Desarrollo de Algoritmos y Diagramas de Flujo para la Programación de Computadoras

Guía de trabajo didáctico

Desarrollo de Algoritmos y Diagramas de Flujo para la Programación de Computadoras. Guía de trabajo didáctico / Vicente Villanueva Hernández , José Rafael Baca Pumarejo .—Cd. Victoria, Tamaulipas : Universidad Autónoma de Tamaulipas ; Ciudad de México : Colofón , 2020.
113 págs. ; 17 x 23 cm.

1. Programación (computadoras) – Enseñanza 2. Algoritmos 3. Diagramas de flujo
LC: **QA76.6** DEWEY: **005.1**

Centro Universitario Victoria
Centro de Gestión del Conocimiento. Tercer Piso
Cd. Victoria, Tamaulipas, México. C.P. 87149
consejopublicacionesuat@outlook.com

Matamoros SN, Zona Centro Ciudad Victoria, Tamaulipas C.P. 87000
Consejo de Publicaciones UAT
Tel. (52) 834 3181-800 • extensión: 2948 • www.uat.edu.mx

Fomento Editorial Una edición del Departamento de Fomento Editorial de la Universidad Autónoma de Tamaulipas

Edificio Administrativo, planta baja, CU Victoria
Ciudad Victoria, Tamaulipas, México
Libro aprobado por el Consejo de Publicaciones UAT
ISBN UAT: 978-607-8750-07-8

Colofón S.A. de C.V.
Franz Hals núm. 130, Alfonso XIII
Delegación Álvaro Obregón C.P. 01460, Ciudad de México
www.colofonlibros.com • colofonedicionesacademicas@gmail.com
ISBN: 978-607-635-143-7

Publicación financiada con recurso PFCE 2019

Impreso en México • Printed in Mexico El tiraje consta de 400 ejemplares

Este libro fue dictaminado y aprobado por el Consejo de Publicaciones UAT mediante un especialista en la materia. Asimismo fue recibido por el Comité Interno de Selección de Obras de Colofón Ediciones Académicas para su valoración en la sesión del primer semestre 2020, se sometió al sistema de dictaminación a "doble ciego" por especialistas en la materia, el resultado de ambos dictámenes fue positivo.

"PARA **CREAR** COSAS BUENAS
PRIMERO HAY QUE **CREER**
EN ELLAS"

70 AÑOS

UNIVERSIDAD
AUTÓNOMA DE
TAMAULIPAS
1950-2020

Desarrollo de Algoritmos y Diagramas de Flujo para la Programación de Computadoras

Guía de trabajo didáctico

Vicente Villanueva Hernández
José Rafael Baca Pumarejo

VERDAD, BELLEZA, PROBIDAD

UAT

Consejo de Publicaciones

Fomento Editorial

COLOFÓN

ÍNDICE

Presentación

Esta guía algorítmica fue elaborada con la intención de que los alumnos que desean emprender el camino de convertirse en programadores de Tecnologías de Información o Sistemas de Información, logren aprender y dominar algunas técnicas sencillas para resolver problemas de la vida diaria usando algoritmos, y aplicarlos mediante su codificación en lenguaje computacional.

El propósito es que sea utilizada en los primeros semestres o cuatrimestres por los alumnos que inician su preparación y no han tenido experiencia previa en esta área del conocimiento.

El modelo se basa en el procedimiento para la solución de problemas cotidianos. Describe la estructura del algoritmo desde cómo se declaran las variables hasta cómo se implementa el proceso una vez que se redacta el código de programación.

La guía se compone de dos grandes temas: el algoritmo y el diagrama de flujo, elementos esenciales para cultivar las destrezas básicas de un futuro programador que desarrolle aplicaciones computacionales. Asimismo, ésta incluye ejercicios para realizar en clase, en casa o en el laboratorio de informática, actividad indispensable para reforzar el conocimiento mediante el uso práctico de estos conceptos, lo que habrá de repercutir en la profundización y consolidación de los conocimientos, habilidades y destrezas en la actividad de programación.

Introducción

Esta guía inicia con el desarrollo de algoritmos aplicables a cualquier lenguaje de programación. Está planeada para empezar el proceso de enseñanza-aprendizaje en el área de tecnologías de la información. La primera parte de cada tema contiene lo que debe saber un alumno sobre el análisis y desarrollo, antes de llegar a la etapa de implementación.

Sirve para manejar variables de diferentes tipos y que el alumno se familiarice con el uso de los contadores y los acumuladores, que son los elementos básicos de la computación; además, le va a permitir desarrollar conceptos de selección, repetición y conjuntos de datos en el contexto de los problemas que habrá de resolver usando como herramienta la computadora.

Sigue de manera simbólica el modelo de máquina computacional de Von Newman, el prototipo que desde su invención hasta nuestros días se usa en las máquinas computacionales para procesar la información, a través de las unidades de entrada, salida y proceso, donde se reciben del exterior o del interior de la computadora los datos, se efectúan cálculos matemáticos sobre ellos, se almacenan y se generan los resultados mediante un trabajo controlado y siempre bajo la administración de un sistema operativo.

Se encontrarán ejercicios básicos, muy útiles para que el alumno desarrolle su capacidad intelectual, creatividad y logre elaborar su propio modelo conceptual de solución de problemas mediante algoritmos, códigos y ejecuciones de los mismos, para generar resultados.

Reglas del curso

- Respetar la agenda (días y horario) de las sesiones.
- Utilizar el espacio de clases para los fines establecidos.
- Se tendrá una tolerancia máxima de 10 minutos después de la hora de entrada.
- Favorecer la comunicación efectiva y asertiva en el desarrollo de la clase, así como de las actividades que de ella deriven.
- Promover la participación colaborativa.
- Excluir el uso del celular.
- Contribuir con una imagen propia de la profesión.
- Fomentar la ética y los valores.
- Practicar el respeto hacia el catedrático y compañeros en un marco incluyente.
- Propiciar un clima académico favorable para el proceso de enseñanza-aprendizaje.

Desarrollo de la práctica

Equipo

- Computadora con acceso a internet, proyector.

Materiales

- Cuaderno, lápiz, plumas.
- Bibliografía recomendada para consulta:
- Joyanes, L. (2008). *Fundamentos de Programación.* México: Ed. Mc Graw Hill.
- López, G. (2009). *Análisis y Diseño de Algoritmos.* México: Ed. Alfa Omega.

Procedimiento

- Leer y analizar el texto las veces que sean necesarias.
- Identificar los objetivos del texto.
- Localizar los conceptos más importantes.
- Tomar nota de las ideas principales y secundarias.
- Redactar y sintetizar las ideas fundamentales mediante tus propias palabras, en armonía con la estructura del texto.
- Cuidar la ortografía, redacción y limpieza.

Sistema de evaluación

Criterios	**Ponderación**
Identificación de los objetivos del texto.	20%
Anotación de los títulos y subtítulos más importantes, con una breve explicación.	20%
Descripción de los conceptos más importantes previo a las actividades de la práctica. Cuidado de la ortografía y la redacción.	30%
	15%
Elaboración de los contenidos.	15%

Objetivos de la guía

- Que el alumno desarrolle sus habilidades para resolver problemas mediante algoritmos y el apoyo de la computadora.
- Que el alumno sea autónomo, creativo y desarrolle sus habilidades como programador.
- Que el profesor cuente con un instrumento que lo apoye en su labor didáctica.
- Que coadyuve a que el alumno principiante logre diseñar, escribir, desarrollar e implementar programas con la computadora.
- Que el alumno conozca y domine las reglas básicas de la programación.

Tema I
Algoritmos

Evaluación diagnóstica

Para iniciarse en la programación, el alumno requiere tener conocimientos previos en matemáticas, aritmética básica, lógica, álgebra, logística, lectura, redacción y disciplinas relacionadas. Además de aptitud y actitud para resolver problemas, elaborar mapas conceptuales, niveles aceptables de creatividad, innovación, agilidad mental, comprensión y trabajo en equipo.

Objetivos del tema

- Identificar las fases que intervienen en el proceso de programación y aplicar las estructuras secuenciales que conforman un algoritmo.
- Diseñar e implementar soluciones a problemas del entorno mediante el diseño de algoritmos.
- Desarrollar las habilidades para aplicar la metodología para resolver problemas del entorno mediante algoritmos.

Competencias básicas a adquirir

- Eficacia en la solución de problemas
- Trabajo en equipo y liderazgo
- Comunicación
- Organización

Elementos de la competencia

- Diseñar e implementar soluciones utilizando las principales técnicas de algoritmos computacionales para identificar la problemática y dar soluciones eficaces.

Criterios de desempeño

- Diseño de la solución, creatividad, presentación, entrega puntual de los ejercicios.
- Eficiencia.

¿Qué es un algoritmo?

El desarrollo de algoritmos es un tema fundamental en la etapa de análisis, diseño y desarrollo de programas o soluciones para la automatización y optimización de los procesos en las organizaciones. Para ello, el alumno debe tener suficientes habilidades para crear de manera fácil y rápida los programas que le permitan generar las soluciones.

Esta obra buscar apoyar el trabajo de tutores y/o profesores en su labor de enseñanza-aprendizaje y facilitar al estudiante el desarrollo de su capacidad analítica y creadora, para mejorar su destreza en el diseño de algoritmos básicos para los diferentes lenguajes de programación que enfrentará durante sus estudios relacionados con las tecnologías de la información.

En la vida diaria todos los problemas y situaciones se rigen por algoritmos, vueltos hábitos y automatizados conforme pasa el tiempo. En nuestra mente se generan ciertas rutinas que, por lógica, aplicamos en algún problema cotidiano. Un ejemplo de algoritmos son los procedimientos que a diario ejecutamos para llegar a la escuela y al salón de clases a tiempo.

Cada mañana nos levantamos, nos bañamos o no, buscamos la ropa, quizá desayunamos, quizá nos cepillamos los dientes, buscamos y tomamos la mochila con los útiles escolares, sigue despedirnos de nuestros familiares y salir rumbo a la escuela. En lo que llegamos podemos iniciar actividades muy diversas, por ejemplo, escuchar música, contestar el celular. Si viajamos en transporte público, habrá que estar atentos a solicitar la parada, bajar y seguir la ruta. En el tramo de la calle al plantel hay probabilidades de encontrar a un amigo, saludarlo y entablar una plática sobre algún tema de las clases, política, el clima, etc. Al llegar, iremos al aula, algunos optimistas o rezagados preguntarán si ya llegó el maestro, si es así, se solicitará permiso para entrar, si no, nos dirigimos a nuestro lugar para cumplir el objetivo.

Otro ejemplo sería diseñar un algoritmo que permita obtener un refresco de una máquina automática expendedora de bebidas embotelladas (Vázquez, 2012).

1. Inicio
2. Localizar la bebida deseada en el panel de la máquina.
3. Identificar su costo.
4. Pulsar el botón que seleccione la bebida.
5. Si existe el producto, introducir en la ranura apropiada la cantidad monetaria que corresponda al precio de la bebida, si no, en el panel se visualizará 'producto agotado' y se finaliza la compra.
6. Si la cantidad introducida es mayor al precio del producto nos devolverá el cambio.

7. El producto descenderá a la bandeja.
8. Fin.

Un ejemplo más sería, dada una lista de números aleatorios, buscar un número determinado.

- La serie de números es 5, 7, 8, 12, 16, 22, 33, 45.
- De los números anteriores, buscar el número 16.
- El primer paso sería aplicar la técnica *Divide y vencerás.*
- Eliminar de la serie los mayores a 16: 22, 33, 45.
- El segundo paso sería eliminar los menores: 5, 7, 8, 12.
- El tercer paso sería encontrar el número 16.

Como puede observarse, son algoritmos que por lógica podemos plantear de manera inconsciente. Todos nuestros procesos mentales son algoritmos. Algunos requieren relacionarlos con otros, crear redes y otros se ejecutan solos. El cerebro es un sistema de información relacionado y entrelazado por medio de nuestras neuronas para cumplir con un objetivo y, dependiendo de cada necesidad, desarrolla sus propias rutinas o algoritmos. En general, no existe un único algoritmo para cada problema que se quiera resolver. Diferentes algoritmos pueden completar la misma tarea, requiriendo cada uno de diferentes cantidades de tiempo, espacio o esfuerzo. Sin embargo, la especificación puede ser exactamente la misma para todos ellos.

Definición de algoritmo

El matemático y astrónomo persa Musa Al-Juarismi (780-850) inventó el algoritmo, es decir, la resolución metódica de problemas de álgebra y cálculo numérico mediante una lista bien definida, ordenada y finita de operaciones (Noguera y Riera, 2010). *Algoritmi* fue la latinización del nombre *Al-Juarismi.* Sus recetas terminaron llamándose *algoritmos* y su aplicación implicó la revolución de la informática.

Según los autores Forsythe et al., (1969):

> […] un algoritmo es una lista de instrucciones para efectuar paso por paso algún proceso. Los algoritmos ejecutados por una computadora pueden combinar en un cálculo matemático complicado millones de pasos elementales, tales como adiciones o sustracciones. En teoría, la programación en computación es sencilla. Sin embargo, hay muchas alternativas posibles, dependiendo de la problemática.

Kosinski (2017) afirma: "los algoritmos computacionales son capaces de tomar información de las huellas digitales que dejamos de registro y detectar rasgos individuales que ningún análisis humano sería capaz de revelar explícitamente" (p. 6).

Una computadora puede también, mediante el uso de algoritmos, controlar procesos de cualquier empresa. Los algoritmos para procesos de gran escala son muy complicados, pero están construidos a base de piezas, es decir, un conjunto de operaciones que se utilizan para resolver un problema específico. Para tal efecto, la serie de instrucciones indican la secuencia de las operaciones que se deben realizar para obtener el resultado buscado a partir de los datos de entrada (Kosinski, 2017).

Deitel y Deitel (2016) definen al algoritmo como "un procedimiento para resolver un problema en términos de las acciones a ejecutar y el orden en el que se ejecutan estas acciones".

El concepto de algoritmo es anterior a los ordenadores y ampliamente utilizado en el mundo de la ciencia para la resolución metódica de problemas. Sin embargo, con la aparición de los ordenadores se comprobó que es una herramienta ideal, ya que cualquier algoritmo puede transformarse en un programa informático (Deitel y Deitel, 2016).

Mientras que para Restrepo y Marín (2011) "un algoritmo resuelve el problema planteado para la estructura de datos de entrada definida y su tiempo de ejecución es finito".

Si se desconoce cómo resolver el problema, difícilmente se podrá solucionarlo. Por eso, es necesario conocer el tipo de problema (Restrepo y Marín, 2011).

Según Miranda y Fuenlabrada (2015), los problemas están clasificados en:

> […] Estructurados: se puede establecer una solución segura y precisa, plantear situaciones claras, bien definidas, delimitadas. En problemas no estructurados su planteamiento es impreciso e incompleto, no son claros los elementos, lo que provoca ambigüedad y confusión en la solución requerida, y los problemas semiestructurados contienen parte de la información que es clara y precisa, pero en el resto hay incógnitas y dudas.

Un algoritmo es un conjunto pre-escrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas en quien lo ejecute. Dados un estado inicial y una entrada, siguiendo los pasos sucesivos se llega a un estado final y se obtiene una solución (Miranda y Fuenlabrada, 2015).

Para Guzdial y Ericson (2013) "un algoritmo es la descripción de un proceso paso a paso que no está enlazado a ningún lenguaje de programación y que el mismo algoritmo puede implementarse en distintos lenguajes con el mismo proceso".

Un algoritmo es una secuencia lógica, ordenada y con un fin determinado, para lograr un objetivo común (Guzdial y Ericson, 2013).

En las actividades cotidianas la gran mayoría de la gente utiliza algoritmos para resolver problemas de la vida diaria; por ejemplo, levantarse para hacer una actividad habitual, parchar una llanta, comprar mandado, realizar algún cálculo de gasto de despensa, etcétera. El proceso pensante de cada individuo es un algoritmo. Sin embargo, puede ser cualquier actividad que funcione paso a paso sin necesidad de describir ni hacer referencia a una computadora (Guzdial y Ericson, 2013).

Para Zúñiga et al., (2017), el "algoritmo consiste en un conjunto de instrucciones claras y precisas que se identifican y se planifican en un determinado orden para la resolución de problemas".

> Existen diferentes métodos o modelos de expresar un problema o un algoritmo, que incluye el pseudocódigo, diagramas de flujo y lenguajes de programación. De hecho, en ese orden, un programador deberá seguir esta secuencia para elaborar un sistema óptimo, sin errores. Los métodos algorítmicos se pueden implementar en cualquier computadora, incluidos los que utilizan la heurística. En este sentido, la inteligencia artificial es un campo donde esta técnica es usada en las computadoras (Zúñiga et al., 2017).

Escolano et al., (2003) definen un método llamado Heurístico, "[…] un procedimiento para resolver un problema de optimización bien definido mediante una aproximación intuitiva en la que la estructura del problema se utiliza de forma inteligente, para obtener una buena solución".

Tipos de algoritmos

En este análisis, algunos autores mencionan que en el proceso de programación una **variable** es una entidad compuesta de un conjunto de caracteres no vacío y finito, que pueden ser alfabéticos, numéricos o alfanuméricos y representa un espacio de memoria en el cual se puede almacenar un dato.

Juganaru (2014) establece que:

> El formato de representación y de estructuración de los datos depende del paradigma del lenguaje de programación y de la opción que el programador ha elegido para representar los datos.

> En el paradigma imperativo y en el caso de algunos otros paradigmas (por ejemplo, lenguaje PROLOG) existe una noción básica común para el manejo de los datos: la noción de variable.
> (…) La ventaja de las variables es que almacenan datos de entrada, de salida o intermedios. Existen lenguajes de tipo SQL o XPath que implementan la noción de variable de manera especial.

Los administradores de bases de datos manejan la información mediante registros y éstos se componen de campos que deben tener una identidad con un tipo declarado de forma puntual, como los ejemplos que ilustran la siguiente tabla (IntelDig, 2018).

Tabla de los tipos de datos alfabéticos que el Lenguaje de Consulta Estructurada, SQL (Structured Query Language), maneja y opera en la administración de una base de datos.

Tipo de datos	**Descripción**
CHAR (tamaño)	Tiene una cadena de longitud fija. (Puede contener letras, números y caracteres especiales). El tamaño fijo se especifica entre paréntesis. Puede almacenar hasta 255 caracteres.
VARCHAR (tamaño)	Tiene una cadena de longitud variable. (Puede contener letras, números y caracteres especiales). El tamaño máximo se especifica entre paréntesis. Puede almacenar hasta 255 caracteres. Nota: Si agrega un valor mayor a 255, se convertirá en un tipo de texto.
TINYTEXT	Tiene una cadena con una longitud máxima de 255 caracteres.
TEXTO	Tiene una cadena con una longitud máxima de 65 535 caracteres.
BLOB	Para BLOB (Objetos grandes binarios). Almacena hasta 65 535 bytes de datos.
MEDIUMTEXT	Tiene una cadena con una longitud máxima de 16 777 215 caracteres.
MEDIUMBLOB	Para BLOB (Objetos grandes binarios). Tiene capacidad para 16 777 215 bytes de datos.
LONGTEXT	Tiene una cadena con una longitud máxima de 4 294 967 295 caracteres.
LONGBLOB	Para BLOB (Objetos grandes binarios). Tiene capacidad para 4 294 967 295 bytes de datos.

Tipo de datos	Descripción
ENUM (x, y, z, etc.)	Permite ingresar una lista de valores posibles. Puede enumerar hasta 65 535 valores en una lista ENUM. Si se inserta un valor que no está en la lista, se insertará un valor en blanco. Nota: los valores se acomodan en el orden en que los ingresas. Ingrese los valores posibles en este formato: ENUM ('X', 'Y', 'Z').
SET	Similar a ENUM, excepto que SET puede contener hasta 64 elementos de lista y puede almacenar más de una opción.

