Kommentare

Das ist ein archivierter Beitrag. Du wirst hier nicht abstimmen oder kommentieren können.

32 Just started learning Rust and was wondering ... does it need to be this verbose? (self.rust)
eingereicht vor 7 Monate * von swiftbeazy
31 Kommentare Weitersagen Speichern Ausblenden Schenke Gold melden crosspost

sortiert nach: beste

Du siehst den Thread eines einzelnen Kommentars.
Übrige Kommentare laden →

[–] fgilcher rust-community · rustfest 82 Punkte vor 7 Monate*

Yes. Rust is, in contrast to Clojure and Haskell, a language that cares about very tight control.

Let's talk about `String` first. Rust has 2 main types of "bunch of characters"-representations. One is the immutable `&str` (often called "string slice"), the other is the *heap-allocated, mutable* `String`. It's not quite true that no other languages have that distinction, just the lingo might be different. If it helps, these types are often compared to `String` and `StringBuffer` in Java, the first one being immutable, the other also a bit verbose and to be explicitely set up :).

That brings me to the *why* of `String::from(...)`: it represents a heap allocation and as such should be visible in Rust. There are no literals in Rust that directly force a value on the heap, strings are no exception, so this needs to happen. Indeed, from Rusts level of working, Strings are *not simple, nor fundamental*. We tend to use them in tutorials through, as heap allocated data is easier to work with in the beginning.

There is a shorter way to writing this, which is `"some string".into()`, which uses type inference to figure out that a `String` is necessary here, but that is avoided in beginner tutorials, as it requires understanding from a few chapters later.

It's rarely a concern in actual code though, when do you ever allocate a whole vector of strings by hand?

Coming to iterators: there's three classes of iterators in Rust - non-mutating (`iter`), mutating (`iter_mut`) and consuming (`into_iter`). On top of that, they are external. The compiler can't easily tell how you want to iterate, so one explicit call to `.iter()` is necessary. In this case, one of the `.iter()` calls can be safely dropped.

```
let scores: HashMap<_, _> =
    teams.iter().zip(initial_scores).collect();
```

(`zip` knows about the chosen iteration already and can do the conversion itself).

The `collect` call is necessary as something needs to kick the iteration off - Rust Iterators are lazy. Other options would the `for` loop or just calling `.next()` many times :).

Clojure and Haskell are problematic in comparison there: Clojure does, AFAIK, not do lazy iteration and Haskell, on the other side, is all built around resolving lazy statements automatically.

By the way, your second example is not the same as the previous.

Finally, I have to say that Rust is a language built for larger pieces of software, and tiny examples are rarely pretty. Rust can often be very on-point, if still detailed, for example in this piece of HTTP server code:

https://gist.github.com/mjohnsullivan/e5182707caf0a9dbdf2d#file-http_server-rs-L33-L49

Permalink embed Speichern Schenke Gold

[–] hoocop 30 Punkte vor 7 Monate

> Rust is a language built for larger pieces of software, and tiny examples are rarely pretty

^ This!

If Rust was so patronizing that it made all those choices for you that would make your example work, it would be more like Haskell and it would leave (language design) space between Rust and asm. And then people would complain that Strings can only be on the heap and would design a language where that isn't the case...

So Rust has it's raison d'être.

Rust is the language that tries to leave no language design space between it and asm while being as high-level and ergonomic as possible, so that it works for systems programming as well as all other kinds of applications.

Permalink embed Speichern Übergeordnet Schenke Gold

[–] fasquoika 13 Punkte vor 7 Monate

> Strings are *not simple, nor fundamental.*

I think this is a lot of the real takeaway. The more you know about strings the more you realize that there is an incredible amount of inherent complexity there. Most languages try to hide the complexity so they can Just Work™ about 95% of the time. Rust does not

Permalink embed Speichern Übergeordnet Schenke Gold

[–] garagedragon 8 Punkte vor 7 Monate

> Coming to iterators: there's three classes of iterators in Rust - non-mutating (iter), mutating (iter_mut) and consuming (iter).

