Symbols & Numbers

& (ampersand), 492

to create a reference, 17

<> (angle brackets), 494–95

for specifying type parameters, 130, 172, 173

for specifying lifetime parameters, 191

-> (arrow), 46–47, 492

\* (asterisk), 492

dereference operator, 68, 311–15, 416

glob operator, 124–25

multiplication operator, 38

\*const T, 415–17, 492

\*mut T, 415–17, 492

@ (at operator), 410–11, 493

: (colon), 492–93, 495

for trait bounds, 183

for struct fields, 82

{} (curly brackets), 496

scope creation, 45, 71

for function bodies, 6, 15

as placeholders in the println! macro, 18

. (dot), 492

for method syntax, 91

for struct field access, 82

for tuple element access, 40

:: (double colon), 494

for associated functions, 94

for enum variants, 96

for namespacing, 111

" (double quote), 39, 493

/ (forward slash), 492

for division, 38

- (hyphen), 492

for subtraction, 38

! (never type), 443–44, 494

() (parentheses), 496

for function parameters, 6, 15

for tuples, 39–40

% (percent), 492

for remainder, 38

+ (plus), 492

for addition of number types, 38

for addition of strings, 137–38

for specifying multiple trait bounds, 183

? (question mark operator), 159–61, 493

?Sized, 446

; (semicolon), 6, 493

' (single quote), 493–94

for characters, 39

for lifetime parameter names, 190

'static lifetime, 198–99, 421, 429, 430, 465

[] (square brackets), 496

for array creation, 41

for element access, 41, 131–32

\_ (underscore), 494

as a catchall pattern, 28, 106–7, 403–5

as a visual separator in integer literals, 37

| (vertical pipe), 493–94

in closure definitions, 261

in patterns, 398

1:1 threading model, 343

A

ABI (application binary interface), 420

abort, 150

addition

of custom types, 432–34

of number types, 38

of strings, 137–38

ahead-of-time compiled, 7

ampersand (&), 492

to create a reference, 17

angle brackets (<>), 494–95

for specifying type parameters, 130, 172, 173

for specifying lifetime parameters, 191

API (Application Programming Interface), iv, 4

application binary interface (ABI), 420

Application Programming Interface (API), iv, 4

Arc<T> type, 361–62, 473–74

arguments, 43

arms

in if expressions, 49

in match expressions, 24, 103

array data type, 40–42

invalid element access, 41–42

iterating over elements of, 54–55

slices of, 78–79

arrow (->), 46–47, 492

as\_bytes method, 73–74

assert\_eq! macro, 208–10

assert! macro, 205–8

assert\_ne! macro, 210

associated function, 16, 93–94

associated types, 431–32

associative array. See HashMap<K, V> type

asterisk (\*), 492

dereference operator, 68, 311–15, 416

glob operator, 124–25

multiplication operator, 38

at operator (@), 410–11, 493

atomically reference counted, 361–62

automatic dereferencing, 92

automatic referencing, 92

B

backtrace, 151–53

binary crate, 8, 19

splitting into a library crate, 233

binary target, 302

blanket implementations, 186

blocking, 345

Boolean data type, 39

in if expressions, 50

borrow checker, 188–89, 190

borrowing

and references, 68–73

Box<T> type, 306–11

break keyword, 27–28

buffer overread, 151

Build Tools for Visual Studio, 3

byte literal syntax, 37, 74

C

Cargo, ii, 7–11

commands

build, 9

check, 10

doc, 22, 287–88

install, 302–3

login, 294

new, 8, 14

publish, 294–96

run, 10, 299

test, 202–5, 215–20, 289, 301–2

update, 21

yank, 296

extending with custom commands, 303

workspaces, 297–302

Cargo.toml

dependencies section in, 19

package section in, 8, 294–95

profile section in, 286–87

updating crate versions in, 21

Cargo.lock, 9, 20–21

carriage return, 454

cfg (configuration) attribute, 221

channels, 349–55, 470–74

character data type, 39

client, 450

clone method

deep copy creation, 65

trade-offs of, 236

Clone trait, 499–500

closed channel, 350

closures, 258–70

capturing the environment with, 268–70, 274–75

returning, 448

running in threads, 344

type inference in, 263–64

cmp method, 23