Tabla de los tipos de datos numéricos que SQL (Structured Query Language) maneja y opera en la administración de una Base de Datos

Tipo de datos	Descripción
TINYINT (tamaño)	De -128 a 127 normal. 0 a 255 SIN FIRMAR *. La cantidad máxima de dígitos se puede especificar entre paréntesis.
SMALLINT (tamaño)	De -32 768 a 32 767 normal. 0 a 65 535 SIN FIRMAR *. La cantidad máxima de dígitos se puede especificar entre paréntesis.
MEDIUMINT (tamaño)	De -8 388 608 a 8 388 607 normal. 0 a 16 777 215 SIN FIRMAR *. La cantidad máxima de dígitos se puede especificar entre paréntesis.
INT (tamaño)	De -2 147 483 648 a 2 147 483 647 normal. 0 a 4 294 967 295 SIN FIRMAR *. La cantidad máxima de dígitos se puede especificar entre paréntesis.
BIGINT (tamaño)	De -9 223 372 036 854 775 808 a 9 223 372 036 854 775 807 normal. 0 a 18 446 744 073 709 551 615 SIN FIRMAR *. La cantidad máxima de dígitos se puede especificar entre paréntesis.
FLOA (tamaño, d)	Un pequeño número con un punto decimal flotante. La cantidad máxima de dígitos se puede especificar en el parámetro de tamaño.
DOBLE (tamaño, d)	El número máximo de dígitos a la derecha del punto decimal se especifica en el parámetro d. Un número grande con un punto decimal flotante. La cantidad máxima de dígitos se puede especificar en el parámetro de tamaño. El número máximo de dígitos a la derecha del punto decimal se especifica en el parámetro d.

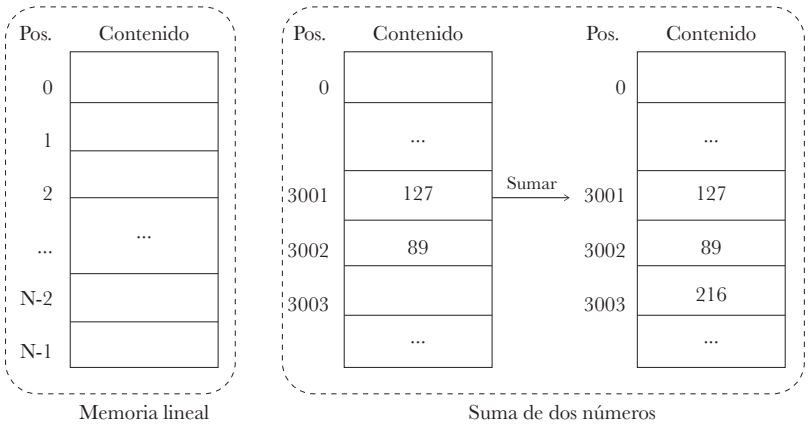
DECIMAL (tamaño, d)	Un DOBLE almacenado como una cadena, lo que permite un punto decimal fijo. La cantidad máxima de dígitos se puede especificar en el parámetro de tamaño. El número máximo de dígitos a la derecha del punto decimal se especifica en el parámetro d.

Como se observa en las dos tablas anteriores, los tipos de datos que se emplean son los campos que se declaran para el manejo de las bases de datos que se conforman a través de la aplicación SQL.

Y se concluye en este análisis que, en general:

> […] en cada programa aparece al menos una variable, lo que significa que en cada programa hay una zona de memoria con un tamaño fijo que contiene un valor de tipo preciso; por ejemplo, un entero representado en forma binaria de Ca2[3] sobre 4 bytes, o una cadena de caracteres de un tamaño máximo de 255.
>
> Cada variable debe tener:
>
> Un tamaño de memoria ocupada y un modo de representación interna. Por ejemplo, un punto flotante simple o doble precisión sobre 4 bytes o cadenas de caracteres de 100 + 1 caracteres.
>
> Un conjunto de operadores y de tratamientos específicos que pueden aplicarse a la variable. Si las variables son, por ejemplo, de tipo lógico, se aplican operadores lógicos; pero si las variables son numéricas, se aplican operadores de cálculo numérico (suma, producto, entre otros) (Juganaru, 2014).

Una variable no es más que un nombre simbólico que identifica una dirección de memoria:



Sea la siguiente operación con variables total=cantidad1+cantidad2, donde cantidad1 es la variable que se almacena en la dirección de memoria 3001 y cantidad2 la variables que se almacena en 3002. Al sumarse este total, se almacena en la dirección de memoria 3003.

El programador, cuando desarrolla una aplicación (o diseña un algoritmo), debe decidir:

- Cuántas son las variables que el programa necesita para realizar las tareas que le han encomendado.
- El tipo de dato que puede almacenar cada una de ellas.

Durante la ejecución de un programa, el valor que tome el dato almacenado en una variable puede cambiar tantas veces como sea necesario, pero siempre tomando valores pertenecientes al tipo de dato que el programador ha decidido que puede almacenar dicha variable, ya que no puede ser cambiado durante la ejecución de un programa.

Para ejemplificar los algoritmos, se manejan los siguientes tipos:

- Cualitativos: Son aquéllos en los que se describen los pasos utilizando palabras. (Son más que nada enunciados verbales de lo que se lleva a cabo en cada paso).
- Cuantitativos: Son aquéllos en los que se utilizan cálculos numéricos para definir los pasos del proceso (Juganaru, 2014).

En la opinión de Oviedo (2015):

> Los algoritmos cualitativos, también llamados procedimientos, son un conjunto de instrucciones o pasos descritos mediante palabras para llegar a la solución o resultado(s) de un problema que no involucra cálculos matemáticos, sino procesos. La descripción de cada paso debe conservar un orden riguroso y la comprensión de éste depende en forma exclusiva de la persona que lo construye. Posiblemente, una persona divida un paso en varios o condense dos o más pasos en uno solo.

Ejemplos de este tipo de algoritmos son los que refieren la serie de pasos que implica ir de nuestra casa a la escuela o para la venta de un refresco en una máquina automática.

Oviedo (2015) establece que los algoritmos cuantitativos son "secuencias finitas y ordenadas de pasos que dan solución a un problema que involucra cálculos matemáticos". De acuerdo a esto, en lo sucesivo, cuando hacemos alusión al término algoritmo, nos referiremos a los algoritmos cuantitativos compuestos por tres partes: Entrada→Proceso→Salida.

Oviedo (2015) precisa:

> La entrada es la información dada al algoritmo por el usuario o la información generada por un conjunto de instrucciones. El proceso son los cálculos necesarios que hay que hacerles a los datos de entrada para que a partir de ellos se pueda llegar a los resultados o datos de salida. La salida son los resultados finales o la información que nos pide el problema como resultados, los cuales obtenemos a partir de las transformaciones que se les han hecho a los datos de entrada.

También, afirma que las características fundamentales que debe cumplir todo algoritmo son:

> *Datos de entrada.* Se refiere a la información proporcionada al algoritmo por el usuario, la cual debe seguir 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. Son datos de entrada aquellos valores que vamos a almacenar en variables, o sea, los valores constantes no son datos de entrada.
>
> *Datos de salida.* Todo algoritmo debe proporcionar uno o más valores como resultado una vez que se ha ejecutado la secuencia de pasos que lo conforman. La salida es la respuesta dada por el algoritmo o el conjunto de valores que el programador espera se le proporcionen. Estos resultados pueden ser de cualquier tipo: uno o más valores numéricos, valores lógicos o caracteres. La facilidad o complejidad de un algoritmo no la determina la cantidad de datos que se desean obtener. Un algoritmo puede tener un alto grado de complejidad y, sin embargo, producir un solo valor como resultado.
>
> *Limitado o finito.* Todo algoritmo debe tener un número de instrucciones que limitan el proceso en algún momento, es decir, la ejecución debe detenerse. No puede existir un algoritmo, por muy grande que sea o por muchos resultados que produzca, que se quede en forma indefinida ejecutando sus instrucciones o repitiendo la ejecución de un subconjunto de ellas. Existen casos excepcionales, como los servidores, que trabajan continuamente debido a que son controlados por rutinas de ejecución constante (en inglés son conocidas como *routines forever*).
>
> *Finalización*. Un algoritmo debe indicar el orden de realización de cada uno de sus pasos. Debe mostrar la primera, la intermedia y la última instrucción que debe realizarse. Esto permite mostrar que en algún momento debe culminar la acción o tarea que hace el algoritmo.

> *Claridad.* Todo el conjunto de pasos debe ser entendible y factible de realizar, de tal manera que al hacer un seguimiento del algoritmo éste produzca siempre los resultados requeridos. No puede entonces existir incertidumbre en las acciones por tomar cuando se sigue la lógica (flujo del programa) del algoritmo (Oviedo, 2015).

Variables

Las variables son entidades para representar valores y éstos son los contenidos que siempre sufrirán cambio con respecto al tiempo.

En los algoritmos se definen nada más algunos tipos de variables para empezar, lo que no significa que solo haya éstas, existe más variedad, pero por lo pronto se empezará con las siguientes: variables numéricas (0-9), variables alfanuméricas (a-z, 0-9) y variables alfabéticas (a-z), pero con éstas dos últimas no se pueden realizar operaciones aritméticas.

Según Juganaru (2014):

> […] el contador es una variable en la memoria que se incrementará en una unidad cada vez que se ejecute el proceso. El contador se utiliza para llevar la cuenta de determinadas acciones que se pueden solicitar durante la resolución de un problema.
>
> […] un acumulador es una variable en la memoria cuya misión es almacenar cantidades variables. Además, se utiliza para efectuar sumas sucesivas. La principal diferencia con el contador es que el incremento o decremento de cada suma es variable en lugar de constante, como en el caso del contador.

Las variables anteriores se pueden utilizar para definir contadores y acumuladores en todos los procesos, para guardar datos, hacer operaciones aritméticas, asignar valores, definir su tipo; también se puede condicionar una variable. Son zonas de memoria cuyo contenido cambia durante la fase de procesamiento de información (Juganaru, 2014).

Tipos de variables:

- Variables Numéricas (Enteras y Reales)
- Variables Alfanuméricas
  a) Caracteres alfabéticos
  b) Dígitos
  c) Caracteres especiales
- Variables Lógicas (Booleanas)

Sin embargo, para efecto de esta guía, la variable numérica debe llevar una letra al principio, seguida de una letra y un número y el guion como único carácter válido.

Las variables alfanuméricas se pueden representar con una letra y un signo de pesos, para diferenciarlas de las numéricas.

Las variables alfabéticas se identifican con una letra seguida de un símbolo de porcentaje. Estas variables pueden ser una sola o un conjunto de éstas, sin espacio en blanco y el guion como único carácter obligatorio.

Ejemplos de representación de estas variables:

Variables numéricas	A1, a-5, b4-c
Variables alfanuméricas	A$, b$, c$-a
Variables alfabéticas	A%, b%-6, c%-a

Para Durán et al., (2007), un identificador es el "nombre que asigna a los diferentes elementos de un programa, como puede ser una variable, un nombre de función, formando una secuencia de caracteres que pueden incluir números (0-9) y caracteres (a-z) y el guion como único caracter válido".

Lo anterior es para ir acostumbrando al programador que se inicia a conocer que hay diferentes tipos de variables para diferentes procesos en cualquier lenguaje de programación (Durán et al., 2007).

Metodología para elaborar un algoritmo

Para Juganaru (2014), los componentes de un algoritmo son:

- Inicio: Comienza la solución del problema.
- Entrada: Se conocen las variables y constantes que van a ser utilizadas en la solución del problema.
- Proceso: Realiza las operaciones necesarias con o para las variables y constantes para dar solución al problema.
- Salida: Impresión de los datos arrojados en el proceso.

Para realizar un algoritmo se llevan a cabo ciertas etapas: se inicia desarrollando el algoritmo de arriba hacia abajo y de izquierda a derecha. Se debe tener un inicio y un fin. Entre esos dos puntos de control se desarrolla el proceso, que consiste en la entrada de datos, su procesamiento y la salida, que coincide con el modelo de máquina computacional de Von Newman (Entrada de datos, Proceso y Salida). Éstas son las etapas para llevar a cabo la solución del problema (Juganaru, 2014).

Según Deitel y Deitel (2015):

> Las computadoras procesan bajo el control de un conjunto de instrucciones conocidas como el programa de computadora. Éstas guían a la máquina computacional a través del conjunto de acciones específicas planeadas por la gente conocida como programadores (es decir, los Usuarios de la Máquina Computacional).

La computadora es un instrumento que por sí solo no puede hacer nada, necesita ser programado, es decir, introducirle órdenes que le digan lo que tiene que hacer (Deitel y Deitel, 2015).

El algoritmo es la primera etapa para la solución de un problema, el proceso de programación es el siguiente: Dado un determinado problema, el programador debe idear una solución y expresarla usando un algoritmo (aquí es donde el alumno empieza a desarrollar ciertas capacidades de análisis creativas y productivas). Luego, debe codificarlo en un determinado lenguaje de programación (Juganaru, 2014).

Pasos para desarrollar un algoritmo

Oviedo (2015) define un algoritmo como "un método que se realiza paso a paso para solucionar un problema, que termina en un número finito de pasos".

Las características fundamentales que debe cumplir todo algoritmo son:

- Debe ser preciso. Indicar el orden de realización de cada paso.
- Debe ser definido. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez.
- Debe ser finito. Si se sigue un algoritmo, se debe terminar en algún momento; es decir, debe tener un número finito de pasos.

En esta línea, Flores (2017) considera que:

Los algoritmos son pilares en la informática, esenciales en el desarrollo de sistemas, por lo tanto representan el tratamiento sistemático de la información, también conocida como procesamiento de datos. La estructura de un algoritmo debe incluir tres partes: Entrada, Proceso y Salida.

1. Definir el problema

Esta fase está dada por el enunciado del problema. Es indispensable captar la esencia del mismo, por lo que se requiere estudiarlo cuantas veces sea necesario para proceder a buscar la manera en que la computadora lo desarrolle.

2. Analizar el problema

Una vez que se ha precisado lo que se desea que haga la computadora, es necesario determinar qué datos se requieren, es decir, los llamados datos de entrada. Luego,

la segunda etapa, llamada proceso, serán las fórmulas para obtener los resultados a partir de esos datos. Finalmente, la salida, lo que constituye la solución, a través de algún dispositivo de salida.
3. Diseñar el algoritmo

Las características de un buen algoritmo, según Oviedo (2015) son:

- Debe tener un punto particular de inicio.
- Debe ser definido, no debe permitir dobles interpretaciones.
- Debe ser general, es decir, soportar la mayoría de las variantes que se puedan presentar en la definición del problema.
- Debe ser finito en tamaño y tiempo de ejecución.
- Nunca debe dejar de considerar alguna opción, aunque esta no sea probable que suceda.

En general, la parte común en todas las definiciones se puede resumir en los siguientes puntos: Tiene que ser finito (con un final), preciso (detallar el orden de las operaciones a realizar) y unívoco (al aplicar el algoritmo a los mismos datos de entrada, siempre se obtendrá el mismo resultado a la salida).

Una vez que la solución de un problema ha sido expresada mediante un algoritmo, el paso siguiente es convertirlo en programa, para lo cual se elige un lenguaje de programación. De modo que un programa resulta ser la implementación de un algoritmo en un determinado lenguaje de programación. Esto significa, por otro lado, que un algoritmo es independiente del lenguaje de programación (Oviedo, 2015).

Ejemplos resueltos de algoritmos del Tema 1

Algoritmos *top down* (de arriba hacia abajo)

I. Elaborar un algoritmo que permita convertir grados Fahrenheit a Celsius (centígrados) y viceversa.

1. Inicio
2. "Dame grados Celsius" c
3. "Dame grados Fahrenheit" f
4. c1=1.8 * c + 32
5. f1= f – 32 /1.8
6. Imprimir "la conversión de Fahrenheit a Celsius es c1"
7. Imprimir "la conversión de Celsius a Fahrenheit es f1"
8. Fin

II. Diseñar un algoritmo que calcule la venta de tres productos.

1. Inicio
2. “Dame el precio del producto 1” p1
3. “Dame el precio del producto 2” p2
4. “Dame el precio del producto 3” p3
5. Total= p1+p2+p3
6. Imprimir “el total de los tres productos es” total
7. Fin

III. Diseñar un algoritmo que obtenga el valor de X con la fórmula X=1/b + 1/c + d.

1. Inicio
2. “Dame el valor de B” b
3. “Dame el valor de C” c
4. “Dame el valor de D” d
5. Y2=1/b
6. Y1=1/c
7. Y3= d
8. X=y2+y1+y3
9. Imprime
10. Fin

IV. Diseñar un algoritmo que lea de un mismo registro el nombre de un empleado, su salario básico por hora y el número de horas trabajadas en el mes. Calcular su salario mensual e imprimir tanto el nombre como su salario mensual.

1. Inicio
2. “Dame el nombre” n$
3. “Dame su sueldo por hora” sh
4. “Dame el número de horas trabajadas” ht
5. sd = sh * ht
6. sm = sd *30
7. Imprimir n$, sd, sm
8. Fin

V. Diseñar un algoritmo que lea un número y calcule el 5% del número leído. Obtener tanto el número como el porcentaje calculado.

1. Inicio
2. “Dame un número” num

3. pc = (5/100) * num
4. Imprimir pc, num
5. Fin

VI. Diseñar un algoritmo que lea de un registro el nombre de un empleado, su salario básico por hora, el número de horas trabajadas en el período y el porcentaje de retención. Calcular el salario bruto, el valor de la retención y su salario neto.

1. Inicio
2. "Dame el nombre" nombre$
3. "Dame el sueldo por hora" sh
4. "Dame horas trabajadas" ht
5. "Dame porcentaje de retención" p
6. sb = s x h * ht
7. r = (p * sb) / 100
8. sn = sb - r
9. Imprimir nombre$, sb, r, sn
10. Fin

VII. Diseñar un algoritmo que lea de un mismo registro el nombre de un empleado, su salario básico por hora y el número de horas trabajadas en el mes. Calcular su salario mensual e imprimir tanto el nombre como su salario mensual bruto y, si su aportación al fondo de pensión es el 12.28% de su salario mensual, imprimir su sueldo neto.

1. Inicio
2. "Dame el nombre "n$
3. "Dame su sueldo por hora" sh
4. "Dame el número de horas trabajadas" ht
5. sd = sh * ht
6. sm=sd*30
7. Pensi = (sm * .1228)
8. smnet = sm - pensi
9. Imprimir n$, sd, sm, Pensi, smnet
10. Fin

VIII. Diseñar un algoritmo que lea un número y calcule el cuadrado y el cubo del número leído. Imprimir el cuadrado y el cubo del número.

1. Inicio
2. "Dame un número" num

3. cuad = num * num
4. cub = num * cuad
5. Imprimir cuad, cub
6. Fin

IX. Diseñar un algoritmo que lea de un registro el nombre de un empleado, su salario básico por hora, el número de horas trabajadas en el período y el porcentaje de retención por seguro de vida, así como el porcentaje de impuesto al producto del trabajo. Calcular el salario bruto, el valor de la retención por seguro de vida, impuesto al producto del trabajo y su salario neto.

1. Inicio
2. "Dame el nombre "nombre$
3. "Dame el sueldo por hora" sh
4. "Dame horas trabajadas" ht
5. "Dame porcentaje de retención de seguro de vida" p
6. "Dame porcentaje de gravamen de impuesto al producto de trabajo" ispt
7. sb = sh * ht
8. r = (p * sb) / 100
9. ispTT = (ispt * sb)/100
10. sn = sb - r - ispTT
11. Imprimir nombre$, sb, r, ispTT, sn
12. Fin

X. Diseñar un algoritmo que calcule el área de un círculo.

1. Inicio
2. "Dame el valor de" pi
3. "Dame el valor de" r2
4. c=pi * r2
5. Imprimir "el área es" c
6. Fin

XI. La papelería "Guerra García" tiene a la venta los siguientes artículos:

Total	Artículos	Precio
2	Cuadernos	27
3	Cartapacios	45
2	Plumas	20

Obtener la venta total de la papelería.

