ESCUELA NACIONAL DE ESTUDIOS SUPERIORES UNIDAD JURIQUILLA (UNAM)

EQUIPO VI:

Pérez Hernández Diego

42506183-1

García Alcaraz Oscar José

42502913-4

Hernández Manzanilla Pablo César

42502786-2

Ingeniería Aeroespacial

Grupo 2002 (Semestre 2)

A 14 de Febrero del 2025

SOLUCIÓN DE PROBLEMAS Y ALGORITMOS DE PROGRAMACION

PRACTICA II

Perteneciente a la materia de:
Fundamentos de programación

Docente:
Ing. Andrés David Flores Ferro

Fecha De Revisión: 02 / 03 / 2025.

Calificación: 88 de 100 pts .

Observaciones:

- Sin contenido de clase y por lo tanto no se conecta bien investigación.
- Agregar dato de tipo de archivo consultado en referencias [Web,PDF, Libro, etc] , no se baja por esta razón si no porque falta citar algo del contenido presentado en clase.
- En el desarrollo solo se menciona el contenido de uno de los ejercicios y en el otro si se desarrolla que es cada etapa pero no como se hace IDEALMENTE se queria descripcion como si fuese un paso a paso y la teoria detras mas sus correcciones.

Actividad en clase 23 de 25 pts	
Protocolo de práctica	**65 de 75 pts**
Portada	10 de 10 pts
Introducción	13 de 15 pts
Desarrollo	19 de 25 pts
Conclusiones	15 de 15 pts
Referencias	08 de 10 pts
CALIFICACIÓN TOTAL	**Entrega 88 pts**

SOLUCIÓN DE PROBLEMAS Y ALGORITMOS

Practica II

Fundamentos De Programación

ENES

JURIQUILLA

Introducción:

Dentro del ámbito de la informática, los algoritmos representan una herramienta práctica para resolver problemas complejos de manera ordenada y eficiente. Un algoritmo informático es un conjunto de instrucciones definidas, ordenadas y acotadas para resolver un problema, realizar un cálculo o desarrollar una tarea. Es decir, un algoritmo es un procedimiento paso a paso para conseguir un fin. A partir de un estado e información iniciales, se siguen una serie de pasos ordenados para llegar a la solución de una situación.

Un algoritmo supone el paso previo a ponerse a escribir el código. Primero debemos encontrar la forma de obtener la solución al problema (definir el algoritmo informático), para luego, a través del código, poder indicarle a la máquina qué acciones queremos que lleve a cabo. De este modo, un programa informático no sería más que un conjunto de algoritmos ordenados y codificados en un lenguaje de programación para poder ser ejecutados en un ordenador.

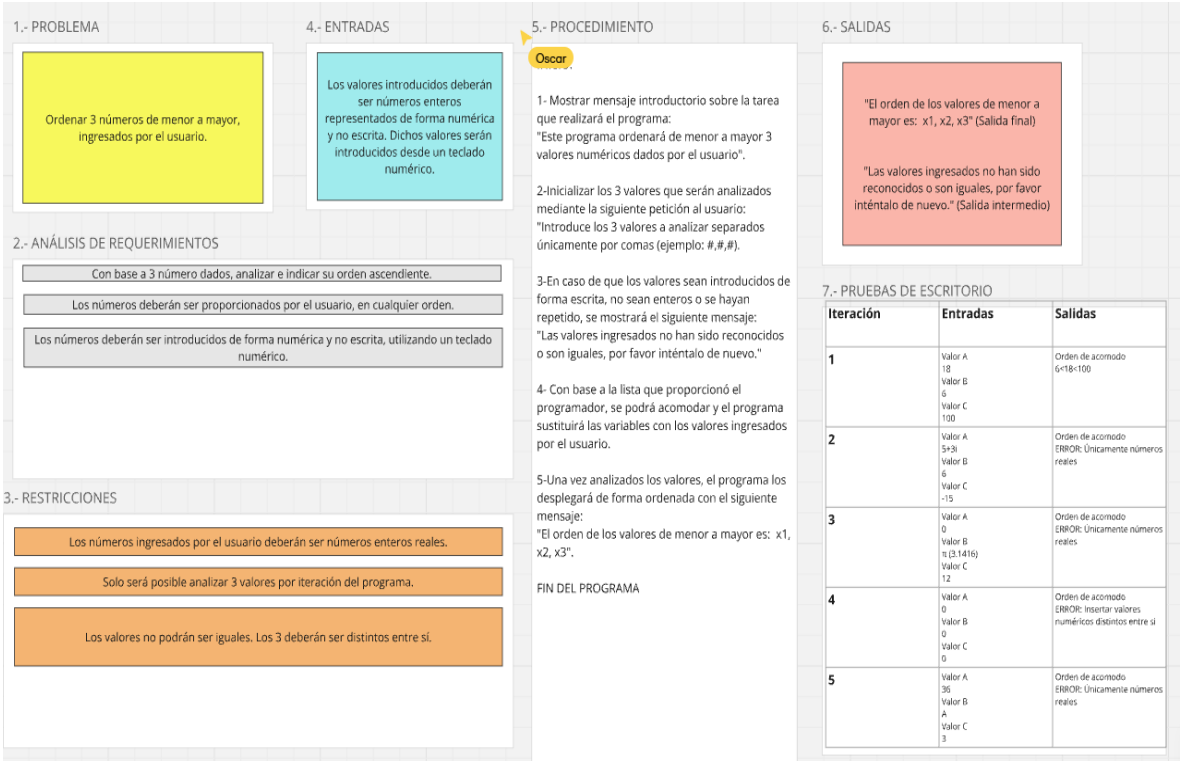
Un problema informático se entiende, en términos exactos, como la asociación entre un conjunto de instancias (o entradas) y sus posibles soluciones, donde para cada instancia puede existir un conjunto (posiblemente vacío) de respuestas válidas.

