Skip to main content

Stack Overflow

1. About
2. Products
3. For Teams

1. Stack Internal Implement a knowledge platform layer to power your enterprise and AI tools.
2. Stack Data Licensing Get access to top-class technical expertise with trusted & attributed content.
3. Stack Ads Connect your brand to the world’s most trusted technologist communities.
4. Releases Keep up-to-date on features we add to Stack Overflow and Stack Internal.
5. About the company Visit the blog

Loading…

current community

Stack Overflow
help chat
- Meta Stack Overflow

your communities

Sign up or log in to customize your list.

more stack exchange communities

company blog

3. Log in
4. Sign up

AI Assist is now on Stack Overflow. Start a chat to get instant answers from across the network. Sign up to save and share your chats.

Home
Questions
AI Assist
Tags
Challenges
7. Chat
Articles
Users
Companies
Collectives

Communities for your favorite technologies. Explore all Collectives

Stack Internal

Stack Overflow for Teams is now called **Stack Internal**. Bring the best of human thought and AI automation together at your work.

Try for free Learn more

Stack Internal

Bring the best of human thought and AI automation together at your work. Learn more

Collectives™ on Stack Overflow

Find centralized, trusted content and collaborate around the technologies you use most.

Learn more about Collectives

Stack Internal

Knowledge at work

Bring the best of human thought and AI automation together at your work.

Explore Stack Internal

Mutable multidimensional array as a function argument

Ask Question

Asked 10 years, 6 months ago

Modified 10 years, 6 months ago

Viewed 3k times

7

In rustc 1.0.0, I'd like to write a function that mutates a two dimensional array supplied by the caller. I was hoping this would work:

```
fn foo(x: &mut [[u8]]) {
   x[0][0] = 42;
}

fn main() {
   let mut x: [[u8; 3]; 3] = [[0; 3]; 3];
   foo(&mut x);
}
```

It fails to compile:

```
$ rustc fail2d.rs
fail2d.rs:7:9: 7:15 error: mismatched types:
expected `&mut [[u8]]`,
   found `&mut [[u8; 3]; 3]`
(expected slice,
   found array of 3 elements) [E0308]
fail2d.rs:7     foo(&mut x);
                   ^~~~~~
error: aborting due to previous error
```

I believe this is telling me I need to somehow feed the function a slice of slices, but I don't know how to construct this.

It "works" if I hard-code the nested array's length in the function signature. This isn't acceptable because I want the function to operate on multidimensional arrays of arbitrary dimension:

```
fn foo(x: &mut [[u8; 3]]) {  // FIXME: don't want to hard code length of nested array
   x[0][0] = 42;
}

fn main() {
   let mut x: [[u8; 3]; 3] = [[0; 3]; 3];
   foo(&mut x);
}
```

tldr; any zero-cost ways of passing a reference to a multidimensional array such that the function use statements like $x[1][2] = 3;$?

- rust

Share

Improve this question

Follow

asked May 21, 2015 at 6:16

romanows

48844 silver badges1212 bronze badges

1

you might be interested in github.com/rust-lang/rfcs/issues/1038 and the corresponding PR about generic value parameters

oli_obk – oli_obk

2015-05-21 14:26:48 +00:00

Commented May 21, 2015 at 14:26

Add a comment |

1 Answer

Sorted by: Reset to default

Highest score (default) Trending (recent votes count more) Date modified (newest first) Date created (oldest first)

7

This comes down to a matter of memory layout. Assuming a type `T` with a size known at compile time (this constraint can be written `T: Sized`), the size of `[T; n]` is known at compile time (it takes `n` times as much memory as `T` does); but `[T]` is an unsized type; its length is not known at compile time. Therefore it can only be used through some form of indirection, such as a reference (`&[T]`) or a box (`Box<[T]>`, though this is of limited practical value, with `Vec<T>` which allows you to add and remove items without needing to reallocate every single time by using overallocation).

