UNIDAD 2

Paradigma Imperativo

Introducción a la unidad

Uno de los paradigmas más representativos de la programación y de hecho el primer paradigma formalmente aceptado es el imperativo. Imperar significa mandar, ordenar, y eso es exactamente lo que hacemos al programar, ordenarle a la computadora algo que queremos que haga. Los primeros lenguajes de programación nacieron de la necesidad del programador de indicarle a la computadora una serie de instrucciones a seguir para resolver un problema. En este tema veremos los conceptos fundamentales del paradigma imperativo que son: celda de memoria variable, operaciones de asignación y operaciones de repetición.

Objetivo particular de la unidad

Identificar los conceptos más importantes que dan soporte y fundamento al paradigma de programación imperativa, y aplicarlos en la resolución de problemas algorítmicos propios de la automatización de la información y manipulación de datos.

LO QUE SÉ

Tomando en cuenta la introducción y el objetivo de la unidad, escribe tu propio significado de los siguientes conceptos:

- Variable
- Estructura de control
- Compilación
- Interprete
- Lenguaje máquina

Para enviar tu respuesta, pulsa el botón **Editar mi envío**; se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios**

Temas de la unidad II

1. Definición
2. Modularidad
3. Concepto de celda de memoria variable
4. Operaciones de asignación
5. Operaciones de repetición
6. Secuencia de transformación de datos
7. Campos de aplicación

Resumen de la unidad

La programación imperativa describe a la programación en términos del estado del programa y sentencias que modifican dichos estados.

Los programas imperativos resultan en un conjunto de instrucciones que indican a la computadora la forma en que realizará su tarea.

Los lenguajes imperativos, en un principio fueron en los lenguajes de máquina de las primeras computadoras. En dichos lenguajes, las instrucciones fueron muy simples, dando como resultado que su implementación fuera simple. A partir de Fortran, la programación imperativa permite la elaboración de programas complejos.

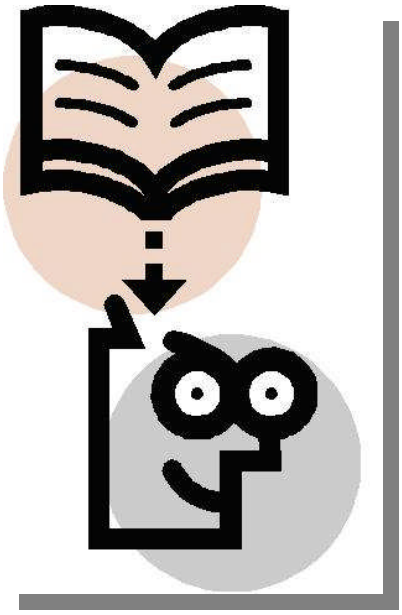
Tema 1. Definición

Objetivo del tema

Identificar el concepto de Programación imperativa.

Desarrollo

El paradigma imperativo apareció en los 50 con los primeros lenguajes de programación. También es llamado procedimental o algorítmico. Se llama así porque está basado en comandos u órdenes (enunciados imperativos) que actualizan variables que están almacenadas en memoria. Está basado en el modelo de Von Neumann, el cual define una máquina capaz de ejecutar una serie de instrucciones de forma secuencial, una tras otra. Estas instrucciones deben estar almacenadas en memoria principal para poder ser leídas y ejecutadas por la CPU (Unidad Central de Proceso).



ACTIVIDAD 1

Lee de nuevo el artículo de Dra. Oktaba, Lenguajes de programación **(ANEXO 1),** y también el capítulo 7 Hacia la estructuración: El paradigma imperativo **(ANEXO 2)** de Peña Marí, e identifica las cualidades principales (declaración de variables, estructuras de control, uso de la memoria), de los lenguajes descritos en las lecturas (Pascal, C, Ada) y elabora un cuadro comparativo con los datos obtenidos.

Después elige uno y realiza una sinopsis no mayor a dos cuartillas, donde describas las cualidades principales de ese lenguaje y justifica los puntos relevantes para catalogarlo como un lenguaje imperativo.