1. Inicio
2. “Dame la cantidad de cuadernos” a
3. “Dame la cantidad de cartapacios” b
4. “Dame la cantidad de plumas” c
5. c= a * 27
6. d= b * 45
7. p= e * 20
8. vt= c + d + p
9. Imprime “la venta total es” vt
10. Fin

XII. Diseñar un algoritmo para un alumno de electrónica de la Unidad Reynosa Rodhe de la Universidad Autónoma de Tamaulipas que desea saber cuál será su calificacion en la materia de Lenguajes I. Dicha calificación se compone de los siguientes porcentajes:

45% Promedio de tres calificaciones parciales.
30% Calificación del examen final.
15% Calificación de un trabajo final.
10% Asistencia. (Si tiene 90% de asistencia).

1. Inicio
2. “Dame la calificación 1” c1
3. “Dame la calificación 2” c2
4. “Dame la calificación 3” c3
5. “Dame la calificación del examen final” cf
6. “Dame la calificación del trabajo final” tf
7. Promedio = (c1 + c2 + c3)/3
8. pprom = prom * .45
9. pef = cf * .30
10. ptf = tf * .15
11. ta = ta *.10
12. cf = pprom + pef + ptf + ta
13. Imprimir cf
14. Fin

XIII. Diseñar un algoritmo que calcule la edad de un alumno de la Preparatoria por Cooperación “José de Escandón”.

1. Inicio

2. “Dame el año de nacimiento” fnac
3. Dame año actual” fac
4. Edad = fac – fnac
5. Imprimir edad
6. Fin

XIV. Diseñar un algoritmo que obtenga una cantidad en pesos dada por cliente, obteniendo la equivalencia en dólares, asumiendo que el valor del dólar es cambiante.

1. Inicio
2. “Dame el total en pesos que quieres cambiar” p
3. “Dame el valor actual del dólar” d
4. e = p/d
5. “Total de pesos convertidos a dólares es” e
6. Fin

XV. Desarrollar un algoritmo para el laboratorio de la Unidad Académica Reynosa Aztlán donde los datos de entrada son volumen, temperatura y presión. Para obtener el valor de la masa se aplicará la fórmula: Masa = (presión*volumen) *(.37*(temperatura + 460)).

1. Inicio
2. “Dame el volumen” v
3. “Dame temperatura” t
4. “Dame presión” p
5. Masa = (presión*volumen) * (.37 * (temperatura + 460))
6. Imprimir masa
7. Fin

XVI. Desarrollar un algoritmo que permita determinar el volumen de un cilindro dado su radio y su altura.

1. Inicio
2. “Dame su altura” h
3. “Dame su radio” r
4. pi = 3.1416
5. y= pi * r**2 * h
6. Imprimir y
7. Fin

$y = pi * r^2 * h$

XVII. Diseñar un algoritmo que calcule el valor absoluto de un número. El número se desconoce, el usuario lo introducirá por medio del teclado.

1. Inicio
2. “Dame un número” x
3. va = x * x
4. va = √va
5. Imprimir “el valor absoluto de x es” va
6. Fin

XVIII. Diseñar un algoritmo que calcule si el número de pulsaciones que tiene un paciente son adecuadas con base en la fórmula: Número de pulsaciones = (220-edad) /10.

1. Inicio
2. “Dame la edad” E
3. Número-de-pulsaciones = (220-edad) /10
4. Imprimir “el total de pulsaciones son” Número-de-pulsaciones
5. Fin

XIX. Diseñar un algoritmo que imprima en pantalla nombre, sexo, edad y estado civil de quienes son ingresados por teclado.

1. Inicio
2. “Dame nombre” n$
3. “Dame género” g$
4. “Dame edad” e
5. “Dame estado civil” ec$
6. Imprimir n$, g$, e, ec$
7. Fin

Ejercicios a resolver por el alumno

Los siguientes cuatro ejercicios valoran indicadores que trabajará el alumno en forma individual:

- Comprensión. Si identificó la información relevante del problema: ¿Cuáles son los datos de entrada? ¿Cuáles son las condiciones que se deben cumplir? ¿Cuáles son los datos de salida? Si recoge en forma organizada la información.
- Aplicación del método. Aquí se valora el proceso, es decir la técnica que exige el tipo de problema y que se hayan utilizado correctamente y en forma ordenada las variables y las fórmulas.
- Generación de resultados. Si son correctos, claros y concisos. Porque la eficiencia para lograr los mismos es importante, ya que se deben presentar diferentes alternativas de solución para elegir la opción de desarrollo óptima desde el punto de vista de programática.
- Análisis crítico. Analizar el procedimiento y proponer posibles mejoras.

1. Diseñar un algoritmo que obtenga lo siguiente: Un cliente de la florería "El romance" compra tres artículos: un mono de peluche, un arreglo floral y un globo. Al precio de cada producto se le agrega el IVA de la siguiente manera: 12% al primero, 15% al segundo y 16% al tercero. Imprimir cuánto pagará por cada uno de ellos y cuánto pagará en total. Ingresar como dato el costo de cada uno de los tres artículos.
2. Realizar un algoritmo que permita determinar el área de un rectángulo.
3. Desarrollar un algoritmo que lea la velocidad en metros por segundo y la convierta a kilómetros por hora.
4. Realizar un algoritmo que determine cuántos segundos hay en 9 horas.

En los ejercicios del cinco al siete trabajarán en equipos de 4 personas. Se evaluará la eficiencia en el desarrollo de cada algoritmo, el análisis crítico y el proceso de toma de decisión acerca de las alternativas de solución del problema abordado. Dicho desarrollo será presentado al grupo. Cada equipo deberá evaluar la aplicación del método, es decir, el proceso que usaron los equipos para llegar a obtener resultados. Cada corrección que se le haga al equipo contrario, para mejorarlo, contará un punto para su calificación diaria.

5. Calcular la venta por cliente y la total del día de la cafetería de la Unidad Reynosa-Rodhe llamada "Bravos" la cual ofrece los siguientes productos para consumir:

Comidas	Precio $	Bebidas	Precio $	Otros	Precio $
Tacos (orden de 5)	25	Refrescos	10	Papitas	12
Tostadas (orden de 4)	20	Jugos	12	Gansitos	8
Tortas	18	Café	14	Pay	12
Hamburguesa sencilla	25	Licuados	20	Helados	16
Hamburguesa especial	35	Té helado	15	Nachos	18
Enchiladas (orden de 5)	35	Agua de frutas	20	Ensaladas	28
Burritos preparados	15			Cigarros	4

6. Realizar un algoritmo que pida un número y luego calcule la raíz cuadrada del mismo.

7. En la clase de matemáticas, el maestro Paco pidió a sus alumnos crear un algoritmo que resuelva una ecuación de segundo grado del tipo ax2+bx+c=0. Las soluciones son x1= (- b + raíz (b2 - 4ac)) / (2 a), x2 = (- b - raíz (b2 - 4ac)) / (2 a). Elabore su algoritmo.

Autoevaluación

Anota en el paréntesis el inciso de la respuesta correcta.

1. ( ) ¿Cómo se llaman los elementos que pueden cambiar durante la ejecución del programa?

a) Condiciones
b) Constantes
c) Variables
d) Ninguna de las anteriores

2. ( ) La solución de un algoritmo por medio de fórmulas y variables es:

a) Cualitativo
b) Cuantitativo
c) Interpretativo
d) Ningunas de las anteriores

3. ( ) Una de las principales características de un algoritmo es:

a) Infinito, organizado, preciso
b) Finito, preciso, organizado
c) Finito, infinito, organizado
d) Todas las anteriores
e) Ninguna de las anteriores

4. ( ) ¿Cuáles son las principales partes de un algoritmo en el modelo de Von Newman?

a) Teclado, monitor, impresora
b) Entrada, proceso, salida
c) Variables, constantes, decisiones
d) Todas las anteriores
e) Ninguna de las anteriores

5. ( ) Un algoritmo puede a llegar ser infinito

a) Falso
b) Verdadero

6. ( )¿Cómo se representa una variable?

a) Una letra y un número
b) Un número y una letra
c) Un número y un guion
d) Un guion y una letra

7. Problema: Una persona sale de su casa rumbo a su trabajo y en el camino se da cuenta de que una de las llantas de su automóvil está ponchada.

Organiza el orden de los pasos a seguir para la solución de este problema.

1) _____Colocar el gato debajo del auto y levantarlo poco a poco hasta que sujete bien la orilla del auto y asegurarse de que quede fijo.
2) _____Guardar en la cajuela todo el material utilizado.
3) _____Poner los tapones y apretarlos bien.
4) _____Bajarse del auto y revisar las cuatro llantas para saber cuál es la que debe cambiarse.
5) _____Retirar la llanta averiada y colocar la nueva.
6) _____Abrir y buscar en la cajuela la llanta de refacción y la herramienta que se requerirá.
7) _____Colocar el auto en algún sitio plano y firme a fin de cambiar la llanta con seguridad.
8)_____Quitar los tapones y usar como herramienta la llave de cruz para aflojar la llanta ponchada.
9) _____Bajar el gato y retirarlo.
10)_____Sacar la llanta de refacción y la herramienta que se va a utilizar y ponerlas en el suelo.

8. Problema: Anota el inciso de la respuesta correcta a las preguntas sobre los algoritmos.

1)_____Conjunto de instrucciones que sirven para ejecutar una tarea.

a) Diagrama de flujo b) Algoritmo c) Problema

2)_____Nos asegura que si seguimos más de una vez la serie de pasos descritos llegaremos al mismo resultado.

a) Definido b) Finito c) Preciso

3)_____Indica el orden en que debe realizarse cada paso.

a) Finito b) Preciso c) Definido

4)_____Conjunto de pasos ordenados y sistematizados de carácter finito que llevan a la solución de un problema.

a) Problema b) Proceso c) Algoritmo

5)_____Son dos características de los algoritmos.

a) Corto y rápido b) Finito y preciso c) Rápido y correcto

6)_____Son etapas de la metodología de solución de problemas.

a) Análisis e Identificación b) Inicio y Solución c) Algoritmo y Diagrama de flujo

Tema II
Estructuras condicionales (Decisión o Condición)

Evaluación diagnóstica del Tema II

El alumno deberá haber asimilado los conceptos del tema anterior, realizado las prácticas del mismo y saber reconocer una necesidad, identificar el problema, recopilar información confiable, determinar causa-efecto, precisar condiciones, plantear varias alternativas de solución y seleccionar la más eficiente.

Objetivo del Tema II

Que el alumno sea capaz de crear una variedad de sentencias que sirvan para clasificar, filtrar, agrupar esos cálculos. Para ello se tiene que proporcionar un elemento o condición que sirva para probar si la condición se cumple o no.

Objetivos del alumno

- Identificar las estructuras selectivas que conforman un algoritmo.
- Reconocer operadores lógicos, relacionales y expresiones lógicas.
- Identificar la estructura **Si, entonces; Si no**.
- Identificar los contadores, acumuladores y centinelas.
- Utilizar las instrucciones condicionales.

Habilidades del alumno

- Comprende el uso de las estructuras selectivas en la solución del problema.
- Aplica operadores relacionales, lógicos y expresiones lógicas.
- Aplica la estructura Si, entonces; usa los contadores, acumuladores y centinelas.

Estrategias de aprendizaje

- Investigación bibliográfica.
- Investigación *web*.
- Solución de problemas reales.

Lecturas recomendadas:

- https://www.programarya.com/Cursos/C++/Condicionales/Condicional-if-else
- http://entrenamiento-python-basico.readthedocs.io/es/latest/leccion4/condicional_if.html

- https://support.office.com/es-es/article/Funci%C3%B3n-SI-69AED7C9-4E8A-4755-A9BC- AA8BBFF73BE2?ui=es-ESyrs=es-ESyad=ES
- https://duglasm.wordpress.com/tutoriales-de-progrmacion/tutorial-del-lenguaje-de- programacion-c/sentencia-de-control-if-else-if-else-en-c/

Algoritmos y lógica racional

El cerebro humano tiene ciertas condiciones que conducen a que una gran cantidad de veces se decida en forma automática si se realiza una acción u otra. Por ejemplo, si la puerta de la casa está abierta, seguimos adelante; si está cerrada, nos detenemos y la abrimos.

Tomando este proceso cerebral como modelo para la toma de decisiones, podemos inducir al alumno a comunicarse con la unidad central de proceso (CPU, es decir, con la computadora que usa) a fin de que se familiarice con los conceptos necesarios para definir la serie de instrucciones en la misma forma armónica y lógica que el ser humano lo hace en sus experiencias cotidianas. Para ello, contamos con las estructuras de control llamadas **Condicionales**, que precisamente sirven para imitar el proceso racional del cerebro humano cuando enfrenta el reto de tomar una decisión (en teoría, la mejor posible en el contexto, dentro de la dimensión finita de espacio-tiempo).

Estructuras condicionales

Las estructuras condicionales comparan una variable contra otra(s) variable(s) o constante(s), para que, con base en el resultado de esta comparación, se siga un curso de acción dentro del programa. Hay tres tipos básicos: las simples, las dobles y las múltiples.

1. Simples:

Se les conocen como "Tomas de decisión", que tienen la siguiente forma:

Si <condición>, entonces
  Acción(es)
Fin Si

2. Dobles:

Permiten elegir entre dos opciones o alternativas posibles en función del cumplimiento o no de una determinada condición. Se representan de la siguiente forma:

Si <condición>, entonces
  Acción(es)
  Si no

Acción(es)
Fin Si

Donde:

Si…….Es el comando de comparación.

Condición……Define la condición a evaluar.

Entonces…….Adverbio (que significa siendo así, en este caso) para el inicio de las acciones a realizar cuando se cumple la condición.

Acción(es)……Son las acciones que se ejecutan cuando se cumple la condición.

Si no…… Conjunción adversativa (de otra manera, en caso contrario) que determina las acciones a realizar cuando no se cumple la condición.

Dependiendo de si la comparación es cierta o falsa, se pueden realizar una o más acciones.

3. Múltiples:

Las estructuras de comparación múltiples son tomas de decisión especializada que permiten comparar una variable contra distintos posibles resultados, ejecutando una serie de instrucciones específicas para cada caso. El formato es el siguiente:

```
Si <condición>, entonces
  Acción(es)
Si no
   Si <condición>, entonces
     Acción(es)
   Si no
     .
     .          Varias condiciones
     .
Fin si
```

Ejemplos donde se aplican las estructuras condicionales

- En una tienda departamental se ofrece un descuento del 40% **si** el usuario compra más de mil pesos. **Si no**, el descuento no se aplica.
- Cuando un docente dice que **si** el promedio de calificaciones del alumno es de 6 o más, será aprobado. **Si no**, estará reprobado.
- Un usuario introduce una moneda en una máquina expendedora. **Si** el valor de esa moneda es suficiente para pagar la mercancía elegida, la máquina le da el producto. **Si no**, le pide que ingrese más dinero.
- **Si** el usuario da más dinero de lo que cuesta el producto, la máquina le devuelve el sobrante. **Si no**, le agradece su compra.

- Al ir manejando un carro, **si** el semáforo está en rojo, nos detenemos. **Si no**, seguimos.
- **Si** el alumno cumple con el 90% de asistencia, aprobará la materia; **si no**, reprobará.
- Una puerta automática se abre **si** se acciona el control de acceso. **Si no**, permanecerá cerrada.
- El uso de las cajas rápidas en un supermercado está condicionado a llevar un máximo de 10 artículos. **Si no**, hay que recurrir a las otras cajas.

Según Medina (2018) "las estructuras selectivas o condicionales se utilizan para tomar una decisión lógica. En ésta se evalúa una condición y en función del resultado se realiza una opción u otra".

Las acciones condicionales están formadas por una o varias condiciones lógicas que generan dos posibles alternativas: cierto o falso. Otro componente son las acciones o secuencias que se van a ejecutar si es verdadero o es falso. Además, las acciones que se ejecutan pueden ser de tres tipos: simples, dobles, o anidadas (Medina, 2018).

En las acciones o condiciones simples el formato es:

Si, condición de control lógica, **entonces**, acciones dentro del rango de la condición.

Si la condición lógica se cumple, se ejecuta todo lo que está después de **entonces** hasta el primer punto y coma que se encuentre. **Si no**, se ejecutará la parte que está **después** del primer punto y coma.

1. Inicio
2. "Leer una calificación" cal
3. Si cal>7 entonces "imprimir aprobado"; Ir a 5;
4. "Imprimir reprobado"
5. Fin

La desventaja en este caso es que si se cumple la condición lógica va a imprimir aprobado y también va a imprimir reprobado, a menos que se ponga un "ve a 5" al final de la línea 3.

En las acciones o condiciones de tipo doble el formato es:

Si condición de control lógica, **entonces**
{Acción, Acción, Acción}
Si no
{Acción, Acción, Acción}

Si la condición lógica se cumple, se ejecutará todo lo que se ponga del lado derecho de la condición hasta antes del primer punto y coma; además, si es más de una acción, deberá usarse la llave "{", que indica que inicia la parte verdadera de la condición. Para finalizar la acción, agregar la llave "}", que indica la terminación de la parte falsa. Pero si la condición no se cumple, se ejecuta todo lo que está después del **si no** hasta el primer punto y coma. También, si es más de una acción, se usan las llaves para indicar dónde inicia la parte falsa y dónde termina (Medina, 2018).

Por ejemplo:
Se requiere saber si un alumno puede votar o no, dependiendo de si tiene 18 o más años de edad. Se debe proporcionar desde el teclado la edad de un alumno.

El algoritmo se escribe de esta forma:

Si la condición, **entonces** {acción 1; acción 2; acción 3…};
Si no {acción 4; acción 5; acción 6…}; La secuencia sería la siguiente:

```
10. Inicio
20. "Dame la edad de un alumno" Ed
30. Si (Ed >=18) entonces
    Inicio
      Imprimir "es mayor o igual a 18"; imprimir "puede votar"; ve a 40;
    Fin
   Si no
    Inicio
      Imprimir "no es mayor o igual a 18"; imprimir "no puede votar"
    Fin
40. Fin
```

Acciones o condiciones de tipo anidado

La anterior forma de instrucción, que es la doble, y la anidada son las más usadas. Es anidada porque hay una acción dentro de otra acción. Esta acción lógica permite expresar las diferentes alternativas donde hay más de una condición a tomarse en cuenta (Medina, 2018). Ejemplo:

Si, condición de control lógica, **entonces**, acción 1
Si no
Si, condición de control lógica, **entonces**, acción 2
Si no, acción 3

Contadores y Acumuladores

Espinosa (2018) considera que "el contador es un valor que se incrementa o decrementa en una cantidad constante y que el acumulador es una variable que se incrementa o decrementa en una cantidad variable".

Contador

Es una variable en la memoria que se incrementará en una unidad cada vez que se ejecute una acción en el proceso. El contador se utiliza para llevar la cuenta de determinadas acciones que se pueden solicitar durante la resolución de un problema (Espinosa, 2018).

Cada vez que se inicie una rutina, dependiendo del lenguaje en que se va a ejecutar, se realiza la inicialización del contador o contadores. La inicialización consiste en poner el valor inicial de la variable que representa al contador. Generalmente se inicializa con el valor 0 (Espinosa, 2018).

Un contador es una variable cuyo valor se incrementa o decrementa en una cantidad constante cada vez que se produce un determinado suceso, acción o iteración. Los contadores se utilizan con la finalidad de contabilizar los sucesos, acciones o iteraciones internas en un bucle, proceso, subrutina o donde se requiera cuantificar. Deben ser inicializados antes del ciclo o proceso, e incrementados o decrementados dentro del ciclo (Espinosa, 2018).

