

Usar condiciones if/else:

Una parte fundamental de la programación implica tomar decisiones basadas en los datos. En esta unidad, descubriremos cómo controlar las acciones del programa probando condiciones.


Podemos crear *ramas condicionales* en nuestro código usando las palabras clave `if` y `else`. Muchos lenguajes de programación ofrecen esta función y usan una sintaxis parecida.

Las palabras clave `if` y `else` se usan con expresiones para probar valores y realizar acciones basadas en el resultado de la prueba. Todas las expresiones condicionales tienen como resultado un valor booleano: `true` o `false`.

Definición de una condición if/else

Este es un ejemplo que comprueba si dos números son iguales e imprime un mensaje basado en el resultado de la prueba:

Rust

 Copiar

```
if 1 == 2 {  
    println!("True, the numbers are equal."); //  
} else {  
    println!("False, the numbers are not equal.");  
}
```

En este ejemplo, la condición de `if` es la expresión `1 == 2`, que se evalúa en un tipo booleano con el valor `false`.

A diferencia de la mayoría de los demás lenguajes, los bloques de `if` en Rust también pueden actuar como expresiones. Todos los bloques de ejecución de las ramas de

condición deben devolver el mismo tipo para que se compile el código.

```
Rust Copiar  
  
let formal = true;  
let greeting = if formal { // if used here as an expression  
    "Good day to you."    // return a String  
} else {  
    "Hey!"                // return a String  
};  
println!("{}", greeting) // prints "Good day to you."
```

En este ejemplo, asignamos un valor a la variable `greeting` en función del resultado de la expresión `if formal`. Cuando la expresión `if formal` es `true`, el valor `greeting` se establece en la cadena "Good day to you". Cuando la expresión es `false`, el valor `greeting` se establece en la cadena "Hey!". Dado que inicializamos la variable `formal` en `true`, sabemos que el resultado de la expresión `if formal` también es `true`.


Combinación de varias condiciones de prueba:

Puede combinar `if` y `else` para formar una expresión `else if`. Se pueden usar varias condiciones `else if` después de la condición `if` de inicio y antes de una `else` de cierre, que es opcional.

Si una expresión condicional se evalúa como `true`, se ejecutará el bloque de acciones correspondiente. Todos los bloques `else if` o `else` siguientes se omitirán. Si una expresión de condición se evalúa como `false`, se omitirá el bloque de acciones correspondiente. Se evaluará cualquier condición `else if` siguiente. Si todas las condiciones `if` y `else if` se evalúan como `false`, se ejecuta cualquier bloque `else`.

En este ejemplo, se comprueba si un número está dentro de un intervalo permitido. Queremos realizar algún procesamiento específico cuando el número sea menor que cero, igual a cero o mayor que 512. Declaramos la variable booleana `out_of_range`, pero no establecemos el valor de variable hasta que el programa entra en la expresión de prueba condicional.

Rust

 Copiar

```
let num = 500 // num variable can be set at some point in the program
let out_of_range: bool;
if num < 0 {
    out_of_range = true;
} else if num == 0 {
    out_of_range = true;
} else if num > 512 {
    out_of_range = true;
} else {
    out_of_range = false;
}
```