Reporte Técnico de Actividades Práctico-Experimentales Nro. 00X

1. Datos de Identificación del Estudiante y la Práctica

Nombre del estudiante(s)	Mateo Yanangómez
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. FEIRNNR - Carrera de Computación
- Conexión a internet estable para acceder a recursos digitales y software en línea.
- Aula física asignada al paralelo

4. Procedimiento / Metodología Ejecutada

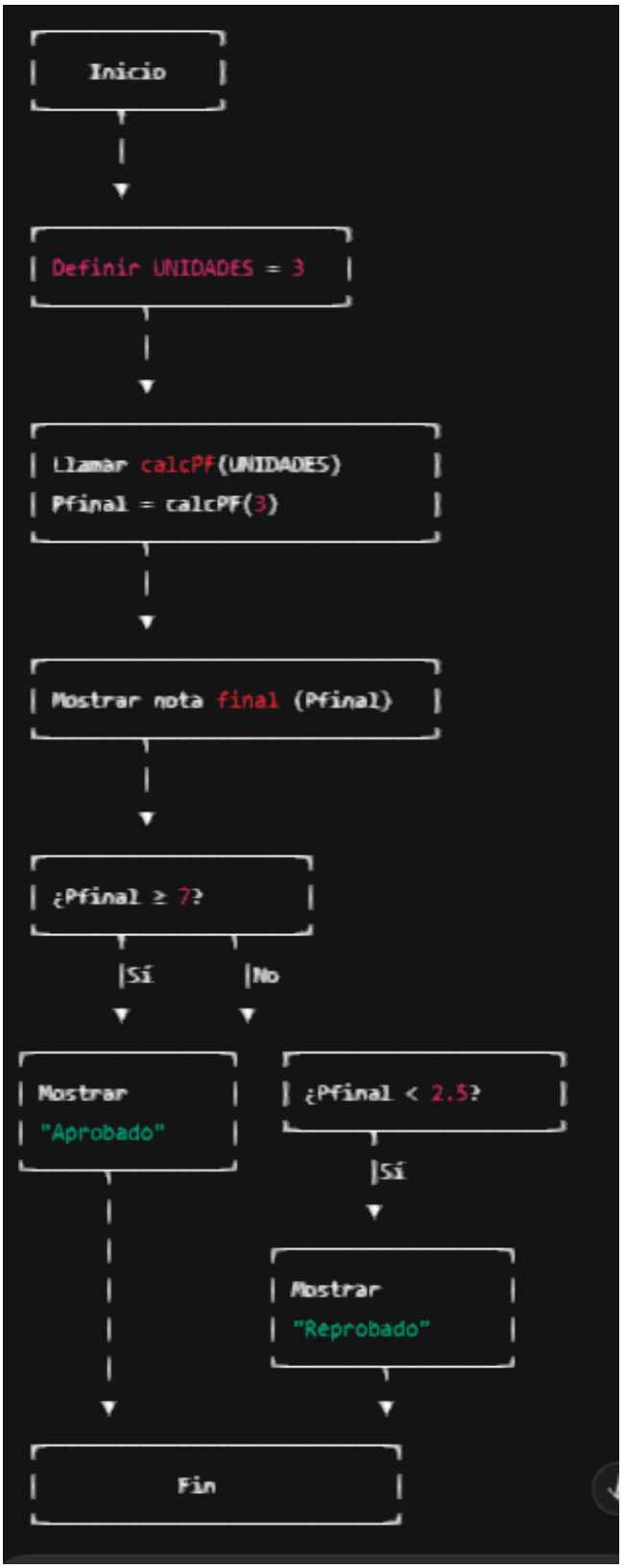
plantear el problema de desarrollar un sistema para calcular la nota final de un estudiante mediante funciones.
Implementar cálculos independientes para el ACD, APE, AA y la evaluación sumativa, aplicando sus respectivos promedios y ponderaciones.
Finalmente obtener el promedio por unidad y el promedio general de la asignatura, determinando si el estudiante aprueba, si debe rendir supletorio o reprueba.

