

Reporte Técnico de Actividades Práctico-Experimentales Nro. 002

1. Datos de Identificación del Estudiante y la Práctica

Nombre del estudiante(s)	Kiara Salomé Condoy Morocho
Asignatura	Teoría de la programación
Ciclo	1 A
Unidad	1
Resultado de aprendizaje de la unidad	Identifica los conceptos fundamentales de la teoría de la programación, bajo los principios de solidaridad, transparencia, responsabilidad y honestidad.
Práctica Nro.	002
Tipo	Individual
Título de la Práctica	Del diseño del algoritmo con estructuras secuenciales a la construcción del programa.
Nombre del Docente	Lisette Geoconda López Faicán
Fecha	Martes 28 de octubre del 2025
Horario	10h30 – 13h30
Lugar	Aula física asignada al paralelo.
Tiempo planificado en el Sílabo	6 horas

2. Objetivo(s) de la Práctica

- Desarrollar la capacidad de transformar un problema en una solución computacional.
- Aplicar estructuras secuenciales en el diseño del algoritmo.
- Validar la lógica del algoritmo mediante pruebas de escritorio.
- Implementar y ejecutar la solución en un lenguaje de programación.

3. Materiales, Reactivos, Equipos y Herramientas

- Herramienta de pseudocódigo y diagramación de algoritmos: PSeInt.
- IDE de programación: Visual Studio Code u otro entorno compatible.
- Lenguaje de programación: C (según los contenidos de la unidad).
- Computador personal con sistema operativo Windows, Linux o macOS.
- Material de apoyo en el Entorno Virtual de Aprendizaje (EVA).
- Editores de texto (Word, Google Docs u otros) para la elaboración del informe técnico en formato PDF. FEIRNNR - Carrera de Computación
- Conexión a internet estable para acceder a recursos digitales y software en línea.

- Aula física asignada al paralelo.

4. Procedimiento / Metodología Ejecutada

La práctica se desarrolló aplicando la metodología de **aprendizaje basado en problemas (ABP)**.

A continuación, se detallan los pasos ejecutados durante el desarrollo de la actividad:

1. **Contextualización del problema**
Se planteó el caso de un estudiante que necesita conocer qué nota debe obtener en el tercer certamen (C3) para aprobar la asignatura con una calificación final de 60/100. Se analizaron las fórmulas proporcionadas para calcular el promedio de certámenes (NC) y la nota final (NF), considerando también la nota de laboratorio (NL).
2. **Análisis del problema**
Se identificaron los **datos de entrada** (C1, C2, NL), el **proceso** (operación matemática para despejar C3) y la **salida esperada** (nota mínima requerida en C3). Este análisis permitió comprender la lógica del problema y establecer las condiciones necesarias para su resolución.
 1. **Diseño del algoritmo**
 - Se elaboró el **pseudocódigo** utilizando la herramienta PSeInt, incluyendo comentarios explicativos que detallan cada paso del proceso.
 - Se diseñó el **diagrama de flujo**, igualmente en PSeInt en donde se emplea símbolos estandarizados y una lógica secuencial clara.
 - Se realizaron **pruebas de escritorio** con al menos tres casos distintos: uno con notas bastante bajas, uno con notas promedio y otro con excelentes calificaciones. Estos casos permitieron validar la lógica del algoritmo antes de su implementación.
 2. **Codificación del algoritmo**
Se trasladó el pseudocódigo al lenguaje de programación **C**, utilizando el entorno de desarrollo **Visual Studio Code**. Se respetó la sintaxis del lenguaje, aplicando buenas prácticas de programación y se incluyeron comentarios explicativos dentro del código fuente para facilitar su comprensión.
 3. **Pruebas de ejecución**
El programa fue compilado y ejecutado en el IDE, utilizando los mismos casos definidos en las pruebas de escritorio. Se verificó que los resultados obtenidos coincidieran con las salidas esperadas, lo que confirmó la validez del algoritmo.
 4. **Documentación del proceso**
Se elaboró un informe técnico en formato PDF, siguiendo la estructura propuesta en la guía. El documento incluye el análisis del problema, el diseño del algoritmo, la codificación, las pruebas realizadas y una reflexión crítica sobre el proceso de aprendizaje.

