Reporte Técnico de Actividades Práctico-Experimentales Nro. 001

1. Datos de Identificación del Estudiante y la Práctica

Nombre del estudiante	Kiara Salomé Condoy Morocho
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 15 de enero del 2026
Horario	10h30 – 13h30
Lugar	Aula física asignada al paralelo.
Tiempo planificado en el Sílabo	6 horas

2. Objetivo 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

1. **Análisis y comprensión del problema**
   Se inició con la lectura y contextualización del problema, identificando la necesidad de calcular la nota final de un estudiante a partir de varios componentes evaluativos (ACD, APE, AA y ES). Se analizaron los porcentajes de ponderación y el rango válido de las notas.
2. **Diseño de la solución**
   Se definió una solución basada en programación estructurada y modular, determinando qué procesos debían implementarse como funciones independientes.
3. **Definición de funciones y parámetros**
   Se establecieron las funciones calculoACD, calculoAPE, calculoAA y calculoES, definiendo para cada una los parámetros de entrada, los valores de retorno y las ponderaciones correspondientes. Además, se definió la función promedioFinal para integrar los resultados de todas las unidades.
4. **Codificación del programa**
   Se procedió a la implementación del código en lenguaje C, trasladando la solución diseñada a un programa funcional. Durante esta etapa se aplicaron estructuras de control como bucles y condicionales, y se utilizaron funciones para evitar la repetición de código y mejorar la organización.
5. **Cálculo del promedio por unidad**
   Para cada unidad académica, el programa solicita la cantidad de actividades y las notas correspondientes, calcula los promedios de ACD, APE y AA, incorpora la Evaluación Sumativa y obtiene el promedio total de la unidad mediante la suma de todos los componentes ponderados.
6. **Cálculo del promedio final de la asignatura**
   Una vez procesadas todas las unidades, el programa calcula el promedio general de la asignatura como un promedio simple de las notas obtenidas en cada unidad.
7. **Asignación de la escala cualitativa**
   Con base en la nota final obtenida, se determinó la escala cualitativa correspondiente (Aprobado, Supletorio o Reprobado), de acuerdo con los rangos establecidos en el enunciado del problema.
8. **Pruebas y verificación del programa**
   Se compiló y ejecutó el programa en el IDE, realizando pruebas con datos reales y simulados para verificar el correcto funcionamiento del programa.
9. **Documentación del trabajo**
   Finalmente, se elaboró el presente informe técnico en formato PDF, documentando el análisis del problema, el desarrollo de la solución, el código fuente, las pruebas realizadas y las conclusiones obtenidas.

5. Resultados

5.1. Contextualización del problema

Se requiere desarrollar un programa que calcule la nota final de un estudiante, aplicando funciones y procedimientos para cada componente evaluativo:

- Calcular nota del Aprendizaje en Contacto con el Docente (ACD): solicita nro de actividades, las notas para cada actividad, calcula promedio, y retorna el total ponderado sobre 2.0.
- Calcular nota del Aprendizaje Práctico Experimental (APE): solicita nro de actividades, las notas para cada actividad, calcula promedio, y retorna el total ponderado sobre 2.5.
- Calcular nota del Aprendizaje Autónomo (AA): solicita nro de actividades, las notas para cada actividad, calcula promedio, y retorna el total ponderado sobre 2.0.
- Calcular Evaluación sumativa (ES): solicita la nota de Aprendizaje Basado en Problemas y Portafolio Digital, calcular el ponderado (60% y 40%), y retornar el total ponderado sobre 3.5.
- Calcular promedio por unidad: ACD + APE + AA + ES (retorna el promedio para una unidad).
- Calcular el promedio de la asignatura: promedio simple de acuerdo con el número de unidades con la escala cualitativa.
  - APROBADO: Si la nota final es mayor o igual a 7.
  - SUPLETORIO: Si la nota final es mayor o igual a 2.5 y menor a 7.
  - REPROBADO: Si la nota final es menor a 2.5.
- Las notas deben estar en el rango de 0.0 a 10.0
- Salida de Resultados: Imprimir la nota cuantitativa y cualitativa final del estudiante.
- Requerimiento del código: El programa debe estar bien estructurado, con comentarios y mensajes descriptivos que faciliten su uso.

