分隔。参数的类型必须在函数签名中声明，这样编译器就能知道如何处理和传递这些参数。函数参数是传递给函数的值，它们可以在函数体内被使用。

返回类型：函数可以返回一个值，这个值可以是任何类型。如果函数需要返回一个值，可以在函数签名中指定返回类型，并在函数体中使用return关键字来返回一个值。如果函数不需要返回值，可以使用()作为返回类型。

函数体：是由一对大括号{}包围的代码块，其中包含了执行特定操作和计算的语句。函数体中的语句以分号;结尾，表示语句的结束。函数体内可以定义其他模块，比如：变量、常量、其他函数等。

函数调用：是通过在函数名后面加上一对圆括号()来完成的，圆括号中可以传递参数给函数。如果函数不需要参数，圆括号可以省略。Rust不关心函数定义所在的位置，只要函数被调用时，出现在调用之处可见的作用域内即可。注意：与C/C++不同，Rust中没有类似头文件的概念，函数的定义和实现都在rs文件中编写。

函数的具体用法

1、无参函数。在下面的示例代码中，我们定义了一个名为greet的无参函数，用于打印一条问候信息。在main函数中，我们调用了这个greet函数。

```
fn greet() {
    println!(”Hello, CSDN”);
}

fn main() {
    greet() {
    // 使用return语句明确返回值
    return x + y;
}
```

2、使用函数体的最后一个表达式作为隐式返回值。

如果函数体只包含一个表达式，并且这个表达式的结果与函数签名中声明的返回类型相匹配，那么这个表达式的结果将自动作为函数的返回值，无需使用return语句。

```
fn add(x: i32, y: i32) -> i32 {
    // 最后一个表达式作为隐式返回值
    x + y
}
```

另外，在Rust中，不仅整个函数体可以有一个返回值，单个表达式也可以有返回值。这意味着你可以在函数中的任何位置使用表达式，并且该表达式的结果可以被用作函数的返回值，或者在更大的表达式中被进一步使用。

```
fn main() {
    let a = 66;

    let b = {
        let a = 99;
        // 代码块中，最后一个表达式作为返回值
        a + 1
    };

    // 输出：66, 100
    println!("{0}, {1}", a, b);
}
```

原文链接: https://juejin.cn/post/7363078360787091495