5. Resultados

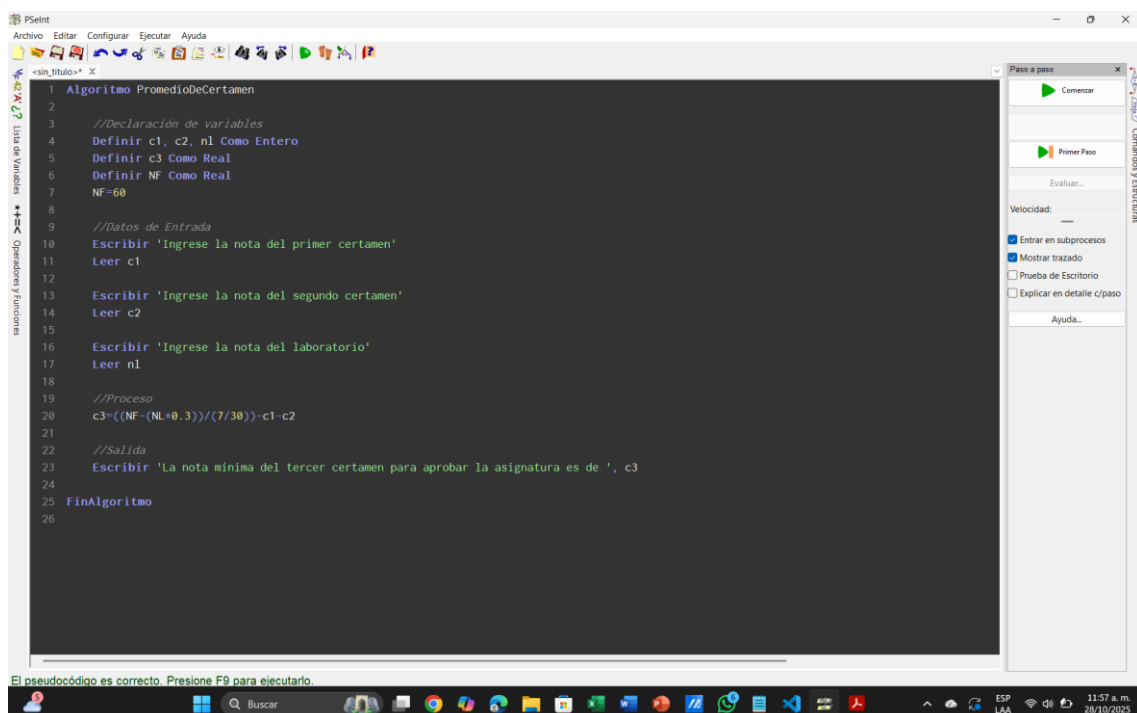
- Análisis del problema:**

Para elaborar el algoritmo en pseudocódigo, fue necesario realizar previamente un análisis detallado del problema, identificando las variables y constantes ($c1$, $c2$, $c3$, nl , NF), los **datos de entrada** ($c1$, $c2$, nl), el **proceso** (operación matemática para despejar $C3$) y la **salida esperada** (nota mínima requerida en $C3$).

A su vez se realizó el análisis mediante la deducción de la ecuación matemática que permite calcular la nota requerida en el tercer certamen ($C3$) para alcanzar la calificación final deseada:

$$\begin{aligned}
 NC &= \frac{c_1 + c_2 + c_3}{3} \\
 NF &= (NC \cdot 0.7) + (NL \cdot 0.3) \\
 NF &= \left[\left(\frac{c_1 + c_2 + c_3}{3} \right) \cdot 0.7 \right] + (NL \cdot 0.3) \\
 NF &= \left[(c_1 + c_2 + c_3) \frac{1}{3} \left(\frac{7}{10} \right) \right] + (NL \cdot 0.3) \\
 NF - (NL \cdot 0.3) &= \left[(c_1 + c_2 + c_3) \left(\frac{7}{30} \right) \right] \\
 c_3 &= \left(\frac{NF - (NL \cdot 0.3)}{\frac{7}{30}} \right) - c_1 - c_2
 \end{aligned}$$

