

EXCEL[®] 2019 PL

Programowanie w VBA



Michael Alexander
Dick Kusleika

Helion 

Tytuł oryginału: Excel 2019 Power Programming with VBA

Tłumaczenie: Grzegorz Kowalczyk

ISBN: 978-83-283-6634-3

Copyright © 2019 by John Wiley & Sons, Inc., Indianapolis, Indiana

All Rights Reserved. This translation published under license with the original publisher John Wiley & Sons, Inc.

Wiley and the Wiley logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. Microsoft and Excel are registered trademarks of Microsoft Corporation. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

Translation copyright © 2020 by Helion SA

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Helion SA dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Helion SA nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Helion SA

ul. Kościuszki 1c, 44-100 Gliwice

tel. 32 231 22 19, 32 230 98 63

e-mail: helion@helion.pl

WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/e19pww>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

Spis treści

Przedmowa	23
Część I. Wprowadzenie do języka Excel VBA	29
Rozdział 1. Podstawy projektowania aplikacji arkusza kalkulacyjnego	31
Czym jest aplikacja arkusza kalkulacyjnego?	31
Etapy projektowania aplikacji	32
Określanie wymagań użytkownika	33
Planowanie aplikacji spełniającej wymagania użytkownika	34
Wybieranie odpowiedniego interfejsu użytkownika	35
Dostosowywanie Wstążki do potrzeb użytkownika	36
Dostosowywanie menu podręcznego do potrzeb użytkownika	36
Definiowanie klawiszy skrótu	37
Tworzenie niestandardowych okien dialogowych	37
Zastosowanie formantów ActiveX w arkuszu	38
Rozpoczęcie prac projektowych	39
Zadania realizowane z myślą o końcowym użytkowniku	40
Testowanie aplikacji	40
Uodpornianie aplikacji na błędy popełniane przez użytkownika	41
Nadawanie aplikacji przyjaznego, intuicyjnego i estetycznego wyglądu	43
Tworzenie systemu pomocy i dokumentacji przeznaczonej dla użytkownika	44
Dokumentowanie prac projektowych	44
Przekazanie aplikacji użytkownikom	44
Aktualizacja aplikacji (kiedy to konieczne)	45
Inne kwestie dotyczące projektowania	45
Wersja Excela zainstalowana przez użytkownika	45
Wersje językowe	46
Wydajność systemu	46
Tryby karty graficznej	46

Rozdział 2. Wprowadzenie do języka VBA	47
Rejestrator makr Excela	47
Tworzenie pierwszego makra	48
Porównanie rejestrowania makr z odwołaniami względnymi i bezwzględnymi	51
Inne zagadnienia związane z makrami	55
Praca z edytorem Visual Basic Editor (VBE)	59
Podstawowe elementy edytora VBE	60
Tajemnice okna Project	61
Tajemnice okna Code	63
Dostosowywanie środowiska edytora Visual Basic	66
Karta Editor Format	68
Karta General	69
Karta Docking	69
Podstawowe informacje o języku VBA	70
Obiekty	70
Kolekcje	71
Właściwości	72
Tajemnice obiektów Range	75
Wyszukiwanie właściwości obiektów Range	75
Właściwość Range	75
Właściwość Cells	76
Właściwość Offset	78
Podstawowe zagadnienia, które należy zapamiętać	80
Nie panikuj — nie jesteś sam	81
Przeczytaj resztę książki	82
Pozwól Excelowi napisać makro za Ciebie	82
Korzystaj z systemu pomocy	82
Używaj przeglądarki obiektów	83
Szukaj kodu w internecie	84
Wykorzystuj fora dyskusyjne użytkowników Excela	84
Odwiedzaj blogi ekspertów	85
Poszukaj szkolenia wideo na YouTube	85
Ucz się z Microsoft Office Dev Center	86
Analizuj inne aplikacje Excela, które są używane w Twojej organizacji	86
Zapytaj lokalnego guru	86
Rozdział 3. Podstawy programowania w języku VBA	87
Przegląd elementów języka VBA	87
Komentarze	89
Zmienne, typy danych i stałe	90
Definiowanie typów danych	91
Deklarowanie zmiennych	93
Zasięg zmiennych	95
Zastosowanie stałych	97
Praca z łańcuchami tekstu	98
Przetwarzanie dat	99

Instrukcje przypisania	100
Tablice	102
Deklarowanie tablic	102
Deklarowanie tablic wielowymiarowych	103
Deklarowanie tablic dynamicznych	103
Zmienne obiektowe	103
Typy danych definiowane przez użytkownika	105
Wbudowane funkcje VBA	105
Praca z obiektami i kolekcjami	108
Polecenie With ... End With	108
Polecenie For Each ... Next	109
Sterowanie sposobem wykonywania procedur	111
Polecenie GoTo	111
Polecenie If ... Then	112
Polecenie Select Case	115
Wykonywanie bloku instrukcji w ramach pętli	118
Rozdział 4. Tworzenie procedur w języku VBA	125
Kilka słów o procedurach	125
Deklarowanie procedury Sub	126
Zasięg procedury	126
Wykonywanie procedur Sub	128
Uruchamianie procedury przy użyciu polecenia Run Sub/UserForm	128
Uruchamianie procedury z poziomu okna dialogowego Makro	129
Uruchamianie procedury przy użyciu skrótu z klawiszem Ctrl	130
Uruchamianie procedury z poziomu Wstążki	131
Uruchamianie procedur za pośrednictwem niestandardowego menu podręcznego	131
Wywoływanie procedury z poziomu innej procedury	131
Uruchamianie procedury poprzez kliknięcie obiektu	135
Wykonywanie procedury po wystąpieniu określonego zdarzenia	137
Uruchamianie procedury z poziomu okna Immediate	137
Przekazywanie argumentów procedurom	138
Metody obsługi błędów	141
Przechwytywanie błędów	141
Przykłady kodu źródłowego obsługującego błędy	142
Praktyczny przykład wykorzystujący procedury Sub	145
Cel	145
Wymagania projektowe	145
Co już wiesz	146
Podejście do zagadnienia	146
Wstępne rejestrowanie makr	147
Przygotowania	148
Tworzenie kodu źródłowego	149
Tworzenie procedury sortującej	150
Dodatkowe testy	154
Usuwanie problemów	154
Dostępność narzędzia	157
Ocena projektu	158

Rozdział 5. Tworzenie funkcji w języku VBA	159
Porównanie procedur Sub i Function	159
Dlaczego tworzymy funkcje niestandardowe?	160
Twoja pierwsza funkcja	160
Zastosowanie funkcji w arkuszu	161
Zastosowanie funkcji w procedurze języka VBA	161
Analiza funkcji niestandardowej	162
Procedury Function	164
Zasięg funkcji	165
Wywoływanie funkcji	165
Argumenty funkcji	168
Przykłady funkcji	168
Funkcja bezargumentowa	168
Funkcje jednoargumentowe	171
Funkcje z dwoma argumentami	173
Funkcja pobierająca tablicę jako argument	174
Funkcje z argumentami opcjonalnymi	175
Funkcje zwracające tablicę VBA	176
Funkcje zwracające wartość błędu	178
Funkcje o nieokreślonej liczbie argumentów	180
Emulacja funkcji arkuszowej SUMA	181
Rozszerzone funkcje daty	183
Wykrywanie i usuwanie błędów w funkcjach	185
Okno dialogowe Wstawianie funkcji	186
Zastosowanie metody MacroOptions	187
Definiowanie kategorii funkcji	188
Ręczne dodawanie opisu funkcji	189
Zastosowanie dodatków do przechowywania funkcji niestandardowych	190
Korzystanie z Windows API	190
Przykłady zastosowania funkcji interfejsu API systemu Windows	191
Identyfikacja katalogu domowego systemu Windows	191
Wykrywanie wciśnięcia klawisza Shift	193
Dodatkowe informacje na temat funkcji interfejsu API	193
Rozdział 6. Obsługa zdarzeń	195
Co powinieneś wiedzieć o zdarzeniach	195
Sekwencje zdarzeń	196
Gdzie należy umieścić procedury obsługi zdarzeń?	196
Wyłączanie obsługi zdarzeń	197
Tworzenie kodu procedury obsługi zdarzeń	198
Procedury obsługi zdarzeń z argumentami	199
Zdarzenia poziomu skoroszytu	201
Zdarzenie Open	202
Zdarzenie Activate	202
Zdarzenie SheetActivate	203
Zdarzenie NewSheet	203
Zdarzenie BeforeSave	203
Zdarzenie Deactivate	204
Zdarzenie BeforePrint	204
Zdarzenie BeforeClose	205

Zdarzenia poziomu arkusza	207
Zdarzenie Change	208
Monitorowanie zmian w wybranym zakresie komórek	209
Zdarzenie SelectionChange	213
Zdarzenie BeforeDoubleClick	214
Zdarzenie BeforeRightClick	214
Zdarzenia dotyczące aplikacji	215
Włączanie obsługi zdarzeń poziomu aplikacji	215
Sprawdzanie, czy skorysyt jest otwarty	216
Monitorowanie zdarzeń poziomu aplikacji	218
Zdarzenia niezwiązane z obiektami	218
Zdarzenie OnTime	219
Zdarzenie OnKey	220
Rozdział 7. Przykłady i techniki programowania w języku VBA	225
Nauka poprzez praktykę	225
Przetwarzanie zakresów	226
Kopiowanie zakresów	226
Przenoszenie zakresów	227
Kopiowanie zakresu o zmiennej wielkości	227
Zaznaczanie oraz identyfikacja różnego typu zakresów	229
Zmiana rozmiaru zakresu komórek	230
Wprowadzanie wartości do komórki	231
Wprowadzanie wartości do następnej pustej komórki	232
Wstrzymywanie działania makra w celu umożliwienia pobrania zakresu wyznaczonego przez użytkownika	233
Zliczanie zaznaczonych komórek	235
Określanie typu zaznaczonego zakresu	235
Wydajne przetwarzanie komórek zaznaczonego zakresu przy użyciu pętli	237
Usuwanie wszystkich pustych wierszy	240
Powielanie wierszy	240
Określanie, czy zakres zawiera się w innym zakresie	242
Określanie typu danych zawartych w komórce	242
Odczytywanie i zapisywanie zakresów	243
Lepsza metoda zapisywania danych do zakresu komórek	245
Przenoszenie zawartości tablic jednowymiarowych	246
Przenoszenie zawartości zakresu do tablicy typu Variant	247
Zaznaczanie komórek na podstawie wartości	248
Kopiowanie nieciągłego zakresu komórek	249
Przetwarzanie skorysytów i arkuszy	251
Zapisywanie wszystkich skorysytów	251
Zapisywanie i zamykanie wszystkich skorysytów	251
Ukrywanie wszystkich komórek arkusza poza zaznaczonym zakresem	252
Tworzenie spisu treści zawierającego hiperłącza	253
Synchronizowanie arkuszy	254
Techniki programowania w języku VBA	254
Przełączanie wartości właściwości typu logicznego	255
Wyświetlanie daty i czasu	255

Wyświetlanie czasu w formie przyjaznej dla użytkownika	257
Pobieranie listy czcionek	258
Sortowanie tablicy	259
Przetwarzanie grupy plików	259
Ciekawe funkcje, których możesz użyć w swoich projektach	261
Funkcja FileExists	261
Funkcja FileNameOnly	261
Funkcja PathExists	262
Funkcja RangeNameExists	262
Funkcja SheetExists	263
Funkcja WorkbookIsOpen	263
Pobieranie wartości z zamkniętego skoroszytu	264
Użyteczne, niestandardowe funkcje arkuszowe	265
Funkcje zwracające informacje o formatowaniu komórki	265
Gadający arkusz	267
Wyświetlanie daty zapisania lub wydrukowania pliku	267
Obiekty nadrzędne	268
Zliczanie komórek, których wartości zawierają się pomiędzy dwoma wartościami	269
Wyznaczanie ostatniej niepustej komórki kolumny lub wiersza	269
Czy dany łańcuch tekstu jest zgodny ze wzorcem?	271
Wyznaczanie n-tego elementu łańcucha	272
Zamiana wartości na postać słowną	272
Funkcja wielofunkcyjna	273
Funkcja SHEETOFFSET	274
Zwracanie maksymalnej wartości ze wszystkich arkuszy	274
Zwracanie tablicy zawierającej unikatowe, losowo uporządkowane liczby całkowite	275
Porządkowanie zakresu w losowy sposób	277
Sortowanie zakresów	278
Wywołania funkcji interfejsu Windows API	279
Deklaracje API	279
Określanie skojarzeń plików	280
Pobieranie informacji dotyczących drukarki domyślnej	281
Pobieranie informacji o aktualnej rozdzielczości karty graficznej	282
Odczytywanie zawartości rejestru systemu Windows i zapisywanie w nim danych	283

Część II. Zaawansowane techniki programowania

287

Rozdział 8. Tabele przestawne	289
Przykład prostej tabeli przestawnej	289
Tworzenie tabel przestawnych	290
Analiza zarejestrowanego kodu tworzącego tabelę przestawną	292
Optymalizacja wygenerowanego kodu tworzącego tabelę przestawną	292
Tworzenie złożonych tabel przestawnych	294
Kod tworzący tabelę przestawną	295
Jak działa złożona tabela przestawna?	296
Jednoczesne tworzenie wielu tabel przestawnych	298
Tworzenie odwróconych tabel przestawnych	300

