



Reporte Técnico de Actividades Práctico-Experimentales Nro. 00X

1. Datos de Identificación del Estudiante y la Práctica

Nombre del estudiante(s)	Joaquin Emilio Moscol Castillo
Asignatura	Teoría de la programación
Ciclo	1 A
Unidad	2
Resultado de aprendizaje de la unidad	Aplica las estructuras de programación en la resolución de problemas básicos, bajo los principios de solidaridad, transparencia, responsabilidad y honestidad
Práctica Nro.	002
Tipo	Individual o Grupal
Título de la Práctica	Aplicación de estructuras repetitivas en la resolución de problemas.
Nombre del Docente	Lisette Geoconda López Faicán
Fecha	Jueves 27 de noviembre del 2025 Jueves 04 de diciembre del 2025
Horario	10h30 – 13h30
Lugar	Aula física asignada al paralelo.
Tiempo planificado en el Sílabo	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 diagram 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).

4. Procedimiento / Metodología Ejecutada

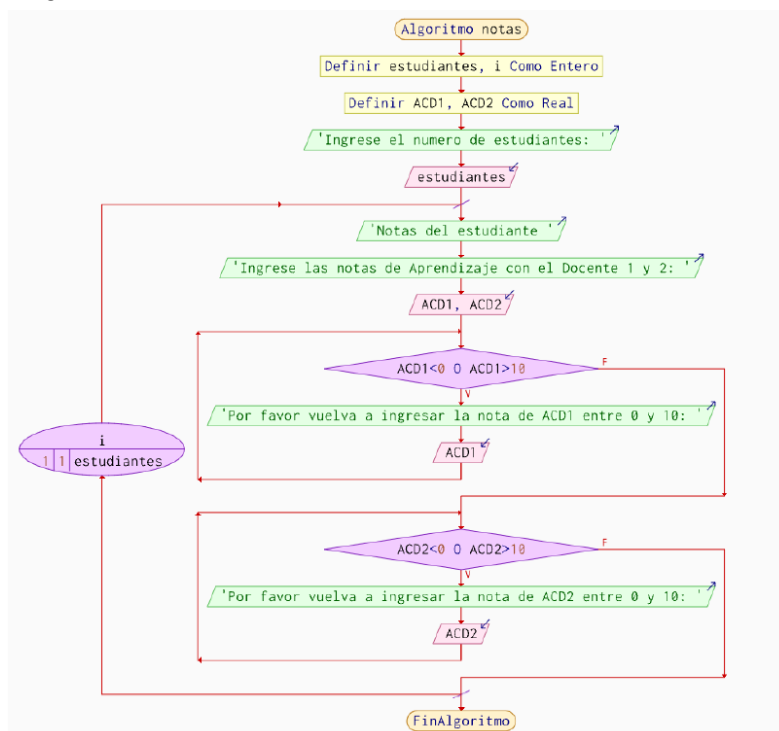
Lo primero que se hizo fue declarar las variables, luego coloque "Printf" que pida al usuario que ingrese el número de estudiantes requiere el programa repetir el bucle luego coloque un "For" antes de todo el texto, que tenga el contador que aumente hasta alcanzar el numero de estudiantes ingresados, una vez realizo esto se pidieron al usuario que ingrese las notas de cada parámetro, luego ingrese un "While" para que vea una condición que es $(ACD1 < 0 \parallel ACD1 > 10)$

dentro de esta condición le coloque un "Printf" que dice que el número ingresado no está dentro del rango 1 al 10 y luego ingrese un "While" para que vea otra condición que es $(ACD2 < 0 \parallel ACD2 > 10)$ y dentro de esta condición le coloque un "Printf" que dice que el numero ingresado no esta dentro del rango 1 al 10, se repite estos mismo bucles en el resto de notas, una vez al finalizar se saca el promedio y se dicta si es una nota cualitativa "Excelente", "Bueno", "Regular", "Deficiente", al final se imprimen las notas finales.

5. Resultados

- 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.

