Spis treści

Przedmowa 9
O książce 11
O autorze 15

Rozdział 1. Wprowadzenie do programowania funkcyjnego 17

1.1. Co to jest programowanie funkcyjne? 18
1.1.1. Związek z programowaniem obiektowym 19
1.1.2. Praktyczny przykład porównania programowania imperatywnego i deklaratywnego 20
1.2. Funkcje czyste 24
1.2.1. Unikanie stanu mutowalnego 27
1.3. Myślenie w sposób funkcjonalny 29
1.4. Korzyści wynikające z programowania funkcyjnego 31
1.4.1. Zwięzłość i czytelność kodu 32
1.4.2. Współbieżność i synchronizacja 33
1.4.3. Ciągła optymalizacja 33
1.5. Przekształcanie C++ w funkcyjny język programowania 34
1.6. Czego nauczysz się w trakcie czytania tej książki? 36
Podsumowanie 36

Rozdział 2. Pierwsze kroki z programowaniem funkcyjnym 39

2.1. Funkcje używające innych funkcji? 40
2.2. Przykłady z biblioteki STL 42
2.2.1. Obliczanie średnich 42
2.2.2. Zwijanie 45
2.2.3. Przycinanie łańcucha 48
2.2.4. Partycjonowanie kolekcji na podstawie predykatu 50
2.2.5. Filtrowanie i transformacja 52
2.3. Problemy ze składaniem algorytmów STL 53
2.4. Tworzenie własnych funkcji wyższego rzędu 55
2.4.1. Otrzymywanie funkcji w postaci argumentów 55
2.4.2. Implementacja z pętlami 56
2.4.3. Rekurencja oraz optymalizacja przez rekurencję ogonową 57
2.4.4. Implementacja przy użyciu zwijania 61
Podsumowanie 62

Rozdział 3. Obiekty funkcyjne 63

3.1. Funkcje i obiekty funkcyjne 64
3.1.1. Automatyczna dedukcja typu zwracanego 64
3.1.2. Wskaźniki do funkcji 67
3.1.3. Przeciążanie operatora wywołania 68
3.1.4. Tworzenie generycznych obiektów funkcyjnych 70
3.2. Wyrażenia lambda i domknięcia 73
3.2.1. Składnia wyrażenia lambda 74
3.2.2. Co się kryje wewnątrz wyrażenia lambda? 75
3.2.3. Tworzenie dowolnych zmiennych składowych w wyrażeniach lambda 77
3.2.4. Uogólnione wyrażenia lambda 79
3.3. Tworzenie obiektów funkcyjnych, które są jeszcze bardziej zwięzłe niż wyrażenia lambda 80
3.3.1. Obiekty funkcyjne operatorów w bibliotece STL 83
3.3.2. Obiekty funkcyjne operatorów w innych bibliotekach 84
3.4. Opakowywanie obiektów funkcyjnych przy użyciu std::function 86
Podsumowanie 88

Rozdział 4. Tworzenie nowych funkcji na podstawie istniejących 89

4.1. Częściowe stosowanie funkcji 90
4.1.1. Ogólny sposób zamiany funkcji dwuargumentowych na jednoargumentowe 92
4.1.2. Użycie std::bind do wiązania wartości z określonymi argumentami funkcji 95
4.1.3. Zamienianie ze sobą argumentów funkcji dwuargumentowej 97
4.1.4. Użycie std::bind z funkcjami mającymi więcej argumentów 98
4.1.5. Użycie wyrażeń lambda jako alternatywy dla std::bind 101
4.2. Rozwijanie: inny sposób podejścia do funkcji 103
4.2.1. Rozwijanie funkcji w prostszy sposób 104
4.2.2. Użycie rozwijania podczas dostępu do bazy danych 106
4.2.3. Rozwijanie a częściowe stosowanie funkcji 109
4.3. Złożenie funkcji 110
4.4. Podnoszenie funkcji — kolejne podejście 113
4.4.1. Odwracanie elementów par w kolekcji 116
Podsumowanie 117

Rozdział 5. Czystość: unikanie stanu mutowalnego 119

5.1. Problemy ze stanem mutowalnym 120
5.2. Funkcje czyste i przejrzystość referencyjna 122
5.3. Programowanie bez efektów ubocznych 125
5.4. Stan mutowalny i niemutowalny w środowisku współbieżnym 129
5.5. Duże znaczenie kwalifikatora const 132
5.5.1. Logiczna i wewnętrzna zgodność z deklaracją const 134
5.5.2. Optymalizacja funkcji składowych dla zmiennych tymczasowych 136
5.5.3. Pułapki związane z użyciem słowa kluczowego const 138
Podsumowanie 140

Rozdział 6. Wartościowanie leniwe 141

6.1. Wartościowanie leniwe w C++ 142
6.2. Wartościowanie leniwe jako technika optymalizacyjna 145
6.2.1. Sortowanie kolekcji przy wykorzystaniu wartościowania leniwego 145
6.2.2. Wyświetlanie elementów w interfejsach użytkownika 147
6.2.3. Przycinanie drzew rekurencyjnych przez buforowanie wyników funkcji 148
6.2.4. Programowanie dynamiczne jako forma wartościowania leniwego 150
6.3. Uogólnione zapamiętywanie 152
6.4. Szablony wyrażeń i leniwe łączenie łańcuchów 155
6.4.1. Czystość i szablony wyrażeń 159
Podsumowanie 160