- Pseudocódigo en PSeInt**

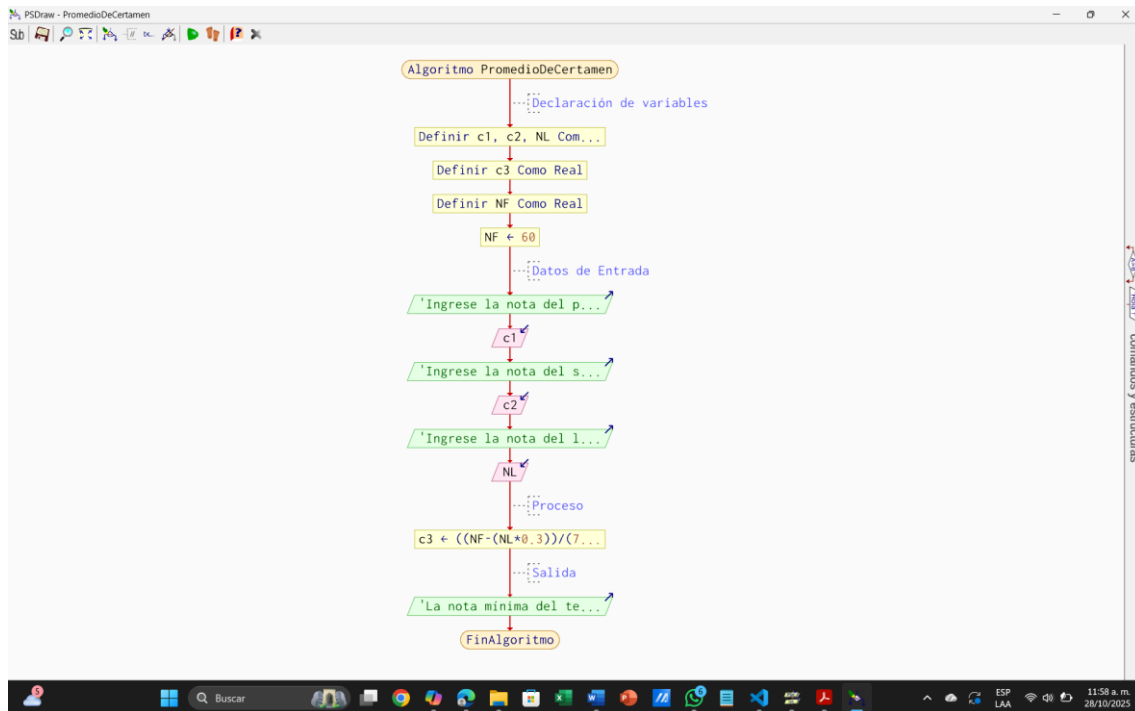


```

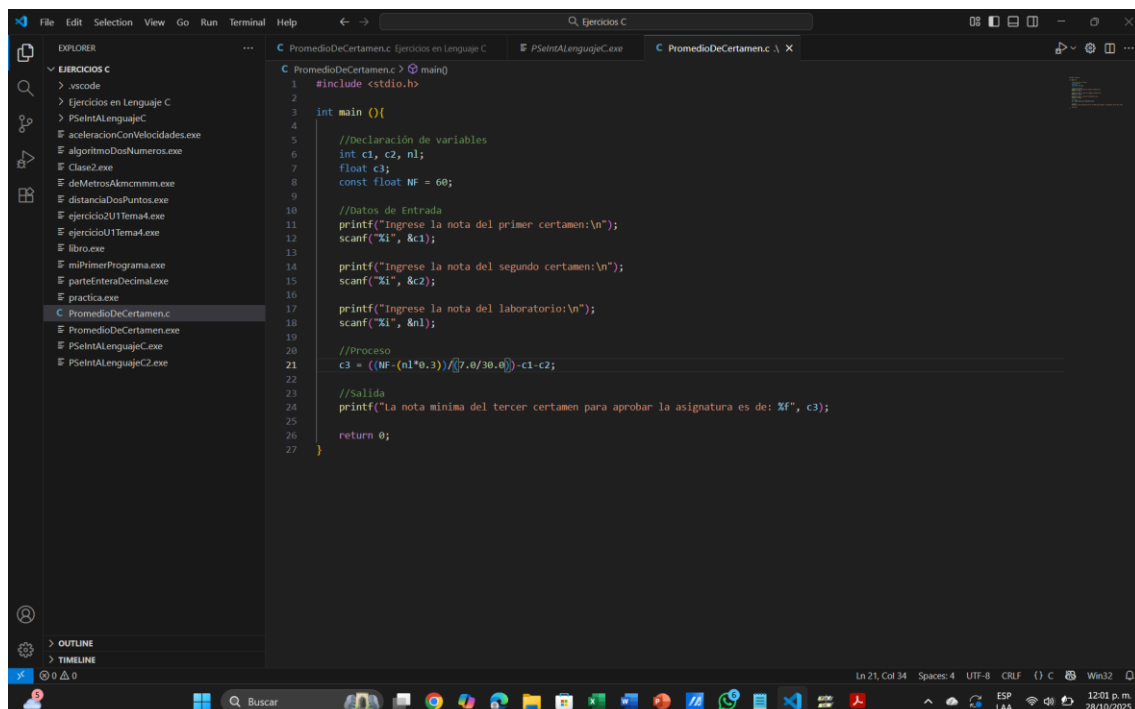
1 Algoritmo PromedioDeCertamen
2
3 //Declaración de variables
4 Definir c1, c2, nl Como Entero
5 Definir c3 Como Real
6 Definir NF Como Real
7 NF=60
8
9 //Datos de Entrada
10 Escribir 'Ingrese la nota del primer certamen'
11 Leer c1
12
13 Escribir 'Ingrese la nota del segundo certamen'
14 Leer c2
15
16 Escribir 'Ingrese la nota del laboratorio'
17 Leer nl
18
19 //Proceso
20 c3=((NF-(NL*0.3))/(7/30))-c1-c2
21
22 //Salida
23 Escribir 'La nota mínima del tercer certamen para aprobar la asignatura es de ', c3
24
25 FinAlgoritmo
26
  
