Who is more foolish?

Novice traps in Rust

Michael Spiegel

September 29, 2016

When I first started learning Rust I started implementing different search trees. Here are some of the pitfalls I ran into. Hopefully my experience will help you.

`https://www.youtube.com/watch?v=x0ow4X8tiMI`

In the Beginning: Pain

```
struct Node {
  key: i32,
  val: i32,
  color: Color,
  left: Option<Box<Node>>,
  right: Option<Box<Node>>,
}
```

1st Attempt: Type Alias

The type keyword lets you declare an alias of another type. The alias does not declare a new type.

```
type Link = Option<Box<Node>>;

impl fmt::Display for Link {
  ...
}
```

1st Attempt: Type Alias

The type keyword lets you declare an alias of another type. The alias does not declare a new type.

```
type Link = Option<Box<Node>>;

impl fmt::Display for Link {
  ...
}
```

```
error: the impl does not reference any types defined in this crate;
only traits defined in the current crate can be implemented for
arbitrary types [E0117]
```

2nd Attempt: Newtype Pattern

The tuple struct is a hybrid between a tuple and a struct. A 1-dimensional tuple struct is called the 'newtype' pattern.

```
struct Link(Option<Box<Node>>);
```

2nd Attempt: Newtype Pattern

The tuple struct is a hybrid between a tuple and a struct. A 1-dimensional tuple struct is called the 'newtype' pattern.

```
struct Link(Option<Box<Node>>);
```

The ownership semantics are 2x as complicated. Am I borrowing the tuple or the value inside the tuple? Reddit thread just on this topic https://www.reddit.com/r/rust/comments/2cmjfn/how_to_do_typedefsnewtypes

3rd Attempt: Wrapper struct

The dumbest solution is to wrap the type inside a struct.

```
struct Link {
  link: Option<Box<Node>>,
}
```

3rd Attempt: Wrapper struct

The dumbest solution is to wrap the type inside a struct.

```
struct Link {
  link: Option<Box<Node>>,
}
```

This works but the code is cluttered with ".link" everywhere. The code is harder to understand.

Reboot

So what is the idiomatic way to express this type?

Vanilla Rust

```
struct Node {
  key: i32,
  val: i32,
  color: Color,
  left: Option<Box<Node>>,
  right: Option<Box<Node>>,
}
```

Use Traits

```
trait OptionBoxNode {
  fn get(&self, key: i32) -> Option<i32>;
  fn is_red(&self) -> bool;
  fn color(&self) -> Color;
}

impl OptionBoxNode for Option<Box<Node>> {
  ...
}
```

Escape Hatch

Newtype when necessary.

```
struct Link<'a>(&'a Option<Box<Node>>)

impl<'a> fmt::Display for Link<'a> {
  ...
}
```

Pro Tips

The following idioms are helpful...

Multiple impl sections

Declare impls for both the definite type and the optional type. Write functions in the appropriate section depending on whether it manipulates a node or a “nullable” node.

```
impl Node {
    ...
}

impl OptionBoxNode for Option<Box<Node>> {
    ...
}
```

Helper functions everywhere

```
impl OptionBoxNode for Option<Box<Node>> {

fn reference(&self) -> &Box<Node>
  {  self.as_ref().unwrap()  }

fn mutate(&mut self) -> &mut Box<Node>
  {  self.as_mut().unwrap() }

fn left(&self) -> &Option<Box<Node>>
  { &self.as_ref().unwrap().left }

}
```

Beware the let ref

4 ways to declare a variable.

- `let x : i32`
- `let mut x : i32`
- `let ref x : i32`
- `let mut ref x : i32`

Beware the let ref

2 sane ways to declare a variable.

- `let x : i32;`
- `let mut x : i32;`
- `let ref x : i32;`
- `let mut ref x : i32;`

- `let x : i32;`
- `let mut x : i32;`
- `let x : &i32;`
- `let mut x : &i32;`

Bonus content: RustConf 2016

RustConf was on September 10 in Portland OR.

- Slides and exercises from the tutorials `http://rust-tutorials.com/RustConf16`
- Opening keynote. “Fast, reliable, productive” – Pick three
- Josh Triplett on the RFC process
- If you use Rust commercially, community-team@rust-lang.org wants your feedback

Rustbelt Rust coming up October 27-28 in Pittsburg PA.

Putting everything together

- https://github.com/mspiegel/rust-yaar
- https://github.com/rust-lang-nursery/rustup.rs
- https://github.com/rust-lang-nursery/rustfmt
- https://github.com/Manishearth/rust-clippy