

Tipos de condicionales

1. If simple

Ejecuta un bloque de código solo si la condición es verdadera. Es útil cuando se necesita validar un único caso [1].

2. If-else

Permite elegir entre dos caminos:

- Uno cuando la condición es verdadera y otro cuando es falsa [2].

3. If-else if

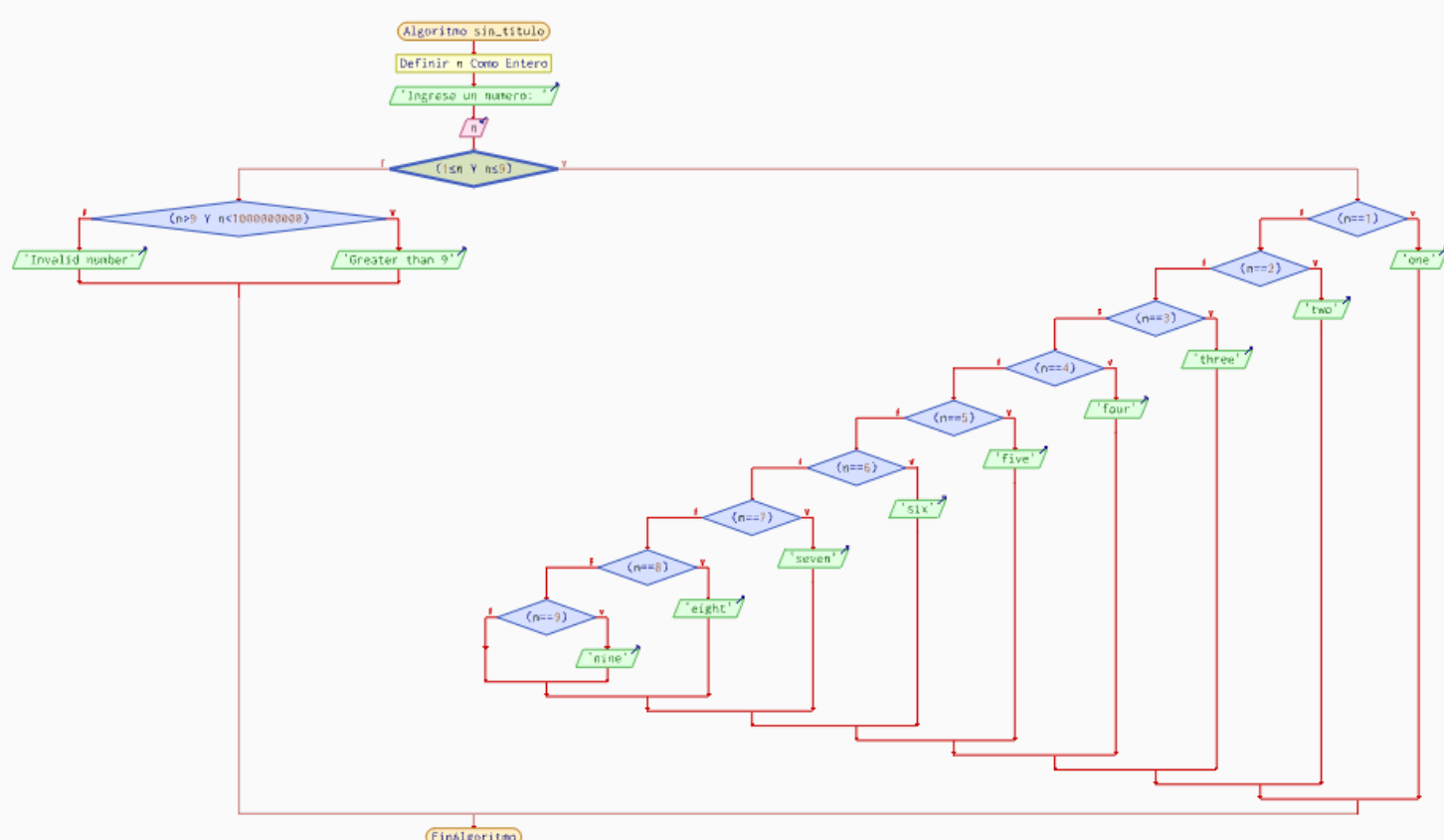
Evalúa varias condiciones en cadena hasta encontrar una verdadera. Sirve para decisiones con múltiples opciones excluyentes [2].

Planteamiento del problema

El programa debe recibir un entero positivo y realizar lo siguiente: si $1 \leq n \leq 9$, debe imprimir la palabra en inglés en minúsculas correspondiente al número (por ejemplo, one, two, etc.). Si $n > 9$, debe imprimir Greater than 9.

- Formato de entrada: la primera línea contiene un único número entero n .
- Restricciones: $1 \leq n \leq 10^9$.
- Formato de salida: si $1 \leq n \leq 9$, imprimir la palabra en inglés en minúsculas correspondiente al número; de lo contrario, imprimir Greater than 9.

Diagrama de flujo



Implementación en lenguaje C.

```
#include <stdio.h>
int main() {
    int n;
    printf("Ingrese un numero: ");
    scanf("%d", &n);
    if (1 <= n && n <= 9) {
        if (n == 1) {
            printf("one\n");
        }
        else if (n == 2) {
            printf("two\n");
        }
        else if (n == 3) {
            printf("three\n");
        }
        else if (n == 4) {
            printf("four\n");
        }
        else if (n == 5) {
            printf("five\n");
        }
    }
```

```
        else if (n == 6) {
            printf("six\n");
        }
        else if (n == 7) {
            printf("seven\n");
        }
        else if (n == 8) {
            printf("eight\n");
        }
        else if (n == 9) {
            printf("nine\n");
        }
    }
    else if (n > 9 && n < 1000000000) {
        printf("Greater than 9\n");
    }
    else {
        printf("Invalid number\n");
    }
    return 0;
}
```

Conclusiones

- Las estructuras condicionales son esenciales porque permiten que los programas tomen decisiones y se ajusten a diferentes situaciones, mostrando su papel central en el control del flujo lógico.
- Su correcta aplicación aporta flexibilidad y precisión a los algoritmos, lo que demuestra su importancia en la construcción de soluciones claras y funcionales.
- Analizar su uso permite comprender cómo influyen en la organización y eficiencia del software, evidenciando su valor fundamental en la programación.

Bibliografia

[1] L. Joyanes Aguilar, Programación en C: Metodología, algoritmos y estructura de datos. McGraw-Hill, 2005. Disponible en:

<https://elhacker.info/manuales/Lenguajes%20de%20Programacion/C/Programacion%20en%20C.%20Metodologia%2C%20algoritmos%20y%20estructura%20de%20datos.%20Luis%20Joyanes.pdf>

[2] Editorial CIMTED, Introducción a la programación con C. CIMTED, 2022. Disponible en:

<https://memoriascimted.com/wp-content/uploads/2022/04/Libro-Introduccion-a-la-programacion-con-C.pdf>