Rozdział 9. Wykresy	303
Podstawowe wiadomości o wykresach	303
Lokalizacja wykresu	303
Rejestrator makr a wykresy	304
Model obiektu Chart	304
Tworzenie wykresów osadzonych na arkuszu danych	306
Tworzenie wykresu na arkuszu wykresu	307
Modyfikowanie wykresów	307
Wykorzystanie VBA do uaktywnienia wykresu	308
Przenoszenie wykresu	309
Wykorzystanie VBA do dezaktywacji wykresu	310
Sprawdzanie, czy wykres został uaktywniony	311
Usuwanie elementów z kolekcji ChartObjects lub Charts	311
Przetwarzanie wszystkich wykresów w pętli	312
Zmiana rozmiarów i wyrównywanie obiektów ChartObject	314
Tworzenie dużej liczby wykresów	315
Eksportowanie wykresów	317
Eksportowanie wszystkich obiektów graficznych	318
Zmiana danych prezentowanych na wykresie	319
Modyfikacja danych wykresu na podstawie aktywnej komórki	319
Zastosowanie języka VBA do identyfikacji zakresu danych prezentowanych na wykresie	321
Wykorzystanie VBA do wyświetlania dowolnych etykiet danych na wykresie	324
Wyświetlanie wykresu w oknie formularza UserForm	327
Zdarzenia związane z wykresami	329
Przykład wykorzystania zdarzeń związanych z wykresami	329
Obsługa zdarzeń dla wykresów osadzonych	331
Przykład zastosowania zdarzeń dla wykresów osadzonych	332
Jak ułatwić sobie pracę z wykresami przy użyciu VBA?	334
Drukowanie wykresów osadzonych na arkuszu	334
Tworzenie wykresów, które nie są połączone z danymi	335
Wykorzystanie zdarzenia MouseOver do wyświetlania tekstu	337
Przewijanie wykresów	339
Tworzenie wykresów przebiegu w czasie	340
Rozdział 10. Interakcje z innymi aplikacjami	345
Automatyzacja zadań w pakiecie Microsoft Office	345
Koncepcja wiązań	345
Przykład prostej automatyzacji	348
Sterowanie bazą danych Access z poziomu Excela	348
Uruchamianie zapytań bazy danych Access z poziomu Excela	348
Uruchamianie makr Accessa z poziomu Excela	349
Sterowanie edytorem Word z poziomu Excela	350
Przesyłanie danych z Excela do dokumentu Worda	350
Symulacja tworzenia korespondencji seryjnej z użyciem Worda	352
Sterowanie programem PowerPoint z poziomu Excela	354
Przesyłanie danych z Excela do prezentacji PowerPoint	354
Przesyłanie wszystkich wykresów z arkusza Excela do prezentacji PowerPoint	355
Zamiana skoroszytu na prezentację PowerPoint	356

Sterowanie programem Outlook z poziomu Excela	357
Wysyłanie aktywnego skoroszytu jako załącznika	357
Wysyłanie wybranego zakresu komórek jako załącznika wiadomości	358
Wysyłanie pojedynczego arkusza jako załącznika wiadomości	359
Wysyłanie wiadomości do wszystkich adresatów z listy kontaktów	360
Uruchamianie innych aplikacji z poziomu Excela	361
Zastosowanie funkcji Shell języka VBA	361
Zastosowanie funkcji ShellExecute interfejsu Windows API	363
Wykorzystanie instrukcji AppActivate	365
Uruchamianie okien dialogowych Panelu sterowania	365
Rozdział 11. Praca z danymi zewnętrznymi i plikami	367
Praca z danymi ze źródeł zewnętrznych	367
Wprowadzenie do zapytań Power Query	367
Etapy tworzenia zapytania	373
Odświeżanie danych z zapytania Power Query	374
Zarządzanie istniejącymi zapytaniem	375
Używanie języka VBA do tworzenia dynamicznych połączeń	375
Przechodzenie w pętli przez wszystkie połączenia skoroszytu	378
Zastosowanie ADO i VBA do pobierania danych ze źródeł zewnętrznych	378
Ciąg połączenia	379
Deklarowanie zestawu rekordów	380
Odwołania do biblioteki obiektów ADO	381
Łączenie wszystkiego razem w kodzie procedury	382
Zastosowanie obiektów ADO w aktywnym skoroszycie	383
Operacje na plikach tekstowych	385
Otwieranie plików tekstowych	386
Odczytywanie plików tekstowych	386
Zapisywanie danych do plików tekstowych	387
Przydzielanie numeru pliku	387
Określanie lub ustawianie pozycji w pliku	387
Instrukcje pozwalające na odczytywanie i zapisywanie plików	388
Przykłady wykonywania operacji na plikach	388
Importowanie danych z pliku tekstowego	388
Eksportowanie zakresu do pliku tekstowego	389
Importowanie pliku tekstowego do zakresu	390
Logowanie wykorzystania Excela	391
Filtrowanie zawartości pliku tekstowego	391
Najczęściej wykonywane operacje na plikach	392
Zastosowanie poleceń języka VBA do wykonywania operacji na plikach	392
Zastosowanie obiektu FileSystemObject	397
Pakowanie i rozpakowywanie plików	399
Pakowanie plików do formatu ZIP	399
Rozpakowywanie plików ZIP	400

Część III. Praca z formularzami UserForm

403

Rozdział 12. Tworzenie własnych okien dialogowych	405
Zanim rozpoczniesz tworzenie formularza UserForm	405
Okno wprowadzania danych	405
Funkcja InputBox języka VBA	406
Metoda Application.InputBox	408
Funkcja MsgBox języka VBA	411
Metoda GetOpenFilename programu Excel	415
Metoda GetSaveAsFilename programu Excel	418
Okno wybierania katalogu	419
Wyświetlanie wbudowanych okien dialogowych Excela	419
Wyświetlanie formularza danych	422
Wyświetlanie formularza wprowadzania danych	422
Wyświetlanie formularza wprowadzania danych za pomocą VBA	423
Rozdział 13. Wprowadzenie do formularzy UserForm	425
Jak Excel obsługuje niestandardowe okna dialogowe	425
Wstawianie nowego formularza UserForm	426
Dodawanie formantów do formularza UserForm	426
Formanty okna Toolbox	427
Formant CheckBox	427
Formant ComboBox	428
Formant CommandButton	428
Formant Frame	428
Formant Image	428
Formant Label	429
Formant ListBox	429
Formant MultiPage	429
Formant OptionButton	429
Formant RefEdit	429
Formant ScrollBar	429
Formant SpinButton	430
Formant TabStrip	430
Formant TextBox	430
Formant ToggleButton	430
Modyfikowanie formantów formularza UserForm	431
Modyfikowanie właściwości formantów	433
Zastosowanie okna Properties	433
Wspólne właściwości	434
Uwzględnienie wymagań użytkowników preferujących korzystanie z klawiatury	436
Wyświetlanie formularza UserForm	438
Zmiana położenia formularza na ekranie	439
Wyświetlanie niemodalnych okien formularzy UserForm	439
Wyświetlanie formularza UserForm na podstawie zmiennej	439
Ładowanie formularza UserForm	440
Procedury obsługi zdarzeń	440

Zamykanie formularza UserForm	440
Przykład tworzenia formularza UserForm	441
Tworzenie formularza UserForm	442
Tworzenie kodu procedury wyświetlającej okno dialogowe	444
Testowanie okna dialogowego	445
Dodawanie procedur obsługi zdarzeń	445
Zakończenie tworzenia okna dialogowego	447
Zdobywanie informacji na temat zdarzeń	447
Zdarzenia związane z formantem SpinButton	449
Współpraca formantu SpinButton z formantem TextBox	450
Odwoływanie się do formantów formularza UserForm	452
Dostosowywanie okna Toolbox do własnych wymagań	454
Dodawanie nowych kart	454
Dostosowywanie lub łączenie formantów	454
Dodawanie nowych formantów ActiveX	455
Tworzenie szablonów formularzy UserForm	456
Lista kontrolna tworzenia i testowania formularzy UserForm	457
Rozdział 14. Przykłady formularzy UserForm	459
Tworzenie formularza UserForm pełniącego funkcję menu	459
Zastosowanie przycisków CommandButton w formularzach UserForm	459
Zastosowanie formantów ListBox w formularzach UserForm	460
Zaznaczanie zakresów przy użyciu formularza UserForm	461
Tworzenie okna powitalnego	463
Wyłączanie przycisku Zamknij formularza UserForm	465
Zmiana wielkości formularza UserForm	465
Powiększanie i przewijanie arkusza przy użyciu formularza UserForm	467
Zastosowania formantu ListBox	469
Tworzenie listy elementów formantu ListBox	469
Identyfikowanie zaznaczonego elementu listy formantu ListBox	473
Identyfikowanie wielu zaznaczonych elementów listy formantu ListBox	474
Wiele list w jednym formancie ListBox	475
Przenoszenie elementów listy formantu ListBox	476
Zmiana kolejności elementów listy formantu ListBox	477
Wielokolumnowe formanty ListBox	479
Zastosowanie formantu ListBox do wybierania wierszy arkusza	481
Uaktywnianie arkusza za pomocą formantu ListBox	483
Filtrowanie zawartości listy za pomocą pola tekstowego	485
Zastosowanie formantu MultiPage na formularzach UserForm	487
Korzystanie z formantów zewnętrznych	488
Animowanie etykiet	490
Rozdział 15. Zaawansowane techniki korzystania z formularzy UserForm	493
Niemodalne okna dialogowe	493
Wyświetlanie wskaźnika postępu zadania	497
Tworzenie samodzielnego wskaźnika postępu zadania	498
Wyświetlanie wskaźnika postępu zintegrowanego z formularzem UserForm	501
Tworzenie innych, niegraficznych wskaźników postępu	504

Tworzenie kreatorów	506
Konfigurowanie formantu MultiPage w celu utworzenia kreatora	507
Dodawanie przycisków do formularza UserForm kreatora	508
Programowanie przycisków kreatora	508
Zależności programowe w kreatorach	509
Wykonywanie zadań za pomocą kreatorów	511
Emulacja funkcji MsgBox	512
Emulacja funkcji MsgBox: kod funkcji MsgBox	512
Jak działa funkcja MsgBox	514
Wykorzystanie funkcji MsgBox do emulacji funkcji MsgBox	515
Formularz UserForm z formantami, których położenie można zmieniać	515
Formularz UserForm bez paska tytułowego	517
Symulacja paska narzędzi za pomocą formularza UserForm	518
Emulowanie panelu zadań za pomocą formularza UserForm	520
Formularze UserForm z możliwością zmiany rozmiaru	521
Obsługa wielu przycisków formularza UserForm za pomocą jednej procedury obsługi zdarzeń ...	525
Wybór koloru za pomocą formularza UserForm	528
Wyświetlanie wykresów na formularzach UserForm	529
Zapisywanie wykresu w postaci pliku GIF	530
Modyfikacja właściwości Picture formantu Image	530
Tworzenie półprzezroczystych formularzy UserForm	531
Układanka na formularzu UserForm	532
Poker na formularzu UserForm	534

Część IV. Tworzenie aplikacji

535

Rozdział 16. Tworzenie i wykorzystanie dodatków	537
Czym są dodatki?	537
Porównanie dodatku ze standardowym skoroszytem	537
Po co tworzy się dodatki?	538
Menedżer dodatków Excela	540
Tworzenie dodatków	541
Przykład tworzenia dodatku	542
Tworzenie opisu dla dodatku	544
Tworzenie dodatku	544
Instalowanie dodatku	545
Testowanie dodatków	546
Dystrybucja dodatków	546
Modyfikowanie dodatku	546
Porównanie plików XLAM i XLSM	548
Pliki XLAM — przynależność do kolekcji z poziomu VBA	548
Widoczność plików XLSM i XLAM	548
Arkusze i wykresy w plikach XLSM i XLAM	549
Dostęp do procedur VBA w dodatku	550
Przetwarzanie dodatków za pomocą kodu VBA	553
Właściwości obiektu AddIn	554
Korzystanie z dodatku jak ze skoroszytu	557
Zdarzenia związane z obiektami AddIn	558

Optymalizacja wydajności dodatków	558
Problemy z dodatkami	559
Upewnij się, że dodatek został zainstalowany	559
Odwoływanie się do innych plików z poziomu dodatku	561
Rozdział 17. Praca ze Wstążką	563
Wprowadzenie do pracy ze Wstążką	563
Dostosowywanie Wstążki do własnych potrzeb	565
Dodawanie nowych przycisków do Wstążki	565
Dodawanie przycisków do paska narzędzi Szybki dostęp	568
Ograniczenia w dostosowywaniu Wstążki	569
Modyfikowanie Wstążki za pomocą kodu RibbonX	570
Dodawanie przycisków do istniejącej karty	570
Dodawanie pola wyboru do istniejącej karty	576
Demo formantów Wstążki	578
Przykład użycia formantu DynamicMenu	585
Więcej wskazówek dotyczących modyfikacji Wstążki	587
VBA i Wstążka	589
Dostęp do poleceń Wstążki	589
Praca ze Wstążką	589
Aktywowanie karty	592
Tworzenie pasków narzędzi w starym stylu	592
Ograniczenia funkcjonalności tradycyjnych pasków narzędzi w Excelu 2007 i nowszych wersjach	592
Kod tworzący pasek narzędzi	593
Rozdział 18. Praca z menu podręcznym	597
Obiekt CommandBar	597
Rodzaje obiektów CommandBar	597
Wyświetlanie menu podręcznych	598
Odwołania do elementów kolekcji CommandBars	598
Odwołania do formantów obiektu CommandBar	599
Właściwości formantów obiektu CommandBar	601
Wyświetlanie wszystkich elementów menu podręcznego	601
Wykorzystanie VBA do dostosowywania menu podręcznego	603
Menu podręczne w jednodokumentowym interfejsie Excela	603
Resetowanie menu podręcznego	603
Wyłączanie menu podręcznego	606
Wyłączanie wybranych elementów menu podręcznego	606
Dodawanie nowego elementu do menu podręcznego Cell	606
Dodawanie nowego podmenu do menu podręcznego	608
Ograniczanie zasięgu modyfikacji menu podręcznego do jednego skoroszytu	610
Menu podręczne i zdarzenia	611
Automatyczne tworzenie i usuwanie menu podręcznego	611
Wyłączanie lub ukrywanie elementów menu podręcznego	612
Tworzenie kontekstowych menu podręcznych	612

