
Spis treści

Wstęp	15
1. Dlaczego Rust?	19
Bezpieczeństwo typów	21
2. Pierwsze spotkanie z Rustem	25
Pobranie i instalacja Rusta	25
Prosta funkcja	28
Pisanie i uruchamianie testów	29
Obsługa argumentów wiersza poleceń	30
Prosty serwer web	34
Programowanie współbieżne	41
Czym jest zbiór Mandelbrota?	42
Parsowanie argumentów wiersza poleceń	46
Odwzorowanie pikseli na liczby zespolone	48
Rysowanie zbioru	50
Zapis obrazu do pliku	51
Program Mandelbrota działający współbieżnie	53
Uruchomienie programu	57
Przezroczyste bezpieczeństwo	57
3. Typy proste	59
Typy maszynowe	62
Typy całkowite	62
Typy zmiennoprzecinkowe	65
Typ logiczny	67
Typ znakowy	67

Krotki	69
Typy wskaźnikowe	71
Referencje	71
Pudełka	72
Wskaźniki niechronione	72
Tablice, wektory i podzbiory	72
Tablice	73
Wektory	74
Dodawanie pojedynczych elementów do wektora	77
Podzbiory	77
Typ String	79
Literały łańcuchowe	79
Łańcuchy bajtów	80
Łańcuchy znaków w pamięci	80
Typ String	82
Podstawowe cechy typu String	83
Inne typy znakowe	83
Co dalej?	84
4. Reguła własności	85
Reguła własności	87
Przeniesienie własności	91
Więcej operacji związanych z przeniesieniem własności	96
Przeniesienie własności a przepływ sterowania	97
Przeniesienie własności a struktury indeksowane	98
Typy kopiowalne	100
Rc i Arc: własność współdzielona	103
5. Referencje	107
Referencje jako wartości	111
Referencje Rusta kontra referencje C++	111
Referencje a operacja przypisania	112
Referencje do referencji	112
Porównywanie referencji	113
Referencje nigdy nie są puste	114
Referencje do wyrażeń	114
Referencje do podzbiorów i zestawów metod	115
Bezpieczeństwo referencji	115
Referencja do zmiennej lokalnej	115
Parametry w postaci referencji	118
Referencje jako argumenty	120

Referencja jako wartość zwracana	121
Struktura zawierająca referencje	122
Odrębny cykl życia	124
Pomijanie parametrów cyklu życia	125
Referencje współdzielone kontra mutowalne	127
Walka ze sztormem na morzu obiektów	134
6. Wyrażenia	137
Język wyrażeń	137
Bloki kodu i średniki	138
Deklaracje	140
if i match	141
if let	143
Pętle	144
Wyrażenie return	146
Analiza przepływu sterowania	146
Wywołania funkcji i metod	148
Pola i elementy	149
Operatory referencji	150
Operatory arytmetyczne, bitowe, porównania i logiczne	150
Przypisanie	151
Rzutowanie typów	152
Domknięcia	153
Priorytety operatorów	153
Co dalej?	156
7. Obsługa błędów	157
Błąd panic	157
Odwinięcie stosu	158
Przerywanie procesu	159
Typ Result	160
Przechwytywanie błędów	160
Alias typu Result	161
Wyświetlanie informacji o błędach	162
Propagacja błędów	163
Jednoczesna obsługa błędów różnych typów	164
Błędy, które nie powinny się zdarzyć	166
Ignorowanie błędów	167
Obsługa błędów w funkcji main()	167
Definiowanie własnego typu błędu	168
Co daje nam typ Result?	169

8. Paczki i moduły	171
Paczki	171
Profile budowania	174
Moduły	174
Umieszczanie modułów w oddzielnych plikach	175
Ścieżki i importy	177
Standardowe preludium	179
Podstawowe składniki modułów	179
Zmiana programu w bibliotekę	181
Katalog src/bin	183
Atrybuty	184
Testy i dokumentacja	186
Testy integracyjne	188
Dokumentacja	189
Doc-testy	191
Definiowanie zależności	193
Wersje	194
Cargo.lock	195
Publikowanie paczek na stronie crates.io	196
Obszary robocze	198
Więcej fajnych rzeczy	199
9. Struktury	201
Struktury z polami nazywanymi	201
Struktury z polami numerowanymi	204
Struktury puste	204
Reprezentacja struktur w pamięci	205
Definiowanie metod w bloku impl	206
Struktury generyczne	209
Struktury z parametrem cyklu życia	210
Dziedziczenie standardowych zestawów metod	211
Zmienność wewnętrzna	212
10. Typy wyliczeniowe i wzorce	217
Typy wyliczeniowe	218
Typy wyliczeniowe zawierające dane	220
Typ wyliczeniowy w pamięci	221
Większe struktury danych stosujące typy wyliczeniowe	222
Generyczne typy wyliczeniowe	223