coherence, 180

collections, 129–47

collect method, 143, 229

colon (:), 492–93

for trait bounds, 183

for struct fields, 82

command line arguments, accepting, 228–31

command line notation, 2

comments, 48

documentation, 287–90

compiler-driven development, 462

compiling

in release mode, 10–11

with rustc, 6–7

with cargo, 7–11

compound data types, 39–42

concurrency, 341–64

concurrent programming, 341

configuration (cfg) attribute, 221

connection, 451–54

cons list, 308–11

constants, 34

vs. variables, 34

vs. static variables, 421–22

constructor, 319

consume, 272–73

consuming adaptors, 272–73

continue keyword, 28–29

contracts, 163

control flow, 48–55

Copy trait, 5, 499–500

ctrl-c, 27, 53, 452, 479

crate, 9

binary vs. library, 8, 19

license of, 295

publishing, 294–96

publishing a new version of, 296

updating versions, 21

using as a dependency, 21–22

yanking, 296

crates.io, 287–96

setting up an account on, 294

CRLF sequence, 454

curly brackets ({}), 496

creating new scopes with, 45, 71

for function bodies, 6, 15

as placeholders in the println! macro, 18

D

dangling pointer, 72

dangling reference, 72–73, 187–89, 193–94

data race, 70–71, 422

data types, 36–42

annotation of, 25, 36

compound, 39–42

scalar, 36–39

deadlock, 343, 362, 484

Debug trait, 89–90, 498

declarative macros, 502–4

deep copy, 64–65

Default trait, 500

default type parameters, 432–34

dependency, 7, 19

dependencies

section in Cargo.toml, 9, 19

deref coercion, 138, 315–17

Deref trait, 311–17, 440

DerefMut trait, 316–17

derive annotation, 88–90, 497–500

destructor, 319

destructuring

of enums, 400–401

of references, 402

of structs, 399–400

of tuples, 40

Dickinson, Emily, 231

Dijkstra, Edsger W., 201

Display trait, 89

diverging functions, 443

division, 38

doc tests, 289

documentation

offline for Rust, 4

tests, 289

viewing a crate’s, 22

writing, 287–90

documentation comments, 287–90, 467

as tests, 289

dot (.), 492

for method syntax, 91

for struct field access, 82

for tuple element access, 40

double colon (::), 494

for associated functions, 94

for enum variants, 96

for namespacing, 111

double free error, 63, 319

double quote ("), 39, 493

Doyle, Sir Arthur Conan, 281

drop function, 62

Drop trait, 317–20, 479–81

incompatible with Copy trait, 65

DST (dynamically sized type), 445–46

duck typing, 373

dynamically sized type (DST), 445–46

dynamic dispatch, 374

E

else if expression, 50–51

else keyword, 49

empty type. See never type

encapsulation, 366–68

entry method, 145–47

Entry type, 145–47

enumerate method, 74

enums, 95–108

defining, 96

instantiating, 96

variants of, 96

environment, 269

environment variables, 249–54

eprintln! macro, 255–56

Eq trait, 498

error handling, 149–66

executable file, 6–7

executing code, 6–7

exit status code, 239–40

expect method, 17–18, 26, 157–58

expressions

vs. statements, 44–46

extern crate, 21–22

extern functions, 420–21

F

fearless concurrency, 342

FFI (Foreign Function Interface), 420

field init shorthand, 83

fields, 82

files

reading, 231–32

floating-point data types, 38

numeric operations with, 38

fn keyword, 15

FnMut trait, 265, 269, 447, 465

FnOnce trait, 265, 269, 447, 465

Fn trait, 265, 269, 447, 465

fn type, 446–47

Foreign Function Interface (FFI), 420

for keyword

in trait implementations, 179–80

loop, 54–55

format! macro, 138

forward slash (/), 492

for division, 38

from function

on String, 60–61, 136

on the From trait, 160

fully qualified syntax, 434–37, 447

functional programming, 257

function pointers, 446–47

functions, 42–47

arguments to, 43

bodies, statements and expressions in, 44–46

parameters of, 43–44

returning early from, 46

with return values, 46–47

with multiple return values using a tuple, 67–68

G

Gallant, Andrew, 228

Gamma, Erich, 366

garbage collector (GC), 61

GC (garbage collector), 61

generics, 167–78, 199

