Reporte Técnico de Actividades Práctico-Experimentales Nro. 001

1. Datos de Identificación del Estudiante y la Práctica

Nombre del estudiante	Selena Castillo
Asignatura	Teoría de la programación
Ciclo	1 A
Unidad	3
Resultado de aprendizaje de la unidad	Desarrolla aplicaciones utilizando el principio de la programación modular y estructuras de datos simples y/o estáticas compuestas, bajo los principios de solidaridad, transparencia, responsabilidad y honestidad.
Práctica Nro.	001
Tipo	Individual
Título de la Práctica	Construcción de funciones y procedimientos en un lenguaje de programación.
Nombre del Docente	Lissette Geoconda López Faicán
Fecha	Jueves 8 de enero del 2026 Jueves 15 de enero del 2026
Horario	10h30 – 13h30
Lugar	Aula física asignada al paralelo.
Tiempo planificado en el Sílabo	6 horas

2. Objetivo(s) de la Práctica

- Aplicar los fundamentos de la programación modular mediante la construcción y uso de funciones y procedimientos, para resolver un problema real, garantizando un código estructurado, reutilizable y correctamente documentado.

3. Materiales, Reactivos, Equipos y Herramientas

- 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.

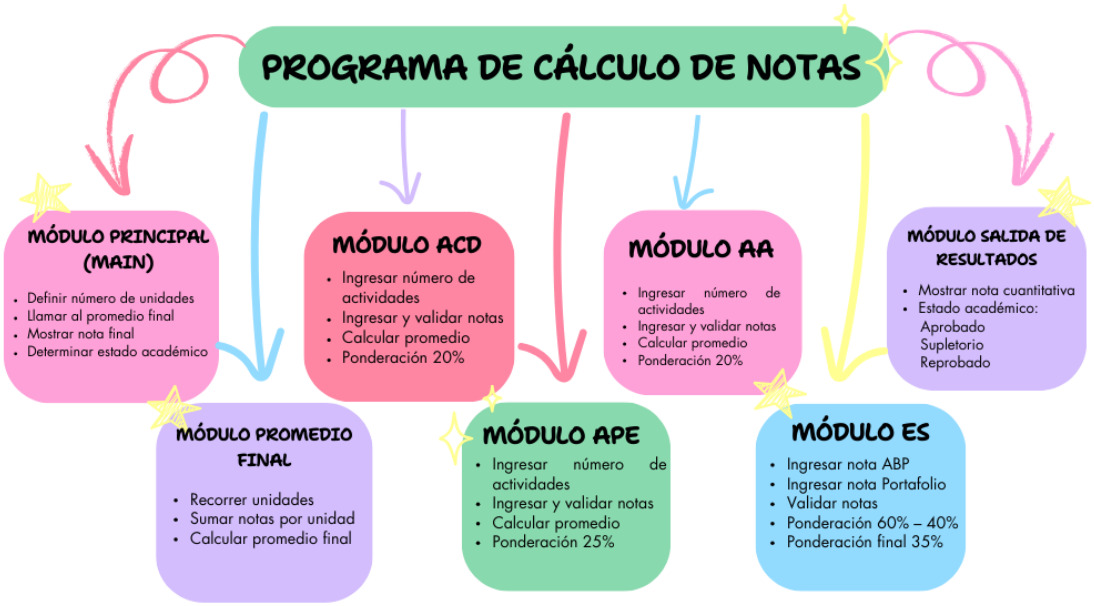
- Conexión a internet estable para acceder a recursos digitales y software en línea.
- Aula física asignada al paralelo.

4. Procedimiento / Metodología Ejecutada

Primeramente, se recibió la explicación correspondiente por parte de la docente sobre el nuevo tema. Posteriormente, se procedió al análisis del problema planteado y a la elaboración del código base en lenguaje C, implementando progresivamente las funciones requeridas. Finalmente, se realizaron las pruebas de escritorio necesarias hasta obtener un programa completamente funcional..

