Rust

Adam Boult (www.bou.lt)

August 25, 2025

Contents

Preface **2**

I Rust **3**

1 Rust literals and variables **4**

2 Rust control flow **6**

3 Rust macros and functions **7**

4 Rust arrays and tuples **8**

5 Rust standard library **9**

6 Cargo **10**

II SORT **11**

7 SORT 2025 **12**

Preface

This is a live document, and is full of gaps, mistakes, typos etc.

Part I

Rust

Chapter 1

Rust literals and variables

1.1 Literals

1.1.1 Integer literals

Decimal literals:

```
12345
12_345
012345
```

Hex, oct and bin literals

```
0xff
0o77
0b110
```

1.1.2 Integer literals in a given format

can do

```
i8, i16, i32, i64, i128
u8, u16, u32, u64, u128
```

```
5i32
0b101i32
```

1.1.3 Assignment and type notations

1.1.4 Type annotations

```
let x:u32 = 2
```

Can split out declaration and definition.

```
let x:u32
x = 2
```

1.1.5 Mutable variables

Can't do this because default immutableimmutable.

```
let x = 1
x = x + 1
```

can do

```
let x = 1
let x = x + 1
```

this is shadowing the variable?

We can make variables mutable.

```
let mut x = 1
x = x + 1
```

1.1.6 Const

Different to immutable because const are known at compile-time, whereas immutables may not be known until run time.

```
const x = 1
```

1.1.7 String literals

```
byte (u8?): b'A'
```

1.1.8 Float literals

```
f32, f64
```

Chapter 2

Rust control flow

2.1 Rust control flow

2.1.1 If statements

if statement else if else

2.1.2 For loops

2.1.3 While loops

2.1.4 Loop

loop break

2.1.5 Match

Like switch in C.

Chapter 3

Rust macros and functions

3.1 Rust functions

3.1.1 Macros

macros end in "!" eg println!

```
macro_rules! say_hello {
    // `()` indicates that the macro takes no argument.
    () => {
        // The macro will expand into the contents of this block.
        println!("Hello!");
    };
}

fn main() {
    // This call will expand into `println!("Hello");`
    say_hello!()
}
```

3.1.2 Functions

```
fn main() {}
fn my_function(x: i32) -> i32 {}
```

Chapter 4

Rust arrays and tuples

4.1 Arrays

4.1.1 Arrays

arrays: like tuples but must be same type let a:[i32,3] = [1,2,3] [3;5] == [3,3,3,3,3] a[0] = 1

4.2 Tuples

4.2.1 Tuples

tuples

let tup: (i32, f64, u8) = (500, 6.4, 1)

let (x, y, z) = tup tup.0

Chapter 5

Rust standard library

5.1 Rust standard library

5.1.1 Rust standard library

Chapter 6

Cargo

6.1 Cargo

6.1.1 Cargo

Built tool and packagae manager.

cargo new cargo build

cargo build –release + makes faster but takes longer to compile. goes to different folder.

toml files cargon run to compile then run.

Part II

SORT

Chapter 7

SORT 2025

7.1 Introduction

7.1.1 Rust

rust has core, alloc and std instead of libc

rust: + passing arrays to functions. what is passed? reference ot first? length separately? + repeating an array n times

linked lists/arrays in rust + concatenation + slice/filter + insert + pop + traverse + map + sort + reverse

ranges in rust. can use ”..”

impl trait

type

structs (access using ::) tuple structs enum

pub keyword (makes thing visible to other files?)

rust generics rust functional, inc lambda. partial application. currying rust oop h3 rust parallel h3