Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los Recursos Naturales no Renovables

Carrera de Computación

Asignatura: Teoría de la Programación

Ciclo 1 - Unidad: 2

AA - Actividad N.º 2: Cuadro comparativo entre las estructuras repetitivas.

Docente: Ing. Lissette Geoconda López Faicán

Estudiante: Jeancarlos Fernando Aguirre Romero

Loja – Ecuador

2025

I. INTRODUCCIÓN

En este Documento se presentarán a través de un cuadro comparativo las diferencias entre las diferentes estructuras cíclicas que existen en el lenguaje C, además de un ejercicio realizado en la plataforma omegaUp todo esto como parte de una actividad AA,

II. TABLA COMPARATIVA REFERENTE A LAS ESTRUCTURAS BUBLES EN C

Nombre de la estructura	Tipo de Estructura	Sintaxis / Estructura	Uso
For (Para)	Ciclo determinado (ciclo con contador)	int VAR; **for (VAR = Vi; VAR<Vf; VAR++) {** Instrucción 1; Instrucción 2; . . . Instrucción n; **}**	La instrucción for explota la capacidad del ordenador para repetir procesos y para contar. Muchas veces esta instrucción se puede reemplazar por otras instrucciones o estructuras de tipo repetición. Se podrá optar por la que se estime más oportuna, que muchas veces será la instrucción for pues reúne unas cualidades interesantes. [1] Se usa cuando el número de iteraciones es conocido o puede determinarse claramente.
While (Mientras)	Ciclo indeterminado (Ciclo controlado por condición)	**while (condición) {** Instrucción 1; Instrucción 2; . . . Instrucción n; **}**	La instrucción while es una de las alternativas para la repetición de procesos en programación. Como veremos, guarda cierta similitud con la instrucción for, hasta el punto de que en algunas ocasiones podrá optarse por el uso de una u otra de forma indistinta. While es una palabra clave en C que admite varios usos.[2] Se usa cuando se debe repetir un bloque mientras una condición sea verdadera.

Do while (Hacer mientras)	Ciclo post-condicionado. (Ciclo indeterminado con condición al final)	**do {** Instrucción 1; Instrucción 2; . . . Instrucción n; **} while (condición);**	Un bucle do … while es anidable dentro de sí mismo o dentro de otras estructuras. Es importante verificar que los bucles diseñados con esta instrucción dispongan de una condición de salida válida. .[2] Se usa cuando es necesario que el bloque se ejecute al menos una vez antes de evaluar la condición

A. Reflexión respective a la comparación de los datos

El cuadro comparativo permite visualizar con claridad las diferencias y usos más adecuados de cada estructura repetitiva. Gracias a esta comparación, es posible comprender que cada ciclo cumple una función específica dentro de un programa: el for resulta ideal cuando se conoce la cantidad exacta de repeticiones, el while es útil cuando la ejecución depende completamente de una condición, y el do…while asegura que el bloque se ejecute al menos una vez. Analizar estas diferencias facilita seleccionar conscientemente la estructura correcta según la necesidad del problema, lo que mejora la eficiencia, organización y claridad del código.

III. EJERCICIO BASADO EN CICLOS – OMEGAUP

Como parte de esta actividad autónoma se propuso el realizar un ejercicio dentro de la plataforma de omegaUp, Por lo que procedió a escoger un ejercicio o problema de uno de los múltiples cursos que ofrece omegaUp, por lo que el proceso para la realización de el mismo se explicara a continuación:

*(**NOTA**: Debido a que omegaUp no ofrece algún tipo de curso basado en C se procedió a tomar un problema del "Curso de introducción a java", esto cabe aclarar en caso de haber confusiones al momento de justificar el ejercicio presentara y de donde se lo tomo, Como nota adicional este problema se realizara basado en la rubrica de trabajo enviada para este trabajo, con la única excepción que se añadirá adicionalmente la resolución del problema en el lenguaje JAVA, por lo que se presentaran 3 formas de resolución: Diagrama de flujo, código en C y código en JAVA)*

A. Planteamiento del problema

G. Calculando el logaritmo base 2

Descripción

El logaritmo base 2 de un número es el número de veces que se tiene que multiplicar el 2 por sí mismo para alcanzar dicho número. Por ejemplo, $\log_2(2) = 1, \log_2(4) = 2, \log_2(8) = 3$ y así sucesivamente. Escribe un programa que calcule el logartimo base 2 de un entero N.

Entrada

El valor del entero N. Puedes suponer que $2 \leq N \leq 2^{30}$ y que N es de la forma 2^i donde i es un entero.

Salida

El logaritmo base 2 de N.

