Rust, make a closure inside a closure avoiding "closure may outlive the current function"

Asked 1 year, 8 months ago

Active 1 year, 8 months ago

Viewed 189 times

1

I'm trying to write a function to transform a data structure in the form:

```
input = [("a", [1,2,3]), ("b", [4,5,6])]
```

Into

```
output = [(a,1), (c,2) ..... (b,6)]
```

My code is currently this:

```
    let foo=vec![('a', vec![1,2,3]), ('v', vec![2,3,4])];
    let baz: Vec<(char,i32)> = foo.into_iter().map(|a|a.1.into_iter().map( |b|(a.0, b))).flatten().collect();
    println!("{:?}",baz);
```

I'm getting this error:

```
error[E0373]: closure may outlive the current function, but it borrows `a`, which is owned by the current function
  --> src/lib.rs:10:76
   |
10 |     let baz: Vec<(char,i32)> = foo.into_iter().map(|a|a.1.into_iter().map( |b|(a.0, b))).flatten().collect();
   |                                                                            ^^^ - `a` is borrowed here
   |                                                                            |
   |                                                                            may outlive borrowed value `a`
   |
note: closure is returned here
  --> src/lib.rs:10:55
   |
10 |     let baz: Vec<(char,i32)> = foo.into_iter().map(|a|a.1.into_iter().map( |b|(a.0, b))).flatten().collect();
   |                                                    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
help: to force the closure to take ownership of `a` (and any other referenced variables), use the `move` keyword
   |
10 |     let baz: Vec<(char,i32)> = foo.into_iter().map(|a|a.1.into_iter().map( move |b|(a.0, b))).flatten().collect();
   |                                                                            ^^^^^^^^

error[E0382]: borrow of moved value: `a`
  --> src/lib.rs:10:76
   |
10 |     let baz: Vec<(char,i32)> = foo.into_iter().map(|a|a.1.into_iter().map( |b|(a.0, b))).flatten().collect();
   |                                                    ---                     ^^^ - borrow occurs due to use in closure
   |                                                    |                       |
   |                                                    value moved here     value borrowed here after partial move
   |
   = note: move occurs because `a.1` has type `std::vec::Vec<i32>`, which does not implement the `Copy` trait
```

I think this means that Rust doesn't know how to copy my vector of i32s, so thinks it must move the vec instead, but cannot do it.

How do I fix this problem? Implement a Copy method for vec, or is there a niftier way to do this?

rust closures borrow

Share

Improve this question

Follow

asked Apr 2 '20 at 23:26

Salim Fadhley

5,167 ● 8 ● 39 ● 63

I don't understand how you get (c,2) from the input. Did you mean to say `output = [(a,1), (a,2) ... (b,6)]` ? And did you mean for the "a" and "b" to be single-quoted, as they are in the code?
– AmigoNico
Apr 3 '20 at 3:09

Add a comment

2 Answers

Active Oldest Votes

4

When you call `a.1.into_iter()`, `a` is moved, and can't be borrowed in the inner closure anymore.

When you call `a.1.into_iter()`, `a` is moved, and can't be borrowed in the inner closure anymore.

The easiest solution is to destructure `a`, so each component can be borrowed/moved individually:

```
.map(|(c, v)| v.into_iter().map(move |b| (c, b)))
```

Also note the `move` keyword, which means that `c` is moved into the inner closure, so it's allowed to outlive the outer closure.

Share
Improve this answer
Follow
answered Apr 3 '20 at 0:13

Aloso
4,380 ● 4 ● 21 ● 36

Add a comment

0

`IntoIterator` consumes and yields values. Since `Vec` doesn't implement `Copy`, when you call `a.1.into_iter()`, it is moved. You can clone it like this: `a.1.clone().into_iter()`

Also, you want to use the `move` keyword to take the ownership of `a` in the closure.

```
let baz: Vec<(char, i32)> = foo
    .into_iter()
    .map(|a| a.1.clone().into_iter().map(move |b| (a.0, b)))
    .flatten()
    .collect();
println!("{:?}", baz);
// [('a', 1), ('a', 2), ('a', 3), ('v', 2), ('v', 3), ('v', 4)]
```

Share
Improve this answer
Follow
answered Apr 2 '20 at 23:52

Kentaro Okuda
1,509 ● 2 ● 10 ● 16

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 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

Is it possible to make a recursive closure in Rust?

Variable binding: moving a &mut or borrowing the referent?

How to capture mutable reference into move closure contained in iterator returned from a closure

"error: closure may outlive the current function" but it will not outlive it

Closure may outlive the current function

What's the real meaning of the error "closure may outlive the current function"?

Rust error[E0373]: closure may outlive the current function, but it borrows `iteration_index`

Cannot move out because of borrowing

Hot Network Questions

- How do I get the http endpoint to work for cardano-wallet?
- Why doesn't dkim sign the letter?
- What was the plutonium for, that was stolen at the start of The Amazing Spider-Man 2?
- Question on connections in general relativity and particle physics
- Without passport stamps, can my country's authorities still know what countries I've visited?

more hot questions

Question feed