

Table of Contents

Preface.....	xv
1. Systems Programmers Can Have Nice Things.....	1
Rust Shoulders the Load for You	2
Parallel Programming Is Tamed	3
And Yet Rust Is Still Fast	4
Rust Makes Collaboration Easier	4
2. A Tour of Rust.....	5
rustup and Cargo	6
Rust Functions	8
Writing and Running Unit Tests	10
Handling Command-Line Arguments	11
Serving Pages to the Web	15
Concurrency	22
What the Mandelbrot Set Actually Is	23
Parsing Pair Command-Line Arguments	28
Mapping from Pixels to Complex Numbers	30
Plotting the Set	32
Writing Image Files	33
A Concurrent Mandelbrot Program	35
Running the Mandelbrot Plotter	40
Safety Is Invisible	41
Filesystems and Command-Line Tools	42
The Command-Line Interface	43
Reading and Writing Files	45
Find and Replace	46

3. Fundamental Types.....	49
Fixed-Width Numeric Types	52
Integer Types	53
Checked, Wrapping, Saturating, and Overflowing Arithmetic	56
Floating-Point Types	58
The bool Type	61
Characters	61
Tuples	63
Pointer Types	65
References	65
Boxes	66
Raw Pointers	66
Arrays, Vectors, and Slices	67
Arrays	67
Vectors	68
Slices	71
String Types	73
String Literals	73
Byte Strings	74
Strings in Memory	74
String	76
Using Strings	77
Other String-Like Types	77
Type Aliases	78
Beyond the Basics	78
4. Ownership and Moves.....	79
Ownership	81
Moves	85
More Operations That Move	90
Moves and Control Flow	91
Moves and Indexed Content	92
Copy Types: The Exception to Moves	94
Rc and Arc: Shared Ownership	98
5. References.....	101
References to Values	102
Working with References	105
Rust References Versus C++ References	105
Assigning References	107
References to References	107
Comparing References	108

References Are Never Null	109
Borrowing References to Arbitrary Expressions	109
References to Slices and Trait Objects	110
Reference Safety	110
Borrowing a Local Variable	110
Receiving References as Function Arguments	113
Passing References to Functions	115
Returning References	116
Structs Containing References	117
Distinct Lifetime Parameters	120
Omitting Lifetime Parameters	121
Sharing Versus Mutation	123
Taking Arms Against a Sea of Objects	130
6. Expressions.....	133
An Expression Language	133
Precedence and Associativity	134
Blocks and Semicolons	137
Declarations	138
if and match	140
if let	142
Loops	142
Control Flow in Loops	144
return Expressions	145
Why Rust Has loop	146
Function and Method Calls	148
Fields and Elements	149
Reference Operators	151
Arithmetic, Bitwise, Comparison, and Logical Operators	151
Assignment	152
Type Casts	153
Closures	154
Onward	154
7. Error Handling.....	157
Panic	157
Unwinding	158
Aborting	159
Result	160
Catching Errors	160
Result Type Aliases	162
Printing Errors	163

Propagating Errors	164
Working with Multiple Error Types	166
Dealing with Errors That “Can’t Happen”	168
Ignoring Errors	169
Handling Errors in main()	169
Declaring a Custom Error Type	171
Why Results?	172
8. Crates and Modules	173
Crates	173
Editions	176
Build Profiles	177
Modules	178
Nested Modules	179
Modules in Separate Files	180
Paths and Imports	183
The Standard Prelude	186
Making use Declarations pub	186
Making Struct Fields pub	186
Statics and Constants	187
Turning a Program into a Library	188
The src/bin Directory	189
Attributes	191
Tests and Documentation	193
Integration Tests	196
Documentation	197
Doc-Tests	199
Specifying Dependencies	202
Versions	202
Cargo.lock	204
Publishing Crates to crates.io	205
Workspaces	207
More Nice Things	208
9. Structs	209
Named-Field Structs	209
Tuple-Like Structs	212
Unit-Like Structs	213
Struct Layout	213
Defining Methods with impl	214
Passing Self as a Box, Rc, or Arc	217
Type-Associated Functions	219

