Carátula para entrega de prácticas	
Facultad de Ingeniería	Laboratorios de docencia

Laboratorio de Computación Salas A y B

Profesor: García Morales Karina

Asignatura: Fundamentos de Programación

Grupo: 1121

No de Práctica(s): 3

Integrante(s): Mena Hernández Hebby Renan

No. de Equipo de cómputo empleado: 1

Semestre: 2019-1

Fecha de entrega: 29-Agosto-2018

Observaciones:

CALIFICACIÓN: _____

Título de la práctica

Guía práctica de estudio 03: Solución de problemas y Algoritmos.

Objetivo

Elaborar algoritmos correctos y eficientes en la solución de problemas siguiendo las etapas de Análisis y Diseño pertenecientes al Ciclo de vida del software.

Desarrollo de la Práctica

Un problema informático se puede definir como el conjunto de instancias al cual corresponde un conjunto de soluciones, junto con una relación que asocia para cada instancia del problema un subconjunto de soluciones (posiblemente vacío).
Para poder solucionar un problema nos apoyamos en la Ingeniería de Software que deacuerdo a la IEEE se define como “La aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software".
Por lo que el uso y establecimiento de principios de ingeniería sólidos, son básicos para obtener un software que sea económicamente fiable y funcione eficientemente.
La Ingeniería de Software provee métodos que indican cómo generar software. Estos métodos abarcan una amplia gama de tareas:

Planeación y estimación del proyecto.
Análisis de requerimientos del sistema y software.
Diseño de la estructura de datos, la arquitectura del programa y el procedimiento algorítmico.
Codificación.
Pruebas y mantenimiento (validación y verificación).

Ciclo de vida del software

La ISO (International Organization for Standarization) en su norma 12207 define al ciclo de vida de un software como:
Un marco de referencia que contiene las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando desde la definición hasta la finalización de su uso.

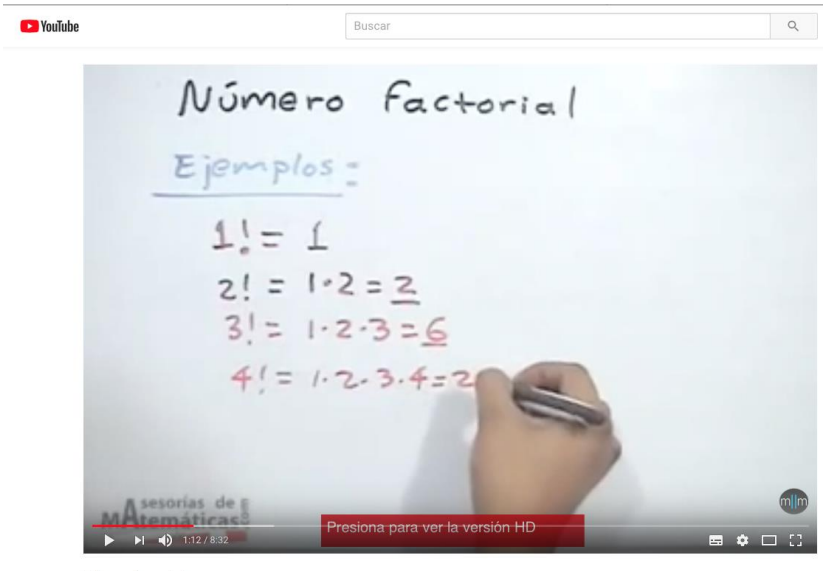
Solución de problemas

Dentro del ciclo de vida del software, en el análisis se busca comprender la necesidad, es decir, entender el problema.
El análisis es el proceso para averiguar qué es lo que requiere el usuario del sistema de software (análisis de requisitos). Esta etapa permite definir las necesidades de forma clara y concisa (especificación de requisitos).
Por lo tanto, la etapa del análisis consiste en conocer qué es lo que está solicitando el usuario. Para ello es importante identificar dos grandes conjuntos dentro del sistema: el conjunto de entrada y el conjunto de salida.

