Reporte Técnico de Actividades Práctico-Experimentales Nro. 001

1. Datos de Identificación del Estudiante y la Práctica

Nombre del estudiante(s)	Valeria Idaly Agila Gomez
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 o Grupal
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 y Reactivos

- IDE de programación: Visual Studio Code u otro entorno compatible.
- Lenguaje de programación: C (según los contenidos de la unidad).

4. Equipos y Herramientas

- 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

5. Procedimiento / Metodología Ejecutada

Como primer paso, se realizó el análisis del problema planteado, identificando los requerimientos del sistema y la lógica necesaria para el cálculo de la nota final del estudiante, considerando cada uno de los componentes evaluativos y sus respectivas ponderaciones.

Posteriormente, se procedió a la implementación de la solución en el lenguaje de programación C, aplicando el principio de programación modular mediante el uso de funciones específicas para cada componente evaluativo (ACD, APE, AA y ES), lo que permitió desarrollar un código estructurado, claro y reutilizable.

Finalmente, se compiló y ejecutó el programa con distintos datos de prueba, con el objetivo de verificar su correcto funcionamiento, comprobar la ausencia de errores y garantizar que los resultados obtenidos correspondan con los esperados según los criterios establecidos.

6. Resultados

- **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 número 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 número de actividades, las notas para cada actividad, calcula promedio, y retorna el total ponderado sobre 2.5.
  - Calcular nota del Aprendizaje Autónomo (AA): solicita número 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 al 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.
- **Código fuente del programa**
- **Documentación del main y funciones utilizadas**

El programa fue desarrollado en el lenguaje de programación C, aplicando el principio de programación modular, mediante el uso de funciones que permiten calcular las notas parciales y el promedio final del estudiante. Esta estructura facilita la organización del código, su comprensión y su mantenimiento.

- **Función principal (main)**

  La función **main** es el punto de inicio del programa. En esta función se declara la variable **numeroUnidades** con el valor de 3, que representa el número de unidades académicas consideradas en el cálculo del promedio final.

  Posteriormente, se declara la variable **promedioFinal**, en la cual se almacena el valor retornado por la función **calcularPromedioFinal**, a la cual se le envía como parámetro el número de unidades **(numeroUnidades).** Esta llamada permite delegar el cálculo del promedio general a una función específica.

  Una vez obtenido el promedio final, este se muestra en pantalla. A continuación, mediante estructuras condicionales **(if, else if),** se evalúa el valor de **promedioFinal** para determinar el estado académico del estudiante, el cual puede ser APROBADO, SUPLETORIO o REPROBADO, de acuerdo con los rangos establecidos en el programa.

  Finalmente, la función **main** finaliza la ejecución del programa retornando el valor 0, indicando que el proceso se realizó correctamente.

```c
#include <stdio.h>
float calcularPromedioFinal(int nu);
float calcularACD();
float calcularAPE();
float calcularAA();
float calcularES();
int  main(){
    int numeroUnidades = 3;
    float promedioFinal;

    promedioFinal = calcularPromedioFinal(numeroUnidades);

    printf("Promedio Final %.2f\n", promedioFinal);
    printf("\n--- RESULTADOS DEL ESTUDIANTE ---\n");
    if(promedioFinal >= 7.0){
        printf("Estado: APROBADO\n");
    }else if( promedioFinal < 7 && promedioFinal >= 2.5){
        printf("Estado: SUPLETORIO\n");
    }else if(promedioFinal < 2.5){
        printf("Estado: REPROBADO\n");
    }

    return 0;
}
```

- **Función calcularPromedioFinal**

  La función **calcularPromedioFinal** recibe como parámetro la variable nu, la cual indica el número de unidades académicas a evaluar. En este caso, el valor de **nu** es 3.

