TECNOLÓGICA DEL ORIENTE
INSTITUCIÓN DE EDUCACIÓN SUPERIOR

FUNDAMENTOS DE PROGRAMACIÓN

ESTRUCTURAS CONDICIONALES EN PROGRAMACIÓN

ESTRUCTURAS CONDICIONALES EN PROGRAMACIÓN

Las estructuras condicionales son mecanismos esenciales que permiten a los programas tomar decisiones según diferentes condiciones lógicas. Su uso es clave para adaptar el comportamiento del software a diversas situaciones, comparando valores, validando datos o controlando el flujo de ejecución. Existen distintos tipos de estructuras condicionales, como la simple, la doble, la múltiple y las condicionales anidadas. Cada una ofrece distintos niveles de complejidad y flexibilidad para gestionar las decisiones dentro del código. Comprender su funcionamiento y sus aplicaciones permite diseñar programas más claros, adaptables y precisos.

Estructura Condicional Simple

La estructura condicional simple es uno de los elementos básicos de control de flujo en programación. Permite a un programa tomar decisiones basadas en la evaluación de una condición. Si la condición es verdadera, se ejecuta un bloque de código; de lo contrario, el programa continúa su ejecución sin realizar ninguna acción en ese bloque condicional. Esta estructura es fundamental para permitir que el programa se comporte de manera dinámica y responda a diferentes situaciones (Juganaru Mathieu, 2015).

Definición de la Estructura Condicional Simple

La estructura condicional simple se compone de la palabra clave if (o su equivalente en otros lenguajes) seguida de una condición que se evalúa. Si la condición es verdadera, se ejecuta un bloque de código; si es falsa, el programa no realiza ninguna acción dentro de ese bloque. No requiere una cláusula “else” para su funcionamiento, ya que su única preocupación es verificar si una condición es verdadera y actuar en consecuencia (Arbones Malisani, 1991).

Sintaxis general:

Figura 1. Condicional simple

```
if (condicion) {
    // Bloque de código a ejecutar si la condición es verdadera
}
```

Funcionamiento de la Estructura Condicional Simple

La condición dentro del if puede ser cualquier expresión que retorne un valor booleano (verdadero o falso). Cuando el programa evalúa esta condición, realiza una comparación, cálculo o comprobación. Si el resultado es verdadero, se ejecuta el bloque de código que sigue al if.

Evaluación de Condición Verdadera

Si la condición dentro del if es verdadera, el bloque de código se ejecuta. Este bloque puede contener una o más instrucciones que se llevarán a cabo.

Figura 2. Condición verdadera

```c
int edad = 18;

if (edad >= 18) {
    printf("Eres mayor de edad.\n");
}
```

En este caso, la condición edad >= 18 se evalúa. Dado que la edad es 18, la condición es verdadera, por lo que el mensaje "Eres mayor de edad." se imprime en la pantalla.

Evaluación de Condición Falsa

Si la condición es falsa, el bloque de código dentro del if se omite y el programa continúa con las siguientes instrucciones que no están dentro del condicional.

Figura 3. Condicion false

```c
int edad = 16;

if (edad >= 18) {
    printf("Eres mayor de edad.\n");
}
```

En este ejemplo, la condición edad >= 18 es falsa porque la edad es 16. Como resultado, el bloque de código no se ejecuta y no se imprime el mensaje.

Ventajas y Limitaciones de la Estructura Condicional Simple

Ventajas:

- **Simplicidad:** La estructura condicional simple es fácil de entender y utilizar, lo que la convierte en una herramienta útil para principiantes.
- **Claridad en la toma de decisiones:** Permite que un programa tome decisiones de manera sencilla sin complicar el flujo de ejecución.

Limitaciones:

- **Solo evalúa una condición:** La estructura condicional simple solo permite verificar una condición. Si se requieren múltiples verificaciones, se debe usar la estructura condicional compleja o combinar múltiples condicionales.
- **No permite tomar acciones cuando la condición es falsa:** A diferencia de otras estructuras, como el if-else, no proporciona una opción de acción en caso de que la condición sea falsa.