Para alcanzar estas soluciones de manera sistemática, es importante contar con la disciplina y los métodos propios de la Ingeniería de Software. Según la IEEE, esta se define como “la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software”. Dicho enfoque no solo asegura que el software sea económicamente viable y funcione eficientemente, sino que también establece una serie de fases (como la planeación, el análisis de requerimientos, el diseño, la codificación y las pruebas), las cuales garantizan la calidad y la robustez del producto final.

El proceso de solución de problemas en informática se basa en un profundo análisis de las necesidades del usuario al que llegará el programa y la identificación de las estructuras de datos y algoritmos más apropiados para abordarlas. Este enfoque metodológico, sustentado en principios matemáticos y de ingeniería, posibilita la generación de software que se adapta a contextos reales, desde simples aplicaciones hasta sistemas complejos y críticos, permitiendo además la integración de técnicas avanzadas como el aprendizaje automático para optimizar la toma de decisiones.

Desarrollo:

Proceso para el problema 1 (Ordenamiento de números de menor a mayor)



1. PROBLEMA:

Elaboración de un algoritmo de programación que sea capaz de ordenar de menor a mayor tres números enteros reales dados por el usuario que opere dicho programa.

2. ANÁLISIS DE REQUERIMIENTOS:

Con base a tres números proporcionados por el usuario, el programa deberá seguir un algoritmo que, mediante su ejecución, deberá indicar el orden de menor a mayor de los valores proporcionados por el usuario.

Los valores deberán ser proporcionados al programa por el usuario en cualquier orden en el momento que el programa lo indique.

Los valores deberán ser introducidos mediante un teclado numérico, esto para evitar la introducción de caracteres alfanuméricos que den como resultado una ejecución incorrecta del programa.

3. RESTRICCIONES:

Aunado al punto anterior, el algoritmo únicamente procederá con la lectura de valores que cumplan con los siguientes requerimientos:

1. El algoritmo/programa únicamente trabajara con valores numéricos pertenecientes al sistema de numeración decimal.
2. Como extensión del punto anterior, los teclados numéricos pertenecientes a otros sistemas de numeración (Binario, hexadecimal, romano u octal, si es que llegasen a existir) quedaran descartados para la ejecución del programa.
3. Pertenecer al conjunto de los números reales
4. Pertenecer al conjunto de los números enteros
5. Únicamente será posible el análisis de tres valores por iteración del programa
6. Los valores introducidos para su posterior comparación no podrán ser iguales. Dicho de otro modo, los tres valores introducidos por el usuario deberán ser distintos entre sí.

