



**Learn**

**Rust**

**Programming**

**Today**

**JERE KÄPYAHO**

**Learn Rust Programming Today**



# **Learn Rust Programming Today**

**From Building Blocks to a Working Application**

**Jere Käpyaho**

Automated analysis of this title to obtain information, especially models, trends, and correlations, is strictly prohibited under applicable laws considering text mining.

Copyright © 2026 Jere Käpyaho

Editing and cover design: Conifer Productions Oy, Tampere, Finland

Publisher: BoD · Books on Demand, Mannerheimintie 12 B, 00100 Helsinki, [bod@bod.fi](mailto:bod@bod.fi)

Printing press: Libri Plureos GmbH, Friedensallee 273, 22763 Hamburg, Germany

ISBN: 978-952-89-0298-0

# Table of Contents

Introduction .....	xvii
Rust in a nutshell .....	xviii
Notational differences .....	xviii
Conceptual differences .....	xix
Learning Rust by doing .....	xix
Who this book is for .....	xx
Source code repositories .....	xxi
Acknowledgments .....	xxi
I Getting Started with Rust .....	1
1 Using the Rust Playground .....	5
2 Getting the Rust tools .....	11
3 Using the Rust compiler .....	15
4 Editors for Rust source code .....	17
Editing Rust source code .....	17
Visual Studio Code .....	17
Vim, Neovim, and Emacs .....	18
Zed .....	19
RustRover .....	19
5 Getting to know Cargo .....	21
Hello, Cargo .....	21
The Cargo configuration file .....	23
II Building Blocks of Rust Programs .....	25
6 Strings and numbers .....	31
Storing text and numbers .....	31
Strings .....	31
String slices .....	33
Numbers .....	34
7 Tuples and arrays .....	39
Storing data in tuples and arrays .....	39
Tuples .....	39
Arrays .....	40
Sorting the array .....	44
8 Iteration and looping .....	47
Looping through the elements .....	47
9 Making decisions .....	51

Truth values .....	51
Deciding which statements to execute .....	51
10 Functions .....	57
Defining and using functions .....	57
11 Structs .....	63
Grouping values into structs .....	63
Initializing structs .....	65
Struct implementations .....	66
Methods .....	69
12 Enums and pattern matching .....	73
Using enums for alternative values .....	73
Enums with associated values .....	78
Pattern matching .....	78
13 Optional values .....	83
Option instead of nullable .....	83
Handling optional values .....	87
14 Ownership, references and borrowing .....	89
Memory safety .....	89
Ownership .....	89
Scope .....	91
Examples of ownership and scope .....	91
References .....	96
Borrowing .....	97
15 Vectors .....	99
Storing elements in a dynamic vector .....	99
Sorting a vector .....	101
16 Command-line arguments .....	103
Getting at the arguments .....	103
Converting a string to a MonthDay .....	103
Handling the month-day argument .....	105
17 Errors .....	109
No panic, but a result .....	109
Using the unwrap and expect methods .....	110
Propagating errors .....	111
18 Traits .....	115
Traits specify behavior .....	115
Implementing the Display trait .....	116
The Default trait .....	118

The FromStr trait .....	120
Defining traits .....	123
19 Crates .....	125
Beyond the standard library .....	125
Making random dates with some crates .....	126
Making random descriptions .....	130
Making random events .....	132
20 Testing .....	135
Testing with Cargo .....	135
A better Date struct .....	135
Creating a test module .....	137
Random dates again .....	139
Writing tests for the Category struct .....	140
21 Comments and documentation .....	143
Documentation is your friend .....	143
Comments .....	143
Documentation comments .....	145
Generating documentation .....	145
22 The Birthday program .....	147
Starting the Birthday program .....	147
Getting the BIRTHDATE environment variable .....	148
Parsing BIRTHDATE as a date .....	149
Constructing the age message .....	152
Adding unit tests .....	153
III Programming Today .....	157
23 Creating the Today program .....	163
Starting the Today program with Cargo .....	163
The revised Event struct .....	164
The revised Category and MonthDay structs .....	165
Testing the Today program .....	167
24 Modules .....	169
Packages and crates .....	169
Modules .....	169
The birthday module .....	171
The events module .....	172
25 Event providers .....	179
Introducing event providers .....	179
Creating a new module for the providers .....	179

