# Reporte Técnico de Actividades Práctico-Experimentales Nro. 002

## 1. Datos de Identificación del Estudiante y la Práctica

| | |
|---|---|
| **Nombre del estudiante(s)** | Irvin Alexey Armijos Guerra |
| **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** | Lissette Geoconda López Faicán |
| **Fecha** | 28/10/2025 |
| **Horario** | 10h30 – 13h30 |
| **Lugar** | Aula física asignada al paralelo |
| **Tiempo planificado en el Sílabo** | 6 horas |

## 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.

## 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)..

## Procedimiento / Metodología Ejecutada

**Metodología de aprendizaje:** aprendizaje basado en problemas.

**Inicio**

● Presentación del objetivo de la práctica: explicación de los propósitos formativos de la actividad y la relevancia de aplicar estructuras secuenciales en el diseño de algoritmos.
● Contextualización del problema a resolver: explicación del enunciado que plantea la situación práctica.

**Problema**

Un estudiante necesita saber qué calificación debe obtener en el tercer certamen (C3) para aprobar la asignatura con una nota final de 60/100 puntos.

Para calcular la nota final se utilizan las siguientes fórmulas:

**1. Promedio de certámenes (NC):**

$$NC = \frac{C1 + C2 + C3}{3}$$

**2. Nota final del ciclo (NF):**

$$NF = (NC \cdot 0.7) + (NL \cdot 0.3)$$

**Donde:**

● C1 y C2 son las notas de los dos primeros certámenes.
● C3 es la nota del tercer certamen (la que se debe calcular).
● NL es la nota de laboratorio.
● NF es la nota final de la asignatura, mínima para pasar es 60/100.

El programa debe permitir ingresar las notas de C1, C2 y NL; calcular automáticamente la nota mínima necesaria en C3 para que el estudiante apruebe la asignatura.

**Nota:** Si el resultado es negativo, significa que ya aprueba con las notas actuales.

**Análisis del problema**

**-Datos de entrada**
**C1 →** Nota del primer certamen
**C2 →** Nota del segundo certamen
**NL →** Nota de laboratorio

**-Proceso**
Aplicar la fórmula despejada para encontrar C3:

1.Remplazamos NC en la segunda formula:

$$NC = \frac{C1 + C2 + C3}{3}$$

$$NF = (NC \cdot 0.7) + (NL \cdot 0.3)$$

$$NF = \left(\frac{C1 + C2 + C3}{3} \times 0.7\right) + (NL \times 0.3)$$

2.Despejamos para encontrar C3, donde NF es 60.

$$\boldsymbol{C3 = (((60 - (NL * 0.3))/0.7) * 3) - (C1 + C2)}$$

***-Datos de salida***

*El programa muestra la nota C3 para comprobar si pasa o no la asignatura.*

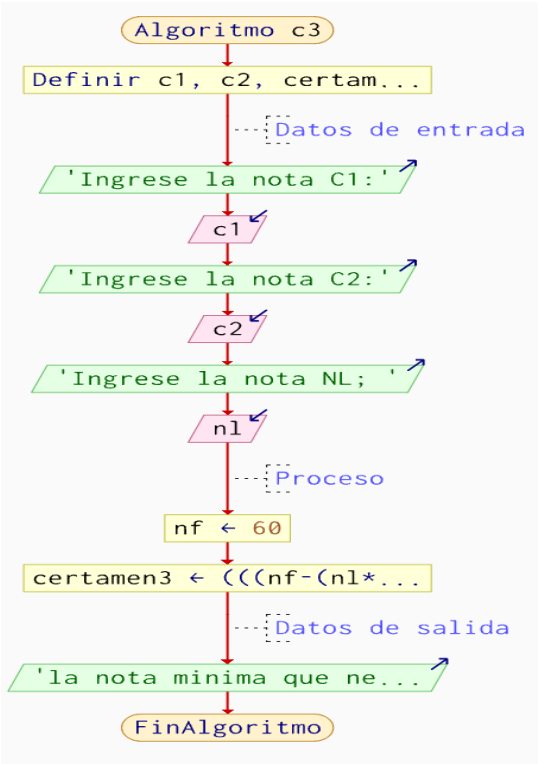
***2. Diseño del algoritmo***

***-Elaborar pseudocódigo en PSeInt con comentarios explicativos.***

***Algoritmo c3***

```
Definir c1,c2,certamen3,nl,nf Como Real

//Datos de entrada
Escribir "Ingrese la nota C1:";
Leer c1;
Escribir "Ingrese la nota C2:";
Leer c2;
Escribir "Ingrese la nota NL; ";
Leer nl;

//Proceso
nf = 60;
certamen3 = (((nf-(nl*0.3))/0.7)*3)-(c1+c2);

//Datos de salida

Escribir "la nota mínima que necesitas de c3 que necesitas para aprobar es: ", certamen3;

FinAlgoritmo
```

