

Wstęp

- Dla kogo jest ta książka? (30)
- Wiemy, o czym myślisz (31)
- Metapoznanie: myślenie o myśleniu (33)
- Zmusz swój mózg do posłuszeństwa (35)
- Przeczytaj to (37)
- Grupa korektorów technicznych (38)
- Podziękowania (39)

Rozdział 1. Aplikacje Visual Studio w 10 minut lub mniej

- Dlaczego powinieneś uczyć się C# (42)
- C# oraz Visual Studio ułatwiają wiele czynności (43)
- Pomóż dyrektorowi naczelnemu zrezygnować z papieru (44)
- Sprawdź potrzeby Twoich użytkowników, zanim zaczniesz tworzyć program (45)
- Oto program, który zamierzasz stworzyć (46)
- Co robisz w Visual Studio (48)
- Co Visual Studio robi za Ciebie (48)
- Stwórz interfejs użytkownika (52)
- Za kulisami Visual Studio (54)
- Dodaj coś do automatycznie wygenerowanego kodu (55)
- Potrzebujemy bazy danych do przechowywania naszych informacji (58)
- IDE utworzyło bazę danych (59)
- SQL jest swoim własnym językiem (59)
- Tworzenie tabeli dla listy kontaktowej (60)
- Zakończ tworzenie tabeli (65)
- Wstaw dane z kart do bazy (66)
- Połącz formularz z bazą danych, korzystając ze źródeł danych (68)
- Dodaj kontrolki powiązane z bazą danych do formularza (70)
- Jak zamienić TWOJĄ aplikację w aplikację WSZYSTKICH (75)
- Przekaz aplikację innym użytkownikom (76)
- Jeszcze nie skończyłeś: przetestuj instalację (77)
- Stworzyłeś pełnowartościową aplikację bazodanową (78)

Rozdział 2. Pod maską

- Kiedy robisz to... (80)
- ...IDE robi to (81)
- Skąd się biorą programy (82)
- IDE pomaga Ci kodować (84)
- Kiedy zmieniasz coś w IDE, zmieniasz także swój kod (86)
- Anatomia programu (88)
- Twój program wie, gdzie zacząć (90)
- W tej samej przestrzeni nazw mogą być dwie klasy (97)
- Twoje programy używają zmiennych do pracy z danymi (98)
- C# używa znanych symboli matematycznych (100)
- Użyj debugera, by zobaczyć, jak zmieniają się wartości zmiennych (101)
- Pętle wykonują czynność wielokrotnie (103)
- Kodowanie czas zacząć (104)

- Instrukcje if/else podejmują decyzje (105)
- Ustal warunki i sprawdź, czy są prawdziwe (106)

Rozdział 3. Tworzenie kodu ma sens

- W jaki sposób Maciek myśli o swoich problemach (122)
- W jaki sposób system nawigacyjny w samochodzie Maćka rozwiązuje jego problemy (123)
- Klasa Navigator napisana przez Maćka posiada metody do ustalania i modyfikacji tras (124)
- Wykorzystaj to, czego się nauczyłeś, do napisania prostego programu używającego klas (125)
- Maciek może użyć obiektów do rozwiązania swojego problemu (128)
- Używasz klasy do utworzenia obiektu (129)
- Kiedy tworzysz obiekt na podstawie klasy, to taki obiekt nazywamy instancją klasy (130)
- Lepsze rozwiązanie... uzyskane dzięki obiektom! (131)
- Instancja używa pól do przechowywania informacji (136)
- Stwórzmy kilka instancji! (137)
- Dzięki za pamięć (138)
- Co Twój program ma na myśli (139)
- Możesz używać nazw klas i metod w celu uczynienia kodu bardziej intuicyjnym (140)
- Nadaj swojej klasie naturalną strukturę (142)
- Diagramy klas pozwalają w sensowny sposób zorganizować klasy (144)
- Utwórz klasę do pracy z kilkoma facetami (148)
- Utwórz projekt dla facetów (149)
- Utwórz formularz do interakcji z facetami (150)
- Jest jeszcze prostszy sposób inicjalizacji obiektów (153)

Rozdział 4. Jest 10:00. Czy wiesz, gdzie są Twoje dane?

