

## Spis treści

Wstęp 15

1. Programiści systemowi mogą mieć fajne zabawki 19

Rust zdejmuje ciężar z naszych barków 20

Programowanie współbieżne zostaje ujarzmione 21

A na dodatek Rust wciąż jest szybki 22

Rust ułatwia współpracę 22

2. Pierwsze spotkanie z Rustem 24

rustup i Cargo 25

Funkcje w języku Rust 27

Pisanie i uruchamianie testów 29

Obsługa argumentów wiersza poleceń 30

Prosty serwer WWW 34

Programowanie współbieżne 40

Czym jest zbiór Mandelbrota? 41

Parsowanie argumentów wiersza poleceń 45

Odwzorowanie pikseli na liczby zespolone 47

Rysowanie zbioru 49

Zapis obrazu do pliku 50

Program Mandelbrota działający współbieżnie 51

Uruchomienie programu 57

Przezroczyste bezpieczeństwo 58

Narzędzia systemów plików i wiersza poleceń 58

Interfejs wiersza poleceń 59

Odczyt i zapis plików 61

Znajdowanie i zastępowanie 62

3. Typy proste 65

Typy numeryczne o ustalonej długości 68

Typy całkowite 68

Sprawdzane, przenoszące, nasycające i przepełniające operacje arytmetyczne 72

Typy zmiennoprzecinkowe 73

Typ logiczny 76

Typ znakowy 76

Krotki 78

Typy wskaźnikowe 80

Referencje 80

Pudełka 81

Wskaźniki niechronione 81

Tablice, wektory i podzbiory 82

Tablice 82

Wektory 83

Podzbiory 86

Typ String 88

Literały łańcuchowe 88

łańcuchy bajtów 89

łańcuchy znaków w pamięci 90

Typ String 91

Podstawowe cechy typu String 92

Inne typy znakowe 93

Nazwy zastępcze typów 93

Co dalej? 93

4. Reguła własności i przenoszenie własności 94

Reguła własności 96

Przenoszenie własności 100

Więcej operacji związanych z przeniesieniem własności 105

Przeniesienie własności a przepływ sterowania 107

Przeniesienie własności a struktury indeksowane 107

Typy kopiowalne — wyjątki od reguł przenoszenia własności 110

Rc i Arc: własność współdzielona 113

5. Referencje 116

Referencje do wartości 117

Stosowanie referencji 120

Referencje Rusta kontra referencje C++ 120

Referencje a operacja przypisania 121

Referencje do referencji 122

Porównywanie referencji 123

Referencje nigdy nie są puste 123

Referencje do wyrażeń 124

Referencje do podzbiorów i zestawów metod 124

Bezpieczeństwo referencji 125

Referencja do zmiennej lokalnej 125

Przekazywanie referencji jako argumentów funkcji 128

Przekazywanie referencji do funkcji 130

Zwracanie referencji 131

Struktura zawierająca referencje 132

Odrębny cykl życia 134

Pomijanie parametrów cyklu życia 136

Referencje współdzielone kontra mutowalne 137

Walka ze sztormem na morzu obiektów 144

6. Wyrażenia 146

Język wyrażeń 146

Priorytety i łączność operatorów 147

Bloki kodu i średniki 150

Deklaracje 151

if i match 153

if let 154

Pętle 155

Sterowanie przepływem w pętlach 157

Wyrażenie return 158

Analiza przepływu sterowania 159

Wywołania funkcji i metod 160

Pola i elementy 161

Operatory referencji 163

Operatory arytmetyczne, bitowe, porównania i logiczne 163

Przypisanie 164

Rzutowanie typów 165

Domknięcia 166

Co dalej? 166

7. Obsługa błędów 167

Błąd panic 167

Odwiniecie stosu 168

Przerywanie procesu 169

Typ Result 170

Przechwytywanie błędów 170

Nazwy zastępcze typu Result 172

Wyświetlanie informacji o błędach 172

Propagacja błędów 174

Jednoczesna obsługa błędów różnych typów 175

Błędy, które nie powinny się zdarzyć 177

Ignorowanie błędów 178

Obsługa błędów w funkcji main() 178

Definiowanie własnego typu błędu 180

Co daje nam typ Result? 181

8. Paczki i moduły 182

Paczki 182

Edycje 185

Profile budowania 186

Moduły 187

