Thank you for printing our content at www.domain-name.com. Please check back soon for new contents.

2d: 4h: 02m: 05s

Stop copy pasting code you don't actually understand

Become a PRO

(https://programiz.pro/offer/black-november-2025?utm_source=programiz.com&utm_medium=referral&utm_audience=ORGANIC-FREEMIUM&utm_campaign=offer-black-november-sale&utm_content=interests_lifetime&utm_term=sticky_banner_launch)

iz utm_source=programiz.com&utm_medium=referral&utm_audience=ORGANIC-FREEMIUM&utm_campaign=free-to-premium&utm_content=interests_learn_coding&utm_term=nav_floating_button)

to handle data and control flow of a Rust program.

We generally use the `match` expressions when it comes to pattern matching.

The syntax of the `match` expressions is:

```
match VALUE {
    PATTERN => EXPRESSION,
    PATTERN => EXPRESSION,
    PATTERN => EXPRESSION,
}
```

Here, `PATTERN => EXPRESSION` are called patterns, a special syntax in Rust which usually works together with the `match` keyword.

Matching a Variable in Rust

We can pattern match against the value of a variable. This is useful if our code wants to take some action based on a particular value. For example,

Thank you for printing our content at www.domain-name.com. Please check back soon for new contents.

2d: 4h: 02m: 05s

Stop copy pasting code you don't actually understand

Become a PRO

(https://programiz.pro/offer/black-november-2025?utm_source=programiz.com&utm_medium=referral&utm_audience=ORGANIC-FREEMIUM&utm_campaign=offer-black-november-sale&utm_content=interests_lifetime&utm_term=sticky_banner_launch)

iz utm_source=programiz.com&utm_medium=referral&utm_audience=ORGANIC-FREEMIUM&utm_campaign=free-to-premium&utm_content=interests_learn_coding&utm_term=nav_floating_button)

```
x is 2
```

Here, we have used the `match` expression to match against `x`. In the `match` body, we pattern matched with values `1`, `2` and `_`.

```
1 => println!("x is 1"),
2 => println!("x is 2"),
_ => println!("x is something else"),
```

Because the value of `x` is `2`, the pattern that matches is:

```
2 => println!("x is 2")
```

Thus, `x is 2` is printed on the screen.

Notice that we also match against underscore `_`. The `_` has a special meaning in pattern matching, if all the other patterns do not match, it defaults to `_`.

> **Note:** `match` body (also known as match arms) should always ensure that all possible cases are being handled. If all possible cases are not handled, the Rust program fails to compile.

Thank you for printing our content at www.domain-name.com. Please check back soon for new contents.

2d: 4h: 02m: 05s

Stop copy pasting code you don't actually understand

Become a PRO

(https://programiz.pro/offer/black-november-2025?utm_source=programiz.com&utm_medium=referral&utm_audience=ORGANIC-FREEMIUM&utm_campaign=offer-black-november-sale&utm_content=interests_lifetime&utm_term=sticky_banner_launch)

iz utm_source=programiz.com&utm_medium=referral&utm_audience=ORGANIC-FREEMIUM&utm_campaign=free-to-premium&utm_content=interests_learn_coding&utm_term=nav_floating_button)

```
    let my_color = Color::Green;

    // use of match expression to match against an enum v
    match my_color {
        Color::Red => println!("The color is red"),
        Color::Green => println!("The color is green"),
        Color::Blue => println!("The color is blue"),
    }
}
```

Output

```
The color is green
```

Here, we created a pattern in the `match` expression to match against all enum variants.

```
Color::Red => println!("The color is red"),
Color::Green => println!("The color is green"),
Color::Blue => println!("The color is blue"),
```

Because the value of `my_color` is `Color::Green`, the pattern that it matches is:

```
Color::Green => println!("The color is green"),
```

Thank you for printing our content at www.domain-name.com. Please check back soon for new contents.

CYBER 90% OFF Week

2d: 4h: 02m: 05s

Stop copy pasting code you don't actually understand

Become a PRO

(https://programiz.pro/offer/black-november-2025?utm_source=programiz.com&utm_medium=referral&utm_audience=ORGANIC-FREEMIUM&utm_campaign=offer-black-november-sale&utm_content=interests_lifetime&utm_term=sticky_banner_launch)

iz utm_source=programiz.com&utm_medium=referral&utm_audience=ORGANIC-FREEMIUM&utm_campaign=free-to-premium&utm_content=interests_learn_coding&utm_term=nav_floating_button)

Matching Option and Result Type in Rust

The most common case for pattern matching is with `Option` and `Result` enum types. Both the `Option` and `Result` type have two variants.

`Option` type has:

- `None` → to indicate failure with no value
- `Some(T)` → a value with type T

`Result` type has:

- `Ok(T)` → operation succeeded with value T
- `Err(E)` → operation failed with an error E

Let's look at examples of how we can use pattern matching on these types.

Thank you for printing our content at www.domain-name.com. Please check back soon for new contents.