Rozdział 7. Zakresy 163

7.1. Wprowadzenie do zakresów 165
7.2. Tworzenie widoków danych tylko do odczytu 166
7.2.1. Użycie funkcji filter z zakresami 166
7.2.2. Użycie funkcji transform z zakresami 167
7.2.3. Wartościowanie leniwe wartości zakresów 168
7.3. Mutowalność zmiennych w zakresach 170
7.4. Używanie zakresów ograniczonych i nieskończonych 172
7.4.1. Użycie zakresów ograniczonych do optymalizacji obsługi zakresów wejściowych 172
7.4.2. Tworzenie zakresów nieskończonych przy użyciu wartowników 173
7.5. Używanie zakresów do obliczania częstości pojawiania się słów 175
Podsumowanie 178

Rozdział 8. Funkcyjne struktury danych 179

8.1. Niemutowalne listy łączone 180
8.1.1. Dodawanie i usuwanie elementów z początku listy 180
8.1.2. Dodawanie i usuwanie elementów z końca listy 181
8.1.3. Dodawanie i usuwanie elementów z wnętrza listy 182
8.1.4. Zarządzanie pamięcią 183
8.2. Niemutowalne struktury danych podobne do wektorów 185
8.2.1. Wyszukiwanie elementu w drzewie trie 187
8.2.2. Dołączanie elementów w drzewie trie 189
8.2.3. Aktualizacja elementów w drzewie trie 191
8.2.4. Usuwanie elementów z końca drzewa trie 192
8.2.5. Inne operacje i całkowita wydajność drzewa trie 192
Podsumowanie 193

Rozdział 9. Algebraiczne typy danych i dopasowywanie do wzorców 195

9.1. Algebraiczne typy danych 196
9.1.1. Typy sumy uzyskiwane poprzez dziedziczenie 197
9.1.2. Typy sumy uzyskiwane dzięki uniom i typowi std::variant 200
9.1.3. Implementacja określonych stanów 204
9.1.4. Szczególny typ sumy: wartości opcjonalne 205
9.1.5. Użycie typów sumy w celu obsługi błędów 207

9.2. Modelowanie dziedziny za pomocą algebraicznych typów danych 212
9.2.1. Rozwiązanie najprostsze, które czasami jest niewystarczające 213
9.2.2. Rozwiązanie bardziej zaawansowane: podejście zstępujące 214
9.3. Lepsza obsługa algebraicznych typów danych za pomocą dopasowywania do wzorców 215
9.4. Zaawansowane dopasowywanie do wzorców za pomocą biblioteki Mach7 218
Podsumowanie 219

Rozdział 10. Monady 221

10.1. Funktory pochodzące od nie Twojego rodzica 222
10.1.1. Obsługa wartości opcjonalnych 223
10.2. Monady: więcej możliwości dla funktorów 226
10.3. Proste przykłady 229
10.4. Składane zakresy i monady 231
10.5. Obsługa błędów 234
10.5.1. Typ std::optional<T> jako monada 234
10.5.2. Typ expected<T, E> jako monada 236
10.5.3. Monada Try 237
10.6. Obsługa stanu przy użyciu monad 238
10.7. Monada współbieżnościowa i kontynuacyjna 240
10.7.1. Typ future jako monada 242
10.7.2. Implementacja wartości przyszłych 244
10.8. Składanie monad 245
Podsumowanie 247

Rozdział 11. Metaprogramowanie szablonów 249

11.1. Zarządzanie typami w czasie kompilacji 250
11.1.1. Debugowanie typów dedukowanych 252
11.1.2. Dopasowywanie do wzorców podczas kompilacji 254
11.1.3. Udostępnianie metainformacji o typach 257
11.2. Sprawdzanie właściwości typu w czasie kompilacji 258
11.3. Tworzenie funkcji rozwiniętych 261
11.3.1. Wywoływanie wszystkich obiektów wywoływalnych 263
11.4. Tworzenie języka dziedzinowego 265
Podsumowanie 271

Rozdział 12. Projektowanie funkcyjne systemów współbieżnych 273

12.1. Model aktora: myślenie komponentowe 274
12.2. Tworzenie prostego źródła wiadomości 278
12.3. Modelowanie strumieni reaktywnych w postaci monad 281
12.3.1. Tworzenie ujścia w celu odbierania wiadomości 283
12.3.2. Transformowanie strumieni reaktywnych 286
12.3.3. Tworzenie strumienia z określonymi wartościami 288
12.3.4. Łączenie strumienia strumieni 289
12.4. Filtrowanie strumieni reaktywnych 290
12.5. Obsługa błędów w strumieniach reaktywnych 291

12.6. Odpowiadanie klientowi 293
12.7. Tworzenie aktorów ze stanem mutowalnym 297
12.8. Tworzenie systemów rozproszonych z użyciem aktorów 298
Podsumowanie 299

Rozdział 13. Testowanie i debugowanie *301*

13.1. Czy program, który się kompiluje, jest poprawny? 302
13.2. Testy jednostkowe i funkcje czyste 304
13.3. Testy generowane automatycznie 305
13.3.1. Generowanie przypadków testowych *306*
13.3.2. Testowanie oparte na właściwościach *307*
13.3.3. Testy porównawcze *309*
13.4. Testowanie systemów współbieżnych opartych na monadach 311
Podsumowanie 314

Skorowidz *315*