Sube ambas actividades a la plataforma.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

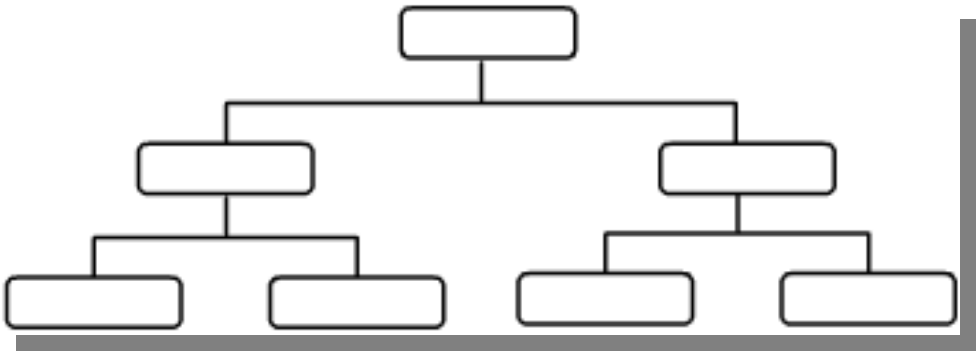
Tema 2. Modularidad

Objetivos del tema

Identificar a la modularidad como técnica de organizar el desarrollo de programa de cómputo.

Desarrollo

La programación imperativa se rige por dos conceptos básicos para la construcción de programas: la estructura y el módulo. De ahí que se hable de programación estructurada y de programación modular. Los dos tipos de programación imperativa señalan que un programa se debe dividir en subprogramas, segmentos o módulos para hacerlo más legible y manejable. Posteriormente se enlazan para lograr la funcionalidad deseada. Esta técnica también es conocida como refinamiento sucesivo, divide y vencerás ó análisis descendente (Top-Down).



Cada uno de los módulos en que se dividió el programa tiene una tarea bien definida y algunos necesitan de otros para poder operar. En caso de que un módulo necesite de otro, puede comunicarse con éste mediante una interfaz de comunicación.

Cada módulo es un procedimiento o función o conjunto de éstas, incluso son librerías con determinada funcionalidad que se pueden llamar desde un programa.

ACTIVIDAD 1

Investiga en tres fuentes distintas, tres definiciones sobre el algoritmo “divide y vencerás.

Después elabora tu propia definición y menciona dos ejemplos de este tipo de algoritmo.

Para enviar tu respuesta, pulsa el botón **Editar mi envío;** se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios.**

ACTIVIDAD 2

Supón que se va a desarrollar un sistema de información para controlar la recepción de vehículos en un taller mecánico.

La recepción del vehículo se lleva a cabo directamente con el cliente, mismo que da sus observaciones sobre los problemas o detalles que el vehículo presenta. El mecánico efectúa un diagnóstico inicial y lo deja sentado en la orden. Si el cliente acepta al diagnóstico, el jefe de mecánicos define las tareas y piezas que se emplearán en el servicio del vehículo.

Finalmente, el cliente es informado del presupuesto propuesto y decide si da la autorización o niega el servicio.

Tomando en cuenta lo anterior, elabora un documento de no más de tres cuartillas, definiendo a detalle los módulos conceptuales que propondrías para dividir el sistema a desarrollar. Cada módulo deberá indicar lo siguiente:

- Nombre
- Objetivo
- Parámetros de entrada
- Procesamiento de los datos
- Valores de salida (si es que aplica).

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 3

Se le va dejar el desarrollo de una agenda electrónica para el control de citas de un consultorio dental .Se requiere que se pueda dar de alta en ella los datos de los pacientes, las fechas de sus consultas, alergias a medicamentos y generar para cada persona un plan de pagos para cada cita. Es necesario que la agenda pueda generar la lista de pacientes de un día dado, informar sobre los medicamentos que puedan ser peligrosos para el paciente y el estado de pagos de cada paciente.

Basado en lo anterior, elabora una propuesta conceptual, de no más de tres cuartillas, indicando los módulos en que se dividirá la agenda. Cada módulo deberá indicar lo siguiente:

- Nombre
- Parámetros de entrada
- Operaciones que desarrollará
- Salida de datos (si así se requiere)
- Relación con otros módulos del sistema

Incluye un cronograma que refleje el orden de operación de cada módulo, con base en las acciones del usuario.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

Autoevaluación

Lee cuidadosamente cada oración e indica si es verdadera o falsa. Al final obtendrás tu calificación de manera automática.