2d: 4h: 02m: 05s

Stop copy pasting code you don't actually understand

Become a PRO

(https://programiz.pro/offer/black-november-2025?utm_source=programiz.com&utm_medium=referral&utm_audience=ORGANIC-FREEMIUM&utm_campaign=offer-black-november-sale&utm_content=interests_lifetime&utm_term=sticky_banner_launch)

iz utm_source=programiz.com&utm_medium=referral&utm_audience=ORGANIC-FREEMIUM&utm_campaign=free-to-premium&utm_content=interests_learn_coding&utm_term=nav_floating_button)

```
The option has a value of 222
```

In this example, `my_option` is an `Option` type that contains either a `Some` variant with an `i32` value or a `None` variant.

The `match` expression compares the value of `my_option` to the `Some` and `None` variants, and binds the value of `Some` variant to the `value` variable.

When a match is found, the corresponding code block is executed.

```
Some(value) => println!("The option has a value of {}", v
```

Thus, `The option has a value of 222` is printed on the screen.

Thank you for printing our content at www.domain-name.com. Please check back soon for new contents.

2d: 4h: 02m: 05s

Stop copy pasting code you don't actually understand

Become a PRO

(https://programiz.pro/offer/black-november-2025?utm_source=programiz.com&utm_medium=referral&utm_audience=ORGANIC-FREEMIUM&utm_campaign=offer-black-november-sale&utm_content=interests_lifetime&utm_term=sticky_banner_launch)

iz utm_source=programiz.com&utm_medium=referral&utm_audience=ORGANIC-FREEMIUM&utm_campaign=free-to-premium&utm_content=interests_learn_coding&utm_term=nav_floating_button)

```
The result is 100
```

In this example, `my_result` is a `Result` type that contains either an `Ok` variant with an `i32` value, or an `Err` variant with an error message of type `&str`.

The match expression compares the value of `my_result` to the `Ok` and `Err` variants, and binds the value of `Ok` variant to the `value` variable or the `Err` variant to the `error` variable.

```
Ok(value) => println!("The result is {}", value),
Err(error) => println!("The error message is {}", error)
```

When a match is found, the corresponding code block is executed.

```
Ok(value) => println!("The result is {}", value),
```

Thus, `The result is 100` is printed on the screen.

Thank you for printing our content at www.domain-name.com. Please check back soon for new contents.

2d: 4h: 02m: 05s

Stop copy pasting code you don't actually understand

Become a PRO

(https://programiz.pro/offer/black-november-2025?utm_source=programiz.com&utm_medium=referral&utm_audience=ORGANIC-FREEMIUM&utm_campaign=offer-black-november-sale&utm_content=interests_lifetime&utm_term=sticky_banner_launch)

iz utm_source=programiz.com&utm_medium=referral&utm_audience=ORGANIC-FREEMIUM&utm_campaign=free-to-premium&utm_content=interests_learn_coding&utm_term=nav_floating_button)

```
    } else {
        println!("The option has no value");
    }
}
```

Output

```
The option has a value of 111
```

Here, the `if let` expression is matching on the `my_option` variable and binding the value of `Some` variant to the `value` variable.

If the match is successful, the code inside the `if` block is executed. If the match is not successful, the code inside the `else` block is executed.

Common Use Cases of Pattern Matching in Rust

As you have seen, pattern matching is useful in numerous situations. Some common use cases for pattern matching include:

Thank you for printing our content at www.domain-name.com. Please check back soon for new contents.

2d: 4h: 02m: 05s

Stop copy pasting code you don't actually understand

Become a PRO

(https://programiz.pro/offer/black-november-2025?utm_source=programiz.com&utm_medium=referral&utm_audience=ORGANIC-FREEMIUM&utm_campaign=offer-black-november-sale&utm_content=interests_lifetime&utm_term=sticky_banner_launch)

iz utm_source=programiz.com&utm_medium=referral&utm_audience=ORGANIC-FREEMIUM&utm_campaign=free-to-premium&utm_content=interests_learn_coding&utm_term=nav_floating_button)

Next Tutorial:
Rust Array (/rust/array)

Previous Tutorial:
Rust break and continue (/rust/break-continue)

Your builder path starts here.

Escape tutorial hell and ship real projects.

Thank you for printing our content at www.domain-name.com. Please check back soon for new contents.

CYBER 90% OFF Week

2d: 4h: 02m: 05s

Stop copy pasting code you don't actually understand

Become a PRO

(https://programiz.pro/offer/black-november-2025?utm_source=programiz.com&utm_medium=referral&utm_audience=ORGANIC-FREEMIUM&utm_campaign=offer-black-november-sale&utm_content=interests_lifetime&utm_term=sticky_banner_launch)

iz utm_source=programiz.com&utm_medium=referral&utm_audience=ORGANIC-FREEMIUM&utm_campaign=free-to-premium&utm_content=interests_learn_coding&utm_term=nav_floating_button)

(/rust/error-handling)

Programming

Rust unwrap() and expect()

(/rust/unwrap-and-expect)

Programming

Rust Macro

(/rust/macro)

Programming

Rust Enum

(/rust/enum)