, mostrar, etc.

h. **Apoyo instruccional.** Consiste en el apoyo instruccional con el fin de guiar el aprendizaje de los alumnos, así como facilitar un exitoso desarrollo del producto del proyecto. Aunque algunos tipos de apoyo se dan de manera imprevista, en general pueden ser planeados con anticipación.

i. **Ambiente de aprendizaje.** Los profesores pueden promover el éxito del proyecto creando óptimas condiciones de trabajo. Crear y mejorar los ambientes de aprendizaje es una estrategia que los profesores pueden utilizar para elevar el interés de los alumnos por el proyecto.

j. **Identificación de recursos.** Los recursos de información (libros, gente, Internet), así como las herramientas tecnológicas (computadoras, cámaras, impresoras) suministran lo necesario para que los alumnos logren desarrollar los productos del proyecto. Los recursos pueden ser elementos disponibles y son incorporados al proyecto como elementos que deben ser localizados, colectados, construidos o comprados.

Lo que se espera del estudiante

- Se sienta más motivado, ya que él es quien resuelve los problemas, planea y dirige su propio proyecto.
- Dirija por sí mismo las actividades de aprendizaje.
- Se convierta en un descubridor, integrador y presentador de ideas.
- Defina sus propias tareas y trabaje en ellas, independientemente del tiempo que requieren.
- Se muestre comunicativo, afectuoso, productivo y responsable.
- Use la tecnología para manejar sus presentaciones o ampliar sus capacidades.
- Trabaje en grupo.
- Trabaje colaborativamente con otros.
- Construya, contribuya y sintetice información.
- Encuentre conexiones interdisciplinarias entre ideas.
- Se enfrente a ambigüedades, complejidades y a lo impredecible.
- Se enfrente a obstáculos, busque recursos y resuelva problemas para enfrentarse a los retos que se le presentan.

- Adquiera nuevas habilidades y desarrolle las que ya tiene.
- Use recursos o herramientas de la vida real.
- Forme parte activa de su comunidad al desarrollar el trabajo del curso en un contexto social.
- Genere resultados intelectualmente complejos que demuestren su aprendizaje.
- Se muestre responsable de escoger cómo demostrará su competencia.
- Muestre un desarrollo en áreas importantes para la competencia en el mundo real: habilidades sociales, habilidades de vida, habilidades de administración personal y disposición al aprendizaje por sí mismo.
- Tenga clara la meta y se dé cuenta de que existe un reto en el que hay que trabajar.
- No se sienta temeroso de manejar cosas que no conoció a través del profesor y sepa que puede avanzar hasta donde piense que está bien.
- Se sienta útil y responsable de una parte del trabajo. Nadie se sienta relegado.
- No sea necesario usar tanto los textos, aunque continuamente se estén haciendo cosas y/o aprendiendo algo.
- Use habilidades que sabe le serán necesarias en su trabajo, como, por ejemplo, administrar el tiempo sabiamente, ejercitar la responsabilidad y no dejar caer al grupo.

Sobre el profesor

- El aprendizaje pasa de las manos del profesor a las del alumno, de tal manera que éste pueda hacerse cargo de su propio aprendizaje.
- El profesor está continuamente monitoreando la aplicación en el salón de clase, observando qué funcionó y qué no.
- El profesor deja de pensar que tiene que hacerlo todo y da a sus estudiantes la parte más importante.
- El profesor se vuelve estudiante al aprender cómo los estudiantes aprenden, lo que le permite determinar cuál es la mejor manera en que puede facilitarles el aprendizaje.
- El profesor se convierte en un proveedor de recursos y en un participante de las actividades de aprendizaje.
- El profesor es visto por los estudiantes más que como un experto, como un asesor o colega.

A medida que se incrementa el uso del método de proyectos la mayoría de los profesores consideran:

- Ser más entrenados y modelador. Hablar menos.
- Actuar menos como especialista.
- Usar más un pensamiento interdisciplinario.
- Trabajar más en equipo.
- Usar más variedad de fuentes primarias.
- Tener menos confianza en fuentes secundarias.
- Realizar más evaluación multidimensional.
- Realizar menos pruebas a lápiz y papel.

- Realizar más evaluación basada en el desempeño.
- Realizar menos evaluación basada en el conocimiento.
- Utilizar más variedad de materiales y medios.
- Estar menos aislados.

El reto más grande, tanto para los estudiantes como para los profesores, es desaprender los roles tradicionales del salón de clase (del estudiante como un receptor y el profesor como un proveedor de conocimiento). Saber cuándo meterse y cuándo dejar que los estudiantes trabajen las cosas por sí mismos lleva a tomar una nueva responsabilidad. Lo más relevante del método de proyectos es que cada participante sea visto como un estudiante y como un profesor. Este método requiere que el profesor esté muy atento e involucrado. Es responsabilidad del profesor asegurarse de que el programa y las habilidades apropiadas estén contenidas en el proyecto.

Dificultades

El método de proyectos presenta algunas dificultades y es oportuno trabajarlas para poder comprender mejor la forma de enfrentarlas. Una objeción es que los proyectos pueden gastar grandes cantidades de tiempo de instrucción, reduciendo las oportunidades para otros aprendizajes. Estos grandes bloques de tiempo algunas veces solo cubren una pequeña cantidad del contenido del programa. Más importante es el hecho de que el tiempo dedicado al proyecto no es tiempo dedicado a la instrucción directa en habilidades básicas.

Adicionalmente, dentro de una unidad del método de proyectos puede ser difícil obtener evidencia de que los estudiantes han alcanzado los objetivos establecidos (o han aprendido algo de valor relacionado con el programa). Finalmente, los proyectos son vulnerables a la crítica de que los estudiantes pasan la mayor parte de su tiempo llevando a cabo actividades que pueden no estar directamente relacionadas con el tema o no representar nuevos aprendizajes.

Estrategias centradas en el docente

Enseñanza tradicional

En este modelo, el profesor es un proveedor de conocimientos ya elaborados, listos para el consumo y el estudiante, en el mejor de los casos, el consumidor de esos conocimientos acabados, que se presentan casi como hechos, como algo dado y aceptado por todos aquellos que se han tomado la molestia de pensar sobre el tema.

Si la ciencia transmite un saber verdadero, avalado por las autoridades académicas, el profesor es su portavoz y su función es presentar a los estudiantes los productos del conocimiento científico de la forma más rigurosa y comprensible posible. El verbo que define la actividad profesional de muchos profesores es, aún hoy, "explicar" la ciencia a sus estudiantes y el que define lo que hacen sus estudiantes suele ser "copiar" y "pegar".

Las "clases magistrales" se basan en exposiciones del profesor ante una audiencia más o menos interesada que intenta tomar nota de lo que ese profesor dice y se acompañan con algunos ejercicios y demostraciones que sirven para ilustrar o apoyar las explicaciones. Así, aunque cada profesor desarrolla de forma más o menos intuitiva sus propias rutinas didácticas, este tipo de enseñanza implica idealmente

una secuencia de actividades como la que refleja la secuencia que se presenta a continuación.

Fase	Actividad educativa
Atención	Anunciar a la clase que es hora de comenzar.
Expectativas	Informar a la clase de los objetivos de la lección y de la clase, y del tipo y monto del rendimiento esperado.
Recuperación	Pedir a la clase que recuerde las reglas y los conceptos subordinados.
Percepción selectiva	Presentar ejemplos del nuevo concepto o regla.
Codificación semántica	Ofrecer claves para recordar la información.
Recuperación y respuesta	Pedir a los estudiantes que apliquen el concepto o la regla a nuevos ejemplos.
Refuerzo	Confirmar la exactitud de las respuestas de los estudiantes.
Clave para la recuperación	Practicar exámenes breves sobre el material nuevo.
Generalización	Ofrecer repasos especiales.

Dificultades

Esta concepción educativa responde a una larga tradición que se remonta a los propios orígenes de los sistemas educativos formales, que desde siempre han tenido como una de sus funciones básicas lograr que los estudiantes y futuros ciudadanos reproduzcan y, por tanto, perpetúen los conocimientos, valores y destrezas propias de una cultura. Sin embargo, este modelo tradicional resulta poco funcional en el contexto de las nuevas demandas y escenarios de aprendizaje que caracterizan a la sociedad de hoy.

El modelo tradicional basado en la transmisión de saberes conceptuales establecidos no asegura un uso dinámico y flexible de esos conocimientos fuera del aula, pero además plantea numerosos problemas y dificultades dentro de las aulas. Con mucha frecuencia se produce un divorcio muy acusado entre las metas y motivos del profesor y de los estudiantes, con lo que éstos se sienten desconectados y desinteresados, al tiempo que el profesor se siente cada vez más frustrado.

La enseñanza expositiva

Los problemas generados por la enseñanza tradicional no se deberían tanto a su enfoque expositivo como al inadecuado manejo que hacía de los procesos de aprendizaje de los estudiantes, por lo que, para fomentar la comprensión, o en su terminología un aprendizaje significativo, no hay que recurrir tanto al descubrimiento como a mejorar la eficacia de las exposiciones.

Para ello hay que considerar no sólo la lógica de las disciplinas sino también la lógica de los estudiantes. El aprendizaje de la ciencia consiste en "transformar el significado lógico en significado psicológico", es decir en lograr que los estudiantes asuman como propios los significados científicos. Para ello la estrategia didáctica deberá consistir en un acercamiento progresivo de las ideas de los alumnos a los conceptos científicos, que constituirán el núcleo de los currículos.

Fases

- Fase primera
- Presentación del organizador

- Aclarar los objetivos de la lección
- Presentar el organizador
- Aclarar las propiedades definitorias
- Dar ejemplos
- Aportar un contexto
- Repetir
- Incitar el conocimiento y experiencia del sujeto

- Fase segunda
- Presentación del material de trabajo
- Explicitar la organización
- Organizar lógicamente el aprendizaje
- Mantener la atención
- Presentar el material

- Fase tercera
- Potenciar la organización
- Utilizar los principios de reconciliación integradora
- Promover un aprendizaje de recepción activa
- Suscitar un enfoque crítico
- Explicar

Estrategias centradas en el proceso

La simulación

Consiste en que los participantes, organizados en equipos, asumen los roles en los que se colocan en un sistema de condiciones, limitaciones y relaciones de una organización económica dada, es decir, un modelo que reproduce condiciones similares a las existentes en la práctica.

Con la simulación se puede anticipar las consecuencias de las decisiones por tomar en condiciones reales, y por tanto aprender de la conducta propia y de los demás ya que este método, al igual que otros, parte del hecho de que el aprendizaje más efectivo se produce en las personas por sí mismas, a partir de sí mismas y de los demás.

Elementos

a. **El modelo.** Expresa la situación en que se va a desarrollar el proceso, proporciona el escenario para los hechos, proveyendo a los jugadores de datos históricos sobre la entidad y de datos económicos que permitan un punto de partida. Contempla también los roles por desempeñar; así como las normas generales del funcionamiento de la entidad, los sistemas informativos y el ambiente del medio en que se encuentra.

b. **Los grupos de jugadores.** La cantidad de grupos dependerá del ejercicio en particular y en su formación no es necesaria la homogeneidad en su composición atendiendo a la especialidad o el cargo; todo lo contrario, en la mayoría de los casos resulta útil la formación de equipos heterogéneos y a la vez equilibrados entre ellos, a fin de que todos los equipos que se formen estén en igualdad de condiciones para trabajar y que los resultados por evaluar respondan a la gestión iniciativa de uno u otro equipo y no a ventajas en su composición.

c. **Los expertos y organizadores del juego.** Los profesores deben poseer una vasta experiencia de manera que pueda conducir el ejercicio con éxito, manejando las posibles situaciones de tensión y conduciendo al grupo por el camino trazado por ellos mismos para que aprendan de sus aciertos y errores.

   En este sentido, se debe destacar que el tutor debe actuar sin intervenir en las decisiones que deben tomar los participantes aunque sí puede preguntar, exigir y forzar a los mismos a valorar sus decisiones y ser más analíticos, sin tomar partido, posición o sugerir la actuación futura, sino obtener toda la información posible de la actuación de los participantes y los grupos para poder conducir una sesión de conclusiones exitosas, donde todos aprenden de su propio comportamiento y de los demás.

d. **El material.** El material sobre el juego debe caracterizar el modelo que se utiliza, es decir, el escenario, las características organizativas del juego, la descripción de los roles, las normas que regirán el funcionamiento de la entidad, así como las instrucciones precisas a los participantes de las reglas de juego, los materiales a su disposición y lo que se espera de ellos. Es decir, constituye el reglamento para el trabajo de los participantes, que incluye también el horario y los modelos o documentos necesarios para elaborar.