Wzorce	226
Literały, zmienne i symbole wieloznaczne	228
Wzorce w postaci krotki lub struktury	230
Wzorce z referencjami	231
Dopasowanie do wielu wartości	233
Warunki dodatkowe	234
Wzorce @	235
Gdzie używamy wzorców	235
Wypełnianie drzewa binarnego	236
Podsumowanie	238
11. Zestawy metod (interfejsy) i typy generyczne	239
Stosowanie zestawów metod	241
Obiekt implementujący zestaw metod	242
Struktura obiektu implementującego	243
Funkcje generyczne	244
Na co się zdecydować?	247
Definiowanie i implementacja zestawów metod	249
Metody domyślne	250
Implementacja zestawów metod dla istniejących już typów	251
Zestaw metod a słowo kluczowe Self	252
Rozszerzanie zestawu metod (dziedziczenie)	254
Metody statyczne	254
W pełni kwalifikowana nazwa metody	255
Zestawy metod definiujące relacje między typami	257
Typy powiązane	257
Generyczny zestaw metod (czyli jak działa przeciążanie operatorów)	260
Zaprzyjaźnione zestawy metod (czyli jak działa generator liczb pseudolosowych)	261
Inżynieria wsteczna ograniczeń	263
Wnioski	266
12. Przeciążanie operatorów	267
Operatory arytmetyczne i bitowe	268
Operatory jednoargumentowe	270
Operatory dwuargumentowe	271
Operatory przypisania złożonego	272
Test równości	273
Porównania szeregujące	276
Interfejsy Index i IndexMut	278
Inne operatory	281

13. Interfejsy narzędziowe	283
Drop	284
Sized	287
Clone	289
Copy	290
Deref i DerefMut	291
Default	294
AsRef i AsMut	296
Borrow i BorrowMut	297
From i Into	299
ToOwned	301
Borrow i ToOwned w działaniu	302
14. Domknięcia	305
Przechwytywanie zmiennych	306
Domknięcia, które pożyczają wartość	307
Domknięcia, które przejmują własność	308
Typy funkcji i domknięć	309
Domknięcia a wydajność	311
Domknięcia a bezpieczeństwo	313
Domknięcia, które zabijają	313
FnOnce	314
FnMut	315
Funkcje zwrotne	317
Skuteczne korzystanie z domknięć	320
15. Iteratory	323
Iterator i IntoIterator	324
Tworzenie iteratorów	326
Metody iter i iter_mut	326
Implementacje interfejsu IntoIterator	326
Metoda drain	328
Inne źródła iteratorów	329
Adaptery	330
map i filter	330
filter_map i flat_map	333
scan	335
take i take_while	335
skip i skip_while	336
peekable	337
fuse	338

Iteratory obustronne i rev	339
inspect	340
chain	341
enumerate	341
zip	342
by_ref	342
cloned	344
cycle	344
Konsumowanie iteratorów	345
Proste agregaty: count, sum i product	345
max i min	345
max_by i min_by	346
max_by_key i min_by_key	346
Porównywanie sekwencji elementów	347
any i all	348
position, rposition oraz ExactSizeIterator	348
fold	349
nth	349
last	350
find	350
Tworzenie kolekcji: collect i FromIterator	350
Zestaw metod Extend	352
partition	353
Implementacja własnych iteratorów	354
16. Kolekcje	359
Przegląd kolekcji	360
Vec<T>	361
Dostęp do elementów	362
Iteracja	363
Zwiększanie i zmniejszanie wielkości wektora	364
Łączenie	367
Podział	367
Zamiana	369
Sortowanie i wyszukiwanie	370
Porównywanie podzbiorów	371
Elementy losowe	372
Reguły zapobiegające konfliktom w czasie iteracji	372
VecDeque<T>	373
LinkedList<T>	375

BinaryHeap<T>	376
HashMap<K, V> i BTreeMap<K, V>	377
Entry	380
Iterowanie map	381
HashSet<T> i BTreeSet<T>	382
Iteracja zbioru	383
Kiedy równe wartości są różne	383
Operacje dotyczące całego zbioru	383
Haszowanie	385
Niestandardowe algorytmy haszujące	386
Kolekcje standardowe i co dalej?	387

