**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 02 de Marzo del 2025

# PSEUDO-CÓDIGOS

## PRACTICA IV

**Perteneciente a la materia de:**
Fundamentos de programación

**Docente:**
Ing. Andrés David Flores Ferro

**Fecha De Revisión:** 02 / 03 / 2025.

**Calificación:** 91 de 100 pts.

**Observaciones:**

- Agregar dato de tipo de archivo consultado en referencias [Web,PDF, Libro, etc] , no hay contenido de clase citado o algun texto de clase.
- Se menciona el contenido de los ejercicios y en el otro si se desarrolla que es cada etapa pero no como se hace , que involucra lo que usan IDEALMENTE se queria descripcion como si fuese un paso a paso, y la teoria detras mas sus correcciones.
- En introduccion se tiene poco contenido de clase(descripcion pseudo) pero es una buena investigación

| | |
|---|---|
| **Actividad en clase 22 de 25 pts** | |
| **Protocolo de práctica** | **69 de 75 pts** |
| Portada | 10 de 10 pts |
| Introducción | 13 de 15 pts |
| Desarrollo | 23 de 25 pts |
| Conclusiones | 15 de 15 pts |
| Referencias | 08 de 10 pts |
| **CALIFICACIÓN TOTAL** | **Entrega 91 pts** |

# PSEUDO-CÓDIGOS

## Practica IV

### Fundamentos De Programación

ENES
JURIQUILLA



ENES
JURIQUILLA

# Introducción:

El pseudocódigo es la representación escrita y estructurada de un algoritmo utilizando un lenguaje natural con ciertos elementos formales de los lenguajes de programación. Se utiliza para plasmar, de forma clara y comprensible, los pasos necesarios para resolver un problema sin atarse a la sintaxis rígida de un lenguaje específico. Esto permite a los desarrolladores concentrarse en la lógica y la estructura del algoritmo, facilitando el diseño, la depuración y la posterior implementación en el lenguaje que se elija.

Una vez que se ha analizado un problema y se han identificado tanto los datos de entrada como los de salida, el siguiente paso es diseñar un algoritmo que lo resuelva de manera eficiente. Aquí es donde el pseudocódigo cobra un papel esencial: actúa como un puente entre el planteamiento abstracto del problema y la codificación concreta. Antes de escribir una sola línea de código, se debe generar una representación que permita visualizar el proceso de solución. El pseudocódigo cumple con este cometido al detallar, paso a paso, el flujo lógico del algoritmo, permitiendo identificar posibles errores o ineficiencias en una fase temprana del desarrollo.

## Características y usos:

1. **Claridad y simplicidad:** Al escribirlo en un lenguaje cercano al natural, se evita la complejidad de la sintaxis de lenguajes específicos, haciendo que la descripción del algoritmo sea accesible para desarrolladores y personas con distintos niveles de experiencia.

2. **Independencia del lenguaje:** No está ligado a ningún lenguaje de programación en particular, lo que facilita la traducción del algoritmo a cualquier otro lenguaje. Esta independencia lo convierte en una herramienta versátil tanto para la enseñanza como para la práctica profesional.

3. **Facilita la comunicación:** Al servir de lenguaje común, el pseudocódigo permite que los miembros de un equipo, incluso si trabajan con diferentes tecnologías, comprendan la lógica subyacente de un programa sin ambigüedades.

4. **Planificación y depuración:** Su uso permite detectar errores lógicos y optimizar la solución antes de enfrentarse a los detalles técnicos de la codificación real.

El pseudocódigo es fundamental durante la etapa de diseño de algoritmos. Una vez definido el proceso mediante este “lenguaje intermedio”, el programador puede traducirlo de forma más sencilla a código ejecutable. Esto no solo reduce la probabilidad de errores, sino que también mejora la documentación y el mantenimiento del software. Además, en el ámbito educativo resulta una herramienta invaluable para enseñar los conceptos básicos de la programación, ya que permite concentrarse en el desarrollo del pensamiento algorítmico sin la distracción de la sintaxis.

# Desarrollo:

**El pseudocódigo desarrollado permite seleccionar 3 opciones de un menú principal:**

1. Calculadora capaz de operar únicamente 2 valores por iteración (suma, resta, multiplicación, división y potenciación).
2. Calcular la distancia entre dos puntos P1(x1,y1) y P2(x2,y2).
3. Ordenar 3 números distintos entre sí de menor a mayor.