	Verdadera	Falsa
1. Un módulo es un componente de un sistema más grande y opera dentro del sistema independientemente de las operaciones de otros componentes.	( )	( )
2. El paradigma imperativo es ajeno al concepto de comandos u órdenes que modifican el valor de las variables.	( )	( )
3. En caso de que un módulo necesite de otro, puede comunicarse con éste mediante una interfaz de comunicación.	( )	( )
4. El diseño Top-Down puede apreciarse como un refinamiento sucesivo pariendo de lo general a lo específico	( )	( )
5. Las librerías que pueden llamarse desde un programa no puede ser consideradas como un ejemplo de modularidad.	( )	( )

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

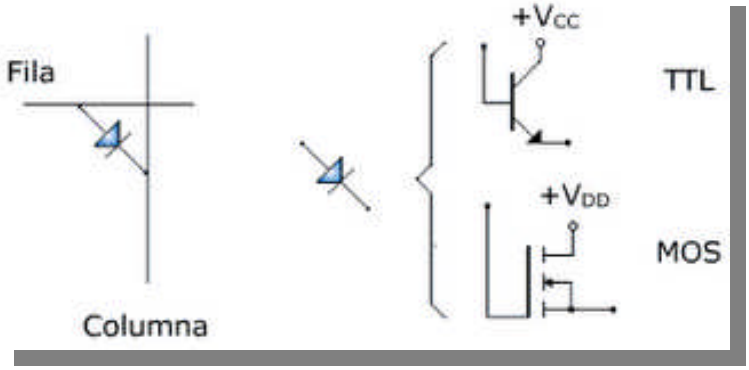
Tema 3. Concepto de celda de memoria variable

Objetivos del tema

Identificar el uso de variables en el desarrollo de programas y su relación con la memoria.

Desarrollo

La programación imperativa se basa en tres conceptos importantes: celda de memoria variable, operaciones de asignación y operaciones de repetición. La memoria de una computadora juega un papel primordial para que un programa pueda ser ejecutado. En ella se almacenan las instrucciones y los datos con los que el programa trabajará. La memoria es un conjunto de celdas identificadas por una dirección que es única: dos celdas no pueden tener la misma dirección. Pero en lugar de que en nuestros programas hagamos referencia a las direcciones de memoria (que están definidas en números hexadecimales), las “bautizamos” con un nombre (variable), así es más fácil referenciar el contenido de una celda de memoria.



ACTIVIDAD 1

En un documento Word, elabora un cuadro comparativo que sintetice el uso de variables entre los siguientes lenguajes: C, Java, PHP , ADA y SmallTalk. Los aspectos a comparar son:

- Declaración de variables
- Tipos de datos que maneja el lenguaje
- Representación en bits de cada uno de los tipos de datos disponibles.
- Ejemplo de tres declaraciones distintas de variables en el lenguaje.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 2

Investiga en mínimo dos fuentes distintas la definición de los siguientes términos. Después escribe tu propia definición sobre cada uno de ellos. No olvides citar tus fuentes.

- Locación de memoria.
- Apuntador
- Lenguajes tipificados
- Lenguajes no tipificados

Para enviar tu respuesta, pulsa el botón **Editar mi envío;** se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios.**

Autoevaluación

Elige la opción que complete correctamente cada una de las siguientes oraciones. Al final obtendrás tu calificación de manera automática.

1. En un momento dado, dos variables distintas ____ dirección de memoria:

- a) pueden tener la misa.
- b) no pueden tener la misma.
- c) establecen una relación de continuidad con una misma.

2. Las direcciones de memoria se expresan con números _____.

- a) decimales.
- b) octales.
- c) hexadecimales.

3. Las variables permiten _____ las direcciones de memoria para utilizarlas en la programación:

- a) nombrar.
- b) asignar.
- c) cambiar.

4. La _____ se conforma por celdas donde cada una se identifican por una dirección que es única.

- a) memoria.
- b) declaración de variables.
- c) asignación de direcciones.
- d)

5. Las operaciones de repetición _____ un conjunto de instrucciones.

- a) permiten la ejecución de
- b) la agrupación conceptual de.
- c) permiten varias veces la ejecución de.

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 4. Operaciones de asignación

Objetivo del tema

Identificar el uso de las operaciones de asignación en la programación.

