



# Reporte Técnico de Actividades Práctico-Experimentales Nro. 002

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

<b>Nombre del estudiante(s)</b>	José Valencia
<b>Asignatura</b>	Teoría de la programación
<b>Ciclo</b>	1 A
<b>Unidad</b>	2
<b>Resultado de aprendizaje de la unidad</b>	Aplica las estructuras de programación en la resolución de problemas básicos, bajo los principios de solidaridad, transparencia, responsabilidad y honestidad
<b>Práctica Nro.</b>	002
<b>Tipo</b>	Individual
<b>Título de la Práctica</b>	Aplicación de estructuras repetitivas en la resolución de problemas
<b>Nombre del Docente</b>	Lisette Geoconda López Faicán
<b>Fecha</b>	04/12/2025
<b>Horario</b>	10h30 – 13h30
<b>Lugar</b>	Aula física asignada al paralelo.
<b>Tiempo planificado en el Sílabo</b>	6 horas

## 2. Objetivo(s) de la Práctica

- Comprender y aplicar las estructuras repetitivas en la resolución de problemas.
- Diseñar y codificar un algoritmo que utilice bucles para resolver un problema de tipo iterativo.
- Validar el funcionamiento del programa mediante la ejecución práctica.

## 3. Materiales, Reactivos, Equipos y Herramientas

- Herramientas de modelado de diagrama de flujo (Psient, Draw.io, Lucidchart, otros)
- 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

Los pasos que se siguió son:

- Identificar el problema planteado basado en el ejercicio “Cálculo de la nota final de la Unidad 1”.
- Implementar estructuras repetitivas que permita al usuario ingresar el numero de estudiantes a evaluar, y otros que evalúan si los parámetros de calificación están fuera del rango entre 0 y 10.
- Demostrar la estructura condicional del ACDI dentro del diagrama de flujo
- Compilar, ejecutar y realizar los 3 casos de pruebas con notas reales de compañeros de clase.

