	Carátula para entrega de prácticas	
Facultad de Ingeniería		Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Adrian Ulises Mercado Martínez

Asignatura: Fundamentos de programación

Grupo: 07

No de Práctica(s): 03

Integrante(s): Arredondo Granados Gerardo
Casillas Herrera Leonardo Didier
Diaz Gonzalez Rivas Angel Iñaqui
Galván Castro Martha Selene

No. de Equipo de cómputo empleado:

No. de Lista o Brigada: 04

Semestre: 2022-1

Fecha de entrega: 10 de octubre del 2021

Observaciones:

CALIFICACIÓN: _____

Práctica 3
Solución de problemas y algoritmos

ÍNDICE

Objetivo:

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

Introducción:

Dentro del mundo de la informática nos estaremos enfrentando diariamente con problemas para resolver, ya sea estando en el trabajo o en la escuela, lógicamente estos problemas se tienen que solucionar correctamente para que nosotros podamos continuar con nuestras actividades; gracias a la Ingeniería de Software, nosotros podemos resolver estos problemas.
Nosotros como Ingenieros en Computación, nos estaremos enfrentando constantemente a problemas relacionados con el Software, es aquí cuando la Ingeniería de Software nos es de ayuda, ya que esta nos provee métodos con los cuales nosotros podemos solucionar estos problemas.
Ciclo de vida del software:
Este llamado ciclo de vida, son los “pasos” que se tienen que seguir para el desarrollo, explotación y el mantenimiento de un proyecto de software, este ciclo está mejor representado como lo que es, un ciclo, lo mostramos a continuación:

Figura 1: Ciclo de vida del software.

Solución de problemas:
Como pudimos ver anteriormente, el primer paso para el desarrollo de un proyecto de software, es la definición de las necesidades, con esto buscamos qué es lo que queremos que suceda, dentro de este primer paso nosotros entendemos el problema.
Ésta siendo una etapa de análisis, se trata de conocer qué es lo que se le está solicitando al usuario, los datos con los que trabajaremos, qué se hará con esos datos, y cuáles son los datos que nos devolverán, a estos los conocemos como los **conjuntos de entrada** y **conjuntos de salida.** El conjunto de entrada es todo lo que nosotros le daremos al programa, mientras que el conjunto de salida, es lo que el programa nos devolverá después de ejecutarse. Todo esto se puede ejemplificar mucho mejor con la siguiente imagen.

Figura 2. Sistema

Algoritmos:
Entramos a los algoritmos una vez que entendimos completamente el problema, qué es lo que nos piden y qué es lo que necesitamos, es aquí cuando nos adentramos en diseñar la solución que habíamos planeado inicialmente, esto es, como es de esperarse, el algoritmo.
Un algoritmo se define como un conjunto de reglas y estas son utilizadas para realizar una tarea en específico, nuestro problema planteado.
Existen reglas específicas que los algoritmos deben cumplir, estas reglas son:

- Debe ser preciso: Los pasos del algoritmo deben de seguir un orden especificado y éste no puede tener ambigüedad
- Debe ser definido: Sean n número de veces que se ejecute, debe de llegar al mismo resultado todas las veces
- Debe ser finito: El algoritmo en algún momento tiene que acabar.
- Debe ser correcto: El algoritmo debe de llegar al resultado que se estaba esperando todas las veces
- Debe ser sencillo y sobre todo legible para la mayoría de personas

Existe una metodología para la elaboración de algoritmos, esto para elaborarlos de la manera más correcta posible y más rápida posible. Esta metodología es la siguiente:

- Resultados del análisis del problema: Para empezar a construir el algoritmo debemos de saber con certeza qué es lo que estamos realizando, es aquí donde los resultados del análisis entran en juego, con estos análisis tendremos bien en claro qué algoritmo estaremos diseñando.
- Construcción del algoritmo: Una vez que tengamos claro en mente qué es lo que nuestro algoritmo debe de contener, empezamos con la construcción del mismo, siguiendo un orden específico.
- Verificación del algoritmo: Una vez que se construyó el algoritmo llega el tiempo de verificar que esté correcto, que cumpla con todas las características que debe de tener, que se llegue al resultado esperado, etc.