Funcionamiento

La conducción de una simulación atraviesa por tres etapas:

- **La introducción al juego.** En esta etapa se deben abordar los siguientes aspectos:
    - Los objetivos del ejercicio, es decir, su razón de ser. Esto debe abordarse de forma breve, sin abundar en detalles que descubran la esencia del mismo y pongan sobre aviso a los participantes, eliminando la espontaneidad necesaria en su comportamiento.
    - La situación general en que se desenvuelve el juego, es decir, el tipo de empresas que se van a formar, el tipo de producción o tarea por cumplir, los materiales por utilizar, especificaciones de calidad, situación del mercado, etc.
    - El procedimiento del juego, o sea las reglas de juego. Aquí debe aclararse a los participantes cómo se va a desarrollar el juego, los pasos por dar. Se debe informar del horario de trabajo y las principales restricciones y posibilidades.
    - La asignación de los participantes a los diferentes roles.
- **El desarrollo del juego.** Se produce dirigido por los propios participantes, pero bajo el control del tutor del ejercicio, quien mantendrá las reglas del juego y hará

cumplir el horario establecido para el mismo. Es muy importante que se mantengan durante el desarrollo del juego las condiciones normales para el desarrollo de las actividades, aunque en ocasiones resulta conveniente crear situaciones de tensión en el trabajo, lo que demanda de ellos solucionar estos problemas, sin descuidar la actividad de la empresa.

- **La evaluación del juego.** Es la etapa dedicada a evaluar los resultados de la gestión de cada equipo. Es crucial para el éxito desde el punto de vista de la capacitación, ya que es el momento en que se hace el balance de los conocimientos y habilidades demostradas por los participantes para obtener los resultados. Debe tratarse de que la sesión de conclusiones no se convierta en una sesión de análisis de información eminentemente cuantitativa, ya que el objetivo de esta etapa es el análisis del proceso y no de los resultados en sí. El énfasis debe hacerse en el comportamiento de los participantes. Es importante precisar que es en esta etapa donde se sistematizan y ordenan las experiencias de los participantes para lograr el reconocimiento de las necesidades de perfeccionarse o de reforzar su comportamiento.

El método de los cuatro pasos

Pasos

- EL INSTRUCTOR DICE Y HACE
- EL ESTUDIANTE DICE Y EL INSTRUCTOR HACE
- EL ESTUDIANTE DICE Y HACE
- EL ESTUDIANTE HACE Y EL INSTRUCTOR SUPERVISA

El instructor dice y hace

1. Hace demostraciones y explicaciones paulatinas.
2. Insiste en los puntos clave de lo que está enseñando.
3. Motiva a los alumnos.
4. Hace la introducción previa del tema por desarrollar.
5. Logra la participación del grupo.

El aprendiz dice y el instructor hace

1. El instructor solicita que uno o varios aprendices expliquen la operación.
2. El instructor va ejecutando las operaciones indicadas por el instructor.
3. El instructor y demás estudiantes permanecen atentos para corregir posibles errores.
4. Se repite la demostración si es necesario.

El estudiante dice y hace

1. El instructor solicita que uno de los aprendices ejecute las operaciones y explique cada punto clave a medida que los va realizando.
2. El instructor promueve el uso de preguntas por parte de los aprendices.
3. El instructor corrige errores.
4. El instructor verifica a cada paso si el aprendiz comprendió.

El estudiante hace y el instructor supervisa

1. El instructor solicita al aprendiz ejecutar las operaciones.
2. El instructor verifica aciertos y errores y corrige si es necesario.
3. El instructor informa a los aprendices a quien debe consultar en caso necesario.

4. De manera gradual el instructor procura dejar que los aprendices realicen solos el trabajo.
5. Se promueve en los estudiantes explicaciones sobre las demostraciones.

Estrategias centradas en el conocimiento

El aprendizaje basado en analogías o aprendizaje por transferencia analógica

Las analogías consisten en relaciones de semejanza, son comparaciones entre relaciones similares. Los estudiantes utilizan con frecuencia analogías y aunque lo hagan en forma ingenua el concepto de analogía resulta novedoso para el aprendiz.

La forma en que el estudiante utiliza por su cuenta la analogía para aprender un concepto o una información nueva, es extrayendo ideas o esquemas que ya posee en su intelecto, por eso el presentarle el nuevo conocimiento con base en analogías le facilita este proceso cognitivo. Este es un principio básico del aprendizaje basado en analogías. La eficacia de la analogía depende entonces del grado en que exista una estructura y sea coherente en la mente del alumno.

Por otra parte, la analogía en sí misma presenta una estructura integrada que puede ser asimilada de una sola vez. Se puede construir analogías que muestren relaciones como las siguientes:

- Relaciones de causa efecto (A es la causa de B; A es el resultado de B; A da origen a B).
- Relaciones medio-fin (está definido, está proyectado para, está hecho para).
- Relaciones parte-todo (es parte, participa en, está implicado en, pertenece a).
- Relaciones de clase-miembro (es un miembro de, pertenece a la clase, es uno de).

Uno de los errores básicos en la construcción de analogías es relacionar atributos o elementos en lugar de relaciones. A continuación, se presenta el procedimiento para la utilización analógica como estrategia de aprendizaje:

1. Aclare por sí mismo cuáles son las esencias del tema que trabajará con sus estudiantes. Puede ayudarse mediante acuerdos con otros colegas que trabajan la misma asignatura o módulo, o que sean de su misma especialidad.
2. Asegúrese del estado satisfactorio de la estructura de conocimientos previos, pertinentes al nuevo material que se ha de aprender.
3. Oriente los objetivos a la sesión.
4. Enseñe y ejercite en sus estudiantes y a usted mismo en la identificación de relaciones. Para ello es conveniente ejercitarlos primero en identificar relaciones generales, presentes en la experiencia vital de cualquiera, y que no dependan de un saber específico.
5. Muestre a sus estudiantes una analogía explicativa, directa, bien explícita y en la que se relacionen esencias sobre un tema.
6. Desmonte la analogía, desmenuzándola y explicándola.
7. Presente a los estudiantes un conjunto de analogías deliberadamente incorrectas sobre las esencias que ha de aprender. Pídales detectar las incorrectas y debatir por qué lo son, mediante argumentación bien sustentada.

8. Presénteles a los estudiantes analogías correctas mezclándolas con incorrectas. Pídales que las analicen, detecten, argumenten y debatan por qué son o no son correctas.
9. Presente analogías semi-estructuradas, correctas, para completar la relación faltante, mediante diálogo reflexivo.

Sistema de evaluación por competencias

La evaluación, en un proceso de aprendizaje, es de vital importancia para todos sus participantes, ya sea para el docente o para el estudiante, además de ser un factor motivador que brinda la oportunidad al aprendiz de conocer cuáles son sus resultados de aprendizaje en cuanto al "qué" se ha aprendido y al "cómo" habría podido hacerse.

De acuerdo con (Jesús Valverde Berrocoso, 2012), en un sistema de evaluación por competencias (SEC), los evaluadores hacen valoraciones según las evidencias obtenidas de diversas actividades de aprendizaje, que definen si un estudiante alcanza o no los requisitos recogidos por un conjunto de indicadores en un determinado grado.

Hay que recordar que una competencia está compuesta por habilidades cognitivas, interactivas, afectivas, actitudes y valores, las cuales son necesarias para la ejecución de tareas, resolver problemas e inclusive el correcto desempeño en la vida profesional.

Además, (Jesús Valverde Berrocoso, 2012) expone que la evaluación necesita ser considerada como un proceso de recogida de evidencias (a través de actividades de aprendizaje) y de formulación de valores sobre la medida y la naturaleza del progreso del estudiante, según unos resultados de aprendizaje esperados.

Según el (Tecnológico de Monterrey), la evaluación por competencias presenta, entre otras, las siguientes ventajas para los procesos de enseñanza y de aprendizaje:

Para la enseñanza	Para el aprendizaje
El profesor puede identificar las áreas de la instrucción que necesitan mejoras. El docente puede constatar las competencias logradas por sus alumnos a nivel personal y grupal. Aporta evidencias de habilidades, destrezas y logros alcanzados.	El aprendizaje mejora cuando el estudiante sabe claramente lo que se espera de él. Motiva al alumno al saber cómo se evaluará su desempeño. Ayuda al alumno a determinar su propio progreso y así identificar sus áreas fuertes y débiles. Permite conocer las competencias logradas.

Competencias del Tecnólogo Superior en Desarrollo de Software

Competencias básicas

- CB1. Poseer y comprender conocimientos que aporten una base u oportunidad de ser creativos en el desarrollo y/o aplicación de ideas, a menudo en un contexto de producción de software.
- CB2. Que los estudiantes sepan comunicar sus conclusiones y los conocimientos y razones últimas que las sustentan a públicos especializados y no especializados de un modo claro y sin ambigüedades.
- CB3. Que los estudiantes posean habilidades de aprendizaje que les permitan continuar estudiando de un modo que habrá de ser en gran medida autodirigido o autónomo.
- CB4. Capacidad para trabajar en equipo y para relacionarse con otras personas del mismo o distinto ámbito profesional.

Competencias del perfil profesional

- CPP1. Utilizar sistemas informáticos mediante procesos de enseñanza-aprendizaje adecuados para poder aplicarlos en otras áreas del saber, junto con la administración de proyectos de redes de comunicaciones a partir de las innovaciones tecnológicas con eficiencia y eficacia y el diseño de base de datos de información integrales y consistentes con fines de automatización de procesos empresariales e institucionales.
- CPP2. Desarrollar proyectos de software mediante el uso de metodologías de ingeniería de software actuales y diseñar y mantener infraestructuras tecnológicas de vanguardia locales, remotas, físicas y virtuales.
- CPP3. Administrar proyectos tecnológicos y estratégicos de media y alta complejidad.
- CPP4. Gestionar su trabajo en áreas relacionadas al desarrollo y/u operación de sistemas de información de automatización de procesos, áreas de soporte de infraestructuras tecnológicas, áreas de aprovisionamiento y soporte de redes de datos, áreas de gestión de proyectos y procesos tecnológicos.
- CPP5. Utilizar y/u operar sistemas informáticos de cualquier tipo basados en las habilidades del conocimiento cierto de los fundamentos de diseño de estructuras y plataformas informáticas estandarizadas a nivel internacional, es decir podrá operar un sistema informático basado en el conocimiento de las características genéricas de todos los sistemas para aprender y operar de manera más rápida el funcionamiento de nuevas plataformas cuando sea necesario.

Competencias de la asignatura

- CA1. Conocer el concepto de lógica de programación para focalizar esfuerzos en el análisis y diseño de algoritmos antes que su programación en un lenguaje de alto nivel.
- CA2. Resolver algoritmos haciendo uso de una metodología que permita alcanzar el objetivo.
- CA3. Utilizar técnicas y herramientas TIC para la representación y simulación de algoritmos.

- CA4. Crear algoritmos aplicando la estructura secuencial, decisión, ciclo y mixtas utilizando DFD y PseInt.
- CA5. Utilizar arreglos y matrices en la construcción de algoritmos utilizando PseInt.
- CA6. Implementar funciones y subprocesos en la lógica algorítmica.

Competencias de la Unidad 1

- CU1.1. Conocer la importancia de desarrollar la lógica de programación.
- CU1.2. Entender una metodología para resolver problemas.
- CU1.3. Resolver algoritmos informales.
- CU1.4. Conocer la base teórica de algoritmos.
- CU1.5. Conocer las consideraciones algorítmicas sobre el pensamiento humano.
- CU1.6. Utilizar variables, constantes y operadores.
- CU1.7. Crear algoritmos computacionales de forma gráfica utilizando diagramas de flujograma y la estructura secuencial en la herramienta DFD.
- CU1.8. Crear algoritmos computacionales utilizando pseudocódigo y la estructura secuencial en la herramienta PseInt.
- CU1.9. Crear un videojuego en Scratch aplicando variables y operadores.

Competencias de la Unidad 2

- CU2.1. Comprender que es, cuando y como utilizar las estructuras de decisión.
- CU2.2 Crear algoritmos utilizando la estructura de decisión en DFD y PseInt.
- CU2.3 Comprender que es, cuando y como utilizar las estructuras de ciclos.
- CU2.4 Crear algoritmos utilizando la estructura de ciclos en DFD y PseInt.
- CU2.5 Comprender que es, cuando y como utilizar las estructuras mixtas.
- CU2.6 Crear algoritmos utilizan estructuras mixtas en DFD y PseInt.

Competencias de la Unidad 3

- CU3.1 Comprender como funcionan los arreglos y las matrices.
- CU3.2 Utilizar arreglos y matrices en la construcción de algoritmos.
- CU3.3 Comprender la utilidad de las funciones y subprocesos en PseInt.
- CU3.4 Utilizar funciones y subprocesos.
- CU3.5 Conocer los principales consejos y reflexiones a la hora de crear algoritmos.

Indicadores de evaluación

Con base en (Jiménez, Martínez G, Sánchez G, Juárez J, & Paredes S, 2009), los indicadores o elementos de competencia son parámetros concretos de desempeño para evaluar la competencia. Cada indicador tiene asociado un nivel de logro que refleja el nivel de calidad en la adquisición de la competencia para el indicador.

