Reuse binding in Rust closure

Asked 5 years, 1 month ago
Active 5 years, 1 month ago
Viewed 233 times

1

I am attempting to generate a `Vec<(Point, f64)>` :

```
let grid_size = 5;

let points_in_grid = (0..grid_size).flat_map(|x| {
    (0..grid_size)
        .map(|y| Point::new(f64::from(x), f64::from(y)))
        .collect::<Vec<Point>>()
});

let origin = Point::origin();

let points_and_distances = points_in_grid
    .map(|point| (point, point.distance_to(&origin)))
    .collect::<Vec<(Point, f64)>>();
```

I get the following error:

```
use of moved value: point
```

I understand that I cannot use `point` in both elements of the tuple, but when I attempt to store a reference, I get an error regarding lifetime.

rust closures borrow-checker borrowing

edited Nov 26 '16 at 14:55
Shepmaster
305k • 59 • 824 • 1083

asked Nov 26 '16 at 4:11
davenportw15
167 • 2 • 10

Can you please provide a complete compilable (even with an error) example, preferably which will work on play.rust-lang.org? It would be easier to be sure what error you're getting and where, and to suggest a fix.
– Chris Emerson
Nov 26 '16 at 9:32

1 Answer

1

I am presuming your `Point` struct looks like the following:

```
#[derive(Debug)]
struct Point(f64, f64);

impl Point {
    fn new(x: f64, y: f64) -> Self { Point(x, y) }
    fn origin() -> Self { Point(0.,0.) }
    fn distance_to(&self, other: &Point) -> f64 {
        ((other.0 - self.0).powi(2) + (other.1 - self.1).powi(2)).sqrt()
    }
}
```

Now let's look at an even simpler example that will not compile:

```
let x = Point::new(2.5, 1.0);
let y = x;
let d = x.distance_to(&y);
```

Which gives the error:

```
error[E0382]: use of moved value: `x`
 --> <anon>:15:13
  |
14 |     let y = x;
  |         - value moved here
15 |     let d = x.distance_to(&y);
  |             ^ value used here after move
  |
  = note: move occurs because `x` has type `Point`, which does not implement the `Copy` trait
```

Because `x` has been moved into `y`, it now can't have a reference taken in order to call the `distance_to` function.

The important thing to note here is that order matters - if we swap the lines over we can call `distance_to` by borrowing `x`, the borrow will end and *then* `x` can be moved into `y`.

```
let x = Point(0., 0.);
let d = x.distance_to(&y);
let y = x; // compiles
```

In your case, a very similar thing is happening when constructing the tuple. `point` gets moved into the tuple, and *then* tries to borrow it to form the second element. The simplest solution is to do the same thing as here: swap the order of the elements of the tuple.

```
let points_and_distances = points_in_grid
    .map(|point| (point.distance_to(&origin), point))
    .collect::<Vec<(f64, Point)>>(); // compiles
```

Playground link

N.B. if you want to retain the order:

```
.map(|(a, b)| (b, a))
```

Share
Improve this answer
Follow
edited Nov 26 '16 at 14:55

Shepmaster
305k ● 59 ● 824 ● 1083

answered Nov 26 '16 at 11:13

Djzin
1,028 ● 7 ● 11

1

Said another way, `Point` does not implement Copy. You could also use a temporary variable - `.map(|point| { let d = point.distance_to(&origin); (point, d) })`.
– Shepmaster
Nov 26 '16 at 14:57

Add a comment

Your Answer

Post Your Answer

By clicking "Post Your Answer", you agree to our terms of service, privacy policy and cookie policy

Not the answer you're looking for? Browse other questions tagged rust closures borrow-checker borrowing or ask your own question.

The Overflow Blog

- Sequencing your DNA with a USB dongle and open source code
- Don't push that button: Exploring the software that flies SpaceX rockets and...

Featured on Meta

- Providing a JavaScript API for userscripts

- Congratulations to the 59 sites that just left Beta

Related

What is the difference between a 'closure' and a 'lambda'?

Access to Modified Closure

JavaScript closure inside loops – simple practical example

In PHP, what is a closure and why does it use the "use" identifier?

Why doesn't println! work in Rust unit tests?

How to convert C variable-length array code to Rust?

Cannot move out of borrowed content from closure return value

Why I get "temporary value dropped while borrowed" if I assign, but not when passing via function?

Can I mutate a vector with a borrowed element?

Rust: Cannot reference local variable in return value - but the "local variable" is passed to the caller

Hot Network Questions

What does this entry on the Rocinante's pilot quick-menu mean?

Regular expressions within QGIS expressions: logical operator AND

Question on OEIS A000085

Holding entry is counted as a one turn holding?

Company kept previous personal phone number

more hot questions

Question feed