- Typ zmiennej określa rodzaj danych, jakie zmienna może przechowywać (160)
- Zmienna jest jak kubek z danymi (162)
- 10 kilogramów danych w pięciokilogramowej torebce (163)
- Nawet wtedy, gdy liczba ma prawidłowy rozmiar, nie możesz przypisać jej do każdej zmiennej (164)
- Kiedy rzutujesz wartość, która jest zbyt duża, C# dopasowuje ją automatycznie (165)
- C# przeprowadza niektóre rzutowania automatycznie (166)
- Kiedy wywołujesz metodę, zmienne muszą pasować do typów parametrów (167)
- Połączenie = z operatorem (172)
- Także obiekty używają zmiennych (173)
- Korzystaj ze swoich obiektów za pomocą zmiennych referencyjnych (174)
- Referencje są jak etykiety do Twoich obiektów (175)
- Jeżeli nie ma już żadnej referencji, Twoje obiekty są usuwane z pamięci (176)
- Referencje wielokrotne i ich efekty uboczne (177)
- Dwie referencje oznaczają DWA sposoby na zmianę danych obiektu (182)
- Specjalny przypadek: tablice (183)
- Witamy w barze Niechlujny Janek - najtańsze kanapki w mieście! (185)
- Obiekty używają referencji do komunikacji między sobą (187)
- Tam, gdzie obiektów jeszcze nie było (188)

- Napisz grę w literki (193)

Laboratorium C# numer 1. Dzień na wyścigach

- Specyfikacja: stwórz symulator wyścigów (202)
- Końcowy produkt (210)

Rozdział 5. Co ma być ukryte... niech będzie ukryte

- Krystyna planuje przyjęcia (212)
- Co powinien robić program szacujący? (213)
- Jazda próbna Krystyny (218)
- Każda opcja powinna być obliczana osobno (220)
- Bardzo łatwo przez przypadek źle skorzystać z obiektów (222)
- Hermetyzacja oznacza, że niektóre dane w klasie są prywatne (223)
- Użyj hermetyzacji w celu kontroli dostępu do metod i pól Twojej klasy (224)
- Ale czy jego prawdziwa tożsamość jest NAPRAWDĘ chroniona? (225)
- Dostęp do prywatnych pól i metod można uzyskać tylko z wnętrza klasy (226)
- Hermetyzacja utrzymuje Twoje dane w nieskazitelnym stanie (234)
- Właściwości sprawiają, że hermetyzacja będzie łatwiejsza (235)
- Stwórz aplikację do przetestowania klasy Farmer (236)
- Użyj automatycznych właściwości do ukończenia klasy (237)
- Co wtedy, gdy chcemy zmienić pole mnożnika wyżywienia? (238)
- Użyj konstruktora do inicjalizacji pól prywatnych (239)

Rozdział 6. Drzewo genealogiczne Twoich obiektów

- Krystyna organizuje także przyjęcia urodzinowe (248)
- Potrzebujemy klasy BirthdayParty (249)
- Stwórz program Planista przyjęć w wersji 2.0 (250)
- Kiedy klasy używają dziedziczenia, kod musi być napisany tylko raz (258)
- Zbuduj model klasy, rozpoczynając od rzeczy ogólnych i przechodząc do bardziej konkretnych (259)
- W jaki sposób zaprojektowałbyś symulator zoo? (260)
- Użyj dziedziczenia w celu uniknięcia zwielokrotniania kodu w klasach potomnych (261)
- Pomyśl, w jaki sposób pogrupować zwierzęta (263)
- Stwórz hierarchię klas (264)
- Każda klasa pochodna rozszerza klasę bazową (265)
- Klasa pochodna może przesłaniać odziedziczone metody w celu ich modyfikacji lub zmiany (270)
- W każdym miejscu, gdzie możesz skorzystać z klasy bazowej, możesz zamiast niej użyć jednej z jej klas pochodnych (271)
- Klasa pochodna może ukrywać metody klasy bazowej (278)
- Używaj override i virtual, by dziedziczyć zachowania (280)
- Teraz jesteś już gotowy do dokończenia zadania Krystyny (284)
- Stwórz system zarządzania ulem (289)
- Najpierw stworzysz system podstawowy (290)
- Użyj dziedziczenia, aby rozszerzyć system zarządzania pszczołami (294)