Diagram de flujo



Codigo en C

```
1 #include <stdio.h>
2
3 int main() {
4
5     // Definición de variables
6     double notaACD1, notaACD2, notaAPE1, notaAPE2, notaAA1, notaAA2;
7     int numEstudiante;
8     double sumativaES1, sumativaES2, notaFinal;
9     double notaExce, notaBueno, notaRegular, notaDeficiente;
10    double promedioACD, promedioAPE, promedioAA, promedioSumativa, promesumativaES1, promesumativaES2;
11    double pondeACD = 0.02, pondeAPE = 0.025, pondeAA = 0.02, pondeES = 0.35;
12
13    // Datos de entrada
14
15    printf("----- Ingrese Cuantos Estudiantes Desea Calcular la Nota Final de la UNIDAD 1: -----\\n");
16    scanf("%i", &numEstudiante);
17
18    for(int i = 1; i <= numEstudiante; i++) {
19
20        notaExce = 0;
21        notaBueno = 0;
22        notaRegular = 0;
23        notaDeficiente = 0;
24
25        printf("-----\\n");
26        printf("CALCULANDO DATOS DEL ESTUDIANTE Nro: %d\\n", i);
27        printf("-----\\n");
28
29        do {
30            printf("-----Ingrese la nota del ACD Nro1:-----\\n");
31            scanf("%lf", &notaACD1);
32            getchar();
33            if (notaACD1 < 0 || notaACD1 > 10) {
34                printf("-----ERROR: La nota debe estar entre 0 y 10. Ingrese el dato nuevamente.-----\\n");
35            }
36        } while (notaACD1 < 0 || notaACD1 > 10);
37
38        do {
39            printf("-----Ingrese la nota del ACD Nro2:-----\\n");
40            scanf("%lf", &notaACD2);
41            getchar();
42            if (notaACD2 < 0 || notaACD2 > 10) {
43                printf("-----ERROR: La nota debe estar entre 0 y 10. Ingrese el dato nuevamente.-----\\n");
44            }
45        } while (notaACD2 < 0 || notaACD2 > 10);
46
47        do {
48            printf("-----Ingrese la nota del APE Nro1:-----\\n");
49            scanf("%lf", &notaAPE1);
50            getchar();
51            if (notaAPE1 < 0 || notaAPE1 > 10) {
52                printf("-----ERROR: La nota debe estar entre 0 y 10. Ingrese el dato nuevamente.-----\\n");
53            }
54        } while (notaAPE1 < 0 || notaAPE1 > 10);
55
56        do {
57            printf("-----Ingrese la nota del APE Nro2:-----\\n");
58            scanf("%lf", &notaAPE2);
59            getchar();
60            if (notaAPE2 < 0 || notaAPE2 > 10) {
61                printf("-----ERROR: La nota debe estar entre 0 y 10. Ingrese el dato nuevamente.-----\\n");
62            }
63        } while (notaAPE2 < 0 || notaAPE2 > 10);
64    }
```

```

64
65     do {
66         printf("-----Ingrese la nota del AA Nro1:-----\n");
67         scanf("%lf", &notaAA1);
68         getchar();
69         if (notaAA1 < 0 || notaAA1 > 10) {
70             printf("-----ERROR: La nota debe estar entre 0 y 10. Ingrese el dato nuevamente.-----\n");
71         }
72     } while (notaAA1 < 0 || notaAA1 > 10);
73
74     do {
75         printf("-----Ingrese la nota del AA Nro2:-----\n");
76         scanf("%lf", &notaAA2);
77         getchar();
78         if (notaAA2 < 0 || notaAA2 > 10) {
79             printf("-----ERROR: La nota debe estar entre 0 y 10. Ingrese el dato nuevamente.-----\n");
80         }
81     } while (notaAA2 < 0 || notaAA2 > 10);
82
83     do {
84         printf("-----Ingrese la nota de la Evaluacion Sumativa Nro1:-----\n");
85         scanf("%lf", &sumativaES1);
86         getchar();
87         if (sumativaES1 < 0 || sumativaES1 > 10) {
88             printf("-----ERROR: La nota debe estar entre 0 y 10. Ingrese el dato nuevamente.-----\n");
89         }
90     } while (sumativaES1 < 0 || sumativaES1 > 10);
91
92     do {
93         printf("-----Ingrese la nota de la Evaluacion Sumativa Nro2:-----\n");
94         scanf("%lf", &sumativaES2);
95         getchar();
96         if (sumativaES2 < 0 || sumativaES2 > 10) {
97             printf("-----ERROR: La nota debe estar entre 0 y 10. Ingrese el dato nuevamente.-----\n");
98         }
99     } while (sumativaES2 < 0 || sumativaES2 > 10);
100
101     // Proceso
102     promedioACD = (((notaACD1 + notaACD2) / 2) * 100) / 10 * pondeACD;
103     promedioAPE = (((notaAPE1 + notaAPE2) / 2) * 100) / 10 * pondeAPE;
104     promedioAA = (((notaAA1 + notaAA2) / 2) * 100) / 10 * pondeAA;
105     promesumativaES1 = (sumativaES1 * 0.4);
106     promesumativaES2 = (sumativaES2 * 0.6);
107     promedioSumativa = (promesumativaES1 + promesumativaES2) * pondeES;
108     notaFinal = promedioACD + promedioAPE + promedioAA + promedioSumativa;
109
110     if(notaFinal >= 9){
111         notaExce = (notaFinal >= 9);
112     } else if (notaFinal >= 7 && notaFinal < 9){
113         notaBueno = (notaFinal >= 7 && notaFinal < 9);
114     } else if(notaFinal >= 5 && notaFinal < 7){
115         notaRegular = (notaFinal >= 5 && notaFinal < 7);
116     } else {
117         notaDeficiente = (notaFinal < 5);
118     }
119
120     // Datos de salida
121     printf("\n--> RESULTADOS DEL ESTUDIANTE NRO: %d <-- \n", i);
122
123     printf("-----La nota en el Aprendizaje en Contacto con el Docente (ACD) es:----- %f\n", promedioACD);
124     printf("-----La nota en el Aprendizaje Práctico Experimental (APE) es:----- %f\n", promedioAPE);
125     printf("-----La nota en el Autoaprendizaje (AA) es:----- %f\n", promedioAA);
126     printf("-----La nota en la Evaluacion Sumativa (ES) es:----- %f\n", promedioSumativa);
127     printf("-----La nota Final de la UNIDAD 1 es:----- %f\n", notaFinal);
128     printf("%s\n", notaExce ? "-----EXCELENTE-----" : (notaBueno ? "-----BUENO-----" : (notaRegular ? "-----REGULAR-----" : (notaDeficiente ? "-----DEFICIENTE-----" : "")));
129
130     printf("-----\n");
131
132 }
133
134 getchar();
135 getchar();
136
137 return 0;
138
139 }

