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 21 de Febrero del 2025

DFD

PRACTICA III

Perteneciente a la materia de:
Fundamentos de programación
Docente:
Ing. Andrés David Flores Ferro
Fecha De Revisión: _____.
Calificación: _____.
Observaciones:

DFD

Practica III

Fundamentos De Programación

ENES

JURIQUILLA

ENES
JURIQUILLA

Introducción:

Los diagramas de flujo son herramientas visuales fundamentales en el ámbito de la programación, ya que permiten representar de manera gráfica y ordenada la secuencia de operaciones, decisiones y procesos que conforman un algoritmo. Utilizando símbolos estandarizado (como el óvalo para marcar el inicio y el fin, el rectángulo para representar acciones o procesos, el diamante para indicar decisiones y el paralelogramo para las operaciones de entrada y salida) estos diagramas actúan como un "mapa" que ilustra la lógica que seguirá un programa, facilitando tanto su diseño como su posterior mantenimiento.

Un diagrama de flujo es, en esencia, una representación gráfica que desglosa un proceso complejo en pasos simples y conectados mediante flechas, las cuales indican el orden cronológico de las operaciones. Su propósito es doble: por un lado, sirven para planificar y diseñar la lógica de un algoritmo antes de que se escriba el código, permitiendo a los programadores visualizar y depurar la estructura del proceso; por otro lado, actúan como documentación que puede ser compartida y entendida por otros miembros del equipo, incluso por aquellos que no están familiarizados con el lenguaje de programación específico.

En la fase de planificación, un diagrama de flujo ayuda a organizar las ideas y a prever posibles errores o cuellos de botella en la lógica del algoritmo. Al visualizar cada paso, los desarrolladores pueden identificar de forma temprana las áreas que podrían optimizarse o que requieren una revisión más detallada, reduciendo así el riesgo de fallos durante la implementación. Además, durante el desarrollo, estos diagramas se utilizan como una forma de documentación visual, facilitando la comprensión del código por parte de nuevos integrantes del equipo y permitiendo un mantenimiento más sencillo cuando se realizan modificaciones o actualizaciones en el sistema. Estos diagramas traen consigo algunos beneficios como:

1. **Claridad y simplificación:** Los diagramas de flujo permiten descomponer procesos complejos en pasos individuales, lo que ayuda a comprender la lógica del algoritmo de forma rápida y sencilla. Esta representación gráfica facilita la identificación de errores y la optimización de la secuencia de acciones.

2. **Mejora en la comunicación:** Al ser un lenguaje visual estandarizado, los diagramas de flujo pueden ser entendidos por profesionales de distintos niveles y áreas, lo que favorece la colaboración y la coordinación en equipos de desarrollo. Esto resulta especialmente útil en entornos donde se trabaja en conjunto en proyectos complejos

3. **Documentación y mantenimiento:** Una representación gráfica clara actúa como una guía de referencia que facilita la actualización y el mantenimiento del software. Esto permite que, ante cualquier cambio en los requerimientos del sistema, se pueda evaluar de forma rápida el impacto en la lógica del proceso.

4. **Detección temprana de errores:** La visualización de cada paso permite identificar rápidamente inconsistencias o fallos en el algoritmo, lo que agiliza el proceso de depuración y contribuye a la creación de código más robusto.

Pero, así como beneficios también hay limitaciones

5. **Complejidad en sistemas grandes:** En proyectos muy extensos o en procesos con múltiples ramificaciones, el diagrama de flujo puede volverse excesivamente complejo y difícil de interpretar. Esto puede generar confusión y disminuir la eficacia del diagrama como herramienta de comunicación.

6. **Tiempo de elaboración y actualización:** La creación de un diagrama de flujo detallado requiere una inversión considerable de tiempo y esfuerzo. Además, cuando el proceso evoluciona, es necesario actualizar el diagrama para que siga siendo una representación fiel del sistema, lo que puede consumir recursos adicionales.

7. **Posible ambigüedad en la interpretación:** Si no se emplean correctamente los símbolos estandarizados o si la conexión entre ellos no es clara, el diagrama puede inducir a errores de interpretación, lo que afectaría negativamente a la implementación del algoritmo.

Desarrollo:

Problema 1: Calculadora De Distancia De Los Dos Puntos En Un Plano De (X,Y):

1. OBJETIVO DE LA PRACTICA:
Elaboración de un diagrama de flujos que representé el proceso que debe llevar a cabo un programa que calculé la distancia entre dos puntos en un plano cartesiano.