default types for, 432–34

in enum definitions, 174–75

in function definitions, 170–73

in method definitions, 175–77

in struct definitions, 173–74

performance of, 177–78

get method

on HashMap<K, V>, 144

on Vec<T>, 131–32

getter, 165

Git, 8, 11

global variables, 421–22

grapheme clusters, 140, 142

green threads, 343

grep, 227

guarding, 356

guessing game, 13–30

H

hash. See HashMap<K, V> type

hasher, 147

hashing function, 142, 147

hash map. See HashMap<K, V> type

HashMap<K, V> type, 142–47

entry method on, 145–47

get method on, 144

insert method on, 142–43

iterating over, 144–45

new function on, 142–43

hash table. See HashMap<K, V> type

Hash trait, 500

heap

and the stack, 58–59

allocating on, 58

Helm, Richard, 366

Hoare, Tony, 100

HTTP (Hypertext Transfer Protocol), 450, 454–56

Hypertext Transfer Protocol (HTTP), 450, 454–56

hyphen (-), 492

for subtraction, 38

I

IDE (Integrated Development Environment), ii, 4

if keyword, 48–52

if let syntax, 107–8

ignore attribute, 219–220

immutability, See Mutability

impl keyword

for defining methods, 91

for defining associated functions, 93–94

for implementing traits, 179–80

indexing syntax, 131–32

indirection, 310–11

inheritance, 368–69

input lifetimes, 196

input/output (io) library, 15, 231–32

installation, 1–4

instance, 82

integer data types, 36–38

numeric operations with, 38

signed, 36–37

type suffixes of, 37

unsigned, 36–37

Integrated Development Environment (IDE), ii, 4

integration tests, 222–25

interfaces. See traits

interior mutability, 323–32, 362

invalidated variable, 64

io (input/output) library, 15, 231–32

IpAddr type, 96–98

irrefutable patterns, 395–96

isize type

architecture dependent size of, 37

indexing collection with, 38

iterator adaptors, 273–74, 280–81

Iterators, 270–77

creating with iter method, 73–74

enumerate method on, 74

next method on, 271–272

performance of, 281–83

iter method, 73–74

J

Johnson, Ralph, 366

JoinHandle type, 345

K

Kay, Alan, 365

keywords, 32, 487–89

L

last in, first out, 58

lazy evaluation, 264, 270

len method, 74

let keyword, 16

library crate, 7, 8, 19, 110

license, 295

license identifier value, 295

lifetime bounds, 428–29

lifetime elision rules, 196–98

lifetimes, 187–99

annotation of, 190–95

elision of, 195–98

inferring for trait objects, 429–430

lifetime subtyping, 423–28

line feed, 454

linker, 2–3

Linux installation, 2–3

Little Book of Rust Macros, The, 504

lock, 356–59

loop keyword, 26–27, 53

M

macOS installation, 2–3

macro\_export annotation, 503

macro\_rules! macro, 502–3

macros, 501–10

declarative, 502–4

procedural, 504–9

macro\_use annotation, 502–3

main function, 5–6

mangling, 421

map. See HashMap<K, V> type

match expression, 102–7

exhaustiveness of, 106

handling comparison results with, 23–24

handling Result values with, 28–29

match guard, 156, 408–10

memoization, 264

memory leak, 332

message passing, 349–55

metaprogramming, 502

methods

defined on structs, 90–93

defined on enums, 99

method syntax, 91

M:N threading model, 343

mock object, 325–30

mod keyword, 110–112

modules, 109–27

moving to other files, 112–18

root, 118

monomorphization, 177–78

move keyword, 269–70, 347–49

moving ownership, 62–64

with function calls, 66

with function return values, 66–68

vs. borrowing, 68–73

mpsc (multiple producer, single consumer), 350, 354, 473

multiple producer, single consumer (mpsc), 350, 354, 473

multiplication, 38

Mutex<T> type, 356–62, 473–74, 478–79

poisoned, 475

mut keyword

making a variable mutable with, 33

making a reference mutable with, 69–71

mutable, See mutability

mutability

of variables, 32–33

of references, 69–71

mutual exclusion, 356

N

never type (!), 443–44, 494

new function

on HashMap<K, V>, 142–43

on String, 135–36

on Vec<T>, 130

new project setup, using cargo, 14

newtype pattern, 439–441

null, 100–102

numeric operations, 38

O