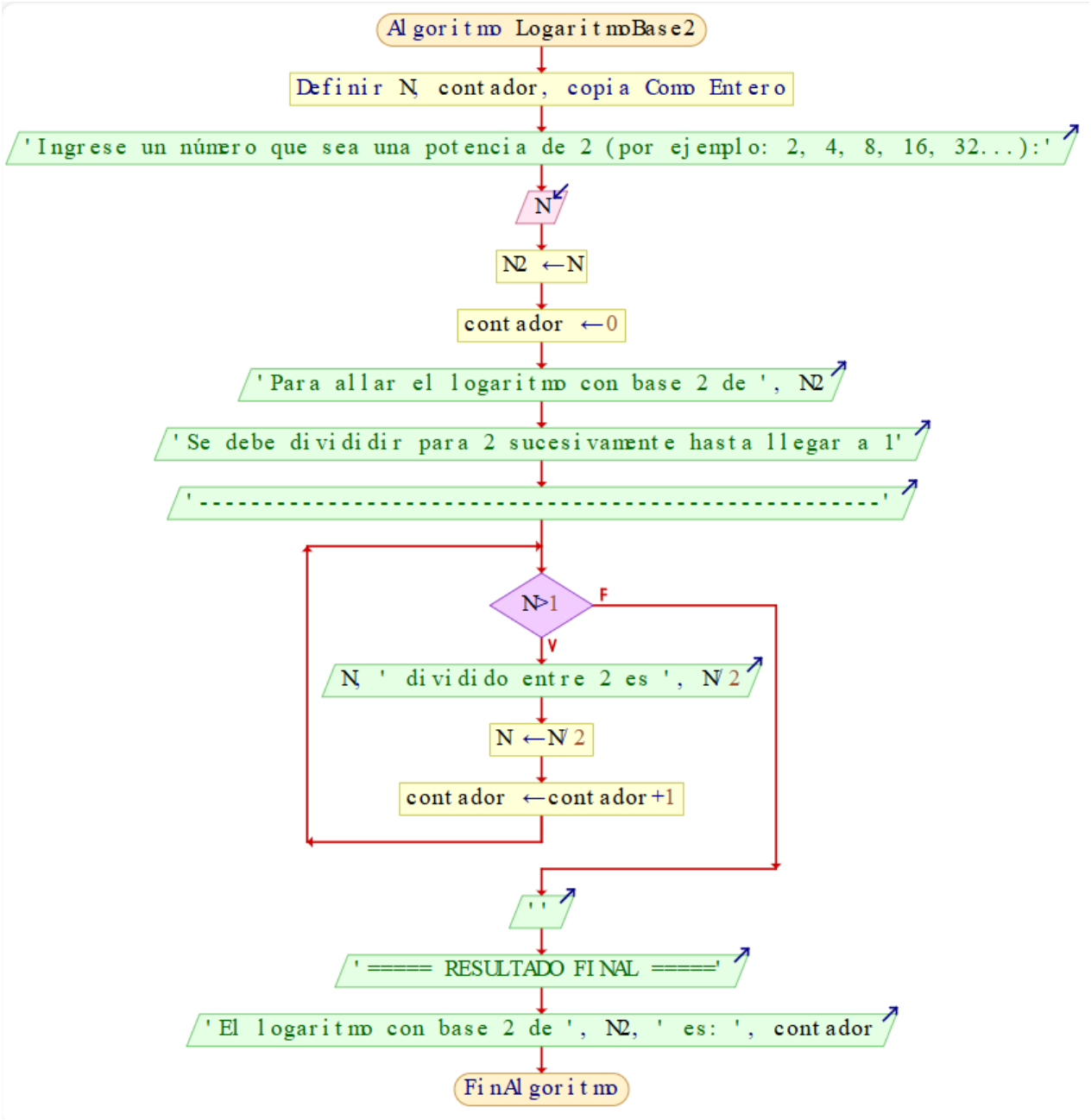
Ejemplo

Entrada	Salida
2	1
4	2
8	3
16	4
32	5

[3]

Y con este planteamiento inicial de este problema podemos proceder con su respectiva resolución del ejercicio en seudocódigo (PSeInt) para obtener nuestro diagrama de flujo:

B. *Resolución del problema en Seudocódigo*



Y con el Diagrama de flujo comprobado y Verificado podemos Proceder con el traspaso de este código a C y Java como resolución Final y completa de ejercicio propuesto por la omgaUp:

C. *Resolución del problema en el Lenguaje C*

```c
#include <stdio.h>

int main() {

    int N, contador, N2;

    printf("Ingrese un numero que sea una potencia de 2 (por ejemplo: 2, 4, 8, 16, 32...):\n");
    scanf("%d", &N);

    N2 = N;
    contador = 0;

    printf("Para hallar el logaritmo con base 2 de %d\n", N2);
    printf("Se debe dividir para 2 sucesivamente hasta llegar a 1\n");
    printf("-----------------------------------------------------\n");

    while (N > 1) {
        printf("%d dividido entre 2 es %d\n", N, N / 2);
        N = N / 2;
        contador = contador + 1;
    }

    printf("\n");
    printf("===== RESULTADO FINAL =====\n");
    printf("El logaritmo con base 2 de %d es: %d\n", N2, contador);

    return 0;
}