El conjunto de entrada está compuesto por todos aquellos datos que pueden alimentar al sistema.
El conjunto de salida está compuesto por todos los datos que el sistema regresará como resultado del proceso. Estos datos se obtienen a partir de los datos de entrada.
La unión del conjunto de entrada y el conjunto de salida forman lo que se conoce como el dominio del problema, es decir, los valores que el problema puede manejar.
La etapa de análisis es crucial para la creación de un software de calidad, ya que si no se entiende qué es lo que se desea realizar, no se puede generar una solución. Sin embargo, es común caer en ambigüedades debido al mal entendimiento de los requerimientos iniciales.

Desarrollo de factorial
Al número deseado para sacar su número factorial se debe multiplicar todos sus antecesores.
Por ejemplo factorial de 5!. Se multiplicará 5x4x3x2x1= 120.
Video ilustrativo (link)
https://www.youtube.com/watch?v=zNq6ifSsw3k

$$n! = n \cdot (n-1) \cdots 2 \cdot 1$$



¿Que es la teoría de computabilidad?
Es la parte de la computación que estudia los problemas de decisión que pueden ser resueltos con un algoritmo.
La Teoría de la Computabilidad es el estudio matemático de los modelos de computación. Como tal estudio teórico, se originó en la década de los años 30 con los trabajos de los lógicos Church, Gödel, Kleene, Post y Turing.

La teoría de la computabilidad se interesa por cuatro preguntas:

¿Qué problemas puede resolver una máquina de Turing?

¿Qué otros formalismos equivalen a las máquinas de Turing?

¿Qué problemas requieren máquinas más poderosas?

¿Qué problemas requieren máquinas menos poderosas?

Algoritmo

Se denomina algoritmo a un grupo finito de operaciones organizadas de manera lógica y ordenada que permite solucionar un determinado problema. Se trata de una serie de instrucciones o reglas establecidas que, por medio de una sucesión de pasos, permiten arribar a un resultado o solución.

Según los expertos en matemática, los algoritmos permiten trabajar a partir de un estado básico o inicial y, tras seguir los pasos propuestos, llegar a una solución. Cabe resaltar que, si bien los algoritmos suelen estar asociados al ámbito matemático (ya que permiten, por citar casos concretos, averiguar el cociente entre un par de dígitos o determinar cuál es el máximo común divisor entre dos cifras pertenecientes al grupo de los enteros), aunque no siempre implican la presencia de números.

El término suele ser señalado como el número fijo de pasos necesarios para transformar información de entrada (un problema) en una salida (su solución).

Las principales características con las que debe cumplir un algoritmo son:

Preciso: Debe indicar el orden de realización de paso y no puede tener ambigüedad

Definido: Si se sigue dos veces o más se obtiene el mismo resultado.

Finito: Tiene fin, es decir tiene un número determinado de pasos.

Correcto: Cumplir con el objetivo.

Debe tener al menos una salida y esta debe de ser perceptible

Debe ser sencillo y legible

Eficiente: Realizarlo en el menor tiempo posible

Eficaz: Que produzca el efecto esperado

Construcción de algoritmo ejemplo 1

1 Inicio

2 Leer x si x=0 regresa al número 1 si no ir a 3

3 Si x!= 0 pasar al paso 4

4 Leer el valor de x y pasar a 5

5 Si x es mayor a cero se dice que “es positivo” y pasar a 7 si no pasar a 6

6 Se dice que x “es negativo” y pasar a 7

7 Fin

-

Itineración	Dato de entrada	Salida
1	0	-
2	5	Que es positivo
3	-2	Que es negativo