5. Resultados

- **Esquema del sistema**



- **Código en C**

```c
#include <stdio.h>
float calcularPromedioFinal();
float calcularACD();
float calcularAPE();
float calcularAA();
float calcularES();

int main() {
    int NUMUNIDADES = 3;
    float promedioFin;
    promedioFin= calcularPromedioFinal(NUMUNIDADES);
    printf("Su nota final de la Asignatura es: %.2f\n", promedioFin);
    if (promedioFin >= 7){
        printf("Aprobado\n");
    } else if (promedioFin < 7 && promedioFin >= 2.5 ){
        printf("Supletorio\n");
    } else {
        printf("Reprobado\n");
    }
    return 0;
}
```

```c
float calcularPromedioFinal(int nu){ //3
    float notaUnidad;
    float notaSuma = 0;
    float promFinal;

    for (int i=1; i<=nu; i++){ //ingresar 3 veces
        printf("Unidad %i\n", i);
        notaUnidad = calcularACD() + calcularAPE() + calcularAA() + calcularES();
        notaSuma += notaUnidad;
    }
    promFinal= notaSuma/nu;


    return promFinal;
}
```

```c
float calcularACD(){
    int numeroActividades;
    float notaActividad;
    float notaAcumulativa = 0;
    float notaPromedio=0;

    printf("Ingrese el numero de actividades para ACD: ");
    scanf("%i", &numeroActividades);

    for(int i = 1; i <= numeroActividades; i++){
        do {
            printf("Ingrese la nota de la Actividad %d (0 a 10): ", i);
            scanf("%f", &notaActividad);

            if (notaActividad < 0 || notaActividad > 10) {
                printf("Error: la nota debe estar entre 0 y 10.\n");
            }
        } while (notaActividad < 0 || notaActividad > 10);

        notaAcumulativa += notaActividad;
    }

    notaPromedio=(notaAcumulativa/numeroActividades)*0.2;

    return notaPromedio;
}
```

```c
float calcularAPE(){
    int numeroActividades;
    float notaActividad;
    float notaAcumulativa;
    float notaPromedio=0;

    printf("Ingrese el numero de actividades para APE: ");
    scanf("%i", &numeroActividades);

    notaAcumulativa = 0;

    for(int i = 1; i <= numeroActividades; i++){
        do {
            printf("Ingrese la nota de la Actividad %d (0 a 10): ", i);
            scanf("%f", &notaActividad);

            if (notaActividad < 0 || notaActividad > 10) {
                printf("Error: la nota debe estar entre 0 y 10.\n");
            }
        } while (notaActividad < 0 || notaActividad > 10);

        notaAcumulativa += notaActividad;
    }

    notaPromedio=(notaAcumulativa/numeroActividades)*0.25;

    return notaPromedio;
}
```

```c
float calcularAA(){
    int numeroActividades;
    float notaActividad;
    float notaAcumulativa;
    float notaPromedio=0;

    printf("Ingrese el numero de actividades para AA: ");
    scanf("%i", &numeroActividades);

    notaAcumulativa = 0;

   for(int i = 1; i <= numeroActividades; i++){
        do {
            printf("Ingrese la nota de la Actividad %d (0 a 10): ", i);
            scanf("%f", &notaActividad);

            if (notaActividad < 0 || notaActividad > 10) {
                printf("Error: la nota debe estar entre 0 y 10.\n");
            }
        } while (notaActividad < 0 || notaActividad > 10);

        notaAcumulativa += notaActividad;
    }

    notaPromedio=(notaAcumulativa/numeroActividades)*0.2;

    return notaPromedio;
}
```

```c
float calcularES(){
    float notaPromedio;
    float notaPortafolio;
    float notaABP;
    float notaES;

   do {
        printf("Ingrese la nota del ABP (0 a 10): ");
        scanf("%f", &notaABP);
    } while (notaABP < 0 || notaABP > 10);

    do {
        printf("Ingrese la nota del Portafolio (0 a 10): ");
        scanf("%f", &notaPortafolio);
    } while (notaPortafolio < 0 || notaPortafolio > 10);

    notaES=(notaABP*0.6) + (notaPortafolio*0.4);

    notaPromedio= notaES * 0.35;

    return notaPromedio;
}
```

