

## Spis treści

Przedmowa 17

Wprowadzenie 19

Podziękowania 21

O książce 23

O autorze 27

CZĘŚĆ 1. KONTEKST JĘZYKA C# 29

Rozdział 1. Przetwarzają najbystrzejsi 31

1.1. Ewoluuujący język 31

1.1.1. System typów pomocny w dużej i małej skali 32

1.1.2. Jeszcze bardziej zwięzły kod 34

1.1.3. Prosty dostęp do danych w technologii LINQ 38

1.1.4. Asynchroniczność 38

1.1.5. Równowaga między wydajnością a złożonością 40

1.1.6. Przyspieszona ewolucja - używanie podwersji 41

1.2. Ewoluuująca platforma 42

1.3. Ewoluuująca społeczność 43

1.4. Ewoluuująca książka 44

1.4.1. Wyjaśnienia na różnym poziomie 45

1.4.2. Przykłady, w których wykorzystano projekt Noda Time 45

1.4.3. Terminologia 46

Podsumowanie 47

CZĘŚĆ 2. C# 2 - 5 49

Rozdział 2. C# 2 51

- 2.1. Typy generyczne 52
  - 2.1.1. Wprowadzenie z użyciem przykładu - kolekcje przed wprowadzeniem typów generycznych 52
  - 2.1.2. Typy generyczne ratują sytuację 55
  - 2.1.3. Jakie elementy mogą być generyczne? 59
  - 2.1.4. Wnioskowanie typu argumentów określających typ w metodach 60
  - 2.1.5. Ograniczenia typów 62
  - 2.1.6. Operatory default i typeof 64
  - 2.1.7. Inicjowanie typów generycznych i ich stan 67
- 2.2. Typy bezpośrednie przyjmujące wartość null 69
  - 2.2.1. Cel - reprezentowanie braku informacji 69
  - 2.2.2. Wsparcie w środowisku CLR i platformie - struktura Nullable 70
  - 2.2.3. Obsługa dostępna w języku 74
- 2.3. Uprozczone tworzenie delegatów 80
  - 2.3.1. Konwersje grupy metod 81
  - 2.3.2. Metody anonimowe 81
  - 2.3.3. Zgodność delegatów 83
- 2.4. Iteratory 84
  - 2.4.1. Wprowadzenie do iteratorów 85
  - 2.4.2. Leniwe wykonywanie 86
  - 2.4.3. Przetwarzanie instrukcji yield 87
  - 2.4.4. Znaczenie leniwego wykonywania 88
  - 2.4.5. Przetwarzanie bloków finally 89
  - 2.4.6. Znaczenie obsługi bloku finally 92
  - 2.4.7. Zarys implementacji 93
- 2.5. Mniej istotne mechanizmy 98
  - 2.5.1. Typy częściowe 98
  - 2.5.2. Klasy statyczne 100