```

El pseudocódigo es correcto. Presione F9 para ejecutarlo.

- Diagrama de Flujo en PSeInt



- Código fuente en lenguaje C



```

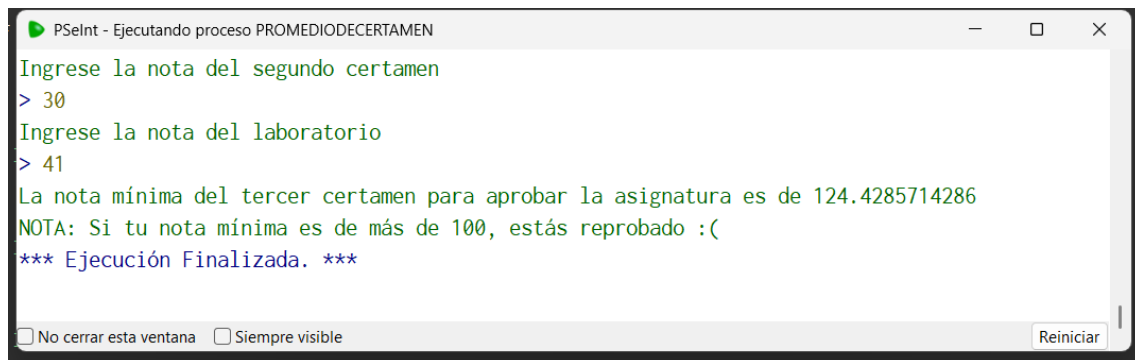
1 #include <stdio.h>
2
3 int main ()
4 {
5     //Declaración de variables
6     int c1, c2, nl;
7     float c3;
8     const float NF = 60;
9
10    //Datos de Entrada
11    printf("Ingrese la nota del primer certamen:\n");
12    scanf("%i", &c1);
13
14    printf("Ingrese la nota del segundo certamen:\n");
15    scanf("%i", &c2);
16
17    printf("Ingrese la nota del laboratorio:\n");
18    scanf("%i", &nl);
19
20    //Proceso
21    c3 = ((NF-(nl*0.3))/(7.0/30.0))-c1-c2;
22
23    //Salida
24    printf("La nota minima del tercer certamen para aprobar la asignatura es de: %f", c3);
25
26    return 0;
27 }
  