Desarrollo

Cada valor calculado en un programa debe ser "almacenado", es decir asignado a una celda. Esta es la razón de la importancia de la sentencia de asignación en el paradigma imperativo. Las asignaciones se definen así:[1]

Expresión 1 = Expresión 2

Donde:

Expresión 1	es la localidad de memoria ya “bautizada”
Expresión 2	es el valor que le estamos asignando
=	lexema de asignación

Existen tres tipos de asignación:

Tipos de asignación
Aritmética:
Se refiere a cuando asignamos un valor numérico derivado de alguna operación aritmética. La variable que recibe el valor tuvo que haber sido declarada como tipo de dato numérico como *integer*, *long* o *double*, salvo algunas excepciones como en PHP o Ruby, en donde no hay declaración de variables. Ejemplo:
valor = 3 + 5 + 9;
Lógica:
Se refiere a cuando asignamos un valor de cierto o falso a una variable. La variable

[1] Rafael Menéndez-Barzanallana; “Metodologías usadas en ingeniería del software: Paradigma imperativo, material en línea, disponible en: http://www.wikilearning.com/curso_gratis/metodologias_usadas_en_ingenieria_del_software-paradigma_imperativo/3618-4, recuperado el 28/01/09.

que recibe el valor tuvo que haber sido declarada como tipo de dato *boolean*, salvo algunas excepciones como en PHP o Ruby, en donde no hay declaración de variables. Ejemplo:

valor = 20 < 5; (el valor que se asigna a esta variable es *false*)

Cadena de caracteres:

Se refiere a cuando asignamos como valor una cadena de caracteres (o uno solo, incluyendo el espacio en blanco) a una variable. La variable que recibe el valor tuvo que haber sido declarada como tipo de dato *string* o *char*, salvo algunas excepciones como en PHP o Ruby, en donde no hay declaración de variables. Ejemplo:

valor = “este es un ejemplo de asignación de cadena de caracteres”;

ACTIVIDAD 1

Genera una tabla donde se listen los operadores de asignación implementados en el lenguaje de programación Java (al menos 10), e incluye ejemplos de su utilización.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 2

En Java, las cadenas de caracteres no se les consideran como un tipo de dato.

Genera un documento, de no más de una cuartilla, donde se explique la naturaleza de estos elementos. Además se deberá incluir ejemplos para las operaciones de comparación, concatenación y modificación de cadenas.

Para enviar tu respuesta, pulsa el botón **Editar mi envío;** se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios**

Autoevaluación

Lee cuidadosamente cada oración e indica si son verdaderas o falsas. Al final obtendrás tu calificación de manera automática.

		Verdadera	Falsa
1	La asignación aritmética se puede emplear con variables de tipo booleanas.	( )	( )
2	En lenguajes tipificados, al declarar variables estas pueden almacenar cualquier tipo de dato aún cuando no corresponda al que se declaró originalmente.	( )	( )
3	Las operaciones de asignación lógica se pueden emplear con operaciones de comparación.	( )	( )
4	Las variables numéricas pueden cambiar su valor con las operaciones de asignación aritmética.	( )	( )
5	La asignación de cadenas de caracteres contempla a los espacios vacios como un caracter más.	( )	( )

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 5. Operaciones de repetición

Objetivo del tema

Identificar el funcionamiento de las operaciones de repetición y sus condiciones de funcionamiento.

Desarrollo

Un programa imperativo, normalmente realiza su tarea ejecutando repetidamente una secuencia de pasos elementales, ya que en este modelo computacional la única forma de ejecutar algo complejo es repitiendo una secuencia de instrucciones.[2]

El flujo lógico de un programa se controla a través de estructuras, que pueden ser secuenciales, repetitivas (iterativas), selectivas y de salto. Las operaciones de repetición o iterativas son implementadas en un programa de alto nivel con las sentencias *for*, *while* y *repeat*.

[2] "Paradigmas, Imperativo", 15/08/03, material en línea, disponible en: http://wilucha.com.ar/Paradigma/A_ParaImpera.html, recuperado el 28/01/09.

ACTIVIDAD 1

Elabora un cuadro comparativo sobre el uso de las siguientes estructuras de control entre C y Java:

- FOR
- WHILE
- DO .. WHILE

El cuadro comparativo deberá señalar las diferencias entre dichos lenguajes y sus particularidades.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 2

Empleando un lenguaje de programación que elijas, envía un ejemplo que represente el recorrido sobre un arreglo utilizando la estructura de control FOR.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 3

En el lenguaje de programación PHP se tiene la estructura de control FOREACH. Genera un documento donde se indique ,para la estructura anterior, los siguientes puntos:

- Sintaxis
- Operación
- Variantes
- Ejemplo

El documento no deberá ser de más de dos cuartillas

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

Autoevaluación

Elige la opción que conteste correctamente cada una de las siguientes oraciones. Al final obtendrás tu calificación de manera automática.