2. DESARROLLO:
En este DFD, podemos observar el proceso que debería de llevarse a cabo en un programa en los primeros recuadros se le da una bienvenida al usuario, para más adelante decirle al usuario para qué sirve el programa con el que está trabajando. Mientras esto pasa se le indica al programa que los valores de los diferentes puntos (x1, x2, y1, y2) tiene que empezar en 0, ya que el que se encargará de dar los valores más adelante para los puntos es el usuario.
Una vez que esto sucede, el programa solicita al usuario que proporcione los valores, que pueden ser en números enteros o decimales, de los puntos, empezando con x1, después y1 cuando el usuario ingresa el programa le solicita nuevamente el mismo proceso, pero para los puntos x2 y y2. Una vez que el programa a obtenido los valores aplica la fórmula para la distancia entre dos puntos:

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

3. RESULTADOS:
Al final el programa le proporciona al usuario la distancia que hay entre los dos puntos, cuando el usuario continúa el mismo programa le indica el fin del programa y se cierra finalmente.

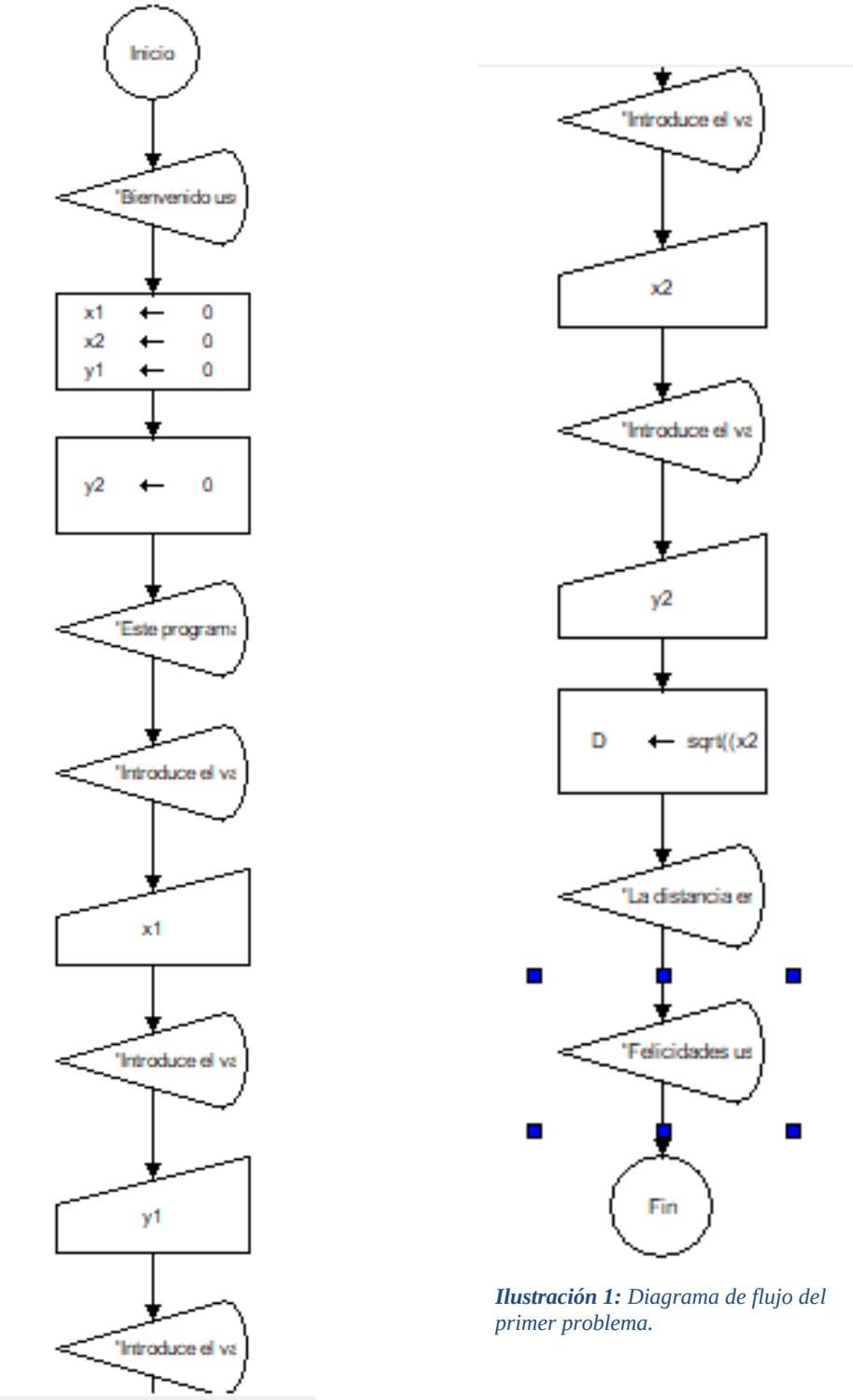

Ilustración 1: *Diagrama de flujo del primer problema.*

Problema 2: Acomodador De Números De Menor A Mayor (Comparador)

1. PROBLEMA PLANTEADO:

Elaboración de un diagrama de flujo que, dados tres números reales (introducidos por el usuario) realice el acomodo de estos de menor a mayor.

2. EL CONCEPTO E IDEAS BASICAS:

Para realizar un programa capaz de comparar tres números y ordenarlos de menor a mayor, se pensó primeramente en la definición de restricciones. Esto para que los valores introducidos por el usuario cumplieran con una serie de requisitos específicos y que el programa fuera capaz de procesarlos.

Posteriormente, se dedujo que una comparación por medio de desigualdades era lo mejor para el diagrama de flujo- programa.

3. BIENVENIDA AL PROGRAMA:

El programa primeramente da la bienvenida al usuario, seguido de las instrucciones generales y normas básicas que se deben seguir para la correcta ejecución del programa.

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.

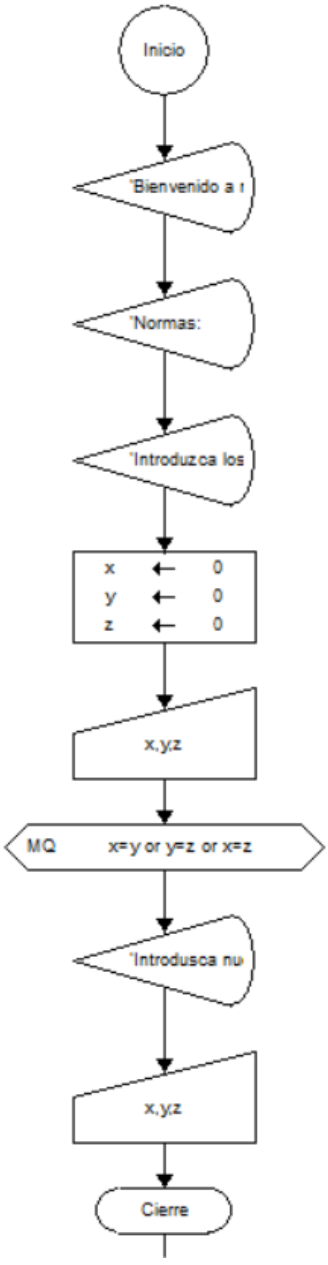

***Ilustración 2**: Primera parte del diagrama de flujo del segundo problema.*

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".

Después de haber desplegado los mensajes de guía, al diagrama de flujo se le añadieron dos elementos sumamente importantes: El símbolo de asignación (Con las variables x, y, z) y el símbolo de lectura.

4. EJECUCIÓN:

Para la realización de este diagrama de flujo se optó por hacer uso de una serie de condicionales Si/No. Otra idea que fue descartada fue hacer uso de un ciclo mientras.

Nota de observación: Al momento de desarrollar el diagrama de flujo para el "núcleo del comparador" se optó por condicionales que hiciesen uso de una sola desigualdad. (Ej. X<y o z<y), por lo que después de colocar tres posibles salidas nos encontramos con lo que resultó ser un problema de permutaciones con un número indeterminado de elementos comparativos para la solución de los otros tres casos restantes.

Para solucionar el problema anterior, se hicieron uso de elementos condicionales con dos desigualdades (Ej. X<y<z)

5. PRIMERA CORRECCIÓN Y DESARROLLO:

Al utilizar símbolos condicionales que compararan las tres variables juntas, se procedió al cálculo de todos los posibles casos a través de la fórmula para la obtención de permutaciones.

$$P_r^n = \frac{n!}{(n-r)!} = \frac{3!}{(3-3)!} = \frac{6}{0!} = \frac{6}{1} = 6$$

Donde:

$$\boldsymbol{n} = \textit{ Número total de elementos} = 3$$
$$\boldsymbol{r} = \textit{ Numero de objetos seleccionados} = 3$$

El resultado dado por la anterior formula era el número total de casos (salidas) posibles que podían generarse. A continuación, se anexa la tabla con todas las salidas.

