# Reporte Técnico de Actividades Práctico-Experimentales Nro. 001

## 1. Datos de Identificación del Estudiante y la Práctica

| | |
|---|---|
| **Nombre del estudiante(s)** | José Valencia |
| **Asignatura** | Teoría de la programación |
| **Ciclo** | 1 |
| **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<br>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

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

**CODIFICACIÓN:**

Incluimos la librería "studio.h".

```
#include <stdio.h>
```

Dentro de la función "main" declaramos variables y almacenamos la función "calcPromedioF" (promedio final) dentro de la variable total, al momento de almacenar esta función también es invocada una vez.

```
int main(void) {

    float total;
    int unidades;

    unidades = 3;

    total = calcPromedioF(unidades);
```

En la función "calcPromedioF" calculamos el promedio final de tres unidades. Cada una se compone de cuatro secciones: AA, ACD, APE y ES. Cada uno esta invocado dentro de esta función y almacenado en notau para poder calcular este promedio final.

```
float calcPromedioF(int num){
    float notau, promediof, acm = 0;
    int i;
    for (i = 1; i <= num; i++) {
        printf("Unidad %d\n", i);
        notau = calcACD() + calcAA() + calcAPE() + calcES();
        printf("Calificacion final de la unidad %d: %.2f\n\n", i, notau);
        acm += notau;
    }
    promediof = acm / num;
    return promediof;
}
```

En la función "calcACD" calculamos el ACD de cada unidad, independientemente de cuantos haya, un acumulador llamado "acm" para poder sacar cada ponderado y devuelve el mismo.

```c
float calcACD(void){
    float acd;
    int i, j;
    float acm, ponderado;

    printf("Ingrese cuantos ACD tiene en su unidad: ");
    scanf("%d", &j);
    getchar();

    acm = 0;

    for(i = 1; i <= j; i++){

        printf("\nACD %d\n", i);

        printf("Introduce la calificacion de tu ACD %d (0-10): ", i);
        scanf("%f", &acd);
        getchar();

        while (acd < 0 || acd > 10) {
            printf("\n¡Error!, la calificacion debe estar entre 0 y 10.\nPor favor, ingrese nuevamente.\n");
            printf("Introduce la calificacion de tu primer ACD (0-10): ");
            scanf("%f", &acd);
            getchar();
        }

        acm += acd;
    }
    acd = acm / j;
    ponderado = acd * 0.2;
    return ponderado;
}
```

En la función “calcAA” calculamos el AA de cada unidad, independientemente de cuantos haya, un acumulador llamado “acm” para poder sacar cada ponderado y devuelve el mismo.

```c
float calcAA(){
    float aa;
    int i, j;
    float acm, ponderado;

    printf("Ingrese cuantos AA tiene en su unidad: ");
    scanf("%d", &j);
    getchar();

    acm = 0;

    for(i = 1; i <= j; i++){

        printf("\nAA %d\n", i);

        printf("Introduce la calificacion de tu AA %d (0-10): ", i);
        scanf("%f", &aa);
        getchar();

        while (aa < 0 || aa > 10) {
            printf("\n¡Error!, la calificacion debe estar entre 0 y 10.\nPor favor, ingrese nuevamente.\n");
            printf("Introduce la calificacion de tu AA %d (0-10): ", i);
            scanf("%f", &aa);
            getchar();
        }

        acm += aa;
    }
    aa = acm / j;
    ponderado = aa * 0.2;
    return ponderado;
}
```

En la función “calcAPE” calculamos el APE de cada unidad, independientemente de cuantos haya, un acumulador llamado “acm” para poder sacar cada ponderado y devuelve el mismo.

```c
float calcAPE(){
    float ape;
    int i, j;
    float acm, ponderado;

    printf("Ingrese cuantos APE tiene en su unidad: ");
    scanf("%d", &j);
    getchar();

    acm = 0;

    for(i = 1; i <= j; i++){

        printf("\nAPE %d\n", i);

        printf("Introduce la calificacion de tu APE %d (0-10): ", i);
        scanf("%f", &ape);
        getchar();

        while (ape < 0 || ape > 10) {
            printf("\n¡Error!, la calificacion debe estar entre 0 y 10.\nPor favor, ingrese nuevamente.\n");
            printf("Introduce la calificacion de tu primer APE (0-10): ");
            scanf("%f", &ape);
            getchar();
        }

        acm += ape;
    }
    ape = acm / j;
    ponderado = ape  * 0.25;
    return ponderado;
}
```

En la función “calcES” calculamos la ES de cada unidad, el cual se basa en un Portafolio Digital y el Aprendizaje Basado en problemas, las guardamos en diferentes variables y devuelve el ponderado calculado.

```c
float calcES(){
    float est, es1, es2;
    float acm, ponderado;

    acm = 0;

    printf("\nES 1\n");

    printf("Introduce la calificacion de tu ES Portafolio Digital (0-10): ");
    scanf("%f", &es1);
    getchar();

    while (es1 < 0 || es1 > 10) {
        printf("\n¡Error!, la calificacion debe estar entre 0 y 10.\nPor favor, ingrese nuevamente.\n");
        printf("Introduce la calificacion de tu ES Portafolio Digital (0-10): ");
        scanf("%f", &es1);
        getchar();
    }
    printf("\nES 2\n");

    printf("Introduce la calificacion de tu ES Aprendizaje Basado en Problemas (0-10): ");
    scanf("%f", &es2);
    getchar();

    while (es2 < 0 || es2 > 10) {
        printf("\n¡Error!, la calificacion debe estar entre 0 y 10.\nPor favor, ingrese nuevamente.\n");
        printf("Introduce la calificacion de tu ES Aprendizaje Basado en Problemas (0-10): ");
        scanf("%f", &es2);
        getchar();
    }

    est = (es1 * 0.4 + es2 * 0.6);
    ponderado = est * 0.35;
    return ponderado;
}
```