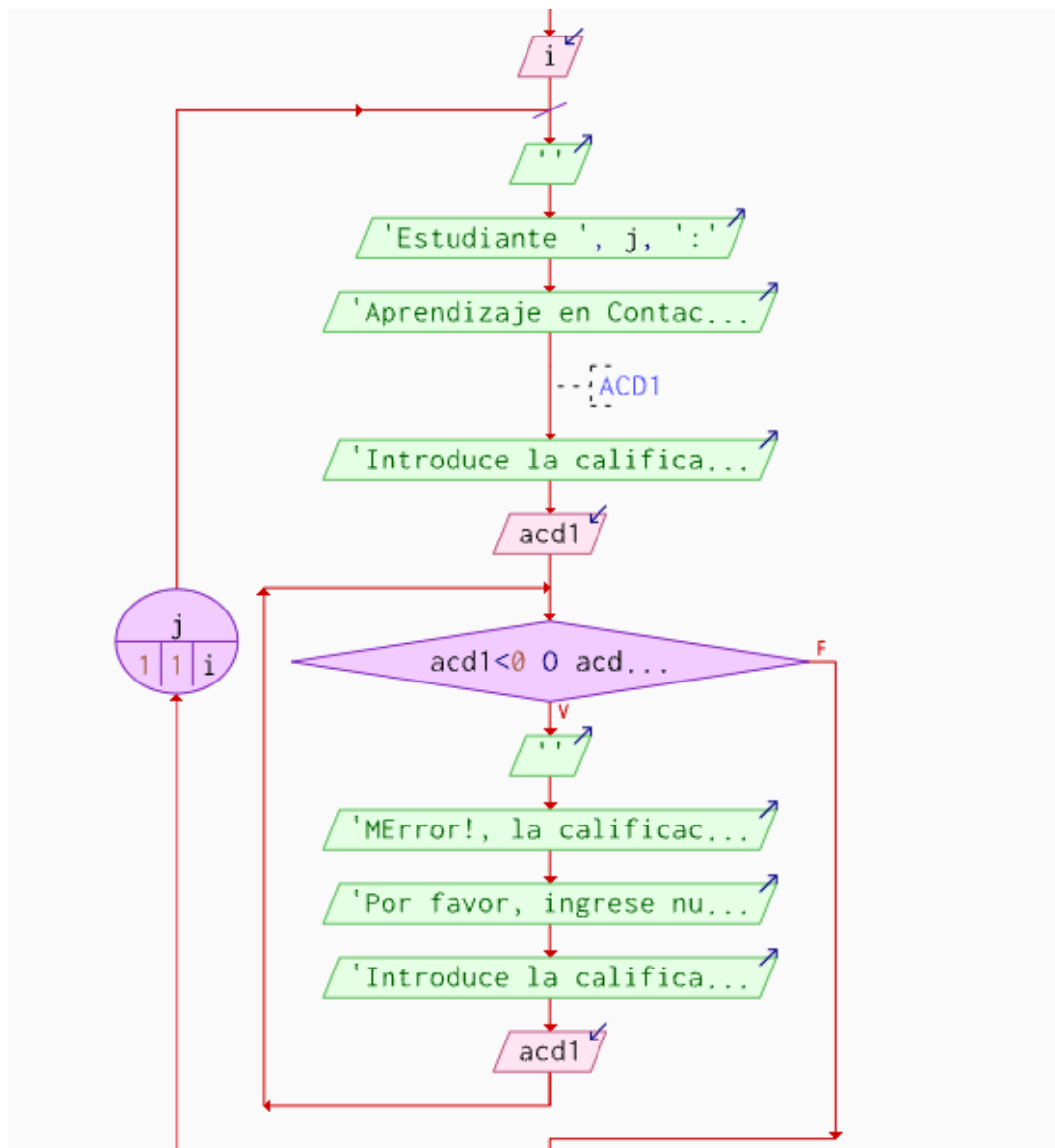
## 5. Resultados

### Contextualización del Problema:

- Contextualización del problema: Basado en el ejercicio del “Cálculo de la nota final de la Unidad 1 mediante estructuras secuenciales en C”, se desea automatizar el proceso de cálculo para varios estudiantes utilizando estructuras repetitivas:

- El programa debe permitir ingresar la cantidad total de estudiantes, y mediante un bucle, repetir el proceso de lectura de calificaciones y cálculo de la nota final.
- En cada repetición, el programa solicitará los valores de los componentes (ACD, APE, AA y ES), calculará la nota final y mostrará el resultado antes de pasar al siguiente estudiante.
- Además, el programa debe validar que las notas ingresadas estén dentro del rango permitido (0 a 10). Si el usuario ingresa una nota fuera de este rango, el programa mostrará un mensaje de error y volverá a solicitar el dato hasta que sea correcto.
- No se requiere guardar las notas; el programa únicamente procesará y mostrará el resultado individual en cada iteración.

### Diagrama de Flujo:





## Código Fuente en C:

```
#include <stdio.h>

int main(){

    //Definicion de Variables

    printf("Nota final de Teoria de la Programacion - Unidad 1\n\n");

    float acd1, acd2, aa1, aa2, ape1, ape2, es1, es2, notaAPE, notaACD, notaAA, notaES, notaFinal;
    char *calificacion, est;
    int i, j;

    //Datos de Entrada

    printf("Ingrese el numero de estudiantes que desea evaluar: ");
    scanf("%d", &i);
    getchar();

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

        printf("\nEstudiante %d:\n", j);

        printf("Aprendizaje en Contacto con el Docente (ACD)\n");

        printf("Introduce la calificacion de tu primer ACD (0-10): ");
        scanf("%f", &acd1);
        getchar();

        while (acd1 < 0 || acd1 > 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", &acd1);
            getchar();
        }

        printf("Introduce la calificacion de tu segundo ACD (0-10): ");
        scanf("%f", &acd2);
        getchar();
    }
}
```



```
while (acd2 < 0 || acd2 > 10) {
    printf("\n¡Error!, la calificación debe estar entre 0 y 10.\nPor favor, ingrese nuevamente.\n");
    printf("Introduce la calificación de tu segundo ACD (0-10): ");
    scanf("%f", &acd2);
    getchar();
}

printf("Aprendizaje Autonomo (AA)\n");

printf("Introduce la calificación de tu primer AA (0-10): ");
scanf("%f", &aa1);
getchar();

while (aa1 < 0 || aa1 > 10) {
    printf("\n¡Error!, la calificación debe estar entre 0 y 10.\nPor favor, ingrese nuevamente.\n");
    printf("Introduce la calificación de tu primer AA (0-10): ");
    scanf("%f", &aa1);
    getchar();
}

printf("Introduce la calificación de tu segundo AA (0-10): ");
scanf("%f", &aa2);
getchar();

while (aa2 < 0 || aa2 > 10) {
    printf("\n¡Error!, la calificación debe estar entre 0 y 10.\nPor favor, ingrese nuevamente.\n");
    printf("Introduce la calificación de tu segundo AA (0-10): ");
    scanf("%f", &aa2);
    getchar();
}

printf("Aprendizaje Practico Experimental (APE)\n");

printf("Introduce la calificación de tu primer APE (0-10): ");
scanf("%f", &ape1);
getchar();

while (ape1 < 0 || ape1 > 10) {
    printf("\n¡Error!, la calificación debe estar entre 0 y 10.\nPor favor, ingrese nuevamente.\n");
    printf("Introduce la calificación de tu primer APE (0-10): ");
    scanf("%f", &ape1);
}
```



```
        getchar();
    }

    printf("Introduce la calificacion de tu segundo APE (0-10): ");
    scanf("%f", &ape2);
    getchar();

    while (ape2 < 0 || ape2 > 10) {
        printf("\n¡Error!, la calificacion debe estar entre 0 y 10.\nPor favor, ingrese nuevamente.\n");
        printf("Introduce la calificacion de tu segundo APE (0-10): ");
        scanf("%f", &ape2);
        getchar();
    }

    printf("Evaluacion Sumativa (ES)\n");

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

    printf("Introduce la calificacion de tu segunda ES (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 segunda ES (0-10): ");
        scanf("%f", &es2);
        getchar();
    }

    //Proceso

    notaAPE = ((ape1 + ape2) / 2) * 0.25;
    notaACD = ((acd1 + acd2) / 2) * 0.2;
    notaAA = ((aa1 + aa2) / 2) * 0.2;
    notaES = ((es1 * 0.4) + (es2 * 0.6)) * 0.35;

    notaFinal = notaAPE + notaACD + notaAA + notaES;

    // Estructura del Condicional para calificacion cualitativa

    if (notaFinal >= 9) {
        calificacion = "Excelente";
    } else if (notaFinal < 9 && notaFinal >= 7) {
        calificacion = "Bueno";
    } else if (notaFinal < 7 && notaFinal >= 5) {
        calificacion = "Regular";
    } else {
        calificacion = "Deficiente";
    }

    //Salida

    printf("\nResultados del estudiante %d:\n", j);
    printf("Su nota en el APE es de %.2f", notaAPE);
    printf("\nSu nota en el ACD es de %.2f", notaAPE);
    printf("\nSu nota en el AA es de %.2f", notaAA);
    printf("\nSu nota en el ES es de %.2f\n", notaES);
    printf("La calificacion final del estudiante %d en la asignatura de Teoria de la Programacion es de: %.2f/10", j, notaFinal);

    printf("\nLa calificacion cualitativa del estudiante %d es: %s\n", j, calificacion);

}
return 0;
}
```