De esta manera, se establecen los siguientes niveles de logro:

Indicador	Valor
No lo hace.	0
Las evidencias indican poca comprensión del problema. No incluye los elementos requeridos en la actividad.	1
Se evidencia comprensión del problema. Incluye algunos elementos requeridos en la actividad.	2
Se evidencia comprensión del problema. Incluye un alto porcentaje de los elementos requeridos en la actividad.	3
Se evidencia comprensión total del problema. Incluye todos los elementos requeridos en la actividad.	4

Hitos de evaluación

Empleando las palabras de (Jiménez, Martínez G, Sánchez G, Juárez J, & Paredes S, 2009), los hitos de evaluación determinan los momentos cronológicos en los cuales se realizan las acciones de evaluación. Así, se proponen los siguientes hitos de evaluación:

Hito	Descripción
1	Al comienzo de la asignatura. Es el hito de evaluación de diagnóstico del estudiante.
2	Al final de la Unidad 1. Corresponde a la evaluación continua del estudiante (EVA).
3	Al final de la Unidad 2. Corresponde a la evaluación continua del estudiante (EVA).
4	Al final de la Unidad 3. Corresponde a la evaluación continua del estudiante (EVA).
5	Al final de la asignatura. Corresponde a un proyecto integrador publicado en el EVA.
6	Al final de la asignatura. Corresponde la evaluación individual (autoevaluada) de las competencias básicas de la carrera y presentar el resultado a nivel de grupo.

Acciones de evaluación

Desde el punto de vista de (Jiménez, Martínez G, Sánchez G, Juárez J, & Paredes S, 2009), las acciones de evaluación son actividades que se realizan para verificar el nivel de logro de los indicadores. Se proponen las siguientes acciones de evaluación:

- AE1. Entrevista personal o grupal que permita obtener un diagnóstico del estudiante o del grupo al iniciar la asignatura.
- AE2. Laboratorio integrador de la unidad, en donde se apliquen los diferentes niveles de aprendizaje propuestos por Bloom.
- AE3. Examen o proyecto final que englobe lo estudiado en la asignatura.
- AE4. Evaluación individual (autoevaluada) de las competencias básicas de la carrera y presentar el resultado a nivel de grupo.

Evaluación	Hitos	Acciones de evaluación		
		AE1	AE2	AE3
Diagnóstico	1	X		
Continua	2		X	
	3		X	
	4		X	
Finalización	5			X
	6			X

Agentes de evaluación

Los agentes de evaluación, son personas o entidades encargadas de realizar la evaluación de las acciones de evaluación propuestas en la sección anterior, de esta manera, se tiene los siguientes agentes:

- AG1. Profesores de la asignatura. Evaluarán total o parcialmente todas las actividades de evaluación.
- AG2. Estudiante sujeto a evaluación. Contribuirá en la evaluación de las actividades AE2 y AE3.
- AG3. Estudiante par que el estudiante sujeto a evaluación. Contribuirá en la evaluación de las actividades AE2 y AE3.

Agentes	Acciones de evaluación		
	AE1	AE2	AE3
AG1	X	X	X
AG2		X	X
AG3		X	X

Evaluación de competencias de los laboratorios

A juicio de (Universitat de Barcelona, 2013) se plantea el uso de rúbricas para la evaluación de los aprendizajes en educación superior, en donde se recalca la importancia de definir competencias, establecer indicadores y los criterios de evaluación. De esta manera, la rúbrica es un instrumento cuya principal finalidad es compartir los criterios de realización de las tareas de aprendizaje y de evaluación con los estudiantes y entre profesorado. Se debe ver a la rúbrica como guía u hoja de ruta de las tareas, muestra las expectativas que alumnado y profesorado tienen y comparten sobre una actividad o varias actividades, organizadas en diferentes niveles de cumplimiento: desde el menos aceptable hasta la resolución ejemplar, desde lo considerado como insuficiente hasta lo excelente.

Laboratorio 1

A la sumatoria total que obtenga el estudiante se aplicará una regla de tres simple para obtener la nota final en escala de diez **(Total obtenido * 10) / 40**. En donde 10 es la escala por transformar y 40 es el total de puntos que se puede obtener en el laboratorio.

LABORATORIO 1					
Competencia	Actividad	Saber (datos, conceptos, conocimientos)	Saber hacer (habilidades, destrezas, métodos de actuación)	Saber ser (actitudes y valores que guían el comportamiento)	Saber estar (comunicación interpersonal y el trabajo cooperativo)
CU1	A1	X		X	X
CU2	A2	X		X	X
CU3	A3	X	X	X	X

CU4	A4	X		X	X
CU5		X		X	X
CU6	A5		X	X	X
CU7	A5		X	X	X
CU8	A5		X	X	X
CU9	A6, A7, A8	X	X	X	X

Competencia	Actividad	Escala de Calificación				
		0	1	2	3	4
CU1	Actividad 1	No lo hace	Explica los términos y realiza la comparación, sin embargo, no realiza el análisis.	Explica los términos y realiza la comparación y el análisis. Sin embargo, se evidencia poca comprensión del tema.	Explica los términos y realiza la comparación y el análisis con buen nivel de comprensión del tema.	Explica los términos y realiza la comparación y el análisis con un total nivel de comprensión del tema.
CU2	Actividad 2	No lo hace	Se aborda una metodología, pero no la explica con sus palabras.	La explicación evidencia poca comprensión de la metodología.	La explicación evidencia buena comprensión de la metodología.	La explicación evidencia total comprensión de la metodología.
CU3	Actividad 3	No lo hace	Analiza las ventajas y desventajas del uso de una metodología a la hora de crear algoritmos informales.	Analiza las ventajas y desventajas del uso de una metodología a la hora de crear algoritmos informales e identifica si se aplica en la vida diaria.	Analiza las ventajas y desventajas del uso de una metodología a la hora de crear algoritmos informales e identifica si se aplica en la vida diaria y ¿Por qué?	Analiza las ventajas y desventajas del uso de una metodología a la hora de crear algoritmos informales e identifica si se aplica en la vida diaria y ¿Por qué? Además, propone y resuelve un algoritmo informal.
CU4 CU5	Actividad 4	No lo hace	Sintetiza la teoría algorítmica, sin embargo, no identifica como se utilizan los algoritmos en la vida cotidiana.	Sintetiza la teoría algorítmica e identifica como se utilizan en la vida cotidiana con poco nivel de comprensión.	Sintetiza la teoría algorítmica e identifica como se utilizan en la vida cotidiana con buen nivel de comprensión.	Sintetiza la teoría algorítmica e identifica como se utilizan en la vida cotidiana con total nivel de comprensión.
CU6	Actividad 5 (Ejercicio 1)	No lo hace	Utiliza correctamente variables, constantes y operadores en uno de cuatro ejercicios propuestos.	Utiliza correctamente variables, constantes y operadores en dos de cuatro ejercicios propuestos.	Utiliza correctamente variables, constantes y operadores en tres de cuatro ejercicios propuestos.	Utiliza correctamente variables, constantes y operadores en todos los ejercicios propuestos.
CU7	Actividad 5 (Ejercicio 2)	No lo hace	Crear algoritmos computacionales evidenciando un correcto diseño y ejecución de al menos uno de cuatro literales propuestos en DFD.	Crear algoritmos computacionales evidenciando un correcto diseño y ejecución de al menos dos de cuatro literales propuestos en DFD.	Crear algoritmos computacionales evidenciando un correcto diseño y ejecución de al menos tres de cuatro literales propuestos en DFD.	Crear algoritmos computacionales evidenciando un correcto diseño y ejecución de todos los literales propuestos en DFD.
CU8	Actividad 5 (Ejercicio 3)	No lo hace	Crear algoritmos computacionales evidenciando un correcto manejo de la estructura secuencial de al menos uno de cuatro literales. propuestos en PseInt.	Crear algoritmos computacionales evidenciando un correcto manejo de la estructura secuencial de al menos dos de cuatro literales. propuestos en PseInt.	Crear algoritmos computacionales evidenciando un correcto manejo de la estructura secuencial de al menos tres de cuatro literales. propuestos en PseInt.	Crear algoritmos computacionales evidenciando un correcto manejo de la estructura secuencial de todos los literales propuestos en PseInt.

CU9	Actividad 6	No lo hace	Compara dos de las tres herramientas propuestas.	Compara las tres herramientas solicitadas, sin embargo, se evidencia poca comprensión del tema.	Compara las tres herramientas solicitadas con una buena comprensión del tema.	Compara las tres herramientas solicitadas con una total comprensión del tema.
CU9	Actividad 7	No lo hace	Instala, pero no describe con sus palabras el proceso.	Instala y describe con sus palabras el proceso, sin embargo, se observa falta de especificaciones técnicas en el procedimiento.	Instala y describe con sus palabras el proceso con un uso adecuado de terminología técnica en el proceso.	Instala y describe con sus palabras el proceso con un total manejo de lenguaje técnico en el procedimiento.
CU9	Actividad 8	No lo hace	Crea un videojuego utilizando variables y operadores, sin embargo, no explica como lo hizo.	Crea un videojuego utilizando variables y operadores, sin embargo, en la explicación se evidencia poco dominio del tema.	Crea un video juego utilizando variables y operadores con una explicación clara y poco técnica de cómo lo realizó.	Crea un videojuego utilizando variables y operadores con una explicación clara y técnica sobre el procedimiento.

Laboratorio 2

A la sumatoria total que obtenga el estudiante se aplicará una regla de tres simple para obtener la nota final en escala de diez **(Total obtenido * 10) / 40**. En donde 10 es la escala por transformar y 40 es el total de puntos que se puede obtener en el laboratorio.

LABORATORIO 1					
Competencia	**Actividad**	**Saber (datos, conceptos, conocimientos)**	**Saber hacer (habilidades, destrezas, métodos de actuación)**	**Saber ser (actitudes y valores que guían el comportamiento)**	**Saber estar (comunicación interpersonal y el trabajo cooperativo)**
CU2.1	A1	X		X	X
CU2.1	A2	X		X	X
CU2.3	A3	X		X	X
CU2.1	A4	X		X	X
CU2.3		X		X	X
CU2.2	A5		X	X	X
CU2.4			X	X	X
CU2.6			X	X	X
CU2.1 CU2.3 CU2.5	A6	X	X	X	X

Compe-tencia	Actividad	Escala de Calificación				
		0	1	2	3	4
CU2.1	Actividad 1	No lo hace	Establece las diferencias entre dos de cuatro variaciones.	Establece las diferencias entre las cuatro variaciones, sin embargo, se evidencia poca comprensión del tema.	Establece las diferencias entre las cuatro variaciones con buena de comprensión del tema.	Establece las diferencias entre las cuatro variaciones con total comprensión del tema.
CU2.1	Actividad 2	No lo hace	Proporciona un texto sin comparar las estructuras.	Realiza la comparación, sin embargo, se evidencia poca comprensión del tema.	Realiza la comparación con buena comprensión del tema.	Realiza la comparación con total comprensión del tema.
CU2.3	Actividad 3	No lo hace	Realiza las características de dos estructuras.	Realiza las características de las tres estructuras, sin embargo, se evidencia poca comprensión del tema, además, el texto carece de lenguaje técnico y es poco comprensivo.	Realiza las características de las tres estructuras con buena comprensión del tema y el texto carece de lenguaje técnico, pero es comprensivo.	Realiza las características de las tres estructuras con total comprensión del tema utilizando un lenguaje técnico y comprensivo.
CU2.1 CU2.3	Actividad 4	No lo hace	Identifica situaciones de la vida diaria en dos de cinco estructuras.	Identifica situaciones de la vida diaria en tres de cinco estructuras.	Identifica situaciones de la vida diaria en cuatro de cinco estructuras.	Identifica situaciones de la vida diaria en todas las estructuras solicitadas.
CU2.2 CU2.4 CU2.6	Actividad 5 (Ejercicio 1)	No lo hace	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no es acorde a lo solicitado.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no alcanza completamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, alcanza medianamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt alcanzando el objetivo.
CU2.2 CU2.4 CU2.6	Actividad 5 (Ejercicio 2)	No lo hace	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no es acorde a lo solicitado.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no alcanza completamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, alcanza medianamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt alcanzando el objetivo.
CU2.2 CU2.4 CU2.6	Actividad 5 (Ejercicio 3)	No lo hace	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no es acorde a lo solicitado.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no alcanza completamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, alcanza medianamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt alcanzando el objetivo.
CU2.2 CU2.4 CU2.6	Actividad 5 (Ejercicio 4)	No lo hace	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no es acorde a lo solicitado.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no alcanza completamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, alcanza medianamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt alcanzando el objetivo.
CU2.2 CU2.4 CU2.6	Actividad 5 (Ejercicio 5)	No lo hace	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no es acorde a lo solicitado.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no alcanza completamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, alcanza medianamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt alcanzando el objetivo.
CU2.1 CU2.3 CU2.5	Actividad 6	No lo hace	Crea un videojuego utilizando lo estudiado en la Unidad 1 y Unidad 2, sin embargo, no explica cómo lo hizo.	Crea un videojuego utilizando lo estudiado en la Unidad 1 y Unidad 2, sin embargo, en la explicación se evidencia poco dominio del tema.	Crea un video juego utilizando lo estudiado en la Unidad 1 y 2 con una explicación clara y poco técnica de cómo lo realizó.	Crea un videojuego utilizando lo abordado en la Unidad 1 y 2 con una explicación clara y técnica sobre el procedimiento.