Rozdział 7. Klasy, które dotrzymują swoich obietnic

- Wróćmy do pszczelej korporacji (300)
- Możemy użyć dziedziczenia do utworzenia klas dla różnych typów pszczół (301)
- Interfejs daje klasie do zrozumienia, że musi ona zaimplementować określone metody i właściwości (302)
- Użyj słowa kluczowego interface do zdefiniowania interfejsu (303)
- Klasy implementujące interfejsy muszą zawierać WSZYSTKIE ich metody (305)
- Nie możesz utworzyć instancji interfejsu, ale możesz uzyskać jego referencję (308)
- Referencje interfejsów działają tak samo jak referencje obiektów (309)
- Za pomocą "is" możesz sprawdzić, czy klasa implementuje określony interfejs (310)
- Interfejsy mogą dziedziczyć po innych interfejsach (311)
- Rzutowanie w górę działa w odniesieniu do obiektów i interfejsów (315)
- Rzutowanie w dół pozwala zamienić urządzenie z powrotem w ekspres do kawy (316)
- Rzutowanie w górę i w dół działa także w odniesieniu do interfejsów (317)
- Jest coś więcej niż tylko public i private (321)
- Modyfikatory dostępu zmieniają widoczność (322)
- Obiekty niektórych klas nigdy nie powinny być tworzone (325)
- Klasa abstrakcyjna jest jak skrzyżowanie klasy i interfejsu (326)
- Metoda abstrakcyjna nie ma ciała (329)
- Polimorfizm oznacza, że jeden obiekt może przyjmować wiele różnych postaci (337)

Rozdział 8. Przechowywanie dużej ilości danych

- Łańcuchy znaków nie zawsze sprawdzają się przy kategoryzowaniu danych (356)
- Typy wyliczeniowe pozwalają Ci wyliczyć prawidłowe wartości (357)
- Typy wyliczeniowe pozwalają na reprezentowanie liczb za pomocą nazw (358)
- Możesz użyć tablicy, aby stworzyć talię kart... (361)
- Listy są bardziej elastyczne niż tablice (364)
- Typy generyczne mogą przechowywać każdy typ (368)
- Inicjalizatory kolekcji działają tak samo jak inicjalizatory obiektu (372)
- Stwórzmy listę kaczek (373)
- Listy są proste, ale SORTOWANIE może być skomplikowane (374)
- IComparable<T> pomoże Ci posortować listę kaczek (375)
- Użyj interfejsu IComparer, aby powiedzieć liście, jak ma sortować (376)
- Utwórz instancję obiektu porównującego (377)
- IComparer może wykonywać złożone porównania (378)
- Przesłonięcie metody ToString() pozwala obiektom przedstawiać się (381)
- Zmień pętle foreach tak, by obiekty Duck i Card same się opisywały (382)
- Używając IEnumerable, możesz rzutować całą listę w górę (384)
- Możesz tworzyć własne przeciążone metody (385)
- Wybrane funkcjonalności słownika (392)
- Napisz program korzystający ze słownika (393)
- I jeszcze WIĘCEJ typów kolekcji... (405)
- Kolejka działa według reguły: pierwszy przyszedł, pierwszy wyszedł (406)
- Stos działa według reguły: ostatni przyszedł, pierwszy wyszedł (407)

Laboratorium C# numer 2. Wyprawa

- Specyfikacja: utwórz grę przygodową (412)

- Zabawa dopiero się zaczyna! (432)

Rozdział 9. Zapisz tablice bajtów, zapisz świat

- C# używa strumieni do zapisu i odczytu danych (434)
- Różne strumienie zapisują i odczytują różne rzeczy (435)
- FileStream zapisuje bajty do pliku (436)
- W jaki sposób zapisać tekst do pliku w trzech prostych krokach (437)
- Zapis i odczyt wymaga dwóch obiektów (441)
- Dane mogą przechodzić przez więcej niż jeden strumień (442)
- Użyj wbudowanych obiektów do wyświetlenia standardowych okien dialogowych (445)
- Okna dialogowe są kolejnymi kontrolkami .NET (446)
- Okna dialogowe także są obiektami (447)
- Używaj wbudowanych klas File oraz Directory do pracy z plikami i katalogami (448)
- Używaj okien dialogowych do otwierania i zapisywania plików (451)
- Dzięki IDisposable obiekty usuwane są prawidłowo (453)
- Unikaj błędów systemowych, korzystając z instrukcji using (454)
- Zapisywanie danych do plików wymaga wielu decyzji (460)
- Użyj instrukcji switch do wyboru właściwej opcji (461)
- Serializacja pozwala Ci zapisywać lub odczytywać całe obiekty naraz (468)
- .NET automatycznie konwertuje tekst do postaci Unicode (473)
- C# może użyć tablicy bajtów do przesyłania danych (474)
- Pliki utworzone dzięki serializacji można także zapisywać i odczytywać ręcznie (477)
- Praca z plikami binarnymi może być skomplikowana (479)
- StreamReader i StreamWriter będą do tego odpowiednie (481)
- Użyj Stream.Read() do odczytywania bajtów ze strumienia (482)