2.5.3. Inny poziom dostępu do getterów i setterów właściwości 101

2.5.4. Aliasy przestrzeni nazw 101

2.5.5. Dyrektywy pragma 103

2.5.6. Bufory o stałej wielkości 104

2.5.7. Atrybut InternalsVisibleTo 105

Podsumowanie 106

Rozdział 3. C# 3 - technologia LINQ i wszystko, co z nią związane 107

3.1. Automatycznie implementowane właściwości 108

3.2. Niejawne określanie typów 108

3.2.1. Terminologia związana z typami 109

3.2.2. Zmienne lokalne z typowaniem niejawnym (var) 110

3.2.3. Tablice z niejawnym typowaniem 112

3.3. Inicjalizatory obiektów i kolekcji 113

3.3.1. Wprowadzenie do inicjalizatorów obiektów i kolekcji 113

3.3.2. Inicjalizatory obiektów 115

3.3.3. Inicjalizatory kolekcji 116

3.3.4. Zalety inicjowania za pomocą jednego wyrażenia 118

3.4. Typy anonimowe 118

3.4.1. Składnia i podstawy działania 119

3.4.2. Typ generowany przez kompilator 121

3.4.3. Ograniczenia 122

3.5. Wyrażenia lambda 123

3.5.1. Składnia wyrażen lambda 124

3.5.2. Przechwytywanie zmiennych 126

3.5.3. Drzewa wyrażen 133

3.6. Metody rozszerzające 135

3.6.1. Deklarowanie metody rozszerzającej 136

3.6.2. Wywoływanie metod rozszerzających 136

3.6.3. Łączenie wywołań metod w łańcuch 138

3.7. Wyrażenia reprezentujące zapytania 140

3.7.1. Wyrażenia reprezentujące zapytania są przekształcane z kodu C# na inny kod C# 140

3.7.2. Zmienne zakresu i identyfikatory przezroczyste 141

3.7.3. Kiedy stosować poszczególne składnie w LINQ? 142

3.8. Efekt końcowy - technologia LINQ 143

Podsumowanie 144

Rozdział 4. Zwiększanie współdziałania z innymi technologiami 145

4.1. Typowanie dynamiczne 146

4.1.1. Wprowadzenie do typowania dynamicznego 146

4.1.2. Dynamiczne operacje poza mechanizmem refleksji 151

4.1.3. Krótkie spojrzenie na zaplecze 156

4.1.4. Ograniczenia i niespodzianki związane z typowaniem dynamicznym 160

4.1.5. Sugestie dotyczące użytkowania 164

4.2. Parametry opcjonalne i argumenty nazwane 166

4.2.1. Parametry o wartościach domyślnych i argumenty z nazwami 167

4.2.2. Określanie znaczenia wywołań metody 168

4.2.3. Wpływ na wersjonowanie 170

4.3. Usprawnienia w zakresie współdziałania z technologią COM 172

4.3.1. Konsolidacja podzespołów PIA 172

4.3.2. Parametry opcjonalne w COM 174

4.3.3. Indeksery nazwane 175

4.4. Wariacja generyczna 176

4.4.1. Proste przykłady zastosowania wariacji 176

4.4.2. Składnia wariacji w deklaracjach interfejsów i delegatów 178

4.4.3. Ograniczenia dotyczące wariacji 179

4.4.4. Wariacja generyczna w praktyce 181

Podsumowanie 183

Rozdział 5. Pisanie kodu asynchronicznego 185

5.1. Wprowadzenie do funkcji asynchronicznych 187

5.1.1. Bliskie spotkania asynchronicznego stopnia 187

5.1.2. Analiza pierwszego przykładu 189

5.2. Myślenie o asynchroniczności 190

5.2.1. Podstawy asynchronicznego wykonywania kodu 191

5.2.2. Konteksty synchronizacji 192

5.2.3. Model działania metod asynchronicznych 193

5.3. Deklaracje metod asynchronicznych 195

5.3.1. Typy wartości zwracanych przez metody asynchroniczne 196

5.3.2. Parametry metod asynchronicznych 197

5.4. Wyrażenia await 197

5.4.1. Wzorzec awaitable 198

5.4.2. Ograniczenia dotyczące wyrażen await 200

5.5. Opakowywanie zwracanych wartości 202

5.6. Przepływ sterowania w metodzie asynchronicznej 203

5.6.1. Na co kod oczekuje i kiedy? 203

5.6.2. Przetwarzanie wyrażen await 205

5.6.3. Używanie składowych zgodnych ze wzorcem awaitable 208

5.6.4. Wypakowywanie wyjątków 208

5.6.5. Ukończenie pracy metody 211

5.7. Asynchroniczne funkcje anonimowe 216

5.8. Niestandardowe typy zadań w C# 7 217

5.8.1. Typ przydatny w 99,9% przypadków - ValueTask 217

5.8.2. 0,1% sytuacji - tworzenie własnych niestandardowych typów zadań 220

5.9. Asynchroniczne metody main w C# 7.1 222

5.10. Wskazówki dotyczące korzystania z asynchroniczności 223

5.10.1. Jeśli jest to akceptowalne, używaj ConfigureAwait, aby nie przechwytywać kontekstu 223

5.10.2. Włączanie przetwarzania równoległego dzięki uruchomieniu wielu niezależnych zadań 225

5.10.3. Unikaj łączenia kodu synchronicznego z asynchronicznym 226

5.10.4. Wszędzie, gdzie to możliwe, zezwalaj na anulowanie operacji 226

5.10.5. Testowanie kodu asynchronicznego 227