Numero de permutación:	Permutación:
1	x>y>z
2	x>z>y
3	y>x>z
4	y>z>x
5	z>x>y
6	z>y>x

Posteriormente, se procedió a la elaboración del diagrama de flujo del "Núcleo del comparador" (Parte en la cual se comparan las variables), quedando el diagrama de flujo de la siguiente manera:

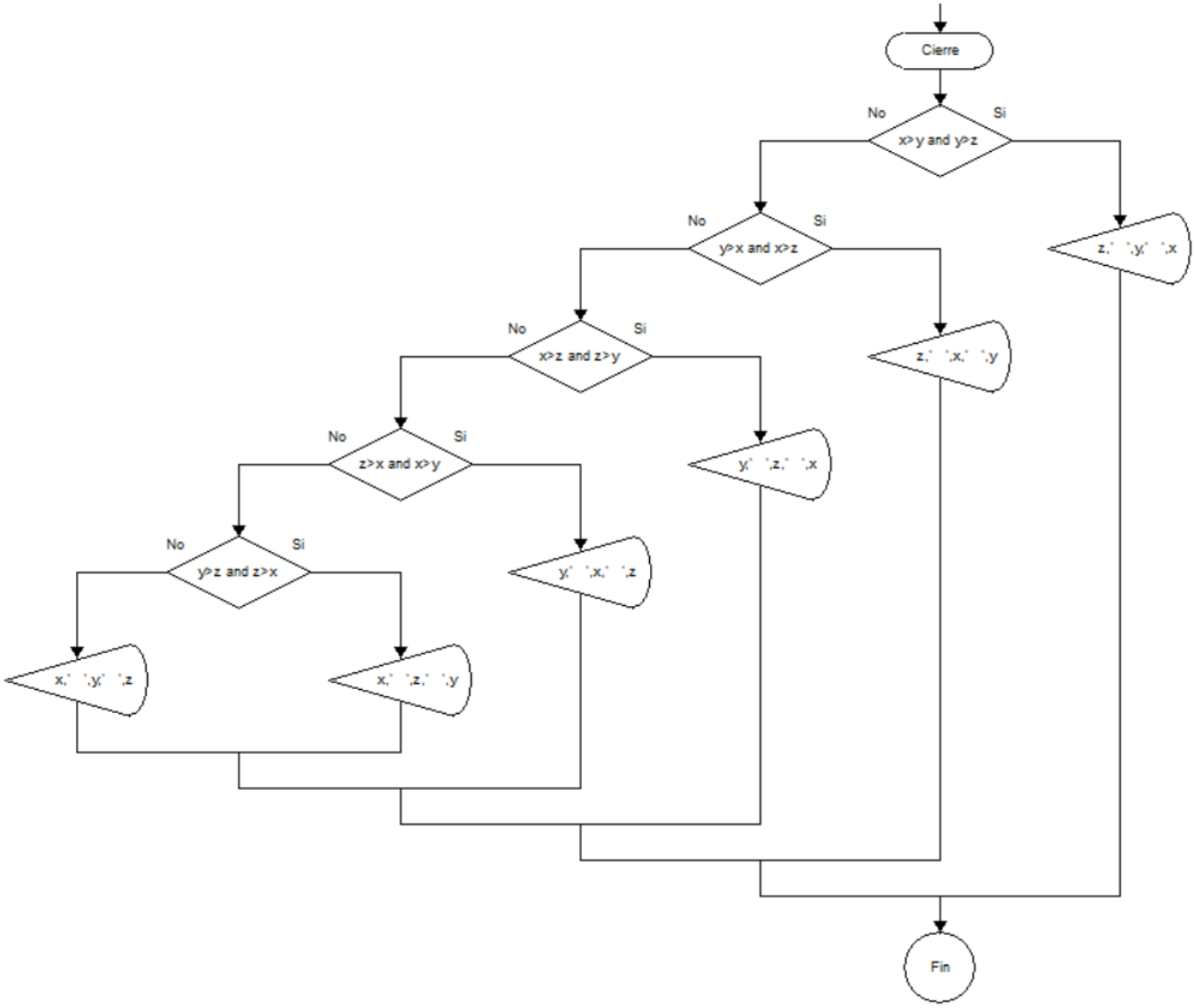

Ilustración 3: *Segunda parte del diagrama de flujo (Segundo problema)*

6. ELABORACIÓN DE LAS RESTRICCIONES DEL PROGRAMA:

A continuación, se anexa una tabla con las restricciones que el programa fue capaz de realizar.