A simple event provider for testing .....	180
Creating submodules for providers .....	183
26 Text file event provider .....	187
Putting events in a text file .....	187
Developing a text file event provider .....	188
Integrating the text file provider .....	193
27 CSV file event provider .....	195
The CSV file format .....	195
Using the csv crate .....	196
Integrating the CSV file provider .....	199
28 Configuration directories .....	201
Everything in its right place .....	201
Getting the configuration directory name .....	202
Building the configuration path .....	203
Placing the data files .....	205
29 Configuration file .....	207
Using a TOML file to store the configuration .....	207
Parsing the TOML file .....	208
Instantiating the event providers .....	211
Preventing duplicate event providers .....	215
30 Reading events from a SQLite database .....	217
Adding a database to the program .....	217
Getting SQLite .....	218
The event database schema in SQLite .....	218
Using the rusqlite crate .....	221
Testing database operations .....	224
Getting events from the database .....	226
Integrating the SQLite event provider .....	228
31 Reading events from a web service .....	231
Reading data from the web .....	231
The web service and REST API .....	231
Testing the API with curl .....	231
Making the web event provider .....	233
Making an HTTP request with the request crate .....	234
Preparing for the web event provider .....	237
32 Filtering events at the source .....	239
Getting just the events we want .....	239
Introducing event filters .....	239

Filter options .....	239
Accepting the event .....	240
Constructing filters with the Builder pattern .....	242
Testing the filters and the builder .....	244
Refactoring the event providers .....	246
Refactoring the run function .....	251
33 Handling command-line arguments .....	255
Controlling the program from the command line .....	255
Parsing command-line arguments with clap .....	257
Defining subcommands .....	261
34 Adding new event kinds .....	265
More than just singular events .....	265
Annual events .....	265
Updating event providers to support annual events .....	267
Rule-based events .....	270
Resolving the rule .....	277
Updating the CSV file event provider to support rule-based events .....	284
35 Logging .....	287
Leaving a log of the execution .....	287
Adding the logging crates .....	287
Logging levels and macros .....	288
Converting println! output to logging macros .....	290
36 Adding new events .....	293
New events from the command line .....	293
Revising the EventProvider trait .....	294
Implementing event addition for TextFileProvider .....	294
Implementing event addition for CSVFileProvider .....	296
Command-line arguments for adding events .....	298
Adding events to the SQLite database .....	302
Trying to add an event from the command line .....	304
Updating the providers subcommand .....	305
Adding an event (finally) .....	307
37 Grouping and sorting the events .....	309
All the events, grouped and sorted .....	309
Partitioning the events .....	312
38 Today 1.0 .....	315
First public version of Today .....	315

Building a release version .....	315
Installing the program .....	315
Configuring the program .....	316
IV Going Further than Today .....	319
39 Refactoring the Event struct .....	323
A different kind of Event .....	323
Focusing on the date .....	323
Simplified .....	328
40 Making an XML event provider .....	329
Making an XML event provider .....	329
Defining an XML Schema .....	329
Making the XML provider .....	333
Parsing XML .....	334
41 One final refactoring .....	341
Configuration of event provider activity .....	341
Introducing the event manager .....	342
Checking for active providers .....	344
Updating the provider information .....	348
Final main function .....	351
42 Future development ideas .....	355
Localized category names .....	355
Category exclusions .....	355
Year count for events .....	355
Showing upcoming events .....	355
Color terminal display .....	356
A About this book .....	357
B About the author .....	359
Further reading .....	361
Index .....	363

