Symbols & Numbers

& (ampersand), to create a reference, 17

&str (string slice type), 75–78

-> (arrow), for function return values, 46–47

\* (asterisk), for dereferencing, 68

{} (curly brackets)

creating new scopes with, 45, 71

for function bodies, 15

as placeholders in the println! macro, 18

" (double quote), for strings, 39

' (single quote), for characters, 39

[] (square brackets)

for array creation, 41

for array element access, 41

() (parentheses)

for function parameters, 15

for tuples, 39–40

. (period), for tuple element access, 40

\_ (underscore)

as a catchall pattern, 28

as a visual separator in integer literals, 37

A

Addition, of number types, 38

Ampersand (&), to create a reference, 17

Arguments, to function calls, 43

Arms

in if expressions, 49

in match expressions, 24

Array data type, 40–42

invalid element access, 41–42

iterating over elements of, 54–55

slices of, 78–79

Arrow (->), for function return values, 46–47

as\_bytes method, on String, 73–74

Associated function, implemented on a type, 16

Asterisk (\*), for dereferencing, 68

B

Binary crate, for creating an executable, 19

Boolean data type, 39

in if expressions, 50

Borrowing, and references, 68–73

break keyword, exiting a loop with, 27–28

Byte literal syntax, 37, 74

C

Cargo

commands

doc, 22

new, 14

update, 21

Cargo.toml

dependencies section in, 19

updating crate versions in, 21

Cargo.lock, ensuring reproducible builds with, 20–21

Character data type, 39

clone method, creating deep copies with, 65

cmp method, comparing with, 24

Comments, 48

Compound data types, 39–42

Constants, vs. variables, 34

continue keyword, going to the next iteration of a loop with, 28–29

Control flow, 48–55

Copy trait, for stack-only data, 65

ctrl-c, halting a program with, 27, 53

Curly brackets ({})

creating new scopes with, 45, 71

for function bodies, 15

as placeholders in the println! macro, 18

D

Dangling pointer, 72

Dangling reference, 72–73

Data race, 70–71

Data types, 36–42

annotation of, 25, 36

compound, 39–42

scalar, 36–39

Deep copy

never made automatically, 64

similar to clone, 65

Dependency, using a crate as, 19

dependencies, section in Cargo.toml, 19

Destructuring, of tuples, 40

Division, of number types, 38

Documentation, viewing a crate’s, 22

Double free error, 63

Double quote ("), for strings, 39

drop function, called when an owner goes out of scope, 62

Drop trait, incompatible with Copy trait, 65

E

else if expression, for control flow, 50–51

else keyword, for control flow, 49

enumerate method, on Iterator, 74

expect method, on Result, 17–18, 26

Expressions, vs. statements, 44–46

extern crate, for using an external dependency, 21–22

F

Floating-point data types, 38

numeric operations with, 38

fn keyword, declaring a new function with, 15

for loop, iterating over a collection with, 54–55

from function, on String, 60–61

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

Garbage collector (GC), 61

GC (garbage collector), 61

Guessing game, 13–30

H

Heap

and the stack, 58–59

allocating on, 58

I

if keyword, for control flow, 48–52

in let statements, 51–52

Immutable, See Mutability

Input/output (io) library, for accepting user input, 15

Integer data types, 36–38

numeric operations with, 38

signed, 36–37

type suffixes of, 37

unsigned, 36–37

Invalidated variable, as a result of a move, 64

io (input/output) library, for accepting user input, 15

isize type

architecture dependent size of, 37

indexing collection with, 38

Iterator type

creating with iter method, 73–74

enumerate method on, 74

iter method, creating Iterator with, 73–74

J

K

Keywords, reserved, 32

L

Last in, first out, ordering of a stack, 58

len method, on String, 74

let keyword, declaring a new variable with, 16

Library crate, for using code in another project, 19

loop keyword, creating an infinite loop with, 26–27, 53

M

match expression

handling comparison results with, 23–24

handling Result values with, 28–29

Moving ownership, 62–64

with function calls, 66

with function return values, 66–68

vs. borrowing, 68–73

Multiplication, of number types, 38

mut keyword, making a variable mutable with, 33

Mutable, See Mutability

Mutability

of variables, 32–33

of references, 69–71

N

New project setup, using cargo, 14

Numeric operations, with number types, 38

O

Ordering, comparison results of type, 23

Ownership, 57–79

rules, 59

and functions, 66–68

P

Panic, on invalid array element access, 42

Parameters, of functions, 43–44

Parentheses, (())

for function parameters, 15

for tuples, 39–40

parse method, on String, 25

Period (.), for tuple element access, 40

Pointer

dangling, 72

to data on the heap, 58

Prelude, types brought into the scope of every program, 15

println! macro, placeholders in, 18

push\_str method, on String, 61

Q

R

RAII (Resource Acquisition Is Initialization), 62

rand crate, generating random numbers with, 19, 21–23

Random number functionality, from the rand crate, 19, 21–23

Range, generating a sequence with, 55

read\_line, to get input from the user, 16–17

References

and borrowing, 68–73

dangling, 72–73

dereferencing, 68

for accessing data from multiple places, 17

mutability of, 69–71, 73

rules of, 73

Registry, of open source crates, 20

Remainder, of number types, 38

Resource Acquisition Is Initialization (RAII), 62

Result

expect method on, 17–18, 26

for handling errors, 17

return keyword, returning early from functions with, 46

Return values

of functions, 46–47

multiple using a tuple, 67–68

rev method, reversing ranges with, 55

S

Scalar data types, 36–39

and Copy trait, 65–66

Scope, of variables, 60

Semantic Versioning (SemVer), specifying dependencies with, 19

SemVer (Semantic Versioning), specifying dependencies with, 19

Sequence, generating with Range, 55

Shadowing, of variables, 25, 34–36

Shallow copy, similar to move, 64

Signed integer types, 36–37

Single quote ('), for characters, 39

Slice type, 73–79

of array, 78–79

string slices, 75–78

Snake case, for function and variable names, 42

Square brackets ([])

for array creation, 41

for array element access, 41

Stack

and the heap, 58–59

last in, first out ordering of, 58

popping off of, 58

pushing onto, 58

Statements, vs. expressions, 44–46

Statically typed, 36

Static method. See associated function

stdin, function to get a handle to, 16

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

as\_bytes method on, 73–74

from function on, 60–61

internal structure of, 62–63

len method on, 74

parse method on, 25

push\_str method on, 61

trim method on, 25

Subtraction, of number types, 38

T

trim method, on String, 25

Tuple data type, 39–40

returning multiple values from a function with, 67–68

Type annotation, 25, 36

Type inference, 24

Type suffixes, of number literals, 37

U

Underscore (\_)

as a catchall pattern, 28

as a visual separator in integer literals, 37

Unicode Scalar Value, 39

Unsigned integer types, 36–37

use, bringing a type into scope with, 22

User input, with the io library, 15

usize type

architecture dependent size of, 37

indexing collection with, 38

V

Variables

mutability, 32–33

shadowing, 25, 34–36

storing values in, 15–16

vs. constants, 34

W

while loop, repeating conditionally with, 53–54

X

Y

Z