```



Caso de prueba

```
----- Ingrese Cuantos Estudiantes Desea Calcular la Nota Final de la UNIDAD 1: -----
3
-----
CALULANDO DATOS DEL ESTUDIANTE NRO: 1
-----
-----Ingrese la nota del ACD Nro1:-----
10
-----Ingrese la nota del ACD Nro2:-----
10
-----Ingrese la nota del APE Nro1:-----
10
-----Ingrese la nota del APE Nro2:-----
10
-----Ingrese la nota del AA Nro1:-----
10
-----Ingrese la nota del AA Nro2:-----
10
-----Ingrese la nota de la Evaluacion Sumativa Nro1:-----
10
-----Ingrese la nota de la Evaluacion Sumativa Nro2:-----
10

---> RESULTADOS DEL ESTUDIANTE NRO: 1 <---
-----La nota en el Aprendizaje en Contacto con el Docente (ACD) es:----- 2.000000
-----La nota en el Aprendizaje Práctico Experimental (APE) es:----- 2.500000
-----La nota en el Autoaprendizaje (AA) es:----- 2.000000
-----La nota en la Evaluacion Sumativa (ES) es:----- 3.500000
-----La nota Final de la UNIDAD 1 es:----- 10.000000
-----EXCELENTE-----
-----
```

```
-----
CALULANDO DATOS DEL ESTUDIANTE NRO: 2
-----
-----Ingrese la nota del ACD Nro1:-----
1
-----Ingrese la nota del ACD Nro2:-----
1
-----Ingrese la nota del APE Nro1:-----
1
-----Ingrese la nota del APE Nro2:-----
1
-----Ingrese la nota del AA Nro1:-----
1
-----Ingrese la nota del AA Nro2:-----
1
-----Ingrese la nota de la Evaluacion Sumativa Nro1:-----
1
-----Ingrese la nota de la Evaluacion Sumativa Nro2:-----
1

---> RESULTADOS DEL ESTUDIANTE NRO: 2 <---
-----La nota en el Aprendizaje en Contacto con el Docente (ACD) es:----- 0.200000
-----La nota en el Aprendizaje Práctico Experimental (APE) es:----- 0.250000
-----La nota en el Autoaprendizaje (AA) es:----- 0.200000
-----La nota en la Evaluacion Sumativa (ES) es:----- 0.350000
-----La nota Final de la UNIDAD 1 es:----- 1.000000
-----DEFICIENTE-----
```

```
-----
CALULANDO DATOS DEL ESTUDIANTE NRO: 3
-----
-----Ingrese la nota del ACD Nro1:-----
5
-----Ingrese la nota del ACD Nro2:-----
5
-----Ingrese la nota del APE Nro1:-----
5
-----Ingrese la nota del APE Nro2:-----
5
-----Ingrese la nota del AA Nro1:-----
5
-----Ingrese la nota del AA Nro2:-----
5
-----Ingrese la nota de la Evaluacion Sumativa Nro1:-----
5
-----Ingrese la nota de la Evaluacion Sumativa Nro2:-----
5

----> RESULTADOS DEL ESTUDIANTE NRO: 3 <---
-----La nota en el Aprendizaje en Contacto con el Docente (ACD) es:----- 1.000000
-----La nota en el Aprendizaje Práctico Experimental (APE) es:----- 1.250000
-----La nota en el Autoaprendizaje (AA) es:----- 1.000000
-----La nota en la Evaluacion Sumativa (ES) es:----- 1.750000
-----La nota Final de la UNIDAD 1 es:----- 5.000000
-----REGULAR-----
-----
```