# List of Examples

1.1 The classic "Hello, world!" program in Rust .....	6
1.2 Printing a variable in Rust .....	7
1.3 Trying to assign to an immutable variable .....	8
1.4 Changing the variable to be mutable .....	9
3.1 The traditional "Hello, world!" program .....	15
5.1 The Cargo.toml configuration file .....	24
6.1 Using a string variable .....	32
6.2 Using the println! macro .....	32
6.3 Using a string slice .....	33
6.4 Using an integer variable .....	34
6.5 Using floating-point variables .....	36
7.1 Using a tuple variable .....	39
7.2 Swapping the components of two tuples .....	40
7.3 Making an array with tuple values as elements .....	41
7.4 Sorting an array of tuples .....	44
8.1 Iterating the elements of an array with a while loop .....	47
8.2 Using an iterator and the for statement on the array .....	49
9.1 Listing events that match a target date .....	53
10.1 A function to extract year, month, and day .....	57
10.2 Using the get_year_month_day function .....	59
11.1 The Date struct .....	64
11.2 The Event struct .....	64
11.3 Replacing tuples with structs .....	64
11.4 Utility functions to make dates and events .....	67
11.5 Using the field init shorthand .....	67
11.6 An impl block for the Date struct .....	68
11.7 An impl block for the Event struct .....	69
11.8 A helper struct and a related accessor function .....	69
11.9 The complete events program .....	70
12.1 An enum representing the months .....	74
12.2 Date struct using the Month enum .....	74
12.3 The events program updated to use the Month enum .....	76
12.4 Day count without leap year checking .....	79
12.5 Day count with less match arms .....	79
12.6 Day count with leap year check .....	80

12.7 Implementing a method for an enum .....	81
13.1 Category struct with optional field .....	84
13.2 The impl block for the Category struct .....	84
13.3 Event struct with a category field .....	85
13.4 Event declarations updated to include category .....	85
13.5 Handling the optional second category component .....	87
14.1 Trying to use a variable that is out of scope .....	92
14.2 Using a variable in the current scope .....	93
14.3 Trying to assign a type that does not implement the Copy trait .....	93
14.4 Deriving the Copy trait .....	94
14.5 Deriving both Copy and Clone along with Debug .....	96
14.6 Multiple immutable references .....	96
14.7 Mutable references to a variable .....	97
15.1 Declaring a vector and adding elements .....	99
15.2 Using the vec! macro to initialize a vector .....	100
16.1 Making a MonthDay value from a string .....	103
16.2 Making a MonthDay value from an integer .....	104
16.3 Handling the month-day argument .....	105
17.1 Trying to open a file .....	110
17.2 Using the unwrap method to handle the error .....	111
17.3 Using the expect method to handle the error .....	111
17.4 Propagating errors from a function .....	112
18.1 Using the derive attribute .....	115
18.2 Implementing the Display trait for Category .....	116
18.3 Printing out struct values normally .....	117
18.4 Display trait for Event .....	117
18.5 Implementing the Default trait .....	118
18.6 Testing the Display and Default traits .....	119
18.7 Implementing the FromStr trait .....	121
18.8 Implementing FromStr for MonthDay .....	122
18.9 Dealing with Result when parsing MonthDay .....	123
19.1 Method for checking the validity of a date .....	126
19.2 Checking some good and bad dates .....	127
19.3 Generating random dates .....	129
19.4 Testing random date generation .....	129
19.5 Generating random sentences .....	131
19.6 Generating a random event .....	132
19.7 The first ten of 10,000 random events .....	133