I assume one of those is a type for `into_iter`?

Permalink embed Speichern Übergeordnet Schenke Gold

[–] Cobrand rust-sdf2 3 Punkte vor 7 Monate

That last one is the `into_iter`. It's pretty straighforward to remember, say you have a Vec, you want to transform it *into* an interator, discarding the Vec altogether.

Permalink embed Speichern Übergeordnet Schenke Gold

[–] fgilcher rust-community · rustfest 1 Punkt vor 7 Monate

Yes! Consuming is `into_iter`. Thanks for catching.

Permalink embed Speichern Übergeordnet Schenke Gold

[–] im-a-koala 6 Punkte vor 7 Monate

It's also worth mentioning that if you specify the actual type of the HashMap (required in languages like Java), you can shorten/simplify the code:

```
let mut scores: HashMap<String, i32> = HashMap::new();
scores.insert("Blue".into(), 10);
scores.insert("Yellow".into(), 50);
```

Permalink embed Speichern Übergeordnet Schenke Gold

[–] Sharlinator 4 Punkte vor 7 Monate

Just a small correction: like Rust's, Clojure sequence operations *are* lazy and if you want to "materialize" a lazy seq you need to call the appropriate constructor such as `vec`. Doing that is rarely necessary though as typically sequences are operated in a fire-and-forget manner.

Permalink embed Speichern Übergeordnet Schenke Gold

[–] fgilcher rust-community · rustfest 1 Punkt vor 7 Monate

Ah, thanks for the correction!

Permalink embed Speichern Übergeordnet Schenke Gold

[–] Treyzania 1 Punkt vor 7 Monate

Basically OP, if you want a less-verbose language that compiles to safe native code with extremely easy C interop and has a *relatively*-nice type system, look at D. It doesn't have linear types and uses a GC, but besides that it fits in a lot of the same places that Java does.

Permalink embed Speichern Übergeordnet Schenke Gold

Suchen

Dieser Beitrag wurde eingereicht am 03 Nov 2017
32 Punkte (92 % gefällt das)
shortlink: https://redd.it/7amv58

submit a link

submit a text post

abbestellen

39.049 Rustaceans
702 Mitglieder anwesend

Set your user flair
kellerkindt

The Rust Programming Language

A place for all things related to the Rust programming language, an open-source systems programming language that emphasizes zero-overhead memory safety, fearless concurrency, and blazing speed.

Rules

+ Respect our Code of Conduct.
+ Constructive comments only.
+ No memes.
+ No zealotry.
+ Submissions must be on-topic.
+ Chill out!
+ Be Safe

Resources

+ Learn
+ Install
+ Discuss
+ Contribute

Nachricht an die Moderatoren schicken

MODERATOREN
azth
kitwen
dbaupp · rust
flaper87 · rust
mrmonday · libpnet · rust
howellnick · rust
wrongerontheinternet
Manishearth · servo · rust · clippy
sho_kde
llogiq
clippy · twir · rust · mutagen · flamer · overflower · bytecount
...und 3 mehr »

KÜRZLICH ANGESEHENE LINKS
Ignition, another classic game re-released on GOG with Linux support
36 Punkte | 23 Kommentare
VP9 encoding/decoding performance vs. HEVC/H.264
314 Punkte | 42 Kommentare
Video games website Steam fined $3 million for refusing refunds
23.0k Punkte | 3141 Kommentare
This might just be the worst I've ever seen
4707 Punkte | 317 Kommentare
My triumph in Rocket League and why you need to own it on Linux
4 Punkte | 6 Kommentare
Löschen

Account-Aktivität

Das Benutzen dieser Website begründet sich auf der Verpflichtung unserer Endbenutzervereinbarung und Datenschutzerklärung. © 2018 reddit inc. Alle Rechte vorbehalten.
REDDIT and the ALIEN Logo are registered trademarks of reddit inc.