Moduły zagnieżdżone 188

Umieszczanie modułów w oddzielnych plikach 190

Ścieżki i importy 192

Standardowe preludium 195

Publiczne deklaracje use 195

Publiczne pola struktur 195

Stałe i zmienne statyczne 196

Zmiana programu w bibliotece 197

Katalog src/bin 198

Atrybuty 200

Testy i dokumentacja 202

Testy integracyjne 204

Dokumentacja 205

Doc-testy 208

Definiowanie zależności 210

Wersje 211

Cargo.lock 212

Publikowanie paczek na stronie crates.io 213

Obszary robocze 215

Więcej fajnych rzeczy 216

9. Struktury 217

Struktury z polami nazywanymi 217

Struktury z polami numerowanymi 220

Struktury puste 221

Reprezentacja struktur w pamięci 221

Definiowanie metod w bloku impl 222

Przekazywanie self z użyciem typów Box, Rc lub Arc 225

Funkcje powiązane z typami 227

Powiązane stałe 228

Struktury generyczne 228

Struktury z parametrem cyklu życia 230

Dziedziczenie standardowych zestawów metod 231

Zmienność wewnętrzna 232

10. Typy wyliczeniowe i wzorce 236

Typy wyliczeniowe 237

Typy wyliczeniowe zawierające dane 239

Typ wyliczeniowy w pamięci 240

Większe struktury danych stosujące typy wyliczeniowe 241

Generyczne typy wyliczeniowe 243

Wzorce 245

Literały, zmienne i symbole wieloznaczne 247

Wzorce w postaci krotki lub struktury 249

Wzorce typu tablica i fragment 250

Wzorce z referencjami 251

Strażniki wzorców 253

Dopasowanie do wielu wartości 254

Wzorce @ 254

Gdzie używamy wzorców 255

Wypełnianie drzewa binarnego 256

Podsumowanie 258

11. Zestawy metod i typy generyczne 259

Stosowanie zestawów metod 261

Obiekt implementujący zestaw metod 262

Funkcje generyczne i parametry typów 264

Na co się zdecydować? 267

Definiowanie i implementacja zestawów metod 269

Metody domyślne 270

Implementacja zestawów metod dla istniejących już typów 272

Zestaw metod a słowo kluczowe Self 273

Rozszerzanie zestawu metod (dziedziczenie) 275

Funkcje powiązane z typami 276

W pełni kwalifikowana nazwa metody 277

Zestawy metod definiujące relacje między typami 278

Typy powiązane 279

Generyczny zestaw metod (czyli jak działa przeciążanie operatorów) 282

Impl Zestaw jako typ wyniku 283

Stałe powiązane 285

Inżynieria wsteczna ograniczeń 286

Zestawy metod u podstaw 289

12. Przeciążanie operatorów 290

Operatory arytmetyczne i bitowe 291

Operatory jednoargumentowe 293

Operatory dwuargumentowe 294

Operatory przypisania złożonego 295

Test równości 297

Porównania szeregujące 300

Interfejsy Index i IndexMut 302

Inne operatory 304

### 13. Interfejsy narzędziowe 306

Drop 307

Sized 310

Clone 313

Copy 314

Deref i DerefMut 315

Default 318

AsRef i AsMut 320

Borrow i BorrowMut 321

From i Into 323

TryFrom i TryInto 326

ToOwned 327

Borrow i ToOwned w działaniu 328

### 14. Domknięcia 330

Przechwytywanie zmiennych 331

Domknięcia, które pożyczają wartość 332

Domknięcia, które przejmują własność 333

Typy funkcji i domknięć 334

Domknięcia a wydajność 336

Domknięcia a bezpieczeństwo 338

Domknięcia, które zabijają 338

FnOnce 338

FnMut 340

Copy i Clone dla domknięć 342

Funkcje zwrotne 343

Skuteczne korzystanie z domknięć 347

### 15. Iteratory 349

Iterator i Intoliterator 350

Tworzenie iteratorów 352

Metody iter i iter\_mut 352

Implementacje interfejsu Intoliterator 353

Funkcje from\_fn i successors 355

Metody drain 356

Inne źródła iteratorów 357

Adaptery iteratorów 358

map i filter 358

filter\_map i flat\_map 361

flatten 363

take i take\_while 364

skip i skip\_while 365

