

Poznaj najskrytsze zakamarki C#!

Obecnie szczególnie cenione są te języki programowania, które pozwalają na błyskawiczne osiągnięcie oczekiwanych efektów. Dodatkowo absolutnie niezbędne jest zachowanie możliwości uruchamiania raz napisanej aplikacji na różnych platformach bez konieczności jej przepisywania. Nikt nie ma na to czasu! C# to nowoczesny język, który zdobył uznanie programistów na całym świecie, ponieważ spełnia nawet najbardziej wyśrubowane wymagania!

Ten bezcenny podręcznik zaprowadzi Cię w najskrytsze zakamki języka C#. Autorzy założyli, że znasz jego podstawy - to pozwoliło im skupić uwagę na niuansach, ciekawostkach oraz subtelnych szczegółach. W trakcie lektury zrozumiesz, co się dzieje w czeluściach C#, a dzięki temu unikniesz ukrytych pułapek. Książka ta jest obowiązkową pozycją każdego programisty C#. Bez niej najciekawsze funkcje C# wciąż będą Ci obce!

Dowiedz się:

- do czego przydają się typy generyczne
- jak zaimplementować iteratory
- jak zoptymalizować skompilowany kod
- w czym może pomóc język LINQ
- jak testować kod

Fascynująca podróż w głąb C#!

Przedmowa (17)

Wstęp (19)

Podziękowania (21)

O książce (23)

Część I Przygotowanie do wyprawy (31)

1. Nieustająca metamorfoza C# (33)

- 1.1. Na początek prosty typ danych (34)
 - 1.1.1. Typ Product w C# 1 (35)
 - 1.1.2. Kolekcje o mocnym typie w C# 2 (36)
 - 1.1.3. Właściwości implementowane automatycznie w C# 3 (37)
 - 1.1.4. Argumenty nazwane w C# 4 (38)
- 1.2. Sortowanie i filtrowanie (39)
 - 1.2.1. Sortowanie produktów po nazwie (40)
 - 1.2.2. Wyszukiwanie elementów w kolekcjach (43)
- 1.3. Obsługa braku danych (44)
 - 1.3.1. Reprezentacja nieznannej ceny (45)
 - 1.3.2. Parametry opcjonalne i wartości domyślne (46)
- 1.4. Wprowadzenie do LINQ (46)
 - 1.4.1. Wyrażenia kwerendowe i zapytania wewnętrzprocesowe (47)
 - 1.4.2. Wykonywanie zapytań na XML-u (48)
 - 1.4.3. LINQ dla SQL-a (49)
- 1.5. COM i typy dynamiczne (50)
 - 1.5.1. Uproszczenie współpracy z COM (50)
 - 1.5.2. Współpraca z językiem dynamicznym (51)
- 1.6. Analiza zawartości środowiska .NET (52)
 - 1.6.1. Język C# (53)
 - 1.6.2. Środowisko wykonania (53)
 - 1.6.3. Biblioteki środowiska (54)
- 1.7. Jak zmienić Twój kod w kod doskonały? (54)

- 1.7.1. Prezentacja pełnych programów w formie fragmentów kodu (55)
- 1.7.2. Kod dydaktyczny nie jest kodem produkcyjnym (56)
- 1.7.3. Twój nowy najlepszy przyjaciel - specyfikacja języka (56)
- 1.8. Podsumowanie (57)

2. Rdzeń systemu - C# 1 (59)

- 2.1. Delegaty (60)
 - 2.1.1. Przepis na prosty delegat (61)
 - 2.1.2. Łączenie i usuwanie delegatów (66)
 - 2.1.3. Dygresja na temat zdarzeń (67)
 - 2.1.4. Podsumowanie delegatów (68)
- 2.2. Charakterystyka systemu typów (69)
 - 2.2.1. Miejsce C# w świecie systemów typów (69)
 - 2.2.2. Kiedy system typów C# 1 jest niedostatecznie bogaty? (73)
 - 2.2.3. Podsumowanie charakterystyki systemu typów (75)
- 2.3. Typy wartościowe i referencyjne (76)
 - 2.3.1. Wartości i referencje w rzeczywistym świecie (76)
 - 2.3.2. Podstawy typów referencyjnych i wartościowych (77)
 - 2.3.3. Obalenie mitów (79)
 - 2.3.4. Opakowywanie i odpakowywanie (81)
 - 2.3.5. Podsumowanie typów wartościowych i referencyjnych (82)
- 2.4. Więcej niż C# 1 - nowe cechy na solidnym fundamencie (83)
 - 2.4.1. Cechy związane z delegatami (83)
 - 2.4.2. Cechy związane z systemem typów (85)
 - 2.4.3. Cechy związane z typami wartościowymi (87)
- 2.5. Podsumowanie (88)

