Reporte Técnico de Actividades Práctico-Experimentales Nro. 001

1. Datos de Identificación del Estudiante y la Práctica

Nombre del estudiante(s)	Josè Daniel Maldonado Rodriguez
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	Miércoles 15 de enero de 2026
Horario	07h30 – 09:30
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.

4. Procedimiento / Metodología Ejecutada

Metodología de aprendizaje: aprendizaje basado en problemas.

- El código se organiza en funciones independientes para cada componente de evaluación (ACD, APE, AA y ES).
- **calcularACD():** Solicita al usuario el número de actividades correspondientes al ACD, permite ingresar las notas de cada actividad validando que estén entre 0 y 10, calcula el promedio de dichas notas y aplica la ponderación del 20%. Finalmente, retorna la nota obtenida para este componente.

- **calcularAPE():** Permite ingresar el número de actividades del APE y las notas respectivas con validación. Calcula el promedio de las actividades ingresadas y aplica la ponderación del 25%, devolviendo la nota correspondiente a este componente.

- **calcularAA():** Solicita el número de actividades del AA, ingresa y valida las notas de cada actividad, calcula el promedio y aplica la ponderación del 20%. Retorna la nota final de este componente.

- **calcularES():** Solicita las notas del portafolio y del examen final, valida que ambas estén dentro del rango permitido, calcula el promedio ponderado interno (40% portafolio y 60% examen) y luego aplica la ponderación del 35%. Retorna la nota del examen sumativo.

- **calcularPromedioFinal(int nu):** Recibe como parámetro el número de unidades. Para cada unidad, llama a las funciones APE, ACD, AA y ES, suma sus resultados y acumula la nota total. Finalmente, calcula y retorna el promedio final de la asignatura.

- En cada función se solicita al usuario el número de actividades y las notas correspondientes. Para garantizar datos correctos, se utilizan bucles while que obligan a ingresar valores válidos dentro del rango de 0 a 10, evitando errores en los cálculos posteriores.
- Las notas ingresadas se acumulan y se calcula el promedio de cada componente. Luego, este promedio se multiplica por su respectivo porcentaje de ponderación establecido en el sistema de evaluación, asegurando que cada actividad aporte correctamente el ponderado final de cada parámetro.
- Se suman los resultados de todos los componentes para cada unidad, se calcula el promedio general de la asignatura y finalmente se muestra en pantalla la nota final junto con el estado académico del estudiante, indicando si aprueba, va a supletorios o reprueba la materia.