6. Preguntas de Control

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

Una estructura condicional (como if-else o switch) permite ejecutar un bloque de código **una sola vez** si se cumple una condición específica. En cambio, una estructura repetitiva (como for, while) permite ejecutar el mismo bloque de código **múltiples veces** de forma automática mientras la condición se mantenga verdadera, siendo ideal para procesar conjuntos de datos o iteraciones.

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

For: Se utiliza cuando se conoce de antemano el número exacto de iteraciones (ej. "ingresar 5 estudiantes"). Agrupa la inicialización, condición y actualización en una sola línea.

While: Evalúa la condición antes de ejecutar el código. Si la condición es falsa al inicio, el ciclo nunca se ejecuta. Se usa cuando no sabemos cuántas veces se repetirá el proceso.

Do...while: Ejecuta el bloque de código al menos una vez y luego evalúa la condición. Es la estructura ideal para validaciones de entrada de datos, garantizando que se pida el dato al usuario antes de verificar si es correcto.

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

Las validaciones son cruciales para mantener la integridad de los datos y la estabilidad del programa. Sin ellas, un usuario podría ingresar valores ilógicos (como una nota de -5 o 20) que romperían la lógica matemática del cálculo de



promedios, o incluso ingresar tipos de datos erróneos que podrían causar que el programa falle (crash). Garantizan que el programa solo procese información útil y correcta.

7. Conclusiones

Se logró diseñar y codificar exitosamente un algoritmo en lenguaje C que utiliza un bucle for para automatizar el cálculo de notas de múltiples estudiantes, optimizando el tiempo de procesamiento en comparación con una ejecución manual secuencial.

La implementación de estructuras repetitivas do-while demostró ser la estrategia más efectiva para la validación de datos, asegurando que el programa sea robusto y no acepte calificaciones fuera del rango establecido de 0 a 10.

Se comprendió la importancia crítica de la inicialización y reinicio de variables dentro de los bucles (ámbito de variables), ya que no hacerlo provoca errores lógicos graves donde los datos de un estudiante anterior afectan los resultados del siguiente (problema de "código basura").

8. Recomendaciones

Se recomienda siempre inicializar o reiniciar las variables acumuladoras y banderas al comienzo de cada iteración de un bucle principal para evitar errores de persistencia de datos, tal como se corrigió en esta práctica.

Para mejorar la legibilidad del código y el diseño del algoritmo, se sugiere en futuras prácticas modularizar el programa utilizando funciones separadas para la validación de notas y los cálculos matemáticos, en lugar de tener todo el código dentro de la función main.

Al realizar pruebas de escritorio o ejecuciones de prueba, es fundamental probar no solo los casos de éxito, sino también los "casos de borde" (notas 0, notas 10, y notas inválidas) para asegurar que las estructuras repetitivas de validación no permitan fugas de datos incorrectos.

9. Anexos

Recursos de apoyo para cada una de las secciones anteriores.