Część II C# 2 - rozwiązanie problemów C# 1 (89)

3. Parametryzowanie typów i metod (91)

- 3.1. Czemu potrzebne są typy generyczne? (92)
- 3.2. Proste typy generyczne do codziennego użycia (94)
 - 3.2.1. Nauka przez przykład - słownik typu generycznego (94)
 - 3.2.2. Typy generyczne i parametry typów (96)
 - 3.2.3. Metody generyczne i czytanie deklaracji generycznych (100)
- 3.3. Wkraczamy głębiej (103)
 - 3.3.1. Ograniczenia typów (104)
 - 3.3.2. Interfejs argumentów typu dla metod generycznych (109)
 - 3.3.3. Implementowanie typów generycznych (110)
- 3.4. Zaawansowane elementy typów generycznych (116)
 - 3.4.1. Pola i konstruktory statyczne (117)
 - 3.4.2. Jak kompilator JIT traktuje typy generyczne (119)
 - 3.4.3. Iteracja przy użyciu typów generycznych (121)
 - 3.4.4. Refleksja i typy generyczne (123)
- 3.5. Ograniczenia typów generycznych C# i innych języków (128)
 - 3.5.1. Brak wariacji typów generycznych (128)
 - 3.5.2. Brak ograniczeń operatorów lub ograniczeń "numerycznych" (133)
 - 3.5.3. Brak generycznych właściwości, indeksatorów i innych elementów typu (135)
 - 3.5.4. Porównanie z C++ (136)
 - 3.5.5. Porównanie z typami generycznymi Javy (137)
- 3.6. Podsumowanie (139)

4. Wyrażanie "niczego" przy użyciu typów nullable (141)

- 4.1. Co robisz, kiedy zwyczajnie nie masz wartości? (142)
 - 4.1.1. Czemu zmienne typu wartościowego nie mogą zawierać null? (142)
 - 4.1.2. Metody reprezentacji wartości null w C# 1 (143)
- 4.2. System.Nullable<T> i System.Nullable (145)
 - 4.2.1. Wprowadzenie Nullable<T> (146)
 - 4.2.2. Opakowywanie i odpakowywanie Nullable<T> (149)
 - 4.2.3. Równość instancji Nullable<T> (150)
 - 4.2.4. Wsparcie ze strony niegenerycznej klasy Nullable (151)
- 4.3. Składnia C# 2 dla typów nullable (152)
 - 4.3.1. Modyfikator ? (152)
 - 4.3.2. Przypisania i porównania z null (154)
 - 4.3.3. Konwersje i operatory nullable (156)
 - 4.3.4. Logika typów nullable (159)
 - 4.3.5. Stosowanie operatora as z typami nullable (161)
 - 4.3.6. Nullowy operator koalescencyjny (161)
- 4.4. Nowatorskie zastosowania typów nullable (164)
 - 4.4.1. Testowanie operacji bez parametrów zwrotnych (165)
 - 4.4.2. Bezbolesne porównania przy użyciu nullable operatora koalescencyjnego (167)
- 4.5. Podsumowanie (169)