object, 366, 370. See also HashMap<K, V> type

object-oriented programming (OOP), 365–87

object-safe traits, 374–75, 383

OOP (object-oriented programming), 365–87

operator overloading, 432–34

operators, 491–93

optimizations

compiling with, 10–11

Option<T> enum, 100–102

Ordering type, 23

Ord trait, 499

orphan rule, 180, 439

output lifetimes, 196

ownership, 57–79

rules, 59

and functions, 66–68

P

package

section in Cargo.toml, 8, 294–95

panic

on invalid array element access, 42

panic! macro, 150–53

vs. Result<T, E>, 161–65

parallel programming, 341

parameters, 43–44

parentheses, (()), 496

for function parameters, 6, 15

for tuples, 39–40

parse method, 25

PartialEq trait, 498

PartialOrd trait, 499

PATH system variable, 2, 3, 302–3

patterns, 389–411

binding to values with, 104–5

in for loops, 392–93

in function parameters, 394

in if let syntax, 107, 390–91

in let statements, 393–94

in match expressions, 102–3, 390

in while let syntax, 392

refutable vs. irrefutable, 395–96

percent (%), 492

for remainder, 38

plus (+), 492

for addition of number types, 38

for addition of strings, 137–38

for specifying multiple trait bounds, 183

pointer, 305

dangling, 72

raw, 415–17

to data on the heap, 58

poisoned mutex, 475

polymorphism, 368

prelude, 15

primitive obsession, 235

println! macro, 6

placeholders in, 18

privacy rules, 121

private

vs. public, 119

procedural macros, 504–9

process, 342

proc\_macro crate, 507

profile

section in Cargo.toml, 286–87

profiles, 286–87

propagating errors, 158–61

pub keyword, 119–21

public

API, 120, 290–93

vs. private, 119

pub use, 290–93

push method, 131, 137

push\_str method, 61, 137

Q

question mark operator (?), 159–61, 493

quote crate, 507–9

R

race conditions, 70, 343

RAII (Resource Acquisition Is Initialization), 62

rand crate, 19, 21–23

random number functionality, 19, 21–23

Range type, 55

raw pointers, 415–17

Rc<T> type, 320–23, 330–39

read\_line method, 16–17

receiver, 350

recoverable errors, 149

recursive types, 308–11

re-export, 290–93

RefCell<T> type, 323–39

reference counting, 306, 320–23

atomically, 361–62

reference cycles, 332–39

references

and borrowing, 68–73

creating in patterns, 407–8

dangling, 72–73

dereferencing, 68

for accessing data from multiple places, 17

mutability of, 69–71, 73

rules of, 73

refutable patterns, 395–96

registry, 20, 287–96

release mode, 10–11

release profiles, 286–87

remainder, 38

request line, 454

request-response protocol, 450

Resource Acquisition Is Initialization (RAII), 62

Result<T, E> type, 17, 153–61

expect method on, 17–18, 26, 157–58

type aliases for, 442–43

unwrap method on, 157–58

unwrap\_or\_else method on, 239

vs. panic!, 161–65

return keyword, 46

return values

of functions, 46–47

multiple using a tuple, 67–68

rev method, 55

ripgrep, 228, 302–3

RLS (Rust Language Server), ii

root module, 118

.rs file extension, 5

running code, 6–7

runtime, 343

Rustaceans, 3–4

rustc, 3, 5, 6–7

rustfmt, ii, 6

Rust Language Server (RLS), ii

Rustonomicon, The, 133, 339, 363

rustup, 1–4

commands

doc, 4

uninstall, 3

update, 3

S

scalar data types, 36–39

and Copy trait, 65–66

scope, 60

SCREAMING\_SNAKE\_CASE, 421

self parameter, 90

Self keyword, 374–75

Semantic Versioning (SemVer), 19, 296

semicolon (;), 6, 493

SemVer (Semantic Versioning), 19, 296

Send trait, 362–63, 423, 465

sequence, 55

server, 450

shadowing, 25, 34–36

shallow copy, 64

shared-state concurrency, 355–62

should\_panic attribute, 212–15

signed integer types, 36–37

single quote ('), 493–94

for characters, 39

for lifetime parameter names, 190

Sized trait, 445–46, 448

slice type, 73–79

of array, 78–79

string slices, 75–78

smart pointer, 305–39

snake case, 42

Software Package Data Exchange (SPDX), 295

SPDX (Software Package Data Exchange), 295

square brackets ([]), 496

for array creation, 41

for element access, 41, 131–32