- **Prueba con un caso real**

```
PS C:\Users\HP\Documents\PROGRAMACION\PROYECTO> .\main.exe
Unidad 1
Ingrese el numero de actividades para ACD: 2
Ingrese la nota de la Actividad 1 (0 a 10): 10
Ingrese la nota de la Actividad 2 (0 a 10): 9
Ingrese el numero de actividades para APE: 2
Ingrese la nota de la Actividad 1 (0 a 10): 20
Error: la nota debe estar entre 0 y 10.
Ingrese la nota de la Actividad 1 (0 a 10): 10
Ingrese la nota de la Actividad 2 (0 a 10): 9
Ingrese el numero de actividades para AA: 2
Ingrese la nota de la Actividad 1 (0 a 10): 10
Ingrese la nota de la Actividad 2 (0 a 10): 10
Ingrese la nota del ABP (0 a 10): 9
Ingrese la nota del Portafolio (0 a 10): 9
Unidad 2
Ingrese el numero de actividades para ACD: 2
Ingrese la nota de la Actividad 1 (0 a 10): 9
Ingrese la nota de la Actividad 2 (0 a 10): 10
Ingrese el numero de actividades para APE: 2
Ingrese la nota de la Actividad 1 (0 a 10): 8
Ingrese la nota de la Actividad 2 (0 a 10): 9
Ingrese el numero de actividades para AA: 2
Ingrese la nota de la Actividad 1 (0 a 10): 9.5
Ingrese la nota de la Actividad 2 (0 a 10): 10
Ingrese la nota del ABP (0 a 10): 10
Ingrese la nota del Portafolio (0 a 10): 9
```

```
Unidad 3
Ingrese el numero de actividades para ACD: 3
Ingrese la nota de la Actividad 1 (0 a 10): 10
Ingrese la nota de la Actividad 2 (0 a 10): 10
Ingrese la nota de la Actividad 3 (0 a 10): 9
Ingrese el numero de actividades para APE: 2
Ingrese la nota de la Actividad 1 (0 a 10): 10
Ingrese la nota de la Actividad 2 (0 a 10): 9
Ingrese el numero de actividades para AA: 2
Ingrese la nota de la Actividad 1 (0 a 10): 8
Ingrese la nota de la Actividad 2 (0 a 10): 9
Ingrese la nota del ABP (0 a 10): 9
Ingrese la nota del Portafolio (0 a 10): 9
Su nota final de la Asignatura es: 9.31
Aprobado
```

6. Preguntas de Control

- **¿Cuál es la diferencia entre una función y un procedimiento?**
  Una función devuelve un valor al finalizar; un procedimiento solo ejecuta acciones y no devuelve un valor.
- **¿Qué ventajas aporta dividir un programa en funciones (modularidad)?**
  Hace el programa más ordenado, fácil de entender, mantener y reutilizar.
- **¿Qué se mejoraría del programa si se tuviera que usarlo para varios estudiantes?**
  Se podría reutilizar el código, evitar repetir instrucciones y facilitar cambios para todos los estudiantes a la vez.

7. Conclusiones

- El uso de funciones permitió estructurar mejor el programa y cumplir correctamente con el objetivo planteado.
- La modularidad facilitó la comprensión del código y la detección de errores durante las pruebas.
- El programa desarrollado funciona correctamente y puede adaptarse a diferentes situaciones académicas.

.

8. Recomendaciones.

- Aplicar siempre funciones para dividir el código y mejorar su organización.
- Realizar pruebas de escritorio en cada etapa para asegurar el correcto funcionamiento del programa.
- Adaptar el programa para manejar mayor cantidad de datos si se usa en contextos reales con varios estudiantes.