5. Przyspieszone delegaty (171)

- 5.1. Pożegnanie z dziwną składnią delegatów (173)
- 5.2. Konwersja grupy metod (174)
- 5.3. Kowariancja i kontrawariancja (176)
 - 5.3.1. Kontrawariancja parametrów delegatów (176)
 - 5.3.2. Kowariancja typu zwracanego delegata (178)
 - 5.3.3. Małe ryzyko niekompatybilności (179)
- 5.4. Akcje delegatów tworzone w miejscu przy użyciu metod anonimowych (180)
 - 5.4.1. Rozpoczynamy prosto - operując na parametrach (181)
 - 5.4.2. Zwracanie wartości z metod anonimowych (183)
 - 5.4.3. Ignorowanie parametrów typu (185)
- 5.5. Przechwytywanie zmiennych w metodach anonimowych (186)
 - 5.5.1. Definicja domknięcia i różnych typów zmiennych (187)
 - 5.5.2. Analiza zachowania zmiennych przechwyconych (188)
 - 5.5.3. Jaki cel mają zmienne przechwycone? (189)
 - 5.5.4. Przedłużone życie zmiennych przechwyconych (190)
 - 5.5.5. Instancje zmiennej lokalnej (192)
 - 5.5.6. Mieszanka zmiennych współdzielonych i odrębnych (194)
 - 5.5.7. Wskazówki odnośnie do zmiennych przechwyconych i podsumowanie (196)
- 5.6. Podsumowanie (197)

6. Implementowanie iteratorów w prosty sposób (199)

- 6.1. C# 1 - udręka ręcznego pisania iteratorów (201)
- 6.2. C# 2 - proste iteratory z wyrażeniami yield (204)
 - 6.2.1. Wprowadzenie do uproszczeń iteratorów i wyrażenia yield return (204)
 - 6.2.2. Wizualizacja toku wykonania iteratora (206)
 - 6.2.3. Zaawansowany tok wykonania iteratora (208)
 - 6.2.4. Dziwactwa w implementacji (211)

- 6.3. Przykłady z życia (213)
 - 6.3.1. Iterowanie po datach w harmonogramie (213)
 - 6.3.2. Iterowanie po wierszach pliku (214)
 - 6.3.3. Leniwe filtrowanie elementów z użyciem uproszczenia iteratora i predykatu (217)
- 6.4. Pseudosynchroniczny kod z użyciem biblioteki CCR (219)
- 6.5. Podsumowanie (222)

7. Pozostałe cechy C# 2 (225)

- 7.1. Typy częściowe (227)
 - 7.1.1. Tworzenie typu składającego się z kilku plików (227)
 - 7.1.2. Użycie typów częściowych (229)
 - 7.1.3. Metody częściowe - tylko w C# 3 (231)
- 7.2. Klasy statyczne (233)
- 7.3. Niezależny poziom dostępu do getterów i setterów właściwości (235)
- 7.4. Aliasy przestrzeni nazw (237)
 - 7.4.1. Kwalifikowanie aliasów przestrzeni nazw (238)
 - 7.4.2. Aliasy globalnej przestrzeni nazw (239)
 - 7.4.3. Aliasy zewnętrzne (240)
- 7.5. Dyrektywy pragma (241)
 - 7.5.1. Pragmy ostrzeżeń (242)
 - 7.5.2. Pragmy sum kontrolnych (243)
- 7.6. Bufory o stałym rozmiarze w kodzie niezarządzanym (243)
- 7.7. Udostępnianie wybranych elementów innym modułom (245)
 - 7.7.1. Zaprzyjaźnione moduły w prostym przypadku (246)
 - 7.7.2. Do czego warto używać InternalsVisibleTo? (247)
 - 7.7.3. InternalsVisibleTo i moduły podpisane (247)
- 7.8. Podsumowanie (248)

Część III C# 3 - rewolucja w metodzie programowania (251)