Evaluación de competencias básicas de la carrera

Desde el punto de vista de (Universitat Politécnica de Valéncia, 2015) propone indicadores de evaluación para la evaluación de la Competencia Innovación, Creatividad y Emprendimiento en máster. Se ha modificado y adaptado la siguiente rúbrica para evaluar las competencias básicas de la carrera de Tecnología Superior en Desarrollo de Software identificando cuatro ejes de orientación (Emprendimiento, Comunicación Oral y Escrita, Aprendizaje continuo y Trabajo colaborativo).

Competencia	Orientación
CB1. Poseer y comprender conocimientos que aporten una base u oportunidad de ser creativos en el desarrollo y/o aplicación de ideas, a menudo en un contexto de producción de software.	Emprendimiento
CB2. Que los estudiantes sepan comunicar sus conclusiones y los conocimientos y razones últimas que las sustentan a públicos especializados y no especializados de un modo claro y sin ambigüedades.	Comunicación oral y escrita
CB3. Que los estudiantes posean habilidades de aprendizaje que les permitan continuar estudiando de un modo que habrá de ser en gran medida autodirigido o autónomo.	Aprendizaje continuo
CB4. Capacidad para trabajar en equipo y para relacionarse con otras personas del mismo o distinto ámbito profesional.	Trabajo colaborativo

Competencia	Indicador	Escala de calificación			
		1	2	3	4
CB1. Poseer y comprender conocimientos que aporten una base u oportunidad de ser creativos en el desarrollo y/o aplicación de ideas, a menudo en un contexto de producción de software.	Adoptar enfoques creativos en relación al contenido propio de la situación y al modo de realización.	Se ofrecen pocas ideas innovadoras o no son adecuadas para la situación estudiada.	Suficientes ideas y enfoques bien adaptados a la situación planteada, pero sin especificar correctamente el modo de ejecución.	Adopción de enfoques creativos al contenido adecuado de la situación y un correcto planteamiento del modo de realización.	Adopta enfoques creativos bien adaptados y ampliados en el tiempo.
	Evaluar críticamente los datos y sacar conclusiones.	Se realiza un análisis y evaluación limitado de los datos sin obtener conclusiones.	Evaluación crítica correcta de los datos, pero limitada y unas conclusiones claras pero escasas.	Evaluación crítica correcta de los datos definiendo unas conclusiones claras y detalladas.	Evaluación crítica de calidad, obteniéndose un gran número de conclusiones claras y detalladas.
	Introducir soluciones singulares o punteras que mejoran varios aspectos o ámbitos de actuación.	No se lleva a cabo la adaptación de soluciones a las situaciones planteadas.	Limitado manejo de soluciones o escaso interés o novedad de estas.	Correcta adaptación de ideas y soluciones novedosas que permiten apreciar las mejoras.	Alto número de soluciones singulares que resuelven y se adaptan a un número alto también de ámbitos de actuación.
	Adoptar un punto de vista crítico y autónomo respecto al conocimiento relativo a su propia disciplina.	No se argumenta de modo racional y coherente los propios puntos de vista, para poder adoptar un punto de vista crítico y autónomo.	En ocasiones se adopta en los problemas una actitud personal y críticas ante las cuestiones teóricas y prácticas, pero no estando siempre debidamente fundamentadas.	En la mayoría de las ocasiones se adopta en los problemas una actitud personal y crítica ante las cuestiones teóricas y prácticas, pero no estando siempre debidamente fundamentadas.	Siempre adopta un punto de vista crítico y autónomo.

CB2. Que los estudiantes sepan comunicar sus conclusiones y los conocimientos y razones últimas que las sustentan a públicos especializados y no especializados de un modo claro y sin ambigüedades.	Crear textos comprensibles, de fácil lectura y organizado en párrafos.	Crea textos confusos y desordenados.	Crea textos claros pero desordenados en algunas partes del documento.	Crea textos claros y ordenados en la mayor parte del documento.	Crea textos claros y ordenados en todo el documento.
	Utilizar buena gramática y ortografía en sus documentos.	Incumple con los indicadores requeridos en la mayor parte del documento.	Sin faltas, vocabulario mejorable y puntuación gramatical incorrecta en ocasiones.	Sin faltas y vocabulario adecuado, pero puntuación gramatical incorrecta en ocasiones.	Cumple con todos los indicadores requeridos.
	Generar documentos con una estructura formal (título, autores, fecha, índice, objetivos, introducción, desarrollo, bibliografía, conclusiones).	Los documentos generados no presentan la estructura establecida.	Los documentos generados no presentan objetivos claros.	Los documentos generados no presentan introducción.	Los documentos generados cumplen con todos los indicadores exigidos.
	Realizar edición formal en sus documentos (tamaño de letra, justificación, interlineado y márgenes, imágenes, tablas, anexos).	El documento no presenta ninguno de los criterios requeridos.	El documento presenta interlineado, márgenes inadecuados e imágenes bien incluidas.	El documento presenta un interlineado adecuado, pero no homogéneo.	El documento presenta todos los aspectos requeridos.
	Utilizar correctamente la bibliografía para la creación de documentos (citas al final del documento o pie de página normas APA).	Carece de citas bibliográficas.	Contiene referencias, pero incumple la normativa APA.	Casi todas las citas cumplen los indicadores.	Todas las citas cumplen los indicadores.
	Utilizar un lenguaje claro y académico con ritmo y tono.	Lenguaje coloquial sin ritmo y monótono.	Lenguaje poco claro y monótono.	Falta de tono y ritmo.	Cumple con los requisitos.
	Utiliza expresión corporal y obtiene la atención de la audiencia.	No cumple con los indicadores exigidos.	Expresión corporal adecuada pero no se dirige a la audiencia.	Expresión corporal adecuada pero la atención a la audiencia es esporádica.	Se ajusta a los indicadores exigidos en todo momento.

CB3. Que los estudiantes posean habilidades de aprendizaje que les permitan continuar estudiando de un modo que habrá de ser en gran medida autodirigido o autónomo.	Realizar búsquedas de literatura técnica utilizando bases de datos y otras fuentes de información.	Se realizan pocas búsquedas de literatura técnica utilizando bases de datos y otras fuentes de forma desorganizada.	Se realizan algunas búsquedas de literatura técnica utilizando bases de datos, pero falta precisión.	Se realizan algunas búsquedas de literatura técnica utilizando bases de datos y otras fuentes con adecuada calidad.	Se realizan algunas búsquedas de literatura técnica utilizando bases de datos y otras fuentes con adecuada calidad.
	Identificar, localizar y obtener los datos requeridos.	Se identifican, localizan y obtienen de forma no adecuada los datos requeridos.	Se identifican, localizan y obtienen adecuadamente algunos datos requeridos con insuficiente calidad.	Se identifican, localizan y obtienen adecuadamente algunos datos requeridos con suficiente calidad.	Se identifican, localizan y obtienen adecuadamente los datos requeridos con la adecuada calidad.
	Investigar la aplicación de tecnologías nuevas y emergentes en su disciplina.	Se investiga poco la aplicación de tecnologías nuevas y emergentes, y de forma desorganizada.	Se conocen tecnologías nuevas y emergentes en su disciplina.	Se investigan tecnologías nuevas y emergentes en su disciplina.	Se investiga la aplicación de tecnologías nuevas y emergentes en su disciplina.
	Identificar necesidades de mejora en situaciones y contextos complejos.	Apenas se identifican aspectos de mejora.	Identificación de necesidades, pero de forma limitada o en contextos simples.	Identificación adecuada y completa.	Buena y amplia identificación de necesidades que además aumentó en el tiempo.
CB4. Capacidad para trabajar en equipo y para relacionarse con otras personas del mismo o distinto ámbito profesional.	Es puntual y respetuoso con el calendario de trabajo del grupo.	Ni asiste ni se preocupa por el seguimiento de los temas tratados.	No asiste a las reuniones en fecha y hora, pero está en contacto con los compañeros.	Asiste a todas las reuniones en fecha, pero no es puntual.	Siempre.
	Prepara previamente las sesiones de trabajo del grupo.	Nunca.	Alguna vez.	Casi siempre.	Siempre.
	Muestra esfuerzo y mejora.	Nunca.	Se esfuerza y se preocupa por alcanzar un mínimo.	Casi siempre.	Siempre.
	Contribuye al aprendizaje del grupo.	Nunca.	Alguna vez.	Habitualmente.	Siempre.
	Utiliza los conocimientos propios para tratar de resolver el tema.	Nunca.	Alguna vez.	Habitualmente.	Siempre.
	Sabe escuchar en el debate del grupo sin interrumpir.	Nunca.	Alguna vez.	Habitualmente.	Siempre.
	Aporta ideas al grupo.	Nunca.	Alguna vez.	Habitualmente.	Siempre.
	Hace y recibe críticas constructivas.	Nunca.	Alguna vez.	Habitualmente.	Siempre.

Plan de clase

Unidad 1

Lección 1.1. Lógica y Programación

INSTITUTO TECNOLÓGICO SUPERIOR CEMLAD

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

Materia: **Lógica de Programación**

Nivel: Tercer Nivel (Tecnología Superior)

Nivel: 1

Profesor: Ing. Carlos Salazar MS.

SECUENCIA DE ACTIVIDADES

Tema: Lógica y Programación

OBJETIVOS

1. Conocer el concepto de lógica.
2. Comprender el concepto de programación.
3. Diferenciar entre lógica y programación.
4. Conocer los requisitos para aprender lógica de programación.

DESARROLLO

1. Presentación del profesor y los estudiantes.
2. Evaluación diagnóstica (grupal o personal).
3. Pasar lista.
4. Presentación de la asignatura.
5. Clase magistral. (Lección 1.1 Lógica y Programación EVA).
6. Método de indagación.

TÉCNICAS DIDÁCTICAS

Exposición. Interrogatorio. Lectura en clase. Redacción de conceptos operativos.

MATERIAL DIDÁCTICO

PC. USB. Pantalla. Pizarrón.

BIGLIOGRAFÍA

Carlos Salazar, *Lógica de programación* (2020).

Omar Trejos, *La esencia de la lógica de programación* (1999).

Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín, *Introducción a la lógica de programación* (2017).

Lección 1.2. Metodología para resolver problemas

INSTITUTO TECNOLÓGICO SUPERIOR CEMLAD

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

Materia: **Lógica de Programación**

Nivel: Tercer Nivel (Tecnología Superior)

Nivel: 1

Profesor: Ing. Carlos Salazar MS.

SECUENCIA DE ACTIVIDADES

Tema: Metodología para resolver problemas

OBJETIVOS

1. Solucionar problemas utilizando la metodología estudiada.
2. Conocer la definición e importancia de los algoritmos.
3. Conocer los tipos de algoritmos existentes.
4. Comprender las características y elementos de los algoritmos.
5. Resolver algoritmos informales.

DESARROLLO

1. Pasar lista.
2. Realizar un recordatorio de la clase anterior.
3. Clase Magistral. (Lección 1.2 Metodología para resolver problemas EVA).
4. Realizar ejemplos de resolución de algoritmos informales.
5. Aplicar lo estudiado para crear algoritmos informales.

TÉCNICAS DIDÁCTICAS

Exposición. Interrogatorio. Lectura en clase. Redacción de conceptos operativos.

MATERIAL DIDÁCTICO

PC. USB. Pantalla. Pizarrón.

BIGLIOGRAFÍA

Carlos Salazar, *Lógica de programación* (2020).

Omar Trejos, *La esencia de la lógica de programación* (1999).

Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín, *Introducción a la lógica de programación* (2017).

Lección 1.3. Variables, constantes y operadores

INSTITUTO TECNOLÓGICO SUPERIOR CEMLAD

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

Materia: **Lógica de Programación**

Nivel: Tercer Nivel (Tecnología Superior)

Nivel: 1

Profesor: Ing. Carlos Salazar MS.

SECUENCIA DE ACTIVIDADES

Tema: Variables, constantes y operadores

OBJETIVOS

1. Conocer el uso y la importancia de variables a la hora de crear algoritmos.
2. Comprender el uso de las constantes.
3. Conocer el uso y la importancia de los operadores en la resolución de algoritmos.
4. Aprender que es un dato y sus diferentes tipos.
5. Recordar la regla de prioridades en aritmética.
6. Resolver algoritmos matemáticos.