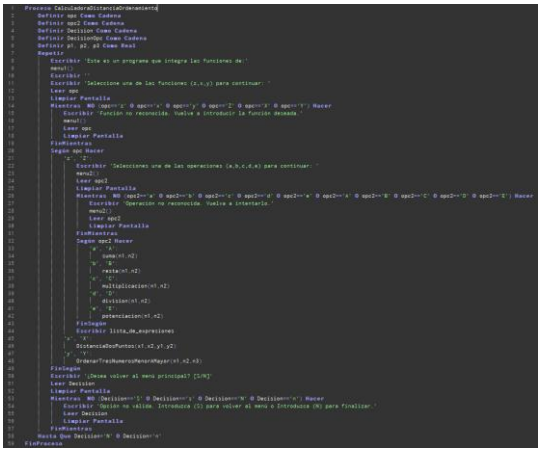
Una vez que el programa arroja el resultado se le da la opción al usuario de volver a realizar la misma operación. Si el usuario decide no volver a realizar la misma operación, se le da la opción de volver al menú principal o finalizar el programa.

De igual forma el programa verifica que cualquier input del usuario corresponda a lo que se solicita: ya sea una letra específica para navegar en los menús o tomar decisiones, o que se introduzcan valores numéricos al realizar los cálculos. Mostrando así un mensaje que indica el error ocurrido con el input para después volver a solicitar la información.

# Segmento principal del pseudocódigo:

**Proceso Calculadora Distancia Ordenamiento**

Esta es la parte más importante de todo el pseudocódigo ya que es la que da estructura y orden al programa. Todas las operaciones son procesadas por medio de Sub-Algoritmos, al igual que los menús.



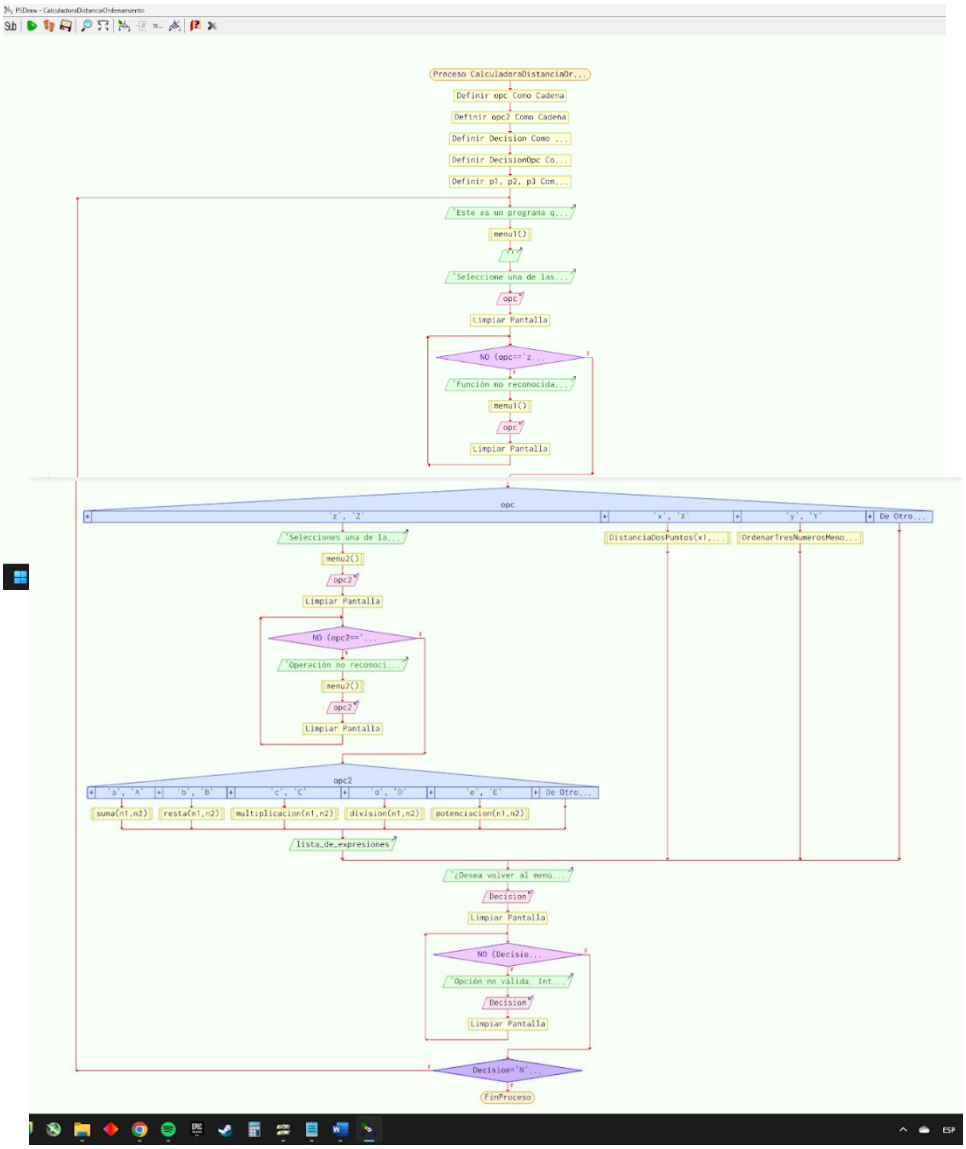
Empieza definiendo las variables según el tipo de información que guardarán más adelante, después se abre un ciclo repetir que cierra hasta el fin del proceso. Ahora se le pide al usuario que seleccione una de las opciones del menú principal ingresando una letra específica, si el usuario introduce otro valor de los que se muestran en el menú, el programa solicitará nuevamente que se seleccione una opción.