Associated Consts	220
Generic Structs	221
Structs with Lifetime Parameters	223
Deriving Common Traits for Struct Types	224
Interior Mutability	225
10. Enums and Patterns.	229
Enums	230
Enums with Data	232
Enums in Memory	233
Rich Data Structures Using Enums	234
Generic Enums	236
Patterns	239
Literals, Variables, and Wildcards in Patterns	242
Tuple and Struct Patterns	243
Array and Slice Patterns	244
Reference Patterns	245
Match Guards	247
Matching Multiple Possibilities	248
Binding with @ Patterns	248
Where Patterns Are Allowed	249
Populating a Binary Tree	250
The Big Picture	252
11. Traits and Generics.	253
Using Traits	255
Trait Objects	256
Generic Functions and Type Parameters	258
Which to Use	261
Defining and Implementing Traits	264
Default Methods	265
Traits and Other People's Types	266
Self in Traits	268
Subtraits	269
Type-Associated Functions	270
Fully Qualified Method Calls	271
Traits That Define Relationships Between Types	273
Associated Types (or How Iterators Work)	273
Generic Traits (or How Operator Overloading Works)	277
impl Trait	278
Associated Consts	280
Reverse-Engineering Bounds	281

Traits as a Foundation	284
12. Operator Overloading.....	285
Arithmetic and Bitwise Operators	286
Unary Operators	289
Binary Operators	290
Compound Assignment Operators	291
Equivalence Comparisons	292
Ordered Comparisons	295
Index and IndexMut	298
Other Operators	300
13. Utility Traits.....	301
Drop	302
Sized	305
Clone	308
Copy	309
Deref and DerefMut	310
Default	313
AsRef and AsMut	315
Borrow and BorrowMut	316
From and Into	318
TryFrom and TryInto	321
ToOwned	322
Borrow and ToOwned at Work: The Humble Cow	323
14. Closures.....	325
Capturing Variables	327
Closures That Borrow	328
Closures That Steal	328
Function and Closure Types	330
Closure Performance	332
Closures and Safety	333
Closures That Kill	334
FnOnce	334
FnMut	336
Copy and Clone for Closures	338
Callbacks	339
Using Closures Effectively	343
15. Iterators.....	345
The Iterator and IntoIterator Traits	347

Creating Iterators	348
iter and iter_mut Methods	348
IntoIterator Implementations	349
from_fn and successors	351
drain Methods	353
Other Iterator Sources	354
Iterator Adapters	356
map and filter	356
filter_map and flat_map	359
flatten	361
take and take_while	363
skip and skip_while	363
peekable	364
fuse	365
Reversible Iterators and rev	366
inspect	367
chain	368
enumerate	369
zip	370
by_ref	370
cloned, copied	371
cycle	372
Consuming Iterators	373
Simple Accumulation: count, sum, product	373
max, min	374
max_by, min_by	374
max_by_key, min_by_key	375
Comparing Item Sequences	375
any and all	376
position, rposition, and ExactSizeIterator	376
fold and rfold	377
try_fold and try_rfold	378
nth, nth_back	379
last	380
find, rfind, and find_map	380
Building Collections: collect and FromIterator	381
The Extend Trait	383
partition	383
for_each and try_for_each	384
Implementing Your Own Iterators	385

16. Collections.....	391
Overview	392
Vec<T>	393
Accessing Elements	394
Iteration	396
Growing and Shrinking Vectors	396
Joining	400
Splitting	400
Swapping	403
Sorting and Searching	404
Comparing Slices	406
Random Elements	406
Rust Rules Out Invalidation Errors	407
VecDeque<T>	408
BinaryHeap<T>	410
HashMap<K, V> and BTreeMap<K, V>	411
Entries	415
Map Iteration	417
HashSet<T> and BTreeSet<T>	418
Set Iteration	419
When Equal Values Are Different	420
Whole-Set Operations	420
Hashing	422
Using a Custom Hashing Algorithm	423
Beyond the Standard Collections	424
17. Strings and Text.....	427
Some Unicode Background	428
ASCII, Latin-1, and Unicode	428
UTF-8	428
Text Directionality	430
Characters (char)	430
Classifying Characters	431
Handling Digits	432
Case Conversion for Characters	433
Conversions to and from Integers	434
String and str	434
Creating String Values	435
Simple Inspection	436
Appending and Inserting Text	437
Removing and Replacing Text	438
Conventions for Searching and Iterating	439