1. La diferencia entre una sentencia de repetición WHILE y un DO ..WHILE es:

- a) El WHILE se ejecuta al menos una vez y el DO .. WHILE no se puede garantizar que se ejecute.
- b) El DO .. WHILE se ejecuta al menos una vez y el WHILE no se puede garantizar que se ejecute.
- c) Su diferencia radica en el lenguaje de programación que se empleé.

2. El flujo lógico de la ejecución de un programa se controla por medio de:

- a) las condiciones de operación.
- b) el algoritmo que se implementa.
- c) las estructuras de control.

3. En lenguajes de alto nivel, las operaciones de repetición se pueden expresar como:

- a) Sentencias FOR, WHILE y REPEAT
- b) Sentencias IF CASE o SWITCH
- c) Sentencias de asignación tales como: =, := o ==

4. Como una solución clásica al recorrido de arreglos se emplear la sentencia:

- a) IF
- b) FOR
- c) CASE

5. La sentencia WHILE se emplea cuando:

- a) Se desea entrar en un ciclo de ejecución y se evalúa la condición antes de cada iteración.
- b) Se desea entrar en un ciclo de ejecución y se evalúa la condición después de cada iteración.
- c) Se desea entrar en un ciclo de ejecución, donde se inicializan valores y antes de cada iteración se evalúa una condición para continuar.

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

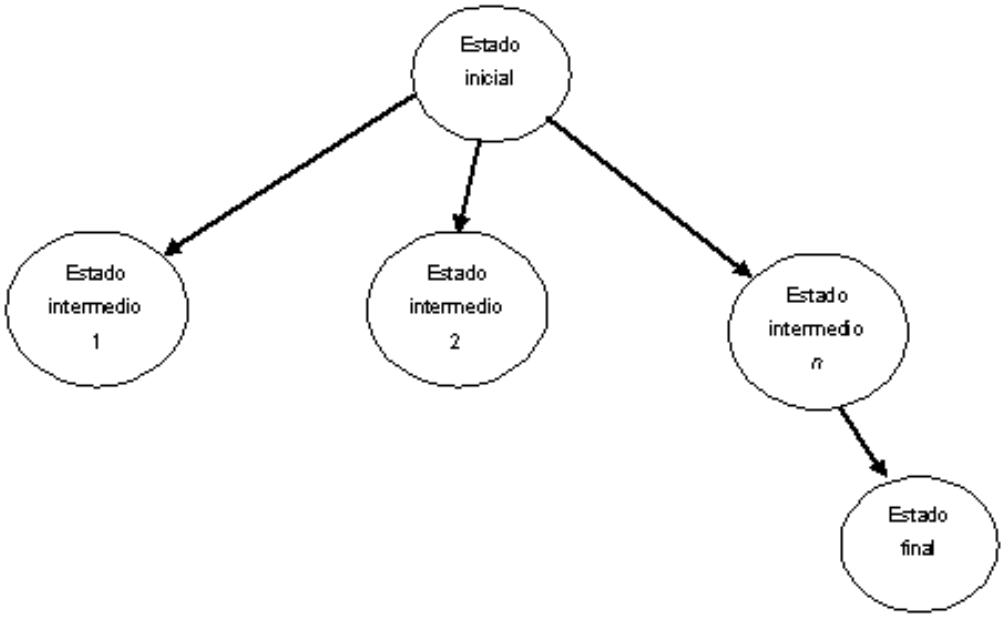
Tema 6. Secuencia de transformación de datos

Objetivo del tema

Identificar la forma en que la memoria cambia su estado al ir ejecutándose un programa.

Desarrollo

Una vez que se ha determinado qué datos son los que un programa va a necesitar, se les asocia una dirección de memoria y se efectúa una secuencia de transformaciones en los datos hasta obtener el valor esperado. Se transforman los datos desde un estado *inicial* hasta un estado *final* usando *reglas* de transformación. Se crea un grafo de estados intermedios con estas reglas y se analiza cuál es el estado intermedio más próximo al objetivo final. Ejemplo:



Representación de un grafo de estados del proceso de transformación de datos[3]

[3] Fuente: Ismael Perea: A*puntes de Programación del tercer semestre de la carrera de informática.* 2008.