***-Diseñar el diagrama de flujo en una herramienta digital.***



***-Definir pruebas de escritorio con al menos 3 casos y su validación con la ejecución del algoritmo.***

| ***Datos de entrada*** | | | ***Proceso*** | ***Salida*** |
|---|---|---|---|---|
| ***C1*** | ***C2*** | ***NL*** | ***Formula*** | ***Nota que necesita sacar en c3 para pasar*** |
| *40* | *55* | *70* | *(((60 - (70×0.3))/0.7)×3) - (40+55) = 72.14* | *C3 = 72.14* |
| *80* | *90* | *85* | *(((60 - (85×0.3))/0.7)×3) - (80+90) = -22.14* | *C3 = -22.14(si C3 es negativo aprueba con las notas actuales)* |
| *10* | *20* | *40* | *(((60 - (40×0.3))/0.7)×3) - (10+20) = 150.71* | *C3 = 175.71(aunque saque 100 no aprueba)* |

***3. Codificación: trasladar la solución a un lenguaje de programación C.***

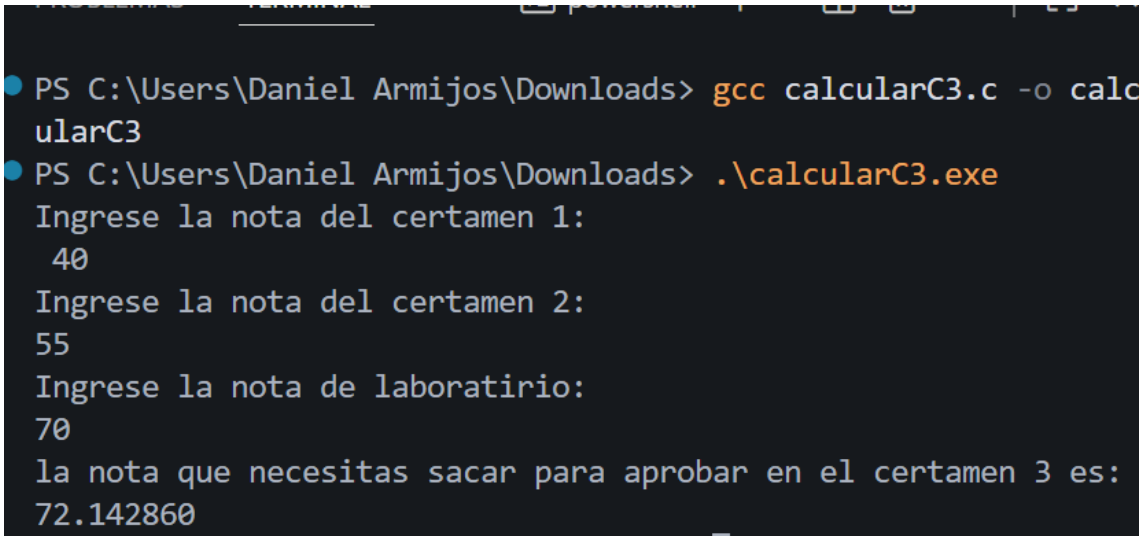
```
#include <stdio.h>

int main(){

   float c1, c2, c3, nl;
   float nf = 60;

   //Datos de entrada
   printf("Ingrese la nota del certamen 1:\n ");
   scanf("%f", &c1);
   printf("Ingrese la nota del certamen 2: \n");
   scanf("%f", &c2);
   printf("Ingrese la nota de laboratirio: \n");
   scanf("%f", &nl);

  //Proceso

   c3 = (((nf-(nl*0.3))/0.7)*3)-(c1+c2);
   //Datos de salida

    printf("la nota que necesitas sacar para aprobar en el certamen 3 es: %f",
c3);


   return 0;
}
```