4. ENTRADAS:

Los valores introducidos deberán ser números enteros representados de forma numérica y no escrita. Dichos valores únicamente deben ser introducidos desde un teclado numérico perteneciente al sistema de numeración decimal (del 0 al 9)

5. PROCEDIMIENTO:

Se mostrará una ventana con el título del programa, además de un pequeño texto introductorio a modo de tutorial:

Título: *"Comparador de números".*

Texto que se mostrará en la ventana:

"Bienvenido a nuestro programa "Comparador de números", por favor antes de ingresar los caracteres a analizar verifique que estos cumplan con todas las normas mostradas a continuación:

Normas:

Al introducir un valor, verifique que este sea un numero entero positivo, ya que de lo contario el programa no procederá con los siguientes pasos. Caracteres alfanuméricos no procederán en el programa.

Así mismo, verifique que no haya dos o más valores iguales, ya que el programa arrojara un mensaje de advertencia.

Por último, este programa únicamente compara un máximo de tres valores, ni más, ni menos.

Si se ha tomado el tiempo de leer las normas de nuestro programa por favor, proceda con su uso".

Introduzca los tres valores a comparar:

Valor uno: ____

Valor dos: ____

Valor tres: ____

Posteriormente se procederá con la lectura de los tres valores:

NOTA: En caso de que los valores sean introducidos de forma escrita, no sean enteros, sean iguales, o cualquier otra falta de seguimiento a las normas del programa se mostrara el siguiente mensaje:

"Los valores ingresados no han sido reconocidos o son iguales, por favor verifíquelos e inténtelo nuevamente"

Si al momento de introducir los valores se detecta que estos cumplen con los requisitos para su posterior análisis, se procederá con la sustitución de variables (previamente declaradas) por los valores introducidos por el usuario.

Una vez analizados los valores, el programa desplegará una ventana con los resultados de la ejecución.

"El orden de los valores de menor a mayor es: X1,X2, X3

Gracias por utilizar nuestro comparador de número"

FIN DEL PROGRAMA

6. SALIDAS:

Es la información que el programa otorga al usuario al finalizar la ejecución, para este programa hay dos posibles salidas:

En caso de la detección de un error:

"Los valores ingresados no han sido reconocidos o son iguales, por favor verifíquelos e inténtelo nuevamente"

En el caso de una ejecución exitosa:

"El orden de los valores de menor a mayor es: X1,X2, X3

Gracias por utilizar nuestro comparador de números"

7. PRUEBAS DE ESCRITORIO:

Las pruebas de escritorio son una técnica de verificación manual utilizada en programación para evaluar la lógica de un algoritmo o programa antes de su ejecución en una computadora. Se realiza siguiendo paso a paso cada instrucción con valores de entrada específicos y registrando los resultados en una tabla o diagrama para detectar errores lógicos o de flujo.

Pasos para realizar una prueba de escritorio:

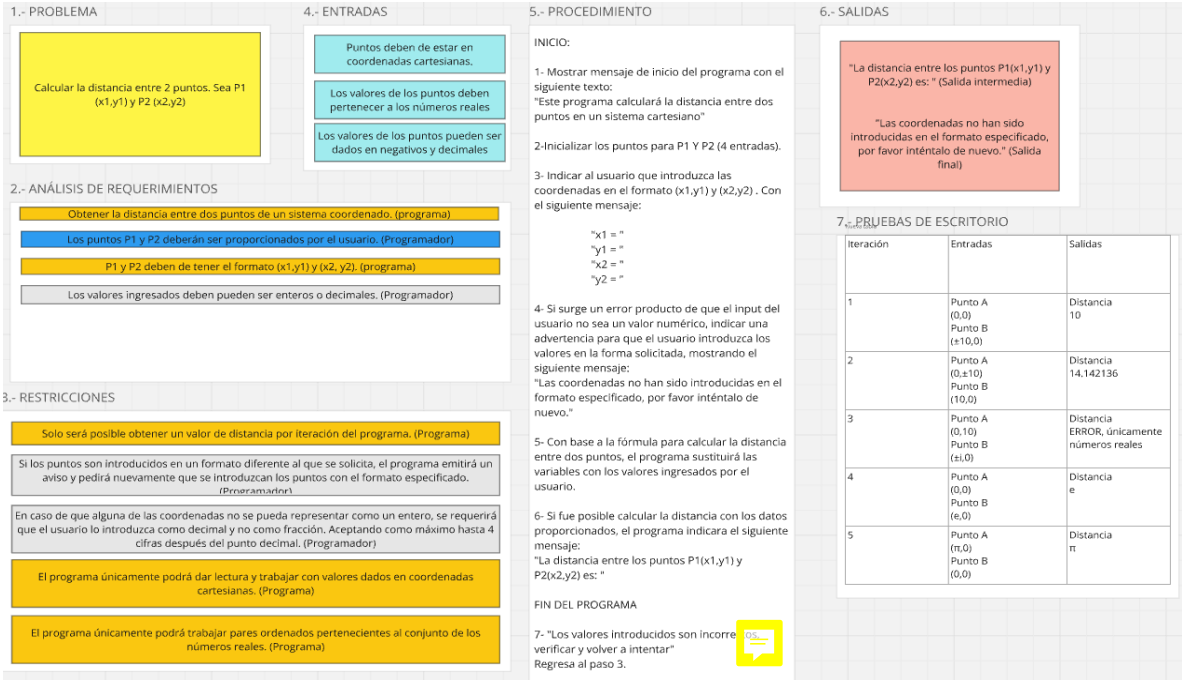
1. Definir el algoritmo a evaluar (puede ser en pseudocódigo o código).
2. Seleccionar valores de entrada para probar distintos casos.
3. Ejecutar manualmente cada paso del algoritmo, anotando los valores de las variables en cada etapa.
4. Comparar los resultados esperados con los obtenidos para detectar errores o inconsistencias.
5. Corregir errores y repetir la prueba si es necesario.

Es una técnica útil para depuración y aprendizaje, especialmente en etapas iniciales del desarrollo de software.

Para nuestro código, las pruebas de escritorio quedarían de la siguiente forma:

Iteración	Entradas	Salidas
1	Valor A 18 Valor B 6 Valor C 100	Orden de acomodo 6<18<100
2	Valor A 5+3i Valor B π Valor C -15	Orden de acomodo Los valores ingresados no han sido reconocidos o son iguales, por favor verifíquelos e inténtelo nuevamente
3	Valor A 0 Valor B π Valor C 12	Orden de acomodo Los valores ingresados no han sido reconocidos o son iguales, por favor verifíquelos e inténtelo nuevamente
4	Valor A 0 Valor B 0 Valor C 0	Orden de acomodo Los valores ingresados no han sido reconocidos o son iguales, por favor verifíquelos e inténtelo nuevamente
5	Valor A 36 Valor B A Valor C 3	Orden de acomodo Los valores ingresados no han sido reconocidos o son iguales, por favor verifíquelos e inténtelo nuevamente

Proceso para el problema 2 (Calcular distancia de dos puntos sea P1(X1,Y1) y P2(X2,Y2)



1. PROBLEMA:

Para dar solución a lo que solicita el problema es necesario que este sea claro y conciso. Debe centralizar lo que se pide en una idea fácil de comprender. El problema en este escenario establece la necesidad de calcular la distancia entre dos puntos.

2. ANÁLISIS DE REQUERIMIENTOS:

Son las condiciones o características necesarias para que el programa funcione correctamente. Describe qué debe hacer el sistema y cómo debe hacerlo para resolver el problema planteado inicialmente. En este caso se incluyó dos tipos de requerimientos: los que el programa debe cumplir para operar sin errores y satisfacer las necesidades del usuario final, y los que establece el programador para que el programa funcione de manera eficiente y productiva. Como requerimientos del programa se estableció el formato en el que deben ser ingresados los valores de las coordenadas y que sea capaz de calcular la distancia con los datos proporcionados. Como requerimientos del programador se estableció que las coordenadas deben ser proporcionadas por el usuario y que dichos valores pueden ser enteros o decimales.

3. RESTRICCIONES:

Son los límites o condiciones que deben considerarse durante el diseño y desarrollo del programa. Al igual que el paso anterior, se consideran restricciones propias del programa y del programador. Estas restricciones deben analizarse y definirse desde el inicio para evitar problemas durante el desarrollo y garantizar que el programa sea viable y eficiente. Las restricciones del programa se enfocaron en que los puntos deben ser introducidos en forma de coordenadas cartesianas, que a su vez pertenezcan a los números reales, y que solo será posible obtener una medición por iteración del programa (si el usuario desea calcular otra distancia reutilizando uno de los puntos usados en una iteración anterior, deberá volver a introducirlo). Las restricciones del programador se enfocaron en la manera en que deben ser introducidos los valores por el usuario, haciendo que el programa muestre un aviso en caso de que se haya detectado un formato incorrecto, pidiendo así al usuario que los ingrese nuevamente de forma correcta.