ACTIVIDAD 1

Sea el algoritmo para calcula la potencia p de un número n como:

n, si p tiene el valor de 1.

Potencia (n,p) =

n * Potencia(n, p-1)

Generar un documento donde se muestren los valores intermedios y final de las siguientes expresiones:

- Potencia(5,3)
- Potencia(17,3)
- Potencia(9,5)
- Potencia(3,8)
- Potencia(7,2)
- Potencia(11,4)
- Potencia(6,9)
- Potencia(2,7)
- Potencia(13,6)

Ejemplo:

Potencia(2,3) = 2 * Potencia(2, 2)

= 2 * (2 * Potencia(2, 1)) = 2*(2 * (2 * Potencia (2, 0)))

= 2 * (2 * (2 * 1))) = 2 * (2 * (2)) = 2 * (4) = 8

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

Autoevaluación

Dada la siguientes líneas de programación en Java

```
1.  int a = 5;
2.  int b = 7;
3.  int c = a + b;
4.  a = c - 5;
5.  b = c - 1;
6.  c = 8;
7.  a = b - c;
8.  b = c + a;
9.  a = a + b;
10. b = b + 5;
11. c = c - 3;
12. a *= 8;
13. b += 2 * a;
14. c -= 5 + b;
```

Elige la opción que conteste correctamente cada una de las siguientes preguntas.

1. ¿Cuál es el valor de la variable ‘a’ en la línea 9?

- a) 11
- b) 14
- c) 16

2. ¿Cuál es el valor de la variable ‘’a en la línea 12?

- a) 112
- b) 240
- c) -240

3. ¿Cuál es el valor de la variable ‘b’ en la línea 5?

- a) 7
- b) 11
- c) 8

4. ¿Cuál es el valor de la variable ‘c’ en la línea 14?

- a) 112
- b) 240
- c) -240

5. ¿Cuál es el valor de la variable ‘b’ en la línea 10?

- a) 14
- b) 16
- c) 5

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 7. Campos de aplicación

Objetivo del tema

Identificar la utilidad de los lenguajes imperativos en la solución de problemas.

Desarrollo

Los lenguajes imperativos pueden resolver prácticamente cualquier problema en cualquier área: desde simples hasta complejos cálculos matemáticos. Se pueden hacer cualquier tipo de aplicaciones: de nóminas, de control aéreo, de inteligencia artificial, de control de dosis de medicamentos, para cajeros automáticos, para naves espaciales, para dispositivos móviles, aplicaciones en línea y en tiempo real, pago de impuestos, etc. Hay que recordar que fue el primer paradigma que le vino a poner orden a la manera de hacer programas, y por tanto su filosofía marcó la línea a seguir para resolver problemas de la vida cotidiana.

ACTIVIDAD 1

Visita la página **http://www.tiobe.com** y consulta el índice de los 20 lenguajes de programación más populares según este organismo. Revisa cuál es el paradigma que tiene más presencia en el top 20

ACTIVIDAD 2

Elabora y envía el código fuente del programa el “¡Hola Mundo!” en los siguientes lenguajes:

- C
- Java
- HTML

En cada uno de ellos, explica el significado de las instrucciones que presentas.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 3

Elabora un documento, no mayor a dos cuartillas, explicando a detalle el siguiente código en C y explica el concepto de memoria variable a partir de los valores que se obtienen:

```
//Apuntadores
#include<stdio.h>
int main(){
        int alpha=1;
        int *beta;  //el * permite una dirección de memoria como valor asignado a una variable
        alpha=1;
        beta=&alpha
        printf("El valor %d está almacenado en la dirección %u\n",alpha,&alpha);
        printf("El valor %u está almacenado en la dirección %u\n",beta,&beta);
        printf("El valor %d está almacenado en la dirección %u\n",*beta,&beta);

return 0;

}
```

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 4

Busca en la Web el algoritmo de las torres de Hanoi
Genera y envía el código fuente en C, agrupando las operaciones que definas en funciones.