## Solución:

Compilación:

```
PS C:\Users\Usuario iTC\Documents\io\c2> gcc calculoUnidad1.c -o calculoUnidad1
PS C:\Users\Usuario iTC\Documents\io\c2> .\calculoUnidad1.exe
```

Estudiante 1: Javier Guarnizo

```
Nota final de Teoria de la Programacion - Unidad 1

Ingrese el numero de estudiantes que desea evaluar: 3

Estudiante 1:
Aprendizaje en Contacto con el Docente (ACD)
Introduce la calificacion de tu primer ACD (0-10): 10
Introduce la calificacion de tu segundo ACD (0-10): 10
Aprendizaje Autonomo (AA)
Introduce la calificacion de tu primer AA (0-10): 10
Introduce la calificacion de tu segundo AA (0-10): 10
Aprendizaje Practico Experimental (APE)
Introduce la calificacion de tu primer APE (0-10): 10
Introduce la calificacion de tu segundo APE (0-10): 10
Evaluacion Sumativa (ES)
Introduce la calificacion de tu primera ES (0-10): 10
Introduce la calificacion de tu segunda ES (0-10): 10

Resultados del estudiante 1:
Su nota en el APE es de 2.50
Su nota en el ACD es de 2.50
Su nota en el AA es de 2.00
Su nota en el ES es de 3.50
La calificacion final del estudiante 1 en la asignatura de Teoria de la Programacion es de: 10.00/10
La calificacion cualitativa del estudiante 1 es: Excelente
```

Estudiante 2: Darío Chillogallo

```
Estudiante 2:
Aprendizaje en Contacto con el Docente (ACD)
Introduce la calificacion de tu primer ACD (0-10): 10
Introduce la calificacion de tu segundo ACD (0-10): 10
Aprendizaje Autonomo (AA)
Introduce la calificacion de tu primer AA (0-10): 9.5
Introduce la calificacion de tu segundo AA (0-10): 9.25
Aprendizaje Practico Experimental (APE)
Introduce la calificacion de tu primer APE (0-10): 9.5
Introduce la calificacion de tu segundo APE (0-10): 9
Evaluacion Sumativa (ES)
Introduce la calificacion de tu primera ES (0-10): 8
Introduce la calificacion de tu segunda ES (0-10): 9

Resultados del estudiante 2:
Su nota en el APE es de 2.31
Su nota en el ACD es de 2.31
Su nota en el AA es de 1.88
Su nota en el ES es de 3.01
La calificacion final del estudiante 2 en la asignatura de Teoria de la Programacion es de: 9.20/10
La calificacion cualitativa del estudiante 2 es: Excelente
```