```

• PRUEBAS DE ESCRITORIO

Datos de Entrada			Constante	Proceso	Datos de Salida
Variable 1(c1)	Variable 2(c2)	Variable 3(nl)	Constante 1 (NF)	Proceso $c3=((NF-(NL*0.3)) / (7/30))-c1-c2$	Resultado (c3)
50	30	41	60	124.43	124.43
70	85	75	60	5.71	5.71
100	100	100	60	-71.43	-71.43

• EJECUCIÓN DEL ALGORITMO EN PSEINT Y EN C

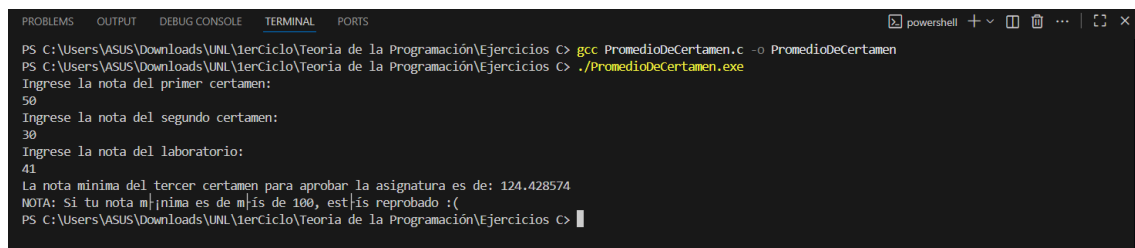
- Caso 1



```

PSeInt - Ejecutando proceso PROMEDIODECERTAMEN
Ingrese la nota del segundo certamen
> 30
Ingrese la nota del laboratorio
> 41
La nota mínima del tercer certamen para aprobar la asignatura es de 124.4285714286
NOTA: Si tu nota mínima es de más de 100, estás reprobado :(
*** Ejecución Finalizada. ***
☐ No cerrar esta ventana ☐ Siempre visible Reiniciar

```

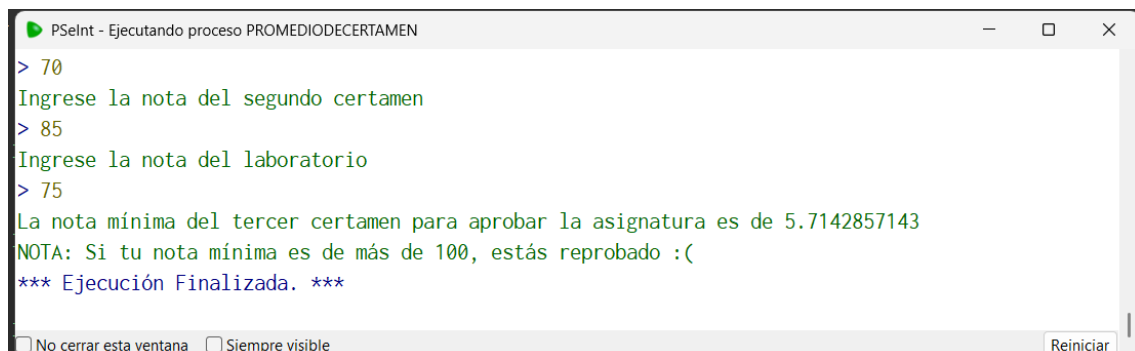


```

PS C:\Users\ASUS\Downloads\UNL\terCiclo\Teoria de la Programación\Ejercicios> gcc PromedioDeCertamen.c -o PromedioDeCertamen
PS C:\Users\ASUS\Downloads\UNL\terCiclo\Teoria de la Programación\Ejercicios> ./PromedioDeCertamen.exe
Ingrese la nota del primer certamen:
50
Ingrese la nota del segundo certamen:
30
Ingrese la nota del laboratorio:
41
La nota mínima del tercer certamen para aprobar la asignatura es de: 124.428574
NOTA: Si tu nota mínima es de más de 100, estás reprobado :(
PS C:\Users\ASUS\Downloads\UNL\terCiclo\Teoria de la Programación\Ejercicios>