La inicialización consiste en asignarle al contador un valor inicial, es decir, el número desde el cual establecemos que comience el conteo. El contador llevará la cuenta de las iteraciones dentro del algoritmo o el número de registro que se desee.

Acumulador(es)

"Se utilizan para ir almacenando los resultados parciales de las operaciones" (Espinosa, 2018).

Un acumulador es una variable en la memoria cuya misión es almacenar cantidades cambiantes. Esta variable ya debe de estar definida desde la entrada de datos.

La diferencia entre un contador y un acumulador es que mientras el primero va aumentando en el incremento constante elegido (es decir de 1 en 1, de 2 en 2, de 3 en 3, según se haya convenido), el acumulador va aumentando en una cantidad variable (Espinosa, 2018).

Sintaxis:

Acumulador:= Acumulador + Variable;

Ejemplo:

Suma:= Suma + Edad;

En este caso, a la variable **Suma** se le está incrementando una cantidad variable almacenada en **Edad**.

De igual forma, se pueden efectuar decrementos en un totalizador.

Ejemplo:

Total: = Total - Descuento;

Valores de inicialización

En el caso de que un programa requiera el uso de contadores o acumuladores, es importante inicializar los mismos. Normalmente, cuando se desea efectuar sumatorias, el valor se debe iniciar en 0 (cero). Si se desea totalizar multiplicaciones, el valor se debe iniciar en 1 (uno).

Operadores lógicos

Otros elementos importantes de aplicación frecuente son los operadores lógicos.

Los datos numéricos, los de serie y el valor nulo pueden funcionar como datos lógicos (Hernández, 2012). Los datos numéricos y los de serie pueden tener el valor lógico verdadero o falso. El valor numérico 0 (cero) es falso; todos los demás valores numéricos son verdaderos. Los datos de serie de caracteres que no son una serie vacía son verdaderos; una serie vacía es falsa. El valor nulo no es verdadero ni falso. Tiene el valor lógico especial nulo.

Los operadores lógicos realizan pruebas en expresiones lógicas. Las expresiones lógicas que se evalúan como cero o una serie vacía son falsas. Las expresiones lógicas que se evalúan como valor nulo son nulas. Las expresiones que se evalúan como cualquier otro valor son verdaderas.

Según Morris (2003), el resultado de las operaciones AND y OR se define en la tabla siguiente:

Valor de P	Valor de Q	Resultado de P y Q	Resultado de P o Q
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE
TRUE	UNKNOWN	UNKNOWN	TRUE
FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE
FALSE	UNKNOWN	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	TRUE
UNKNOWN	FALSE	FALSE	UNKNOWN
UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN

El resultado de las operaciones NOT se define mediante la tabla siguiente:

Operando	Resultado de NOT
TRUE	FALSE
FALSE	TRUE
UNKNOWN	UNKNOWN

Operador Y

Si la primera condición es verdadera y la segunda es verdadera, el resultado de la operación Y es verdadera. Si la primera condición es verdadera y la segunda condición es falsa, el resultado es falso. Si la primera condición es falsa y la segunda es verdadera, el resultado es falso. Si la primera condición es falsa y la segunda es falsa, el resultado es falso.

Por lo tanto, cuando se utiliza la operación Y, si las dos condiciones son verdaderas el resultado va ser verdadero, de lo contrario, es falso (Morris, 2003). Resultado de estas aseveraciones:

Condición 1	Condición 2	Resultado
Verdadera	Verdadera	Verdadera
Verdadera	Falsa	Falsa
Falsa	Verdadera	Falsa
Falsa	Falsa	Falsa

Operador O

Si la primera condición es verdadera y la segunda condición es verdadera, el resultado de la condición es verdadera. Si la primera condición es verdadera y la segunda condición es falsa, el resultado es verdadero. Si la primera condición es

falsa y la segunda es verdadera, el resultado es verdadero. Si la primera condición es falsa y la segunda es falsa el resultado es falso.

Por lo tanto, cuando se utiliza la O, con una condición que se cumpla, el resultado es verdadero. Si no, el resultado va a ser falso (Morris, 2003).

Condición 1	Condición 2	Resultado
Verdadera	Verdadera	Verdadera
Verdadera	Falsa	Verdadera
Falsa	Verdadera	Verdadera
Falsa	Falsa	Falsa

Operador No

Si la condición es verdadera, el resultado es falso y viceversa.

Condición	Resultado
Verdadera	Falso
Falso	Verdadera

Las operaciones aritméticas y relacionales tienen preferencia sobre las operaciones lógicas. Las operaciones lógicas se evalúan de izquierda a derecha (las sentencias **Y** tienen preferencia sobre las sentencias **O**) (Morris, 2003).

Operadores relacionales

Los operadores relacionales son símbolos que se usan para comparar dos valores. Si el resultado de la comparación es correcto, la expresión considerada es verdadera; en caso contrario, es falsa.

Deitel y Deitel (2015) indican que invertir el orden del par de símbolos en cualquiera de los operadores al escribirlo como =, =>, =<, respectivamente, es comúnmente un error de sintaxis y éste es un error de tipo lógico.

Significado	Operador	Ejemplo	Resultado
Mayor o igual que	>=	A >= B	A es mayor o igual que B
Mayor que	>	A > B	A es mayor que B
Menor o igual que	<=	A <= B	A es menor o igual que B
Menor que	<	A < B	A es menor que B
Diferente que	< >	A< > B	A es diferente que B
Igual que	= =	A = = B	A es igual a B
Asignación	=	A = 5	Asigna 5 a A

Sin embargo, Torres (2017) menciona que “los operadores aritméticos, relacionales y lógicos pueden aparecer mezclados en una misma expresión”. Siendo así, es conveniente indicar un orden jerárquico, para poder establecer el orden de ejecución.

Confundir el operador de igualdad con el operador de asignación produce un error lógico, no de sintaxis (Deitel y Deitel, 2015).

Tabla de orden de ejecución de todos los operadores

( )
*, /, mod, no
+, -, y
>, <, >=, <=, < >, =, 0

Se debe tomar en consideración que los operadores del mismo nivel se ejecutan de izquierda a derecha, para evitar errores de operación.

Ejemplos resueltos de algoritmos del tema II

I. Hacer un algoritmo que lea los nombres y edades de dos personas e imprima cuál de ellas tiene más edad.

1. Inicio
2. “Dame el nombre” nombre1
3. “Dame la edad” e1
4. “Dame el nombre” nombre2
5. “Dame la edad” e2
6. Si e1 = e2 imprimir (“Edades iguales”) ve a 9
7. Si (e1 > e2) entonces imprimir “nombre1 es mayor”
8. Si no, imprimir (“nombre2 es mayor”)
9. Fin

II. En una Granja están en venta N conejos: N1 blancos y N2 negros. Se venden X negros y Y blancos. Hacer un algoritmo que:

a) Imprima la cantidad de conejos vendida.

b) Si P1 es el precio de los conejos blancos y P2 es el de los conejos negros, imprima el monto total de la venta.

c) Imprima el color de los conejos que se vendieron más. Solución:

1. Inicio
2. “Cuántos conejos hay” N
3. “Cuántos conejos blancos hay” N1
4. “Cuántos conejos negros hay” N2
5. “El precio de venta de conejos negros es” P1
6. “El precio de venta de conejos blancos es” P2
7. “Se vendieron x conejos blancos” X
8. “Se vendieron y conejos negros” Y
9. CCV = X + Y Imprimir “La cantidad de conejos vendidos es……” CCV
10. MVenta = X * P2 + Y * P1
11. “Imprimir “El monto de la venta es……..” MVenta
12. Si (X > Y) entonces Imprimir “El color de conejo que más se vendió fue blanco” Si no, imprimir “El color de conejo que más se vendió fue negro”
13. Imprimir X “conejos blancos se vendieron”
14. Imprimir Y “conejos negros se vendieron”
15. Fin

III. Algoritmo que lea el importe bruto de una factura de la florería “El Romance” y determine el importe neto según los siguientes criterios:

a) Importe bruto menor de 20 000 no tendrá descuento.
b) Importe bruto mayor o igual de 20 000 tendrá 15% de descuento.

1. Inicio
2. “Dame el importe” importeb
3. Si (importeb >= 20000) entonces descuento = importeb * 0.15;
4. Si no, descuento = 0
5. Total = importe - descuen
6. Imprimir “el total es...” Total
7. Fin

IV. Algoritmo que obtenga cuatro números desde el teclado y muestre el mayor de ellos.

1. Inicio
2. Leer (a, b, c, d)
3. si (a >b) y (a>c) y (a>d) entonces
   Inicio
     Imprimir “a es mayor”
   Fin

4. Si (b> c) y (b>d) y (b>a) entonces
   Inicio
     Imprimir “b es el mayor “ Fin
5. Si c>a y c>b y c>d entonces
   Inicio
     Imprimir ‘el mayor es c”
   Fin
     Si no, imprimir “d es mayor”
   Fin

V. Algoritmo que dice el sexo mediante la inserción de un valor numérico en una variable.

1. Inicio
2. Imprimir “qué sexo eres, 1-hombre 2-mujer”, sexo
3. Si (sexo==1) entonces Imprimir “eres hombre” ve a 5
4. Si no, Imprimir “eres mujer”
5. Fin

Ejercicios a resolver por el alumno

Los siguientes cuatro algoritmos evalúan habilidades que trabajará el alumno en forma individual:

- Si identifica la información relevante del problema, ¿cuáles son los datos de entrada?, ¿cuáles son las condiciones que se deben cumplir?, ¿cuáles son los datos de salida? Si recopila en forma organizada la información.
- Asimismo, la aplicación del método, donde se valora el proceso, es decir, la técnica que exige el tipo de problema y que haya utilizado de forma correcta y ordenada las variables a usar y las fórmulas a utilizar.
- Otro aspecto a valorar son los resultados: si son correctos, claros y concisos. creatividad también es importante, porque deben presentarse diferentes alternativas de solución, para elegir en forma razonada la mejor opción de desarrollo.

1. En la Unidad Reynosa Rodhe se califica a N alumnos de la materia de Fundamentos. La materia se reprueba si se tiene un 10% de inasistencia y la evaluación es menor a 7. Se solicita que se imprima la relación de alumnos reprobados.

2. Proporcionar desde el teclado tres números y que se obtenga como resultado cuál es el mayor y cuál el menor.
3. A un trabajador de una maquiladora de la ciudad de Reynosa, Tamaulipas se le paga de acuerdo a sus horas laboradas. Más allá de las 40 horas se consideran horas extras, las cuales se pagan al doble. Obtener el sueldo neto del trabajador.
4. A un alumno de la Unidad Reynosa Rodhe se le dan tres calificaciones. Obtener el promedio. Si el promedio es de 10, imprimir en número romano su calificación X. Si es de 9, se imprimirá IX y así sucesivamente hasta el 7.

En los ejercicios del cinco al siete trabajarán en equipos de 4 personas. Se evaluará la eficiencia en el desarrollo de cada algoritmo, el análisis crítico y el proceso de toma de decisión acerca de las alternativas de solución del problema abordado. Dicho desarrollo será presentado al grupo. Cada equipo deberá evaluar la aplicación del método, es decir, el proceso que usaron los equipos para llegar a obtener resultados. Cada corrección que se le haga al equipo contrario, para mejorarlo, contará un punto para su calificación diaria.

5. Dada la siguiente tabla de la cafetería de la Unidad Reynosa Rodhe, llamada Bravos.

Comidas	**Precio $**	**Bebidas**	**Precio $**	**Varios**	**Precio $**
Tacos de fajita	10 c/u	Refrescos de lata	12 c/u	Papitas	12 c/u
Tacos de queso	8 c/u	Agua 500 ml	10 c/u	Paletas de hielo	10 c/u
Tacos de trompo	5 c/u	Agua 1000 ml	20 c/u	Cono de nieve	10 c/u
Tortas	25 c/u	Jugos	15 c/u	Chicles	2 c//u
Hamburguesa sencilla	30 c/u	Café	10 c/u	Cigarros	5 c/u
Hamburguesa doble	45 c/u	Chocolate	15 c/u	Chetos con queso	15 c/u
Guisados	60 c/u	Licuados	20 c/u		

Calcular cuánto se obtuvo de la venta por cliente y el monto de las ventas del día. Se sabe que diariamente se registra un aforo de aproximadamente 120 clientes con diversos tipos de consumo entre comidas, bebidas y/o varios, es decir, cada uno de los 120 clientes podría comprar al menos un tipo de producto o, a lo más, los tres tipos.

6. Dada la siguiente tabla de Arrendadora de Autos llamada "Tigres".

Tipo de auto	Costo por milla en pesos	Costo por día en pesos
Grande	8	16
Mediano	4	14
Chico	3	12

Los datos de entrada son los que necesitas para obtener el resultado, no se sabe cuántos clientes llegaron (suponer para el cálculo N clientes).

Generar los siguientes resultados:

a) Cuál carro se rentó más.

b) Obtener la utilidad total de cada tipo de carro.

c) Obtener la utilidad total de todos los carros.

7. El equipo de béisbol de la Unidad Académica Reynosa Rodhe, llamado "Bravos", se enfrentó en una serie de 5 partidos al equipo "Tigres", quedando de la siguiente manera:

Partido	Bravos	Tigres
1	5	5
2	4	2
3	3	5
4	4	4
5	5	6

a) Cuántos empates hubo.

b) Cuántos juegos ganó cada equipo.

c) Qué equipo ganó más juegos.

Autoevaluación

1. En una estructura condicional, ¿se puede comparar una variable con una constante? En una estructura condicional, ¿se puede comparar una variable con un valor?

De las dos preguntas anteriores:

a) ¿Cuál es verdadera o falsa? b) ¿Las dos son verdaderas? c)¿Las dos son falsas?

2. En el siguiente ejemplo ¿Cuál tipo de condición es?

Si estatura >20 entonces imprimir "ya es deportista"

Si no, imprimir “y juega futbol”
Fin.

3. ¿En qué tipo de condición se corre el riesgo de que se ejecute la parte falsa?
   a) Anidada b) Doble c) Simple d) Triple

4. ¿El contador es el que aumenta su valor en una cantidad variable?
   a) Verdadero b) Falso c) Ninguna de las anteriores

Tema III
Ciclo o Estructura Hacer-Mientras

Evaluación diagnóstica del Tema III

En este punto del programa el alumno requiere estar ejercitado en la solución de problemas y ser capaz de:

- Examinar el problema y determinar qué es lo que se quiere obtener.
- Planear la serie de pasos que le van a permitir obtener lo que se propuso en el análisis del problema.
- Tener claridad del proceso o el planteamiento a realizar.
- Determinar con qué equipo cuenta para lograr el objetivo.

Objetivos del Tema III

- Que el alumno sea capaz de evaluar la solución a problemas que requieren repetir uno o más bloques de instrucciones.
- Que el alumno utilice apropiadamente la estructura repetitiva **Mientras**.

Objetivos del alumno

- Reconocer operadores lógicos, relacionales y expresiones lógicas.
- Identificar la estructura **Hacer-Mientras**.
- Utilizar los contadores, acumuladores y centinelas.
- Utilizar las instrucciones condicionales.

Habilidades del alumno

- Saber usar las estructuras selectivas en la solución del problema.
- Saber aplicar los operadores relacionales, lógicos y expresiones lógicas.
- Saber usar la estructura **Hacer-Mientras** y los conceptos contadores, acumuladores y centinelas.

Estrategias de aprendizaje

- Investigación bibliográfica.
- Investigación *web*.
- Desarrollo de problemas reales tanto personales como del entorno cercano y remoto.

Lecturas recomendadas para saber más sobre la estructura de control Hacer-Mientras

- http://michelletorres.mx/ciclos-while/
- https://querocarlos.wordpress.com/ciclo-while-3/
- http://www.udb.edu.sv/udb/archivo/guia/informatica-tecnologico/introduccion-a-la- programacion/2013/i/guia-4.pdf

Ejemplos donde se aplica la estructura condicional Hacer-Mientras

En primera instancia, necesitamos comprender que el estatuto o acción **Hacer-Mientras** se utiliza cuando requerimos repetir uno o varios eventos un número finito de veces. Por ejemplo, podemos pedirle a un empleado de una maquiladora que introduzca desde el teclado una cantidad deseada de tacos a comprar las veces que él desee, por lo que podrá poner 5, 10, 15, 20, etcétera, y el algoritmo seguirá corriendo e ingresando órdenes de tacos. Solo dejará de hacerlo cuando se tenga una condición lógica, como podría ser ponerle un límite al empleado de solo 20 tacos. Los que excedan esta cantidad los pagará él. Una expresión lógica en este caso sería: cuando el usuario ingrese una cantidad menor o igual a 20.

Otros ejemplos:

- Una calificación mayor a 6 es requerida para aprobar una materia.
- Mientras pueda pagarlos, una persona podrá comprar artículos de un supermercado.
- Los alumnos de la Unidad pueden ingresar al sistema de la Universidad mientras estén inscritos en el sistema.
- Mientras tenga el carro gasolina, podrá usarse.
- Mientras se siga la dieta, se bajará de peso.
- Mientras se tenga un promedio de 8.5, se otorgará una beca.
- Un niño tiene permiso de jugar, mientras haya terminado la tarea.

Según Deitel y Deitel (2015) "una instrucción de repetición especifica que un programa debe repetir una acción mientras cierta condición sea verdadera".

En este ciclo, primero tenemos que verificar que exista una variable de control definida, aunque ésta inicie con un valor 0. Esta variable está antes del ciclo **Mientras**, para iniciar evaluando la condición. Si la condición se cumple, ejecuta todas las acciones que hay dentro del ciclo hasta que la condición ya no se cumpla. **Mientras** nos permite repetir un bloque de instrucciones. Si la condición no se cumple, no entra (Deitel & Deitel, 2015).

El formato de la acción o estatuto es el siguiente:

```
Inicio
Asignación de variable
Hacer-mientras (condición lógica)
Inicio
   Acción
   Acción
   Variable
Fin
```

Según Peñaloza (2005), "la sentencia o grupo de sentencias que se repiten en un bloque se denominan **cuerpo del ciclo** y cada repetición del cuerpo del ciclo se llama **iteración del ciclo**".

El ciclo **Mientras** tiene que tener su final, tiene un límite y su límite es hasta que la condición ya no se cumpla, o sea, que sea falsa.

Para Deitel (2015):

> Un error lógico conocido como ciclo infinito de repetición nunca termina si no se proporciona una acción en el cuerpo del ciclo de una instrucción que asegure que en algún momento se torne falsa la condición **Mientras**, para interrumpir y quebrar el ciclo.
>
> Si la condición no se cumple, nunca va a entrar al ciclo, pero si se cumple y por alguna razón la variable de la condición es alterada, jamás se interrumpirá el ciclo y esto provoca un ciclo infinito (Deitel & Deitel, 2015).

Centinelas y Banderas

De igual manera, "cuando no se conoce *a priori* el número de iteraciones que se van a realizar, el ciclo puede ser controlado por centinelas" (De-Lobos, 2010).

En un ciclo controlado por centinelas el usuario puede suspender la introducción de datos cuando lo desee, colocando una señal adecuada llamada centinela. Un ciclo **Repetir** controlado por centinela es cuando el usuario digita una letra para salir, como por ejemplo S o N, para indicar si desea continuar o no. El bucle debe repetirse hasta que la respuesta del usuario sea "n" o "N" (De Lobos, 2010).

En un ciclo **Mientras** controlado por tarea, la condición **Mientras** especifica que el cuerpo del ciclo debe continuar ejecutándose en tanto la tarea no haya sido completada. Cuando una decisión toma los valores de -1 o algún posible valor que no esté dentro del rango válido en un momento determinado, se le denomina centinela y su función primordial es detener el proceso de entrada de datos en una corrida de programa (De Lobos, 2010).