Estudiante 3: Valeria Agila



unl

Universidad  
Nacional  
de Loja

```
Estudiante 3:
Aprendizaje en Contacto con el Docente (ACD)
Introduce la calificacion de tu primer ACD (0-10): 10
Introduce la calificacion de tu segundo ACD (0-10): 10
Aprendizaje Autonomo (AA)
Introduce la calificacion de tu primer AA (0-10): 10
Introduce la calificacion de tu segundo AA (0-10): 10
Aprendizaje Practico Experimental (APE)
Introduce la calificacion de tu primer APE (0-10): 9.5
Introduce la calificacion de tu segundo APE (0-10): 8
Evaluacion Sumativa (ES)
Introduce la calificacion de tu primera ES (0-10): 9.75
Introduce la calificacion de tu segunda ES (0-10): 10

Resultados del estudiante 3:
Su nota en el APE es de 2.19
Su nota en el ACD es de 2.19
Su nota en el AA es de 2.00
Su nota en el ES es de 3.46
La calificacion final del estudiante 3 en la asignatura de Teoria de la Programacion es de: 9.65/10
La calificacion cualitativa del estudiante 3 es: Excelente
```

## 6. Preguntas de Control

- **¿En qué se diferencia una estructura repetitiva de una condicional?**

Las estructuras condicionales se caracterizan por evaluar solo una vez debido a la toma de decisiones de una condición, si la condición se cumple la evalúa y da un solo resultado; las estructuras repetitivas evalúan una condición, si es verdadera, se ejecutará y nuevamente evaluará si se sigue cumpliendo la misma condición, va a entrar en un bucle, hasta que la condición sea falsa.

Ejemplo Condicional:

If, else if, switch.

Ejemplo Repetitivas:

while, do while, for.

- **¿Qué diferencia existe entre las estructuras for, while y do...while en cuanto a su funcionamiento y uso?**

El for se usa cuando sabes cuántas veces se repetirá el ciclo, tiene todo en una misma línea: inicialización, condición y aumento.

Ejemplo: Sumar 3 veces un número.

El while se usa cuando no sabes cuántas veces se repetirá, se ejecuta solo si la condición es verdadera desde el inicio, primero evalúa la condición, luego ejecuta.

Ejemplo: Mostrar mensaje de error si el usuario se equivoca al ingresar un valor.

El do while se ejecuta al menos una vez, aunque la condición sea falsa, primero ejecuta el bloque, luego evalúa la condición.

Ejemplo: Hacer un bloque de una suma algebraica con un almacenador, mientras la respuesta es menor a 50.

- **¿Por qué es importante incluir validaciones dentro de un programa cuando se solicitan datos al usuario?**



unl

Universidad  
Nacional  
de Loja

Porque los usuarios pueden ingresar valores incorrectos, fuera de rango o inesperados que pueden causar errores o fallos en el programa. Las validaciones sirven para prevenir errores en tiempo de ejecución, garantizar que los cálculos sean correctos para evitar que el programa se dañe o se cierre y mejorar la experiencia del usuario

Ejemplo:

Si pides una nota (0-10) y el usuario pone 500 o -3, sin validación el programa calcula basura.

## **7. Conclusiones**

La elaboración del programa para evaluar las calificaciones de tres compañeros de clase permitió comprender el uso práctico de estructuras repetitivas dentro del desarrollo de software. Las estructuras repetitivas facilitaron procesar varios estudiantes sin duplicar código, mientras que las condicionales fueron esenciales para validar los datos ingresados y asegurar que todas las notas estuvieran dentro del rango permitido. Esto garantizó que los resultados obtenidos fueran correctos y confiables. Además, incluir un diagrama de flujo permitió visualizar de forma clara la lógica del programa, identificar el orden de los procesos y comprender cómo interactúan las decisiones y los ciclos dentro de la solución. En conjunto, este ejercicio reforzó la importancia de una buena estructura lógica, el control de errores y la automatización de tareas repetitivas en la programación.

## **8. Recomendaciones**

Antes de usar cualquier valor ingresado por el usuario, es recomendable validar rangos y tipos para evitar errores y mantener la integridad de los resultados.

Añadir comentarios claros ayuda a comprender la lógica del programa y facilita futuras modificaciones o revisiones.

Probar el programa con valores extremos, incorrectos y casos típicos asegura que la lógica sea robusta y funcione en todo momento.