5.2. Código fuente del programa y llamada a funciones

1. El programa inicia su ejecución en la función main(). En donde, se solicita al usuario ingresar la cantidad de unidades de la asignatura. Luego, main llama a la función promedioFinal(unidades) para calcular la nota final, asignándola a una variable llamada notaFinal.

```
81  int main (){
82      int unidades;
83      float notaFinal;
84      char *escala;
85      printf("Ingrese la cantidad de unidades que tiene: ");
86      scanf("%i", &unidades);
87      notaFinal = promedioFinal(unidades);
```

2. La función promedioFinal inicia un ciclo que se ejecuta por cada unidad. En cada unidad, se solicita al usuario la cantidad de ACD, APE y AA.

```
62  float promedioFinal(int numUnidades){
63      int numACD, numAPE, numAA;
64      float notaUnidad, promedioFinal = 0;
65      for (int i = 1; i <= numUnidades; i++){
66          printf("Ingrese la cantidad de ACD en la unidad %i: ", i);
67          scanf("%i", &numACD);
68
69          printf("Ingrese la cantidad de APE en la unidad %i: ", i);
70          scanf("%i", &numAPE);
71
72          printf("Ingrese la cantidad de AA en la unidad %i: ", i);
73          scanf("%i", &numAA);
74
75          notaUnidad = calculoACD(numACD, i) + calculoAPE(numAPE, i) + calculoAA(numAA, i) + calculoES(i);
76          printf("\nEl promedio de la Unidad %i es: %.2f\n\n", i, notaUnidad);
77          promedioFinal += notaUnidad;
78      }
79      return (promedioFinal/numUnidades);
80  }
```

3. Para calcular la nota de la unidad, promedioFinal llama a las funciones:
   3.1. CalculoACD para calcular y ponderar las actividades ACD

```c
Main.c > main()
#include <stdio.h>
float calculoACD(int numACD, int numUnidades){
    float nota, notaTotal = 0;
    for (int i = 1; i <= numACD; i++){
        do{
            printf("Ingrese la nota del ACD %i de la unidad %i: ", i, numUnidades);
            scanf("%f", &nota);
            if (nota<0.00 || nota>10.0){
                printf("La nota ingresada es invalida\n");
            }
        } while (nota<0.00 || nota>10.0);
        notaTotal+=nota;
    }
    return (notaTotal/numACD)*0.2;
}
```

   3.2. CalculoAPE para calcular y ponderar las actividades APE.

```c
float calculoAPE(int numAPE, int numUnidades){
    float nota, notaTotal = 0;
    for (int i = 1; i <= numAPE; i++){
        do{
            printf("Ingrese la nota del APE %i de la unidad %i: ", i, numUnidades);
            scanf("%f", &nota);
            if (nota<0.00 || nota>10.0){
                printf("La nota ingresada es invalida\n");
            }
        } while (nota<0.00 || nota>10.0);
        notaTotal+=nota;
    }
    return (notaTotal/numAPE)*0.25;
```

   3.3. CalculoAA para calcular y ponderar las actividades AA.

```c
float calculoAA(int numAA, int numUnidades){
    float nota, notaTotal = 0;
    for (int i = 1; i <= numAA; i++){
        do{
            printf("Ingrese la nota del AA %i de la unidad %i: ", i, numUnidades);
            scanf("%f", &nota);
            if (nota<0.00 || nota>10.0){
                printf("La nota ingresada es invalida\n");
            }
        } while (nota<0.00 || nota>10.0);
        notaTotal+=nota;
    }
    return (notaTotal/numAA)*0.2;
}
```