  Dentro de esta función se utiliza un ciclo **for** que se ejecuta desde 1 hasta **nu**, lo que permite calcular la nota de cada unidad de manera repetitiva y ordenada. En cada iteración del ciclo:

  - Se muestra en pantalla el número de la unidad correspondiente.
  - Se realizan las llamadas a las funciones **calcularACD**, **calcularAPE**, **calcularAA** y **calcularES**, las cuales calculan cada componente evaluativo.
  - La suma de los valores retornados por estas funciones se almacena en la variable **notaUnidad**, la cual representa el promedio de una unidad académica.
  - El valor de **notaUnidad** se acumula en la variable **notaSuma** para posteriormente calcular el promedio general.

  Una vez finalizado el ciclo, se calcula el promedio final dividiendo **notaSuma** para el número de unidades **(nu).** El resultado se almacena en la variable **promFinal** y finalmente se retorna

mediante la instrucción **return promFinal;** a la función **main**, donde es utilizado para mostrar el resultado final del estudiante.

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

    for(int i=1; i<=nu; i++){ //Ingresa 3 veces
        printf("Unidad %i\n", i);

        notaUnidad = calcularACD() + calcularAPE() + calcularAA() + calcularES();
        printf("Promedio de la Unidad %i: %.2f\n", i, notaUnidad);
        notaSuma += notaUnidad;
    }
    promFinal = notaSuma / nu;
    return promFinal;
}
```

- **Función calcularACD**

  La función **calcularACD** solicita al usuario el número de actividades correspondientes al Aprendizaje en Contacto con el Docente. A continuación, mediante un ciclo **for,** se ingresan las notas de cada actividad.

  Cada nota es validada para asegurar que se encuentre dentro del rango de 0.0 a 10.0, evitando el ingreso de valores incorrectos. Las notas válidas se acumulan en la variable **notaAcumulativa**. Al finalizar el ciclo, se calcula el promedio dividiendo la suma total de las notas para el número de actividades y se aplica la ponderación 0.2. El valor obtenido es retornado a la función que realizó la llamada.

```c
float calcularACD(){
    int numActividades;
    float notaActividad;
    float notaAcumulativa;
    float notaPromedio;
    printf("Ingrese el numero de actividades en ACD: ");
    scanf("%i", &numActividades);

    notaAcumulativa = 0;

    for(int i=1; i<=numActividades; i++){

        printf("Ingrese la nota de la actividad %i: ", i);
        scanf("%f", &notaActividad);

        while (notaActividad < 0.0 || notaActividad > 10.0) {
            printf("Nota invalida. Ingrese nuevamente (0 - 10): ");
            scanf("%f", &notaActividad);
        }

        notaAcumulativa += notaActividad;

    }
    notaPromedio = notaAcumulativa / numActividades;
    notaPromedio = notaPromedio *0.2;
    return notaPromedio;
}
```

- **Función calcularAPE**

  La función **calcularAPE** solicita al usuario el número de actividades correspondientes al Aprendizaje Práctico Experimental. A continuación, se inicializa la variable **notaAcumulativa** en cero y, mediante un ciclo for, se ingresan las notas de cada actividad.

  Cada nota ingresada es validada para asegurar que se encuentre dentro del rango de 0.0 a 10.0. Las notas válidas se van sumando en la variable **notaAcumulativa**. Al finalizar el ciclo, se calcula el promedio dividiendo la suma total de las notas para el número de actividades y

se aplica la ponderación 0.25. El valor obtenido es retornado a la función que realizó la llamada.

```
float calcularAPE(){
    int numActividades;
    float notaActividad;
    float notaAcumulativa;
    float notaPromedio;
    printf("Ingrese el numero de actividades en APE: ");
    scanf("%i", &numActividades);

    notaAcumulativa = 0;

    for(int i=1; i<=numActividades; i++){

        printf("Ingrese la nota de la actividad %i: ", i);
        scanf("%f", &notaActividad);
        while (notaActividad < 0.0 || notaActividad > 10.0) {
            printf("Nota invalida. Ingrese nuevamente (0 - 10): ");
            scanf("%f", &notaActividad);

        }

        notaAcumulativa += notaActividad;