DESARROLLO

1. Pasar lista.
2. Realizar un recordatorio de la clase anterior.
3. Clase Magistral. (Lección 1.3 Variables, constantes y operadores EVA).
4. Realizar ejemplos de resolución de algoritmos computacionales.
5. Aplicar lo estudiado para crear algoritmos computacionales.

TÉCNICAS DIDÁCTICAS

Exposición. Interrogatorio. Lectura en clase. Redacción de conceptos operativos.

MATERIAL DIDÁCTICO

PC. USB. Pantalla. Pizarrón.

BIGLIOGRAFÍA

Carlos Salazar, *Lógica de programación* (2020).

Omar Trejos, *La esencia de la lógica de programación* (1999).

Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín, *Introducción a la lógica de programación* (2017).

Lección 1.4. Consideraciones algorítmicas sobre el pensamiento humano

INSTITUTO TECNOLÓGICO SUPERIOR CEMLAD

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

Materia: **Lógica de Programación**

Nivel: Tercer Nivel (Tecnología Superior)

Nivel: 1

Profesor: Ing. Carlos Salazar MS.

SECUENCIA DE ACTIVIDADES

Tema: Consideraciones algorítmicas sobre el pensamiento humano

OBJETIVOS

1. Identificar como utilizar la secuencia de acciones.
2. Conocer la estructura de decisión de acción.
3. Comprender la estructura de ciclo de acción.
4. Identificar ejemplos en la vida cotidiana en donde se utilizan las estructuras algorítmicas.

DESARROLLO

1. Pasar lista.
2. Realizar un recordatorio de la clase anterior.
3. Seguir el discurso del profesor. (Lección 1.4 Consideraciones algorítmicas sobre el pensamiento humano EVA).
4. Identificar ejemplos en donde se utilicen las diferentes estructuras algorítmicas.
5. Aplicar lo estudiado para crear algoritmos computacionales utilizando la secuencia de acciones.

TÉCNICAS DIDÁCTICAS

Exposición. Interrogatorio. Lectura en clase. Redacción de conceptos operativos.

MATERIAL DIDÁCTICO

PC. USB. Pantalla. Pizarrón.

BIGLIOGRAFÍA

Carlos Salazar, *Lógica de programación* (2020).

Omar Trejos, *La esencia de la lógica de programación* (1999).

Lección 1.5. Técnicas para representar algoritmos

INSTITUTO TECNOLÓGICO SUPERIOR CEMLAD

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

Materia: **Lógica de Programación**

Nivel: Tercer Nivel (Tecnología Superior)

Nivel: 1

Profesor: Ing. Carlos Salazar MS.

SECUENCIA DE ACTIVIDADES

Tema: Técnicas para representar algoritmos

OBJETIVOS

1. Conocer el uso y la importancia de utilizar diagramas de flujograma para la representación de algoritmos.
2. Conocer el uso y la importancia de utilizar pseudocódigo para la representación de algoritmos.
3. Instalar las aplicaciones DFD y PseInt para la representación de algoritmos.
4. Relacionar los aprendizajes con la creación de algoritmos utilizando las herramientas.
5. Resolver ejercicios con guías establecidas.

DESARROLLO

1. Pasar lista.
2. Realizar un recordatorio de la clase anterior.
3. Seguir el discurso del profesor. (Lección 1.5 Técnicas para representar algoritmos EVA).
4. Instalar y configurar la herramienta DFD y realizar un primer acercamiento.
5. Instalar y configurar la herramienta PseInt y realizar un primer acercamiento.
6. Crear algoritmos utilizando estas herramientas de representación.

TÉCNICAS DIDÁCTICAS

Exposición. Interrogatorio. Lectura en clase. Redacción de conceptos operativos.

MATERIAL DIDÁCTICO

PC. USB. Pantalla. Pizarrón.

BIGLIOGRAFÍA

Carlos Salazar, *Lógica de programación* (2020).

Omar Trejos, *La esencia de la lógica de programación* (1999).

Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín, *Introducción a la lógica de programación* (2017) p. 47-66

Laboratorio 1

1. DATOS INFORMATIVOS

a. DOCENTE
Ing. Carlos Salazar Ms.

b. Unidad I. Introducción a la programación

c. Lecciones:
1.1 Lógica y programación.
1.2 Metodología para resolver problemas.
1.3 Variables constantes y operadores.
1.4 Consideraciones algorítmicas sobre el pensamiento humano.

1.5 Técnicas para representar algoritmos.

2. DATOS DE LA PRÁCTICA

a. OBJETIVO GENERAL
Preparar en competencias profesionales al Tecnólogo Superior en Desarrollo de Software para su desempeño profesional, permitiéndole aplicar los conocimientos, métodos y técnicas adquiridas en la Unidad I.

3. COMPETENCIAS POR LOGRAR (UNIDAD 1)

- CU1. Conocer la importancia de desarrollar la lógica de programación.
- CU2. Entender una metodología para resolver problemas.
- CU3. Resolver algoritmos informales.
- CU4. Conocer la base teórica de algoritmos.
- CU5. Conocer las consideraciones algorítmicas sobre el pensamiento humano.
- CU6. Utilizar variables, constantes y operadores.
- CU7. Crear algoritmos computacionales de forma gráfica utilizando diagramas de flujograma y la estructura secuencial en la herramienta DFD.
- CU8. Crear algoritmos computacionales utilizando pseudocódigo y la estructura secuencial en la herramienta PseInt.

4. EVALUACIÓN POR COMPETENCIAS

Competencias de la Unidad 1

Competencia	Actividad	Escala de calificación				
		0	1	2	3	4
CU1	Actividad 1	No lo hace	Explica los términos y realiza la comparación, sin embargo, no realiza el análisis.	Explica los términos y realiza la comparación y el análisis. Sin embargo, se evidencia poca comprensión del tema.	Explica los términos y realiza la comparación y el análisis con buen nivel de comprensión del tema.	Explica los términos y realiza la comparación y el análisis con un total nivel de comprensión del tema.
CU2	Actividad 2	No lo hace	Se aborda una metodología, pero no la explica con sus palabras.	La explicación evidencia poca comprensión de la metodología.	La explicación evidencia buena comprensión de la metodología.	La explicación evidencia total comprensión de la metodología.
CU3	Actividad 3	No lo hace	Analiza las ventajas y desventajas del uso de una metodología a la hora de crear algoritmos informales.	Analiza las ventajas y desventajas del uso de una metodología a la hora de crear algoritmos informales e identifica si se aplica en la vida diaria.	Analiza las ventajas y desventajas del uso de una metodología a la hora de crear algoritmos informales e identifica si se aplica en la vida diaria y ¿Por qué?	Analiza las ventajas y desventajas del uso de una metodología a la hora de crear algoritmos informales e identifica si se aplica en la vida diaria y ¿Por qué? Además, propone y resuelve un algoritmo informal.
CU4 CU5	Actividad 4	No lo hace	Sintetiza la teoría algorítmica, sin embargo, no identifica como se utilizan los algoritmos en la vida cotidiana.	Sintetiza la teoría algorítmica e identifica como se utilizan en la vida cotidiana con poco nivel de comprensión.	Sintetiza la teoría algorítmica e identifica como se utilizan en la vida cotidiana con buen nivel de comprensión.	Sintetiza la teoría algorítmica e identifica como se utilizan en la vida cotidiana con total nivel de comprensión.
CU6	Actividad 5 (Ejercicio 1)	No lo hace	Utiliza correctamente variables, constantes y operadores en uno de cuatro ejercicios propuestos.	Utiliza correctamente variables, constantes y operadores en dos de cuatro ejercicios propuestos.	Utiliza correctamente variables, constantes y operadores en tres de cuatro ejercicios propuestos.	Utiliza correctamente variables, constantes y operadores en todos los ejercicios propuestos.
CU7	Actividad 5 (Ejercicio 2)	No lo hace	Crear algoritmos computacionales evidenciando un correcto diseño y ejecución de al menos uno de cuatro literales propuestos en DFD.	Crear algoritmos computacionales evidenciando un correcto diseño y ejecución de al menos dos de cuatro literales propuestos en DFD.	Crear algoritmos computacionales evidenciando un correcto diseño y ejecución de al menos tres de cuatro literales propuestos en DFD.	Crear algoritmos computacionales evidenciando un correcto diseño y ejecución de todos los literales propuestos en DFD.
CU8	Actividad 5 (Ejercicio 3)	No lo hace	Crear algoritmos computacionales evidenciando un correcto manejo de la estructura secuencial de al menos uno de cuatro literales propuestos en PseInt.	Crear algoritmos computacionales evidenciando un correcto manejo de la estructura secuencial de al menos dos de cuatro literales propuestos en PseInt.	Crear algoritmos computacionales evidenciando un correcto manejo de la estructura secuencial de al menos tres de cuatro literales propuestos en PseInt.	Crear algoritmos computacionales evidenciando un correcto manejo de la estructura secuencial de todos los literales propuestos en PseInt.
CU9	Actividad 6	No lo hace	Compara dos de las tres herramientas propuestas.	Compara las tres herramientas solicitadas, sin embargo, se evidencia poca comprensión del tema.	Compara las tres herramientas solicitadas con una buena comprensión del tema.	Compara las tres herramientas solicitadas con una total comprensión del tema.
CU9	Actividad 7	No lo hace	Instala, pero no describe con sus palabras el proceso.	Instala y describe con sus palabras el proceso, sin embargo, se observa falta de especificaciones técnicas en el procedimiento.	Instala y describe con sus palabras el proceso con un uso adecuado de terminología técnica en el proceso	Instala y describe con sus palabras el proceso con un total manejo de lenguaje técnico en el procedimiento.
CU9	Actividad 8	No lo hace	Crea un videojuego utilizando variables y operadores, sin embargo, no explica como lo hizo.	Crea un videojuego utilizando variables y operadores, sin embargo, en la explicación se evidencia poco dominio del tema.	Crea un video juego utilizando variables y operadores con una explicación clara y poco técnica de cómo lo realizó.	Crea un videojuego utilizando variables y operadores con una explicación clara y técnica sobre el procedimiento.

5. DESARROLLO

Actividad 1. Explique con sus palabras: lógica y programación. Realice una comparación entre estos dos términos y analice la importancia del desarrollo de la lógica de programación para un programador o desarrollador de software.

Actividad 2. Con sus propias palabras explique la metodología utilizada para resolver problemas.

Actividad 3. Analice las ventajas y desventajas de usar una metodología para crear algoritmos informales. ¿Se aplica en la vida diaria? Sí, no. ¿Por qué? Plantee un algoritmo informal y resuélvalo.

Actividad 4. Sintetice la teoría algorítmica e identifique: ¿Cómo se utilizan los algoritmos en la vida cotidiana?

Actividad 5. Aplique lo aprendido resolviendo los siguientes ejercicios:

Ejercicio 1:

Automatice las siguientes operaciones utilizando:

DFD (Literal 1 y 2)

PseInt (Literal 3 y 4)

Qué valores tienen las variables a, b y c

1)	2)	3)	4)

Ejercicio 2:

Calcular el área de las siguientes figuras geométricas utilizando diagramas de flujograma con la herramienta DFD. Se deben solicitar los datos necesarios por teclado y mostrar el resultado final en pantalla.

Se debe insertar una imagen del diseño en DFD y la ejecución para evidenciar el buen funcionamiento del algoritmo.

Ejercicio 3:

Resolver los siguientes ejercicios utilizando pseudocódigo con la herramienta PseInt. Se debe insertar una imagen para el código y otra para la ejecución.

1. Crear un algoritmo que convierta un valor ingresado en pies a yardas, metros, pulgadas y centímetros.

2. Imagínese que usted es un desarrollador de software en una institución financiera, y se le solicita crear un algoritmo para:
   - Simular la ganancia de un cliente si invierte su capital a una tasa del 2% de interés mensual.
   - ¿Cuánto dinero gana un cliente en un año?
3. Un vendedor recibe un sueldo base más un 10% extra de su sueldo por comisión de cada venta realizada. El vendedor desea saber cuánto dinero obtendrá por concepto de comisiones por las tres ventas que realizó en el mes y el total que recibirá como pago tomando en cuenta su sueldo base y comisiones.
4. Un alumno desea saber cuál será su calificación final en la materia de Algoritmos. Dicha calificación se compone de los siguientes porcentajes:

   55% del promedio de sus tres calificaciones parciales.

   30% de la calificación del examen final.

   15% de la calificación de un trabajo final.

Actividad 6. Elabore una comparación sobre la herramienta PseInt, DFD y Scratch.

Actividad 7. Instale y describa los pasos a seguir para instalar Scratch en su computador.

Actividad 8. Cree un videojuego utilizando Scratch en donde:

- Ponga en práctica el manejo de variables y operadores.
- Debe proporcionar el link de acceso para verificar el correcto funcionamiento.
- Explicar cómo lo realizó (en vivo).