Rozdział 19. Tworzenie systemów pomocy w aplikacjach	615
Systemy pomocy w aplikacjach Excela	615
Systemy pomocy wykorzystujące komponenty Excela	617
Wykorzystanie komentarzy do tworzenia systemów pomocy	617
Wykorzystanie pól tekstowych do wyświetlania pomocy	618
Wykorzystanie arkusza do wyświetlania tekstu pomocy	620
Wyświetlanie pomocy w oknie formularza UserForm	621
Wyświetlanie pomocy w oknie przeglądarki sieciowej	624
Zastosowanie plików w formacie HTML	624
Zastosowanie plików w formacie MHTML	625
Wykorzystanie systemu HTML Help	626
Wykorzystanie metody Help do wyświetlania pomocy w formacie HTML Help	628
Łączenie pliku pomocy z aplikacją	629
Przypisanie tematów pomocy do funkcji VBA	630
Rozdział 20. Moduły klas	631
Czym jest moduł klasy?	631
Wbudowane moduły klas	632
Niestandardowe moduły klas	632
Tworzymy klasę NumLock	633
Wstawianie modułu klasy	634
Dodawanie kodu VBA do modułu klasy	634
Zastosowanie klasy NumLock	635
Programowanie właściwości, metod i zdarzeń	636
Programowanie właściwości obiektów	636
Programowanie metod obiektów	638
Zdarzenia modułu klasy	638
Zdarzenia obiektu QueryTable	639
Tworzenie klas przechowujących inne klasy	642
Tworzenie klas CSalesRep oraz CSalesReps	642
Tworzenie klas CInvoice oraz CInvoices	644
Wypełnianie klasy nadrzędnej obiektami	645
Obliczanie prowizji	646
Rozdział 21. Problemy z kompatybilnością aplikacji	649
Co to jest kompatybilność?	649
Rodzaje problemów ze zgodnością	650
Unikaj używania nowych funkcji i mechanizmów	651
Czy aplikacja będzie działać na komputerach Macintosh?	653
Praca z 64-bitową wersją Excela	654
Tworzenie aplikacji dla wielu wersji narodowych	655
Aplikacje obsługujące wiele języków	656
Obsługa języka w kodzie VBA	657
Wykorzystanie właściwości lokalnych	658
Identyfikacja ustawień systemu	658
Ustawienia daty i godziny	660

Dodatki	661
Dodatek A. Instrukcje i funkcje VBA	663
Skorowidz	671

Praca z danymi zewnętrznymi i plikami

W TYM ROZDZIALE:

- Praca z danymi zewnętrznymi
- Zastosowanie obiektów ActiveX do pobierania danych ze źródeł zewnętrznych
- Najczęściej wykonywane operacje na plikach
- Praca z plikami tekstowymi

Praca z danymi ze źródeł zewnętrznych

Dane zewnętrzne są dokładnie tym, co sugeruje ich nazwa, czyli danymi, które nie są zlokalizowane w skoroszycie Excela, z którym pracujesz. Przykładami zewnętrznych źródeł danych mogą być pliki tekstowe, tabele w bazach danych takich jak Access czy SQL Server lub nawet inne skoroszyty Excela.

Istnieje bardzo wiele sposobów na załadowanie danych zewnętrznych do Excela. Jeżeli weźmiemy pod uwagę możliwości zarówno interfejsu użytkownika, jak i języka VBA, to okaże się, że liczba dostępnych technik importowania danych zewnętrznych do skoroszytów Excela jest zdecydowanie zbyt duża, aby omówić je wszystkie w jednym czy nawet w kilku przeznaczonych jedynie im rozdziałach. Dlatego w tym rozdziale skoncentrujemy się na kilku starannie wybranych technikach, których możesz używać w większości standardowych scenariuszy importu danych i które nie kryją w sobie zbyt wielu pułapek i utrudnień.

Pierwsza z omawianych technik wykorzystuje ciekawy mechanizm Excela o nazwie Power Query.

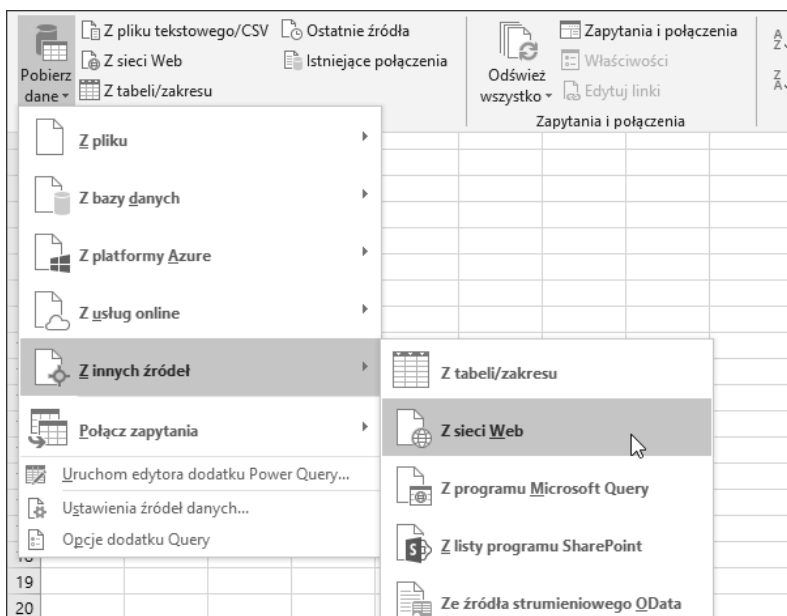
Wprowadzenie do zapytań Power Query

Zapytania Power Query to intuicyjny mechanizm pozwalający na pobieranie danych z wielu różnych źródeł, wykonywanie złożonych operacji na tych danych, a następnie ładowanie ich do skoroszytu.

Aby się przekonać, że zapytania Power Query nie są takie straszne, jak by się mogło na pierwszy rzut oka wydawać, przejdziemy od razu do prostego przykładu. Wyobraź sobie, że musisz zaimportować ceny akcji firmy Microsoft Corporation za pomocą usługi Yahoo Finance i załadować je do skoroszytu Excela. W takim scenariuszu musisz wykonać odpowiednie zapytanie internetowe, aby pobrać potrzebne dane z serwisu Yahoo Finance.

Aby utworzyć zapytanie, wykonaj następujące polecenia opisane poniżej:

1. W nowym skoroszycie programu Excel przejdź na kartę *Dane* i z grupy *Pobieranie i przekształcanie danych* wybierz polecenie *Pobierz dane/Z innych źródeł/Z sieci Web* (zobacz rysunek 11.1).



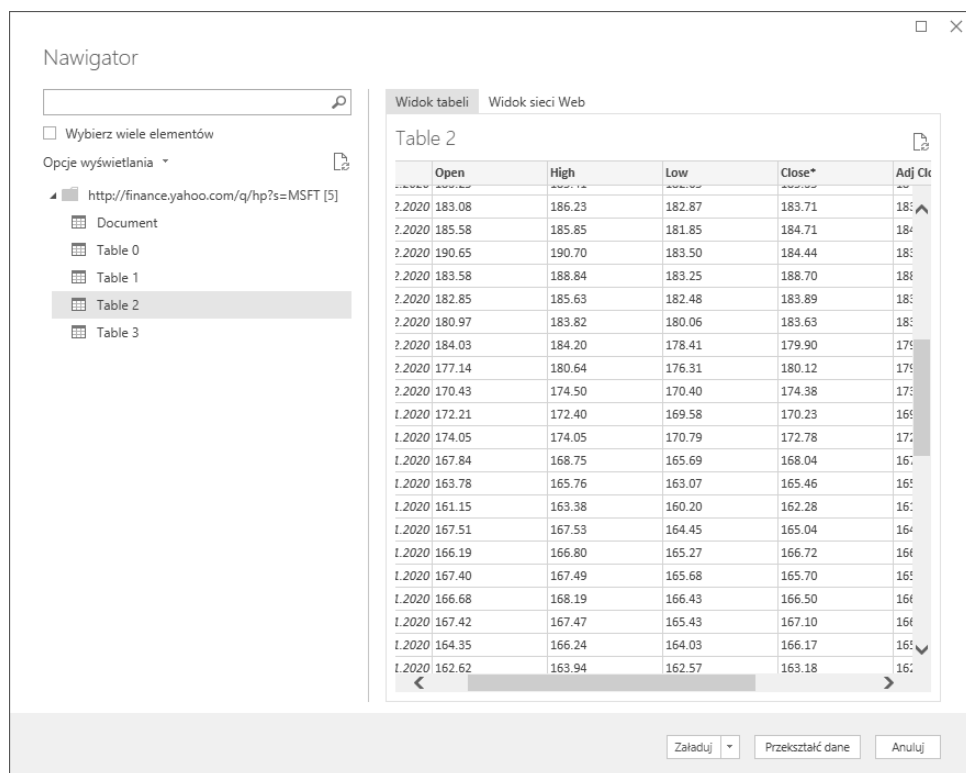
RYSUNEK 11.1. Tworzenie zapytania Power Query

2. W oknie dialogowym, które pojawi się na ekranie, wpisz adres URL potrzebnych danych i naciśnij przycisk *OK*; w naszym przypadku będzie to `http://finance.yahoo.com/q/hp?s=MSFT` (zobacz rysunek 11.2).



RYSUNEK 11.2. Wpisywanie adresu URL danych

3. Po krótkiej chwili na ekranie pojawi się okienko nawigatora pokazane na rysunku 11.3. Wybierz źródło danych, które chcesz pobrać. Aby zobaczyć podgląd danych, kliknij wybraną tabelę. W naszym przypadku potrzebne dane historyczne dotyczące notowań akcji zawiera tabela z etykietą *Table 2*, dlatego zaznacz tę tabelę, klikając jej nazwę lewym przyciskiem myszy, a następnie naciśnij przycisk *Przekształć dane*.



RYСУNEK 11.3. Wybierz żądane źródło danych i naciśnij przycisk **Przekształć dane**



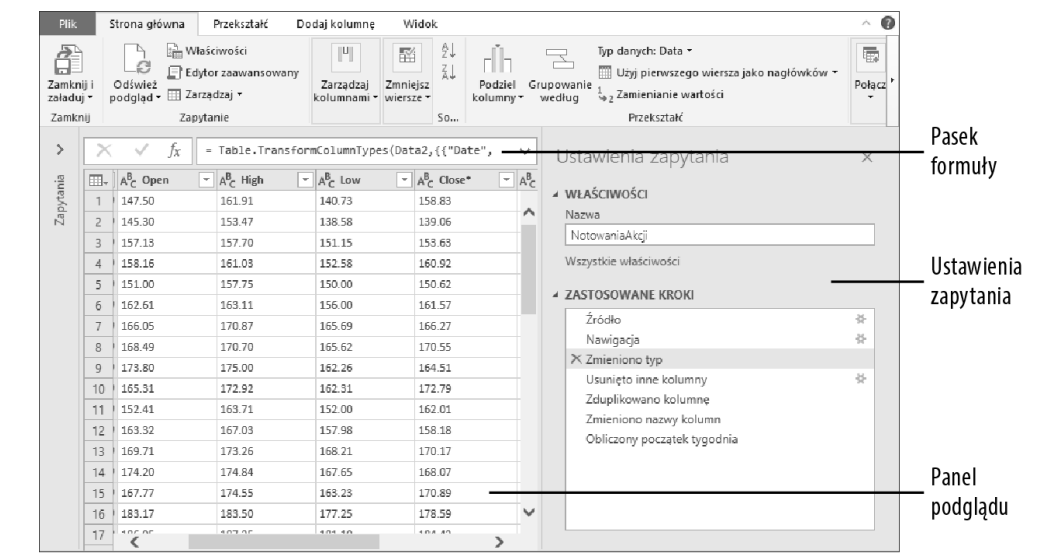
Być może zauważyłeś, że w okienku *Nawigator*, pokazanym na rysunku 11.3, znajduje się również przycisk *Załaduj* (obok przycisku *Przekształć dane*). Przycisk *Załaduj* pozwala pominąć edycję i zaimportować dane docelowe bez żadnych zmian. Jeżeli masz pewność, że nie będziesz musiał w żaden sposób przekształcać ani modyfikować pobieranych danych, możesz po prostu nacisnąć przycisk *Załaduj* i zaimportować dane bezpośrednio do modelu danych lub arkusza w skoroszybie.



Excel posiada jeszcze inny przycisk polecenia *Z sieci Web*, który znajduje się na karcie *Dane* po prawej stronie przycisku polecenia *Pobierz dane*. To niefortunnie zduplikowane (i zlokalizowane) polecenie jest w rzeczywistości starszym narzędziem do pobierania danych z internetu, które można znaleźć już w Excelu 2000 i wszystkich nowszych wersjach tego programu.

Polecenie *Z sieci Web* w wersji Power Query (znajdujące się w menu rozwijanym polecenia *Pobierz dane*) znacznie wykracza swoją funkcjonalnością poza zwykłe pobieranie danych z internetu. Zapytanie Power Query jest w stanie pobierać dane z rozbudowanych stron internetowych i może je odpowiednio przekształcać w zależności od potrzeb użytkownika. Z tego względu, pobierając dane z internetu, powinieneś zawsze upewnić się, że używasz prawidłowego polecenia.

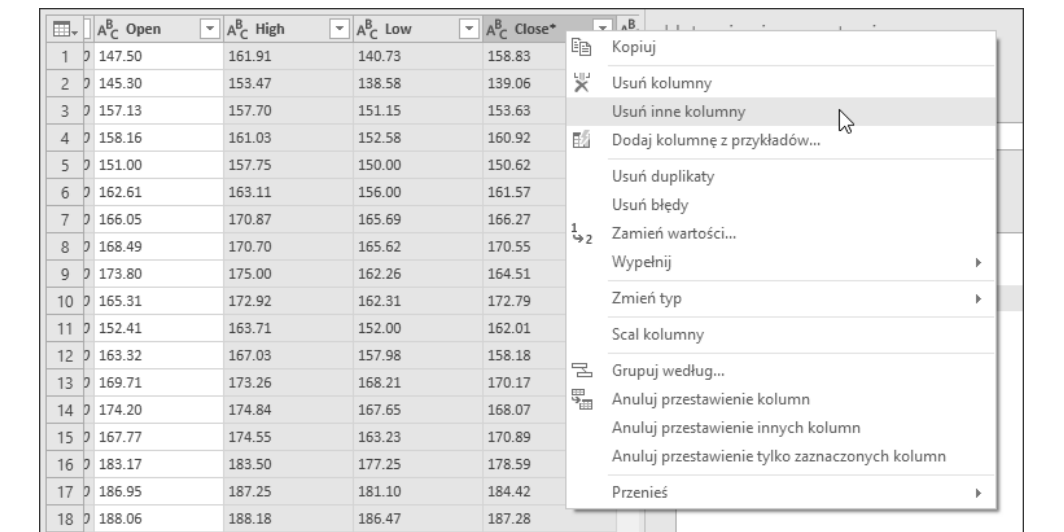
Po naciśnięciu przycisku *Przekształć dane* na ekranie pojawi się okno edytora zapytań Power Query, które posiada własną Wstążkę i panel pokazujący podgląd pobieranych danych (zobacz rysunek 11.4). Korzystając z tego okna, możesz zdefiniować określone działania w celu modyfikowania, czyszczenia i przekształcania danych przed zaimportowaniem do arkusza Excela.