peekable 366

fuse 367

Iteratory obustronne i rev 368

inspect 369

chain 370

enumerate 370

zip 371

by\_ref 372

cloned i copied 373

cycle 373

Konsumowanie iteratorów 374

Proste agregaty: count, sum i product 374

max i min 375

max\_by i min\_by 375

max\_by\_key i min\_by\_key 376

Porównywanie sekwencji elementów 376

any i all 377

position, rposition oraz ExactSizeIterator 377

fold i rfold 378

try\_fold i try\_rfold 379

nth i nth\_back 380

last 381

find, rfind i find\_map 381

Tworzenie kolekcji: collect i FromIterator 382

Zestaw metod Extend 384

partition 384

for\_each i try\_for\_each 385

Implementacja własnych iteratorów 386

16. Kolekcje 391

Przegląd kolekcji 392

Vec<T> 393

Dostęp do elementów 394

Iteracja 395

Zwiększanie i zmniejszanie wielkości wektora 396

Łączenie 400

Podział 400

Zamiana 403

Sortowanie i wyszukiwanie 403

Porównywanie podzbiorów 405

Elementy losowe 405

Reguły zapobiegające konfliktom w czasie iteracji 406

VecDeque<T> 407

BinaryHeap<T> 409

HashMap<K, V> i BTreeMap<K, V> 410

Entry 414

Iterowanie map 416

HashSet<T> i BTreeSet<T> 417

Iteracja zbioru 418

Kiedy równe wartości są różne 418

Operacje dotyczące całego zbioru 419

Haszowanie 420

Niestandardowe algorytmy haszujące 421

Kolekcje standardowe i co dalej? 423

17. Tekst i łańcuchy znaków 424

Podstawy Unicode 424

ASCII, Latin-1 i Unicode 425

UTF-8 425

Kierunek tekstu 427

Znaki (typ char) 427

Klasyfikacja znaków 427

Obsługa cyfr 429

Zmiana wielkości liter 430

Konwersja znaku do i z liczby całkowitej 430

Typy String i str 431

Tworzenie łańcuchów znaków 432

Prosta inspekcja 432

Dołączanie i wstawianie tekstu 433

Usuwanie i zastępowanie tekstu 435

Konwencje nazewnicze dotyczące wyszukiwania i iterowania 436

Wyszukiwanie tekstu i wzorce 436

Wyszukiwanie i zamiana 437

Iterowanie tekstu 438

Obcinanie 440

Zmiana wielkości liter w łańcuchach 441

Konwersja tekstu do wartości innego typu 441

Konwersja wartości innego typu do tekstu 442

Tworzenie referencji innego typu 443

Tekst jako UTF-8 443

Tworzenie tekstu na podstawie danych UTF-8 444

Alokacja warunkowa 445

łańcuchy znaków jako kolekcje generyczne 447

Formatowanie wartości 447

Formatowanie tekstu 449

Formatowanie liczb 450

Formatowanie innych typów 452

Formatowanie wartości z myślą o debugowaniu 453

Formatowanie i debugowanie wskaźników 454

Wiązanie argumentów za pomocą indeksu i nazwy 455

Dynamiczne definiowanie długości i precyzji 455

Formatowanie własnych typów 456

Użycie języka formatowania we własnym kodzie 458

Wyrażenia regularne 459

Podstawowe użycie wyrażeń regularnych 460

Wyrażenia regularne w trybie leniwym 461

Normalizacja 462

Rodzaje normalizacji 463

Biblioteka unicode-normalization 464

18. Operacje wejścia – wyjścia 465

Obiekty typu reader i writer 466

Obiekty typu reader 467

Buforowany obiekt typu reader 469

Przeglądanie tekstu 470

Pobieranie tekstu 473

Obiekty typu writer 473

Pliki 475

Wyszukiwanie 476

Inne rodzaje obiektów reader i writer 476

Dane binarne, kompresja i serializacja 478

Pliki i katalogi 480

OsStr i Path 480

Metody typów Path i PathBuf 481

Funkcje dostępu do systemu plików 483

Odczyt zawartości katalogu 484

Funkcje bezpośrednio związane z platformą 486

Obsługa sieci 487

19. Programowanie współbieżne 490

Podział i łączenie wątków 492

spawn i join 493

Obsługa błędów w różnych wątkach 495

Współdzielenie niemutowalnych danych przez różne wątki 496

Rayon 498