ENES JURIQUILLA

Posteriormente se abre un bloque “según” que ejecuta la opción deseada por el usuario. Si se selecciona la calculadora se abrirá un nuevo menú del cual el usuario deberá seleccionar una de las 5 operaciones disponibles (validando nuevamente que se introduzca de forma correcta), dicha selección se operará en otro bloque “según”.

Una vez terminado el cálculo de cualquiera de las 3 funciones principales (Calculadora, Ordenamiento y Distancia) del programa se le da a elegir al usuario entre volver a realizar la operación previa o volver al menú principal, continuando así en bucle hasta que el usuario finalice el programa gracias al ciclo repetir.






ENES JURIQUILLA

# Sub-Algoritmos De Suma, Resta, Producto y Potencia: (Según A, B, C y E):

**Proceso Calculadora Distancia Ordenamiento**

La opción de operar dos números se encuentra disponible en el ejecutable, esto se logra mediante la lectura de dos datos que son validados por el Sub-Algoritmo de validación:
La validación de los datos permite que el procesamiento de las variables proceda únicamente si estas son números reales. Discriminando textos o posibles caracteres alfanuméricos.
Al finalizar la operación, el programa le dará la posibilidad al usuario de repetir nuevamente el proceso con números distintos (O iguales, si es que el usuario lo decide), o si por el contrario, desea finalizar la ejecución.

Para esto, el pseudocódigo cuenta con la implementación de un ciclo mientras, que únicamente permitirá el desarrollo del programa si el usuario define una de las dos opciones. Los operadores matemáticos “Igual que” permiten que la variable “OPC” (La variable de control) se encuentre restringida a dos posibles casos.

# Sub-Algoritmo de División: (Según D)

**Proceso Calculadora Distancia Ordenamiento**

El algoritmo del cociente de dos números presenta algunos cambios significativos con respecto al de sus similares.

Como es conocido por gran parte de la comunidad escolar, la división de cualquier número real entre cero es indeterminada. Por esta misma situación es que se decidió añadir una restricción que no permita la continuidad de la ejecución si el segundo número introducido por el usuario es igual a cero.

La restricción se logra hacer mediante dos ciclos mientras, que al evaluar la lectura de los datos introducidos por el usuario detecta esta situación. Esto a su vez, también permite la reentrada de datos de manera indefinida hasta que la lectura del valor proporcionado sea distinta de cero.