Banderas

Para Torres (2017):

> La variable conmutadora, que puede estar representada por los tipos de interruptores, switches, banderas o centinelas, puede tomar diferentes valores en la ejecución de un programa y dependiendo de dichos valores el programa puede variar la secuencia de instrucciones a ejecutar, es decir, tomar decisiones. En el caso de los switches, interruptores y banderas pueden tomar solamente dos valores durante la ejecución del programa, los cuales pueden ser 0 o 1. Se les suele llamar interruptores porque cuando toman los valores 0 o 1 están simulando un interruptor abierto/cerrado o encendido/apagado.

Ejemplo:

Hacer un algoritmo que lea 10 números y determine la suma entre ellos.

```
1. Inicio
2. i=0;
3. Suma=0;
4. Hacer Mientras (i <=10)
  Inicio
  Imprimir ("el valor de i es….", i);
  i=i+1;
  suma=suma + i;
  Fin;
5, Imprimir ("la suma es…,", suma);
6. Fin
```

Como vemos, la condición al principio se cumple, pero como i se va incrementando, al llegar a ser mayor que 10 se sale del *Mientras* y termina la ejecución.

Ejemplos resueltos de algoritmos del Tema III

I. Algoritmo que cuenta hombres y mujeres mientras se cumpla la condición verdadero o falso en la condición del estatuto *Mientras.*

```
1. Inicio
2. cm=0, ch=0, p, sexo=1, res=1;
3. Hacer Mientras (res==1)
  Inicio
  Imprimir "¿Cuál es el sexo de la persona? 1-mujer 2-hombre"
  leer sexo
  Si (sexo==1) cm=cm+1
```

si no
ch=ch+1
Imprimir "¿Hay más personas? 1-si 2-No",
Leer res
Fin
4. Imprimir "el total de hombres es" ch
5. Imprimir "el total de mujeres es" cm
6. Fin

II. Algoritmo que hace la sumatoria de números mientras se siga cumpliendo la condición de que haya más números que sumar.

1. Inicio
2. c, núm; res=1
3. Hacer Mientras (res==1)
Inicio
Imprimir "¿Qué número sumarás?"
Leer núm
c=c + núm
Imprimir "¿Hay más números? 1-sí 2-No",
Leer res
Fin
4. Imprimir "el total de la suma es" c
5. Fin

III. Algoritmo que hace el control de vacunas.

1. Inicio
2. Existencias= 1000, Entregadas=0
3. HacerMientras (Existencias >= 200)
Inicio
Imprimir "Introduzca el número de unidades entregadas"
Leer Entregadas
Existencias = Existencias – Entregadas
Fin
4. Imprimir "El inventario ha bajado de 200 unidades. Debe comunicarlo"
5. Fin

IV. Algoritmo que determina si cierta edad es aprobada o no.

1. Inicio
2. Edad =20;
3. Hacer Mientras (Edad> 18)
   Inicio
   Imprimir "¿Cuál es tu edad?" edad
   Imprimir "esaprobada"
   Leer edad
   Fin
4. Imprimir "No apruebas, tu edad no es la adecuada"
5. Fin

V. Algoritmo que sigue ejecutándose únicamente para los hombres.

1. Inicio
2. sexo=1;
3. Hacer Mientras (sexo==1)
   Inicio
   Imprimir "¿Cuál es tu sexo? Presiona tecla 1-hombre 2- mujer"
   Leer sexo
   Fin
4. Imprimir "¡Eres mujer!"
5. Fin

Ejercicios a resolver por el alumno

En los siguientes ejercicios algorítmicos se enumeran varios indicadores que trabajará el alumno en forma individual: La comprensión, gestión de información, el desarrollo de la capacidad de análisis y de síntesis.

Las actividades propician que el alumno presente diferentes alternativas de solución y, mediante su pensamiento crítico, seleccione la mejor opción de desarrollo.

En este proceso el alumno debe analizar el procedimiento y proponer posibles mejoras, evidenciando su capacidad de adaptación a nuevas situaciones. En esta línea, también manifestará su nivel de creatividad e innovación.

1. Imprimir los números del 1 al 100.
2. Calcular la suma de los cinco números enteros positivos.
3. Proporcionar desde el teclado una serie de números hasta llegar al 300.
4. Proporcionar desde el teclado una serie de números y sumar nada más los positivos.
5. Diseñar el algoritmo para calcular la siguiente serie: 0, 1, 1, 2, 3, 5, 8, 13...
6. Desarrollar el algoritmo que calcule las siguientes series: A = 3* 6 * 9 *12* 15*… N

En los ejercicios siete al once trabajarán en equipos de 4 personas para lograr la solución de un objetivo considerando los diferentes puntos de vista y evidenciando la creatividad, innovación y adaptación a nuevas situaciones.

7. Leer una serie de números diferentes a cero. Utilizar un centinela para terminar o empezar otra acción.
8. Imprimir tres números enteros y obtener el promedio.
9. Leer dos números e indicar si son primos.
10. Calcular el salario de un grupo de N trabajadores de la empresa "Tigres" proporcionando por el teclado el sueldo por hora y la cantidad de horas trabajadas.
11. Calcular el promedio de edades de hombres y mujeres de un grupo de alumnos de la carrera de LTI de la Unida Reynosa-Rodhe. Considerando que no se sabe la cantidad de alumnos del grupo, se puede pedir el valor o realizar el proceso mientras se tengan alumnos.

Autoevaluación

1. ¿Qué valores debe tener una bandera?
   a) 0 y 1 b) 1 y 2 c) 0 y 0 d) 0 y 2 e) 1 y 1

2. ¿De qué depende que funcione el ciclo Mientras?
   a) Variable b) Constante c) Constante y variable d) Ninguna de las anteriores

3. El ciclo Hacer-Mientras es:
   a) Una función de acceso a datos
   b) Una sentencia de control
   c) Un tipo de datos
   d) Ninguna de las anteriores

4. El ciclo Hacer-Mientras se ejecuta cuando:
   a) La expresión lógica es positivo
   b) La expresión lógica es negativo
   c) La expresión lógica sea verdadera
   d) La expresión lógica sea falsa

5. ¿De qué modo finaliza la sentencia Hacer-Mientras?
   a) Cuando no se cumpla la expresión
   b) Cuando encuentre el Fin del programa
   c) Cuando usamos la tecla control

6. ¿Cuál es el propósito de la sentencia Mientras?
   a) Actualizar el flujo de datos
   b) Gestionar el flujo de datos
   c) Administrar el flujo de datos
   d) Modificar el flujo de datos

7. ¿Cuándo se evalúa la expresión lógica Hacer-Mientras?
   a) Al principio b) En el proceso c) Al final

Tema IV
Ciclo o Estructura Repetir-Hasta

Evaluación diagnóstica del Tema IV

El alumno requiere tener conocimiento de conceptos como: variables, contadores, banderas, centinelas, condiciones y reglas de manejo a usar en el formato **Repetir-Hasta**, además de los conceptos de ciclos anidados.

Objetivos del Tema IV

- Que el alumno sea capaz de solucionar problemas que requieren el bloque **Repetir-Hasta** de acciones con aplicaciones.
- Que el alumno utilice apropiadamente la estructura o ciclo **Repetir-Hasta**.
- Que el alumno aplique las diferentes técnicas algorítmicas para la solución de problemas planteados.
- Que el alumno desarrolle el proceso de refinamiento de arriba-abajo y de izquierda a derecha.

Objetivos del alumno

- Reconocer operadores lógicos, relacionales y expresiones lógicas.
- Identificar la estructura **Repetir-Hasta**.
- Reutilizar los contadores, acumuladores y centinelas.
- Distinguir entre un ciclo **Hacer-Mientras** y un ciclo **Repetir-Hasta**.
- Utilizar las instrucciones condicionales.

Habilidades del alumno

- Analiza y comprende las diferentes formas de realizar una repetición, así como sus diferentes usos y aplicaciones en la solución de problemas.
- Maneja operadores relacionales, lógicos y expresiones lógicas.
- Aplica la estructura **Repetir** en problemas matemáticos y emplear los conceptos de contadores, acumuladores y centinelas.

Estrategias de aprendizaje

- Investigación bibliográfica.
- Investigación *web*.

- Desarrollo de problemas reales tanto personales como del entorno cercano y remoto.

Lecturas recomendadas para saber más sobre el tema de repetir

- http://www.udb.edu.sv/udb/archivo/guia/informatica-tecnologico/introduccion-a-la- programacion/2013/i/guia-4.pdf
- http://escodigo.com/algoritmos/procesos-repetitivos.html
- https://es.slideshare.net/jmachuca/taller-algoritmos-ciclo-repetir

La estructura condicional Repetir-Hasta

Deitel (2015) opina que "cuando termina la ejecución del ciclo, siempre se ejecutará, cuando menos una vez, el cuerpo del ciclo. Generalmente se utiliza como encabezado de una instrucción **Repetir-Hasta** sin llaves, alrededor de un cuerpo con una sola instrucción".

Un ciclo **Repetir-Hasta** se utiliza cuando queremos o necesitamos ejecutar uno o varios eventos un número específico de veces.

Por ejemplo, en la Unidad Reynosa-Rodhe podemos pedirle a un docente que introduzca desde el teclado la cantidad de oportunidades para presentar examen que pueda tener un alumno y el algoritmo seguirá su ejecución hasta que se tenga la condición lógica que representa el límite o fin en el **Hasta** (Deitel y Deitel, 2015).

Otros ejemplos serían:

- Cuando en la casa se lavan trastes (platos, cucharas, vasos etcétera), hasta que ya no queden trastes sucios.
- Cuando se cocina un platillo en el horno, se revisa periódicamente hasta que se haya terminado la cocción.
- Cuando se lava la ropa en la lavadora, se saca hasta que haya terminado el proceso de exprimido.
- Batir los ingredientes hasta obtener una mezcla homogénea.
- Si se hace un experimento químico y no se presenta el resultado esperado, se repite el procedimiento, hasta obtenerlo.
- El alumno acude a la escuela hasta que egrese.
- Un jugador de futbol soccer estará realizando una ronda de 5 penales consecutivos. Hasta que los anote dejará de tirarlos.

En esta estructura la acción o acciones del ciclo se repiten **Hasta** que la condición se cumpla. Permite realizar el proceso cuando menos una vez, ya que la condición se

evalúa al final del bucle, a diferencia del **Mientras**, donde el proceso puede ser que nunca llegue a entrar si la condición a evaluar no se cumple desde el principio. No necesita de un cuerpo principal o ciclo, el inicio o fin, el mismo ciclo de **Repetir-Hasta** sirve como delimitador para eliminar las palabras inicio y fin (Deitel y Deitel, 2015).

Según Márquez (2011), "funciona similar a la estructura o ciclo Mientras. La diferencia es que una evalúa al inicio del ciclo y la otra al final".

El cuerpo del ciclo es una pequeña parte dentro de un algoritmo o programa que será repetido. La condición es una variable o una función reducible a valor booleano.

Deitel y Deitel (2016) opinan que "primero se debe mostrar el valor del control, luego se incrementa el valor, después se condiciona, para observar si se va a continuar el ciclo o no."

Pero Gottfried (1994) dice que "la secuencia de sentencias se ejecutará rápidamente hasta que la expresión lógica llegue a ser cierta".

La representación de este ciclo o formato es:

Repetir
Acciones…
Hasta condición

Además, Gottfried (1994) establece que "la secuencia de acciones se ejecutará al menos una vez, ya que la expresión lógica se comprueba hasta el final".

Sanford y Nyhoff (1999) dice que:

> La expresión booleana es evaluada en ese momento y si es falsa, entonces se ejecutan todas las acciones otra vez. Este proceso de evaluar la expresión booleana y de ejecutar acciones se repite mientras que la condición sea falsa. Cuando sea cierta, la repetición se terminará.

Según Huamai (2015):

Toda estructura repetitiva tiene las siguientes partes:

- Inicialización, en la cual se asignan valores iníciales a las variables que intervienen en el test de salida.
- Actualización, en la que se actualizan las variables que intervienen en el test de salida.
- Instrucción de proceso, parte del bucle en el que se escriben las instrucciones que se deben repetir.
- Test de salida, donde se controla si el bucle continúa o se corta o finaliza el bucle.

Según Gottfried (1994), "no necesita estar comprendida entre un inicio y un fin, las palabras claves **repetir** y **hasta** actúan como corchetes que señalan el comienzo y el final de la secuencia de acciones".

Joyanes (2008) opina que hay reglas de funcionamiento que son:

1. La condición (expresión lógica) se evalúa al final del bucle, después de ejecutarse todas las sentencias.
2. Si la expresión lógica es falsa, se vuelve a repetir el ciclo y se ejecutan todas las sentencias.
3. Si la expresión lógica es verdadera, se sale del ciclo y se ejecuta la siguiente acción fuera del rango del ciclo.
4. El formato no requiere inicio y fin.

La diferencia entre un Hacer-Mientras y un Repetir-Hasta
La siguiente tabla representa la comparación de los ciclos **Mientras** y **Repetir** elaborada por Joyanes (2008).

Ciclo Mientras	**Ciclo Repetir**
1.* La condición (expresión lógica) se verifica antes de que se ejecute el cuerpo del ciclo.	1.** La condición se verifica después de que el cuerpo del ciclo se ha ejecutado.
2. Como resultado de (1*) el cuerpo del ciclo puede no ser ejecutado si la condición es falsa.	2. Como resultado de (1**), el cuerpo del ciclo se ejecutará al menos una vez (no importa cuál sea el valor de la condición).
3. Como resultado de (1*), las variables de la condición deben haber sido inicializadas antes de alcanzar la sentencia **Mientras**, de modo que la condición pueda ser verificada.	3. Como resultado de (1**) las variables de la condición no necesitan ser inicializadas antes de alcanzar la sentencia **Repetir**. A estas variables se le pueden asignar valores en el cuerpo del ciclo que se ejecutarán antes de que se verifique la condición.
4. Si la condición es verdadera, el cuerpo del ciclo se ejecutará y continuará.	4. Si la condición es verdadera, el cuerpo del ciclo habrá sido ejecutado, pero se detiene.
5. Para evitar el ciclo infinito debe asegurarse que la condición contenga una variable cuyo valor modifique el cuerpo del ciclo, pasando a tomar valor falso.	5. Para evitar un ciclo infinito, debe asegurarse que la condición contenga una variable cuyo valor se modifique en el cuerpo del ciclo, pasando a tomar el valor verdadero.

Según (Goldber et al., 1984), "el mayor defecto del ciclo **Repetir** es que por lo menos una vez se ejecuta el cuerpo del ciclo y el menor defecto es que la condición está hasta el final".

Ejercicios del ciclo Repetir-Hasta

I. Imprime los números del 1 al 5.

En el siguiente ejemplo imprime los números del 1 al 5, cuando llegue a 5 se sale del ciclo e imprime el mensaje "ya salí" y se acaba el algoritmo.

1. Inicio
2. a=1
3. Repetir
4. Imprimir a
5. a=a+1
6. Hasta (a=5)
7. Imprimir "ya salí"
8. Fin

II. Calcula la media de N números usando el ciclo Repetir

1. Inicio
2. Cuenta=1;
3. Suma=0;
4. "Dame valor de n" N
5. Repetir
   "Dame valor de r" r
   suma= suma+r;
   cuenta= cuenta+1
6. Hasta (cuenta > N)
7. Media=suma/N
8. Imprime "el resultado de la media es..." Media.
9. Fin

III. Se quiere obtener una agenda del salón de clases con el nombre, dirección y teléfono de cada alumno. No se sabe cuántos alumnos hay.

1. Inicio
2. CA=0;
3. Repetir
   "Dame el nombre" n$
   "Dame la dirección" d$
   "Dame el teléfono" t$
   "Ya son todos" r$
   Imprimir "el nombre es…" n$, "la dirección es…" d$, "su teléfono es…" t$
   CA=CA+1

4. Hasta R$="si"
5. Imprimir "el total de alumnos es…" CA
6. Fin

IV. Sumar un número que esté entre 10 y 20

1. Inicio
2. Sumar=0;
3. Repetir
4. "escribe un número" N
5. Sumar=sumar + N
6. Imprimir "el número sumado es…", suma
7. Hasta (N>=10 and N<=20)
8. Fin

Ejercicios para el alumno

Los siguientes cuatro ejercicios, a trabajar en forma individual, valoran varios indicadores:

- Comprensión del problema y toma de decisiones al presentar diferentes alternativas de solución y la mejor opción de desarrollo.
- Análisis crítico: el examen del procedimiento y la propuesta de posibles mejoras, lo cual muestra la flexibilidad y la capacidad de adaptación a nuevas situaciones.

1. Diseñar un algoritmo que reciba desde el teclado cifras hasta el número 500. Mientras no sea el 500, sumará todos los números y los imprimirá.
2. Diseñar un algoritmo que reciba desde el teclado el número de empleado y su edad de una empresa de Reynosa. Deberá repetirse hasta que el número de empleado sea 227 055 y la edad esté entre 20 y 25. Entonces imprimirá el mensaje "Este empleado sí está dado de alta".
3. Diseñar un algoritmo que calcule la suma de los números pares hasta que la suma llegue a 100.
4. Diseñar un algoritmo que reciba desde el teclado dos números, para dividirlos, y el cociente se imprimirá cuando el usuario diga que ya no quiere capturar más números. Finalizará la demanda de datos por teclado y la impresión de los cocientes.

En los ejercicios del cinco al nueve trabajarán en equipos de 4 personas. Se evaluará la eficiencia en el desarrollo de cada algoritmo, el análisis crítico y el proceso de toma de decisión acerca de las alternativas de solución del problema abordado. Dicho desarrollo será presentado al grupo. Cada equipo deberá evaluar la aplicación del método, es decir, el proceso que usaron los equipos para llegar a obtener resultados. Cada corrección que se le haga al equipo contrario, para mejorarlo, contará un punto para su calificación diaria.

5. Diseñar un algoritmo que calcule la tabla de multiplicar del número dado desde el teclado, hasta el número 10.
6. Diseñar un algoritmo que calcule el pago de 5 refrescos, el costo del refresco es de $35.90.
7. Diseñar un algoritmo que calcule el mayor de 10 números e imprima el resultado.
8. Diseñar un algoritmo que sume 5 números impares capturados por el usuario desde el teclado.
9. Diseñar un algoritmo que obtenga de los 10 números capturados por el usuario, cuántos son pares, cuántos son nones y cuántos son 0.

Autoevaluación

1. ¿Cuál es la diferencia entre el ciclo Repetir y el ciclo Mientras?
   a) Los delimitadores
   b) Los contadores
   c) Los acumuladores
   d) La condición

2. ¿Por qué se mantiene dentro del ciclo Repetir?
   a) Porque se cumplió el repetir
   b) Porque se imprimió el resultado
   c) Porque se activó el centinela
   d) Porque no se cumplió la condición

3. ¿Cuándo se finaliza el ciclo repetir?
   a) Cuando se suma el contador
   b) Cuando se cumple la condición
   c) Cuando no se cumple la condición asociada al contador
   d) Cuando no se cumple el centinela

4. Escribe una V de Verdadero o una F de Falso a las siguientes aseveraciones sobre el ciclo Repetir.

a)___Inicia con una condición.
b)___Se puede salir del ciclo antes de que termine la condición.
c)___Necesita de los corchetes para delimitar el cuerpo del ciclo.
d)___Para iniciar el ciclo empieza con un Hasta condición.
e)___Para iniciar el ciclo necesita de un contador.
f)___Podrá hacerse infinito.
g)___Se parece a la condición Si entonces.
h)___Es mejor que el ciclo Mientras.

Tema V
Ciclo o Estructura Desde-Hasta

Evaluación diagnóstica del Tema V

Se requiere previo conocimiento de variables de control, conceptos de incrementos o decrementos y repetición controlada mediante contadores.