Construcción de algoritmo ejemplo 2

1 Inicio

2 Leer x

3 Leer y

4 Si y es igual a x, regresar al paso 3 si no ir a 5

5 Si x>y entonces se afirma que x es el “primer número es mayor” e ir a 7 si no ir a 6

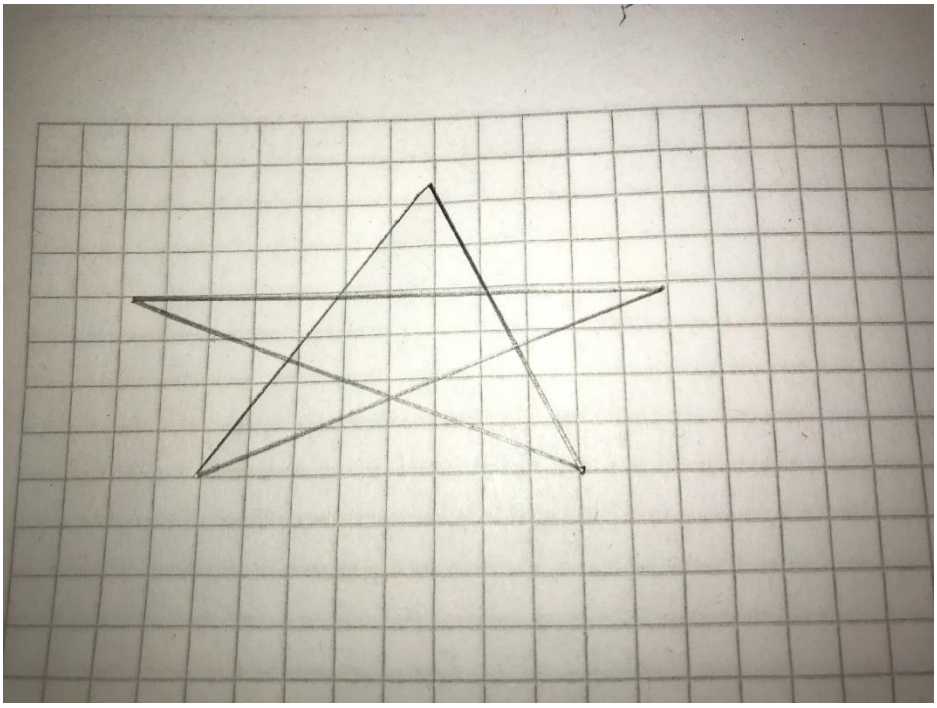
6 Si x<y entonces se afirma que y es el “segundo número es mayor” e ir a 7

7 Fin

Itineración	x Dato de entrada (1)	y Dato de entrada (2)	Salida
1	4	4	-
2	8	5	Primer número es mayor
3	5	8	Segundo número es mayor

Algoritmo

1. Dibuja una V invertida. Empieza desde el lado izquierdo, sube, y baja hacia el lado derecho, no levantes el lápiz.
2. Ahora dibuja una línea en ángulo ascendente hacia la izquierda. Debe cruzar la
3. primera línea más o menos a 1/3 de la altura. Todavía no levantes el lápiz del papel.
4. Ahora, dibuja una línea horizontal hacia la derecha. Debe cruzar la V invertida más o menos a 2/3 de la altura total. Sigue sin levantar el lápiz.
5. Dibuja una línea en un ángulo descendente hasta el punto de inicio. Las líneas deben unirse.
6. Ahora ya puedes levantar el lápiz del papel. Has terminado la estrella de 5 puntas.



Algoritmo

Empieza dibujando un círculo con un compás. Coloca un lápiz en el compás. Coloca la punta del compás en el centro de una hoja de papel.

Ahora gira el compás, mientras mantienes la punta apoyada en el papel. El lápiz dibujará un círculo perfecto alrededor de la punta del compás.

Marca un punto en la parte superior del círculo con el lápiz. Ahora, coloca la punta del compás en la marca. No cambies el radio del compás con que hiciste el círculo.

Gira el compás para hacer una marca en el propio círculo hacia la izquierda. Haz una marca también en el lado derecho.

Ahora, coloca la punta del compás en uno de los puntos. Recuerda no cambiar el radio del compás. Haz otra marca en el círculo.