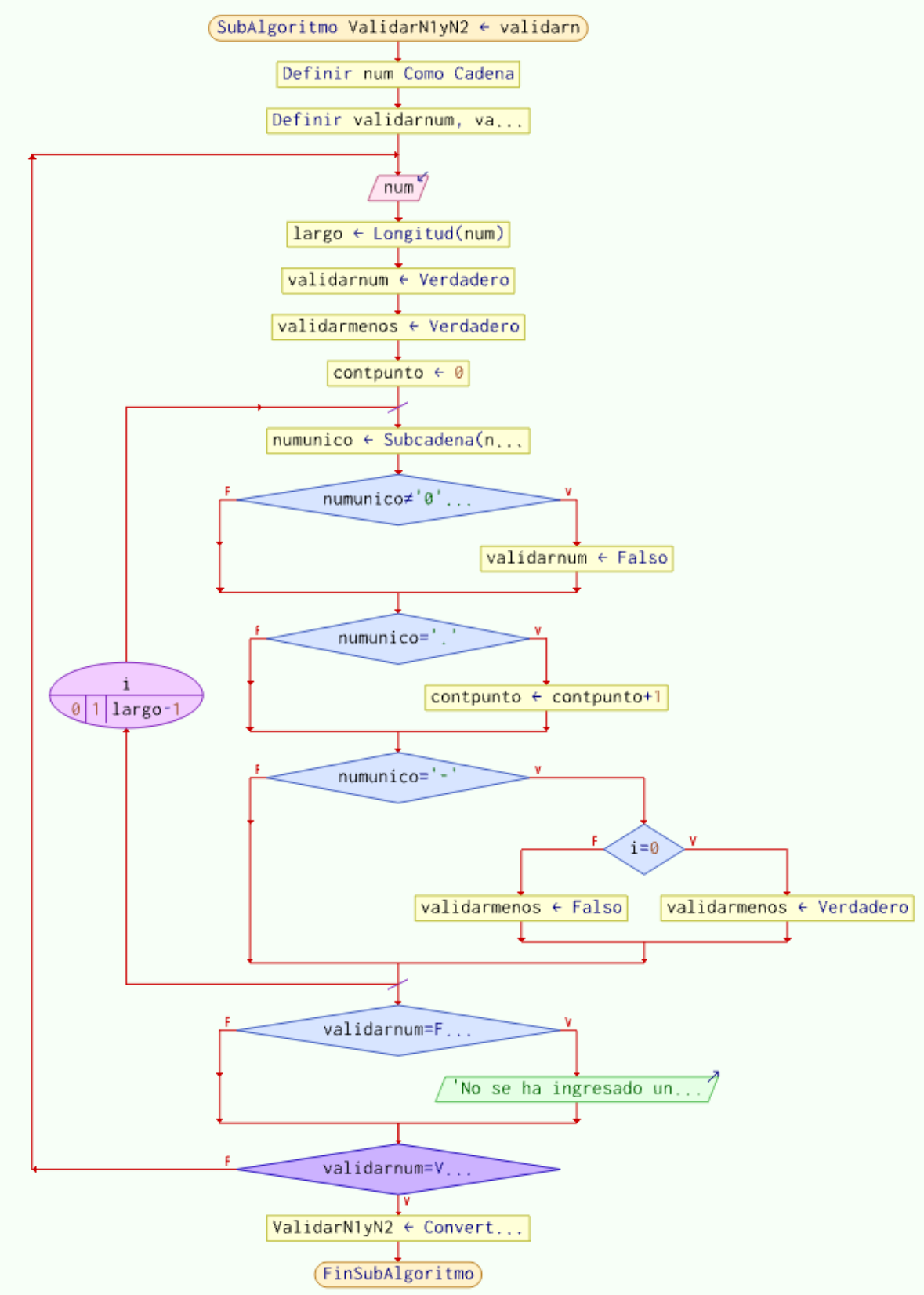
# Sub-Algoritmo ValidarN1yN2 = validarn

**Proceso Calculadora Distancia Ordenamiento**

Este Sub-Algoritmo es el encargado de comprobar que se ingresen valores numéricos y no otro tipo de información cuando el programa lo solicita. Empieza definiendo el input del usuario como cadena, ya que dicho input será validado carácter por carácter.
Posteriormente con un ciclo "para" se valida que cada carácter ingresado sea igual a cualquiera de los números del 0 al 9, a un punto o al signo menos. Con bloques "si" se condiciona a que solo haya un punto y a que, si se usó el signo menos, este ocupe el espacio del primer carácter y solo haya uno. Si no se cumplen estas condiciones, se arroja el mensaje de error para volver a pedir que se ingrese un valor numérico.