También es necesario tener práctica en la solución de problemas y ser capaz de:

- Analizar el problema, para concretar qué es lo que se quiere obtener.
- Precisar el proceso o planteamiento a realizar.
- Planear la serie de pasos que le van a permitir obtener lo que se propuso en el análisis del problema.
- Determinar con qué infraestructura TIC se cuenta para lograr el objetivo.

Objetivos del Tema V

- Que el alumno logre identificar problemas para cuya solución se requiere la estructura **Desde-Hasta** con una o más acciones de instrucciones con sus diferentes usos y aplicaciones personales y sociales.
- Que el alumno utilice apropiadamente la estructura o ciclo **Desde-Hasta** para ejecutar las acciones de manera repetitiva en un problema.

Objetivos del alumno

- Reconocer los operadores lógicos, relacionales y expresiones lógicas para formar condiciones de control.
- Identificar la estructura **Desde-Hasta**.
- Reconocer los contadores, acumuladores y centinelas para operaciones controladas de repetición.
- Saber diferenciar los ciclos **Hacer-Mientras**, **Repetir-Hasta** y **Desde-Hasta**.
- Utilizar las instrucciones condicionales para alterar el flujo de información.

Habilidades del alumno

- Analiza y comprende los diferentes usos de realizar una repetición y sus aplicaciones en la solución de problemas.
- Aplica operadores relacionales, lógicos y expresiones lógicas.

- Aplica la estructura **Desde-Hasta** a problemas matemáticos, físicos, químicos y de cualquier área que lo necesite.
- Usa de manera adecuada los conceptos contadores, acumuladores y centinelas.

Estrategias de aprendizaje

- Investigación bibliográfica.
- Investigación *web*.
- Desarrollo de problemas reales tanto personales como del entorno cercano y remoto.

Lecturas recomendadas para conocer más sobre el ciclo Desde-Hasta

- https://sistemasumma.com/2010/11/04/ciclos-desde-hasta/
- https://es.wikipedia.org/wiki/Bucle_(programaci%C3%B3n)
- http://colabora.inacap.cl/sitios/merlot/Materiales%20MerlotChile/mlcastro/Negocios/Contabilidad/auditoria_computacion_contabilidad_estadistica/computacion/contenido_multimedia/elementos_de_programacion/5_4.html
- https://www.programarya.com/Cursos/C++/Ciclos/Ciclo-for

Ejemplos donde se aplica el ciclo o estructura Desde-Hasta

1. En una maquiladora se procesan piezas n veces, hasta que se tenga cierta cantidad de productos.
2. Para que un alumno de primaria se aprenda las tablas de multiplicar, tendrá que repetirlas hasta memorizarlas.
3. Para aprobar el examen, el alumno tendrá que estudiar hasta que maneje el tema.
4. En un videojuego el puntaje más alto es de 10. El usuario jugará hasta conseguirlo.
5. Hasta que se tengan 18 años serán aceptados en este trabajo.
6. Puedes comer hasta cinco galletas.
7. Los diez primeros alumnos que cumplan tales condiciones estarán exentos.

Este ciclo ejecuta una serie de instrucciones. El número de veces que lo haga se controla por medio de una variable que se inicializa con un valor. En cada ejecución del ciclo la variable sufre un incremento o decremento. El bloque de instrucciones se consuma mientras la condición sea verdadera (Jiménez et al., 2015).

La estructura o ciclo **Desde** se emplea para realizar ciertas repeticiones de una acción o acciones durante un número específico de veces.

El siguiente formato es el más común:

Desde (variable de control), **Hasta** (condición con incrementos o decrementos). Ejemplo: Desde (i=1; hasta condición; Incrementos o decrementos).

La estructura del **Desde** puede ser simple o compuesta, aunque la compuesta, al igual que los otros ciclos, puede incluir dentro de sus acciones otras estructuras de control. La variable de control determina el número de repeticiones que hará el ciclo.

Esta variable de control debe ser entera, jamás debe ser fraccionaria, y debe ser definida por el programador, ya sea iniciando una variable con diferente valor a cero o definiendo una variable de tipo constante antes de empezar el ciclo.

Sin embargo, (Jiménez et al., 2015) opinan que todos los ciclos tienes tres elementos para controlar el número de veces que se ejecutan las instrucciones que se encuentran dentro de ellos y son: inicio, una condición y un incremento o decremento.

Esta variable de control se puede incrementar o decrementar, dependiendo de la naturaleza del problema.

Analizando esta situación, el inicio, condición e incremento están incrustados dentro de la figura del ciclo (Jiménez et al., 2015).

Ejercicios del ciclo Desde-Hasta

I. Diseñar un algoritmo que imprima los números del 1 al 100.

1. Inicio
2. Desde (p==1; hasta p<101; p++) Imprimir “número” p;
3. Fin

II. Diseñar un algoritmo que Imprima los números impares del 1 al 10.

1. Inicio
2. Desde (I = 1; hasta I<10; I = I + 2) Imprimir “el número impar” I;
3. Fin

III. Diseñar un algoritmo que imprima los números pares del 1 al 10.

1. Inicio
2. Desde (i=2; hasta i<11; i=i+2)
3. Imprimir “el número par” i;
4. Fin

IV. Diseñar un algoritmo para obtener el número de hombres y mujeres de un grupo. La información será capturada mediante el teclado.

1. Inicio
2. cm=0, ch=0, p=0, sexo=0
3. Desde (p==0; Hasta p<40; p++)
   {
    "dame un valor: 1-mujer 2-hombre" sexo
    Si (sexo==1)
       cm=cm+1
    Si no
       ch=ch+1
   }
4. Imprimir "el total de mujeres es" cm;
5. Imprimir "el total de hombre es" ch;
6. Fin

V. Diseñar un algoritmo que lea 5 números dados por el usuario y luego muestre el resultado de la suma.

1. Inicio
2. entero: n, i, suma
3. suma=0
4. Desde (i= 1; hasta i<=5 ; i++)
   {
    imprimir "digite un número"; leer (n);
    suma = suma + n;
   }
5. Imprimir "la suma es:", suma
6. Fin

VI. Diseñar un algoritmo que lea un número y lo imprima; además, que lea otro número y que sume esos números.

1. Inicio
2. entero: a=0, i, n;
3. Desde (i= 1; hasta i<=3 ; i++)
   {
    imprimir "digite el número" i; leer n;
    a = a + n;
   }

4. imprimir “la suma de los valores n es:”, a;
5. Fin

Ejercicios para el alumno

Los siguientes algoritmos evalúan varias habilidades que el alumno mostrará en forma individual. La primera es la comprensión: evaluar la eficiencia de un algoritmo, para elegir entre diferentes alternativas la mejor opción de desarrollo. Asimismo, se evalúa la capacidad de gestionar la información y la habilidad de trabajar en forma autónoma, además de poner en práctica la creatividad e innovación para abordar los problemas. La resolución de estos ejercicios propiciará mayor nivel de práctica en estos indicadores.

1. Diseñar un algoritmo que imprima los números del 1 al 100.
2. Diseñar un algoritmo que calcule la suma de los cinco números enteros positivos.
3. Diseñar un algoritmo que calcule la suma de números impares de los primeros 100 números.
4. Diseñar un algoritmo que dibuje la siguiente figura:
   *
   * *
   * * *
5. Diseñar un algoritmo que calcule el promedio de N alumnos, capturando calificaciones desde el teclado. Además, que imprima la media de las calificaciones, la calificación mayor y el mensaje de “Excelente” y, si obtuvo el promedio menor a 6, mandar el mensaje “No acreditado”.
6. Diseñar un algoritmo que imprima los números del 10 al 1.
7. Diseñar un algoritmo que imprima cuántos números impares hay hasta el número 20.
8. Diseñar un algoritmo que calcule la suma de los números pares hasta el 20.
9. Diseñar un algoritmo que lea el número de estudiantes y tres calificaciones de cada alumno, para obtener el promedio de sus calificaciones y la calificación más alta.
10. Diseñar un algoritmo que calcule los primeros 5 factoriales usando ciclos anidados.
11. Diseñar un algoritmo que imprima las tablas de multiplicar del 1 al 10.
12. Diseñar un algoritmo que calcule el salario de un grupo de N trabajadores de la empresa “Tigres”. Dar la cantidad de horas trabajadas y el sueldo por

hora por trabajador. Además, calcular el sueldo promedio de los empleados.

13. Calcular el promedio de edades de hombres y mujeres de un grupo de alumnos de la carrera de LTI de la Unidad Reynosa-Rodhe. Considerando que no se sabe la cantidad de alumnos del grupo, se puede pedir ese dato. Además, imprimir el promedio de estatura y, de los diferentes deportes (futbol, beisbol, basquetbol), cuál es el que más juegan. Utilizar diferentes ciclos.

Autoevaluación

1. Determina el resultado o resultados dado el siguiente valor a la variable de control.

10. Inicio
20. J=5
30. Desde (I=1; Hasta I=5; I++) Imprimir "el valor de i es", i;
40. Fin

2. ¿Qué se define en el encabezado del ciclo?

a) La forma de imprimir
b) Los contadores
c) Los acumuladores
d) La variable de control

3. ¿De qué tipo debe ser la variable de control?

a) Fraccionaria
b) Entera
c) Booleana
d) Ninguna de las anteriores

4. ¿Dónde se incrementa la variable de control?

a) En el encabezado del programa
b) Dentro del cuerpo del ciclo
c) Fuera del ciclo
d) Antes de empezar el ciclo

5. Escribe una V de Verdadero o una F de Falso a las siguientes aseveraciones sobre el ciclo Desde- Hasta.

a) ___ No es repetitivo.
b) ___ La variable de control es fraccionaria.
c) ___ No hay delimitadores.

d) ___ Es considerado como uno de los limitadores el punto y coma.
e) ____ Dentro del cuerpo principal se incrementa o decrementa la variable de control.
f) ___ No se usan otras estructuras diferentes.

Tema VI
Arreglos

Evaluación diagnóstica del Tema VI

El alumno debe tener conocimiento de los conceptos de datos, información, estructura de datos, contadores, acumuladores, vectores y matrices matemáticas, elementos o componentes, tipos de datos o de referencia, índice o subíndice. También es importante tener práctica en solución de problemas.

Además, debe ser capaz de:

- Analizar el problema, para identificar sus elementos y planear su solución.
- Precisar con claridad los procesos a diseñar.
- Determinar los pasos para lograr la implementación del diseño propuesto.

Estas habilidades quedarán evidenciadas en la resolución de los problemas planteados mediante el desarrollo de algoritmos que generen los resultados esperados.

Objetivos del Tema Arreglos

- Conocer, construir, diseñar y escribir los arreglos.
- Utilizar los datos en un arreglo para manejarlos en una o dos dimensiones.
- Saber utilizar las listas y tablas.

Objetivos del alumno

- Comprender conceptos básicos para la manipulación de arreglos.
- Crear nuevos arreglos con sus diferentes formatos.
- Utilizar diversas formas de variables con índice.
- Presentar las diferentes alternativas o manipulaciones comunes.
- Usar una lista de variables-argumentos para crear procedimientos que puedan llamar a la rutina correspondiente con esos argumentos.
- Explicar cómo procesar los argumentos de la línea de comandos.

Habilidades del alumno

- Analizar y diseñar los diferentes modos de realizar un arreglo y sus aplicaciones en la solución del problema.
- Abordar en forma adecuada problemas matemáticos utilizando arreglos para su solución a través de un algoritmo o programa computacional.

Estrategias de aprendizaje

- Investigación bibliográfica.
- Investigación web.
- Desarrollo de problemas reales tanto personales como del entorno cercano y lejano.

Para conocer más sobre Arreglos

- http://mimosa.pntic.mec.es/~flarrosa/pseudoco.pdf
- https://users.dcc.uchile.cl/~lmateu/CC10A/Apuntes/ordenamiento/index.html
- http://aprendeenlinea.udea.edu.co/lms/men_udea/mod/page/view.php?id=14493
- http://www.monografias.com/trabajos100/diseno-algotirmos-arreglos/diseno-algotirmos- arreglos.shtml

Qué son los Arreglos y en qué casos pueden aplicarse

De acuerdo con (Jiménez et al., 2015), "un arreglo es un conjunto de datos o elementos de un mismo tipo y con un mismo identificador". También se define como un conjunto de variables del mismo tipo, que a la vez tiene diferentes formas de declararse, para darle valores a las variables del arreglo.

Estos autores señalan que el primer elemento de un arreglo tiene el índice de 0 y en algunas ocasiones se le denomina elemento cero. Todos los arreglos empiezan a definirse con el nombre de un arreglo, seguido de un corchete que se abre y se cierra, dentro del cual va una variable o un subíndice.

Una empresa comercializadora de productos de primera necesidad requiere saber: ¿Cuál producto se vendió más? ¿Qué día se vendió más? ¿Qué vendedor vendió más? Está información está en una tabla o matriz llamada ventas. Esta tabla sirve para tomar decisiones, desde comprar más producto o incentivar al empleado que vendió más, hasta saber qué día se vendió más producto y de qué tipo. También, cuánto se vendió en total de productos en la semana, la venta de cada empleado y del producto por día.

Ventas de la semana representadas en miles de pesos

	Lunes	Martes	Miércoles	Jueves	Viernes	Sábado
Vend1	50	45	54	34	10	34
Vend2	67	23	43	45	19	56
Vend3	78	34	21	56	18	67
Vend4	90	78	11	67	16	47
Vend5	89	09	12	78	14	39
Vend6	78	87	13	89	29	20
Vend7	45	65	25	10	23	29

La tabla anterior es una matriz de ventas; en otras palabras, es un arreglo. Además, se puede cruzar toda la información tanto horizontal como vertical. Es decir, los arreglos son los equivalentes en programación de las **matrices** y **vectores** de las matemáticas.

Otros ejemplos:

- Un edificio de departamentos, donde el nombre del edificio es el nombre del arreglo y los números de los departamentos son los índices o subíndices. Los elementos son los nombres de los renteros.
- Los libros de una biblioteca están clasificados por áreas del conocimiento. El nombre del arreglo es la biblioteca, los índices son las áreas del conocimiento y los elementos son los libros.
- En la lista de calificaciones de los alumnos de la Unidad Reynosa Rodhe, el nombre del arreglo es la materia, el índice es el número del alumno en la lista y los elementos del arreglo son los nombres de los alumnos y sus calificaciones.

Cómo se declara un arreglo

Se asigna a un arreglo el nombre de una variable y luego, dentro de corchetes, se especifica la cantidad de elementos que contendrá el arreglo. Después se le asignan valores establecidos por el usuario. Según Márquez y Osorio (2011) "a estos arreglos, también llamados vectores, listas o arreglos unidimensionales, su declaración, como la de cualquier variable, requiere de un nombre y un tipo de datos". El siguiente formato explica el párrafo anterior:

```
10. Inicio
20.  A [4]= {2, 3, 4, 5} //en esta línea se define un arreglo llamado A con
cuatro elementos definidos por el usuario //
30.  Imprimir a [2] // imprimirá el número 4//
40. Fin
```

Los vectores son arreglos unidimensionales que necesitan de un índice para diferenciar a cada elemento (Jiménez et al., 2015).

La otra forma de almacenar datos en una variable de arreglo es la siguiente: primero se define la variable con un índice y luego se le asigna un valor y así va a seguir hasta que coincida el número de elementos que se hayan definido en el arreglo. Es igual que el anterior procedimiento, solo que aquí se asignan los valores a variables de arreglo (Jiménez et al., 2015).

Por ejemplo:

```
10. Inicio
20. b [3] //se define un arreglo llamado b con tres elementos y el siguiente
paso es asignarles un valor a los elementos //
30. b [0] = 1; // en la posición b [0] se le asignara el valor 1// 40. b [1] = 3;
50. b [2] = 4;
60. b [3] = 5;
70. Imprimir el valor de b [2] //imprimiría 4//
80. Fin
```

La última forma de dar valores diferentes desde el teclado a las variables de arreglos es utilizando el ciclo Desde-Hasta, por ejemplo:

```
10. Inicio
20. A [5]
    Desde i=0, Hasta i<=6,i=i+1
    {
      Dame "valores del arreglo" A[i] //aquí se capturan mediante el teclado los
      cinco valores que el usuario quiera dar al arreglo A//
      Imprimir "valores de a[i]" A[i] //aquí se está imprimiendo cada valor que
      ha sido capturado por el usuario mediante el teclado //
    }
30. Fin
```

"Los arreglos bidimensionales reciben el nombre de **matrices**. Las matrices hacen uso de al menos dos índices. Uno es para indicar el número de filas y el otro es para indicar el número de columnas" (Jiménez et al., 2015).

Los **arreglos bidimensionales**, también llamados tablas o matrices, se diferencian de los vectores o listas en que los arreglos bidimensionales tienen dos subíndices, que significa columnas y filas que señalan la posición donde se encuentra el elemento indicado en la **matriz**. En cambio, en las listas o vectores

la posición dentro del arreglo la proporciona solo un índice (Jiménez et al., 2015).

Por ejemplo, un arreglo bidimensional podría ser una matriz de A[3,3] que describe el siguiente código algorítmico, donde los datos son introducidos de tres distintas formas con el auxilio de sus índices:

```
10. Inicio
20. A [ 3 ] [ 3 ]={4,74,6,43,68,8,34,57,89] //Sería la primera forma de llenado
de una matriz bidimensional//
30.  A [0,0]=4;
40. A [0,1]=74 ;// De la línea 30 a la 110 //Es la segunda forma de llenado de
una matriz bidimensional //
50.   A [0,2]=6;
60.  A [1,0]=43;
70.  A [1,1]=68;
80.  A [1,2]=8;
90.  A [2,0]=34;
100. A [2,1]=57;
110. A [2,2]=89;
120. Desde i=0 Hasta i=3;i=i+1 // Es la tercer forma de capturar valores en
el arreglo //
   {
    Desde j=0 Hasta j=3;j=j+1// Se utiliza 2 desde por los dos subíndices//
      {
       "dame valores para el arreglo de A", A [ i , j ]
      }
   }
130. Fin
```

En el paso 120 del programa algorítmico anterior se tiene que recorrer toda la matriz, para poder llenarla con los datos que se describen en la matriz explicita siguiente, recordar que los datos pueden ser de naturaleza no numérica como los casos de A y O.

A	0	1	2
O	4	74	6
1	43	68	8
2	34	57	89

Ejercicios resueltos de Arreglo

I. Crear un arreglo unidimensional de 4 elementos e imprimirlos.

```
10. Inicio
20.  A[4]={5,2,3,4};
30.  Imprimir a[0];a[1];a[2],a[3];
40. Fin
```

II. Crear un arreglo unidimensional de 3 elementos, dándoles valores a las variables y después imprimirlas.

```
10. Inicio
20. A[3];
30. A[0]=5;
40. A[1]=4;
50. A[2]=7
60. Imprimir A[0]; A[1]; A[2];
70. Fin
```

III. Crear un arreglo de 4 elementos que capture los valores del arreglo y que luego los imprima dentro de un ciclo Desde-Hasta.

```
10. Inicio
20. B [ 3 ]
30. Desde i=0; Hasta i<4; i=i+1
    {
     "Dame valor de" B[ i ];
     Imprimir "los datos de un arreglo son…" B[ i ];
    }
40. Fin
```

IV. Crear un arreglo que sume todos los elementos de un arreglo unidimensional.

```
10. Inicio
20. C [5];suma=0;
30. Desde i=0;Hasta i<6; i=i+1
    {
     "Dame valores del arreglo" C[ i ]
     Suma = suma + C[ i ];
    }
40. Imprimir "la suma de los elementos del arreglo es…..", suma;
50. Fin
```

V. Crear un vector de 5 elementos y pasarlos a otro arreglo con la misma dimensión.

```
10. Inicio
20. A[ 5 ]; B[ 5 ];
30. Desde i=1; Hasta i<6; i=0+1
    {
     "dame valores del primer arreglo" A [ i ];
     B [ i ] = A [ i ];//Pasa todos los elementos del arreglo A al B//
     Imprimir A[ i ]; B [ i ],
    }
40. Fin
```

VI. Crear un arreglo bidimensional 3*3. Sumar todos los elementos del arreglo.

```
10. Inicio
20. A [ 3 ] [ 3 ] Reservaste 3 columnas y tres renglones y tu algoritmo trabaja
con 4 x 4
30. Desde i=0 Hasta i=3; i=i+1 //Este es la otra forma de capturar valores al
arreglo//
    {
     Desde j=0 Hasta j=3;j=j+1// Se utiliza 2 desde por los dos  subíndices
       {
        "dame valores para el arreglo de A", A [i , j ]
       }
    }
40. Desde i=0 Hasta i=3;i=i+1
    {
     Desde j=0 Hasta j=3;j=j+1
       {
        Suma = Suma + A [ i , j ] // suma todos los elementos del arreglo//
       }
    }
50. Fin
```

Ejercicios

Los siguientes 7 algoritmos evalúan varias habilidades que el alumno evidenciará en forma individual. El primero es la comprensión, evaluar la eficiencia de un algoritmo, para elegir entre diferentes alternativas la mejor opción de desarrollo. Asimismo, se evalúa la capacidad de gestionar la información y la habilidad de trabajar en forma autónoma, además de poner en práctica su creatividad e innovación para abordar los problemas. La resolución de estos ejercicios propiciará mejor nivel de práctica en estos indicadores.