Patterns for Searching Text	440
Searching and Replacing	441
Iterating over Text	442
Trimming	444
Case Conversion for Strings	445
Parsing Other Types from Strings	445
Converting Other Types to Strings	446
Borrowing as Other Text-Like Types	447
Accessing Text as UTF-8	447
Producing Text from UTF-8 Data	448
Putting Off Allocation	449
Strings as Generic Collections	451
Formatting Values	451
Formatting Text Values	453
Formatting Numbers	454
Formatting Other Types	456
Formatting Values for Debugging	456
Formatting Pointers for Debugging	458
Referring to Arguments by Index or Name	458
Dynamic Widths and Precisions	459
Formatting Your Own Types	460
Using the Formatting Language in Your Own Code	462
Regular Expressions	463
Basic Regex Use	464
Building Regex Values Lazily	465
Normalization	466
Normalization Forms	467
The unicode-normalization Crate	468
18. Input and Output	471
Readers and Writers	472
Readers	473
Buffered Readers	475
Reading Lines	477
Collecting Lines	479
Writers	480
Files	481
Seeking	482
Other Reader and Writer Types	483
Binary Data, Compression, and Serialization	485
Files and Directories	486
OsStr and Path	486

Path and PathBuf Methods	488
Filesystem Access Functions	490
Reading Directories	491
Platform-Specific Features	493
Networking	494
19. Concurrency.....	497
Fork-Join Parallelism	499
spawn and join	500
Error Handling Across Threads	502
Sharing Immutable Data Across Threads	503
Rayon	505
Revisiting the Mandelbrot Set	508
Channels	510
Sending Values	511
Receiving Values	514
Running the Pipeline	515
Channel Features and Performance	517
Thread Safety: Send and Sync	519
Piping Almost Any Iterator to a Channel	521
Beyond Pipelines	523
Shared Mutable State	523
What Is a Mutex?	524
Mutex<T>	526
mut and Mutex	527
Why Mutexes Are Not Always a Good Idea	528
Deadlock	529
Poisoned Mutexes	529
Multiconsumer Channels Using Mutexes	530
Read/Write Locks (RwLock<T>)	531
Condition Variables (Condvar)	532
Atomics	533
Global Variables	535
What Hacking Concurrent Code in Rust Is Like	538
20. Asynchronous Programming.....	539
From Synchronous to Asynchronous	541
Futures	542
Async Functions and Await Expressions	544
Calling Async Functions from Synchronous Code: block_on	547
Spawning Async Tasks	550
Async Blocks	554

Building Async Functions from Async Blocks	556
Spawning Async Tasks on a Thread Pool	557
But Does Your Future Implement Send?	558
Long Running Computations: <code>yield_now</code> and <code>spawn_blocking</code>	561
Comparing Asynchronous Designs	562
A Real Asynchronous HTTP Client	563
An Asynchronous Client and Server	564
Error and Result Types	566
The Protocol	567
Taking User Input: Asynchronous Streams	568
Sending Packets	570
Receiving Packets: More Asynchronous Streams	571
The Client's Main Function	573
The Server's Main Function	574
Handling Chat Connections: Async Mutexes	575
The Group Table: Synchronous Mutexes	578
Chat Groups: tokio's Broadcast Channels	579
Primitive Futures and Executors: When Is a Future Worth Polling Again?	582
Invoking Wakers: <code>spawn_blocking</code>	584
Implementing <code>block_on</code>	586
Pinning	588
The Two Life Stages of a Future	588
Pinned Pointers	592
The Unpin Trait	594
When Is Asynchronous Code Helpful?	595
21. Macros.....	599
Macro Basics	600
Basics of Macro Expansion	601
Unintended Consequences	603
Repetition	605
Built-In Macros	607
Debugging Macros	609
Building the <code>json!</code> Macro	610
Fragment Types	611
Recursion in Macros	615
Using Traits with Macros	615
Scoping and Hygiene	617
Importing and Exporting Macros	620
Avoiding Syntax Errors During Matching	622
Beyond <code>macro_rules!</code>	623

22. Unsafe Code.....	625
Unsafe from What?	626
Unsafe Blocks	628
Example: An Efficient ASCII String Type	629
Unsafe Functions	631
Unsafe Block or Unsafe Function?	633
Undefined Behavior	634
Unsafe Traits	636
Raw Pointers	638
Dereferencing Raw Pointers Safely	641
Example: RefWithFlag	642
Nullable Pointers	644
Type Sizes and Alignments	645
Pointer Arithmetic	645
Moving into and out of Memory	647
Example: GapBuffer	651
Panic Safety in Unsafe Code	657
Reinterpreting Memory with Unions	658
Matching Unions	661
Borrowing Unions	661
23. Foreign Functions.....	663
Finding Common Data Representations	664
Declaring Foreign Functions and Variables	668
Using Functions from Libraries	669
A Raw Interface to libgit2	673
A Safe Interface to libgit2	679
Conclusion	690
Index.....	691