Todo está dentro de un ciclo "repetir" para que sea posible la reentrada del input si no se registró correctamente. Por último, una vez que se validó que se cumplen todas las condiciones de forma satisfactoria, se usa la función "ConvertirANumero" para que sea posible operar el valor, de otra forma falla el código porque no se pueden hacer operaciones con variables definidas como cadenas.

```
// validar que n1 y n2 sean numeros (neg, pos, decim) y no cadena de texto
SubAlgoritmo ValidarN1yN2 <- validarn
    Definir num Como Cadena
    Definir validarnum, validarmenos Como Lógico
    Repetir
        Leer num
        largo <- Longitud(num)
        validarnum <- Verdadero
        validarmenos <- Verdadero
        contpunto <- 0
        Para i<0 Hasta largo-1 Con Paso 1 Hacer
            numunico <- Subcadena(num,i,i)
            Si numunico<>'0' Y numunico<>'1' Y numunico<>'2' Y numunico<>'3' Y numunico<>'4' Y numunico<>'5' Y numunico<>'6' Y numunico<>'7' Y numunico<>'8' Y numunico<>'9' Y numunico<>'.' Y numunico<>'-' Entonces
                validarnum <- Falso
            FinSi
            Si numunico='.' Entonces
                contpunto <- contpunto+1
            FinSi
            Si numunico='-' Entonces
                Si i=0 Entonces
                    validarmenos <- Verdadero
                SiNo
                    validarmenos <- Falso
                FinSi
            FinSi
        FinPara
        Si validarnum=Falso O (contpunto>1) O validarmenos=Falso Entonces
            Escribir 'No se ha ingresado un numero de forma correcta. Intentalo de nuevo.'
        FinSi
    Hasta Que validarnum=Verdadero Y (contpunto=0 O contpunto=1) Y validarmenos=Verdadero
    ValidarN1yN2 <- ConvertirANumero(num)
```


SubAlgoritmo ValidarN1yN2 ← validarn
Definir num Como Cadena
Definir validarnum, va...
num
largo ← Longitud(num)
validarnum ← Verdadero
validarmenos ← Verdadero
contpunto ← 0
numunico ← Subcadena(n...
numunico≠'0'...
F
V
validarnum ← Falso
numunico='.'
contpunto ← contpunto+1
i
0 1 largo-1
numunico='-'
i=0
validarmenos ← Falso
validarmenos ← Verdadero
validarnum=F...
'No se ha ingresado un...
validarnum=V...
ValidarN1yN2 ← Convert...
FinSubAlgoritmo

# Sub-Algoritmo Ordenar Tres Números De Menor A Mayor

**Proceso Calculadora Distancia Ordenamiento**

El ordenador de números de menor a mayor, (También denominado como comparador), es uno de los tres principales Sub-Algoritmos de nuestro programa.

**Funcionamiento:**
Como su nombre lo indica, este Sub-Algoritmo tiene por principal objetivo el acomodo de tres números dados por el usuario para posteriormente ser comparados y reacomodados de menor a mayor de acuerdo su magnitud.