1. Crear una lista de 5 elementos y obtener el mayor de los cinco.
2. Crear un vector de 5 elementos y obtener el menor de los cinco.
3. Crear un arreglo de 5 elementos y obtener la suma de los números pares.
4. Crear un arreglo unidimensional de 20 elementos y obtener la suma de todos los números que se repitan.
5. Crear la tabla de multiplicar del número 5.
6. Crear, mediante el teclado y utilizando ciclo Desde-Hasta, una rutina para introducir todos los números del 1 al 20 y obtener el promedio de esos números.
7. Crear un arreglo bidimensional y obtener el mayor valor del arreglo.

En los siguientes ejercicios trabajarán en equipos de 4 personas. Se evaluarán la comprensión, gestión de información, capacidad de análisis y de síntesis. Las actividades propician que el alumno presente diferentes alternativas de solución y mediante su pensamiento crítico seleccione la mejor opción de desarrollo.

El planteamiento y los resultados serán presentados en equipo frente al resto de los compañeros. Cada equipo deberá evaluar la aplicación del método, es decir, el proceso que usaron los equipos para llegar a obtener resultados. Cada corrección que se le haga al equipo contrario, para mejorarlo, contará un punto para su calificación diaria.

8. Crear un arreglo de 6*6. Obtener la suma de los números pares.
9. Crear un arreglo de 3*3. Obtener la suma de la diagonal principal.
10. Crear un arreglo de 4*4. Obtener la suma de los números nones de la diagonal principal.
11. Crear un arreglo de 3*3. Obtener la suma de los números negativos.
12. Crear un arreglo de 3*5. Ordenar el arreglo de mayor a menor.

Autoevaluación

Subraya la respuesta correcta.

1. ¿Qué tipo de variables puede manejar un arreglo?
   a) Enteros
   b) Enteros y con punto decimal
   c) Cadena de caracteres
   d) Todas las anteriores

2. ¿Para qué se utiliza el subíndice?
   a) Para definir variables
   b) Para declarar archivos
   c) Para definir el contenido
   d) Para tener un número
   e) Para darle valor a una variable

3. ¿Cuántas dimensiones puede tener un arreglo?
   a) 1
   b) 2
   c) 3
   d) Infinito

4. ¿En qué posición se recomienda iniciar un arreglo al momento de crear los datos?
   a) 1
   b) 2
   c) 0
   d) 5
   e) Ninguna de las anteriores

5. Escribe una V de Verdadero o una F de Falso a las siguientes aseveraciones sobre el arreglo.
   1. ___ Se puede iniciar con cualquier índice.
   2. ___ Se pueden asignar valores al arreglo al momento de crearlo.
   3. ___ Un arreglo puede tener varias dimensiones.
   4. ___ A un arreglo bidimensional se le puede llamar vector.
   5. ___ Un arreglo es un conjunto de datos de diferente tipo.
   6. ___ Un vector es lo mismo que una matriz.
   7. ___ El índice es importante para un arreglo.
   8. ___ El índice se puede usar para asignar valores.

9. ___ Una lista es lo mismo que una tabla.
10. ___ En una tabla un elemento podría estar posicionado mediante el mismo índice.

Tema VII
Diagrama de Flujo

Evaluación diagnóstica del Tema VII

Al llegar a este nivel el alumno debió haber desarrollado de forma eficaz una serie de algoritmos, además de poner en práctica la metodología adecuada para su solución.

También, haber madurado sus habilidades en:

- Análisis del problema, para precisar la forma de resolverlo.
- Clarificar el planteamiento de la solución.
- Especificar el equipo TIC que se requiere para lograr el resultado.
- El proceso para obtener los resultados deseados.

Objetivos del Tema VII

- Que el alumno conozca, analice, construya, diseñe y desarrolle procesos computacionales mediante el uso de los bloques de diagramas de flujo.
- Que el alumno sea capaz de utilizar la simbología de los diagramas de flujo en cualquier tipo de problema.
- Que el alumno proponga mejoras al proceso.

Objetivos del alumno

- Conocer las diferentes fases de un proceso y su funcionamiento.
- Manejar conceptos básicos para la elaboración de los diagramas de flujo.

Habilidades del alumno

- Analizar y usar los diferentes bloques para llevar a cabo un diagrama de flujo para la solución de problemas.

Estrategias de aprendizaje

- Investigación bibliográfica.
- Investigación web.
- Practicar la aplicación de los conceptos estudiados en la solución de problemas reales, tanto personales como del entorno cercano y lejano.

Qué es un Diagrama de Flujo

Es la representación gráfica de un algoritmo o problemática.

Según Pinales y Velázquez (2014), "el diagrama de flujo es una herramienta que permite representar visualmente, mediante símbolos especiales, qué operaciones o pasos se requieren y en qué secuencia se deben efectuar para solucionar un problema dado".

Para Rancel (2015):

> Un diagrama de flujo es una representación esquemática de los distintos pasos de un programa. Constituyen pues, otra forma de representar algoritmos, distinta al pseudocódigo, pero que nos sirve de forma complementaria en el proceso de creación de la estructura del programa antes de ponernos delante del ordenador.
>
> (…) Es un mapa conceptual que trata de interpretar, facilitar la problemática de cualquier tipo en forma de figuras y con textos cortos. Además, el ahorro de tiempo es importante.

Los **Diagramas de Flujo** son una representación esquemática o gráfica, muy práctica, de mostrar el trabajo a seguir para describir un proceso. Nos permiten extraer de un algoritmo lo más relevante del problema, en un contexto específico, para poder analizar cada parte y cómo será abordada en el análisis. Aunque se acostumbra elaborarlos a mano, es preferible recurrir a *software* especializado que nos permita extendernos tanto como sea necesario y poder compartir fácilmente el resultado a nuestro equipo de trabajo.

Son muy útiles y se acostumbran en las dependencias gubernamentales y en las empresas en sus diferentes departamentos, ya que permiten visualizar y verificar la distribución del trabajo o procesos productivos industriales, proyectos de investigación y desarrollo, en el diseño de nuevas tecnologías y en departamentos de mejora continua, así como en la modernización de los procesos.

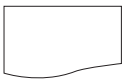
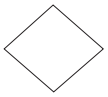
Ventajas y desventajas de los Diagramas de Flujo

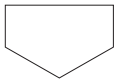
Una de las ventajas es que son fáciles de interpretar, más precisos, sencillos de entender. Además, se adaptan mejor a las necesidades del usuario; pero se convierten en desventaja si no se sabe utilizar de forma apropiada esta herramienta. Si hay ambigüedades, el diagrama de flujo se complicará; si no se sabe manejar el *software* adecuado para el manejo de este tipo de diagramación, se ocupará demasiado tiempo (Rancel, 2015).

Características de un Diagrama de Flujo

1. Claridad. Se refiere a la legibilidad general del problema. Si un diagrama de flujo está bien detallado, será sencillo que otro usuario lo entienda.
2. Simplicidad. Tanto como sea posible.
3. Eficacia. Se refiere a la velocidad de ejecución y el uso eficiente de la memoria.
4. Modularidad. La mayor parte de los problemas grandes pueden dividirse en pequeños módulos, procedimientos, funciones o sub-tareas por separado, para que cada uno realice operaciones (Rancel, 2015).

Simbología más usual de los Diagramas de Flujo

Bloque	Descripción del bloque
	Indica comienzo o final de un programa, subprograma o módulo.
	Impresión o salida de datos en pantalla u otro dispositivo.
	Captura y emisión de datos. Entrada o salida de información desde o hacia el ordenador.
	Entrada
	Pregunta
	Evaluación-decisión. Evalúa una expresión como cierta o falsa, siguiendo el programa distinta vía en función del resultado.
	Sentido del flujo de datos
	Sentido del flujo de datos
	Conector de secuencia de datos. Su etiqueta debe utilizar el mismo símbolo o carácter en el otro conector que se liga a él.
	Módulo independiente. Recibe distintos nombres, como subprograma, subrutina, proceso, procedimiento, función, etc. Al llegar a esta llamada el programa pasa a ejecutar todas las instrucciones contenidas en la subrutina, para una vez terminadas, continuar el flujo.

	Conector de página. Es el mismo concepto que el conector de secuencia, pero cambia al pasar a otra página.
	Proceso. Cualquier proceso interno realizado por la computadora, como asignación de valor a variables, operaciones matemáticas, etc.
	Significa ciclos. Serie de repeticiones de un proceso.
	Línea de flujo o flujo de datos. Sentido del flujo de procesos. Indica qué proceso viene a continuación del otro.

Flores (2003) propone que en primer lugar se analice el problema a resolver y, una vez que se estableció la ruta de solución, se proceda a elaborar el diagrama, respetando que el flujo deberá ser de izquierda a derecha, es decir, que la materia prima entre por la izquierda y los productos terminados o corrientes de desecho salgan por la derecha. Además:

- Deberá ser orientado horizontalmente, con bloques en el diagrama que emulen a los procesos reales.
- Deberá dejar espacio suficiente entre los bloques, que permita la inclusión de líneas de flujo adicionales, además de proporcionar un uso eficaz del papel y lograr un equilibrio visual.
- Las corrientes del proceso se designarán con líneas gruesas.
- Si las corrientes se cruzan sin mezclarse, una de las líneas se corta (generalmente la vertical), para permitir un espacio en el punto de cruce.
- Las cabezas de las flechas deberán ser dibujadas en todas las corrientes, entrando en la intersección cuando las líneas de flujo se unan.

Reglas simplificadas de los Diagramas de Flujo

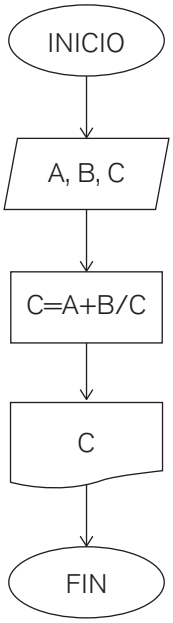
1. Todo diagrama debe tener inicio y fin.
2. Toda simbología debe estar conectada.
3. Los flujos de datos debe ir de arriba hacia abajo y de izquierda a derecha, no utilizar diagonales ni verticales. No deben cruzarse.
4. Utilice tantos conectores como sea necesario.
5. Puede utilizar cualquier tipo de decisión o condición un rombo o un triángulo. Cualquiera de los dos tiene una entrada y dos salidas (verdadero o falso).

6. Dentro de la figura debemos escribir información los más explícita posible, pero también lo más implícita posible y unívoca.
7. Nunca se debe dejar de considerar alguna aunque no sea probable que suceda.

Algunos ejemplos de Diagramas de flujo

I. Diseñar un algoritmo y un diagrama de flujo que Sume A y B, dividido entre C.

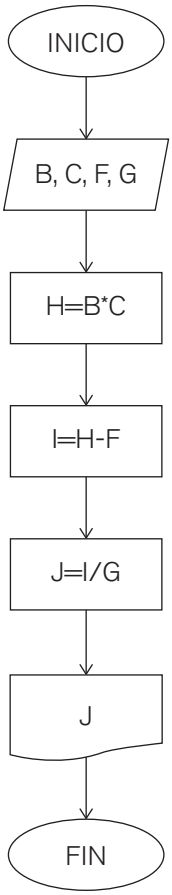
1. Inicio
2. "Dame el valor de a" A
3. "Dame el valor de b" B
4. "Dame el valor de c" C
5. C=A+B/C
6. Imprimir "El resultado es" C
7. Fin



Fuente: Elaboración propia.

II. Diseñar un algoritmo y un diagrama de flujo que obtenga el valor de "A" A=(B)(C)-F/G.

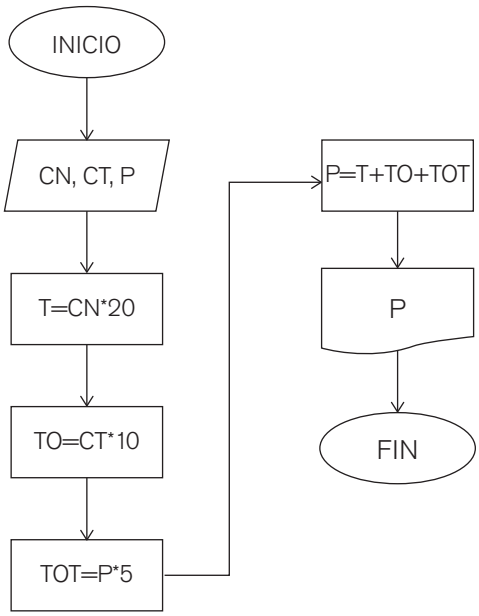
1. Inicio
2. "Dame el valor de b" B
3. "Dame el valor de c" C
4. "Dame el valor de f" F
5. "Dame el valor de g" G
6. H=B*C
7. I=H-F
8. J=I/G
9. Imprimir "El resultado es" J
10. Fin



Fuente: Elaboración propia.

III. Diseñar algoritmo y diagrama de flujo que obtenga el precio total en la venta de 3 productos. Los cuadernos valen 20 pesos cada uno. Los cartapacios valen 10 cada uno y las plumas valen 5 cada una.

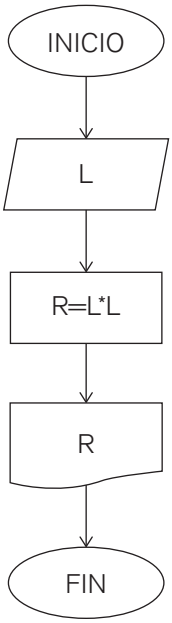
1. Inicio
2. “Dame cantidad de cuadernos” CN
3. “Dame cantidad de cartapacios” CT
4. “Dame cantidad de plumas” P
5. T=CN*20
6. TO=CT*10
7. TOT=P*5
8. P=T+TO+TOT
9. Imprimir “El total es” P
10. Fin



Fuente: Elaboración propia.

IV. Diseñar un algoritmo y un diagrama de flujo para obtener al área de un cuadrado.

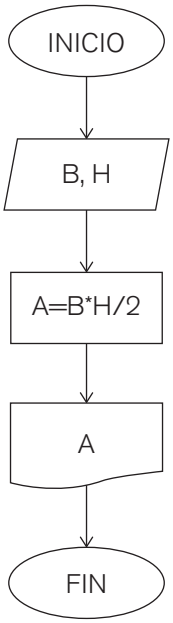
1. Inicio
2. “Dame la medida”. L
3. R=L*L
4. Imprimir “El resultado es” R
5. Fin



Fuente: Elaboración propia.

V. Diseñar un algoritmo que obtenga el área de un triángulo, dando desde el teclado la base y la altura.

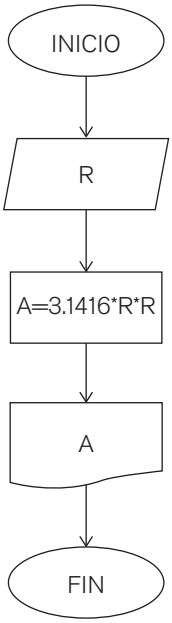
1. Inicio
2. “Dame base” B
3. “Dame Altura” H 4. A=B*H/2
4. Imprimir “El resultado es” A
5. Fin



Fuente: Elaboración propia.

VI. Diseñar un algoritmo y un diagrama de flujo que obtenga el área de un círculo.

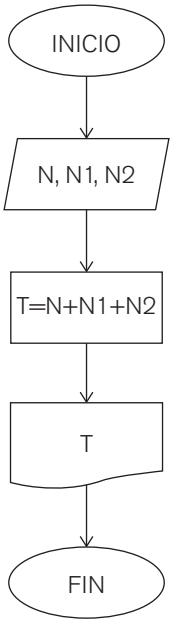
1. Inicio
2. “Dame el radio” R
3. A= 3.1416*R*R
4. Imprimir “El resultado es” A
5. Fin



Fuente: Elaboración propia.

VII. Diseñar un algoritmo y un diagrama de flujo que obtenga la suma de 3 números e imprima el resultado.

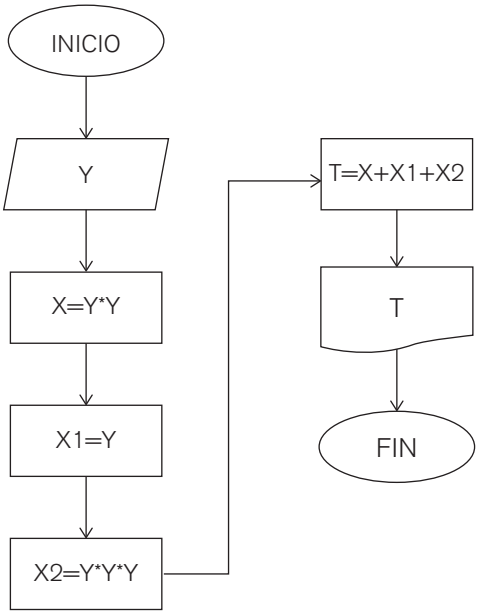
1. Inicio
2. “Dame el primer número” N
3. “Dame el segundo número” N1
4. “Dame el tercer número” N2
5. T=N+N1+N2
6. Imprimir “El resultado es” T
7. Fin



Fuente: Elaboración propia.

VIII. Obtener el resultado de Y usando la siguiente fórmula: X=y*y
X1=y
X2=y*y*y
T=X+X1+X2

1. Inicio
2. “Dame el valor de y” Y
3. X=Y*Y
4. X1=Y
5. X2=Y*Y*Y
6. T=X+X1+X2
7. Imprimir T
8. Fin



Fuente: Elaboración propia.

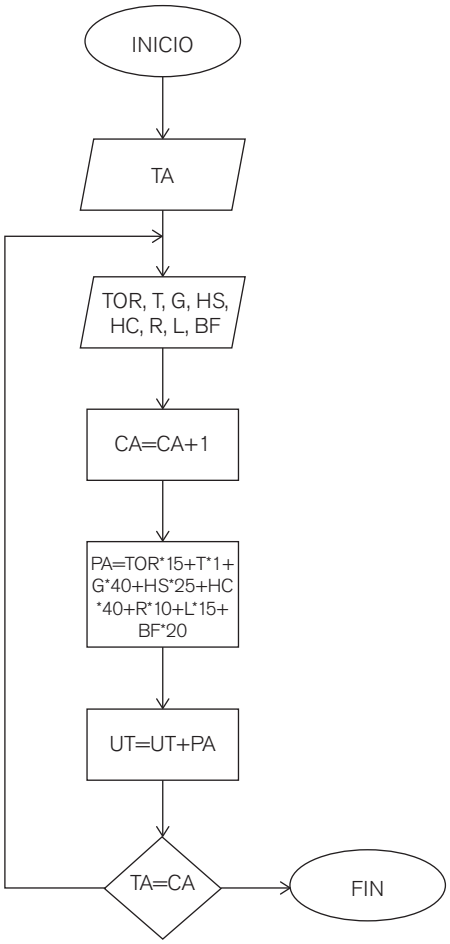
IX. La cafetería *Tigres* ofrece los siguientes productos:

Producto	Precio $	Producto	Precio $
Tortas	15	Bebidas frutales	20
Refrescos	10	Burritos	15
Tortilla con sal	1	Hamburguesas sencilla	25
Licuados	15	Hamburguesa doble carne	40
Guisados	40		

Diseñar un algoritmo y un diagrama de flujo que obtenga lo siguiente:

- Cuánto paga cada alumno por su consumo.
- Monto total del consumo de todos los alumnos.