```

Resultado en Consola:

```
PS C:\Users\USUARIO\Documents\Programacion\Lenguaje C\Unidad 2 TDP\C> gcc omegaup.c -o omegaup
PS C:\Users\USUARIO\Documents\Programacion\Lenguaje C\Unidad 2 TDP\C> ./omegaup.exe
Ingrese un numero que sea una potencia de 2 (por ejemplo: 2, 4, 8, 16, 32...):
32
Para hallar el logaritmo con base 2 de 32
Se debe dividir para 2 sucesivamente hasta llegar a 1
-----------------------------------------------------
32 dividido entre 2 es 16
16 dividido entre 2 es 8
8 dividido entre 2 es 4
4 dividido entre 2 es 2
2 dividido entre 2 es 1

===== RESULTADO FINAL =====
El logaritmo con base 2 de 32 es: 5
PS C:\Users\USUARIO\Documents\Programacion\Lenguaje C\Unidad 2 TDP\C>
```

D. *Resolución del problema en el Lenguaje en Java*

```java
import java.util.Scanner;

public class omegaup {

    Run | Debug
    public static void main(String[] args) {

        int N, contador, N2;

        Scanner sc = new Scanner(System.in);

        System.out.println(x: "Ingrese un número que sea una potencia de 2 (por ejemplo: 2, 4, 8, 16, 32...):");
        N = sc.nextInt();

        N2 = N;
        contador = 0;

        System.out.println("Para allar el logaritmo con base 2 de " + N2);
        System.out.println(x: "Se debe divididir para 2 sucesivamente hasta llegar a 1");
        System.out.println(x: "-----------------------------------------------------");

        while (N > 1) {
            System.out.println(N + " dividido entre 2 es " + (N / 2));
            N = N / 2;
            contador = contador + 1;
        }

        System.out.println(x: "");
        System.out.println(x: "===== RESULTADO FINAL =====");
        System.out.println("El logaritmo con base 2 de " + N2 + " es: " + contador);
    }
}
```

Resultado en Consola:

```
PS C:\Users\USUARIO\Documents\Programacion\Lenguaje C\Unidad 2 TDP\JAVA> javac omegaup.java
PS C:\Users\USUARIO\Documents\Programacion\Lenguaje C\Unidad 2 TDP\JAVA> java omegaup
Ingrese un número que sea una potencia de 2 (por ejemplo: 2, 4, 8, 16, 32...):
32
Para allar el logaritmo con base 2 de 32
Se debe divididir para 2 sucesivamente hasta llegar a 1
-----------------------------------------------------
32 dividido entre 2 es 16
16 dividido entre 2 es 8
8 dividido entre 2 es 4
4 dividido entre 2 es 2
2 dividido entre 2 es 1

===== RESULTADO FINAL =====
El logaritmo con base 2 de 32 es: 5
PS C:\Users\USUARIO\Documents\Programacion\Lenguaje C\Unidad 2 TDP\JAVA>
```

IV. CONCLUSIONES

Las estructuras repetitivas permiten automatizar procesos y reducir la cantidad de código necesario para resolver un problema. Su correcto uso facilita la eficiencia, claridad y organización de los algoritmos, además de mejorar la escalabilidad de los programas. Comprender las diferencias entre for, while y do...while es esencial para seleccionar la estructura adecuada según las necesidades del problema.

V. BIBLIOGRAFIAS

[1] «Bucles en lenguaje C (estructura de repetición). Condición, contador. Ejemplos. Tabla de multiplicar (CU00533F)». Accedido: 7 de diciembre de 2025. [En línea]. Disponible en: https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=931:bucles-en-lenguaje-c-estructura-de-repeticion-condicion-contador-ejemplos-tabla-de-multiplicar-cu00533f&catid=82&Itemid=210

[2] «while y do while break en lenguaje C. Bucles mientras hacer. Forzar salida o terminación. Ejemplo (CU00534F)». Accedido: 7 de diciembre de 2025. [En línea]. Disponible en: https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=933:while-y-do-while-break-en-lenguaje-c-bucles-mientras-hacer-forzar-salida-o-terminacion-ejemplo-cu00534f&catid=82&Itemid=210

[3] «Introducción a Java - 6-P. Práctica de Ciclos y Cadenas – omegaUp». Accedido: 7 de diciembre de 2025. [En línea]. Disponible en: https://omegaup.com/course/curso_de_introduccion_a_java/assignment/java_6p#problems/Calculando-el-logaritmo-base-2