```

- Caso 2



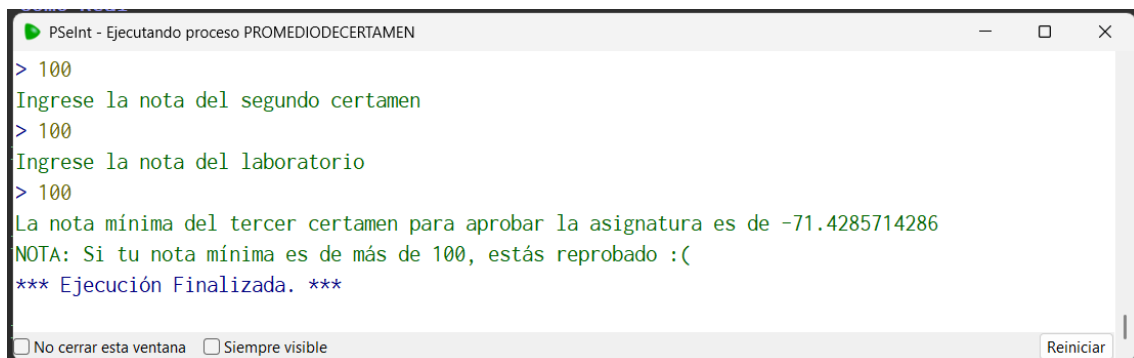
```

PSeInt - Ejecutando proceso PROMEDIODECERTAMEN
> 70
Ingrese la nota del segundo certamen
> 85
Ingrese la nota del laboratorio
> 75
La nota mínima del tercer certamen para aprobar la asignatura es de 5.7142857143
NOTA: Si tu nota mínima es de más de 100, estás reprobado :(
*** Ejecución Finalizada. ***
☐ No cerrar esta ventana ☐ Siempre visible Reiniciar

```

```
PS C:\Users\ASUS\Downloads\UNL\1erCiclo\Teoria de la Programación\Ejercicios C> ./PromedioDeCertamen.exe
Ingrese la nota del primer certamen:
70
Ingrese la nota del segundo certamen:
85
Ingrese la nota del laboratorio:
75
La nota mínima del tercer certamen para aprobar la asignatura es de: 5.714286
NOTA: Si tu nota mínima es de más de 100, estás reprobado :(
PS C:\Users\ASUS\Downloads\UNL\1erCiclo\Teoria de la Programación\Ejercicios C>
```

- Caso 3



```
PSeInt - Ejecutando proceso PROMEDIODECERTAMEN
> 100
Ingrese la nota del segundo certamen
> 100
Ingrese la nota del laboratorio
> 100
La nota mínima del tercer certamen para aprobar la asignatura es de -71.4285714286
NOTA: Si tu nota mínima es de más de 100, estás reprobado :(
*** Ejecución Finalizada. ***
☐ No cerrar esta ventana ☐ Siempre visible Reiniciar
```

```
PS C:\Users\ASUS\Downloads\UNL\1erCiclo\Teoria de la Programación\Ejercicios C> ./PromedioDeCertamen.exe
Ingrese la nota del primer certamen:
100
Ingrese la nota del segundo certamen:
100
Ingrese la nota del laboratorio:
100
La nota mínima del tercer certamen para aprobar la asignatura es de: -71.428571
NOTA: Si tu nota mínima es de más de 100, estás reprobado :(
PS C:\Users\ASUS\Downloads\UNL\1erCiclo\Teoria de la Programación\Ejercicios C>
```

6. Preguntas de Control

• ¿Qué elementos deben identificarse en el análisis de un problema computacional?

Para realizar un análisis correcto de un problema computacional, es necesario identificar varios elementos clave. En primer lugar, se deben reconocer las **variables y constantes** involucradas, determinando cuáles de ellas corresponden a **datos de entrada** (información que el usuario proporciona al sistema) y cuáles representan **datos de salida** (resultados generados).

