Rust 讀書會 (2)

劉安齊 Liu, An-Chi

Variable and Mutability

```rust
fn main() {
    let x = 5;
    println!("The value of x is: {}", x);
    x = 6;
    println!("The value of x is: {}", x);
}
```

```
error[E0384]: cannot assign twice to immutable variable `x`
 --> src/main.rs:4:5
  |
2 |     let x = 5;
  |         - first assignment to `x`
3 |     println!("The value of x is: {}", x);
4 |     x = 6;
  |     ^^^^^ cannot assign twice to immutable variable

error: aborting due to previous error
```

Mutable

```rust
fn main() {
    let mut x = 5;
    println!("The value of x is: {}", x);
    x = 6;
    println!("The value of x is: {}", x);
}
```

```
The value of x is: 5
The value of x is: 6
```

```rust
fn main() {
    const PI : f64 = 3.14;
    println!("PI is {}", PI);
}

// PI is 3.14
```

```rust
fn main() {
    const MAX_POINTS: u32 = 100_000;
    println!("{}", MAX_POINTS);
}

// 100000
```

Shadowing

```rust
fn main() {
    let x = 5;

    let x = x + 1; // 6

    let x = x * 2; // 12

    println!("The value of x is: {}", x);
}
```

Shadowing

```rust
fn main() {
    let x = "tiger";
    println!("{}", x); // tiger
    let x = x.len();
    println!("{}", x); // 5
}
```

Shadowing

```
fn main() {
    let mut x = "tiger";
    println!("{}", x);
    x = x.len();
    println!("{}", x);
}

error[E0308]: mismatched types
 --> src/main.rs:4:9
  |
4 |     x = x.len();
  |         ^^^^^^^ expected &str, found usize
  |
  = note: expected type `&str`
             found type `usize`
```

Type

- Scalar type
  - Integers
  - Floating-point numbers
  - Booleans
  - Characters

Number

Length	Signed	Unsigned
8-bit	`i8`	`u8`
16-bit	`i16`	`u16`
32-bit	`i32`	`u32`
64-bit	`i64`	`u64`
arch	`isize`	`usize`

$-(2^{n-1})$ to $2^{n-1} - 1$

32 or 64 by OS

Number

```rust
fn main() {
    let x = 2.0; // f64 as defualt

    let y: f32 = 3.0; // f32
}
```

Operator

```rust
fn main() {
    // addition
    let sum = 5 + 10;

    // subtraction
    let difference = 95.5 - 4.3;

    // multiplication
    let product = 4 * 30;

    // division
    let quotient = 56.7 / 32.2;

    // remainder
    let remainder = 43 % 5;
}
```

Character

```rust
fn main() {
    let c = 'z';
    let z = 'ℤ';
    let heart_eyed_cat = '😻';
}
```

Boolean

```rust
fn main() {
    let t = true;

    let f: bool = false; // with explicit type annotation
}
```

Tuple

```rust
fn main() {
    let tup: (i32, f64, u8) = (500, 6.4, 1);
}

fn main() {
    let tup = (500, 6.4, 1);

    let (x, y, z) = tup;

    println!("{}, {}, {}", x, y, z);
}
```

Tuple

```rust
fn main() {
    let tup: (i32, i32, i32);
    tup = (0, 1, 2);

    println!("{}, {}, {}", tup.0, tup.1, tup.2);
}
```

Tuple

```rust
fn main() {
    let x: (i32, f64, u8) = (500, 6.4, 1);

    let five_hundred = x.0;

    let six_point_four = x.1;

    let one = x.2;
}
```

Tuple

```rust
fn main() {
    let mut tup: (i32, i32);

    tup.0 = 5;  // not run
    tup.1 = 6;  // not run

    println!("{} {}", tup.0, tup.1);  // Error, uninitialized variable
}
```

Array

```rust
fn main() {
    let a = [1, 2, 3, 4, 5];

    let first = a[0];
    let second = a[1];
}
```