4. ENTRADAS:

Son los datos o información que el usuario proporciona al programa para que este pueda procesarlos. Para este programa se espera que las entradas sean introducidas de forma manual mediante un teclado de distribución QWERTY. También se espera que el usuario proporcione las coordenadas en el sistema cartesiano y de forma numérica (siempre números reales) aceptando valores negativos y decimales.

5. PROCEDIMIENTO:

Es la etapa en la que se estructura el programa para que sea capaz de interactuar intuitiva y claramente con el usuario para procesar los datos introducidos. El programa inicia con un mensaje de bienvenida en el que se indica que se calculará la distancia entre dos puntos proporcionados por el usuario. Posteriormente se pedirá el input de las coordenadas al usuario en el formato especificado. Una vez que se han proporcionado los valores el programa analizará que sean datos válidos, es decir, que se trate de números reales positivos o negativos y números enteros o decimales, en caso de que se haya introducido algún valor inválido como texto o símbolos el programa arrojará un mensaje de error y pedirá al usuario que vuelva a introducir los valores recordando el formato correcto. Cuando se haya validado que los datos son útiles se asigna a una variable el cálculo de la distancia entre dos puntos, en dicha variable se evaluarán los datos proporcionados. Por último, el programa indicará en pantalla un mensaje final al usuario en el que se muestra el resultado de la operación.

6. SALIDAS:

Son los datos o resultados generados por el programa después de procesar las entradas. Se implementó una salida intermedia para cuando los valores del usuario están en un formato diferente al solicitado para que vuelvan a ser ingresados y una salida final en el que se muestra el resultado de la operación.

7. PRUEBAS DE ESCRITORIO:

Se representa en una tabla ordenada las simulaciones manuales previas a la ejecución del programa y posteriormente el resultado de varias iteraciones del programa para comprobar que esté funcionando de forma adecuada.

Conclusión:

Diego: De esta práctica mi principal conclusión es acerca de la importancia de las especificaciones del programa para el posterior desarrollo de un código o pseudocódigo (Como en nuestro caso), pues es en la definición de la idea donde realmente radica la importancia de ser sumamente analíticos y específicos, ya que cualquier falla en esta etapa puede por terminar mermando nuestros avances a la hora de codificar.

Oscar: La práctica recalca la importancia de tener un entendimiento adecuado acerca del problema antes de proponer soluciones. Como se observó en el desarrollo, es crucial analizar las restricciones y requerimientos ya que estos definen los límites y condiciones bajo las cuales el programa debe funcionar correctamente. No hacerlo puede resultar en un proyecto mal diseñado, costoso o con muchos errores. Este fue mi aprendizaje de la actividad.

César: Se puede sacar de esto que si no se sabe identificar lo que se necesita para un programa, el mismo jamás funcionará. Necesitamos entender el problema por todos los ángulos posibles para poder pasar siquiera a plantear soluciones. Como un rompecabezas, necesitamos las piezas para poder armarlo. Si no logramos apreciar todo lo que necesitamos jamás se depegará el proyecto y valdrá 2 centavos y poco más

Bibliografía

Material de ayuda para la elaboración del documento

ENES
JURIQUILLA

1. Arboleya, J. (2010, 13 junio). Informática = Entrada-Proceso-Salida. Velneo. https://www.velneo.com/blog/informatica-entrada-proceso-salida
2. Maluenda, R. (s. f.). *Qué es un algoritmo informático: características, tipos y ejemplos.* Profile. https://profile.es/blog/que-es-un-algoritmo-informatico/
3. *Requerimientos en el desarrollo de software y aplicaciones.* (2022, 26 mayo). Northware. https://www.northware.mx/blog/requerimientos-en-el-desarrollo-de-software-y-aplicaciones/
4. ¿Qué es la resolución de problemas? (2023, 21 abril). AppMaster. https://appmaster.io/es/blog/que-es-la-resolucion-de-problemas