19.8 A sampling of randomly generated events .....	133
20.1 Improving the Date struct .....	136
20.2 New random function for Date .....	137
20.3 Unit tests for the Date struct .....	138
20.4 Making random dates again .....	139
20.5 Unit tests for the Category struct .....	140
21.1 Example of a line comment .....	144
21.2 Example of a block comment .....	144
21.3 Example of a documentation comment .....	145
22.1 Generating the Birthday program .....	147
22.2 Getting the BIRTHDATE value .....	148
22.3 Handling the result of date parsing .....	150
22.4 Stub implementation of the make_message function .....	151
22.5 The complete make_message function .....	152
22.6 Testing the birthday message generation .....	153
23.1 Initial Cargo.toml file for Today .....	163
23.2 The EventKind enum and the revised Event struct .....	164
23.3 Singular event constructor and helpers .....	164
23.4 Improved FromStr implementation for Category .....	165
23.5 Improved FromStr implementation for MonthDay .....	166
23.6 Testing the initial program version .....	167
24.1 Initial contents of lib.rs .....	170
24.2 The revised main function .....	170
24.3 The birthday module .....	171
24.4 Invoking the handle_birthday function .....	171
24.5 The new birthday module .....	172
24.6 Invoking handle_birthday from the birthday module .....	172
24.7 Accessor functions for Event .....	173
24.8 Complete listing of src/lib.rs up to this point .....	174
24.9 Complete listing of src/events.rs up to this point .....	175
25.1 The EventProvider trait in src/providers/mod.rs .....	180
25.2 A simple event provider .....	181
25.3 Using the event providers in lib.rs .....	182
25.4 Example of a nested module in src/providers.rs .....	183
25.5 An event provider in a separate module .....	184
26.1 Events in a plain text file .....	187
26.2 Text file event provider .....	188
26.3 EventProvider trait implementation for text files .....	189

26.4 Possible states of reading the event text file .....	190
26.5 The <code>get_events</code> method implementation for <code>TextFileProvider</code> .....	190
26.6 Making use of the text file event provider .....	193
27.1 List of events as a CSV file .....	195
27.2 CSV file provider .....	196
27.3 <code>EventProvider</code> trait implementation for the CSV file provider .....	197
27.4 Using the CSV file provider .....	199
28.1 Helper program to print out the configuration directory .....	202
28.2 Helper function to get the configuration directory .....	204
28.3 Constructing the configuration directory .....	204
28.4 Constructing paths to data files .....	205
29.1 The configuration file in TOML format .....	207
29.2 Project file with <code>toml</code> and <code>serde</code> crates added .....	208
29.3 Configuration data structures .....	209
29.4 Updated <code>run</code> function signature .....	209
29.5 Parsing the TOML configuration file in <code>main.rs</code> .....	210
29.6 Instantiating the event providers in <code>src/lib.rs</code> .....	213
29.7 Revised <code>run</code> function implementation .....	214
29.8 Checking for existing event providers by name .....	215
30.1 Database schema SQL statements .....	219
30.2 Adding a category and some events to the database .....	220
30.3 SQLite event provider implementation .....	221
30.4 Type aliases for category ID and hash map .....	222
30.5 Getting the categories from the database .....	223
30.6 Testing the database operations .....	225
30.7 Implementing the <code>EventProvider</code> trait for <code>SQLiteProvider</code> .....	226
30.8 Getting events from the SQLite database .....	227
30.9 Testing the <code>get_events</code> method of <code>SQLiteProvider</code> .....	228
30.10 Adding an SQLite event provider to the configuration .....	228
31.1 The <code>Cargo.toml</code> entries for web-related crates .....	234
31.2 Web event provider implementation .....	234
31.3 <code>EventProvider</code> implementation for <code>WebProvider</code> .....	235
31.4 A helper for JSON parsing .....	236
31.5 The rest of the <code>get_events</code> method .....	236
32.1 The <code>EventFilter</code> struct with accessors .....	240
32.2 The <code>accepts</code> method of the event filter .....	241
32.3 The filter builder .....	243
32.4 Using the filter builder .....	244

