switch(expresión) {

case constante1:

// Código a ejecutar si expresión == constante1

break;

case constante2:

// Código a ejecutar si expresión == constante2

break;

// ... más casos

default:

// Código a ejecutar si ninguna de las constantes coincide

}

Usa el código [con precaución](/faq#coding).

**Explicación:**

* **switch(expresión):**
* switch es la palabra clave que inicia el ciclo.
* expresión es una expresión cuyo valor se evaluará en tiempo de ejecución. Debe ser un tipo de dato entero o un tipo de dato enumerado.
* **case constante1::**
* case es la palabra clave que introduce un caso específico.
* constante1 es un valor constante que se compara con el resultado de la expresión.
* Si la expresión es igual a constante1, se ejecuta el código que sigue a este case.
* **break;:**
* break es una palabra clave que termina el bloque de código actual y sale del ciclo switch. Si no se incluye break, el programa continuará ejecutando los casos siguientes hasta encontrar un break o el final del ciclo.
* **default::**
* default es la palabra clave que introduce un caso por defecto.
* Si ninguna de las constantes de los casos anteriores coincide con el valor de la expresión, se ejecuta el código que sigue a default.

#include <stdio.h>

int main() {

int dia = 3;

switch (dia) {

case 1: printf("Lunes\n");

break;

case 2: printf("Martes\n");

break;

case 3: printf("Miércoles\n");

break;

default: printf("Día inválido\n");

}

return 0;

}

**Puntos clave a recordar:**

* La expresión debe ser de un tipo de dato entero o enumerado.
* Los valores de los case deben ser constantes y únicos.
* El break es fundamental para evitar que el programa siga ejecutando los casos siguientes innecesariamente.
* El default es opcional, pero es útil para manejar casos no contemplados.

**While**

**Explicación:**

* **while:** Es la palabra clave que indica el inicio de un bucle.
* **condición:** Es una expresión lógica que se evalúa antes de cada iteración del bucle. Puede ser cualquier expresión que resulte en un valor verdadero (1) o falso (0).
* **{ y }:** Las llaves delimitan el bloque de código que se ejecutará repetidamente mientras la condición sea verdadera.

**Cómo funciona:**

1. **Evaluación de la condición:** Al llegar al ciclo while, se evalúa la condición.
2. **Ejecución del bloque:** Si la condición es verdadera, se ejecuta todo el código dentro del bloque de llaves.
3. **Repetición:** Después de ejecutar el bloque, se vuelve al paso 1 y se vuelve a evaluar la condición. Si sigue siendo verdadera, se vuelve a ejecutar el bloque.
4. **Terminación:** El ciclo continúa hasta que la condición se vuelve falsa. En ese momento, el programa continúa con la siguiente instrucción después del bloque while.

**Ejemplo:**

C

#include <stdio.h>

int main() {

int i = 1;

while (i <= 5) {

printf("El valor de i es: %d\n", i);

i++;

}

return 0;

}

Usa el código [con precaución](/faq#coding).

Este código imprimirá los números del 1 al 5 en la consola.

**Explicación del ejemplo:**

1. Se inicializa la variable i en 1.
2. Se entra al ciclo while. La condición i <= 5 es verdadera, así que se ejecuta el bloque.
3. Se imprime el valor actual de i y luego se incrementa en 1.
4. Se vuelve al inicio del ciclo y se vuelve a evaluar la condición. Como i ahora es 2, la condición sigue siendo verdadera y se ejecuta el bloque nuevamente.
5. Este proceso se repite hasta que i sea mayor que 5. En ese momento, la condición se vuelve falsa y el ciclo termina.

**Usos comunes del ciclo while:**

* **Repetir un bloque de código un número desconocido de veces:** Por ejemplo, leer datos de un archivo hasta encontrar un valor específico.
* **Crear menús interactivos:** Presentar opciones al usuario y repetir el menú hasta que el usuario seleccione una opción válida.
* **Implementar algoritmos iterativos:** Como la búsqueda binaria o el algoritmo de ordenamiento burbuja.

**Ciclo do while**

do {

// Bloque de código a ejecutar al menos una vez

} while (condición);

Usa el código [con precaución](/faq#coding).

**Explicación:**

* **do:** Esta palabra clave inicia el ciclo do-while.
* **{ y }:** Las llaves delimitan el bloque de código que se ejecutará al menos una vez.
* **while (condición);:** Después del bloque de código, se evalúa la condición. Si la condición es verdadera, se vuelve a ejecutar el bloque. Si es falsa, el ciclo termina.

**¿Cuál es la principal diferencia con el ciclo while?**

* En el ciclo while, la condición se evalúa al principio. Si la condición es falsa desde el inicio, el bloque de código no se ejecuta ni una sola vez.
* En el ciclo do-while, el bloque de código se ejecuta **al menos una vez** y luego se evalúa la condición. Esto significa que incluso si la condición es falsa desde el principio, el bloque se ejecutará una vez antes de terminar el ciclo.

**Ejemplo:**

C

#include <stdio.h>

int main() {

int numero;

do {

printf("Ingrese un número positivo: ");

scanf("%d", &numero);

} while (numero <= 0);

printf("¡Gracias! El número ingresado es positivo: %d\n", numero);

return 0;

}

Usa el código [con precaución](/faq#coding).

En este ejemplo:

1. Se declara una variable numero.
2. El ciclo do-while se ejecuta al menos una vez. Se solicita al usuario que ingrese un número.
3. Se evalúa la condición numero <= 0. Si el número es negativo o cero, se vuelve a pedir al usuario que ingrese un número.
4. Si el número es positivo, la condición es falsa y el ciclo termina.

**Usos comunes del ciclo do-while:**

* Cuando necesitas que un bloque de código se ejecute al menos una vez, independientemente de la condición inicial.
* En menús interactivos donde quieres asegurarte de que el usuario vea al menos una vez las opciones disponibles.
* En situaciones donde la condición de salida se determina al final de la iteración.

**En resumen:**

El ciclo do-while es una estructura de control que garantiza que un bloque de código se ejecute al menos una vez. Es útil cuando necesitas ejecutar un conjunto de instrucciones y luego decidir si repetirlas basándote en una condición.