Rozdział 10. Gaszenie pożarów nie jest już popularne

- Damian potrzebuje swoich wymówek, aby być mobilnym (488)
- Kiedy program zgłasza wyjątek, .NET tworzy obiekt Exception (492)
- Wszystkie obiekty wyjątków dziedziczą po Exception (496)
- Debugger pozwala Ci wyśledzić wyjątki w kodzie i zapobiec im (497)
- Użyj debugera wbudowanego w IDE, aby znaleźć problem w programie do zarządzania wymówkami (498)
- Obsłuż wyjątki za pomocą try i catch (503)
- Co się stanie, jeżeli wywoływana metoda będzie niebezpieczna? (504)
- Użyj debugera do prześledzenia przepływu w blokach try/catch (506)
- Jeśli posiadasz kod, który ZAWSZE musi zostać wykonany, zastosuj finally (508)
- Jedna klasa zgłasza wyjątek, inna klasa go wyłapuje (515)
- Pszczoły potrzebują wyjątku OutOfHoney (516)
- Łatwy sposób na uniknięcie licznych problemów: using umożliwia Ci stosowanie try i finally za darmo (519)
- Unikanie wyjątków: zaimplementuj IDisposable, aby przeprowadzić własne procedury sprzątnięcia (520)
- Najgorszy z możliwych bloków catch: komentarze (522)
- Tymczasowe rozwiązania są dobre (tymczasowo) (523)
- Kilka wskazówek dotyczących obsługi wyjątków (524)
- Damian w końcu pojechał na urlop... (527)

Rozdział 11. Co robi Twój kod, kiedy nie patrzysz

- Czy kiedykolwiek marzyłeś o tym, aby Twoje obiekty potrafiły samodzielnie myśleć? (530)
- Ale skąd obiekt WIE, że ma odpowiedzieć? (530)
- Kiedy wystąpi ZDARZENIE... obiekty nasłuchują (531)
- Jeden obiekt wywołuje zdarzenie, inne nasłuchują... (532)
- Potem inne obiekty obsługują zdarzenie (533)
- Łącząc punkty (534)
- IDE automatycznie tworzy za Ciebie procedury obsługi zdarzeń (538)
- Ogólny typ EventHandler pozwala definiować własne typy zdarzeń (544)
- Wszystkie formularze, które utworzyłeś, używają zdarzeń (545)
- Jedno zdarzenie, wiele procedur obsługi (546)
- Połączenie nadawców zdarzenia z jego odbiorcami (548)
- Delegat ZASTĘPUJE właściwą metodę (549)
- Delegat w akcji (550)
- Każdy obiekt może subskrybować publiczne zdarzenie... (553)
- Użyj funkcji zwrotnej, by wiedzieć, kto nasłuchuje (554)
- Funkcje zwrotne są jedynie sposobem używania delegatów (556)

Rozdział 12. Wiedza, moc i tworzenie ciekawych rzeczy

- Przebyłeś długą drogę, mały (564)
- Zajmowaliśmy się także pszczołami (565)
- Architektura symulatora ula (566)
- Budowanie symulatora ula (567)
- Życie i śmierć kwiatów (571)
- Teraz potrzebujemy klasy Bee (572)
- PPBP (Programiści Przeciwko Bezdolnym Pszczołom) (576)
- Ul działa na miód (576)
- Wypełnianie klasy Hive (580)
- Metoda Go() klasy Hive (581)
- Jesteśmy gotowi na stworzenie świata (582)
- Tworzymy system turowy (583)
- Oto kod klasy World (584)
- Uczenie pszczoł zachowań (590)
- Główny formularz wywołuje Go() dla całego świata (592)
- Możemy użyć obiektu World do pobrania statystyk (593)
- Zegary sygnalizują zdarzenia wielokrotnie (594)
- Za kulisami zegar używa zdarzeń (595)
- Pracujemy z grupami pszczoł (602)
- Kolekcje kolekcjonują... DANE (603)
- LINQ ułatwia pracę z danymi w kolekcjach i bazach danych (605)
- Ostatnie wyzwanie: Otwórz i Zapisz (607)

Rozdział 13. Upiększ to

- Cały czas do interakcji z programami używałeś kontrolek (612)
- Kontrolki formularza są tylko obiektami (613)
- Użyj kontrolek do animacji symulatora ula (614)