Podsumowanie 228

Rozdział 6. Implementacja asynchroniczności 229

6.1. Struktura wygenerowanego kodu 231

6.1.1. Metoda kontrolna - przygotowania i pierwszy krok 233

6.1.2. Struktura maszyny stanowej 235

6.1.3. Metoda MoveNext() (ogólny opis) 238

6.1.4. Metoda SetStateMachine i taniec z opakowywaniem maszyny stanowej 240

6.2. Prosta implementacja metody MoveNext() 241

6.2.1. Kompletny konkretny przykład 241

6.2.2. Ogólna struktura metody MoveNext() 243

6.2.3. Zbliżenie na wyrażenia await 245

6.3. Jak przepływ sterowania wpływa na metodę MoveNext()? 247

6.3.1. Przepływ sterowania między wyrażeniami await jest prosty 247

6.3.2. Oczekiwanie w pętli 248

6.3.3. Oczekiwanie w bloku try/finally 250

6.4. Kontekst wykonania i przekazywanie kontekstu 253

6.5. Jeszcze o niestandardowych typach zadań 254

Podsumowanie 255

Rozdział 7. Dodatkowe mechanizmy z C# 5 257

7.1. Przechwytywanie zmiennych w pętlach foreach 257

7.2. Atrybuty z informacjami o jednostce wywołującej 259

7.2.1. Podstawowe działanie 259

7.2.2. Rejestrowanie informacji w dzienniku 261

7.2.3. Upraszczenie implementacji interfejsu INotifyPropertyChanged 261

7.2.4. Przypadki brzegowe dotyczące atrybutów z informacjami o jednostce wywołującej 263

7.2.5. Używanie atrybutów z informacjami o jednostce wywołującej w starszych wersjach platformy .NET 269

Podsumowanie 270

CZĘŚĆ 3. C# 6 271

Rozdział 8. Odchudzone właściwości i składowe z ciałem w postaci wyrażenia 273

8.1. Krótka historia właściwości 274

8.2. Usprawnienia automatycznie implementowanych właściwości 276

8.2.1. Automatycznie implementowane właściwości przeznaczone tylko do odczytu 276

8.2.2. Inicjalizowanie automatycznie implementowanych właściwości 277

8.2.3. Automatycznie implementowane właściwości w strukturach 279

8.3. Składowe z ciałem w postaci wyrażenia 281

8.3.1. Jeszcze prostsze obliczanie właściwości tylko do odczytu 281

8.3.2. Metody, indeksery i operatory z ciałem w postaci wyrażenia 284

8.3.3. Ograniczenia dotyczące składowych z ciałem w postaci wyrażenia w C# 6 286

8.3.4. Wskazówki dotyczące używania składowych z ciałem w postaci wyrażenia 287

Podsumowanie 290

Rozdział 9. Mechanizmy związane z łańcuchami znaków 291

9.1. Przypomnienie technik formatowania łańcuchów znaków w .NET 292

9.1.1. Proste formatowanie łańcuchów znaków 292

9.1.2. Niestandardowe formatowanie z użyciem łańcuchów znaków formatowania 293

9.1.3. Lokalizacja 295

9.2. Wprowadzenie do literałów tekstowych z interpolacją 299

9.2.1. Prosta interpolacja 299

9.2.2. Łańcuchy znaków formatowania w literałach tekstowych z interpolacją	300
9.2.3. Dosłowne literały tekstowe z interpolacją	300
9.2.4. Obsługa literałów tekstowych z interpolacją przez kompilator (część 1.)	302
9.3. Lokalizacja z użyciem typu FormattableString	302
9.3.1. Obsługa literałów tekstowych z interpolacją przez kompilator (część 2.)	303
9.3.2. Formatowanie obiektu typu FormattableString z użyciem określonych ustawień regionalnych	305
9.3.3. Inne zastosowania typu FormattableString	306
9.3.4. Używanie typu FormattableString w starszych wersjach platformy .NET	310
9.4. Zastosowania, wskazówki i ograniczenia	311
9.4.1. Programiści i maszyny, ale raczej nie użytkownicy końcowi	311
9.4.2. Sztywne ograniczenia literałów tekstowych z interpolacją	314
9.4.3. Kiedy można stosować literały tekstowe z interpolacją, ale nie należy tego robić?	315
9.5. Dostęp do identyfikatorów za pomocą operatora nameof	317
9.5.1. Pierwsze przykłady stosowania operatora nameof	317
9.5.2. Standardowe zastosowania operatora nameof	319
9.5.3. Sztuczki i kruczki związane z używaniem operatora nameof	322
Podsumowanie	325
Rozdział 10. Szwedzki stół z funkcjami do pisania zwięzłego kodu	325
10.1. Dyrektywa using static	325
10.1.1. Importowanie składowych statycznych	326
10.1.2. Metody rozszerzające i dyrektywa using static	329
10.2. Usprawnienia inicjalizatorów obiektów i kolekcji	331
10.2.1. Indeksery w inicjalizatorach obiektów	331
10.2.2. Używanie metod rozszerzających w inicjalizatorach kolekcji	335
10.2.3. Kod testów a kod produkcyjny	339
10.3. Operator ?. 340	
10.3.1. Proste i bezpieczne dereferencje właściwości	340