Dentro de los algoritmos existen módulos, hay 3 módulos básicos, estos son los siguientes:

- Módulo de entrada: Estos son los datos que son requeridos para la resolución del problema, los datos que nosotros ingresamos para que el programa pueda funcionar correctamente.
- Módulo de procesamiento: Aquí irán las operaciones necesarias para que el algoritmo se ejecute.
- Módulo de salida: Aquí es donde se muestran los resultados en sí del algoritmo, si el algoritmo es correcto, la respuesta que nos dará será la que nosotros esperábamos.

Variables:

Los algoritmos necesitan un lugar en donde almacenar la información que nosotros le estaremos dando, estas son las llamadas variables, las variables pueden almacenar un valor numérico o no numérico. Gracias a estas variables se le permite fluir al algoritmo correctamente.

Desarrollo:

Ejercicios:

- Construye un algoritmo que reciba tres números reales, identifique cuál es el mayor. Considere que los números pueden ser iguales.

Restricción: que el número sea real y entero
Datos de entrada:
3 números reales
Datos de salida:
El número mayor
Dominio:
Todos los números enteros

1. Solicitar un número real y almacenarlo en una variable.
2. Solicitar otro número real y almacenarlo en otra variable.
3. Solicitar otro número real y almacenarlo en otra variable.
4. Hacer una condición if, si el primer número es mayor al segundo y tercero, imprimir que el número 1 es mayor.
5. Si no se cumple esto, abrir otra condición if en el else del primer if.
6. En este if se validará si el número 2 es mayor al número 3.
7. Si esto es verdad, se imprimirá el número 2 es mayor.
8. Si no se cumple, llegamos al segundo else.
9. Se imprimirá el número 3 es mayor.

Iteración	x	y	z	Salida
1	24	24	24	los tres números son iguales

Iteración	x	y	z	Salida
1	5000	8	76	el número 1 es mayor

Iteración	x	y	z	Salida
1	347	84482	848	el número 2 es mayor

- Elabore un algoritmo que permita imprimir de forma separada los cuatro dígitos que forman un número. **Sugerencia:** Utilice la división entera y el operador de módulo.

Datos de entrada:
Número entero de 4 dígitos
Datos de salida:
4 dígitos que formen un número
Dominio:
Números enteros mayores a 999 y menores a 10 000

1. Solicitar un número de 4 dígitos
   1.1 Si el número entero no tiene 4 cifras, vuelve al paso 1
2. Almacenar este número en una variable
3. Separar el número en millares, centenas, decenas, unidades.
4. Imprimir los dígitos formados con separación en el orden millar, centena, decena y unidad.

Iteración	X	Salida
1	1245	1 millares 2 centenas 4 decenas 5 unidades

Iteración	X	Salida
1	8452	8 millares 4 centenas 5 decenas 2 unidades

Iteración	X	Salida
1	1	Escribe un número válido
2	215425	Escribe un número válido
3	9651	9 millares 6 centenas 5 decenas 1 unidad

- Elabore un algoritmo para determinar los divisores de un número.

Restricción: que el número no sea negativo o igual a cero
Datos de entrada:
Un número natural que no sea 0
Datos de salida:
Divisores del número entero

Dominio:
Todos los números naturales excepto el 0

1. Solicitar al usuario un número natural que no sea 0 y guardarlo como una variable..
   1.1 Si el número entero es menor a cero, vuelve al paso 1.
2. Dividir el número entre valores menores o iguales al mismo, tales que su residuo sea igual al número cero
3. Todos los números con los que el residuo NO fue igual a cero, descartarlos
4. Imprimir los números para los cuales el residuo de la división es igual a cero

Iteración	X	Salida
1	12	El número es divisible entre 1,2,3,4,6,12

Iteración	X	Salida
1	20	El número es divisible entre 1,2,4,5,10,20

Iteración	X	Salida
1	0	Escribe un número válido
2	-5	Escribe un número válido
3	3	El número es divisible entre 1,3

Ejercicios:

- Se dice que un número N es primo si los únicos enteros positivos que lo dividen son exactamente 1 y N. Elabore el algoritmo que de solución al problema.