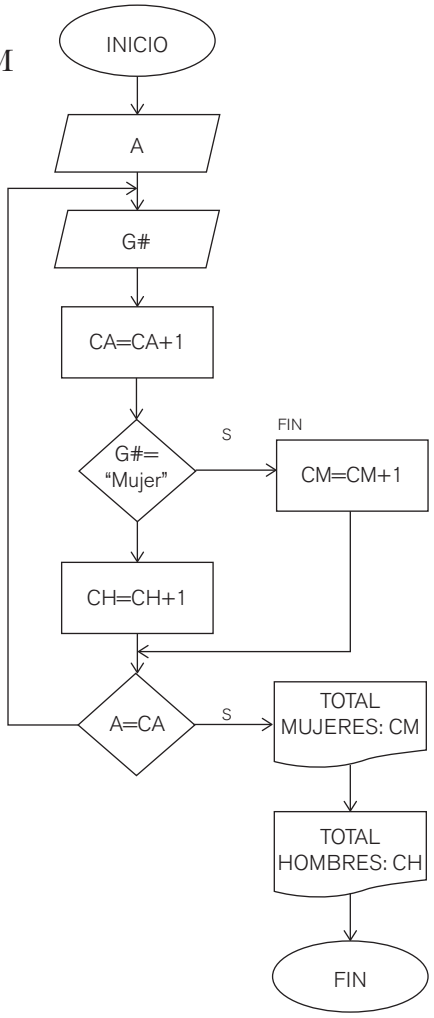
1. Inicio
2. “Dame total de alumnos” TA
3. “Dame total de tortas” TOR
4. “Dame total de tortillas” T
5. “Dame total de guisados” G
6. “Dame el total de H. Sencilla” HS
7. “Dame el total de H. Doble” HC
8. “Dame el total de refrescos” R
9. “Dame el total de licuados” L
10. “Dame el total de B. Frutales” BF
11. CA=CA+1
12. PA=Tor*15+T*1+G*40+HS*25+HC*40+R*10+L*15+BF*20
13. UT=UT+PA
14. Si TA=CA ir a 16
15. Ve a 3
16. Fin



Fuente: Elaboración propia.

X. Diseñar un algoritmo que indique la cantidad de mujeres y hombres que hay en un salón de clase.

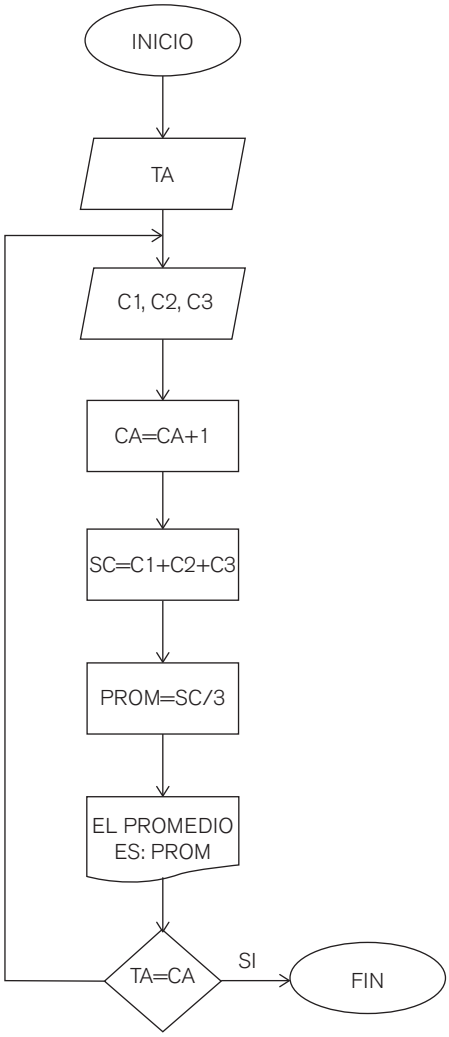
1. Inicio
2. “Cuántos alumnos son” A
3. “Dame género” G#
4. CA=CA+1
5. Si G#= “Mujer” CM=CM+1: ve a 7
6. CH=CH+1
7. Si A=CA Imprimir “El total de mujeres es” CM: ve a 9
8. Ve a 3
9. Imprimir “El total de hombres es” CH
10. Imprimir “El total de mujeres es” CM
11. Fin



Fuente: Elaboración propia.

XI. Diseñar un algoritmo y un diagrama de flujo que obtenga el promedio de calificaciones de 3 alumnos.

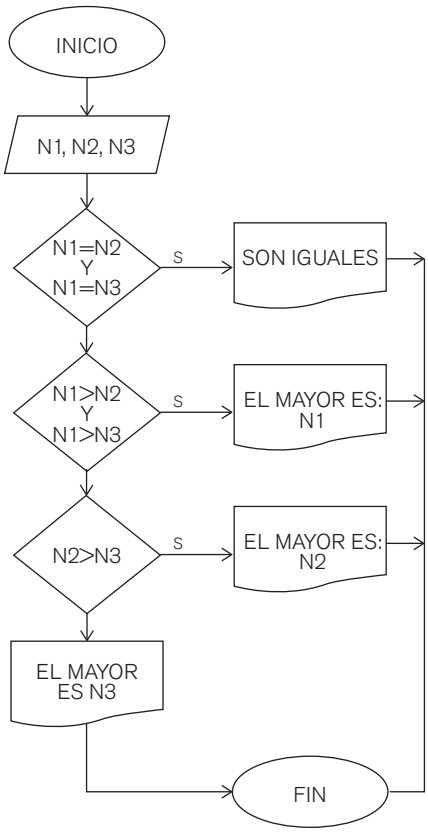
1. Inicio
2. “Dame total de alumnos ” TA
3. “Dame calificación 1” C1
4. “Dame calificación 2” C2
5. “Dame calificación 3” C3
6. CA=CA+1
7. SC=C1+C2+C3
8. Prom=SC/3
9. Imprimir “El promedio es” Prom
10. Si TA=CA: ve a 12
11. Ve a 3
12. Fin



Fuente: Elaboración propia.

XII. Determinar para tres números N1, N2 y N3 cuál es el mayor.

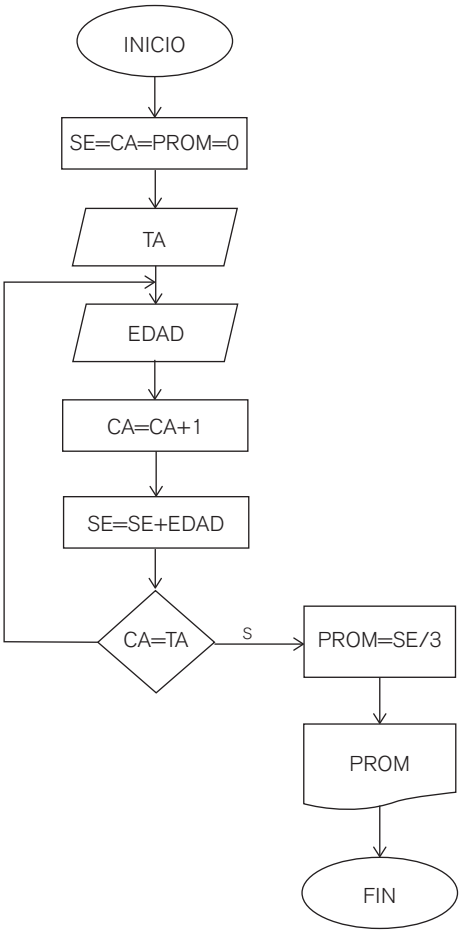
1. Inicio
2. “Dame el primer número” N1
3. “Dame el segundo número” N2
4. “Dame el tercer número” N3
5. Si (N1=N2) y (N1=N3) Imprimir “Son iguales” ve a 9
6. Si (N1>N2) y (N1>N3) Entonces imprimir “Es mayor” N1 ve a 9
7. (N2>N3) Entonces imprimir “Es mayor” N2 ve a 9
8. Imprimir “Es mayor” N3
9. Fin



Fuente: Elaboración propia.

XIII. Diseñar un algoritmo que calcule el promedio de edad de 3 alumnos.

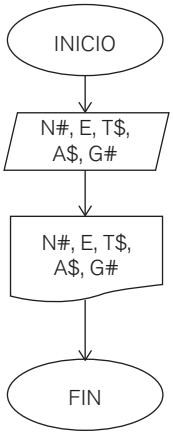
1. Inicio
2. SE=CA=PROM=0
3. “Dame cantidad de alumnos” TA
4. “Dame edad de alumno” EDAD
5. CA=CA+1
6. SE=SE+EDAD
7. Si TA=CA PROM=SE/3, VE A 9
8. VE A 4
9. Imprimir “El promedio es” PROM
10. Fin



Fuente: Elaboración propia.

XIV. Desarrollar un algoritmo y un diagrama de flujo que lea los datos de una persona para generar un registro de una agenda.

1. Inicio
2. “Dame el nombre” N#
3. “Dame la edad” E
4. “Dame el teléfono” T$
5. “Dame la dirección” A$
6. “Dame género” G#
7. Imprimir N#, E, T$, A$, G#
8. Fin



Fuente: Elaboración propia.

Referencias

De Lobos, M. E. (2010). *"Aprende a programar".* [Versión electrónica]. Recuperado el 24 de febrero de 2018: de http//www.mailxmail.com/curos-aprende-programar/tipos-estructuras-programacion-estructuras-basicas-secuencial

Deitel, P., y Deitel, H. (2015). "Cómo programar C++" (Novena ed., Vol. 9). Pearson Education. [Versión electrónica]. Recuperado el 31 de julio de 2018 de: www.pearsonenesespañol.com

——— (2016). *Cómo programar en Java.* Séptima edición Pearson Educación, 1152 p. México: Pearson.

Durán, F., Gutiérrez, F. y Pimentel, E. (2007). *Programación orientada a objetos en JAVA*. Madrid, España: Thomson. Escolano, F. et al. (2003). *Inteligencia artificial. Modelos, Técnicas y Áreas de Aplicación*. Madrid: Thomson.

Espinosa, S. (2018). *"Estructura repetitiva. Universidad Continental".* Recuperado el 25 de julio de 2018 de: http://repositorio.continental.edu.pe/bitstream/continental/4427/2/DO_UC_CFF_PO_Estructura%20repetitiva%20Mient ras%202018.pdf

Flores, J. (2003). *Método para la solución de problemas utilizando la programación orientada a objetos*. Perú: Universidad de San Martín de Porres.

Flores, L. (2017). "Algoritmos". *Vida cotidiana*. Boletín de la Escuela preparatoria no. 4. Repository revistas, 5(10). Recuperado el 25 de julio de 2018 de: https://repositury.uaeh.edu.mx/revistas/index.php/prepa4/article/view/2575

Forsythe, A., Keenan, T., Organick, E. y Stenberg W. (1969). *Lenguajes de diagramas de Flujo*. México: Limusa.

Goldber, C., Brainerd, W. y Gross, J. (1984). *Pascal*. Boston: Boyd & Fraser Publishing CompanyBoston. Gottfried, B. (1994). *Programación en Pascal* (Primera ed.). Naucalpan, México: Mc Graw Hill.

Guzdial, J. y Ericson, B. (2013). *Introducción a la computación y programación con PYTON* (Tercera ed.). (B. G. Hernández, Ed.) Naucalpan de Juárez, México: Pearson Educación.

Hernández, J. (2012). "Operadores de Basic". Recuperado el 16 de enero de 2018 de: https://www.ibm.com/support/knowledgecenter/es/SSZJPZ_9.1.0/com.ibm.swg.im.iis.ds.basic.doc/topics/r_dsbasic_Log ical_Operators.html

Huamai, J. (2015). "Algoritmo de estructura repetitiva. ESCODIGO". [Versión electrónica]. Recuperado el 25 de enero de 2018 de: http://escodigo.com/algoritmos/procesos-repetitivos.html

IntelDig. (2018). "Tipos de datos SQL para MySQL, SQL Server y MS Access". *Tecnologías de Información.* Recuperado el 18 de marzo de 2019 de: https://www.tecnologias-informacion.com/tipos-sql.html

Jiménez, J., Jiménez, E. y Alvarado, L. (2015). "Fundamentos de programación Diagramas de flujo, Diagramas N-S, Pseudocódigo y Java". México: Alfaomega. [Versión

electrónica]. Recuperado el 7 de abril de 2019 de: http://www.alfaomega.com.mx

Joyanes, L. (2008). *Fundamentos de Programación* (segunda ed.). México: Mc Graw Hill. Juganaru, M. (2014). *Introducción a la programación*. México: Grupo Editorial Patria.

Kosinski, M. (2017). "Bigdata, Inteligencia artificial y el futuro de la democracia". [Versión electrónica]. Recuperado el 25 de julio de 2018 de: http://bibliodigital.senado.gob.mx/handle/123456789/3499

Márquez, G. y Osorio, A. (2011). *Introducción a la programación estructurada en C.* (español ed.). México: Pearson Educación.

Medina, R. (2018). "Algoritmos. Selectivos o Condicionales". Recuperado el 25 de julio de 2018 de http://repositorio.continental.edu.pe/bitstream/continental/4433/1/DO_UC_CPGQT_PO_

Miranda, E. y Fuenlabrada, S. (2015). *Manejo de técnicas de programación* (primera ed.). (Sponsor, Ed.) México: Pearson Educación.

Morris, M. (2003). *Diseño digital* (3a. ed.). México: Pearson Educación.

Noguera, F. y Riera, D. (2010). *Programación*. En M. Marco, J. Marco, J. Prieto, & R. Segret, Escaneando la informática (págs. 115-117). Barcelona: UOC.

Oviedo, E. (2015). *Lógica de programación orientada a objetos* (1ª ed.). (U. d. Antioquía, Ed.) Antioquía: ECOE Ediciones.

Pinales, F. y Velázquez, C. (2014). *"Problemario de algoritmos resueltos con diagramas de flujo y pseudocódigo",* (Primera ed.). (D. E. Vinculación, Ed.) Universitaria, Aguascalientes, México: Universidad Autónoma de Aguascalientes. [Versión electrónica]. Recuperado el 06 de agosto de 2018 de: www.uaa/direcciones/dgdv/editorial/docs/algoritmos.pdf

Rancel, M. (2015). "Diagramas de flujo representan programas, símbolos básicos", (primera ed.). (N. M. Salas Ed.). [Versión electrónica]. Recuperado el 06 de agosto de 2018 de: www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=305:concepto-de-diagramas-de-flujo- para-representar-programas-simbolos-basicos-y-ejemplos

Restrepo, P. y Marín, L. (2011). "Un método computacional para la obtención de rutas óptimas en sistemas. DYNAL", 78(167), 112-121. Recuperado el 25 de julio de 2018 de: https://revistass.unal.edu.co/index.php/dinal/articule/view/25773

Sanford, L. y Nyhoff, L. (1999). *Programación Pascal* (Cuarta ed.). Madrid, España: Prentice Hall.

Torres, A. C. (2017). "Aprenda a diseñar algoritmos". Universidad Nacional Abierta y a Distancia, Bogotá DC: sello editorial UNAD. [Versión electrónica]. Recuperado el 25 de julio de 2018 de: https://books.google.com.mx/books?hl=es&lr=&id=SCpADwAAQBAJ&oi=fnd&pg=PP14&dq=OPERADORES+LOGICOS++en+un+algoritmo&ots=mW1Dm-eRnH&sig=OMCSyj

Peñaloza, H. (2005). *Fundamentos de Programación C/C++*. (4ª Ed.). México: Alfaomega Grupo Editor.

Vázquez, J. (2012). "Análisis y diseño de Algoritmos". [Versión electrónica]. Recuperado el 14 de octubre de 2019 de: http://www.aliat.org.mx/BibliotecasDigitales/sistemas/Analisis_y_disenio_de_algoritmos.pdf

Zúñiga, J., Fernández, R. y Guerrero, R. (2017). "El pensamiento computacional: Experiencia de una aplicación en el aprendizaje de la resolución del problema". Argentina: Congreso Argentino de Ciencias de la Computación. Recuperado el 25 de julio de 2018 de: htttp://hd.handle.net/10915/63918

Dr. Vicente Villanueva Hernández

Es Doctor en en Investigación Educativa por la Escuela Normal Superior de Ciudad Madero y Maestro en Educación Superior y Licenciado en Computación Administrativa por la Universidad Autónoma de Tamaulipas.

Profesor de Tiempo Completo de la Universidad Autónoma de Tamaulipas, adscrito a la Unidad Académica Multidisciplinaria Reynosa Rodhe. Es Integrante del Cuerpo Académico Consolidado Nuevas Tecnologías, Capital Humano y Competitividad. Pertenece al Sistema Nacional de Investigadores (SNI).

Es Coordinador de la carrera de Licenciado en Tecnologías de Información en Reynosa-Rhode. Ha recibido el reconocimiento al Mérito Universitario y al Profesor Extraordinario de la UAT.

Entre otras obras, ha participado en los libros: *Análisis del uso de las TIC en MiPyMEs en la región central de Tamaulipas*; *Estudio de la cultura multinacional, liderazgo y rendimiento laboral en maquiladora mexicana*. Ha participado en diversos congresos nacionales e internacionales.

Dr. José Rafael Baca Pumarejo

Es Doctor en Educación Internacional y Maestro en Administración por la Universidad Autónoma de Tamaulipas e Ingeniero en Sistemas Computacionales por el Tecnológico de Monterrey.

Profesor Investigador de Tiempo Completo de la Universidad Autónoma de Tamaulipas, adscrito a la Facultad de Comercio y Administración Victoria. Es Líder del Cuerpo Académico Consolidado Nuevas Tecnologías, Capital Humano y Competitividad. Pertenece al Sistema Nacional de Investigadores (SNI) Nivel 1.

Ha recibido el Premio a la Investigación de Tesis de Calidad, en los años 2014 y 2017. Así como el reconocimiento como Maestro Emérito de la UAT.

Sus más recientes proyectos de investigación están enfocados al tema de la Brecha Digital. Entre otras obras, ha publicado los libros: *Fortaleciendo el capital humano a través del crédito educativo, un análisis de su impacto internacional y en México*; *Análisis del uso de las TIC en MiPyMEs en la región central de Tamaulipas*; *Estudio de la cultura multinacional, liderazgo y rendimiento laboral en maquiladora mexicana.* Ha participado en diversos congresos internacionales con el tema de la brecha digital, tanto en escuelas como en empresas; formando asimismo capital humano en educación superior en carreras de Informática, de Contabilidad y Administración.

Desarrollo de Algoritmos y Diagramas de Flujo para la Programación de Computadoras. Guía de trabajo didáctico de Vicente Villanueva Hernández y José Rafael Baca Pumarejo, publicado por la Universidad Autónoma de Tamaulipas y Colofón, se terminó de imprimir en diciembre de 2020 en los talleres de Ultradigital Press S.A. de C.V. Centeno 195, Col. Valle del Sur, C.P. 09819, Ciudad de México. El tiraje consta de 400 ejemplares impresos de forma digital en papel Cultural de 75 gramos. El cuidado editorial estuvo a cargo del Consejo de Publicaciones UAT.