Restricción	Resultado
El algoritmo/programa únicamente trabajara con valores numéricos pertenecientes al sistema de numeración decimal.	Completado
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.	Completado
Pertenecer al conjunto de los números reales	Completado
Pertenecer al conjunto de los números enteros	Incompleto
Únicamente será posible el análisis de tres valores por iteración del programa	Completado
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.	Completado

Algunas de estas restricciones no se pudieron cumplir del todo debido a falta de conocimiento u experiencia previa del programa. A pesar de esto, el desempeño de nuestro algoritmo pudo ser probado mediante la realización de pruebas de escritorio, quedando estas de la siguiente manera:

7. TABLA DE PRUEBAS DE ESCRITORIO:

A continuación, se muestran algunas pruebas realizadas al programa con diversas entradas.

Iteración	Entradas	Salidas
1	Valor A 1 Valor B 2 Valor C 3	1 2 3
2	Valor A 1 Valor B 1 Valor C 2	Ingrese los datos nuevamente
3	Valor A 0 Valor B 0 Valor C 0	Ingrese los datos nuevamente
4	Valor A p Valor B 5 Valor C 2	Debe ingresar un valor constante
5	Valor A 3.5 Valor B 2.66 Valor C 5.0002	2.66 3.5 5.0002

Conclusión:

Diego: Al concluir esta práctica, es más que evidente la importancia de los diagramas de flujo y los elementos gráficos como herramientas para la realización de algoritmos y programas más complejos que sean capaces de resolver problemáticas de una mayor dificultad. Pues es debido a estas que se puede realizar un mejor acomodo de los diversos procesos que requiere seguir un ejecutable.

César: Una vez que se identifican los criterios, las entradas, salidas y las restricciones, en sí, ya tienes el panorama completo del problema o lo que quieres realizar, el siguiente paso es realizar los DFD o diagramas de flujo, con esta herramienta pasas ya a tratar de resolver el problema con un algoritmo, los DFD son prácticamente un ayuda visual antes de meterte a programar un algoritmo completo, cada etapa te deja más claro los procesos que se van a llevar a cabo durante el programa completo. Haciendo de los DFD una buena antesala a la programación cuando apenas vas aprendiendo.

Oscar: Con la práctica queda claro que los diagramas de flujo son especialmente útiles para representar de forma clara y visual cómo funciona un proceso o un algoritmo. Es más fácil descomponer un problema en pasos pequeños y detallados, lo que facilita su análisis y resolución. DFD parece ser una buena opción para comenzar con el desarrollo de estos diagramas. Al mostrar la información de manera secuencial los diagramas de flujo ayudan a entender mejor los pasos a seguir en un procedimiento. También permiten detectar errores o inconsistencias antes de llevar a cabo la ejecución de un algoritmo o proceso, reduciendo el tiempo y esfuerzo necesarios para corregir problemas.

Bibliografía

Material de ayuda para la elaboración del documento

1. (N.d.). Dongee.com. Retrieved February 20, 2025, from https://www.dongee.com/tutoriales/ventajas-y-desventajas-diagrama-de-flujo/
2. (N.d.-b). Unir.net. Retrieved February 20, 2025, from https://mexico.unir.net/noticias/ingenieria/diagrama-flujo/
3. Gonzalez, O. (2022). ¿Qué es un diagrama de flujo y para qué sirve? ¡Planificación y eficiencia en un solo paso! https://www.crehana.com. https://www.crehana.com/blog/negocios/que-es-un-diagrama-de-flujo/
4. ¿Qué es un diagrama de flujo de programación y cómo hacerlo? (n.d.). Boardmix. Retrieved February 20, 2025, from https://boardmix.com/es/knowledge/programming-flowchart/
5. Alam, M. (2023). Ventajas del diagrama de flujo: 9 razones para utilizar una plantilla de organigrama. IdeaScale. https://ideascale.com/es/blogs/flowchart-ventajas/