   3.4. CalculoES para calcular y ponderar la evaluación sumativa.

```c
}
float calculoES(int numUnidad){
    float evaluacion, portafolio, notaTotal;
    do{
        printf("Ingrese la nota de la evaluacion de la unidad %i: ", numUnidad);
        scanf("%f", &evaluacion);
        if (evaluacion<0.00 || evaluacion>10.0){
            printf("La nota ingresada es invalida\n");
        }
    } while (evaluacion<0.00 || evaluacion>10.0);
    do{
        printf("Ingrese la nota del portafolio de la unidad %i: ", numUnidad);
        scanf("%f", &portafolio);
        if (portafolio<0.00 || portafolio>10.0){
            printf("La nota ingresada es invalida\n");
        }
    } while (portafolio<0.00 || portafolio>10.0);
    return ((evaluacion*0.6)+(portafolio*0.4))*0.35;
}
```

4. Los resultados de estas funciones se suman para obtener la nota total de la unidad. La nota de cada unidad se acumula para el cálculo del promedio general. Al finalizar todas las unidades, promedioFinal **retorna el promedio final** al main.

```c
float promedioFinal(int numUnidades){
    int numACD, numAPE, numAA;
    float notaUnidad, promedioFinal = 0;
    for (int i = 1; i <= numUnidades; i++){
        printf("Ingrese la cantidad de ACD en la unidad %i: ", i);
        scanf("%i", &numACD);

        printf("Ingrese la cantidad de APE en la unidad %i: ", i);
        scanf("%i", &numAPE);

        printf("Ingrese la cantidad de AA en la unidad %i: ", i);
        scanf("%i", &numAA);

        notaUnidad = calculoACD(numACD, i) + calculoAPE(numAPE, i) + calculoAA(numAA, i) + calculoES(i);
        printf("\nEl promedio de la Unidad %i es: %.2f\n\n", i, notaUnidad);
        promedioFinal += notaUnidad;
    }
    return (promedioFinal/numUnidades);
}
```

5. El main muestra la nota final numérica de la asignatura, main determina la escala cualitativa (Aprobado, Supletorio o Reprobado) según la nota obtenida. El programa finaliza su ejecución.

```c
int main (){
    int unidades;
    float notaFinal;
    char *escala;
    printf("Ingrese la cantidad de unidades que tiene: ");
    scanf("%i", &unidades);
    notaFinal = promedioFinal(unidades);
    printf("\nSu nota final de la asignatura es de: %.2f\n", notaFinal);
    if (notaFinal>=7){
        escala = "Aprobado";
    } else if (notaFinal>= 2.5 && notaFinal<7){
        escala = "Supletorio";
    } else if (notaFinal<2.5){
        escala = "Reprobado";
    }
    printf("Su nota cualitativa es: %s", escala);
    return 0;
}
```