Function

```rust
fn main() {
    another_function(5);
}

fn another_function(x: i32) {
    println!("The value of x is: {}", x);
}

// The value of x is: 5
```

Function

```rust
fn main() {
    let is_true_1: bool;
    is_true_1 = check_true_1(false);

    let is_true_2: bool;
    is_true_2 = check_true_2(true);

    println!("{} {}", is_true_1, is_true_2);
}

fn check_true_1(x: bool) -> bool {
    return x;
}

fn check_true_2(x: bool) -> bool {
    x
}
```

Block

```rust
fn main() {
    let x;
    {
        x = 1;
        let y = 2;
    }

    println!("{}", x); // OK
    println!("{}", y); // Error
}
```

Block

```rust
fn main() {
    let x = 5; // unused in this case

    let y = {
        let x = 3;
        x + 1  //  without ";" means the return value,
               //  but we cannot use "return" syntax here
    };

    println!("The value of y is: {}", y);
}
```

```
The value of y is: 4
```

```rust
enum Browser {
    Chrome,
    Firefox
}

fn do_something(browser: Browser) -> bool {
    match browser {
        Browser::Chrome => {
            println!("we are using chrome");
        },
        Browser::Firefox => {
            println!("we are using firefox");
            return false;
        }
    }

    true
}

fn main() {
    println!("{}", do_something(Browser::Chrome));
    println!("{}", do_something(Browser::Firefox));
}
```

Why don’t put return?

Thanks @cybai

Comments

```rust
fn main() {
    let x = 5; // blablablablabla

    // blablablablabla
    let y = 6;

    if x > 5 /*...*/ {
        // ...
    }

    /* Start
    ...
    ...
    End */
}
```

Condition

```rust
fn main() {
    let number = 3;

    if number < 5 {
        println!("condition was true");
    } else {
        println!("condition was false");
    }
}


//   if (number < 5) { ... }

    warning: unnecessary parentheses around `if` condition
     --> src/main.rs:4:8
      |
    4 |     if (number < 5) {
      |        ^^^^^^^^^^^^ help: remove these parentheses
      |
      = note: #[warn(unused_parens)] on by default
```

Condition

```
fn main() {
    if x {  //  x MUST be bool.
        // ...
    }
}
```

Condition

```rust
fn main() {
    let number = 6;

    if number % 4 == 0 {
        println!("number is divisible by 4");
    } else if number % 3 == 0 {
        println!("number is divisible by 3");
    } else if number % 2 == 0 {
        println!("number is divisible by 2");
    } else {
        println!("number is not divisible by 4, 3, or 2");
    }
}
```

ternary operator

```
let x = true ? 6 : 7;
```

```rust
fn main() {
    let condition = true;
    let number = if condition {
        5  // i32
    } else {
        6  // i32, which is equal to the above
    };

    println!("The value of number is: {}", number);
}
```

```rust
fn main() {
    loop {
        println!("again!");
    }
}
```

只有遇見你，能使我永遠不停下來

While

```
fn main() {
    let a = [10, 20, 30, 40, 50];
    let mut index = 0;

    while index < 5 {
        println!("the value is: {}", a[index]);

        index = index + 1;
    }
}
```

For

```rust
fn main() {
    let a = [10, 20, 30, 40, 50];

    for element in a.iter() {
        println!("the value is: {}", element);
    }
}
```

For

```rust
fn main() {
    for number in 1..4 {
        println!("{}!", number);
    }

    println!("======");

    for number in 1..=4 {
        println!("{}!", number);
    }

    println!("======");

    for number in (1..4).rev() {
        println!("{}!", number);
    }
}
```

```
1!
2!
3!
======
1!
2!
3!
4!
======
3!
2!
1!
```

「撰寫 Rust 程式的確需要更多精力。
但相對地，當你的程式寫完且成功編譯，很高的機率它
會安然無恙的運作。」

–*Julio Merino*

“Hack Without Fear !”

–Niko Matsakis