8. Redukcja nadmiarowości przez zmyślny kompilator (253)

- 8.1. Właściwości implementowane automatycznie (255)
- 8.2. Zmienne lokalne o typie niejawnym (257)
 - 8.2.1. Zastosowanie var do deklarowania zmiennych lokalnych (257)
 - 8.2.2. Ograniczenia w typach niejawnych (259)
 - 8.2.3. Zalety i wady typów niejawnych (260)
 - 8.2.4. Zalecenia (261)
- 8.3. Uproszczona inicjalizacja (262)
 - 8.3.1. Definicja prostych typów (262)
 - 8.3.2. Ustawianie prostych właściwości (263)
 - 8.3.3. Ustawianie właściwości obiektów zagnieżdżonych (265)
 - 8.3.4. Inicjalizatory kolekcji (266)
 - 8.3.5. Zastosowanie inicjalizatorów (269)
- 8.4. Tablice o typie niejawnym (270)
- 8.5. Typy anonimowe (272)
 - 8.5.1. Pierwsze spotkania z gatunkiem anonimowym (272)
 - 8.5.2. Części składowe typów anonimowych (274)
 - 8.5.3. Inicjalizatory odwzorowujące (275)
 - 8.5.4. Jaki to ma sens? (276)
- 8.6. Podsumowanie (277)

9. Wyrażenia lambda i drzewa wyrażen (279)

- 9.1. Wyrażenia lambda i delegaty (281)
 - 9.1.1. Przygotowanie - wprowadzenie delegatów typu Func<...> (281)
 - 9.1.2. Pierwsze przekształcenie na wyrażenie lambda (282)
 - 9.1.3. Używanie pojedynczego wyrażenia jako ciała (283)
 - 9.1.4. Lista parametrów o typie niejawnym (284)
 - 9.1.5. Skrót dla pojedynczego parametru (284)
- 9.2. Proste przykłady użycia List<T> i zdarzeń (286)
 - 9.2.1. Filtrowanie, sortowanie i operacje na listach (286)
 - 9.2.2. Logowanie w metodach obsługi zdarzeń (288)
- 9.3. Drzewa wyrażeń (289)
 - 9.3.1. Budowanie drzew wyrażeń w sposób programistyczny (289)
 - 9.3.2. Kompilowanie drzew wyrażeń do postaci delegatów (291)
 - 9.3.3. Konwersja wyrażeń lambda C# na drzewa wyrażeń (292)
 - 9.3.4. Drzewa wyrażeń w sercu LINQ (296)
 - 9.3.5. Drzewa wyrażeń poza LINQ (297)
- 9.4. Zmiany we wnioskowaniu typów i rozwiązywaniu przeciążeń (299)
 - 9.4.1. Powód do zmiany - usprawnienie wywołań metod generycznych (300)
 - 9.4.2. Wnioskowanie typu zwracanego funkcji anonimowych (301)
 - 9.4.3. Dwufazowe wnioskowanie typu (302)
 - 9.4.4. Wybieranie odpowiedniego przeciążenia metody (306)
 - 9.4.5. Podsumowanie wnioskowania typów i rozwiązywania przeciążeń (308)
- 9.5. Podsumowanie (308)

10. Metody rozszerzające (311)

- 10.1. Życie przed metodami rozszerzającymi (312)
- 10.2. Składnia metod rozszerzających (315)
 - 10.2.1. Deklarowanie metod rozszerzających (315)
 - 10.2.2. Wywoływanie metod rozszerzających (316)
 - 10.2.3. Wykrywanie metod rozszerzających (318)
 - 10.2.4. Wywołanie metody na pustej referencji (319)
- 10.3. Metody rozszerzające w .NET 3.5 (321)
 - 10.3.1. Pierwsze kroki z Enumerable (321)
 - 10.3.2. Filtrowanie z użyciem Where i spinania wywołań metod (323)
 - 10.3.3. Antrakt: czy metody Where nie widzieliśmy już wcześniej? (325)
 - 10.3.4. Projekcja przy użyciu metody Select i typów anonimowych (326)
 - 10.3.5. Sortowanie przy użyciu OrderBy (327)
 - 10.3.6. Przykłady logiki biznesowej z użyciem łańcuchowania (328)
- 10.4. Wskazówki i uwagi odnośnie do użycia (330)
 - 10.4.1. "Rozszerzanie świata" i wzbogacanie interfejsów (330)
 - 10.4.2. Płynne interfejsy (331)
 - 10.4.3. Rozważne użycie metod rozszerzających (332)
- 10.5. Podsumowanie (334)