RYSUNEK 11.4. Edytor Power Query pozwala na modyfikowanie, czyszczenia i przekształcanie danych

W edytorze Power Query możesz pracować z każdą kolumną danych osobno, definiując niezbędne działania, które zapewnią Ci potrzebne dane oraz ich strukturę. Więcej szczegółowych informacji na ten temat znajdziesz w dalszej części tego rozdziału, a teraz skoncentrujemy się na tym, jak pobrać notowania akcji firmy Microsoft Corporation z ostatnich 30 dni.

4. Usuń wszystkie niepotrzebne kolumny, klikając je po kolei prawym przyciskiem myszy i wybierając z menu podręcznego polecenie *Usuń* (oprócz pola *Date* będą Ci potrzebne jedynie kolumny *High*, *Low* i *Close*). Możesz też wcisnąć i przytrzymać klawisz *Ctrl*, zaznaczyć kolumny, które chcesz zachować, a następnie kliknąć prawym przyciskiem myszy dowolną z wybranych kolumn i z menu podręcznego wybrać polecenie *Usuń inne kolumny* (zobacz rysunek 11.5).



RYSUNEK 11.5. Wybierz kolumny, które chcesz zachować, i z menu podręcznego wybierz opcję *Usuń inne kolumny*

5. Upewnij się, że pola *High*, *Low* i *Close* są odpowiednio sformatowane jako liczby. Aby to zrobić, przytrzymaj klawisz *Ctrl* na klawiaturze, zaznacz trzy kolumny z danymi, kliknij prawym przyciskiem myszy jeden z nagłówków kolumn, a następnie z menu podręcznego wybierz polecenie *Zmień typ / Liczba dziesiętna*. Po wykonaniu tej czynności możesz zauważyć, że niektóre wiersze zawierają słowo *Error*. Są to wiersze zawierające wartości tekstowe, których nie można przekonwertować do postaci liczbowej.
6. W razie potrzeby usuń wiersze zawierające błędy, wybierając z menu podręcznego polecenie *Usuń błędy*, tak jak to zostało pokazane na rysunku 11.6.

	Date	High	Low	Close
1	13.03.2020	161.91		
2	12.03.2020	153.47		
3	11.03.2020	157.70		
4	10.03.2020	161.03		
5	09.03.2020	157.75		
6	06.03.2020	163.11		
7	05.03.2020	170.87		
8	04.03.2020	170.70		
9	03.03.2020	175.00		
10	02.03.2020	172.92		
11	28.02.2020	163.71		
12	27.02.2020	167.03		

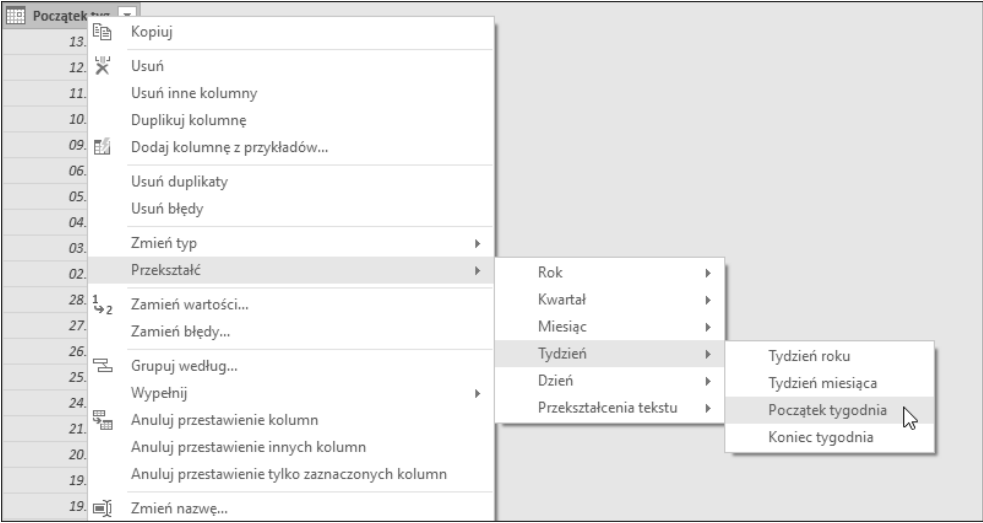
RYСУNEK 11.6. Z menu podręcznego możesz wybrać różne polecenia, które zostaną zastosowane do tabeli danych (na przykład polecenie *Usuń błędy*)

7. Po usunięciu wszystkich błędów dodaj nowe pole po nazwie *Początek tygodnia*, w którym będzie wyświetlany dzień rozpoczynający dany tydzień. Aby to zrobić, kliknij prawym przyciskiem myszy pole *Date* i wybierz opcję *Duplikuj kolumnę*. Do podglądu danych zostanie dodana nowa kolumna o nazwie *Date — Copy*.
8. Kliknij prawym przyciskiem myszy nowo dodaną kolumnę i z menu podręcznego wybierz polecenie *Zmień nazwę*, a następnie wpisz nową nazwę kolumny (*Początek tygodnia*).
9. Kliknij prawym przyciskiem myszy właśnie utworzoną kolumnę *Początek tygodnia* i z menu podręcznego wybierz polecenie *Przekształć/Tydzień/Początek tygodnia*, jak pokazano na rysunku 11.7. Excel przekształci daty tak, aby wyświetlić początek tygodnia dla kolejnych dat.
10. Po zakończeniu konfigurowania zapytania Power Query zapisz je i załaduj wyniki działania. Aby to zrobić, naciśnij przycisk *Zamknij i załaduj*, znajdujący się na karcie *Strona główna* wstążki edytora Power Query, i wybierz jedną z dwóch dostępnych opcji: *Zamknij i załaduj* lub *Zamknij i załaduj do...*

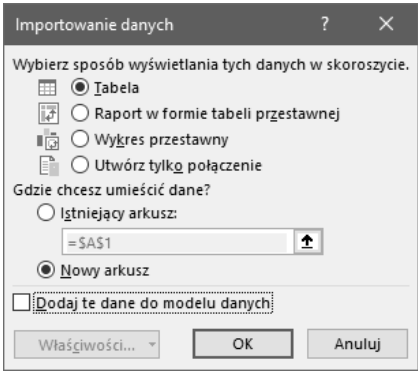
Opcja *Zamknij i załaduj* powoduje zapisanie zapytania i załadowanie wyników jego działania w nowym arkuszu w skoroszycie w postaci tabeli programu Excel. Opcja *Zamknij i wczytaj do* powoduje wyświetlenie okna dialogowego *Importowanie danych*, które zostało pokazane na rysunku 11.8. Możesz w nim określić, czy pobierane dane zostaną załadowane do arkusza, czy do wewnętrznego modelu danych oraz w jaki sposób będą wyświetlane.

Okno dialogowe *Importowanie danych* umożliwia również zapisanie zapytania jako połączenia ze źródłem danych, co oznacza, że będziesz mógł użyć zapytania bezpośrednio w różnych procesach w pamięci bez potrzeby wyświetlania wyników w arkuszu.

11. Wybierz opcję *Nowy arkusz*, aby wyświetlić wyniki jako tabelę w nowym arkuszu w aktywnym skoroszycie.



RYSUNEK 11.7. Za pomocą Edytora Power Query możesz dokonywać transformacji danych, takich jak np. wyświetlanie pierwszego dnia tygodnia dla określonej daty



RYSUNEK 11.8. Okno dialogowe Importowanie danych zapewnia pełną kontrolę nad sposobem wykorzystania wyników zapytań

Po wykonaniu zapytania dane zostaną pobrane i wyświetlone w postaci tabeli, tak jak to zostało pokazane na rysunku 11.9; takiej tabeli możemy teraz użyć na przykład do utworzenia potrzebnej tabeli przestawnej.

	A	B	C	D	E
1	Date	High	Low	Close*	Początek tyg.
2	13.03.2020	161.91	140.73	158.83	09.03.2020
3	12.03.2020	153.47	138.58	139.06	09.03.2020
4	11.03.2020	157.70	151.15	153.63	09.03.2020
5	10.03.2020	161.03	152.58	160.92	09.03.2020
6	09.03.2020	157.75	150.00	150.62	09.03.2020
7	06.03.2020	163.11	156.00	161.57	02.03.2020
8	05.03.2020	170.87	165.69	166.27	02.03.2020
9	04.03.2020	170.70	165.62	170.55	02.03.2020
10	03.03.2020	175.00	162.26	164.51	02.03.2020

RYSUNEK 11.9. Wyniki działania zapytania pobierającego dane z internetu zostały przekształcone, umieszczone w tabeli Excela; dane są gotowe do użycia w tabeli przestawnej

Teraz powinieneś poświęcić chwilę, aby dobrze zrozumieć i docenić to, co Power Query pozwoliło Ci przed chwilą zrobić. Za pomocą kilku kliknięć przeszukałeś internet, znalazłeś podstawowy zestaw danych, przetworzyłeś je, aby zachować tylko te kolumny, których potrzebujesz, a nawet zmodyfikowałeś je tak, aby dodać do podstawowych danych kolumnę zawierającą pierwszy dzień tygodnia. Właśnie o to chodzi w zapytaniach Power Query: umożliwienie łatwego pobierania, filtrowania i przekształcania danych bez konieczności posiadania jakiegokolwiek doświadczenia w programowaniu.

Najlepsze jest to, że zapytania Power Query mają możliwość łączenia się z szeroką gamą źródeł danych. Niezależnie od tego, czy chcesz pobierać dane z zewnętrznej strony internetowej, pliku tekstowego, systemu baz danych, Facebooka, czy serwisu internetowego, Power Query może zaspokoić większość, jeśli nie wszystkie Twoje potrzeby dotyczące źródeł danych. Wszystkie dostępne typy połączeń możesz wyświetlić, naciskając przycisk *Pobierz dane* znajdujący się na karcie *Dane*.

Zapytania Power Query oferują możliwość pobierania informacji z szerokiej gamy źródeł danych:

- *Z pliku* — pobiera dane z określonych plików Excela, plików tekstowych, plików CSV, plików XML lub całych folderów.
- *Z bazy danych* — pobiera dane z baz danych, takich jak Microsoft Access, SQL Server czy SQL Server Analysis Services.
- *Z platformy Azure* — pobiera dane z usługi Azure Cloud Services firmy Microsoft.
- *Z usług online* — pobiera dane z aplikacji w chmurze, takich jak Facebook, Salesforce czy Microsoft Dynamics online.
- *Z innych źródeł* — pobiera dane z szerokiej gamy innych źródeł, takich jak sieć WWW, chmury danych czy źródła danych ODBC.

Etapy tworzenia zapytania

Power Query używa własnego języka formuł, znanego jako język **M**, do kodowania Twoich zapytań. Podobnie jak w przypadku rejestrowania makr, każde działanie podejmowane podczas pracy z Power Query powoduje zapisywanie kodu w danym kroku zapytania. Poszczególne etapy zapytania są zarejestrowane w kodzie **M**, który umożliwia powtarzanie działań przy każdym odświeżeniu danych zapytania Power Query.

Możesz wyświetlić kolejne etapy zapytania, aktywując w oknie edytora Power Query panel *Ustawienia zapytania* (patrz rysunek 11.10). Aby to zrobić, wystarczy kliknąć polecenie *Ustawienia zapytania*, znajdujące się na karcie *Widok* Wstążki.

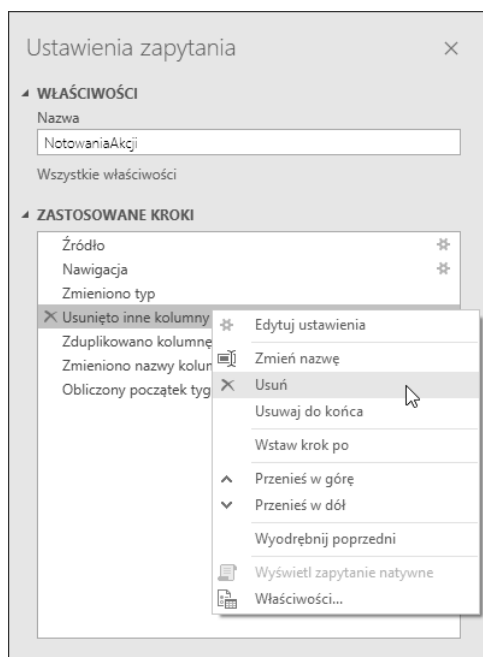
Zwróć uwagę na pokazane na rysunku 11.10 pole *Nazwa*, za pomocą którego możesz nadać danemu zapytaniu przyjazną nazwę. W naszym przykładzie nadaliśmy zapytaniu nazwę *NotowaniaAkcji*.

Możesz także włączyć wyświetlanie paska formuły, w którym zawarte są polecenia języka **M** dla wybranego kroku. Aby to zrobić, powinieneś w edytorze zapytań przejść na kartę *Widok* i włączyć opcję *Pasek formuły*, znajdującą się w grupie opcji *Układ*.

Każdy krok zapytania reprezentuje działanie podjęte w celu uzyskania dostępu do danych. Aby na pasku formuły Power Query wyświetlić kod w języku **M**, powinieneś po prostu kliknąć dowolny krok zapytania. Na przykład kliknięcie kroku o nazwie *Usuń błędy* spowoduje wyświetlenie na pasku formuły kodu tego kroku.



Po kliknięciu wybranego kroku zapytania dane wyświetlane w okienku podglądu są obrazem danych aż do klikniętego kroku. Na przykład kliknięcie pokazanego na rysunku 11.10 kroku wykonanego przed wykonaniem polecenia *Usuń inne kolumny* pozwala zobaczyć, jak wyglądały dane przed usunięciem niepotrzebnych kolumn.