6. LO MÁS RECOMENDADO

Lógica de Programación por Carlos Salazar (IST CEMLAD).

La esencia de la lógica de programación por Omar Iván Trejos.

Introducción a la lógica de programación por Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín.

7. NO DEJES DE VER

Los recursos de aprendizaje propuestos en el aula virtual de la asignatura (videos, lecciones, libros).

8. ENTREGA

Se debe subir el laboratorio resuelto en formato PDF a la plataforma virtual.

Unidad 2

Lección 2.1. Decisiones

INSTITUTO TECNOLÓGICO SUPERIOR CEMLAD

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

Materia: **Lógica de Programación**

Nivel: Tercer Nivel (Tecnología Superior)

Nivel: 1

Profesor: Ing. Carlos Salazar MS.

SECUENCIA DE ACTIVIDADES

Tema: Decisiones – Marco Teórico

OBJETIVOS

1. Comprender que es, cuando y como utilizar la estructura de decisión.
2. Entender las posibles variaciones que se pueden combinar con esta estructura.
3. Identificar escenarios de la vida cotidiana en donde se utilizan estas estructuras.

DESARROLLO

1. Pasar lista.
2. Recordatorio de la Unidad 1 y una breve introducción a la Unidad 2.
3. Clase magistral. (Lección 2.1 Decisiones EVA).
4. Identificar escenarios en la vida cotidiana en donde se utiliza estas estructuras.

TÉCNICAS DIDÁCTICAS

Exposición. Interrogatorio. Lectura en clase. Redacción de conceptos operativos.

MATERIAL DIDÁCTICO

PC. USB. Pantalla. Pizarrón.

BIGLIOGRAFÍA

Carlos Salazar, *Lógica de programación* (2020).

Omar Trejos, *La esencia de la lógica de programación* (1999).

Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín, *Introducción a la lógica de programación* (2017).

Lección 2.1. Decisiones

INSTITUTO TECNOLÓGICO SUPERIOR CEMLAD

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

Materia: **Lógica de Programación**

Nivel: Tercer Nivel (Tecnología Superior)

Nivel: 1

Profesor: Ing. Carlos Salazar MS.

SECUENCIA DE ACTIVIDADES

Tema: Decisiones – Manos a la obra

OBJETIVOS

1. Comprender la diferencia entre las variaciones de la estructura de decisión.
2. Crear algoritmos en DFD aplicando la estructura de decisión.

DESARROLLO

1. Pasar lista.
2. Realizar un recordatorio de la clase anterior.
3. Lluvia de ideas sobre la aplicación de las variaciones de esta estructura.
4. Crear algoritmos en DFD aplicando esta estructura.

TÉCNICAS DIDÁCTICAS

Exposición. Interrogatorio. Lectura en clase. Redacción de conceptos operativos.

MATERIAL DIDÁCTICO

PC. USB. Pantalla. Pizarrón.

BIGLIOGRAFÍA

Carlos Salazar, *Lógica de programación* (2020).

Omar Trejos, *La esencia de la lógica de programación* (1999).

Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín, *Introducción a la lógica de programación* (2017).

Lección 2.1. Decisiones

INSTITUTO TECNOLÓGICO SUPERIOR CEMLAD

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

Materia: **Lógica de Programación**

Nivel: Tercer Nivel (Tecnología Superior)

Nivel: 1

Profesor: Ing. Carlos Salazar MS.

SECUENCIA DE ACTIVIDADES

Tema: Decisiones – Manos a la obra

OBJETIVOS

1. Comprender la diferencia entre las variaciones de la estructura de decisión.
2. Crear algoritmos en PseInt.

DESARROLLO

1. Pasar lista.
2. Realizar un recordatorio de la clase anterior.
3. Crear algoritmos en PseInt a partir de los ejercicios realizados la clase anterior en DFD.

TÉCNICAS DIDÁCTICAS

Exposición. Interrogatorio. Lectura en clase. Redacción de conceptos operativos.

MATERIAL DIDÁCTICO

PC. USB. Pantalla. Pizarrón.

BIGLIOGRAFÍA

Carlos Salazar, *Lógica de programación* (2020).

Omar Trejos, *La esencia de la lógica de programación* (1999).

Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín, *Introducción a la lógica de programación* (2017).

Lección 2.2. Ciclos

INSTITUTO TECNOLÓGICO SUPERIOR CEMLAD

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

Materia: **Lógica de Programación**

Nivel: Tercer Nivel (Tecnología Superior)

Nivel: 1

Profesor: Ing. Carlos Salazar MS.

SECUENCIA DE ACTIVIDADES

Tema: Ciclos – Marco Teórico

OBJETIVOS

1. Comprender que es, cuando y como utilizar la estructura Para.
2. Comprender que es, cuando y como utilizar la estructura Mientras.
3. Comprender que es, cuando y como utilizar la estructura Repetir.

DESARROLLO

1. Pasar lista.
2. Realizar un recordatorio de la clase anterior.
3. Seguir el discurso del profesor. (Lección 2.2 Decisiones EVA).
4. Identificar ejemplos en la vida cotidiana donde se utilicen las diferentes estructuras repetitivas.

TÉCNICAS DIDÁCTICAS

Exposición. Interrogatorio. Lectura en clase. Redacción de conceptos operativos.

MATERIAL DIDÁCTICO

PC. USB. Pantalla. Pizarrón.

BIGLIOGRAFÍA

Carlos Salazar, *Lógica de programación* (2020).

Omar Trejos, *La esencia de la lógica de programación* (1999).

Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín, *Introducción a la lógica de programación* (2017).

Lección 2.2. Ciclos - Para

INSTITUTO TECNOLÓGICO SUPERIOR CEMLAD

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

Materia: **Lógica de Programación**

Nivel: Tercer Nivel (Tecnología Superior)

Nivel: 1

Profesor: Ing. Carlos Salazar MS.

SECUENCIA DE ACTIVIDADES

Tema: Ciclos – Manos a la obra

OBJETIVOS

1. Crear algoritmos utilizando el ciclo repetitivo "Para" en DFD y PseInt.

DESARROLLO

1. Pasar lista.
2. Realizar un recordatorio de la clase anterior.
3. Crear algoritmos utilizando la estructura repetitiva "Para" en DFD y PseInt.
   - Organizar grupos de trabajo para resolver un ejercicio y después exponerlo en clase.

TÉCNICAS DIDÁCTICAS

Exposición. Interrogatorio. Lectura en clase. Redacción de conceptos operativos.

MATERIAL DIDÁCTICO

PC. USB. Pantalla. Pizarrón.

BIGLIOGRAFÍA

Carlos Salazar, *Lógica de programación* (2020).

Omar Trejos, *La esencia de la lógica de programación* (1999).

Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín, *Introducción a la lógica de programación* (2017).

Lección 2.2. Ciclos - Mientras

INSTITUTO TECNOLÓGICO SUPERIOR CEMLAD

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

Materia: **Lógica de Programación**

Nivel: Tercer Nivel (Tecnología Superior)

Nivel: 1

Profesor: Ing. Carlos Salazar MS.

SECUENCIA DE ACTIVIDADES

Tema: Ciclos – Manos a la obra

OBJETIVOS

1. Crear algoritmos utilizando el ciclo repetitivo "Mientras" en DFD y PseInt.

DESARROLLO

1. Pasar lista.
2. Realizar un recordatorio de la clase anterior.
3. Crear algoritmos utilizando la estructura repetitiva "Mientras" en DFD y PseInt.
   - Organizar grupos de trabajo para resolver un ejercicio y después exponerlo en clase.

TÉCNICAS DIDÁCTICAS

Exposición. Interrogatorio. Lectura en clase. Redacción de conceptos operativos.

MATERIAL DIDÁCTICO

PC. USB. Pantalla. Pizarrón.

BIGLIOGRAFÍA

Carlos Salazar, *Lógica de programación* (2020).

Omar Trejos, *La esencia de la lógica de programación* (1999).

Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín, *Introducción a la lógica de programación* (2017).

Lección 2.2. Ciclos - Repetir

INSTITUTO TECNOLÓGICO SUPERIOR CEMLAD

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

Materia: **Lógica de Programación**

Nivel: Tercer Nivel (Tecnología Superior)

Nivel: 1

Profesor: Ing. Carlos Salazar MS.

SECUENCIA DE ACTIVIDADES

Tema: Ciclos – Manos a la obra

OBJETIVOS

1. Crear algoritmos utilizando el ciclo repetitivo "Repetir" en DFD y PseInt.

DESARROLLO

1. Pasar lista.
2. Realizar un recordatorio de la clase anterior.
3. Crear algoritmos utilizando la estructura repetitiva "Repetir" en DFD y PseInt.
   - Organizar grupos de trabajo para resolver un ejercicio y después exponerlo en clase.

TÉCNICAS DIDÁCTICAS

Exposición. Interrogatorio. Lectura en clase. Redacción de conceptos operativos.

MATERIAL DIDÁCTICO

PC. USB. Pantalla. Pizarrón.

BIGLIOGRAFÍA

Carlos Salazar, *Lógica de programación* (2020).

Omar Trejos, *La esencia de la lógica de programación* (1999).

Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín, *Introducción a la lógica de programación* (2017).

Lección 2.2. Ciclos – Estructuras mixtas

INSTITUTO TECNOLÓGICO SUPERIOR CEMLAD

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

Materia: **Lógica de Programación**

Nivel: Tercer Nivel (Tecnología Superior)

Nivel: 1

Profesor: Ing. Carlos Salazar MS.

SECUENCIA DE ACTIVIDADES

Tema: Ciclos – Manos a la obra

OBJETIVOS

2. Crear algoritmos utilizando estructuras Mixtas en DFD y PseInt.

DESARROLLO

4. Pasar lista.
5. Realizar un recordatorio de la clase anterior.
6. Crear algoritmos utilizando estructuras Mixtas en DFD y PseInt.
   - Organizar grupos de trabajo para resolver un ejercicio y después exponerlo en clase.

TÉCNICAS DIDÁCTICAS

Exposición. Interrogatorio. Lectura en clase. Redacción de conceptos operativos.

MATERIAL DIDÁCTICO

PC. USB. Pantalla. Pizarrón.

BIGLIOGRAFÍA

Carlos Salazar, *Lógica de programación* (2020).

Omar Trejos, *La esencia de la lógica de programación* (1999).

Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín, *Introducción a la lógica de programación* (2017).

Laboratorio 2

1. DATOS INFORMATIVOS

a. DOCENTE
Ing. Carlos Salazar Ms.

b. Unidad II. Estructuras

c. Lecciones
2.1 Decisiones
2.2 Ciclos
2.3 Estructuras Mixtas

2. DATOS DE LA PRÁCTICA

a. OBJETIVO GENERAL
Preparar en competencias profesionales al Tecnólogo Superior en Desarrollo de Software para su desempeño profesional, permitiéndole aplicar los conocimientos, métodos y técnicas adquiridas en la Unidad II.

3. COMPETENCIAS A LOGRAR (UNIDAD 2)

- CU2.1. Comprender que es, cuando y como utilizar las estructuras de decisión.
- CU2.2 Crear algoritmos utilizando la estructura de decisión en DFD y PseInt.
- CU2.3 Comprender que es, cuando y como utilizar las estructuras de ciclos.
- CU2.4 Crear algoritmos utilizando la estructura de ciclos en DFD y PseInt.
- CU2.5 Comprender que es, cuando y como utilizar las estructuras mixtas.
- CU2.6 Crear algoritmos utilizando estructuras mixtas en DFD y PseInt.