- 10.3.2. Szczegółowe omówienie operatora ?. 341
- 10.3.3. Obsługa porównań logicznych 342
- 10.3.4. Indeksery i operator ?. 344
- 10.3.5. Skuteczne używanie operatora ?. 344
- 10.3.6. Ograniczenia operatora ?. 346
- 10.4. Filtry wyjątków 346
- 10.4.1. Składnia i semantyka filtrów wyjątków 347
- 10.4.2. Ponawianie operacji 352
- 10.4.3. Zapis danych w dzienniku jako efekt uboczny 354
- 10.4.4. Pojedyncze, specyficzne filtry wyjątków 355
- 10.4.5. Dlaczego po prostu nie zgłaszać wyjątków? 355
- Podsumowanie 356

#### CZĘŚĆ 4. C# 7 I PRZYSZŁE WERSJE 357

- Rozdział 11. Łączenie danych z użyciem krotek 359
- 11.1. Wprowadzenie do krotek 360
- 11.2. Literały i typy krotek 361
- 11.2.1. Składnia 361
- 11.2.2. Wnioskowanie nazw elementów w literałach krotek (C# 7.1) 364
- 11.2.3. Krotki jako zbiory zmiennych 365
- 11.3. Typy krotek i konwersje 369
- 11.3.1. Typy literałów krotek 369
- 11.3.2. Konwersje z literałów krotek na typy krotek 371
- 11.3.3. Konwersja między typami krotek 374
- 11.3.4. Zastosowania konwersji 377
- 11.3.5. Sprawdzanie nazw elementów przy dziedziczeniu 377
- 11.3.6. Operatory równości i nierówności (C# 7.3) 378
- 11.4. Krotki w środowisku CLR 379

11.4.1. Wprowadzenie do typów System.ValueTuple<...>	379
11.4.2. Obsługa nazw elementów	380
11.4.3. Implementacje konwersji krotek	381
11.4.4. Tekstowe reprezentacje krotek	382
11.4.5. Standardowe porównania na potrzeby sprawdzania równości i sortowania	383
11.4.6. Strukturalne porównania na potrzeby sprawdzania równości i sortowania	384
11.4.7. Krotki jednowartościowe i duże krotki	386
11.4.8. Niegeneryczna struktura ValueTuple	387
11.4.9. Metody rozszerzające	387
11.5. Alternatywy dla krotek	388
11.5.1. System.Tuple<...>	388
11.5.2. Typy anonimowe	388
11.5.3. Typy nazwane	389
11.6. Zastosowania i rekomendacje	389
11.6.1. Niepubliczne interfejsy API i kod, który można łatwo modyfikować	390
11.6.2. Zmienne lokalne	390
11.6.3. Pola	392
11.6.4. Krotki i typowanie dynamiczne nie współdziałają dobrze ze sobą	393
Podsumowanie	394
Rozdział 12. Podział krotek i dopasowywanie wzorców	395
12.1. Podział krotek	396
12.1.1. Podział na nowe zmienne	397
12.1.2. Używanie podziału do przypisywania wartości istniejącym zmiennym i właściwościom	399
12.1.3. Szczegóły podziału literałów krotek	403
12.2. Podział typów innych niż krotki	403
12.2.1. Metody instancji odpowiedzialne za podział obiektów	403
12.2.2. Odpowiedzialne za podział metody rozszerzające a przeciążanie metod	404

12.2.3. Obsługa wywołań Deconstruct w kompilatorze 406

12.3. Wprowadzenie do dopasowywania wzorców 407

12.4. Wzorce dostępne w C# 7.0 409

12.4.1. Wzorce stałych 409

12.4.2. Wzorce typów 410

12.4.3. Wzorzec var 413

12.5. Używanie wzorców razem z operatorem is 414

12.6. Używanie wzorców w instrukcjach switch 416

12.6.1. Klauzule zabezpieczające 417

12.6.2. Zasięg zmiennej ze wzorca w klauzulach case 418

12.6.3. Kolejność przetwarzania w instrukcjach switch opartych na wzorcu 420

12.7. Przemyślenia na temat zastosowań opisanych mechanizmów 421

12.7.1. Wykrywanie możliwości podziału obiektów 422

12.7.2. Wykrywanie możliwości dopasowywania wzorców 422

Podsumowanie 423

Rozdział 13. Zwiększanie wydajności dzięki częstszemu przekazywaniu danych przez referencję 425