RYSUNEK 11.10. Kolejnymi krokami zapytania można zarządzać w sekcji Zastosowane kroki w panelu Ustawienia zapytania

Odświeżanie danych z zapytania Power Query

Powinieneś zawsze pamiętać, że dane pobierane za pomocą zapytań Power Query nie są w żaden sposób powiązane z danymi źródłowymi użytymi do ich wyodrębnienia. Innymi słowy, tabela danych uzyskana za pomocą zapytania Power Query jest jedynie migawką danych dostępnych w chwili wykonania zapytania. Innymi słowy, gdy zmieniają się dane źródłowe, zapytania Power Query nie będą automatycznie nadążać za zmianami; musisz ręcznie odświeżać swoje dane.

Jeżeli zdecydujesz się załadować wyniki Power Query do tabeli Excela w istniejącym skoroszybie, możesz je ręcznie odświeżyć, klikając tabelę prawym przyciskiem myszy i wybierając opcję *Odśwież*.

Jeżeli zdecydujesz się załadować dane z zapytania Power Query do wewnętrznego modelu danych, to aby odświeżyć dane, musisz przejść na kartę *Dane*, wybrać polecenie *Zapytania i połączenia* znajdujące się w grupie opcji *Zapytania i połączenia*, a następnie kliknąć prawym przyciskiem myszy wybrane zapytanie docelowe w panelu zadań i z menu podręcznego wybrać opcję *Odśwież*.

Aby nieco bardziej zautomatyzować odświeżanie zapytań, możesz skonfigurować źródła danych, aby automatycznie odświeżały dane z zapytań Power Query. Aby to zrobić, wykonaj następujące kroki:

1. Przejdź na kartę *Dane* na wstążce programu Excel i wybierz polecenie *Zapytania i połączenia*. Na ekranie pojawi się panel *Zapytania i połączenia*.
2. Kliknij prawym przyciskiem myszy zapytanie Power Query, które chcesz odświeżyć, a następnie wybierz opcję *Właściwości*.
3. Na ekranie pojawi się okno dialogowe *Właściwości zapytania*. Przejdź na kartę *Użycie*.
4. Ustaw opcje odświeżania wybranego zapytania zgodnie z potrzebami:
 - **Odśwież co n minut** — zaznaczenie tej opcji powoduje, że program Excel automatycznie odświeża dane wybranego zapytania co określoną liczbę minut. Excel odświeży dane ze wszystkich tabel związanych z tym zapytaniem.

- **Odśwież dane podczas otwierania pliku** — zaznaczenie tej opcji informuje Excela o konieczności odświeżenia danych z wybranego zapytania automatycznie po każdorazowym otwarciu skoroszytu. Excel odświeży dane ze wszystkich tabel powiązanych z tym zapytaniem za każdym razem, kiedy skoroszyt zostanie otwarty.

Wspomniane wyżej opcje odświeżania są bardzo przydatne, gdy chcesz mieć pewność, że zawsze korzystasz z najnowszych, najbardziej aktualnych danych. Oczywiście ustawienie tych opcji nie wyklucza możliwości ręcznego odświeżenia danych.

Zarządzanie istniejącymi zapytaniami

Gdy dodasz różne zapytania do skoroszytu, będziesz potrzebować sposobu na zarządzanie nimi. Excel rozwiązuje ten problem za pomocą panelu *Zapytania i połączenia*, który umożliwia edycję, duplikowanie, odświeżanie i ogólne zarządzanie wszystkimi zapytaniami utworzonymi w skoroszytcie. Aby aktywować panel *Zapytania i połączenia*, powinieneś przejść na kartę *Dane* wstążki Excela i wybrać polecenie *Zapytania i połączenia*.

Kiedy panel pojawi się na ekranie, odszukaj zapytanie, z którym chcesz pracować, a następnie kliknij je prawym przyciskiem myszy i z menu podręcznego wybierz jedno z następujących poleceń:

- *Edytuj* — otwiera edytor Power Query, w którym możesz modyfikować ustawienia zapytania.
- *Usuń* — usuwa wybrane zapytanie.
- *Odśwież* — odświeża dane w wybranym zapytaniu.
- *Załaduj do* — aktywuje okno dialogowe *Importowanie danych*, w którym możesz określić, gdzie będą używane wyniki wybranego zapytania.
- *Duplikuj* — tworzy kopię zapytania.
- *Odwołanie* — tworzy nowe zapytanie, które odwołuje się do wyników działania oryginalnego zapytania.
- *Scalanie* — scala wybrane zapytanie z innym zapytaniem w skoroszytcie, dopasowując określone kolumny.
- *Dołączanie* — dołącza wyniki innego zapytania w skoroszytcie do wybranego zapytania.
- *Eksportuj plik połączenia* — tworzy plik *.odc*, pozwalający na przeniesienie lub udostępnienie wybranego zapytania.
- *Przenieś do grupy* — przenosi wybrane zapytanie do grupy logicznej; takie grupy pozwalają na lepszą organizację zapytań.
- *Przenieś w górę* — przesuwa wybrane zapytanie w górę listy zapytań w panelu *Zapytania i połączenia*.
- *Przenieś w dół* — przesuwa wybrane zapytanie w dół listy zapytań w panelu *Zapytania i połączenia*.
- *Pokaż wartość szczytową* — wyświetla podgląd wyników działania wybranego zapytania.
- *Właściwości* — pozwala na zmianę nazwy zapytania i ustawienie jego wielu różnych właściwości.

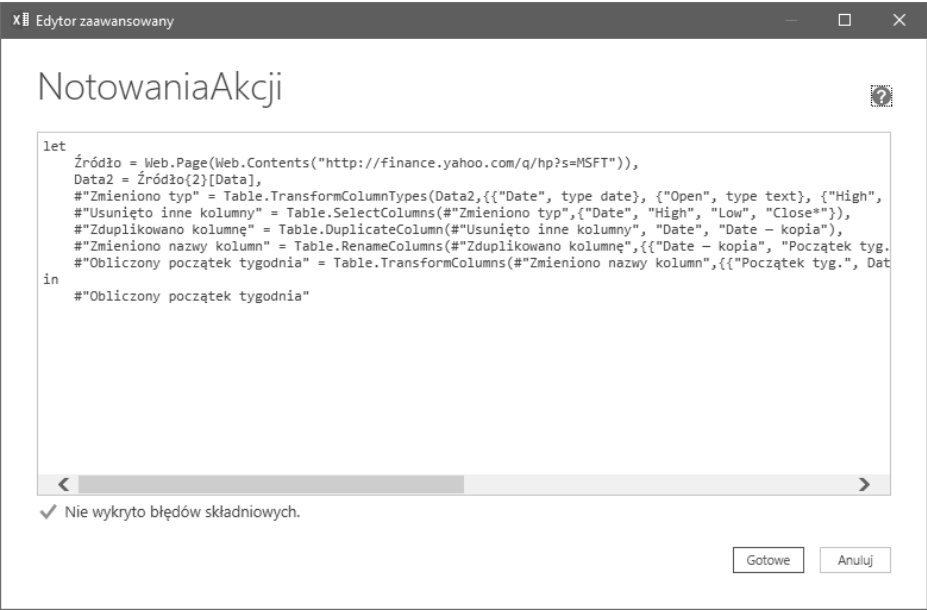
Panel *Zapytania i połączenia* jest szczególnie przydatny, gdy skoroszyt zawiera kilka zapytań. Możemy go traktować jak coś w rodzaju spisu treści, który pozwala łatwo znaleźć i wchodzić w interakcje z poszczególnymi zapytaniami w skoroszytcie.

Używanie języka VBA do tworzenia dynamicznych połączeń

Budując niestandardowe zapytanie Power Query, zasadniczo nie robisz nic poza rejestrowaniem składni potrzebnej do zwrócenia pożądanego wyniku. Kod dowolnego zapytania Power Query może zostać skopiowany z okna zaawansowanego edytora zapytań, a następnie użyty w języku VBA.

Aby przywołać na ekran zaawansowany edytor zapytań, powinieneś w oknie edytora Power Query przejść na kartę *Widok* i wybrać polecenie *Edytor zaawansowany*.

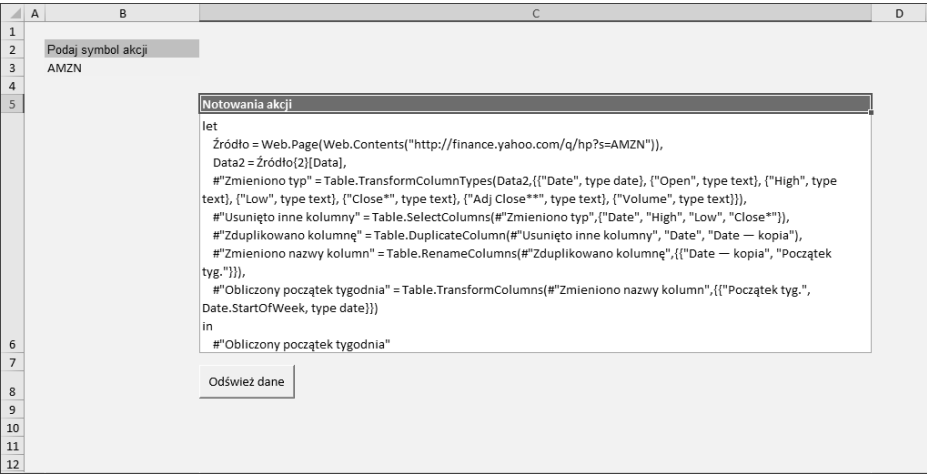
Jeżeli poprawnie wykonałeś pierwsze ćwiczenie, okno zaawansowanego edytora zapytań powinno wyglądać mniej więcej tak, jak to zostało pokazane na rysunku 11.11.



RYSUNEK 11.11. Okno zaawansowanego edytora zapytań

Ogromną zaletą takiego rozwiązania jest to, że nie musisz być ekspertem programowania w języku M, aby za pomocą VBA dynamicznie tworzyć i budować złożone zapytania pobierające dane ze źródeł zewnętrznych.

Na przykład na rysunku 11.12 pokazano arkusz, w którym w komórce C6 można wybrać symbol akcji, aby zmienić składnię zapytania Power Query pobierającego notowania giełdowe wybranej spółki. Kliknięcie przycisku *Odśwież dane* spowoduje przebudowanie zapytania Power Query z uwzględnieniem nowej składni.



RYSUNEK 11.12. Wyznaczona komórka przechowuje kryterium wyboru zapytania

Poniższe makro używa obiektów `Workbook.Query` i `Workbook.Connection` do przebudowania zapytania na podstawie podanego kryterium:

```
Sub RefreshPowerQuery()
    Dim Qry As WorkbookQuery
    Dim QryName As String
    Dim QrySyntax As String
    Dim QryDesc As String
    Dim OutputSheet As Worksheet
    Dim ws As Worksheet

    ' Ustawianie zmiennych
    QryName = ThisWorkbook.Sheets("Query Changer").Range("C5").Value
    QrySyntax = ThisWorkbook.Sheets("Query Changer").Range("C6").Value
    QryDesc = ThisWorkbook.Sheets("Query Changer").Range("C5").Value

    ' Usuwanie istniejącego zapytania
    For Each Qry In ThisWorkbook.Queries
        If Qry.Name = QryName Then
            Set Qry = ThisWorkbook.Queries(QryName)
            Qry.Delete
        End If
    Next Qry

    ' Dodawanie nowego zapytania
    Set Qry = ThisWorkbook.Queries.Add(QryName, QrySyntax, QryDesc)

    ' Usuwanie starego arkusza
    Application.DisplayAlerts = False
    For Each ws In ThisWorkbook.Worksheets
        If ws.Name = QryName Then ws.Delete
    Next ws
    Application.DisplayAlerts = True

    ' Dodawanie do nowego arkusza
    Set OutputSheet = Sheets.Add(After:=ActiveSheet)
    OutputSheet.Name = QryName

    With OutputSheet.ListObjects.Add(SourceType:=0, Source:= _
        "OLEDB;Provider=Microsoft.Mashup.OleDb.1;Data " & _
        "Source=$Workbook$;Location=" & Qry.Name _
        , Destination:=Range("$A$1")).QueryTable
        .CommandType = xlCmdDefault
        .CommandText = Array("SELECT * FROM [" & Qry.Name & "]")
        .RefreshOnFileOpen = False
        .BackgroundQuery = True
    End With
End Sub
```

Jeżeli wszystko pójdzie gładko i zgodnie z oczekiwaniami, po zakończeniu będziesz dysponować sprytnym mechanizmem, który pozwala na dynamiczne edytowanie składni zapytań Power Query, pozwalającym na tworzenie bardziej elastycznych raportów.

Przechodzenie w pętli przez wszystkie połączenia skoroszytu

Kolekcji `Workbook.Connections` możesz również używać do przechodzenia w pętli przez wszystkie połączenia skoroszytu i sprawdzania lub modyfikowania ich właściwości. Na przykład makro przedstawione poniżej wypełnia arkusz listą wszystkich połączeń aktywnego skoroszytu razem z ich ciągami połączenia oraz treścią zapytań SQL zapisanych we właściwościach `CommandText`:

```
Sub ListConnections()

    Dim i As Long
    Dim Cn As WorkbookConnection

    Worksheets.Add
    With ActiveSheet.Range("A1:C1")
        .Value = Array("Cn Name", "Connection String", "Command Text")
        .EntireColumn.AutoFit
    End With

    For Each Cn In ThisWorkbook.Connections
        i = i + 1
        Select Case Cn.Type
            Case Is = xlConnectionTypeODBC
                With ActiveSheet
                    .Range("A1").Offset(i, 0).Value = Cn.Name
                    .Range("A1").Offset(i, 1).Value = Cn.ODBCConnection.
                        ↳Connection
                    .Range("A1").Offset(i, 2).Value = Cn.ODBCConnection.
                        ↳CommandText
                End With
            Case Is = xlConnectionTypeOLEDB
                With ActiveSheet
                    .Range("A1").Offset(i, 0).Value = Cn.Name
                    .Range("A1").Offset(i, 1).Value = Cn.OLEDBConnection.
                        ↳Connection
                    .Range("A1").Offset(i, 2).Value = Cn.OLEDBConnection.
                        ↳CommandText
                End With
        End Select
    Next Cn
End Sub
```

Zastosowanie ADO i VBA do pobierania danych ze źródeł zewnętrznych

Kolejną techniką pozwalającą na pracę z danymi zewnętrznymi jest zastosowanie procedur VBA i obiektów ADO (ang. *ActiveX Data Objects*). Zastosowanie kombinacji ADO z procedurami języka VBA umożliwia pracę z danymi zewnętrznymi przechowywanymi w buforach w pamięci operacyjnej. Takie rozwiązanie jest bardzo wygodne zwłaszcza w sytuacji, kiedy musisz wykonywać złożone, wielopoziomowe procedury przetwarzające dane zewnętrzne, ale jednocześnie nie chcesz tworzyć połączeń danych w skoroszytcie ani zapisywać danych zewnętrznych bezpośrednio w skoroszytcie.