stack

and the heap, 58–59

last in, first out ordering of, 58

popping off of, 58

pushing onto, 58

standard error (stderr), 254–56

standard output (stdout), 254–56

statements

vs. expressions, 44–46

state objects, 376

state pattern, 376–84

statically typed, 36

static dispatch, 374

static method. See associated function

static variables, 421–22

status line, 455

stderr (standard error) , 254–56

stdin function, 16

stdout (standard output) , 254–56

stream, 451–53

stringify! macro, 509

string literal, 60

of string slice type, 75–78

storage in the binary of, 61

string slice type (&str), 75–78

String type, 60–61, 135–142

as\_bytes method on, 73–74

bytes method on, 141–42

chars method on, 141

concatenation with +, 137–38

from function on, 60–61, 136

indexing into, 139–140

internal structure of, 62–63, 139–40

iterating over, 141–42

len method on, 74

new function on, 135–36

parse method on, 25

push method on, 137

push\_str method on, 61, 137

slicing, 140–41

trim method on, 25

UTF-8 encoding of, 136

Stroustrup, Bjarne, 282

structs, 81–94

defining, 81–82

field init shorthand, 83

fields, 82

instantiating, 81–82

ownership of data, 85–86

tuple, 84–85, 439–40

unit-like, 85

update syntax, 84

subtraction, 38

super keyword, 125–27

supertraits, 437–39

symbols, 493–96

syn crate, 507–8

Sync trait, 362–63, 423

T

TCP (Transmission Control Protocol), 450–52

TDD (Test-driven development), 244

test attribute, 202–3

test double, 325

Test-driven development (TDD), 244

test functions, 202–5

tests, 201–26

custom failure messages for, 210–12

filtering, 218–19

ignoring, 219–20

integration, 222–25

of private functions, 221–22

organizing, 220–25

running, 215–20

unit, 220–22

writing, 201–15

thread pool, 461–85

threads, 342–49

creating with spawn, 344, 462–63

joining, 345

pausing with sleep, 344

thunk, 442

TOML (Tom's Obvious, Minimal Language), 8

Tom's Obvious, Minimal Language (TOML), 8

to\_string method, 136, 186

trait bounds, 182–87, 199

conditionally implementing methods with, 185–87

trait objects, 369–75, 448

inferring lifetimes of, 429–30

traits, 178–87

associated types in, 431–32

default implementations of, 181–82

defining, 178–79

implementing, 179–80

unsafe, 422–23

Transmission Control Protocol (TCP), 450–52

transmitter, 350

trim method, 25

tuple data type, 39–40

returning multiple values from a function with, 67–68

tuple structs, 84–85, 439–40

type alias, 441–43, 474

type annotation, 25, 36

type inference, 24

type suffixes, 37

U

underscore (\_), 494

as a catchall pattern, 28, 106–7, 403–5

as a visual separator in integer literals, 37

Unicode Scalar Value, 39, 139–41

Uniform Resource Identifier (URI), 454

Uniform Resource Locator (URL), 454

unit-like structs, 85

unit tests, 220–22

unrecoverable errors, 149–53

unrolling, 283

unsafe, 414–23

functions, 417–20

superpowers, 414, 423

traits, 422–23

unsigned integer types, 36–37

unsized type. See dynamically sized type

unwinding, 150

unwrap method, 157–58

unwrap\_or\_else method, 239

URI (Uniform Resource Identifier), 454

URL (Uniform Resource Locator), 454

use keyword, 22, 123–27

user input, 15

usize type

architecture dependent size of, 37

indexing collection with, 38

UTF-8 encoding, 136, 139–140

V

variables

mutability, 32–33

global, 421–22

shadowing, 25, 34–36

static, 421–22

storing values in, 15–16

vs. constants, 34

variants, 96

vec! macro, 130

vector. See Vec<T> type.

Vec<T> type, 130–34

get method on, 131–32

iterating over, 133–34

new function on, 130

push method on, 131

vertical pipe (|), 493–94

in closure definitions, 261

in patterns, 398

Visual Studio

build tools for, 3

Vlissides, John, 366

W

weak reference, 334–35

Weak<T> type, 334–39

where clause, 183

while loop, 53–54

Windows installation, 3

workspaces, 297–302

X

Y

yanking, 296

Z

zero-cost abstractions, iii, 282–83

zero-overhead, 282