    }
    notaPromedio = notaAcumulativa / numActividades;
    notaPromedio = notaPromedio *0.25;
    return notaPromedio;

}
```

- **Función calcularAA**

  La función **calcularAA** solicita al usuario el número de actividades correspondientes al Aprendizaje Autónomo. Luego, se inicializa la variable **notaAcumulativa** en cero y, mediante un ciclo **for**, se ingresan las notas de cada actividad.

  Cada nota es validada para que se encuentre dentro del rango de 0.0 a 10.0. Las notas válidas se acumulan en la variable **notaAcumulativa**. Al finalizar el ciclo, se calcula el promedio de las actividades y se aplica la ponderación 0.2. El valor resultante es retornado para el cálculo del promedio de la unidad.

```
float calcularAA(){
    int numActividades;
    float notaActividad;
    float notaAcumulativa;
    float notaPromedio;
    printf("Ingrese el numero de actividades en AA: ");
    scanf("%i", &numActividades);

    notaAcumulativa = 0;

    for(int i=1; i<=numActividades; i++){

        printf("Ingrese la nota de la actividad %i: ", i);
        scanf("%f", &notaActividad);
        while (notaActividad < 0.0 || notaActividad > 10.0) {
            printf("Nota invalida. Ingrese nuevamente (0 - 10): ");
            scanf("%f", &notaActividad);

        }
        notaAcumulativa += notaActividad;

    }
    notaPromedio = notaAcumulativa / numActividades;
    notaPromedio = notaPromedio *0.2;
    return notaPromedio;

}
```

- **Función calcularES**

La función **calcularES** solicita al usuario la nota del Portafolio Digital y la nota del Aprendizaje Basado en Problemas. Ambas notas son validadas para asegurar que se encuentren dentro del rango permitido.

Posteriormente, se calcula un ponderado interno aplicando el 60 % a la nota del **Aprendizaje Basado en Problemas** y el 40 % a la nota del **Portafolio Digital**. El resultado obtenido se multiplica por la ponderación **0.35** y el valor final es retornado para el cálculo del promedio de la unidad.

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

    printf("Ingrese la nota de Portafolio digital (40 del ES): ");
    scanf("%f", &notaPortafolio);
    while (notaPortafolio < 0.0 || notaPortafolio > 10.0) {
        printf("Nota invalida. Ingrese nuevamente (0 - 10): ");
        scanf("%f", &notaPortafolio);

    }

    printf("Ingrese la nota de Actividad de Aprendizaje Basado en Problemas (60 del ES): ");
    scanf("%f", &notaABP);
    while (notaABP < 0.0 || notaABP > 10.0) {
        printf("Nota invalida. Ingrese nuevamente (0 - 10): ");
        scanf("%f", &notaABP);

    }

    ponderadoInterno = (notaABP * 0.6) + (notaPortafolio * 0.4);
    notaPromedio = (ponderadoInterno * 0.35) ;

    return notaPromedio;
}
```