Pracując ze złożonymi skoroszytami Excela, które pobierają i przetwarzają dane ze źródeł zewnętrznych, z pewnością wcześniej czy później spotkasz się z procedurami VBA wykorzystującymi obiekty ADO. Z tego powodu powinieneś dobrze poznać i zrozumieć zasady pracy z takimi obiektami, tak abyś mógł swobodnie korzystać z takiego kodu.



W kilku kolejnych podrozdziałach omówimy szereg najważniejszych koncepcji i zagadnień związanych z obiektami ADO oraz pokażemy, w jaki sposób możesz tworzyć swoje własne procedury wykorzystujące ADO do pobierania danych z zewnętrznych źródeł danych. Pamiętaj jednak, że obiekty ADO to bardzo szerokie zagadnienie, którego z oczywistych powodów nie jesteśmy tutaj w stanie wyczerpująco omówić. Jeżeli planujesz wykorzystywanie obiektów ADO do pobierania i przetwarzania danych ze źródeł zewnętrznych na większą skalę w swoich projektach, to prawdopodobnie powinieneś jeszcze dodatkowo zainwestować w kolejną książkę, która będzie w bardziej szczegółowy sposób przedstawiała takie zagadnienia.

Jeżeli chcesz szybko „załapać” podstawową koncepcję obiektów ADO, powinieneś spróbować sobie wyobrazić ADO jako narzędzie, które pozwala na zrealizowanie dwóch zadań: utworzenie połączenia z zewnętrznym źródłem danych oraz wybranie zestawu danych, z którymi chcesz pracować. W kolejnym podrozdziale przedstawimy składnię podstawowych poleceń i wyrażen umożliwiających pracę z obiektami ADO.

Ciąg połączenia

Pierwszą operacją, jaką musisz wykonać, jest nawiązanie połączenia ze źródłem danych. Aby to zrobić, musisz podać językowi VBA kilka podstawowych informacji, które są przekazywane do interpretera w formie tak zwanego *ciągu połączenia* (ang. *connection string*). Poniżej przedstawiamy przykład ciągu połączenia wykorzystywanego do nawiązania połączenia z bazą danych Access:

```
"Provider=Microsoft.ACE.OLEDB.12.0;" & _  
"Data Source= C:\MyDatabase.accdb;" & _  
"User ID=Administrator;" & _  
"Password=AdminPassword"
```

Nie bądź onieśmielony pozornie złożoną składnią przedstawionego przykładu. Ciąg połączenia to w zasadzie nic innego jak tylko ciąg tekstowy zawierający szereg zmiennych (nazywanych również *argumentami*) i ich wartości, których VBA używa do identyfikacji i utworzenia połączenia ze źródłem danych. Choć niektóre ciągi połączenia mogą być naprawdę imponujące i zawierać dziesiątki argumentów i opcji, to jednak do tworzenia połączeń z bazami danych Access czy skoroszytami Excela będziemy wykorzystywać tylko niektóre z nich.

Jeżeli nigdy wcześniej nie miałeś okazji pracować z obiektami ADO, to możesz skoncentrować się tylko na kilku podstawowych argumentach ciągu połączenia, których znajomość jest absolutnie niezbędna: *Provider* (nazwa dostawcy), *Data Source* (źródło danych), *Extended Properties* (rozszerzone właściwości), *User ID* (nazwa konta użytkownika) oraz *Password* (hasło dostępu):

- **Provider** (nazwa dostawcy) — argument ten informuje VBA o tym, z jakim typem źródła danych będziesz pracować. Jeżeli jako źródła danych używasz bazy danych Access lub skoroszytów Excela, to argument ten powinien wyglądać następująco:

```
Provider=Microsoft.ACE.OLEDB.12.0
```

- **Data Source** (źródło danych) — ten argument informuje VBA o tym, gdzie można znaleźć bazę danych lub skoroszyt zawierający dane, z którymi będziesz pracować. Wartością tego argumentu jest zazwyczaj pełna ścieżka wskazująca lokalizację bazy danych lub skoroszytu, na przykład:

```
Data Source=C:\Moje dokumenty\MojaBazaDanych.accdb
```

- **Extended Properties** (rozszerzone właściwości) — tego argumentu używamy zazwyczaj, kiedy tworzymy połączenie ze skoroszytem Excela. Argument informuje VBA o tym, że źródłem danych jest obiekt inny niż baza danych. Jeżeli pracujesz ze skoroszytami Excela, to argument ten powinien wyglądać następująco:

```
Extended Properties=Excel 12.0
```

- **User ID** (nazwa konta użytkownika) — jest to argument opcjonalny, używany wyłącznie w sytuacjach, kiedy podanie nazwy konta użytkownika jest niezbędne do nawiązania połączenia ze źródłem danych, na przykład:

```
User Id=JanKowalski
```

- **Password** (hasło dostępu) — jest to argument opcjonalny, używany wyłącznie w sytuacjach, kiedy podanie hasła dostępu (łącznie z nazwą konta użytkownika) jest niezbędne do nawiązania połączenia ze źródłem danych, na przykład:

```
Password=MojeTajneHasło
```

Przjrzyjmy się teraz kilku wybranym przykładom zastosowania opisanych wyżej argumentów w różnych ciągach połączeń.

- Połączenie z bazą danych Access:

```
"Provider=Microsoft.ACE.OLEDB.12.0;" & _  
"Data Source= C:\MyDatabase.accdb"
```

- Połączenie z bazą danych Access z wykorzystaniem uwierzytelnienia (nazwa konta użytkownika i hasło dostępu):

```
"Provider=Microsoft.ACE.OLEDB.12.0;" & _  
"Data Source= C:\MyDatabase.accdb;" & _  
"User ID=Administrator;" & _  
"Password=AdminPassword"
```

- Połączenie ze skoroszytem Excela:

```
"Provider=Microsoft.ACE.OLEDB.12.0;" & _  
"Data Source=C:\MyExcelWorkbook.xlsx;" & _  
"Extended Properties=Excel 12.0"
```

Deklarowanie zestawu rekordów

Oprócz utworzenia połączenia ze źródłem danych konieczne jest również zdefiniowanie zestawu danych, z którym będziesz pracować. W terminologii obiektów ADO taki zestaw rekordów nosi nazwę **Recordset**. Obiekt **Recordset** to inaczej mówiąc kontener dla rekordów pobieranych ze źródła danych. Najpopularniejszą metodą definiowania zestawu danych jest otwarcie istniejącej tabeli lub widoku z użyciem następujących argumentów:

```
Recordset.Open Source, ConnectString, CursorType, LockType
```

Argument **Source** (źródło) wskazuje dane, które mają być pobierane. Zazwyczaj będzie to tabela, widok lub po prostu zapytanie SQL pobierające odpowiednie rekordy. Argument **ConnectString** definiuje ciąg połączenia wykorzystywany do utworzenia połączenia ze źródłem danych. Argument **CursorType** (typ kursora) określa, w jaki sposób obiekt **Recordset** będzie pozwalał poruszać się po pobieranych danych. Najczęściej spotykane wartości tego argumentu są następujące:

- **adOpenForwardOnly** — jest to ustawienie domyślne, jeżeli w definicji zestawu danych nie podasz żadnego typu kursora, to domyślnie zostanie przyjęta wartość **adOpenForwardOnly**. Taki rodzaj kursora jest najbardziej efektywny, ponieważ pozwala na poruszanie się w zestawie danych tylko w jednym kierunku: od początku do końca. Jest to idealne rozwiązanie do tworzenia raportów, gdzie dane muszą być po prostu sekwencyjnie odczytywane i nie ma

potrzeby odczytywania wybranych rekordów z różnych obszarów zestawu danych. Pamiętaj, że korzystając z tego typu kursora, nie możesz wprowadzać żadnych modyfikacji do danych w takim zestawie.

- **adOpenDynamic** — taki rodzaj kursora jest wykorzystywany w procesach, gdzie wymagane jest przechodzenie przez kolejne rekordy w pętli, poruszanie się w górę i w dół zestawu danych czy możliwość dynamicznego obserwowania zmian wprowadzanych do rekordów w zestawie danych. Zastosowanie takiego kursora wymaga zazwyczaj zaangażowania znacznych ilości pamięci operacyjnej i innych zasobów systemu, dlatego powinien być używany tylko wtedy, kiedy jest to niezbędne.
- **adOpenStatic** — taki kursor jest idealnym rozwiązaniem do szybkiego zwracania wyników i zwraca statyczny zbiór rekordów ze źródła danych. Działa bardzo podobnie jak kursory `adOpenForwardOnly`, ale pozwala na swobodne poruszanie się pośród zwracanych rekordów. Dodatkowo po ustawieniu odpowiedniej blokady `LockType` (innej niż `adLockReadOnly`) kursory `adOpenStatic` umożliwiają wprowadzanie zmian w zwracanych danych.

Argument `LockType` pozwala na określenie, czy dane umieszczane w zestawie danych `Recordset` mogą być modyfikowane. Domyślnie argument ten jest ustawiany na wartość `adLockReadOnly`, która wskazuje, że nie ma potrzeby (ani możliwości) modyfikowania zwracanych danych. Jeśli jednak zechcesz, możesz ustawić ten argument na wartość `adLockOptimistic`, dzięki czemu będziesz mógł swobodnie modyfikować zwracane dane.

Odwołania do biblioteki obiektów ADO

Poznałeś już podstawowe zagadnienia i pojęcia związane z obiektami ADO, możesz więc zacząć pisanie swojej pierwszej procedury, wykorzystującej takie obiekty. Zanim jednak rozpoczniesz, musisz utworzyć w edytorze VBE odwołanie do biblioteki obiektów ADO. Podobnie jak każda aplikacja pakietu Microsoft Office posiada swój zestaw obiektów, właściwości i metod, również ADO posiada swój zestaw takich obiektów. Ponieważ Excel domyślnie nie zna modelu obiektowego ADO, musimy wskazać Excelowi odpowiednie odwołanie do biblioteki obiektów ADO.

Aby to zrobić, otwórz nowy skoroszyt Excela, a następnie uruchom edytor VBE.

Po przejściu do okna edytora VBE wybierz z menu głównego polecenie *Tools/References* (narzędzia/odwołania). Na ekranie pojawi się okno dialogowe *References*, przedstawione na rysunku 11.13. Na liście *Available References* (dostępne odwołania) odszukaj i zaznacz najnowszą wersję biblioteki *Microsoft ActiveX Data Objects Library*, a następnie naciśnij przycisk OK.

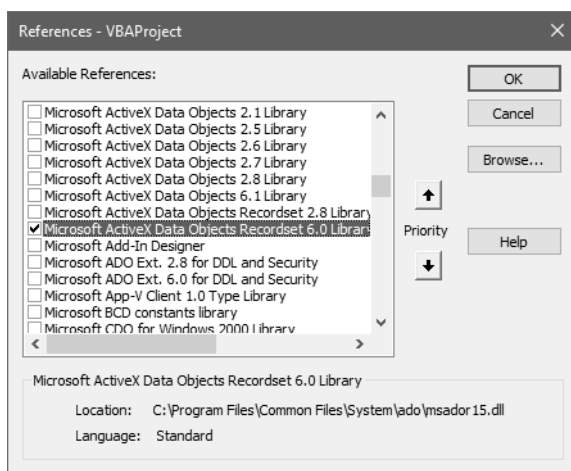


To całkowicie normalne, że na liście dostępnych bibliotek pojawia się po kilka wersji tej samej biblioteki. Zazwyczaj najlepszym rozwiązaniem jest zaznaczenie najnowszej wersji. Zwróć uwagę, że począwszy od wersji 2.8, biblioteka ta nosi nazwę *Microsoft ActiveX Data Objects Recordset Library*.

Po naciśnięciu przycisku OK możesz ponownie otworzyć okno dialogowe *References* i upewnić się, czy odwołanie do nowej biblioteki zostało poprawnie ustawione. Możesz się o tym łatwo przekonać, ponieważ zaznaczone odwołania pojawiają się na szczycie listy.



Pamiętaj, że odwołania, które ustawisz w danym skoroszycie lub bazie danych, nie są ustawiane na poziomie aplikacji. Oznacza to, że ustawianie odwołań do wybranych bibliotek musisz powtarzać dla każdego nowego skoroszytu lub bazy danych, które mają z nich korzystać.



RYSUNEK 11.13. Zaznacz najnowszą wersję biblioteki Microsoft ActiveX Data Objects Library

Łączenie wszystkiego razem w kodzie procedury

Omówiliśmy już podstawowe zagadnienia związane z obiektami ADO, nadszedł więc czas, aby połączyć wszystkie zdobyte wiadomości w kodzie VBA. Przedstawiona poniżej przykładowa procedura wykorzystuje obiekty ADO do nawiązania połączenia z bazą Access i pobrania danych z tabeli Products.

```
Sub GetAccessData()
    Dim MyConnect As String
    Dim MyRecordset As ADODB.Recordset

    MyConnect = "Provider=Microsoft.ACE.OLEDB.12.0;" & _
        "Data Source= C:\MyDir\MyDatabaseName.accdb"
    Set MyRecordset = New ADODB.Recordset
    MyRecordset.Open "Products", _
        MyConnect, adOpenStatic, adLockReadOnly

    Sheets("MySheetName").Range("A2").CopyFromRecordset MyRecordset

    With ActiveSheet.Range("A1:C1")
        .Value = Array("Product", "Description", "Segment")
        .EntireColumn.AutoFit
    End With
End Sub
```

Teraz poświęcimy chwilę na omówienie sposobu działania naszego makra.