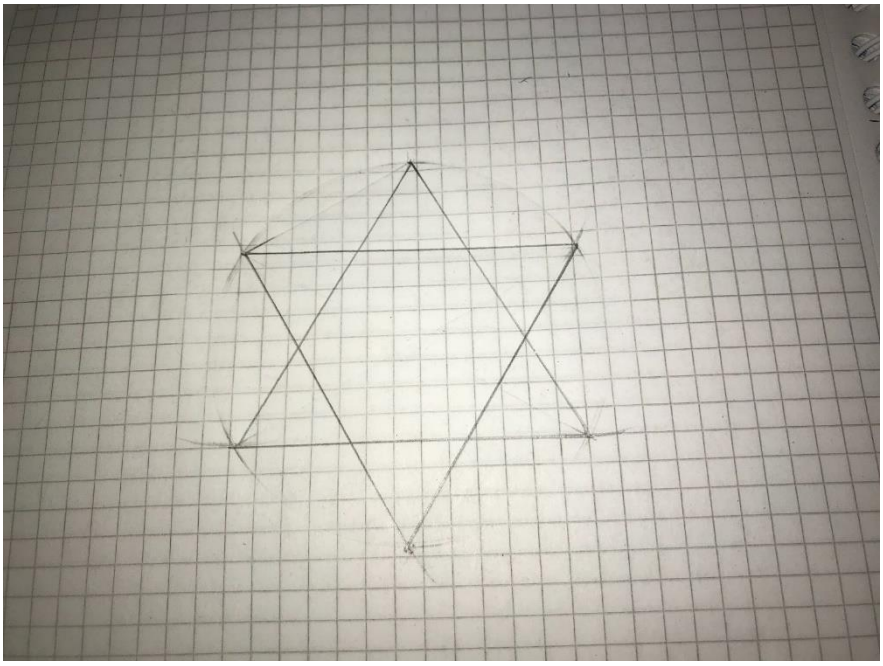
Continúa moviendo la punta del compás a las otras marcas, y continúa hasta que tengas 6 marcas a la misma distancia unas de otras. Ahora, ya puedes dejar tu compás a un lado.

Usa una regla para crear un triángulo que empiece en la marca superior del círculo.

Coloca el lápiz en la marca superior. Ahora dibuja una línea hasta la segunda marca por la izquierda. Dibuja otra línea, ahora hacia la derecha, saltándote la marca de la parte más baja. Complementa el triángulo con una línea hacia la marca superior. Así completarás el triángulo.

Crea un segundo triángulo empezando en la marca en la base del círculo. Coloca el lápiz en la marca inferior. Ahora conéctala con la segunda marca hacia la izquierda. Dibuja una línea recta hacia la derecha, saltándote el punto superior. Completa el segundo triángulo dibujando una línea hasta la marca en la parte inferior.

Borra el círculo. Has terminado de dibujar tu estrella de 6 puntos.



Ejercicios de Tarea

1 Ejercicio de factorial

Análisis Entrada: Un valor “n”
Restricción: n>0
Salida: El factorial de n

Algoritmo
1 Inicio.
2 Escribir n
2.1 Leer n
3 Si n>0 crear contador c=1 si no regresar a paso 2
4 Crear valor v=n
5 Si c<v pasar a 6
6 Se multiplica el valor de c con el siguiente hasta que c<=v almacenar en r
7 C=(c+1) hasta que c<=v

Itineración	N	R	Contador	Salida
1	-1		1	--
2	3	6	3	6
3	5	120	5	120

2 Calcular el volumen de un cilindro a partir del radio de la base y la altura. (Hacer uso de la fórmula V r h2 = π).

Análisis
Entrada radio=r, base=b altura=h
Salida Volumen (V)

Algoritmo
1 Inicio.
2 Definir r,h,V como enteros
3 Escribir r,,h
4 Leer r,h
5 V=((r*r)*(3.141592)*(h)
6 Imprime V
7 Fin