Restricción: en número de entrada debe de ser entero y positivo
Datos de entrada :
Un número entero
Datos de salida :
El número es primo o no lo es
Dominio:
Los números naturales de (0, infinito)

1. Solicitar al usuario un número entero positivo
   1.1 Si el número entero es menor a cero o igual a este, volver al paso 1
2. Almacenar ese número en una variable de tipo entero
3. Empezar a dividir el número empezando con 1 número uno, siguiendo del número 2, siguiendo del número 3, y así sucesivamente hasta alcanzar el mismo número que ha sido ingresado
4. El conjunto de números que arroje como residuo 0, deberán ser el número 1, y el mismo número ingresado
5. Si es que se cumple el paso 4, mostrar en pantalla “El número es primo”
   5.1 De lo contrario, mostrar en pantalla “El número no es primo”

Iteración	X	Salida
1	7	El número es primo

Iteración	X	Salida
1	9	El número no es primo

Iteración	X	Salida
1	0	Ingrese un numero valido
2	3	El número es primo

- Elaborar un algoritmo que reciba un número entero y calcule los números perfectos que existen entre 1 y el número seleccionado. Un número se considera perfecto si la suma de todos sus divisores son igual a dicho número.

Restricción:
El número debe ser entero y mayor a 1
Datos de entrada:
Un número entero
Datos de salida :
Los números perfectos entre el 1 y el número seleccionado
Dominio:
Todos los números enteros empezando por el 1

1. Solicitar al usuario un número entero positivo
   1.1 Si el número entero es menor a cero, vuelve al paso 1.
2. Almacenar el número en una variable de tipo entero.
3. Dividir el número entre todos los números enteros positivos menores a él.
4. Sumar todos los divisores encontrados.
5. Comparar el resultado de la suma con el número dado, si el resultado fue igual al número dado entonces es un número perfecto, si el resultado fue diferente al número dado entonces no es un número perfecto.

Iteración	X	Salida
1	10	Entre 1 y 10, existe un numero perfecto que es 6

Iteración	X	Salida
1	7500	Entre 1 y 7500, existen 3 numeros perfectos que son 6, 28, 496

Iteración	X	Salida
1	0	Ingresa un numero valido
2	-5	Ingresa un numero valido
3	50	Entre 1 y 50, existen 2 numeros perfectos que son 6 y 28

Conclusiones

Conclusión grupal: Creemos que se cumplieron los objetivos en esta práctica ya que se lograron diseñar los algoritmos correctamente, cabe aclarar que para la elaboración de estos mismos se estuvo siguiendo el camino marcado por la práctica, éste siendo primero el análisis del problema, aquí es donde nosotros nos pusimos a pensar qué es lo que el problema requería, después comenzamos con el diseño del mismo y se concluyó con las pruebas de escritorio.

Conclusiones individuales

Arredondo Granados Gerardo: Creo que fue una práctica interesante, más por los problemas planteados ya que algunos de estos requerían un mayor tiempo de análisis, tal como eran los objetivos de la práctica, se aprendió a hacer un algoritmo funcional, se utilizaron adecuadamente las etapas de la creación de este, como el análisis, fue aquí donde pensamos qué es lo que el algoritmo nos estaba pidiendo, el diseño, que fue en sí diseñar el algoritmo, y al final las pruebas.

Casillas Herrera Leonardo Didier: Fue una práctica muy útil en la que se cumplieron todos los objetivos dados ya que pudimos elaborar nuestros propios algoritmos para resolver los problemas que se nos plantearon, analizando los problemas que se nos dieron y diseñando un algoritmo que nos permitiera resolver ese problema, cumpliendo así con el ciclo de vida del software.

Diaz Gonzalez Rivas Angel Iñaqui: En esta práctica se demostró que para solucionar un problema se requiere hacer una análisis y diseño del ciclo de vida del software, después de esto, se puede empezar a realizar el algoritmo que permita solucionar el problema planteado. De la misma manera se podría decir que es importante hacer las pruebas para comprobar que funcione de la manera correcta y pueda dar una solución eficiente.

Galván Castro Martha Selene: En Conclusión se puede decir que durante el desarrollo de la práctica aprendimos cómo realizar algoritmos con diversos problemas matemáticos a resolver y realizamos pruebas de escritorio para validar los algoritmos y su eficacia. Además como eran tan diversos los problemas adquirimos práctica ya que tiene soluciones distintas con esto se apreció la importancia de crear algoritmos correctos por lo cual se puede decir que se cumplieron los objetivos de la práctica.

Referencias

- 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
- Michael Littman. (2012). Intro to Algorithms: Social Network Analysis. Consulta Junio de 2015, de Udacity. Disponible en: https://www.udacity.com/course/viewer#!/c-cs215/l-48747095/m-48691609