Rust Programming

Henry Oehlrich

April 22, 2023

Roadmap

- Introduction
- Memory, the stack, and the heap
- Variable lifetimes and scope
- Borrowing
- Safety

Rust Concepts

- Enums
- Structs
- Traits
- Lifetimes
- Generics
- Primitives
- **References and borrowing**

Memory

Memory is temporary storage of program data at execution

```
fn main() {
    let x: i32 = 10;
    let s1: &str = "I’m a string literal";
}
```

The Stack

- Fast way to store and retrieve data
- Last in first out
- Must know the size of the data

https://thumbs.dreamstime.com/z/stack-clothes-isolated-white-background-stack-clothes-isolated-152319600.jpg

The Heap

- Slower to store and retrieve data
- Need not know the size of the data
- Able to resize, copy, and clone on the fly

https://img.thrfun.com/img/175/807/clothes_wrinkled_x2.jpg

Memory

address	value
...	...
0x7fffac86e908	00000000
0x7fffac86e909	00000000
0x7fffac86e90a	00000111
0x7fffac86e90b	11100111
...	...

```
let x: u32 = 2023;
println!("x addr: {:p}", &x);
// x addr: 0x7fffac86e908
```

Variable scope

```rust
fn main() {
    {
        let x: i32 = 5;
        println!("x = {}", x);
    }
    println!("x = {}", x);
}
```

```
error[E0425]: cannot find value `x` in this scope
 --> src/main.rs:6:24
  |
6 |     println!("x = {}", x);
  |                        ^ not found in this scope
```

The Borrow Checker

1. Data has one owner
2. Data can have multiple readers, or one writer

Passing by Value

```
fn main() {
    let x: i32 = 15;
    printx(x);  // x’s value moved into printx
    printx(x);  // x no longer valid
}

fn printx(x: i32) {
    println!("x = {}", x);
}
```

References

- A reference is the memory address of a value's first byte
- Also known as pointers because they *point* to where the data is located

Borrowing

```
fn main() {
    let ltuae: u32 = 42;
    let ltuaep: &u32 = &ltuae;
}
```

- Borrow a value by prefixing it with *&*
- The resulting type is of *&type*

Dereferencing

```
fn main() {
    let x: i32 = 5;
    let xp: &i32 = &x;
    println!("x = {}", *xp);
    println!("x = {}", xp);
}
```

- References can be dereferenced to retrieve the original value by using *
- Dereferences are usually implicit

Passing By Reference

```
fn main() {
    let x: i32 = 15;
    printx(&x);
    printx(&x);
}

fn printx(x: &i32) {
    println!("x = {}", *x);
}
```

Mutable references

- Variables and references are immutable by default
- Mutable variables and references are defined with *mut*

```
let mut x: i32 = 5;
let xp: &mut i32 = &mut x;
*xp += 1;
```

Rust
I'll be designated driver tonight!
we know
Don't talk while eating you could choke and die haha

https://leftoversalad.com/c/015_programmingpeople/

Thank You