Najpierw deklarujemy dwie zmienne: pierwsza to zmienna tekstowa, która będzie przechowywała ciąg połączenia, a druga to zmienna obiektowa typu Recordset, która będzie przechowywała zestaw danych pobieranych ze źródła. W naszym przykładzie zmienna MyConnect to zmienna tekstowa, w której zapiszemy ciąg połączenia identyfikujący źródło danych. Zmienna MyRecordset będzie przechowywała dane zwracane przez procedurę.

Następnie definiujemy ciąg połączenia dla procedury ADO. W tym scenariuszu łączymy się z bazą danych Access o nazwie *MyDatabaseName.accdb*, znajdującą się w katalogu *C:\MyDir*. Po zdefiniowaniu źródła danych możesz otworzyć zestaw danych i użyć zmiennej MyConnect do pobrania statycznego zestawu danych, który można tylko odczytywać.

Teraz możemy użyć metody `CopyFromRecordset` Excela do pobrania danych z zestawu danych i zapisania ich w skoroszybie. Metoda ta wymaga podania dwóch argumentów: docelowej lokalizacji danych oraz nazwy zestawu danych, z którego mają zostać pobrane rekordy. W naszym przykładzie będziemy kopiować dane z zestawu o nazwie `MyRecordset` do arkusza o nazwie `MySheetName` (rozpoczynając od komórki `A2`).

Co ciekawe, metoda `CopyFromRecordset` nie zwraca nagłówków kolumn ani nazw pól rekordów. Taki sposób działania wymusza wykonanie jeszcze jednej operacji, w której dodamy do arkusza nagłówki kolumn. Aby to zrobić, po prostu zdefiniujemy odpowiednie nagłówki w tablicy i zapiszemy je w aktywnym skoroszybie.

Korzystając z obiektów ADO i języka VBA, możesz utworzyć wszystkie niezbędne elementy połączenia w jednej, zgrabnej procedurze, o której możesz później spokojnie zapomnieć. Dopóki wartości zmiennych zdefiniowanych w kodzie (takie jak ścieżka do źródła danych, nazwa zestawu danych czy lokalizacja docelowa) nie będą się zmieniać, Twoja procedura wykorzystująca obiekty ADO nie będzie wymagała praktycznie żadnej aktualizacji.

Zastosowanie obiektów ADO w aktywnym skoroszybie

Wiedzę, którą nabyłeś w tym rozdziale, możesz wykorzystywać na wiele różnych sposobów. Z oczywistych powodów nie jesteśmy tutaj w stanie opisać wszystkich możliwości, ale istnieje jednak kilka najczęściej powtarzających się scenariuszy, w których VBA może znakomicie ułatwić integrację Excela z bazami danych Access.

Pobieranie danych ze skoroszytów Excela

Skoroszytów Excela możesz używać jako źródeł danych dla procedur ADO. Aby to zrobić, powinieneś po prostu utworzyć odpowiednie zapytania SQL, które będą się odwoływać do danych znajdujących się w skoroszybie Excela. Cała idea polega na tym, aby przypisać zestaw danych ze skoroszytu Excela do zapytania poprzez przekazanie nazwy arkusza, adresów zakresu komórek lub nazwy zakresu do takiego zapytania SQL.

Aby pobrać wszystkie dane z określonego arkusza, musimy przekazać do zapytania nazwę arkusza zakończoną znakiem dolara (\$) jako nazwę tabeli. Pamiętaj, aby nazwę arkusza umieścić w nawiasach kwadratowych. Przykładowe zapytanie może wyglądać tak:

```
SELECT FROM [MySheet$]
```

Jeżeli nazwa arkusza zawiera spacje lub znaki specjalne, to musimy ją ująć w znaki apostrofu, na przykład:

```
SELECT FROM ['January; Forecast vs. Budget$']
```

Aby pobierać dane z wybranego zakresu komórek określonego arkusza, najpierw musimy zdefiniować arkusz w sposób opisany powyżej, a następnie dodać do niego adresy odpowiedniego zakresu komórek, na przykład:

```
SELECT * FROM [MySheet$A1:G17]
```

Aby pobierać dane z nazwanego zakresu komórek, wystarczy po prostu użyć w zapytaniu SQL nazwy zakresu zamiast nazwy tabeli, na przykład:

```
SELECT * FROM MyNamedRange
```

W przykładzie przedstawionym poniżej wykonujemy zapytanie na arkuszu SampleData i zwracamy tylko takie rekordy, dla których nazwa regionu to North.

```
Sub GetData_From_Excel_Sheet()

    Dim MyConnect As String
    Dim MyRecordset As ADODB.Recordset
    Dim MySQL As String

    MyConnect = "Provider=Microsoft.ACE.OLEDB.12.0;" & _
        "Data Source=" & ThisWorkbook.FullName & ";" & _
        "Extended Properties=Excel 12.0"

    MySQL = " SELECT * FROM [SampleData$]" & _
        " WHERE Region ='NORTH'"

    Set MyRecordset = New ADODB.Recordset
    MyRecordset.Open MySQL, MyConnect, adOpenStatic, adLockReadOnly

    ThisWorkbook.Sheets.Add
    ActiveSheet.Range("A2").CopyFromRecordset MyRecordset

    With ActiveSheet.Range("A1:F1")
        .Value = Array("Region", "Market", "Branch_Number", _
            "Invoice_Number", "Sales_Amount", "Contracted Hours")
        .EntireColumn.AutoFit
    End With
End Sub
```

FTP

Skoroszyt z tym przykładem (*Pobieranie danych z Excela.xlsm*) znajdziesz na serwerze FTP wydawnictwa Helion (<ftp://ftp.helion.pl/przyklady/e19pww.zip>).

Dołączanie rekordów do istniejącej tabeli Excela

Bardzo często zdarza się, że nie chcesz w skoroszycie Excela nadpisywać istniejących danych nowymi rekordami. Wtedy lepszym rozwiązaniem będzie dołączanie nowych rekordów na końcu istniejącej tabeli. W typowym scenariuszu początkowa lokalizacja czy zakres komórek arkusza, do których chcesz skopiować nowy zestaw rekordów, są zakodowane na stałe w procedurze. W takich sytuacjach procedura musi dynamicznie modyfikować lokalizację docelową, aby dostosować ją do adresu pierwszej wolnej komórki arkusza. Poniżej przedstawiamy przykład procedury VBA wykorzystującej takie rozwiązanie:

```
Sub Append_Results()
    Dim MyConnect As String
    Dim MyRecordset As ADODB.Recordset
    Dim MyRange As String

    MyConnect = "Provider=Microsoft.ACE.OLEDB.12.0;" & _
        "Data Source= C:\MyDir\MyDatabase.accdb"

    Set MyRecordset = New ADODB.Recordset

    MyRecordset.Open "Products", MyConnect, adOpenStatic

    Sheets("AppendData").Select
    MyRange = "A" & _
```

```
ActiveSheet.Cells.SpecialCells(xlCellTypeLastCell).Row + 1
```

```
ActiveSheet.Range(MyRange).CopyFromRecordset MyRecordset  
End Sub
```

Ponieważ chcemy dołączyć nowe rekordy do istniejącej tabeli, musimy dynamicznie określić adres pierwszej wolnej komórki, której możemy użyć jako docelowej lokalizacji nowych rekordów. Pierwszym krokiem, jaki musimy wykonać, jest znalezienie pierwszego pustego wiersza. Dzięki zastosowaniu metody `SpecialCells` Excela nie jest to na szczęście trudnym zadaniem.

Korzystając z metody `SpecialCells`, możemy znaleźć adres ostatniej używanej komórki arkusza, a następnie odczytać numer wiersza, w którym ta komórka się znajduje. Taka operacja da nam numer ostatniego używanego wiersza arkusza. Aby uzyskać numer pierwszego pustego wiersza, wystarczy po prostu do numeru ostatniego używanego wiersza dodać 1 (wiersz następujący po ostatnim używanym wierszu musi być z definicji pusty).

Cały pomysł opiera się na połączeniu polecenia wykorzystującego metodę `SpecialCells` z literą reprezentującą kolumnę (w naszym przypadku A) do utworzenia ciągu znaków reprezentującego zakres komórek. Jeśli na przykład pierwszy pusty wiersz ma numer 10, to kod przedstawiony poniżej zwróci adres A10.

```
"A" & ActiveSheet.Cells.SpecialCells(xlCellTypeLastCell).Row + 1
```

Jeżeli teraz przypiszemy wynik działania powyższego polecenia do zmiennej tekstowej, np. `MyRange`, będziemy ją mogli następnie przekazać jako argument wywołania metody `CopyFromRecordset`.

Operacje na plikach tekstowych

W języku VBA istnieje szereg poleceń, które pozwalają na wykonywanie niskopoziomowych operacji na plikach. Wspomniane polecenia operacji wejścia-wyjścia dają znacznie większą kontrolę nad plikami niż zwykłe opcje importowania i eksportowania plików dostępne w Excelu.

Wyróżniamy trzy metody dostępu do pliku:

- **Dostęp sekwencyjny** — to metoda najbardziej popularna, pozwalająca odczytywać i zapisywać pojedyncze znaki lub całe wiersze danych.
- **Dostęp swobodny** — wykorzystywany tylko w przypadku tworzenia aplikacji bazodanowych (nie powinno się tego robić w języku VBA, ponieważ istnieją lepsze techniki).
- **Dostęp binarny** — wykorzystywany jest w celu odczytywania lub zapisywania dowolnego bajta w pliku, na przykład podczas operacji zapisywania lub wyświetlania mapy bitowej (ten sposób jest bardzo rzadko wykorzystywany w języku VBA).

Ponieważ w języku VBA rzadko wykorzystuje się losowy lub binarny dostęp do plików, w tym rozdziale skupiamy się na plikach o dostępie sekwencyjnym, w których wiersze danych są odczytywane kolejno od początku pliku. W przypadku zapisywania dane są zapisywane i dołączane na końcu pliku.



W metodzie odczytywania i zapisywania plików tekstowych opisanej w tej książce zastosowano tradycyjne pojęcie kanału danych. Innym, alternatywnym rozwiązaniem jest zastosowanie podejścia obiektowego. Obiekt `FileSystemObject` zawiera obiekt `TextStream`, który można wykorzystać do odczytywania lub zapisywania plików tekstowych. Obiekt `FileSystemObject` należy do biblioteki *Windows Scripting Host*. Jak wspominałem wcześniej, ta właściwość jest wyłączona w wielu systemach ze względu na duże ryzyko rozpowszechniania wirusów.

Otwieranie plików tekstowych

Do otwierania plików do zapisu lub odczytu służy instrukcja `Open` (nie należy jej mylić z metodą `Open` obiektu `Workbook`). Zanim będziesz mógł odczytywać lub zapisywać dane, plik musi zostać wcześniej otwarty.

Instrukcja `Open` jest dość uniwersalna i ma całkiem złożoną składnię:

```
Open ścieżka For tryb [Access rodzaj_dostępu] [blokada] _  
As [#]numer_pliku [Len=rozmiar_rekordu]
```

- *ścieżka* (wymagany) — nazwa i ścieżka (opcjonalnie) pliku, który ma być otwarty.
- *tryb* (wymagany) — może mieć jedną z poniższych wartości:
 - `Append` — tryb dostępu sekwencyjnego, który pozwala na czytanie danych lub ich dołączanie na końcu pliku;
 - `Input` — tryb dostępu sekwencyjnego, który pozwala na czytanie danych, ale nie pozwala na ich zapisywanie;
 - `Output` — tryb dostępu sekwencyjnego, który pozwala na czytanie danych lub ich zapisywanie (w tym trybie zawsze jest tworzony nowy plik, a istniejący wcześniej plik o tej samej nazwie jest usuwany);
 - `Binary` — tryb dostępu losowego, który pozwala na odczytywanie lub zapisywanie danych bajt po bajcie;
 - `Random` — tryb dostępu losowego pozwalający na czytanie lub zapis informacji w blokach, których rozmiar określa ostatni argument instrukcji `Open` — *rozmiar_rekordu*.
- *rodzaj_dostępu* (opcjonalny) — określa rodzaj operacji dozwolonych do wykonania z plikiem. Może mieć wartość `Read` (czytanie), `Write` (zapisywanie) lub `Read Write` (czytanie i zapisywanie).
- *blokada* (opcjonalny) — przydaje się w przypadku używania pliku jednocześnie przez wielu użytkowników. Dopuszczalne wartości to `Shared` (współdzielony), `Lock Read` (blokada odczytu), `Lock Write` (blokada zapisu) oraz `Lock Read Write` (blokada odczytu i zapisu).
- *numer_pliku* (wymagany) — numer pliku w zakresie od 1 do 511. Aby uzyskać następny wolny numer pliku, można skorzystać z funkcji `FreeFile` (opis funkcji `FreeFile` można znaleźć w punkcie „Przydzielanie numeru pliku” w dalszej części rozdziału).
- *rozmiar_rekordu* (opcjonalny) — rozmiar rekordu (dla plików o dostępie losowym) lub rozmiar bufora (dla plików o dostępie sekwencyjnym).

Odczytywanie plików tekstowych

Podstawowa procedura odczytywania danych z pliku tekstowego w języku VBA składa się z następujących kroków:

1. Otwarcie pliku za pomocą instrukcji `Open`.
2. Określenie pozycji w pliku za pomocą funkcji `Seek` (opcjonalnie).
3. Odczytywanie danych z pliku (za pomocą instrukcji `Input`, `Input #` lub `Line Input #`).
4. Zamknięcie pliku za pomocą instrukcji `Close`.

Zapisywanie danych do plików tekstowych

Podstawowa procedura zapisywania danych do pliku tekstowego jest następująca:

1. Otwarcie lub utworzenie pliku za pomocą instrukcji `Open`.
2. Określenie pozycji w pliku za pomocą funkcji `Seek` (opcjonalnie).
3. Zapis danych do pliku za pomocą instrukcji `Write #` lub `Print #`.
4. Zamknięcie pliku za pomocą instrukcji `Close`.