Estructura Condicional Doble

La estructura condicional doble es un mecanismo fundamental en la programación que permite evaluar una condición y ejecutar diferentes bloques de código dependiendo de si la condición es verdadera o falsa. A diferencia de la estructura condicional simple, que solo ejecuta una acción si la condición es verdadera, la

estructura condicional doble garantiza que siempre se ejecute uno de los dos posibles bloques de código (Arbones Malisani, 1991).

Esta estructura es ampliamente utilizada para controlar el flujo del programa y tomar decisiones lógicas basadas en comparaciones, validaciones o cálculos.

Definición de la Estructura Condicional Doble

La estructura condicional doble utiliza la palabra clave if para evaluar una condición y, en caso de que esta sea falsa, ejecuta un segundo bloque de código definido con else.

La lógica detrás de esta estructura es simple:

- Si la condición es verdadera, se ejecuta el bloque de código dentro del if.
- Si la condición es falsa, se ejecuta el bloque de código dentro del else.

Figura 4. Condicional doble

```
if (condicion) {
    // Código que se ejecuta si la condición es verdadera
} else {
    // Código que se ejecuta si la condición es falsa
}
```

Funcionamiento de la Estructura Condicional Doble

La evaluación de la condición se realiza mediante operadores lógicos o de comparación. Dependiendo del resultado de esta evaluación, el programa decide cuál de los dos bloques de código ejecutar.

Figura 5. Ejemplo en C.

```
int main() {
    int edad = 17;

    if (edad >= 18) {
        printf("Eres mayor de edad.\n");
    } else {
        printf("Eres menor de edad.\n");
    }

    return 0;
}
```

Explicación:

- Si la edad es mayor o igual a 18, se ejecuta el bloque dentro de if ("Eres mayor de edad.").
- Si la edad es menor a 18, se ejecuta el bloque dentro de else ("Eres menor de edad.").

Aplicaciones Comunes de la Estructura Condicional Doble

Esta estructura se utiliza en diversas situaciones dentro del desarrollo de software:

- ☑ **Verificación de permisos:** Determinar si un usuario tiene acceso a una sección de una aplicación.
- ☑ **Validación de datos de entrada:** Asegurar que el usuario ingrese valores dentro de un rango permitido.
- ☑ **Clasificación de valores:** Determinar si un número es positivo o negativo.

Figura 6. Ejemplo en Python

```
nota = 85

if nota >= 90:
    print("Obtuvo una calificación excelente.")
else:
    if nota >= 60:
        print("Aprobó el curso.")
    else:
        print("Reprobó el curso.")
```

Aquí se verifican varias condiciones dentro de una estructura anidada:

- Si nota >= 90, se considera una calificación excelente.
- Si la nota está entre 60 y 89, el estudiante aprueba.
- Si es menor a 60, reprueba.
- Estructura Condicional Múltiple (if-else if-else)

La estructura condicional múltiple permite evaluar varias condiciones de manera secuencial dentro de un programa. A diferencia de las estructuras condicionales simples y dobles, esta permite definir múltiples escenarios posibles para la ejecución del código, proporcionando un control de flujo más detallado y flexible (Juganaru Mathieu, 2015).

Este tipo de estructura se basa en el uso de if, else if (o su equivalente en otros lenguajes) y, opcionalmente, else, lo que permite verificar distintas condiciones en orden hasta que una de ellas se cumpla.

Definición de la Estructura Condicional Múltiple

La estructura condicional múltiple se utiliza cuando se deben evaluar varias condiciones posibles dentro de un mismo bloque de código. A medida que el programa ejecuta la estructura, revisa cada condición en orden hasta encontrar la primera que sea verdadera, ignorando las demás. Si ninguna condición es verdadera, el bloque de else se ejecuta, si está presente.

Figura 7. Condicional múltiple

```
if (condicion1) {
    // Código que se ejecuta si la condición1 es verdadera
} else if (condicion2) {
    // Código que se ejecuta si la condición2 es verdadera
} else if (condicion3) {
    // Código que se ejecuta si la condición3 es verdadera
} else {
    // Código que se ejecuta si ninguna de las condiciones anteriores es verdadera
}
```

Esta estructura permite evaluar distintas posibilidades sin necesidad de anidar múltiples estructuras if-else, lo que mejora la legibilidad del código.