13.1. Przypomnienie - co wiesz o słowie kluczowym ref? 427

13.2. Zmienne lokalne ref i referencyjne zwracane wartości 429

13.2.1. Zmienne lokalne ref 430

13.2.2. Instrukcja return ref 435

13.2.3. Operator warunkowy ?: i wartości z modyfikatorem ref (C# 7.2) 437

13.2.4. Modyfikator ref readonly (C# 7.2) 438

13.3. Parametry in (C# 7.2) 440

13.3.1. Zgodność wstecz 441

13.3.2. Zaskakująca modyfikowalność parametrów in - zmiany zewnętrzne 442

13.3.3. Przeciążanie metod z użyciem parametrów in 444

13.3.4. Wskazówki dotyczące parametrów in 444

- 13.4. Deklarowanie struktur tylko do odczytu (C# 7.2) 446
  - 13.4.1. Wprowadzenie - niejawnie kopiowanie zmiennych tylko do odczytu 446
  - 13.4.2. Modyfikator readonly dla struktur 449
  - 13.4.3. Serializowane dane w XML-u są z natury przeznaczone do odczytu i zapisu 450
- 13.5. Metody rozszerzające z parametrami ref i in (C# 7.2) 451
  - 13.5.1. Używanie parametrów ref i in w metodach rozszerzających, aby uniknąć kopiowania 451
  - 13.5.2. Ograniczenia dotyczące metod rozszerzających z pierwszym parametrem ref lub in 453
- 13.6. Struktury referencyjne (C# 7.2) 454
  - 13.6.1. Reguły dotyczące struktur referencyjnych 455
  - 13.6.2. Typ Spani wywołanie stackalloc 456
  - 13.6.3. Reprezentacja struktur referencyjnych w kodzie pośrednim 460
- Podsumowanie 461
- Rozdział 14. Zwięzły kod w C# 7 463
  - 14.1. Metody lokalne 463
    - 14.1.1. Dostęp do zmiennych w metodach lokalnych 465
    - 14.1.2. Implementowanie metod lokalnych 468
    - 14.1.3. Wskazówki dotyczące użytkowania 473
  - 14.2. Zmienne out 476
    - 14.2.1. Wewnątrzwerszowe deklaracje zmiennych na potrzeby parametrów out 476
    - 14.2.2. Zniesione w C# 7.3 ograniczenia dotyczące zmiennych out i zmiennych generowanych we wzorcach 477
  - 14.3. Usprawnienia w literałach liczbowych 478
    - 14.3.1. Dwójkowe literały całkowitoliczbowe 478
    - 14.3.2. Separatory w postaci podkreślenia 479
  - 14.4. Wyrażenia throw 480
  - 14.5. Literał default (C# 7.1) 481
  - 14.6. Argumenty nazwane w dowolnym miejscu listy argumentów (C# 7.2) 482
  - 14.7. Dostęp private protected (C# 7.2) 484

14.8. Drobne usprawnienia z C# 7.3 484

14.8.1. Ograniczenia typów generycznych 484

14.8.2. Usprawnienia w wyborze wersji przeciążonych metod 485

14.8.3. Atrybuty pól powiązanych z automatycznie implementowanymi właściwościami 486

Podsumowanie 487

Rozdział 15. C# 8 i kolejne wersje 489

15.1. Typy referencyjne przyjmujące wartość null 490

15.1.1. Jaki problem rozwiązują typy referencyjne przyjmujące wartość null? 490

15.1.2. Zmiana działania typów referencyjnych w kontekście wartości null 491

15.1.3. Poznaj typy referencyjne przyjmujące null 492

15.1.4. Działanie typów referencyjnych przyjmujących null w czasie kompilacji i w czasie wykonywania kodu 493

15.1.5. Operator "a niech to" 496

15.1.6. Wrażenia z wprowadzania typów referencyjnych przyjmujących null 498

15.1.7. Przyszłe usprawnienia 500

15.2. Wyrażenia switch 504

15.3. Rekurencyjne dopasowywanie wzorców 506

15.3.1. Dopasowywanie z użyciem właściwości we wzorcach 507

15.3.2. Wzorce oparte na podziale 507

15.3.3. Pomijanie typów we wzorcach 508

15.4. Indeksy i przedziały 509

15.4.1. Typy i literały Index i Range 510

15.4.2. Stosowanie indeksów i przedziałów 511

15.5. Lepsza integracja asynchroniczności 512

15.5.1. Asynchroniczne zwalnianie zasobów z użyciem instrukcji using await 512

15.5.2. Asynchroniczne iteracje z użyciem instrukcji foreach await 514

15.5.3. Asynchroniczne iteratory 517

15.6. Funkcje, które nie znalazły się w wersji zapoznawczej 518

15.6.1. Domyślne metody interfejsu 518

15.6.2. Typy rekordowe 520

15.6.3. Krótki opis jeszcze innych funkcji 521

15.7. Udział w pracach 523

Wnioski 523

Dodatek A. Funkcje języka wprowadzone w poszczególnych wersjach 525