11. Wyrażenia kwerendowe i LINQ dla Obiektów (335)

- 11.1. Wprowadzenie do LINQ (336)
 - 11.1.1. Podstawowe koncepcje w LINQ (336)
 - 11.1.2. Definiowanie przykładowego modelu danych (341)
- 11.2. Prosty początek - wybieranie elementów (343)
 - 11.2.1. Rozpoczynanie od źródła i kończenie na selekcji (343)
 - 11.2.2. Translacja kompilatora jako podstawa wyrażeń kwerendowych (344)

- 11.2.3. Zmienne zakresu i projekcje nietrywialne (347)
- 11.2.4. Cast, OfType i zmienne zakresu o typie jawnym (349)
- 11.3. Filtrowanie i porządkowanie sekwencji (351)
 - 11.3.1. Filtrowanie przy użyciu klauzuli where (351)
 - 11.3.2. Zdegenerowane wyrażenia kwerendowe (352)
 - 11.3.3. Porządkowanie przy użyciu klauzuli orderby (353)
- 11.4. Klauzule let i identyfikatory przezroczyste (356)
 - 11.4.1. Wprowadzenie do wykonania pośredniego z użyciem let (356)
 - 11.4.2. Identyfikatory przezroczyste (357)
- 11.5. Złączenia (359)
 - 11.5.1. Złączenia wewnętrzne korzystające z klauzul join (359)
 - 11.5.2. Złączenia grupowe z użyciem klauzul join ... into (363)
 - 11.5.3. Złączenia krzyżowe i spłaszczanie sekwencji przy użyciu wielokrotnych klauzul from (366)
- 11.6. Grupowania i kontynuacje (369)
 - 11.6.1. Grupowanie przy użyciu klauzuli group ... by (369)
 - 11.6.2. Kontynuacje zapytań (373)
- 11.7. Wybór pomiędzy wyrażeniami kwerendowymi a notacją kropkową (375)
 - 11.7.1. Operacje wymagające notacji kropkowej (376)
 - 11.7.2. Wyrażenia kwerendowe, dla których prostszym rozwiązaniem może się okazać notacja kropkowa (377)
 - 11.7.3. Gdzie wyrażenia kwerendowe lśnią? (377)
- 11.8. Podsumowanie (379)

12. LINQ - nie tylko kolekcje (381)

- 12.1. Odpytywanie bazy danych przez LINQ dla SQL-a (382)
 - 12.1.1. Zaczynamy od bazy danych i modelu (383)
 - 12.1.2. Zapytania wstępne (385)
 - 12.1.3. Zapytania wymagające złączeń (388)
- 12.2. Translacje przy użyciu IQueryable i IQueryable (390)
 - 12.2.1. Wprowadzenie do IQueryable<T> i związanych z nim interfejsów (391)
 - 12.2.2. Prototyp - implementacja interfejsów wykonująca wpisy w dzienniku (393)
 - 12.2.3. Spajanie wszystkiego razem - metody rozszerzające typu Queryable (395)
 - 12.2.4. Udawany dostawca w działaniu (397)
 - 12.2.5. Podsumowanie IQueryable (398)
- 12.3. Interfejsy zaprzyjaźnione z LINQ i LINQ dla XML-a (399)
 - 12.3.1. Rdzenne typy LINQ dla XML-a (399)
 - 12.3.2. Konstrukcja deklaratywna (401)
 - 12.3.3. Zapytania na pojedynczych węzłach (404)
 - 12.3.4. Operatory zapytań spłaszczonych (406)
 - 12.3.5. Praca w harmonii z LINQ (407)
- 12.4. Zastąpienie LINQ dla Obiektów Równoległym LINQ (408)
 - 12.4.1. Kreślenie zbioru Mandelbrota przez pojedynczy wątek (409)
 - 12.4.2. Wprowadzenie ParallelEnumerable, ParallelQuery i AsParallel (410)
 - 12.4.3. Podkreślanie zapytań równoległych (411)
- 12.5. Odwrócenie modelu zapytania przy użyciu LINQ dla Rx (413)
 - 12.5.1. IObservable<T> i IObservable<T> (414)
 - 12.5.2. Zaczynamy (ponownie) łagodnie (416)
 - 12.5.3. Odpytywanie obiektów obserwowalnych (417)
 - 12.5.4. Jaki to wszystko ma sens? (419)