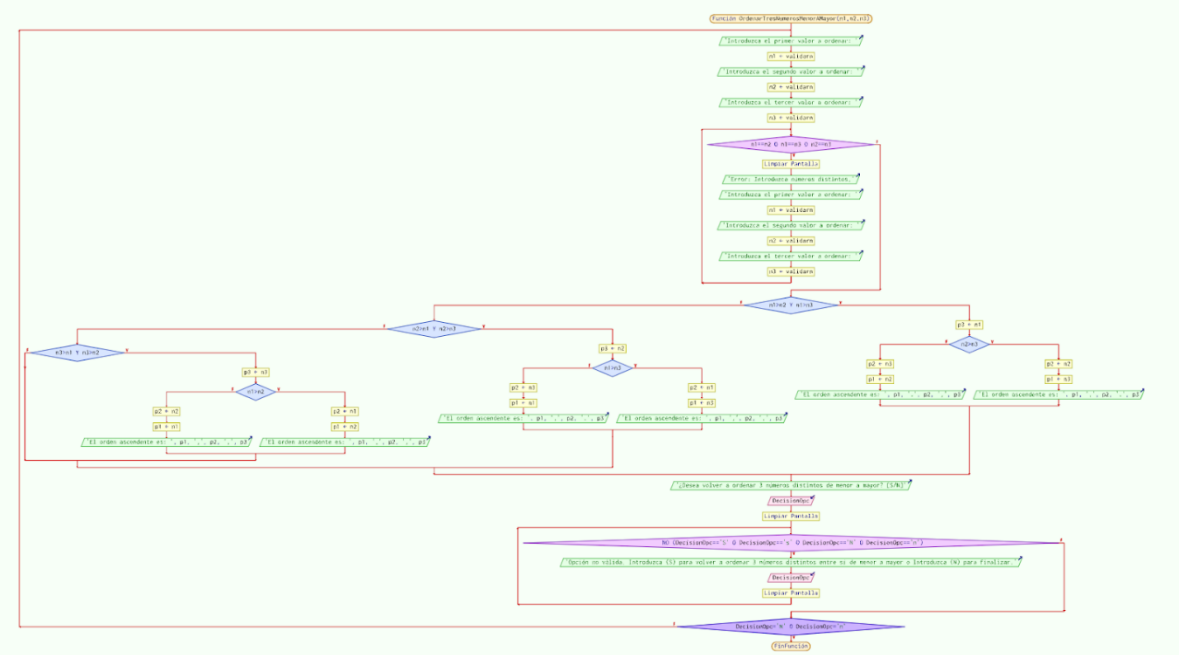
Su funcionamiento se basa principalmente en el uso de condicionales y de ciclos mientras (While). Estos últimos le permiten al usuario la reentrada de datos en caso de que se detecte una de las siguientes condiciones:

1. El conjunto de datos numéricos introducidos por el usuario son iguales.
2. El conjunto de datos numéricos introducidos por el usuario son letras o caracteres alfanuméricos.
3. Solo son introducidos uno o dos números por el usuario.

El ciclo mientras únicamente permitirá el desarrollo del programa si los datos introducidos son tres números distintos entre sí.

```
207 SubAlgoritmo OrdenarTresNumerosMenorAMayor(n1,n2,n3)
208     Repetir
209         Escribir 'Introduzca el primer valor a ordenar: '
210         n1 ← validarn
211         Escribir 'Introduzca el segundo valor a ordenar: '
212         n2 ← validarn
213         Escribir 'Introduzca el tercer valor a ordenar: '
214         n3 ← validarn
215         Mientras n1==n2 O n1==n3 O n2==n3 Hacer
216             Limpiar Pantalla
217             Escribir 'Error: Introduzca números distintos.'
218             Escribir 'Introduzca el primer valor a ordenar: '
219             n1 ← validarn
220             Escribir 'Introduzca el segundo valor a ordenar: '
221             n2 ← validarn
222             Escribir 'Introduzca el tercer valor a ordenar: '
223             n3 ← validarn
224         FinMientras
```

**El comparador:** Una vez que el ejecutable ha realizado una lectura satisfactoria de los datos introducidos por el usuario, se procede a una comparación entre ellos. Esto se logra mediante la lectura de seis condicionales que mediante el uso de los operadores matemáticos “mayor que” y “menor que” evalúan los seis distintos casos posibles de acomodo.



Una vez finalizada esta parte, el programa da más indicaciones al usuario acerca de su uso.

**Fin del Sub-Algoritmo:**
Después de completar el proceso, el programa le ofrece al usuario la posibilidad de volver a realizar el mismo procedimiento, o si, por el contrario, desea dar por finalizada la ejecución del Sub-Algoritmo.

# Sub-Algoritmo Distancia Entre Dos Puntos (x1,x2,y1,y2)

**Proceso Calculadora Distancia Ordenamiento**

Este Sub-Algoritmo es el responsable de calcular la distancia entre dos puntos. Es muy similar a los Sub-Algoritmos propios de la calculadora al englobar dentro del ciclo “repetir” todo el proceso de ingreso de valores validados (por el Sub-Algoritmo validarn) y su posterior procesamiento. Pide cada coordenada de los 2 puntos P1(x1,y1) y P2(x2,y2) en un mensaje diferente, siendo 4 peticiones en total. Las 4 coordenadas se reemplazan en la fórmula para calcular la distancia entre dos puntos, para finalmente mostrar el resultado y cuestionar al usuario si desea volver a realizar la operación.