5. Resultados

APE1.c

APE1.c > calcularES()

```c
#include <stdio.h>

float calcularACD(){
    int numeroActividddades = 0;
    float notaActividad = 0, notaAcumulativa = 0, promedio = 0, ponderado = 0;

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

    for (int i = 1; i <= numeroActividddades; i++){
    printf("Ingrese la nota de la actividad %i: ", i);
    scanf("%f", &notaActividad);

    while (notaActividad > 10 || notaActividad < 0){
        printf("la nota es invalida, escriba nuevamente\n");
        printf("\n");
        printf("Ingrese la nota de la actividad %i: ", i);
        scanf("%f", &notaActividad);
    }

    notaAcumulativa += notaActividad;
    }

    promedio = notaAcumulativa / numeroActividddades;
    promedio = promedio * 0.2;

    return promedio;

}

float calcularAPE(){
    int numeroActividddades = 0;
    float notaActividad = 0, notaAcumulativa = 0, promedio = 0;

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

    for (int i = 1; i <= numeroActividddades; i++){
        printf("Ingrese la nota de la actividad %i: ", i);
        scanf("%f", &notaActividad);

    while (notaActividad > 10 || notaActividad < 0){
        printf("la nota es invalida, escriba nuevamente\n");
        printf("\n");
        printf("Ingrese la nota de la actividad %i: ", i);
        scanf("%f", &notaActividad);
    }
```

```
48          notaAcumulativa += notaActividad;
49      }
50
51      promedio = notaAcumulativa / numeroActividddades;
52      promedio = promedio * 0.25;
53
54      return promedio;
55  }
56
57  float calcularAA(){
58      int numeroActividddades = 0;
59      float notaActividad = 0, notaAcumulativa = 0, promedio = 0;
60
61      printf("Ingrese el numero de actividaddes para AA: ");
62      scanf("%i", &numeroActividddades);
63
64      for (int i = 1; i <= numeroActividddades; i++){
65          printf("Ingrese la nota de la actividad %i: ", i);
66          scanf("%f", &notaActividad);
67          while (notaActividad > 10 || notaActividad < 0){
68              printf("la nota es invalida, escriba nuevamente\n");
69              printf("\n");
70              printf("Ingrese la nota de la actividad %i: ", i);
71              scanf("%f", &notaActividad);
72          }
73
74          notaAcumulativa += notaActividad;
75      }
76
77      promedio = notaAcumulativa / numeroActividddades;
78      promedio = promedio * 0.2;
79
80      return promedio;
81  }
82
83  float calcularES(){
84      float portafolio, examen, promedio = 0;
85
86      printf("Ingrese la nota del portafolio: ");
87      scanf("%f", &portafolio);
88      while (portafolio > 10 || portafolio < 0){
89          printf("la nota es invalida, escriba nuevamente\n");
90          printf("\n");
91          printf("Ingrese la nota del portafolio : \n");
```

```
92          scanf("%f", &portafolio);
93      }
94
95      printf("Ingrese la nota del examen: ");
96      scanf("%f", &examen);
97
98      while ( examen > 10 || examen < 0){
99          printf("la nota es invalida, escriba nuevamente\n");
100         printf("\n");
101         printf("Ingrese la nota del examen: \n");
102         scanf("%f", &examen);
103         printf("\n");
104     }
105
106     promedio = ((portafolio * 0.4) + (examen * 0.6)) * 0.35;
107     return promedio;
108 }
109
110 float calcularPromedioFinal(int nu){
111     float notaUnidad, notaAcumulada = 0, promedio;
112     for(int i = 1; i <= nu; i++){
113         printf("Unidad %i\n", i);
114         notaUnidad = calcularAPE() + calcularACD() + calcularAA() + calcularES();
115         notaAcumulada += notaUnidad;
116     }
117     promedio = notaAcumulada / nu;
118
119     return promedio;
120 }
121
122 int main() {
123     float promedioFinal;
124     int NUMEROUNIDADES = 3;
125
126     promedioFinal = calcularPromedioFinal(NUMEROUNIDADES);
127
128     printf("Su nota final de la asignatura es : %f\n", promedioFinal);
129
130     if(promedioFinal >= 7){
131     printf("Estado: APROBADO\n");
132     }else if(promedioFinal < 7){
133         printf("Estado: SUPLETORIOS\n");
134     }else if(promedioFinal < 3.5){
135         printf("Estado: REPROBADO\n");
136     }
```

```
137     return 0;
138 }
139
```

```
Unidad 1
Ingrese el numero de actividaddes para APE: 2
Ingrese la nota de la actividad 1: 10
Ingrese la nota de la actividad 2: 9.50
Ingrese el numero de actividaddes para ACD: 2
Ingrese la nota de la actividad 1: 10
Ingrese la nota de la actividad 2: 10
Ingrese el numero de actividaddes para AA: 2
Ingrese la nota de la actividad 1: 9.5
Ingrese la nota de la actividad 2: 8
Ingrese la nota del portafolio: 8
Ingrese la nota del examen: 10
Unidad 2
Ingrese el numero de actividaddes para APE: 2
Ingrese la nota de la actividad 1: 10
Ingrese la nota de la actividad 2:
8
Ingrese el numero de actividaddes para ACD: 2
Ingrese la nota de la actividad 1: 10
Ingrese la nota de la actividad 2: 10
Ingrese el numero de actividaddes para AA: 2
Ingrese la nota de la actividad 1: 10
Ingrese la nota de la actividad 2: 10
Ingrese la nota del portafolio: 7
Ingrese la nota del examen: 8.5
Unidad 3
Ingrese el numero de actividaddes para APE: 2
Ingrese la nota de la actividad 1: 10
Ingrese la nota de la actividad 2: 8
Ingrese el numero de actividaddes para ACD: 2
Ingrese la nota de la actividad 1: 7
Ingrese la nota de la actividad 2: 8
Ingrese el numero de actividaddes para AA: 2
Ingrese la nota de la actividad 1: 10
Ingrese la nota de la actividad 2: 10
Ingrese la nota del portafolio: 10
Ingrese la nota del examen: 10
Su nota final de la asignatura es : 9.224167
Estado: APROBADO
```