5. Resultados

```c
#include <stdio.h>
float ACD();
float APE();
float AA();
float ES();
float calcPF(int nu);

float calcPF(int nu){
    float NotaU, NS, Pfinal;

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

        NotaU = ACD() + APE() + AA() + ES();
        printf("Promedio de la Unidad %i: %.2f\n", i, NotaU);
        NS += NotaU;
    }
    Pfinal = NS / nu;
    return Pfinal;
}

float ACD(){

    int numact;
    float notaACD;
    float Nacumulada = 0;

    printf("Ingrese el numero de actividades para ACD\n");
    scanf("%i",  &numact);

    for(int i = 1;  i <= numact; i++){
        printf("Ingrese la nota de la actividad %i\n", i);
        scanf("%f", &notaACD);
        while (notaACD < 0.0 || notaACD > 10.0) {
            printf("Nota invalida. Ingrese nuevamente (0 - 10): ");
            scanf("%f", &notaACD);
        }
        Nacumulada += notaACD;
    }
    float Nprom = (Nacumulada / numact);
    Nprom = Nprom * 0.2;

    return Nprom;
}
```

```c
float APE(){

    int numact;
    float notaAPE;
    float Nacumulada = 0;

    printf("Ingrese el numero de actividades para ACD\n");
    scanf("%i",  &numact);

    for(int i = 1;  i <= numact; i++){
        printf("Ingrese la nota de la actividad %i\n", i);
        scanf("%f", &notaAPE);
        while (notaAPE < 0.0 || notaAPE > 10.0) {
            printf("Nota invalida. Ingrese nuevamente (0 - 10): ");
            scanf("%f", &notaAPE);
        }
        Nacumulada += notaAPE;
    }
    float Nprom = (Nacumulada / numact);
    Nprom = Nprom * 0.25;
    return Nprom;
}

float AA(){

    int numact;
    float notaAA;
    float Nacumulada = 0;

    printf("Ingrese el numero de actividades para ACD\n");
    scanf("%i",  &numact);

    for(int i = 1;  i <= numact; i++){
        printf("Ingrese la nota de la actividad %i\n", i);
        scanf("%f", &notaAA);
        while (notaAA < 0.0 || notaAA > 10.0) {
            printf("Nota invalida. Ingrese nuevamente (0 - 10): ");
            scanf("%f", &notaAA);
        }
        Nacumulada += notaAA;
    }
    float Nprom = (Nacumulada / numact);
    Nprom = Nprom * 0.2;
    return Nprom;
}
```

```
91
92  float ES(){
93
94      float Portafolio, prueba, ponderado, Nprom;
95
96      printf("Ingrese la nota de Portafolio digital:\n");
97       scanf("%f", &Portafolio);
98       while (Portafolio < 0.0 || Portafolio > 10.0) {
99           printf("Nota invalida. Ingrese nuevamente (0 - 10): ");
100          scanf("%f", &Portafolio);
101
102      }
103
104      printf("Ingrese la nota de la prueba de Unidad:\n");
105      scanf("%f", &prueba);
106      while (prueba < 0.0 || prueba > 10.0) {
107          printf("Nota invalida. Ingrese nuevamente (0 - 10): ");
108          scanf("%f", &prueba);
109      }
110
111      ponderado = (prueba * 0.6) + (Portafolio * 0.4);
112      Nprom = (ponderado * 0.35) ;
113
114      return Nprom;
115
116  }
117
118  int main(){
119
120      int UNIDADES = 3;
121      float Pfinal;
122      Pfinal = calcPF(UNIDADES);
123      printf("Su nota final de asignatura es %f\n", Pfinal);
124
125      if(Pfinal >= 7 ){
126          printf("Usted ha aprobado con %f :D\n");
127      }else if(Pfinal < 2.5){
128          printf(" Usted a reprobado con %f c\n");
129      }
130      return 0;
131  }
```



Documentación de las llamadas desde el main hacia las otras funciones de código fuente:

Definición de variable Unidades para mostrar las veces que se repetirá la acción

Llamada a la función calcular promedio, donde esta se conecta a la función ACD, APE, AA, ES donde se hacen los cálculos para cada distribución con sus respectivos ponderados

Imprimir o mostrar la Nota final de las Unidades

Muestra para decidir si aprueba las Unidades, queda en supletorios o reprueba ciclo.

```
1
Ingrese la nota de la actividad 1
9
Ingrese el numero de actividades para ACD
1
Ingrese la nota de la actividad 1
9
Ingrese el numero de actividades para ACD
1
Ingrese la nota de la actividad 1
9
Ingrese la nota de Portafolio digital:
9
Ingrese la nota de la prueba de Unidad:
9
Promedio de la Unidad 1: 9.00
Unidad 2
Ingrese el numero de actividades para ACD
1
Ingrese la nota de la actividad 1
9
Ingrese el numero de actividades para ACD
1
Ingrese la nota de la actividad 1
9
Ingrese el numero de actividades para ACD
1
Ingrese la nota de la actividad 1
9
Ingrese la nota de Portafolio digital:
9
Ingrese la nota de la prueba de Unidad:
9
Promedio de la Unidad 2: 9.00
Unidad 3
Ingrese el numero de actividades para ACD
1
Ingrese la nota de la actividad 1
9
Ingrese el numero de actividades para ACD
1
Ingrese la nota de la actividad 1
9
Ingrese el numero de actividades para ACD
1
Ingrese la nota de la actividad 1
9
Ingrese la nota de Portafolio digital:
9
Ingrese la nota de la prueba de Unidad:
9
Promedio de la Unidad 3: 9.00
Su nota final de asignatura es 9.000000
Usted ha aprobado con 9.000000 :D
```

6. Preguntas de Control

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

La función puede mostrar y ejecutar acciones al programa que lo invoca devolviendo datos pero después devuelven estos datos al lugar de donde fue llamada, mientras que un procedimiento puede ejecutar y mostrar datos de igual manera pero no devuelve ni envía resultados en otros programas es decir no puede ser llamado.

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

Permite más eficiencia y control en el orden del código para poder realizar cambios de manera más fácil en el futuro, además de que se llega a resultados o resolución de problemas más eficazmente.

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

Implementar un apartado donde permita ingresar el número de estudiantes de los que se requiere saber las notas y promedios de Unidad.

.

7. Conclusiones

- La programación modular facilita la organización del código al dividir el problema en funciones y procedimientos reutilizables, mejorando su claridad y mantenimiento.
- El uso correcto de funciones y procedimientos permite desarrollar soluciones estructuradas, eficientes y bien documentadas para resolver problemas reales.

8. Recomendaciones

- Planificar el uso de funciones antes de programar para lograr un código más ordenado y modular.

- Reutilizar funciones en lugar de repetir código, evitando errores y mejorando la eficiencia.

- Documentar correctamente las funciones para facilitar su comprensión y mantenimiento en proyectos reales.