## Resultados

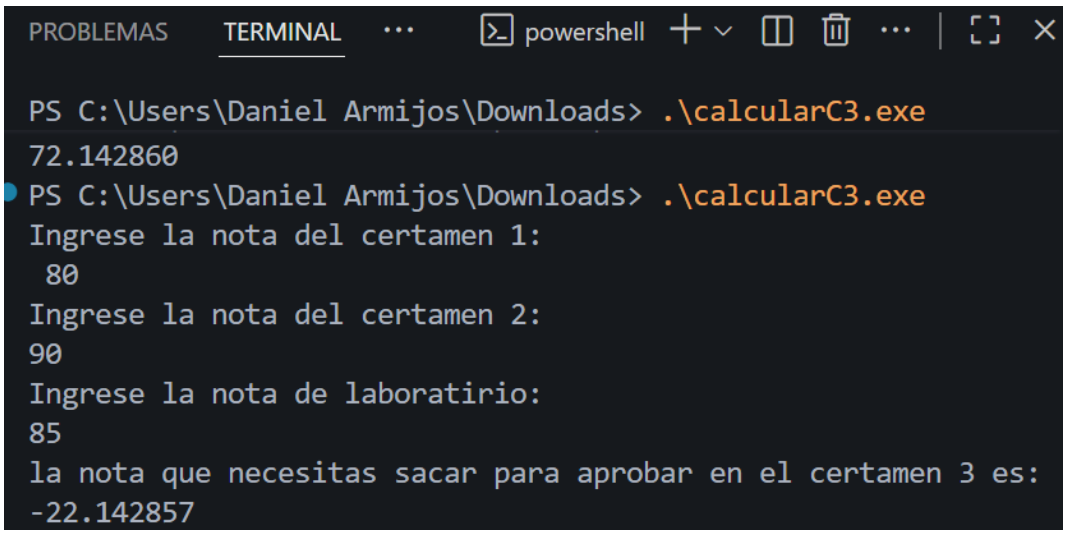
Incluya evidencias del trabajo realizado (tablas, gráficos, capturas de pantalla, registros de ejecución, modelos, programas, informes, etc.).

***Pruebas:*** *compilar y ejecutar el programa en el IDE; verificar que los resultados sean correctos con los mismos casos definidos en las pruebas de escritorio*

***Prueba 1***




**Prueba 2.**



**Prueba 3.**

```
PS C:\Users\Daniel Armijos\Downloads> .\calcularC3.exe
Ingrese la nota del certamen 1:
 10
Ingrese la nota del certamen 2:
20
Ingrese la nota de laboratirio:
40
la nota que necesitas sacar para aprobar en el certamen 3 es:
175.714279
```

## Preguntas de Control

Responda a las preguntas del docente.

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

En el análisis de un problema computacional se deben identificar:

- Entradas (Inputs): Los datos que el programa recibirá.
- Salidas (Outputs): Los resultados que el programa debe producir.
- Procesamiento: Las operaciones y transformaciones necesarias para convertir las entradas en salidas.

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

Las pruebas de escritorio (o "pruebas de escritorio") son importantes porque:

- Detectan errores lógicos antes de implementar el código
- Verifican el flujo del algoritmo paso a paso
- Validan los cálculos intermedios y finales
- Ahorran tiempo de depuración posterior
- Documentan el comportamiento esperado del algoritmo
- Permiten entender mejor el problema y su solución

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

El proceso implica:

1. Estructurar el pseudocódigo en componentes del lenguaje
2. Definir variables con tipos de datos apropiados
3. Traducir estructuras de control (si→if, mientras→while, etc.)
4. Implementar operaciones matemáticas y lógicas
5. Agregar entrada/salida de datos

6. Probar y depurar la implementación

## Conclusiones

La práctica permitió **cumplir el objetivo principal**: desarrollar un algoritmo que calcule automáticamente la nota necesaria en el tercer certamen para aprobar una asignatura. Se concluye que:

- El **análisis matemático previo** es esencial: sin despejar correctamente la fórmula $C3 = \frac{180-0.9\times NL}{0.7} - C1 - C2$, no se podría automatizar el cálculo.

- Las **pruebas de escritorio** verificaron que el algoritmo maneja correctamente tres escenarios:

  ✓ **Caso normal** (necesita nota positiva posible)

  ✓ **Caso favorable** (ya aprueba, resultado negativo)

  ✓ **Caso imposible** (nota requerida > 100)

- La **traducción a código** demostró que un algoritmo bien diseñado en pseudocódigo se implementa directamente en cualquier lenguaje de programación.

- El **pensamiento computacional** aplicado (descomposición, patrones, abstracción) fue clave para resolver el problema sistemáticamente.

## Recomendaciones

Para mejorar la práctica o su aplicación en casos reales, se sugiere:

**1.**Ampliar validaciones en el código: verificar que las notas ingresadas estén en el rango 0-100.

**2.**Incorporar interfaz gráfica para hacer la herramienta más accesible a usuarios no técnicos.

**3.**Generalizar la solución: permitir al usuario definir los porcentajes (70%-30%) y la nota final deseada.

**4.**Agregar funcionalidades adicionales: calcular proyecciones de nota final con diferentes valores de C3.

**5.**Implementar manejo de errores para entradas no numéricas.

**6.**Documentar el código con comentarios que expliquen la fórmula matemática utilizada.

## Anexos

Recursos de apoyo para cada una de las secciones anteriores.