- **Prueba**

```
Unidad 1
Ingrese el numero de actividades en ACD: 2
Ingrese la nota de la actividad 1: 10
Ingrese la nota de la actividad 2: 10
Ingrese el numero de actividades en APE: 2
Ingrese la nota de la actividad 1: 9.5
Ingrese la nota de la actividad 2: 8
Ingrese el numero de actividades en AA: 2
Ingrese la nota de la actividad 1: 10
Ingrese la nota de la actividad 2: 10
Ingrese la nota de Portafolio digital (40 del ES): 9.75
Ingrese la nota de Actividad de Aprendizaje Basado en Problemas (60 del ES): 10
Promedio de la Unidad 1: 9.65
Unidad 2
Ingrese el numero de actividades en ACD: 2
Ingrese la nota de la actividad 1: 6.5
Ingrese la nota de la actividad 2: 8
Ingrese el numero de actividades en APE: 2
Ingrese la nota de la actividad 1: 10
Ingrese la nota de la actividad 2: 10
Ingrese el numero de actividades en AA: 2
Ingrese la nota de la actividad 1: 10
Ingrese la nota de la actividad 2: 10
Ingrese la nota de Portafolio digital (40 del ES): 9.5
Ingrese la nota de Actividad de Aprendizaje Basado en Problemas (60 del ES): 10
Promedio de la Unidad 2: 9.38
```

```
Unidad 3
Ingrese el numero de actividades en ACD: 2
Ingrese la nota de la actividad 1: 10
Ingrese la nota de la actividad 2: 11
Nota invalida. Ingrese nuevamente (0 - 10): 10
Ingrese el numero de actividades en APE: 2
Ingrese la nota de la actividad 1: 10
Ingrese la nota de la actividad 2: 10
Ingrese el numero de actividades en AA: 2
Ingrese la nota de la actividad 1: 10
Ingrese la nota de la actividad 2: 10
Ingrese la nota de Portafolio digital (40 del ES): 10
Ingrese la nota de Actividad de Aprendizaje Basado en Problemas (60 del ES): 10
Promedio de la Unidad 3: 10.00
Promedio Final 9.68

--- RESULTADOS DEL ESTUDIANTE ---
```

7. Preguntas de Control

- **¿Cuál es la diferencia entre una función y un procedimiento?**
  La diferencia entre una función y un procedimiento radica en el valor que retornan.
  Una función realiza una tarea específica y devuelve un valor al programa mediante la instrucción return, el cual puede ser utilizado en otras partes del código.
  Un procedimiento, en cambio, ejecuta una serie de instrucciones, pero no retorna ningún valor, ya que su objetivo principal es realizar una acción determinada.
- **¿Qué ventajas aporta dividir un programa en funciones (modularidad)?**
  Dividir un programa en funciones permite organizar mejor el código, facilitando su lectura y comprensión. Además, la modularidad hace que el programa sea más fácil de mantener, depurar y modificar, ya que cada función cumple una tarea específica. También favorece la reutilización del código y reduce la repetición de instrucciones.
- **¿Qué se mejoraría del programa si se tuviera que usarlo para varios estudiantes?**
  Si el programa se utilizara para varios estudiantes, se podría mejorar implementando estructuras repetitivas que permitan ingresar los datos de múltiples estudiantes sin reiniciar el programa. Además, se podrían almacenar las notas y promedios en arreglos o estructuras, permitiendo generar reportes individuales o generales, lo que haría al programa más eficiente y escalable.

8. Conclusiones

Se concluye lo siguiente:

- La práctica permitió aplicar de manera adecuada el principio de programación modular, utilizando funciones para dividir el problema en partes más pequeñas y específicas, lo que facilitó la comprensión del funcionamiento general del programa y el desarrollo de un código más ordenado y estructurado.
- La implementación de funciones independientes para el cálculo de cada componente evaluativo contribuyó a una mejor organización y claridad del código, permitiendo identificar fácilmente la lógica de cada proceso y realizar futuras modificaciones sin afectar el funcionamiento completo del programa.
- El programa desarrollado cumple con el objetivo planteado, ya que permite calcular correctamente el promedio final del estudiante, validar los datos ingresados y determinar de forma automática el estado académico, garantizando resultados confiables y acordes a los criterios establecidos.

9. Recomendaciones

Se recomienda lo siguiente:

- Debido a que la programación modular permitió desarrollar un código claro y estructurado, se sugiere continuar aplicando este enfoque en futuros programas, reforzando la planificación previa de las funciones a utilizar.
- Considerando que el uso de funciones facilitó la organización y mantenimiento del código, se recomienda ampliar el programa para manejar más estudiantes, incorporando ciclos y estructuras de datos que permitan reutilizar la lógica implementada.
- Dado que el programa calcula correctamente el promedio final y valida las notas ingresadas, se aconseja realizar pruebas constantes y revisar las ponderaciones aplicadas, con el fin de garantizar la precisión de los resultados en escenarios más complejos.