Itineración	r	h	Salida V
1	2	4	50.26
2	3	6	169.6
3	5	2	157.07

3 Calcular la distancia entre dos puntos. (Sea P1 (a1, b1) y P2 (a2, b2), hacer uso de

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Análisis
Entrada: P1(a1,b1) P2(a2,b2)
Salida: D (Distancia entre puntos)

Algoritmo

1. Inicio
2. Se dará el P1 y P2
3. Definir a1, b1, a2, b2, D como real
4. Escribir a1, b1, a2, b2
5. Leer a1, b1, a2, b2
6. D= sqrt((a1-a2)^2+(b1-b2)^2)
7. Imprime D
8. Fin

Itineración	A1	A2	B1	B2	Salida D
1	5	1	-4	6	10.77
2	4	7	1	5	5
3	2	-4	5	-3	10

4 Leer 2 números y verificar si son divisibles, o el resultado no existe, o es infinito. (Considere que los números deben ser enteros)

Análisis
Entrada: x,y (Dos números)
Restricciones números enteros
Salida: Saber si es divisible

Algoritmo

1. Inicio
2. Definir x, y como entero
3. Leer x,y
4. Si x MOD y=0 Entonces ir a 5 si no ir a 6
5. Imprimir “x es divisible entre y”
6. Imprimir “No es divisible”
7. Fin

Itineración	x	Y	Salida
1	2	4	No es divisible
2	6	3	x es divisible entre y
3	5	2	No es divisible

5 Leer un número y verificar si un número es par o impar.

Análisis:
Entrada: Un número (n)

Salida: “Es par” o “Es impar”

Algoritmo

1. Inicio
2. Definir n como entero
3. Escribir n
4. Leer n
5. Si nMOD2=0 si ir a 6 si no ir a 7
6. Imprimir “Es par” ir a 8
7. Imprimir “Es impar” ir a 8
8. Fin

Itineración	n	Salida
1	2	Es par
2	6	Es par
3	5	Es impar

5 Leer del numero 1 al 50 e indicar cuales números son múltiplos de 3

Análisis

Entrada: (1-50)

Salida: Indicar “Múltiplo de 3”

```
Proceso sin_titulo
    x<-1
    Mientras x<=50 Hacer
        Si (x MOD 3)=0 Entonces
            Escribir x
        FinSi
        x<-x+1
    FinMientras
FinProceso
```

Algoritmo

1. Inicio
2. X=1
3. Mientras x<=50 Hacer
4. Si (x MOD 3)=0 Entonces
5. Escribir x
6. Fin si
7. X=x+1
8. Fin mientras Fin

Itineración	x	Salida
1	3	3
2	6	6
3	10	-----

Conclusiones

La práctica me ayudo a recordar conceptos y reafirmar las bases para la resolución de problemas.

Puso en práctica la habilidad para desarrollar problemas tanto de manera digital como física al utilizar el compás y tener que seguir un algoritmo para realizar una acción.

Al aprender a utilizar las estructuras de control como los ciclos simplifica el trabajo y nos ayuda a realizarlo de una manera más eficaz y nos ayuda a entender mejor el proceso utilizando a la par los contadores que son la base para que un ciclo avance.

Bibliografía

Raghu Singh (1995). International Standard ISO/IEC 12207 Software Life Cycle

Processes. Agosto 23 de 1996, de ISO/IEC. Consulta: Junio de 2015. Disponible en:

http://www.abelia.com/docs/12207cpt.pdf

 Carlos Guadalupe (2013). Aseguramiento de la calidad del software (SQA). [Figura

1].Consulta: Junio de 2015. Disponible en:

https://www.mindmeister.com/es/273953719/aseguramiento-de-la-calidad

delsoftware-sqa

 Andrea S. (2014). Ingeniería de Software. [Figura 2]. Consulta: Junio de 2015.

Disponible en: http://ing-software-verano2014.blogspot.mx