- Dodaj do projektu rendering (616)
- Kontrolki są dobrze dostosowane do wyświetlania różnych elementów wizualnych (618)
- Stwórz swoją pierwszą animowaną kontrolkę (621)
- Utwórz przycisk, aby dodać BeeControl do formularza (624)
- Twoje kontrolki także muszą usuwać swoje kontrolki! (625)
- UserControl to dobry sposób na tworzenie kontrolek (626)
- Mechanizm renderujący używa BeeControl do rysowania animowanych pszczoł na formularzu (628)
- Dodaj do projektu formularze reprezentujące ul i pole (630)
- Stwórz klasę Renderer (631)
- Zmieniłeś rozmiar bitmap przy pomocy obiektu Graphics (640)
- Zasoby Twoich obrazków przechowywane są w postaci obiektów Bitmap (641)
- Użyj System.Drawing, by samemu PRZEJĄĆ KONTROLĘ nad grafiką (642)
- 30-sekundowa podróż do świata tajemnic grafiki GDI+ (643)
- Użyj Graphics, aby na formularzu narysować obrazek (644)
- Klasa Graphics może usunąć problem przezroczystości... (649)
- Użyj zdarzenia Paint, aby grafika była mocno związana z formularzem (650)
- Bliższe spojrzenie na sposób rysowania formularzy i kontrolek (653)
- Podwójne buforowanie czyni animację bardziej płynną (656)
- Użyj obiektu Graphics i procedury obsługi zdarzenia do drukowania (662)

Rozdział 14. Kapitan Wspaniały. Śmierć obiektu

- Twoją ostatnią szansą na ZROBIENIE czegoś... jest użycie finalizatora (676)
- Kiedy DOKŁADNIE wywoływany jest finalizator? (677)
- Dispose() działa z using, a finalizatory działają z mechanizmem oczyszczania pamięci (678)
- Finalizatory nie mogą polegać na stabilności (680)
- Spraw, aby obiekt serializował się w Dispose() (681)
- Struktura jest podobna do obiektu... (685)
- ...ale nie jest obiektem (685)
- Wartości są kopiowane, referencje są przypisywane (686)
- Stos i sarta: więcej na temat pamięci (689)
- Używaj parametrów wyjściowych, by zwracać z metody więcej niż jedną wartość (692)
- Przekazuj referencje, używając modyfikatora ref (693)
- Używaj parametrów opcjonalnych, by określać wartości domyślne (694)
- Jeśli musisz używać wartości pustych, stosuj typy, które je akceptują (695)
- Typy akceptujące wartości puste poprawiają odporność programów (696)
- Kapitan Wspaniały... nie tak bardzo (699)
- Metody rozszerzające zwiększają funkcjonalność ISTNIEJĄCYCH klas (700)
- Rozszerzanie podstawowego typu: string (702)

Rozdział 15. Przejmij kontrolę nad danymi

- Łatwy projekt... (708)
- ...ale dane są w różnych miejscach (709)
- Dzięki LINQ możesz pobrać dane z różnych źródeł (710)
- Kolekcje .NET są przystosowane do działania z LINQ (711)

- LINQ ułatwia wykonywanie zapytań (712)
- LINQ jest prosty, ale Twoje zapytania wcale takie być nie muszą (713)
- LINQ ma wiele zastosowań (716)
- LINQ może połączyć Twoje wyniki w grupy (721)
- Połącz wartości Janka w grupy (722)
- Użyj join do połączenia dwóch kolekcji w jednym zapytaniu (725)
- Janek zaoszczędził mnóstwo szmalu (726)
- Połącz LINQ z bazą danych SQL (728)
- Użyj join, aby połączyć dane Starbuzz i Papierni Obiektowo (732)

Laboratorium C# numer 3. Invaders

- Dziadek wszystkich gier (736)
- Można zrobić znacznie więcej... (755)

Dodatek A 11 najważniejszych rzeczy, które chcieliśmy umieścić w tej książce

- 1. Podstawy (758)
- 2. Przestrzenie nazw i złożenia (764)
- 3. Użyj BackgroundWorker, by poprawić działanie interfejsu użytkownika (768)
- 4. Klasa Type oraz metoda GetType() (771)
- 5. Równość, IEquatable oraz Equals() (772)
- 6. Stosowanie yield return do tworzenia obiektów umożliwiających iterację (775)
- 7. Refaktoryzacja (778)
- 8. Anonimowe typy i metody oraz wyrażenia lambda (780)
- 9. Serializacja przy użyciu DataContractSerializer (782)
- 10. Zastosowanie LINQ to XML (784)
- 11. Windows Presentation Foundation (786)
- Czy wiesz, że C# i .NET Framework potrafią... (788)

Skorowidz (791)