Realiza tu actividad, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

Autoevaluación

Lee cuidadosamente cada oración e indica si es verdadera o falsa. Al final obtendrás tu calificación de manera automática.

		Verdadera	Falsa
1	Los lenguajes imperativos iniciaron con fines académicos y actualmente se pueden emplear para la resolución de problemas financieros.	( )	( )
2	En general, un lenguaje imperativo es más claro para los humanos que el lenguaje máquina.	( )	( )
3	Los lenguajes imperativos no son los ideales para el apoyo a la investigación	( )	( )
4	Un lenguaje imperativo para el desarrollo de sistemas de información es PHP.	( )	( )
5	Los lenguajes imperativos no pueden ser modulares.	( )	( )

Cuestionario de Autoevaluación

Contesta las siguientes preguntas.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar**. Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

1. ¿Qué es el paradigma imperativo?
2. ¿Qué es el concepto de celda de memoria variable?
3. ¿Qué son las operaciones de asignación?
4. ¿Qué son las operaciones de repetición?
5. ¿Qué es la transformación de datos?

Examen de Autoevaluación

Contesta las siguientes preguntas eligiendo la opción correcta. Al final obtendrás tu calificación de manera automática.

1. Paradigma de programación que se caracteriza por un modelo abstracto de la computadora que consiste en un almacenamiento en memoria de datos y cálculos codificados y la ejecución de una secuencia de comandos que modifican el contenido de ese almacenamiento:

- a) lógico
- b) algorítmico
- c) orientado a objetos
- d) funcional

2. Son los tres conceptos principales del paradigma imperativo:

a) int x;	|	if (x==0)	|	x=0
b) var x: integer;	|	while (x >0)	|	x=3
c) float x;	|	return x	|	x=3+8
d) boolean x;	|	switch (true)	|	x=true

3. Los tres tipos de asignación son:

- a) aritmética, lógica y por parámetros
- b) de cadena de caracteres, lógica y funcional
- c) funcional, de cadena de caracteres y aritmética
- d) lógica, de cadena de caracteres y aritmética

4. Son estructuras de control iterativas:

- a) break, continue y return
- b) if, switch y case
- c) for, while y repeat
- d) if, while y goto

5. Para guiar los cálculos o flujo de un programa (transformación de los datos a partir de un algoritmo) se utilizan cálculos o flujo de un programa de tipo:

- a) secuenciales, selectivas y de salto
- b) de salto, repetitivas y secuenciales
- c) repetitivas, selectivas y de salto
- d) selectivas, secuenciales e iterativas

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

LO QUE APRENDÍ

Lea el artículo ‘Go To Statement Considered Harmful’ de Edsger W. Dijkstra **(ANEXO 3).** Elabore un documento, de no mayor a dos cuartillas, indicando los puntos centrales de la lectura y una crítica al mismo artículo.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

Glosario de la unidad

Programación imperativa

Es un paradigma de programación que describe la programación en términos del estado del programa y sentencias que cambian dicho estado. También es llamado procedimental o algorítmico. Se llama así porque está basado en comandos u órdenes (enunciados imperativos) que actualizan variables que están almacenadas en memoria.

Celda de memoria

Un bit de memoria. En la memoria RAM dinámica, una celda está compuesta por un transistor y un condensador. En la memoria RAM estática, una celda está compuesta por alrededor de cinco transistores.

Variables

Son estructuras de datos que, como su nombre indica, pueden cambiar de contenido a lo largo de la ejecución de un programa. Una variable corresponde a un área reservada en la memoria principal del ordenador pudiendo ser de longitud fija o variable.

Operadores de asignación

Son operadores definidos en los lenguajes y se utilizan para cambiar (asignar) valores a las variables. Se pueden emplear para guardar el cómputo de otras expresiones.

Sentencias de repetición

Como su propio nombre indica permiten que un conjunto de sentencias sean ejecutadas un cierto número de veces.

MESOGRAFÍA

Bibliografía básica

Bibliografía complementaria

Sitios electrónicos

(ANEXO 1) Descarga el artículo de Dra. Oktaba, Lenguajes de programación **que se encuentra en la plataforma**

(ANEXO 2) Descarga la lectura de la plataforma de la Dra. Oktaba **Oktaba Lenguajes de programación**

(ANEXO 3) Descarga el artículo 'Go To Statement Considered Harmful' de Edsger W. Dijkstra  **PDF que se encuentra en la plataforma**