Universidad Nacionalde Loja

Facultad De La Energía, Las Industrias Y Los Recursos Naturales No Renovables

Carrera de Computación

Docente:
Ing. Lissette Geoconda López Faicán

Asignatura:
Teoría de la Programación - Unidad 1

Estudiante:
Pilar Valentina Naranjo Quizhpe

Curso/Paralelo:
Primer Ciclo “A”

Período Académico:
Septiembre 2025 - Febrero 2026

1. Estructuras repetitivas (Tabla comparativa)

Tipo	Estructura	Uso principal
for	for (inicialización; condición; actualización) { bloque de código }	El bucle **for** combina en una sola línea la inicialización, la condición y la actualización de la variable de control, facilitando un ciclo organizado y sencillo de manejar. Es útil cuando se tiene claro o establecido de antemano cuántas veces se debe repetir. Se emplea especialmente cuando se necesita un número específico de iteraciones, siendo perfecto para recorrer listas, contar ciclos y realizar procesos de manera controlada.[2]
while	while (condición) {instruccioes}	El bucle **while** se emplea cuando no se sabe cuántas veces se repetirá un proceso y la repetición depende de una condición lógica que debe ser revisada antes de comenzar el ciclo, permitiendo que el proceso prosiga hasta que esa condición ya no se cumpla. Es necesario declarar e iniciar la variable de control antes del bucle, y actualizarse dentro de él para evitar iteraciones infinitas. Se usa principalmente en validaciones, lectura de datos o procesos que dependen de una condición lógica evaluada antes de cada iteración. **[1]**
do…while	do { instrucciones; } while(condición);	El bucle do…while se emplea cuando es necesario ejecutar el conjunto de instrucciones un mínimo de una vez, ya que la condición se evalúa al final del ciclo. . Es adecuado en situaciones donde el código debe ejecutarse de una manera inicial sin tener en cuenta si la condición es cierta o no. Este tipo de estructura es útil en menús interactivos, solicitudes repetidas de datos o en procesos donde se requiere realizar una acción mínima antes de validar la condición.[3]

2. Ejercicio (OmegaUp)

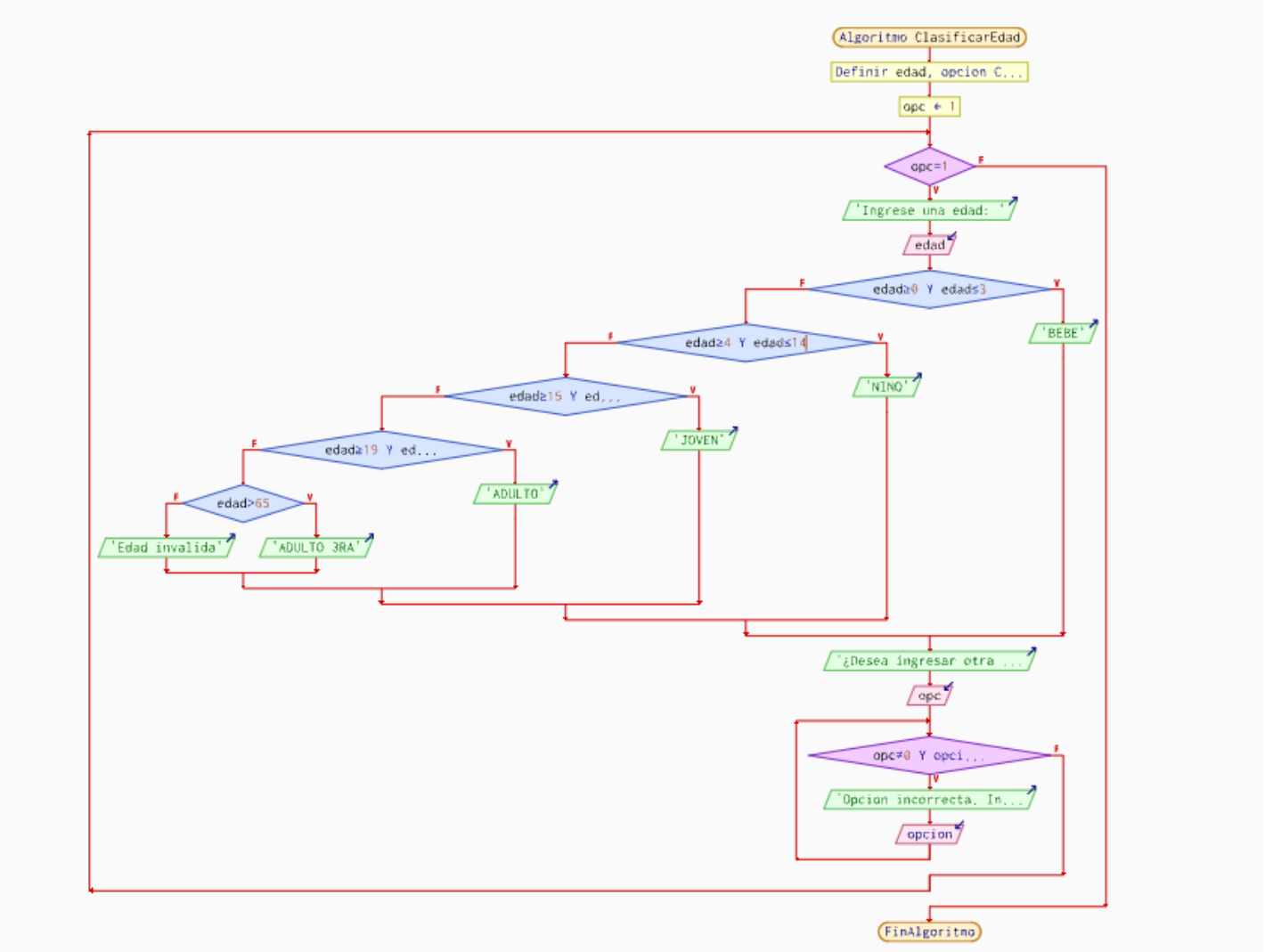
Planteamiento del problema:

Dieguito vio en su clase de biología las etapas de la vida, pero aún no las distingue muy bien. Tu tarea es ayudar a Dieguito haciendo un programa el cual, dado un número de entrada , el cual representa una edad, te diga si dicha edad corresponde a un **bebé**, **niño**, **joven**, **adulto** o **adulto de la tercera edad**. Para lograr lo anterior, considera la siguiente clasificación:

- Bebé: De 0 a 3 años, inclusive.
- Niño: De 4 a 14 años, inclusive.
- Joven: De 15 a 18 años, inclusive.

- Adulto: de 19 a 65 años, inclusive.
- Adulto de la tercera edad: de 65 años en adelante.

3. Diagrama de flujo



4. Código en C

```c
#include <stdio.h>
int main() {

    int edad;
    int repetir = 1;

    do {
        printf("Ingrese una edad: ");
        scanf("%d", &edad);

        if (edad >= 0 && edad <= 3) {
            printf("Bebe\n");
        }
        else if (edad >= 4 && edad <= 14) {
            printf("Niño\n");
        }
        else if (edad >= 15 && edad <= 18) {
            printf("Joven\n");
        }
        else if (edad >= 19 && edad <= 65) {
            printf("Adulto\n");
        }
        else if (edad > 65) {
            printf("Adulto 3RA\n");
        }
        else {
            printf("Edad inválida\n");
        }

        // Preguntar si desea continuar
        printf("¿Desea ingresar otra edad? (1 = SI / 0 = NO): ");
        scanf("%d", &repetir);

        // Validación simple
        while (repetir != 0 && repetir != 1) {
            printf("Opción incorrecta. Ingrese 1 (SI) o 0 (NO): ");
            scanf("%d", &repetir);
        }

    } while (repetir == 1);

    return 0;
}
```

5. Conclusión

Las estructuras repetitivas son fundamentales en la programación, ya que permiten automatizar tareas que requieren varias ejecuciones del mismo bloque de código. Gracias a ciclos como **for, while y do…while**, los programas pueden manejar grandes cantidades de datos, ejecutar procesos controlados, validar entradas y resolver problemas de forma eficiente. Comprender cuándo usar cada estructura facilita la creación de soluciones más óptimas, limpias y fáciles de mantener.

5. Bibliografia

[1] Microsoft, “Sentencias de selección y repetición en C#,” *Microsoft Learn*, 2020. Disponible en: https://learn.microsoft.com/es-es/dotnet/csharp/language-reference/statements/selection-statements
Accedido el: 06-dic-2025.

[2] Studocu, “Estructuras condicionales y repetitivas – Apuntes de programación,” *Studocu*, 2021. Disponible en: https://www.studocu.com
Accedido el: 06-dic-2025.

[3] M. Ben-Ari, *Understanding Programming Languages*. John Wiley & Sons, 1996. Disponible en: https://www.freecomputerbooks.com/Understanding-Programming-Languages.html
Accedido el: 06-dic-2025.