6. Preguntas de Control

- **¿Cuál es la diferencia entre una función y un procedimiento?**

La principal diferencia es que una función devuelve un valor al finalizar su ejecución, el cual puede ser utilizado en otras partes del programa, mientras que un procedimiento realiza una tarea específica pero no retorna ningún valor.

- **¿Qué ventajas aporta dividir un programa en funciones (modularidad)?**

Dividir un programa en funciones permite mejorar la organización y legibilidad del código, facilita la detección y corrección de errores, permite reutilizar el código

y facilita su mantenimiento. Además, cada función cumple una tarea específica, lo que simplifica el trabajo colaborativo y la comprensión del programa.

- **¿Qué se mejoraría del programa si se tuviera que usarlo para varios estudiantes?**

Se podría mejorar el programa incorporando estructuras de datos como arreglos para almacenar la información de varios estudiantes, y tener un registro de sus notas.

7. Conclusiones

La práctica permitió aplicar de manera efectiva el uso de funciones en el lenguaje C para resolver un problema real de cálculo de promedios académicos, reforzando la importancia de la modularidad en el desarrollo de programas. La separación del código en funciones facilitó la comprensión de la lógica, el orden del programa y la reutilización de bloques de código.

Además, se evidenció la importancia de la validación de datos de entrada, ya que controlar que las notas se encuentren dentro de un rango permitido evita errores en los cálculos y garantiza resultados correctos y confiables. El uso de estructuras de control como bucles for y while permitió automatizar el ingreso de datos y mejorar la interacción con el usuario.

Finalmente, la práctica fortaleció habilidades fundamentales de programación como el diseño estructurado, el uso adecuado de funciones y la aplicación de ponderaciones, sentando una base sólida para el desarrollo de programas más complejos y escalables en el futuro.

8. Recomendaciones

Se recomienda fortalecer la modularidad del programa mediante la creación de funciones más genéricas que eviten la repetición de código, como una función única para el ingreso y validación de notas, la cual pueda reutilizarse en los distintos componentes de evaluación. Esto permitiría un diseño más limpio, eficiente y fácil de mantener.

Asimismo, es aconsejable definir claramente la responsabilidad de cada función, asegurando que cada módulo realice una sola tarea específica. Esto mejora la legibilidad del código, facilita la detección de errores y permite realizar cambios o mejoras en una función sin afectar al resto del programa.

Finalmente, se recomienda estructurar el programa de manera que pueda escalar fácilmente, por ejemplo, adaptándolo para manejar múltiples estudiantes o diferentes sistemas de evaluación. Un diseño modular bien implementado simplifica estas ampliaciones y promueve buenas prácticas de programación en proyectos de mayor complejidad.

Declaracion del uso de IA: La Inteligencia Artificial (IA) se utilizó como una herramienta de apoyo para la comprensión, análisis y desarrollo de los contenidos abordados en este trabajo. Su aplicación permitió optimizar el tiempo de investigación, facilitar la explicación de conceptos técnicos y mejorar la organización de ideas,