Funcionamiento de la Estructura Condicional Múltiple

La ejecución de esta estructura sigue un proceso lógico secuencial:

1. Se evalúa la primera condición (if). Si es verdadera, se ejecuta su bloque de código y se ignoran las siguientes condiciones.
2. Si la primera condición es falsa, el programa evalúa la siguiente (else if).
3. Este proceso se repite hasta encontrar una condición verdadera o llegar al bloque else, en caso de que ninguna condición anterior se haya cumplido.

Figura 8. Condicional múltiple en C

```
int main() {
    int temperatura = 18;

    if (temperatura > 30) {
        printf("Hace mucho calor.\n");
    } else if (temperatura >= 20) {
        printf("El clima es agradable.\n");
    } else if (temperatura >= 10) {
        printf("Hace un poco de frío.\n");
    } else {
        printf("Hace mucho frío.\n");
    }

    return 0;
}
```

Explicación:

- Si la temperatura es mayor a 30, se imprime "Hace mucho calor."
- Si está entre 20 y 30, se imprime "El clima es agradable."
- Si está entre 10 y 19, se imprime "Hace un poco de frío."
- Si es menor a 10, se imprime "Hace mucho frío."
- Aplicaciones Comunes de la Estructura Condicional Múltiple

Esta estructura es muy utilizada en distintos contextos dentro del desarrollo de software, como:

- **Clasificación de datos:** Determinar rangos de calificaciones, temperaturas o edades.
- **Validación de entrada del usuario:** Asignar diferentes respuestas según lo que el usuario ingrese.
- **Manejo de estados en un programa:** Controlar diferentes estados en una aplicación o videojuego.

Figura 9. Ejemplo de Clasificación de Usuarios por Edad

```c
int edad;

printf("Ingrese su edad: ");
scanf("%d", &edad);

if (edad < 13) {
    printf("Eres un niño.\n");
} else if (edad < 18) {
    printf("Eres un adolescente.\n");
} else if (edad < 60) {
    printf("Eres un adulto.\n");
} else {
    printf("Eres un adulto mayor.\n");
}
```

Ventajas y Desventajas

Ventajas

- Permite evaluar múltiples condiciones sin anidar múltiples if-else, lo que mejora la claridad del código.
- Evita código redundante al centralizar las condiciones en una única estructura.
- Fácil de mantener y extender, ya que se pueden agregar más else if conforme sea necesario.

Desventajas

- Puede afectar el rendimiento si hay muchas condiciones, ya que el programa evalúa cada else if hasta encontrar una verdadera.
- No es la mejor opción cuando se deben comparar valores exactos, en cuyo caso una estructura switch puede ser más eficiente.

Uso de Condicionales Anidados

Los condicionales anidados son estructuras en las que una sentencia condicional se encuentra dentro de otra. Este tipo de estructuras permiten evaluar situaciones más complejas en las que es necesario verificar múltiples condiciones antes de ejecutar un bloque de código específico. Su uso es común en la programación cuando las decisiones dependen de varias condiciones interrelacionadas (Juganaru Mathieu, 2015).

Definición de Condicionales Anidados

Un condicional anidado ocurre cuando una estructura if, if-else o if-else if-else se encuentra dentro del bloque de otra estructura condicional. Esto permite realizar verificaciones adicionales basadas en el resultado de una condición evaluada previamente.

Figura 10. Condicionales anidados

```
if (condicion1) {
    if (condicion2) {
        // Código si ambas condiciones son verdaderas
    } else {
        // Código si condicion1 es verdadera, pero condicion2 es falsa
    }
} else {
    // Código si condicion1 es falsa
}
```

Funcionamiento de los Condicionales Anidados

Cuando se usa un condicional anidado, la ejecución del código sigue una estructura jerárquica:

1. La condición externa se evalúa primero.
2. Si la condición externa es verdadera, se evalúa la condición interna.
3. Dependiendo del resultado de la condición interna, se ejecuta un bloque de código.
4. Si la condición externa es falsa, el flujo del programa sigue la alternativa correspondiente.