Volvemos a la función “main” para imprimir el promedio final, y la calificación cualitativa del estudiante ya que almacenamos una variable para la función “calcPromedioF”. El programa finaliza al llegar a “*return 0*;”.

```
    printf("\nCalificacion final del curso: %.2f\n", total);

    if (total >= 7.0) {
        printf("APROBADO\n");
    } else if(total < 7.0 && total >= 2.5){
        printf("SUPLETORIO\n");
    } else {
        printf("REPROBADO\n");
    }

    return 0;
}
```

**CASO DE PRUEBA:**

```
PS C:\Users\Usuario iTC\Documents\io\c3> gcc ape.c -o ape
PS C:\Users\Usuario iTC\Documents\io\c3> .\ape.exe
Unidad 1
Ingrese cuantos ACD tiene en su unidad: 2

ACD 1
Introduce la calificacion de tu ACD 1 (0-10): 10

ACD 2
Introduce la calificacion de tu ACD 2 (0-10): 10
Ingrese cuantos AA tiene en su unidad: 2

AA 1
Introduce la calificacion de tu AA 1 (0-10): 10

AA 2
Introduce la calificacion de tu AA 2 (0-10): 10
Ingrese cuantos APE tiene en su unidad: 2

APE 1
Introduce la calificacion de tu APE 1 (0-10): 8.5

APE 2
Introduce la calificacion de tu APE 2 (0-10): 8.5

ES 1
Introduce la calificacion de tu ES Portafolio Digital (0-10): 9.5

ES 2
Introduce la calificacion de tu ES Aprendizaje Basado en Problemas (0-10): 9.5
Calificacion final de la unidad 1: 9.45

Unidad 2
Ingrese cuantos ACD tiene en su unidad: 2

ACD 1
Introduce la calificacion de tu ACD 1 (0-10): 8.5

ACD 2
Introduce la calificacion de tu ACD 2 (0-10): 10
Ingrese cuantos AA tiene en su unidad: 2
```

```
AA 1
Introduce la calificacion de tu AA 1 (0-10): 10

AA 2
Introduce la calificacion de tu AA 2 (0-10): 8
Ingrese cuantos APE tiene en su unidad: 2

APE 1
Introduce la calificacion de tu APE 1 (0-10): 10

APE 2
Introduce la calificacion de tu APE 2 (0-10): 9

ES 1
Introduce la calificacion de tu ES Portafolio Digital (0-10): 9.5

ES 2
Introduce la calificacion de tu ES Aprendizaje Basado en Problemas (0-10): 10
Calificacion final de la unidad 2: 9.45

Unidad 3
Ingrese cuantos ACD tiene en su unidad: 2

ACD 1
Introduce la calificacion de tu ACD 1 (0-10): 10

ACD 2
Introduce la calificacion de tu ACD 2 (0-10): 10
Ingrese cuantos AA tiene en su unidad: 2

AA 1
Introduce la calificacion de tu AA 1 (0-10): 10

AA 2
Introduce la calificacion de tu AA 2 (0-10): 10
Ingrese cuantos APE tiene en su unidad: 2

APE 1
Introduce la calificacion de tu APE 1 (0-10): 9

APE 2
Introduce la calificacion de tu APE 2 (0-10): 9

ES 1
Introduce la calificacion de tu ES Portafolio Digital (0-10): 9

ES 2
Introduce la calificacion de tu ES Aprendizaje Basado en Problemas (0-10): 10
Calificacion final de la unidad 3: 9.61


Calificacion final del curso: 9.51
APROBADO
PS C:\Users\Usuario iTC\Documents\io\c3>
```

## 5. Preguntas de Control

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

Una función en C es un bloque de código que realiza una tarea específica y puede ser reutilizado. En este programa usamos funciones para modularizar el cálculo de cada componente evaluativo, lo que hace el código más ordenado, fácil de entender y mantener.

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

La programación modular permite dividir el problema en partes más pequeñas y manejables. Cada función se encarga de un cálculo específico, lo que facilita la lectura del código, la detección de errores y la reutilización en otros programas.

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

Se mejoraría la escalabilidad y la organización de datos, permitiendo que el programa no solo evalúe a un estudiante, sino que gestione a toda una clase de manera eficiente.

## 6. Conclusiones

- El programa desarrollado cumple con el objetivo de calcular la nota final de un estudiante aplicando funciones y procedimientos, lo que demuestra el uso correcto de la programación modular en C.

- Se logró implementar controles de validación para asegurar que las calificaciones ingresadas estén dentro del rango permitido (0–10), garantizando la confiabilidad de los resultados.

- La estructura del código, con funciones específicas para cada componente evaluativo (ACD, AA, APE y ES), facilita la comprensión, el mantenimiento y la reutilización en futuros proyectos.

## 7. Recomendaciones

- Extender el programa para manejar varias unidades y varios estudiantes, para almacenar resultados.

- Agregar comentarios más detallados en el código fuente, explicando la lógica de cada función y las fórmulas utilizadas, lo que facilitará la comprensión por parte de otros usuarios o compañeros.

- Mejorar la interacción con el usuario, implementando menús o mensajes más claros que guíen el proceso de ingreso de datos.