32.5 Testing the event filter .....	245
32.6 Testing the filter builder .....	245
32.7 The refactored EventFilter trait .....	247
32.8 Constructing the WHERE clause .....	248
32.9 Adding the WHERE clause to the helper function .....	249
32.10 Testing the WHERE clause generation .....	249
32.11 Web provider refactored for filtering .....	250
32.12 The new and improved run function .....	252
33.1 Printing out the command-line arguments .....	255
33.2 Using clap to process command-line arguments .....	258
33.3 Making the date argument optional .....	259
33.4 Constructing an event filter from the command-line arguments ...	260
33.5 Defining subcommands with the clap crate .....	261
33.6 Using match to handle subcommands .....	262
33.7 Helper types for provider configuration .....	263
34.1 Adding a new event kind variant .....	265
34.2 New constructor for creating annual events .....	266
34.3 Display trait implementation for Event .....	267
34.4 Event text file with yearless dates .....	267
34.5 Configuration entry for yearless event provider in today.toml .....	268
34.6 Creating a singular or annual event in get_events .....	269
34.7 Adding a new event kind for rule-based events .....	270
34.8 Constructing a rule-based event .....	271
34.9 Defining a struct for the event rule .....	271
34.10 Enumerated type for the ordinals .....	272
34.11 Enumerated type for the weekdays .....	272
34.12 The parse function in the Rule struct .....	273
34.13 Enriching the Ordinal and Weekday enums with the strum crate .....	275
34.14 Parsing the event rule .....	276
34.15 Parsing the rest of the rule .....	277
34.16 Handling all the EventKind enum variants .....	278
34.17 The year and month_day methods of Rule .....	278
34.18 Tests for the event rule .....	279
34.19 Stub method for resolving the rule .....	280
34.20 Resolving the rule to a date .....	281
34.21 Computing the $n$ th weekday of the month .....	281
34.22 Computing the last weekday of the month .....	282
34.23 Converting Weekday to chrono::Weekday .....	282

34.24 More unit tests for rule resolution .....	283
34.25 Checking for a rule-based event .....	284
35.1 Preparing for logging in the main function .....	288
35.2 Adding some debug output to the program .....	289
35.3 Finding out and printing the configuration directory .....	290
35.4 Properly logging the configuration file search .....	291
36.1 The revised EventProvider trait .....	294
36.2 Adding an event to a text file .....	295
36.3 Adding an event to a CSV file .....	296
36.4 Command-line arguments for adding an event .....	298
36.5 Handling the add subcommand .....	299
36.6 The add_event function in lib.rs .....	299
36.7 Function to create the event providers .....	300
36.8 The add_event function .....	301
36.9 The find_category_id function .....	302
36.10 The EventProvider implementation for SQLiteProvider .....	303
36.11 The insert_event function .....	304
36.12 Showing the listing of event providers .....	306
36.13 Adding the kind for event providers .....	306
37.1 The revised run function .....	309
37.2 Pluralizing the group headings .....	310
37.3 Using the partition method of an iterator .....	312
38.1 A minimal today.toml configuration file .....	316
38.2 Rust tool releases in a CSV file called rustopedia.csv .....	317
39.1 Refactoring Event by the date .....	323
39.2 Constructing an Event with an EventDate .....	324
39.3 Refactoring the year and month_day methods .....	324
39.4 Adding the is_singular method .....	325
39.5 Creating a new-style event in the text file provider .....	326
39.6 Adding a new-style event to a text file .....	327
39.7 Implementing the Display trait for EventDate .....	328
39.8 Collecting events with partition and closure .....	328
40.1 The XML Schema for the event document .....	330
40.2 An instance document with some XML-related events .....	331
40.3 Placeholder implementation of XmlFileProvider .....	333
40.4 Parsing XML events .....	335
40.5 Parsing the many forms of event dates in EventDate::parse .....	338
41.1 The is_active setting for an event provider .....	341

# Learn Rust Programming Today

The Rust programming language empowers everyone to write safer software without sacrificing performance. *Learn Rust Programming Today* is a book that presents the most important features of the Rust language in the context of an actual application called *Today*, showing historical events collected from various data sources.

The application is developed gradually over the course of the book, and all the Rust examples presented are relevant for the end result. Each chapter adds new operations to the program until it is feature complete. The book also has a supporting GitHub repository with all the source code for the examples.

Among the many topics covered are creating domain-specific data types, reading from files, making network requests and database queries, using crates, and writing unit tests. This book offers a unique journey to any practicing programmer who is considering Rust as a second language.