En la línea: resultado <- rc(((x2-x1)^2)+((y2-y1)^2)), la función “rc()” es asumida como el método para calcular la raíz cuadrada. Es la aplicación directa de la fórmula de la distancia euclidiana. Elevar las diferencias al cuadrado y sumarlas asegura que se eliminen los signos negativos; la raíz cuadrada retorna el valor final de la distancia.

```
182  SubAlgoritmo DistanciaDosPuntos(x1,x2,y1,y2)
183      Definir resultado Como Real
184      Repetir
185          Escribir 'Según las coordenadas P1(x1,y1) y P2(x2,y2)'
186          Escribir 'Introduzca x1: '
187          x1 ← validarn
188          Escribir 'Introduzca y1: '
189          y1 ← validarn
190          Escribir 'Introduzca x2: '
191          x2 ← validarn
192          Escribir 'Introduzca y2: '
193          y2 ← validarn
194          resultado ← rc(((x2-x1)↑2)+((y2-y1)↑2))
195          Escribir 'Distancia entre P1 y P2: ', resultado
196          Escribir '¿Desea volver a calcular la distancia entre dos puntos? [S/N]'
197          Leer DecisionOpc
198          Limpiar Pantalla
199          Mientras  NO (DecisionOpc=='S' O DecisionOpc=='s' O DecisionOpc=='N' O DecisionOpc=='n') Hacer
200              Escribir 'Opción no válida. Introduzca (S) para volver a calcular la distancia entre dos puntos o Introduzca (N) para finalizar.
201              Leer DecisionOpc
202              Limpiar Pantalla
203          FinMientras
204      Hasta Que DecisionOpc='N' O DecisionOpc='n'
205  FinSubAlgoritmo
```

# Conclusiones:

**Diego:** Es importante hacer uso de los distintos operadores y ciclos disponibles dentro del programa para poder limitar y restringir las diversas funciones que ofrece nuestro ejecutable. La implementación de los ciclos “mientras” y “para”, así como de los operadores “no igual”, “mayor que”, “menor que”, además de la definición de las variables como enteros o caracteres, nos facilitó el proceso de desarrollo del pseudocódigo y a su vez, también tuvo un impacto muy notorio en el producto final de esta práctica.
Aunado a lo anterior, el desarrollo del pseudocódigo en base al diagrama de flujo también influyó demasiado a la hora de agregar o quitar las distintas modificaciones que se iban agregando al programa.

**Oscar:** Personalmente, considero que revisar e implementar las estructuras de pseudocódigos ayuda a desarrollar la lógica secuencial necesaria para abordar problemas de programación más complejos. Creo que se complementa bastante con los temas de diagramas de flujos y el abordaje de requerimientos y restricciones del problema previos al desarrollo de la solución. En general el tema de pseudocódigos es el que más ha captado mi atención porque se asemeja más a la sintaxis de código en un lenguaje establecido.

**Cesar:** El pseudocódigo es la parte más fundamental de la programación, con el mismo se pueden hacer programas de una forma “sencilla” haciendo uso todavía del lenguaje humano haciendo así más sencillo el programar sabiendo claro las funciones de cada cosa. Los ciclos permiten hacer qué tu programa continúe infinidad de veces para que el usuario continúe haciendo operaciones y el programa no acabe repentinamente. El ciclo nos ayudó precisamente a continuar con el programa haciendo usos de códigos como el “para” o “mientras” dándole más soporte a nuestro programa para realizar más operaciones diversas dándole o enseñándole al programa que cosas tomar en cuenta y que otras no para que direccionará al usuario en la dirección correcta.

# Referencias:

1. Robledano, A. (2024, 23 septiembre). *Qué es pseudocódigo y por qué es esencial en programación.* OpenWebinars.net. https://openwebinars.net/blog/que-es-pseudocodigo/
2. (N.d.-b). *Unir.net.* Retrieved February 20, 2025, from https://mexico.unir.net/noticias/ingenieria/diagrama-flujo/
3. Kinsta. (2025, 26 febrero). *¿Qué es el Pseudocódigo y Cómo Puede Mejorar tu Programación?* Kinsta®. https://kinsta.com/es/base-de-conocimiento/que-es-pseudocodigo/
4. Delgado, J. (2022, 21 junio). *Introducción al Pseudocódigo.* José Delgado. https://josedelgado.net/introduccion-al-pseudocodigo/