Además, es necesario analizar el **proceso lógico** o los pasos intermedios que deben ejecutarse para resolver el problema y que se deben llevar a cabo en el algoritmo. Este análisis incluye la identificación de **operaciones matemáticas** que se aplican, así como la secuencia en que deben realizarse.

Otro aspecto importante es determinar el **tipo de datos** que se manejan, como por ejemplo si son **numéricos** (enteros, decimales) o **cadenas de texto**, ya que esto influye directamente en la forma en que se codifica el algoritmo y en el uso correcto de las estructuras del lenguaje de programación.

- ***¿Por qué es importante validar un algoritmo mediante pruebas de escritorio?***

Porque permite comprobar si la **lógica matemática** aplicada en el diseño es correcta antes de implementarlo en un lenguaje de programación. A través de esta técnica, se simula manualmente, o en este caso, en una tabla de Word, la ejecución del algoritmo utilizando diferentes datos de entrada, para verificar que los resultados obtenidos coincidan con los esperados

Además, facilita la comprensión de la secuencia del algoritmo y fortalece el razonamiento computacional asegurando que la solución propuesta sea eficiente.

- ***¿Cómo se traslada un algoritmo en pseudocódigo a un lenguaje de programación?***

Para hacer el traslado, se debe conocer previamente la **sintaxis específica** del lenguaje al que se va a convertir, ya que cada lenguaje tiene reglas propias para declarar variables, estructuras de control, etc.

También hay que identificar qué elementos del pseudocódigo requieren ajustes o adaptaciones, como el uso de **tipos de datos** y como se maneja la entrada y salida de información. Además, es importante considerar las **librerías necesarias** para ejecutar ciertas operaciones. En otros aspectos como la **declaración de variables**, se debería conocer el uso correcto de **comentarios** para documentar el código, y la organización del programa.

7. Conclusiones

1. A través de la actividad planteada, se logró desarrollar la capacidad de transformar un problema cotidiano en una solución computacional. Esta habilidad resulta fundamental en el proceso de formación como programadora, ya que permite abordar situaciones reales desde una perspectiva lógica y estructurada, fortaleciendo el pensamiento algorítmico.
2. Gracias a los conocimientos adquiridos en la unidad, fue posible aplicar correctamente **estructuras secuenciales** en el diseño del algoritmo. Esto permitió comprender y dominar el manejo de procesos, datos de entrada y salida, tanto en pseudocódigo como en el lenguaje de programación C, consolidando así una base sólida para futuros desarrollos.
3. Durante la práctica se identificó la relevancia de **validar la lógica del algoritmo mediante pruebas de escritorio**. Esta etapa es esencial para detectar errores, confirmar la coherencia de los cálculos y asegurar que el algoritmo responda adecuadamente a distintos escenarios antes de su implementación definitiva.
4. Se logró **implementar y ejecutar** la solución en un lenguaje de programación, demostrando que una correcta traducción del pseudocódigo permite conservar la lógica original del algoritmo. Esto garantiza que el programa funcione adecuadamente en el entorno de desarrollo elegido, cumpliendo con los objetivos propuestos en la práctica.



8. Recomendaciones

1. Se recomienda realizar pruebas de escritorio en todo algoritmo, especialmente cuando se involucran **operaciones matemáticas**, ya que, aunque el programa pueda generar un resultado, este no siempre será el esperado. Las pruebas permiten verificar que la lógica aplicada sea correcta, detectar posibles errores y asegurar que el algoritmo funcione adecuadamente en distintos escenarios. Esta validación previa es clave para garantizar la confiabilidad de la solución antes de su implementación definitiva.
2. Se recomienda verificar previamente las operaciones matemáticas involucradas antes de diseñar el algoritmo o trasladarlo a un lenguaje de programación. Esto permite asegurar que la lógica aplicada sea correcta y que los resultados obtenidos correspondan con lo esperado, evitando errores durante la ejecución del programa.

9. Anexos

No hay anexos en este informe