A slice of an unsized type doesn't make sense; it's *permitted* for reasons that are not clear to me, but you can never actually have an instance of it. (`Vec<T>`, by comparison, requires `T: Sized`.)

`&[T; n]` can coerce to `&[T]`, and `&mut [T; n]` to `&mut [T]`, but this only applies at the outermost level; the contents of slice is fixed (you'd need to create a new array or vector to achieve such a transformation, because the memory layout of each item is different). The effect of this is that arrays work for single■dimensional work, but for multi■dimensional work they fall apart. Arrays are currently very much second■class citizens in Rust, and will be until the language supports making slices generic over length, which it is likely to eventually.

I recommend that you use either a single■dimensional array (suitable for square matrices, indexed by `x * width + y` or similar), or vectors (`Vec<Vec<T>>`). There may also be libraries already out there abstracting over a suitable solution.

answered May 21, 2015 at 7:00

Chris Morgan

91.6k2828 gold badges217217 silver badges220220 bronze badges

Name

Email

Required, but never shown

Post Your Answer Discard

By clicking “Post Your Answer”, you agree to our terms of service and acknowledge you have read our privacy policy.

Start asking to get answers

Find the answer to your question by asking.

Ask question

Explore related questions

- rust

See similar questions with these tags.

- The Overflow Blog
  Introducing Stack Overflow AI Assist—a tool for the modern developer
  Treating your agents like microservices
- Featured on Meta
  Chat room owners can now establish room guidelines
  AI Assist is now available on Stack Overflow
  Policy: Generative AI (e.g., ChatGPT) is banned

Related

3
Passing two dimensional arrays to functions in Rust
59
How do I pass an array to a function in Rust and change its content?
17
How do I pass a mutable vector as a function parameter in Rust?
1
Rust: passing array recursively
2
Returning an multidimensional array of unknown size from a function in Rust
4
How can you iterate and change the values in a mutable array in Rust?
3
How do I reassign an array through a mutable reference in a function
0
How can I pass a 2D String array (unknown dimensions) as a parameter to a function in Rust
1
Passing arrays by value in Rust
3
Why can't I have mutable values in Rust in array elements?

Hot Network Questions

- Diagonal Arrows Under an Equation
- How do Buddhists interpret the Buddha’s explanation of earthquakes in AN 8.70?
- Why does Mtt 13:34 say that Jesus always spoke in parables?
- Vertical alignment of equations inside tabularx
- Is Brian Ripley’s Pattern Recognition and NN book still an insightful read?
- When did "doubt" meaning question or point of uncertainty fall out of use outside of Indian English?
- Can I Automatically List Supported Kernel Parameters in GRUB?
- Explicit examples separating Wasserstein distances from Jensen–Shannon divergence?

- Looking for a proper formal substitute for "quick fix" for formal letter
- The fourth moment of the Riemann zeta function without absolute values
- A hilarious parody of Western movies where an entire town collapses into tunnels
- Misunderstanding the lorentz transformation
- Why are left and right cosets different in the non-normal case?
- The state of icy worlds during differentiation and accretion, and how they survive with their water
- Is there any point in sharing a scientific discovery as a "nobody"?
- Is there any way to restore hearts without consuming anything, without hot springs, and without shrines?
- Do something and "win"
- Explain this seeming contradiction in Euclid Book 1 Proposition 16
- Why does black allow white to have a protected passed central pawn here?
- Can enemies damage each other in Breath of the Wild?
- When the fuel governor fails, why is autorotation the only option for landing?
- Is there a term for forming emotional or relational bonds with AI chatbots? If not, is “cyberpomorphic” a valid neologism?
- How to protect 100-year-old wooden floors for the next 100 years?
- When should I use tied notes vs. the whole note duration?

more hot questions

Question feed

Subscribe to RSS

Question feed

To subscribe to this RSS feed, copy and paste this URL into your RSS reader.