4. EVALUACIÓN POR COMPETENCIAS

Competencia	Actividad	Escala de calificación				
		0	1	2	3	4
CU2.1	Actividad 1	No lo hace	Establece las diferencias entre dos de cuatro variaciones.	Establece las diferencias entre las cuatro variaciones, sin embargo, se evidencia poca comprensión del tema.	Establece las diferencias entre las cuatro variaciones con buena de comprensión del tema.	Establece las diferencias entre las cuatro variaciones con total comprensión del tema.
CU2.1	Actividad 2	No lo hace	Proporciona un texto sin comparar las estructuras.	Realiza la comparación, sin embargo, se evidencia poca comprensión del tema.	Realiza la comparación con buena comprensión del tema.	Realiza la comparación con total comprensión del tema.
CU2.3	Actividad 3	No lo hace	Realiza las características de dos estructuras.	Realiza las características de las tres estructuras, sin embargo, se evidencia poca comprensión del tema, además, el texto carece de lenguaje técnico y es poco comprensivo.	Realiza las características de las tres estructuras con buena comprensión del tema y el texto carece de lenguaje técnico, pero es comprensivo.	Realiza las características de las tres estructuras con total comprensión del tema utilizando un lenguaje técnico y comprensivo
CU2.1 CU2.3	Actividad 4	No lo hace	Identifica situaciones de la vida diaria en dos de cinco estructuras.	Identifica situaciones de la vida diaria en tres de cinco estructuras.	Identifica situaciones de la vida diaria en cuatro de cinco estructuras.	Identifica situaciones de la vida diaria en todas las estructuras solicitadas.
CU2.2 CU2.4 CU2.6	Actividad 5 (Ejercicio 1)	No lo hace	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no es acorde a lo solicitado.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no alcanza completamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, alcanza medianamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt alcanzando el objetivo.
CU2.2 CU2.4 CU2.6	Actividad 5 (Ejercicio 2)	No lo hace	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no es acorde con lo solicitado.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no alcanza completamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, alcanza medianamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt alcanzando el objetivo.
CU2.2 CU2.4 CU2.6	Actividad 5 (Ejercicio 3)	No lo hace	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no es acorde a lo solicitado.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no alcanza completamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, alcanza medianamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt alcanzando el objetivo.
CU2.2 CU2.4 CU2.6	Actividad 5 (Ejercicio 4)	No lo hace	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no es acorde a lo solicitado.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no alcanza completamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, alcanza medianamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt alcanzando el objetivo.
CU2.2 CU2.4 CU2.6	Actividad 5 (Ejercicio 5)	No lo hace	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no es acorde a lo solicitado.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, no alcanza completamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt, sin embargo, alcanza medianamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en DFD y PseInt alcanzando el objetivo.
CU2.1 CU2.3 CU2.5	Actividad 6	No lo hace	Crea un videojuego utilizando lo estudiado en la Unidad 1 y Unidad 2, sin embargo, no explica como lo hizo.	Crea un videojuego utilizando lo estudiado en la Unidad 1 y Unidad 2, sin embargo, en la explicación se evidencia poco dominio del tema.	Crea un video juego utilizando lo estudiado en la Unidad 1 y 2 con una explicación clara y poco técnica de cómo lo realizó.	Crea un videojuego utilizando lo abordado en la Unidad 1 y 2 con una explicación clara y técnica sobre el procedimiento.

5. DESARROLLO

Actividad 1. Establezca las diferencias entre las variaciones de la estructura de decisión estudiadas en la lección 2.1.

Actividad 2. Compare la estructura de decisión y selección múltiple.

Actividad 3. Enumere las características del ciclo repetitivo (Para, mientras y repetir).

Actividad 4. Identifique y analice situaciones en su vida diaria en donde se podría utilizar la estructura de decisión y ciclos (Si-Sino, selección múltiple, para, mientras, repetir).

Actividad 5. Aplique lo aprendido resolviendo los siguientes ejercicios:

Ejercicio 1

En un hospital se pide realizar un algoritmo sobre la altura de un paciente. Que si la altura es menor o igual a 150 cm, envíe el mensaje: "Persona de Altura baja"; si la altura está entre 151 y 170 cm, escriba: "Persona de Altura Media" y si la altura es mayor a 171 cm escriba: "Persona de Altura Alta". Exprese el algoritmo usando Pseudocódigo y diagramas de flujo.

Se requiere analizar (Entrada, Proceso y Salida), crear el algoritmo y representarlo en DFD y PseInt. Se debe proporcionar una captura de pantalla del diagrama DFD, PseInt con su respectiva ejecución.

Ejercicio 2

Una institución financiera ha solicitado a la dirección de tecnología desarrollar un algoritmo para determinar la cantidad de dinero recibida por un trabajador por las horas semanales trabajadas en una empresa. Sabiendo que: cuando las horas de trabajo exceden de 40 el resto se consideran horas extras y se pagan el doble de una hora normal, cuando no exceden de 8; si las horas extras exceden de 8, se pagan las primeras 8 el doble de lo que se paga una hora normal y el resto al triple. Del trabajador se proporcionan los siguientes datos: nombres, número de horas trabajadas en la semana y valor recibido por una hora normal de trabajo.

Se requiere analizar (Entrada, Proceso y Salida), crear el algoritmo y representarlo en DFD y PseInt. Se debe proporcionar una captura de pantalla del diagrama DFD, PseInt con su respectiva ejecución.

Ejercicio 3

Un vendedor ha realizado N ventas y desea saber:

- ¿Cuántas fueron de $1 a $10.000?
- ¿Cuántas fueron por más de $10.000 pero por menos de $20.000?
- ¿Cuánto fue el monto de las ventas de cada una y el monto global?
- Realice un algoritmo para determinar los totales.

Se requiere analizar (Entrada, Proceso y Salida), crear el algoritmo y representarlo en DFD y PseInt. Se debe proporcionar una captura de pantalla del diagrama DFD, PseInt con su respectiva ejecución.

Ejercicio 4

Realice un algoritmo para determinar cuánto pagará una persona que adquiere N artículos que están de promoción. Considere que si su precio es mayor o igual a

$200 se le aplica un descuento de 15%, y si su precio es mayor a $100 pero menor a $200, el descuento es de 12%; de lo contrario, sólo se le aplica 10% de descuento. Se debe saber cuál es el costo y el descuento que tendrá cada uno de los artículos y finalmente cuánto se pagará por todos los artículos vendidos.

Se requiere analizar (Entrada, Proceso y Salida), crear el algoritmo y representarlo en DFD y PseInt. Se debe proporcionar una captura de pantalla del diagrama DFD, PseInt con su respectiva ejecución.

Ejercicio 5

Una empresa fumiga cosechas contra una gran variedad de plagas. Los valores cobrados dependen de lo que se desee fumigar y de cuántas hectáreas se fumigan, según la siguiente distribución:

- Caso 1: fumigación contra malas hierbas, $10 por hectárea.
- Caso 2: fumigación contra langostas, $15 por hectárea.
- Caso 3: fumigación contra gusanos, $20 por hectárea.
- Caso 4: fumigación contra todo lo anterior, $30 por hectárea.

Si el área por fumigar es mayor de 1.000 hectáreas, el granjero goza de 10% de descuento. A cualquier granjero cuya cuenta sobrepase los $3.000 se le descuenta 15% sobre la cantidad que exceda dicho precio. Si se aplican ambos conceptos, el de la superficie se considera primero. Por cada pedido hay la siguiente información: nombre del granjero, tipo de fumigación (1, 2, 3, 4) y número de hectáreas por fumigar. Por cada solicitud se debe suministrar: nombre del granjero y valor por pagar, además, el algoritmo debe permitir realizar N cotizaciones de servicio hasta que el usuario lo desee.

Se requiere analizar (Entrada, Proceso y Salida), crear el algoritmo y representarlo en DFD y PseInt. Se debe proporcionar una captura de pantalla del diagrama DFD, PseInt con su respectiva ejecución.

Actividad 6:

- Cree un videojuego utilizando Scratch en donde:
- Ponga en práctica el manejo de lo aprendido en la Unidad 1 y Unidad 2.
- Proporcione el link de acceso para verificar el correcto funcionamiento.
- Explique cómo lo realizó (en vivo).

6. LO MÁS RECOMENDADO

Lógica de programación por Carlos Salazar (IST CEMLAD)

La esencia de la lógica de programación por Omar Iván Trejos

Introducción a la lógica de programación por Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín

7. NO DEJES DE VER

Recursos de aprendizaje de la asignatura (videos, lecciones, libros).

8. ENTREGA

Se debe subir el laboratorio resuelto en formato PDF a la plataforma virtual.

Unidad 3

Lección 3.1. Arreglos

INSTITUTO TECNOLÓGICO SUPERIOR CEMLAD

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

Materia: **Lógica de Programación**

Nivel: Tercer Nivel (Tecnología Superior)

Nivel: 1

Profesor: Ing. Carlos Salazar MS.

SECUENCIA DE ACTIVIDADES

Tema: Arreglos

OBJETIVOS

1. Comprender que es, cuando y como utilizar un arreglo.
2. Estudiar como manipular un arreglo de manera manual y automática
3. Identificar escenarios en donde sería útil su aplicación.

DESARROLLO

1. Pasar lista.
2. Recordatorio de la Unidad 2 y una breve introducción a la Unidad 3.
3. Clase magistral (Lección 3.1 Arreglos EVA).
4. Identificar escenarios en donde sería útil su aplicación.

TÉCNICAS DIDÁCTICAS

Exposición. Interrogatorio. Lectura en clase. Redacción de conceptos operativos.

MATERIAL DIDÁCTICO

PC. USB. Pantalla. Pizarrón.

BIGLIOGRAFÍA

Carlos Salazar, *Lógica de programación* (2020).

Omar Trejos, *La esencia de la lógica de programación* (1999).

Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín, *Introducción a la lógica de programación* (2017).

Lección 3.1. Arreglos

INSTITUTO TECNOLÓGICO SUPERIOR CEMLAD

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

Materia: **Lógica de Programación**

Nivel: Tercer Nivel (Tecnología Superior)

Nivel: 1

Profesor: Ing. Carlos Salazar MS.

SECUENCIA DE ACTIVIDADES

Tema: Arreglos – Manos a la obra

OBJETIVOS

1. Crear algoritmos utilizando Arreglos y las estructuras estudiadas en la Unidad 1 y Unidad 2.

DESARROLLO

1. Pasar lista.
2. Realizar un recordatorio de la clase anterior.
3. Crear algoritmos en PseInt utilizando arreglos.

TÉCNICAS DIDÁCTICAS

Exposición. Interrogatorio. Lectura en clase. Redacción de conceptos operativos.

MATERIAL DIDÁCTICO

PC. USB. Pantalla. Pizarrón.

BIGLIOGRAFÍA

Carlos Salazar, *Lógica de programación* (2020).

Omar Trejos, *La esencia de la lógica de programación* (1999).

Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín, *Introducción a la lógica de programación* (2017).

Lección 3.2 Matrices

INSTITUTO TECNOLÓGICO SUPERIOR CEMLAD

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

Materia: **Lógica de Programación**

Nivel: Tercer Nivel (Tecnología Superior)

Nivel: 1

Profesor: Ing. Carlos Salazar MS.

SECUENCIA DE ACTIVIDADES

Tema: Matrices

OBJETIVOS

1. Comprender que es, cuando y como utilizar matrices.
2. Estudiar como manipular una matriz de manera manual y automática.
3. Identificar escenarios en donde sería útil su aplicación.

DESARROLLO

1. Pasar lista.
2. Recordatorio de la clase anterior.
3. Clase magistral (Lección 3.2 Matrices EVA).
4. Identificar escenarios en donde sería útil su aplicación.

TÉCNICAS DIDÁCTICAS

Exposición. Interrogatorio. Lectura en clase. Redacción de conceptos operativos.

MATERIAL DIDÁCTICO

PC. USB. Pantalla. Pizarrón.

BIGLIOGRAFÍA

Carlos Salazar, *Lógica de programación* (2020).

Omar Trejos, *La esencia de la lógica de programación* (1999).

Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín, *Introducción a la lógica de programación* (2017).

Lección 3.2 Matrices

INSTITUTO TECNOLÓGICO SUPERIOR CEMLAD

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

Materia: **Lógica de Programación**

Nivel: Tercer Nivel (Tecnología Superior)

Nivel: 1

Profesor: Ing. Carlos Salazar MS.

SECUENCIA DE ACTIVIDADES

Tema: Matrices – Manos a la obra

OBJETIVOS

1. Crear algoritmos utilizando matrices en PseInt.

DESARROLLO

1. Pasar lista.
2. Realizar un recordatorio de la clase anterior.
3. Crear algoritmos en PseInt utilizando matrices.

TÉCNICAS DIDÁCTICAS

Exposición. Interrogatorio. Lectura en clase. Redacción de conceptos operativos.

MATERIAL DIDÁCTICO

PC. USB. Pantalla. Pizarrón.

BIGLIOGRAFÍA

Carlos Salazar, *Lógica de programación* (2020).

Omar Trejos, *La esencia de la lógica de programación* (1999).

Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín, *Introducción a la lógica de programación* (2017).

Lección 3.3 Funciones y Subprocesos

INSTITUTO TECNOLÓGICO SUPERIOR CEMLAD

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

Materia: **Lógica de Programación**

Nivel: Tercer Nivel (Tecnología Superior)

Nivel: 1

Profesor: Ing. Carlos Salazar MS.

SECUENCIA DE ACTIVIDADES

Tema: Funciones y Subprocesos

OBJETIVOS

1. Comprender que son, cuando y como utilizar funciones y subprocesos.
2. Crear algoritmos utilizando funciones y subprocesos.

DESARROLLO

1. Pasar lista.
2. Recordatorio de la clase anterior.
3. Clase magistral (Lección 3.3 Funciones y subprocesos EVA).

TÉCNICAS DIDÁCTICAS

Exposición. Interrogatorio. Lectura en clase. Redacción de conceptos operativos.

MATERIAL DIDÁCTICO

PC. USB. Pantalla. Pizarrón.

BIGLIOGRAFÍA

Carlos Salazar, *Lógica de programación* (2020).

Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín, *Introducción a la lógica de programación* (2017).