- 12.6. Rozszerzanie LINQ dla Obiektów (420)
 - 12.6.1. Wytyczne odnośnie do projektu i implementacji (421)
 - 12.6.2. Proste rozszerzenie - wybieranie losowego elementu (422)
- 12.7. Podsumowanie (424)

Część IV C# 4 - dobra współpraca z innymi interfejsami (427)

13. Małe zmiany dla uproszczenia kodu (429)

- 13.1. Parametry opcjonalne i argumenty nazwane (430)
 - 13.1.1. Parametry opcjonalne (430)
 - 13.1.2. Argumenty nazwane (437)
 - 13.1.3. Złożenie dwóch cech w całość (441)
- 13.2. Usprawnienia we współpracy z COM (446)
 - 13.2.1. Horror automatyzacji Worda przed C# 4 (446)
 - 13.2.2. Zemsta parametrów opcjonalnych i argumentów nazwanych (447)
 - 13.2.3. Kiedy parametr ref nie jest parametrem ref? (448)
 - 13.2.4. Wywoływanie indeksów nazwanych (449)
 - 13.2.5. Łączenie głównych bibliotek COM-owych (451)
- 13.3. Wariancja typów generycznych dla interfejsów i delegatów (453)
 - 13.3.1. Typy wariancji: kowariancja i kontrawariancja (454)
 - 13.3.2. Użycie wariancji w interfejsach (455)
 - 13.3.3. Zastosowanie wariancji w delegatach (458)
 - 13.3.4. Złożone sytuacje (459)
 - 13.3.5. Restrykcje i uwagi (461)
- 13.4. Mikroskopijne zmiany w blokadach i zdarzeniach w formie pól (465)
 - 13.4.1. Solidne blokowanie (465)
 - 13.4.2. Zmiany w zdarzeniach w formie pól (467)
- 13.5. Podsumowanie (467)

14. Dynamiczne wiązanie w języku statycznym (469)

- 14.1. Co? Kiedy? Dlaczego? Jak? (471)
 - 14.1.1. Czym są typy dynamiczne? (471)
 - 14.1.2. Kiedy typy dynamiczne są użyteczne i dlaczego? (472)
 - 14.1.3. W jaki sposób C# zapewnia typy dynamiczne? (474)
- 14.2. Pięciominutowy przewodnik po typie dynamic (474)
- 14.3. Przykłady użycia typów dynamicznych (477)
 - 14.3.1. COM w ogólności i Microsoft Office w szczególności (477)
 - 14.3.2. Języki dynamiczne, takie jak IronPython (479)
 - 14.3.3. Typy dynamiczne w kodzie całkowicie zarządzanym (484)
- 14.4. Zaglądamy pod maskę (490)
 - 14.4.1. Wprowadzenie do DLR (491)
 - 14.4.2. Fundamentalne koncepcje DLR (493)
 - 14.4.3. Jak kompilator C# obsługuje słowo dynamic? (496)
 - 14.4.4. Kompilator C# staje się jeszcze sprytniejszy (500)
 - 14.4.5. Ograniczenia kodu dynamicznego (503)
- 14.5. Implementacja zachowania dynamicznego (506)
 - 14.5.1. Użycie klasy ExpandableObject (506)
 - 14.5.2. Użycie klasy DynamicObject (511)
 - 14.5.3. Implementacja IDynamicMetaObjectProvider (518)
- 14.6. Podsumowanie (522)