Figura 11. Ejemplo en Python: Clasificación de un estudiante según su calificación

```
calificacion = int(input("Ingrese la calificación: "))

if calificacion >= 60:
    print("El estudiante ha aprobado.")

    if calificacion >= 90:
        print("Obtuvo una calificación excelente.")
    elif calificacion >= 80:
        print("Obtuvo una calificación muy buena.")
    elif calificacion >= 70:
        print("Obtuvo una calificación buena.")
    else:
        print("Aprobó con una calificación baja.")

else:
    print("El estudiante ha reprobado.")
```

Explicación:

- Se verifica si la calificación es mayor o igual a 60 (aprobado) o menor a 60 (reprobado).

- Si el estudiante aprueba, se analiza en qué categoría de desempeño se encuentra.

Ventajas y Desventajas de los Condicionales Anidados

Ventajas

- Permiten evaluar múltiples condiciones de manera organizada.
- Evitan la redundancia en el código, al agrupar verificaciones relacionadas en un mismo bloque.
- Mejoran la precisión en la toma de decisiones, ya que permiten manejar casos específicos con detalle.

Desventajas

- Pueden reducir la legibilidad del código si se anidan demasiadas estructuras if-else.
- Aumentan la complejidad del programa, lo que puede dificultar su mantenimiento.
- Si no se estructuran correctamente, pueden generar errores lógicos, como condiciones que nunca se cumplen.

Optimización de Condicionales

La optimización de condicionales es un proceso clave en la programación que permite mejorar el rendimiento y la legibilidad del código al reducir la complejidad innecesaria en las estructuras de control de flujo. Un código más eficiente no solo se ejecuta más rápido, sino que también es más fácil de mantener y depurar.

Importancia de la Optimización de Condicionales

El uso ineficiente de condicionales puede afectar negativamente el rendimiento del software, especialmente cuando se trabaja con grandes volúmenes de datos o procesos repetitivos. Algunas razones por las que es importante optimizar los condicionales incluyen:

- Reducción de la cantidad de comparaciones necesarias para ejecutar un bloque de código.
- Mejora en la claridad del código, facilitando su comprensión y mantenimiento.
- Evitar cálculos innecesarios, disminuyendo la carga computacional.
- Minimiza errores lógicos, reduciendo la probabilidad de comportamientos inesperados.

Uso de Estructuras switch-case en Lugar de if-else

Cuando se trabaja con múltiples comparaciones de igualdad, una estructura switch-case (disponible en lenguajes como C, Java y JavaScript) es más eficiente que una serie de if-else.

Figura 12. Ejemplo antes de la optimización (JavaScript)

```javascript
let opcion = 2;

if (opcion == 1) {
    console.log("Seleccionó la opción 1.");
} else if (opcion == 2) {
    console.log("Seleccionó la opción 2.");
} else if (opcion == 3) {
    console.log("Seleccionó la opción 3.");
} else {
    console.log("Opción no válida.");
}
```

Figura 13. Ejemplo optimizado con switch-case

```javascript
let opcion = 2;

switch (opcion) {
    case 1:
        console.log("Seleccionó la opción 1.");
        break;
    case 2:
        console.log("Seleccionó la opción 2.");
        break;
    case 3:
        console.log("Seleccionó la opción 3.");
        break;
    default:
        console.log("Opción no válida.");
}
```

En este caso, switch-case mejora la legibilidad y la eficiencia del código al evaluar directamente el valor de opción sin necesidad de múltiples comparaciones.

Uso de Operadores Ternarios para Condiciones Simples

El operador ternario (? :) Permite reducir la cantidad de líneas de código cuando se necesita asignar un valor o realizar una acción en función de una condición.

Figura 14. Ejemplo antes de la optimización

```c
if (edad >= 18) {
    printf("Es mayor de edad.\n");
} else {
    printf("Es menor de edad.\n");
}
```

Figura 15. Ejemplo optimizado con operador ternario

```c
printf("%s\n", edad >= 18 ? "Es mayor de edad." : "Es menor de edad.");
```

Aquí, el operador ternario elimina la necesidad de una estructura if-else, reduciendo la cantidad de líneas sin comprometer la claridad.

Bibliografía

- Arbones Malisani, E. A. (1991). Ingeniería de sistemas. Marcombo. https://elibro.net/es/lc/tecnologicadeloriente/titulos/101860
- Juganaru Mathieu, M. (2015). Introducción a la programación. Grupo Editorial Patria. https://elibro.net/es/lc/tecnologicadeloriente/titulos/39449