Trabajo con funciones en Rust:

Las funciones son la principal forma de ejecutar código en Rust. Como ya hemos visto, una de las funciones más importantes en el lenguaje, es la función main(). En esta unidad, se tratarán más detalles sobre cómo definir funciones.

Definir una función:

Las definiciones de funciones en Rust comienzan por la palabra clave fn. Después del nombre de la función, se especifican los argumentos de entrada de esta, como una lista separada por comas de los tipos de datos entre paréntesis. Las llaves indican al compilador dónde comienza y termina el cuerpo de la función.

Rust Copiar

```rust
fn main() {
    println!("Hello, world!");
    goodbye();
}

fn goodbye() {
    println!("Goodbye.");
}
```

Para llamar a una función, se usa su nombre junto con sus argumentos de entrada entre paréntesis. Si una función no tiene argumentos de entrada, los paréntesis se dejan vacíos. En nuestro ejemplo, las funciones main() y goodbye() no tienen argumentos de entrada.

Es posible que haya observado que la función goodbye() se define después de la función main(). Podríamos haber definido la función goodbye() antes de definir main(). A Rust no le importa en qué parte del archivo se definan las funciones, mientras estas se definan dentro de él.

Pasar argumentos de entrada:

Cuando una función tiene argumentos de entrada, se asigna un nombre a cada argumento y se especifica el tipo de dato al principio de la declaración de la función. Dado que los argumentos se llaman como las variables, podemos acceder a los argumentos en el cuerpo de la función.

Ahora se modificará la función goodbye() para que tome un puntero a algunos datos de cadena como argumento de entrada.

Rust Copiar

```rust
fn goodbye(message: &str) {
    println!("\n{}", message);
}

fn main() {
    let formal = "Formal: Good bye.";
    let casual = "Casual: See you later!";
    goodbye(formal);
    goodbye(casual);
}
```

Para probar la función, esta sera llamada desde la función main() pasandole al mismo tiempo dos argumentos diferentes y, después, se comprobará la salida:

Resultados Copiar

```
Formal: Good bye.
Casual: See you later!
```

Devolución de un valor:

Cuando una función devuelve un valor, agregamos la sintaxis (→ <type>), después de la lista de argumentos de función y antes de la llave de apertura del cuerpo de la función. La sintaxis de flecha → indica que la función

devuelve un valor al autor de la llamada. La parte <type> permite al compilador conocer el tipo de datos del valor devuelto.

En Rust, lo habitual es devolver un valor al final de una función haciendo que la última línea de código de la función sea igual al valor que se va a devolver. En el ejemplo siguiente se muestra este comportamiento. La función divide_by_5() devuelve el resultado de dividir el número de entrada entre 5 para la función que realiza la llamada:

Rust Copiar

```rust
fn divide_by_5(num: u32) -> u32 {
    num / 5
}

fn main() {
    let num = 25;
    println!("{} divided by 5 = {}", num, divide_by_5(num));
}
```

Esta es la salida:

Resultados Copiar

```
25 divided by 5 = 5
```

Se puede usar la palabra clave return en cualquier punto de la función para detener la ejecución y devolver un valor al autor de la llamada. Normalmente, la palabra clave return se usa con un condicional.

Este es un ejemplo en el que se usa explícitamente la palabra clave return para devolver anticipadamente un valor desde una función, en el caso del ejemplo este sera devuelto si este es igual a 0:

Rust Copiar

```rust
fn divide_by_5(num: u32) -> u32 {
    if num == 0 {
        // Return early
        return 0;
    }
    num / 5
}
```

Cuando se usa la palabra clave return de forma explícita, se finaliza la instrucción con un punto y coma.

Si se devuelve un valor sin hacer uso de la palabra clave return, la ultima instruccion del bloque no terminara con punto y coma. Es posible que haya observado que no se ha usado el punto y coma final para la instrucción de valor devuelto (num / 5).

Revisión de la firma:

La primera parte de la declaración de una función se denomina firma de función.

La firma de la función goodbye() de nuestro ejemplo tiene estas características:

fn: Palabra clave de la declaración de función en Rust.

goodbye: Nombre de la función.

(message: &str): El argumento o la lista de parámetros de la función. Se espera un puntero a los datos de cadena como valor de entrada.

→ bool: La flecha apunta al tipo de valor que esta función devolverá siempre.

La función goodbye acepta un puntero de cadena como entrada y genera un valor booleano.