15. Jaśniejsze wyrażanie kodu przy użyciu kontraktów kodu (523)

- 15.1. Życie przed kontraktami kodu (525)
- 15.2. Wprowadzenie do kontraktów kodu (527)
 - 15.2.1. Warunki wstępne (529)
 - 15.2.2. Warunki końcowe (530)
 - 15.2.3. Inwarianty (531)
 - 15.2.4. Asercje i założenia (533)
 - 15.2.5. Kontrakty legacyjne (534)
- 15.3. Przepisywanie kodu binarnego przez ccrewrite i crefgen (536)
 - 15.3.1. Proste przepisywanie (536)
 - 15.3.2. Dziedziczenie kontraktów (538)
 - 15.3.3. Referencyjne moduły kontraktów (541)
 - 15.3.4. Zachowanie w przypadku porażki (543)
- 15.4. Sprawdzanie statyczne (545)
 - 15.4.1. Wprowadzenie do analizy statycznej (545)
 - 15.4.2. Zobowiązania niejawne (548)
 - 15.4.3. Selektywne włączanie opcji (551)
- 15.5. Dokumentowanie kodu przy użyciu cdocgen (554)
- 15.6. Kontrakty w praktyce (556)
 - 15.6.1. Filozofia - czym jest kontrakt? (557)
 - 15.6.2. Jak mam zacząć? (558)
 - 15.6.3. Opcje, wszędzie opcje (559)
- 15.7. Podsumowanie (562)

16. Dokąd teraz? (565)

- 16.1. C# - połączenie tradycji z nowoczesnością (566)
- 16.2. Spotkanie .NET z informatyką (567)
- 16.3. Świat informatyki (568)
- 16.4. Pożegnanie (569)

Dodatki (571)

A Standardowe operatory kwerendowe LINQ (573)

- A.1. Agregacja (574)
- A.2. Konkatenacja (575)
- A.3. Konwersja (575)
- A.4. Operatory jednoelementowe (577)
- A.5. Równość (578)
- A.6. Generacja (579)
- A.7. Grupowanie (580)
- A.8. Złączenia (580)
- A.9. Partycjonowanie (582)
- A.10. Projekcja (582)
- A.11. Kwantyfikatory (583)
- A.12. Filtrowanie (584)
- A.13. Operatory bazujące na zbiorach (584)
- A.14. Sortowanie (585)

B Kolekcje generyczne w .NET (587)

- B.1. Interfejsy (588)
- B.2. Listy (590)
 - B.2.1. List<T> (590)
 - B.2.2. Tablice (591)

- B.2.3. LinkedList<T> (592)
- B.2.4. Collection<T>, BindingList<T>, ObservableCollection<T> i KeyedCollection<TKey, TItem> (593)
- B.2.5. ReadOnlyCollection<T> i ReadOnlyObservableCollection<T> (594)
- B.3. Słowniki (594)
 - B.3.1. Dictionary<TKey, TValue> (594)
 - B.3.2. SortedList<TKey, TValue> i SortedDictionary<TKey, TValue> (595)
- B.4. Zbiory (596)
 - B.4.1. HashSet<T> (597)
 - B.4.2. SortedSet<T> (.NET 4) (597)
- B.5. Queue<T> i Stack<T> (598)
 - B.5.1. Queue<T> (598)
 - B.5.2. Stack<T> (598)
- B.6. Kolekcje konkurencyjne (.NET 4) (598)
 - B.6.1. IProducerConsumerCollection<T> i BlockingCollection<T> (599)
 - B.6.2. ConcurrentBag<T>, ConcurrentQueue<T>, ConcurrentStack<T> (600)
 - B.6.3. ConcurrentDictionary<TKey, TValue> (600)
- B.7. Podsumowanie (600)

C Podsumowanie wersji środowisk .NET (603)

- C.1. Główne wersje dystrybucyjne środowiska typu desktop (604)
- C.2. Cechy języka C# (605)
 - C.2.1. C# 2.0 (605)
 - C.2.2. C# 3 (605)
 - C.2.3. C# 4.0 (606)
- C.3. Biblioteki środowiska (606)
 - C.3.1. .NET 2.0 (606)
 - C.3.2. .NET 3.0 (606)
 - C.3.3. .NET 3.5 (607)
 - C.3.4. .NET 4 (607)
- C.4. Cechy środowiska uruchomieniowego (CLR) (608)
 - C.4.1. CLR 2.0 (608)
 - C.4.2. CLR 4.0 (609)
- C.5. Inne rodzaje środowiska uruchomieniowego (609)
 - C.5.1. Compact Framework (609)
 - C.5.2. Silverlight (610)
 - C.5.3. Micro Framework (611)
- C.6. Podsumowanie (611)

Skorowidz (613)