5.3. Pruebas

```
PS C:\Users\ASUS\Downloads\UNL\1erCiclo\Teoria de la Programación\Unidad 3\Ejercicios C\PromedioAsignatura> ./Main.exe
Ingrese la cantidad de unidades que tiene: 3
Ingrese la cantidad de ACD en la unidad 1: 2
Ingrese la cantidad de APE en la unidad 1: 2
Ingrese la cantidad de AA en la unidad 1: 2
Ingrese la nota del ACD 1 de la unidad 1: 10
Ingrese la nota del ACD 2 de la unidad 1: 10
Ingrese la nota del APE 1 de la unidad 1: 10
Ingrese la nota del APE 2 de la unidad 1: -8
La nota ingresada es invalida
Ingrese la nota del APE 2 de la unidad 1: 10
Ingrese la nota del AA 1 de la unidad 1: 10
Ingrese la nota del AA 2 de la unidad 1: 25
La nota ingresada es invalida
Ingrese la nota del AA 2 de la unidad 1: 10
Ingrese la nota de la evaluacion de la unidad 1: 10
Ingrese la nota del portafolio de la unidad 1: 9

El promedio de la Unidad 1 es: 9.86
```

```
Ingrese la cantidad de ACD en la unidad 2: 2
Ingrese la cantidad de APE en la unidad 2: 2
Ingrese la cantidad de AA en la unidad 2: 2
Ingrese la nota del ACD 1 de la unidad 2: 10
Ingrese la nota del ACD 2 de la unidad 2: 10
Ingrese la nota del APE 1 de la unidad 2: 10
Ingrese la nota del APE 2 de la unidad 2: 9.5
Ingrese la nota del AA 1 de la unidad 2: 10
Ingrese la nota del AA 2 de la unidad 2: 8
Ingrese la nota de la evaluacion de la unidad 2: 9.5
Ingrese la nota del portafolio de la unidad 2: 10

El promedio de la Unidad 2 es: 9.63

Ingrese la cantidad de ACD en la unidad 3: 1
Ingrese la cantidad de APE en la unidad 3: 1
Ingrese la cantidad de AA en la unidad 3: 1
Ingrese la nota del ACD 1 de la unidad 3: 10
Ingrese la nota del APE 1 de la unidad 3: 10
Ingrese la nota del AA 1 de la unidad 3: 10
Ingrese la nota de la evaluacion de la unidad 3: 10
Ingrese la nota del portafolio de la unidad 3: -85
La nota ingresada es invalida
Ingrese la nota del portafolio de la unidad 3: 10

El promedio de la Unidad 3 es: 10.00
 Su nota final de la asignatura es de: 9.83
 Su nota cualitativa es: Aprobado
```

6. Preguntas de Control

- **¿Cuál es la diferencia entre una función y un procedimiento?**
  La diferencia es que una función es un bloque de código que realiza una tarea específica y retorna un valor, el cual puede ser utilizado en otras partes del programa, incluyendo la función principal (main). En cambio, un procedimiento también es un bloque de instrucciones que resuelve una tarea determinada, pero no retorna ningún valor; su objetivo es ejecutar acciones dentro del flujo del programa.
- **¿Qué ventajas aporta dividir un programa en funciones (modularidad)?**
  En primer lugar, facilita la lectura y comprensión del código, especialmente cuando un nuevo programador debe trabajar sobre él. Además, permite una mejor organización del programa, haciendo que cada función tenga una responsabilidad específica. Esto ayuda a detectar y corregir errores con mayor facilidad. En programas extensos, la división en módulos permite que el trabajo se distribuya entre varias personas, lo que reduce el tiempo de desarrollo y mejora el mantenimiento del sistema.
- **¿Qué se mejoraría del programa si se tuviera que usarlo para varios estudiantes?**
  Para que el programa sea más flexible y reutilizable, se podría agregar una función que permita indicar la cantidad de estudiantes a promediar. Asimismo, se podrían unificar las actividades en una sola función, utilizando parámetros para evitar la repetición de código. De esta manera, el programa sería más compacto, eficiente y adaptable a diferentes cantidades de estudiantes sin necesidad de modificar gran parte del código.

7. Conclusiones

- La modularidad permite estructurar el programa de una manera más ordenada y eficiente, facilitando su desarrollo y mantenimiento.

- El uso de funciones en el presente trabajo mejoró la comprensión de cada parte del programa, haciendo el código más claro, legible y reduciendo la probabilidad de errores.

8. Recomendaciones

- Antes de iniciar la programación, se recomienda analizar y contextualizar el problema, comprender sus requerimientos y definir previamente las funciones que se utilizarán, así como los valores que cada una retornará.
- Es importante inicializar correctamente las variables en cada una de las funciones que lo requieran, para evitar errores de ejecución.
- Se debe verificar siempre el tipo de valor de retorno de cada función y asegurarse de que las variables utilizadas sean del tipo de dato adecuado, garantizando así un correcto funcionamiento del programa.