17. Tekst i łańcuchy znaków 389

Podstawy Unicode	389
ASCII, Latin-1 i Unicode	390
UTF-8	390
Kierunek tekstu	392
Znaki (typ char)	392
Klasyfikacja znaków	392
Obsługa cyfr	393
Zmiana wielkości liter	394
Konwersja znaku do i z liczby całkowitej	394
Typy String i str	395
Tworzenie łańcuchów znaków	396
Prosta inspekcja	396
Dołączanie i wstawianie tekstu	397
Usuwanie tekstu	398
Konwencje nazewnicze dotyczące wyszukiwania i iterowania	399
Wyszukiwanie tekstu i wzorce	399
Wyszukiwanie i zamiana	400
Iterowanie tekstu	401
Obcinanie	403
Zmiana wielkości liter w łańcuchach	403
Konwersja tekstu do wartości innego typu	404
Konwersja wartości innego typu do tekstu	404
Tworzenie referencji innego typu	405
Tekst jako UTF-8	406
Tworzenie tekstu na podstawie danych UTF-8	406
Alokacja warunkowa	407
Łańcuchy znaków jako kolekcje generyczne	409

Formatowanie wartości	410
Formatowanie tekstu	411
Formatowanie liczb	412
Formatowanie innych typów	414
Formatowanie wartości z myślą o debugowaniu	415
Formatowanie i debugowanie wskaźników	416
Wiązanie argumentów za pomocą indeksu i nazwy	416
Dynamiczne definiowanie długości i precyzji	417
Formatowanie własnych typów	417
Użycie języka formatowania we własnym kodzie	419
Wyrażenia regularne	421
Podstawowe użycie wyrażeń regularnych	421
Wyrażenia regularne w trybie leniwym	422
Normalizacja	423
Rodzaje normalizacji	424
Biblioteka unicode-normalization	425
18. Operacje wejścia/wyjścia	427
Obiekty typu reader i writer	428
Obiekty typu reader	429
Buforowany obiekt typu reader	431
Przeglądanie tekstu	432
Pobieranie tekstu	434
Obiekty typu writer	435
Pliki	436
Wyszukiwanie	437
Inne rodzaje obiektów reader i writer	437
Dane binarne, kompresja i serializacja	439
Pliki i katalogi	440
OsStr i Path	440
Metody typów Path i PathBuf	442
Funkcje dostępu do systemu plików	443
Odczyt zawartości katalogu	444
Funkcje bezpośrednio związane z platformą	446
Obsługa sieci	447
19. Programowanie współbieżne	451
Podział i łączenie wątków	453
spawn i join	454
Obsługa błędów w różnych wątkach	456
Współdzielenie niemutowalnych danych przez różne wątki	457

Rayon	459
Zbiór Mandelbrota raz jeszcze	461
Kanały	463
Wysyłanie wartości	464
Odczyt wartości	467
Uruchomienie potoku	468
Cechy kanałów i ich wydajność	470
Bezpieczeństwo wątków: Send i Sync	472
Współpraca iteratora i kanału	474
Potoki i co dalej?	475
Stan współdzielony mutowalny	476
Czym jest muteks?	476
Mutex<T>	478
mut i Mutex	480
Dlaczego Mutex to nie zawsze dobry pomysł?	480
Zakleszczenie (deadlock)	481
Zatruty muteks	482
Kanały z wieloma nadawcami stosujące muteksy	482
Blokady odczytu/zapisu (RwLock<T>)	483
Zmienne warunkowe (Condvar)	485
Typy atomowe	485
Zmienne globalne	487
Rust i pisanie programów wielowątkowych	489

20. Makra 491

Podstawy	492
Rozwijanie makra	493
Niezamierzone skutki	495
Powtórzenia	496
Makra wbudowane	498
Debugowanie makr	499
Makro json!	500
Typy składników	501
Makra a rekurencja	504
Makra i zestawy metod	505
Zakres i higiena	507
Import i eksport makr	509
Unikanie błędów składniowych w procesie dopasowywania	511
macro_rules! i co dalej?	512

21. Kod niebezpieczny513

 Dlaczego niebezpieczny? 514

 Bloki unsafe 515

 Przykład: skuteczny typ łańcucha znaków ASCII 516

 Funkcje unsafe 518

 Kod niebezpieczny czy funkcja niebezpieczna? 520

 Niezdefiniowane zachowanie 521

 Zestawy metod unsafe 523

 Wskaźniki niechronione 525

 Bezpieczne tworzenie dereferencji wskaźników niechronionych 528

 Przykład: RefWithFlag 529

 Wskaźniki dopuszczające wartość pustą 531

 Rozmiary i rozmieszczanie typów 531

 Operacje arytmetyczne na wskaźnikach 532

 Wchodzenie do pamięci i wychodzenie z pamięci 534

 Przykład: GapBuffer 537

 Bezpieczeństwo błędów paniki w kodzie niebezpiecznym 543

 Funkcje obce: wywoływanie kodu C i C++ w środowisku Rusta 544

 Wyszukiwanie wspólnych reprezentacji danych 544

 Deklarowanie obcych funkcji i zmiennych 547

 Korzystanie z funkcji i bibliotek 549

 Interfejs niechroniony dla biblioteki libgit2 552

 Interfejs bezpieczny dla biblioteki libgit2 558

 Podsumowanie 568

Skorowidz569