Lección 3.4. Consejos y reflexiones

INSTITUTO TECNOLÓGICO SUPERIOR CEMLAD

TECNOLOGÍA SUPERIOR EN DESARROLLO DE SOFTWARE

Materia: **Lógica de Programación**

Nivel: Tercer Nivel (Tecnología Superior)

Nivel: 1

Profesor: Ing. Carlos Salazar MS.

SECUENCIA DE ACTIVIDADES

Tema: Consejos y reflexiones

OBJETIVOS

1. Exponer consejos y reflexiones a la hora de crear algoritmos.

DESARROLLO

1. Pasar lista.
2. Realizar un recordatorio de la clase anterior.
3. Exponer los consejos y reflexiones a la hora de crear algoritmos.

TÉCNICAS DIDÁCTICAS

Exposición. Interrogatorio. Lectura en clase. Redacción de conceptos operativos.

MATERIAL DIDÁCTICO

PC. USB. Pantalla. Pizarrón.

BIGLIOGRAFÍA

Carlos Salazar, *Lógica de programación* (2020).
Omar Trejos, *La esencia de la lógica de programación* (1999).

Laboratorio 3

1. DATOS INFORMATIVOS

a. DOCENTE

Ing. Carlos Salazar Ms.

b. Unidad III. Arreglos, Matrices, Funciones y subprocesos

c. Lecciones:

3.1 Arreglos
3.2 Matrices
3.3 Funciones y subprocesos
3.4 Consejos y reflexiones

2. DATOS DE LA PRÁCTICA

a. OBJETIVO GENERAL

Preparar en competencias profesionales al Tecnólogo Superior en Desarrollo de Software para su desempeño profesional, permitiéndole aplicar los conocimientos, métodos y técnicas adquiridas en la Unidad II.

3. COMPETENCIAS POR LOGRAR (UNIDAD 3)

- CU3.1 Comprender cómo funcionan los arreglos y las matrices.
- CU3.2 Utilizar arreglos y matrices en la construcción de algoritmos.
- CU3.3 Comprender la utilidad de las funciones y subprocesos en PseInt.
- CU3.4 Utilizar funciones y subprocesos.
- CU3.5 Conocer los principales consejos y reflexiones a la hora de crear algoritmos.

4. EVALUACIÓN POR COMPETENCIAS

Competencia	Actividad	Escala de calificación				
		0	1	2	3	4
CU3.1	Actividad 1	No lo hace	Establece las diferencias, sin embargo, se evidencia poca comprensión del tema.	Establece las diferencias con buena de comprensión del tema.	Establece las diferencias con total comprensión del tema.	Establece las diferencias con total comprensión del tema y utiliza correctamente un lenguaje técnico.
CU3.1	Actividad 2	No lo hace	Proporciona un texto sin comparar las estructuras.	Realiza la comparación, sin embargo, se evidencia poca comprensión del tema.	Realiza la comparación con buena comprensión del tema.	Realiza la comparación con total comprensión del tema.
CU3.4	Actividad 3	No lo hace	Realiza las características de subprocesos o funciones, pero no de las dos.	Realiza las características, sin embargo, se evidencia poca comprensión del tema, además, el texto carece de lenguaje técnico y es poco comprensivo.	Realiza las características con buena comprensión del tema y el texto carece de lenguaje técnico, pero es comprensivo.	Realiza las características con total comprensión del tema utilizando un lenguaje técnico y comprensivo.
CU3.1 CU3.3 CU3.4	Actividad 4	No lo hace	Identifica situaciones en dos de las cuatro opciones solicitadas.	Identifica situaciones en tres de las cuatro opciones solicitadas.	Identifica situaciones en todas las opciones solicitadas.	Identifica situaciones en todas las opciones solicitadas y el texto es completamente entendible.
CU3.2 CU3.4	Actividad 5 (Ejercicio 1)	No lo hace	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt, sin embargo, no es acorde a lo solicitado.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt, sin embargo, no alcanza completamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt, sin embargo, alcanza medianamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt alcanzando el objetivo.
CU3.2 CU3.4	Actividad 5 (Ejercicio 2)	No lo hace	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt, sin embargo, no es acorde a lo solicitado.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt, sin embargo, no alcanza completamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt, sin embargo, alcanza medianamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt alcanzando el objetivo.
CU3.2 CU3.4	Actividad 5 (Ejercicio 3)	No lo hace	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt, sin embargo, no es acorde a lo solicitado.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt, sin embargo, no alcanza completamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt, sin embargo, alcanza medianamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt alcanzando el objetivo.
CU3.2 CU3.4	Actividad 5 (Ejercicio 4)	No lo hace	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt, sin embargo, no es acorde a lo solicitado.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt, sin embargo, no alcanza completamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt, sin embargo, alcanza medianamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt alcanzando el objetivo.
CU3.2 CU3.4	Actividad 5 (Ejercicio 5)	No lo hace	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt, sin embargo, no es acorde a lo solicitado.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt, sin embargo, no alcanza completamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt, sin embargo, alcanza medianamente el objetivo.	Resuelve el ejercicio realizando el análisis, identificando el algoritmo, lo representa en PseInt alcanzando el objetivo.
CU2.1 CU2.3 CU2.5	Actividad 6	No lo hace	Crea un videojuego utilizando lo estudiado en toda la asignatura, sin embargo, no explica como lo hizo.	Crea un videojuego utilizando lo estudiado en toda la asignatura, sin embargo, en la explicación se evidencia poco dominio del tema.	Crea un video juego utilizando lo estudiado en toda la asignatura con una explicación clara y poco técnica de cómo lo realizó.	Crea un videojuego utilizando lo abordado en toda la asignatura con una explicación clara y técnica sobre el procedimiento.

5. DESARROLLO

Actividad 1. Establezca las diferencias entre vectores y variables

Actividad 2. Compare vectores con matrices.

Actividad 3. Enumere las características de las funciones y subprocesos.

Actividad 4. Identifique y analice en que situaciones son de utilidad los vectores, matrices, funciones y subprocesos.

Actividad 5. Aplique lo aprendido resolviendo los siguientes ejercicios:

Ejercicio 1:

Elaborar un algoritmo que permita ingresar la edad y los nombres de 5 estudiantes y muestre cuántos estudiantes son mayores a 20 años.

Se requiere analizar (Entrada, Proceso y Salida), crear el algoritmo y representarlo en PseInt. Se debe proporcionar una captura de pantalla del pseudocódigo en PseInt con su respectiva ejecución.

Ejercicio 2:

Implementar un algoritmo que permita realizar operaciones sobre matrices de M x N. El programa debe permitir al usuario la selección de alguna de las siguientes operaciones: sumar, restar, multiplicas o trasponer dos matrices.

Se requiere analizar (Entrada, Proceso y Salida), crear el algoritmo y representarlo en PseInt. Se debe proporcionar una captura de pantalla del pseudocódigo en PseInt con su respectiva ejecución.

Ejercicio 3:

Elaborar un algoritmo que forme una matriz de N * M elementos. Cada elemento del arreglo representa las ventas atribuibles a cada uno de los N vendedores de una empresa, para cada uno de los M años de operaciones que ha tenido la misma y, luego calcular:

- El total de ventas de cada vendedor de los M años.
- El total de ventas en cada año.
- El gran total de ventas de la empresa.

Se requiere analizar (Entrada, Proceso y Salida), crear el algoritmo y representarlo en PseInt. Se debe proporcionar una captura de pantalla del pseudocódigo en PseInt con su respectiva ejecución.

Ejercicio 4:

Se desea obtener el grado de eficiencia de N operarios de una fábrica productora de pupitres, de acuerdo con las siguientes condiciones que se les impone para un periodo determinado. Condiciones:

- Ausencia del trabajo < = 3,5 horas.
- Pupitres defectuosos < 300.
- Pupitres producidos > 10000.

- Los grados de eficiencia del trabajador son asignados de la siguiente forma:
    - Si no cumple ninguna condición: Grado = 5
    - Si sólo cumple la primera condición: Grado = 7
    - Si sólo cumple la segunda condición: Grado = 8
    - Si sólo cumple la tercera condición: Grado = 9
    - Si cumple la primera y la segunda: Grado = 12
    - Si cumple la primera y la tercera: Grado = 13
    - Si cumple la segunda y la tercera: Grado = 15
    - Si cumple las tres condiciones: Grado = 20
- Por cada trabajador se tiene un registro con los siguientes datos:
    - Código del trabajador.
    - Horas de ausencia.
    - Pupitres defectuosos.
    - Pupitres producidos.

Elaborar un algoritmo que calcule para cada trabajador su correspondiente grado de eficiencia e imprima toda su información. Se debe utilizar subprocesos

Se requiere analizar (Entrada, Proceso y Salida), crear el algoritmo y representarlo en PseInt. Se debe proporcionar una captura de pantalla del pseudocódigo en PseInt con su respectiva ejecución.

Ejercicio 5:

En una universidad se tiene un grupo de N estudiantes, cada uno de ellos toma 8 materias. Semestralmente el profesor de cada materia reporta a secretaría general el código del estudiante, el número de la materia y la calificación final. Elaborar un algoritmo que obtenga:

- El promedio de las calificaciones obtenidas por cada estudiante.
- El promedio de la calificación por cada materia.
- El estudiante con mayor promedio.
- El número de estudiantes aprobados (nota final sea mayor o igual a 7) por cada materia.
- El número de estudiantes reprobados (nota final sea menor a 7) por cada materia.

Se debe utilizar subprocesos y funciones.

Se requiere analizar (Entrada, Proceso y Salida), crear el algoritmo y representarlo en PseInt. Se debe proporcionar una captura de pantalla del pseudocódigo en PseInt con su respectiva ejecución.

Actividad 6:

Cree un videojuego utilizando Scratch en donde:

- Ponga en práctica todo lo aprendido en la asignatura.
- Proporcione el link de acceso para verificar el correcto funcionamiento.
- Explique cómo lo realizó (en vivo).

6. LO MÁS RECOMENDADO

Lógica de programación por Carlos Salazar (IST CEMLAD).

La esencia de la lógica de programación por Omar Iván Trejos.

Introducción a la lógica de programación por Jorge Herrera, Julián Gutiérrez, Robinson Pulgarín

NO DEJES DE VER

Recursos de aprendizaje de la asignatura (videos, lecciones, libros).

ENTREGA

Se debe subir el laboratorio resuelto en formato PDF a la plataforma virtual.

Bibliografía

Buriticá, O. I. (1999). *La esencia de la Lógica de Programación.* Pereira.

Corredor, E. R. (s.f.). *Aprender a programar.*

Diccionario de la Lengua Española. (14 de 07 de 2019). *Real Academia Española*. Obtenido de https://dle.rae.es/?w=L%C3%B3gica

EDUTEC. (2013). *Categorización a partir de la taxonomía de Bloom (1956). Diseño de una pauta para clasificar actividades incluidas en cursos de contenido TIC.* Costa Rica.

García Cano, Edgar E. I. J. (s.f.). *Guía práctica de estudio 04: Pseudocódigo.*

Hernández Yánez, Luis. (2014). *Fundamentos de la programación*.

Jesús Valverde Berrocoso, F. I. (2012). *MODELOS DE EVALUACIÓN POR COMPETENCIAS A TRAVÉS DE UN SISTEMA DE GESTIÓN DE APRENDIZAJE. EXPERIENCIAS EN LA FORMACIÓN INICIAL DEL PROFESORADO.* España: Revisata Iberoamericana de educación.

Jiménez, F., Martínez G, Sánchez G, Juárez J, & Paredes S. (2009). Un sistema de evaluación basado en competencias: Ejemplo para la asignatura Tecnológica de la Programación del título de Grado en Ingeniería Informática por la Universidad de Murcia. *XV JENUI*, 194.

Parra, P. C. (2020). *Modelo Educativo CEMLAD.* Quito.

Regino, E. M. (s.f.). *Lógica de programación.*

Reyes Corredor, Edward. (2008). *Aprender a programar*.

SERVICIO NACIONAL DE APRENDIZAJE (SENA). (2003). *Manual de estrategias de enseñanza/aprendizaje.* Medellín: Pregón Ltda.

Tecnológico de Monterrey. (s.f.). *La evaluación por competencias.*

Universitat de Barcelona. (2013). Rúbricas para la evaluación de competencias. *Cuaderno de Docencia Universitaria*, 26.

Universitat Politécnica de Valéncia. (2015). Rúbrica para la Evaluación de la Competencia Innovación, Creatividad y Emprendimiento en máster. *IN-RED*.

Universidad de Matanzas Camilo. (2013). *Gutiérrez Chiñas Agustín*. Matanzas - Cuba: Atenas.

WIKIPEDIA. (08 de 06 de 2019). *WIKIPEDIA*. Recuperado el 14 de 07 de 2019, de https://es.wikipedia.org/wiki/L%C3%B3gica

Zapata Ospina, Carlos A. (2006). *Fundamentos de programación, guía de enseñanza*.