Zbiór Mandelbrota raz jeszcze 501

Kanały 502

Wysyłanie wartości 504

Odczyt wartości 507

Uruchomienie potoku 508

Cechy kanałów i ich wydajność 509

Bezpieczeństwo wątków: Send i Sync 511

Współpraca iteratora i kanału 514

Potoki i co dalej? 515

Stan współdzielony mutowalny 516

Czym jest muteks? 516

Mutex<T> 518

mut i Mutex 520

Dlaczego Mutex to nie zawsze dobry pomysł? 520

Zakleszczenie (deadlock) 521

Zatruty muteks 522

Kanały z wieloma nadawcami stosujące muteksy 522

Blokady odczytu/zapisu (RwLock<T>) 523

Zmienne warunkowe (Condvar) 524

Typy atomowe 525

Zmienne globalne 527

Rust i pisanie programów wielowątkowych 530

20. Programowanie asynchroniczne 531

Od kodu synchronicznego do asynchronicznego 533

Interfejs Future 534

Funkcje asynchroniczne i wyrażenia await 537

Wywoływanie funkcji asynchronicznych z kodu synchronicznego: block\_on 539

Uruchamianie asynchronicznych zadań 542

Asynchroniczne instrukcje blokowe 546

Tworzenie funkcji asynchronicznych

z wykorzystaniem asynchronicznych instrukcji blokowych 549

Uruchamianie zadań asynchronicznych w pulach wątków 550

Czy operacje asynchroniczne implementują Send? 551

Wykonywanie długotrwałych obliczeń: yield\_now i spawn\_local 554

Porównanie różnych rozwiązań asynchronicznych 555

Rzeczywisty asynchroniczny klient HTTP 556

Asynchroniczny klient i serwer 557

Typy błędów i wyników 559

Protokół 559

Pobieranie danych od użytkownika: strumień asynchroniczny 561

Wysyłanie pakietów 563

Pobieranie pakietów: więcej strumieni asynchronicznych 564

Funkcja main klienta 566

Funkcja main serwera 567

Obsługa połączeń z klientami: asynchroniczne muteksy 568

Tablica grup: muteksy synchroniczne 571

Grupy: kanały rozgłoszeniowe paczki tokio 572

Podstawowe wartości future i wykonawcy: kiedy warto ponownie sprawdzić wartość future? 575

Wywoływanie funkcji aktywujących: spawn\_blocking 577

Implementacja funkcji block\_on 579

Typ Pin i jego stosowanie 581

Dwa etapy życia danych typ future 581

Przypięte wskaźniki 585

Zestaw metod Unpin 587

Kiedy warto stosować kod asynchroniczny? 588

21. Makra 591

Podstawy 592

Rozwijanie makra 593

Niezamierzone skutki 595

Powtórzenia 596

Makra wbudowane 598

Debugowanie makr 600

Pisanie makra json! 601

Typy składników 602

Makra a rekurencja 605

Makra i zestawy metod 606

Zakres i higiena 608

Import i eksport makr 610

Unikanie błędów składniowych w procesie dopasowywania 612

macro\_rules! i co dalej? 613

22. Kod niebezpieczny 615

Dlaczego niebezpieczny? 616

Bloki unsafe 617

Przykład: wydajny typ łańcucha znaków ASCII 619

Funkcje unsafe 621

Kod niebezpieczny czy funkcja niebezpieczna? 623

Działanie niezdefiniowane 623

Zestawy metod unsafe 626

Wskaźniki niechronione 628

Bezpieczne tworzenie dereferencji wskaźników niechronionych 631

Przykład: RefWithFlag 632

Wskaźniki dopuszczające wartość pustą 634

Rozmiary i rozmieszczanie typów 634

Operacje arytmetyczne na wskaźnikach 635

Wchodzenie do pamięci i wychodzenie z pamięci 636

Przykład: GapBuffer 640

Bezpieczeństwo błędów paniki w kodzie niebezpiecznym 646

Ponowna interpretacja pamięci z użyciem unii 647

Dopasowywanie unii 650

Pożyczanie unii 650

23. Funkcje obce 651

Wyszukiwanie wspólnych reprezentacji danych 652

Deklarowanie obcych funkcji i zmiennych 655

Korzystanie z funkcji i bibliotek 657

Interfejs niechroniony dla biblioteki libgit2 660

Interfejs bezpieczny dla biblioteki libgit2 666

Podsumowanie 676

Skorowidz 677