Przydzielanie numeru pliku

Większość programistów VBA po prostu przydziela odpowiedni numer pliku i podaje go jako argument instrukcji `Open`, na przykład:

```
Open "mójplik.txt" For Input As #1
```

Gdy taka instrukcja zostanie wykonana, w dalszej części kodu można się odwoływać do pliku jako do #1.

Jeżeli plik jest otwierany w czasie, kiedy inny jest już otwarty, kolejny plik można oznaczyć jako #2:

```
Open "inny.txt" For Input As #2
```

Innym sposobem uzyskania numeru pliku jest użycie funkcji `FreeFile` w celu pobrania uchwytu do pliku. Po wykonaniu tej funkcji można odwoływać się do pliku za pomocą zmiennej. Oto przykład:

```
FileHandle = FreeFile  
Open "mójplik.txt" For Input As FileHandle
```

Określanie lub ustawianie pozycji w pliku

W przypadku sekwencyjnego dostępu do plików znajomość bieżącej lokalizacji wewnątrz pliku jest rzadko potrzebna. Jeżeli jednak z jakiegoś powodu taka informacja jest potrzebna, możesz użyć funkcji `Seek`.

Import i eksport plików tekstowych w Excelu

Excel obsługuje trzy typy plików tekstowych:

- **CSV** (ang. *Comma-Separated Value*) — kolumny danych są rozdzielane przecinkami, a każdy wiersz kończy się znakiem powrotu karetki (w niektórych narodowych wersjach Excela zamiast przecinka używany jest średnik).
- **PRN** — kolumny danych są wyrównywane przez pozycje znaków, a każdy wiersz kończy się znakiem powrotu karetki. Takie pliki są często nazywane plikami z kolumnami o stałej szerokości (ang. *fixed-width files*).
- **TXT** (pliki z danymi rozdzielanymi znakami tabulacji) — kolumny danych są rozdzielane znakami tabulacji, a każdy wiersz kończy się znakiem powrotu karetki.

Jeżeli spróbujesz otworzyć plik tekstowy za pomocą polecenia *Plik/Otwórz*, na ekranie może pojawić się okno *Kreatora importu tekstu*, ułatwiającego poprawne zdefiniowanie poszczególnych kolumn. Jeżeli plik tekstowy jest rozdzielany znakami tabulacji lub spacji, Excel zazwyczaj otwiera plik bez wyświetlania kreatora importowania. Jeżeli dane nie zostaną poprawnie odczytane, zamknij plik i spróbuj zmienić rozszerzenie jego nazwy na *.txt*.

Kreator konwersji tekstu na kolumny jest niemal identyczny, ale działa poprawnie tylko w przypadku danych zapisanych w pojedynczej kolumnie. Aby go uruchomić, przejdź na kartę *Dane* i naciśnij przycisk *Tekst jako kolumny*, znajdujący się w grupie opcji *Narzędzia danych*.

Instrukcje pozwalające na odczytywanie i zapisywanie plików

W języku VBA znajdziemy kilka instrukcji pozwalających na odczytywanie i zapisywanie danych do pliku.

Do odczytywania danych z plików o dostępie sekwencyjnym służą trzy instrukcje:

- **Input** — odczytuje z pliku określoną liczbę znaków.
- **Input #** — odczytuje dane z pliku, przypisując wartości do serii zmiennych oddzielonych od siebie przecinkami.
- **Line Input #** — odczytuje cały wiersz danych, ograniczony znakami powrotu karetki i (lub) wysunięcia wiersza.

Do zapisywania danych w plikach o dostępie sekwencyjnym służą dwie instrukcje:

- **Write #** — zapisuje do pliku ciąg wartości, gdzie kolejne wartości są od siebie oddzielone przecinkami i ujęte w apostrofy. W przypadku zakończenia instrukcji średnikiem po wartościach nie jest wprowadzana sekwencja znaków CR LF. Dane zapisywane do pliku za pomocą instrukcji **Write #** zazwyczaj są odczytywane z pliku za pomocą instrukcji **Input #**.
- **Print #** — zapisuje do pliku ciąg wartości, gdzie kolejne wartości są od siebie oddzielone znakiem tabulacji. W przypadku zakończenia instrukcji średnikiem po wartościach nie jest wprowadzana sekwencja znaków CR LF. Dane zapisywane do pliku za pomocą instrukcji **Print #** zazwyczaj są odczytywane z pliku za pomocą instrukcji **Line Input #** lub **Input**.

Przykłady wykonywania operacji na plikach

W tym podrozdziale przedstawimy kilka przykładów ilustrujących różne techniki wykonywania operacji na plikach tekstowych.

Importowanie danych z pliku tekstowego

Procedura przedstawiona poniżej odczytuje dane z pliku tekstowego, a następnie umieszcza każdy wiersz danych w osobnej komórce (począwszy od aktywnej komórki):

```
Sub ImportData()
    Open "c:\helion\mój_plik.txt" For Input As #1
    r = 0
    Do Until EOF(1)
        Line Input #1, data
        ActiveCell.Offset(r, 0) = data
        r = r + 1
    Loop
    Close #1
End Sub
```

W praktyce taka procedura nie będzie jednak zbyt przydatna, ponieważ każdy wiersz danych jest wpisywany do osobnej komórki. W takiej sytuacji o wiele łatwiejszym rozwiązaniem będzie po prostu bezpośrednie otwarcie pliku tekstowego za pomocą polecenia *Plik/Otwórz*.

Eksportowanie zakresu do pliku tekstowego

Procedura przedstawiona poniżej zapisuje dane z zaznaczonego zakresu komórek arkusza do pliku tekstowego w formacie CSV. Excel potrafi oczywiście bezpośrednio eksportować dane do pliku w formacie CSV, ale w taki sposób można eksportować tylko całe arkusze, podczas gdy nasza procedura działa dla dowolnego, zaznaczonego obszaru arkusza.

```
Sub ExportRange()
    Dim Filename As String
    Dim NumRows As Long, NumCols As Integer
    Dim r As Long, c As Integer
    Dim Data
    Dim ExpRng As Range

    Set ExpRng = Selection
    NumCols = ExpRng.Columns.Count
    NumRows = ExpRng.Rows.Count
    Filename = Application.DefaultFilePath & "\textfile.csv"
    Open Filename For Output As #1
    For r = 1 To NumRows
        For c = 1 To NumCols
            Data = ExpRng.Cells(r, c).Value
            If IsNumeric(Data) Then Data = Val(Data)
            If IsEmpty(ExpRng.Cells(r, c)) Then Data = ""
            If c <> NumCols Then
                Write #1, Data;
            Else
                Write #1, Data
            End If
        Next c
    Next r
    Close #1
    MsgBox ExpRng.Count & " komórek zostało wyeksportowanych do " _
    & Filename, vbInformation
End Sub
```

W procedurze dwukrotnie wykorzystano funkcję `Write #`. Pierwsza instrukcja kończy się średnikiem, a zatem sekwencja CR LF nie będzie zapisywana. Dla ostatniej komórki w wierszu, w drugiej instrukcji `Write #`, nie użyto średnika, dzięki czemu następny zapis do pliku zostanie umieszczony w nowym wierszu.

Do zapisania zawartości komórek wykorzystana została zmienna o nazwie `Data`. Jeżeli komórka zawiera format liczbowy, zmienna jest przekształcana na liczbę. Dzięki tej czynności dane liczbowe nie zostaną zapisane ze znakami cudzysłowu. Jeżeli komórka jest pusta, wartość jej właściwości `Value` wynosi 0. Z tego powodu kod sprawdza, czy komórki nie są puste (za pomocą funkcji `IsEmpty`), i wstawia pusty ciąg znaków zamiast wartości 0.

FTP

Skorzystaj z tym przykładem (*Eksport-import-CSV.xlsm*) znajdziesz na serwerze FTP wydawnictwa Helion (<ftp://ftp.helion.pl/przyklady/e19pww.zip>).

Importowanie pliku tekstowego do zakresu

Procedura przedstawiona poniżej odczytuje dane z pliku CSV utworzonego w poprzednim przykładzie, a następnie zapisuje uzyskane wartości do arkusza, rozpoczynając od aktywnej komórki. Program odczytuje każdy znak i przetwarza wiersze danych w celu wyszukania przecinków oddzielających kolumny i usunięcia cudzysłowów przed zapisaniem danych do arkusza.

```
Sub ImportRange()
    Dim ImpRng As Range
    Dim Filename As String
    Dim r As Long, c As Integer
    Dim txt As String, Char As String * 1
    Dim Data
    Dim i As Integer

    Set ImpRng = ActiveCell
    On Error Resume Next
    Filename = Application.DefaultFilePath & "\textfile.csv"
    Open Filename For Input As #1
    If Err <> 0 Then
        MsgBox "Nie znaleziono pliku: " & Filename, vbCritical, "Błąd"
        Exit Sub
    End If
    r = 0
    c = 0
    txt = ""
    Application.ScreenUpdating = False
    Do Until EOF(1)
        Line Input #1, Data
        For i = 1 To Len(Data)
            Char = Mid(Data, i, 1)
            If Char = "," Then 'przecinek
                ActiveCell.Offset(r, c) = txt
                c = c + 1
                txt = ""
            ElseIf i = Len(Data) Then 'koniec wiersza
                If Char <> Chr(34) Then txt = txt & Char
                ActiveCell.Offset(r, c) = txt
                txt = ""
            ElseIf Char <> Chr(34) Then
                txt = txt & Char
            End If
        Next i
        c = 0
        r = r + 1
    Loop
    Close #1
    Application.ScreenUpdating = True
End Sub
```



Procedura pokazana powyżej poradzi sobie z większością danych, ale ma pewną wadę: nie potrafi poprawnie przetwarzać danych zawierających przecinki lub znaki cudzysłowu. Dodatkowo zaimportowane daty będą otoczone znakami #, na przykład #2013-05-12#.

Logowanie wykorzystania Excela

Kod zaprezentowany w tym punkcie zapisuje dane do pliku tekstowego podczas każdej operacji uruchamiania i zamykania Excela. Aby zaprezentowana procedura działała niezawodnie, musi być umieszczona w skoroszycie, który otwiera się za każdym razem, kiedy uruchamiamy Excela — do tego celu idealnie nadaje się osobisty arkusz makr (ang. *Personal Macro Workbook*).

Poniższa procedura jest umieszczona w module kodu obiektu `ThisWorkbook` i jest wykonywana podczas otwierania pliku:

```
Private Sub Workbook_Open()
    Open Application.Path & "\excelusage.txt" For Append As #1
    Print #1, "Uruchomienie programu Excel " & Now
    Close #1
End Sub
```

Procedura dodaje wiersz do pliku o nazwie *excelusage.txt*. Nowy wiersz zawiera bieżącą datę i godzinę i może mieć następującą postać:

```
Uruchomienie programu Excel 2019-11-16 21:27:43
```

Pokazana poniżej procedura wykonuje się podczas zamykania skoroszytu. Jej działanie polega na dodaniu do pliku tekstowego wiersza zawierającego frazę *Zakończenie pracy programu Excel* wraz z bieżącą datą i godziną.

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
    Open Application.Path & "\excelusage.txt" _
        For Append As #1
    Print #1, "Zakończenie pracy programu Excel " & Now
    Close #1
End Sub
```

FTP

Skoroszyt z tym przykładem (*Excel Log.xlsm*) znajdziesz na serwerze FTP wydawnictwa Helion (<ftp://ftp.helion.pl/przyklady/e19pww.zip>).



Więcej szczegółowych informacji na temat procedur obsługi zdarzeń `Workbook_Open` oraz `Workbook_BeforeClose` znajdziesz w rozdziale 6., „Obsługa zdarzeń”.

Filtrowanie zawartości pliku tekstowego

W przykładzie zaprezentowanym poniżej zademonstrujemy metodę jednoczesnego przetwarzania dwóch plików tekstowych. Procedura `FilterFile` odczytuje dane z pliku tekstowego (*infile.txt*) i kopiuje wiersze zawierające określony ciąg znaków (na przykład *"Styczeń"*) do drugiego pliku tekstowego (*output.txt*).

```
Sub FilterFile()
```

```
    Dim TextToFind As String
    Dim Filtered As Long
    Dim data As String
```

```
    Open ThisWorkbook.Path & "\infile.txt" For Input As #1
    Open Application.DefaultFilePath & "\output.txt" For Output As #2
    If Err <> 0 Then
        MsgBox "Błąd odczytu lub zapisu pliku."
    Exit Sub
```

```

End If
TextToFind = "Styczeń"
Do Until EOF(1)
    Line Input #1, data
    If InStr(1, data, TextToFind) Then
        Print #2, data End If
Loop
Close #1 'Zamknij wszystkie pliki
MsgBox Filtered & " wierszy zostało zapisanych do pliku:" & vbNewLine & _
    Application.DefaultFilePath & "output.txt"
End Sub

```

FTP

Skoroszyt z tym przykładem (*Filtrowanie pliku tekstowego.xlsm*) znajdziesz na serwerze FTP wydawnictwa Helion (<ftp://ftp.helion.pl/przyklady/e19pww.zip>).

Najczęściej wykonywane operacje na plikach

Wiele aplikacji programu Excel wykonuje różne operacje na plikach zewnętrznych. Czasami trzeba wyświetlić listę plików w katalogu, usunąć pliki, zmienić im nazwy itd. Excel pozwala oczywiście na importowanie i eksportowanie różnych typów plików tekstowych, często jednak wbudowane właściwości obsługi plików są niewystarczające. Dobrym przykładem będzie tutaj sytuacja, gdy trzeba dokonać eksportu zakresu komórek do prostego dokumentu HTML (ang. *Hypertext Markup Language*).

W tym podrozdziale dowiesz się, jak używać języka Visual Basic for Applications (VBA) do realizacji zarówno tych najczęściej wykonywanych, jak i tych mniej popularnych operacji na plikach.

W Excelu można wykonywać operacje na plikach na dwa sposoby:

- **Za pomocą tradycyjnych instrukcji i funkcji języka VBA** (ta metoda działa we wszystkich wersjach Excela).
- **Za pomocą obiektu `FileSystemObject` wykorzystującego bibliotekę *Microsoft Scripting Library*** (ta metoda działa w Excelu 2000 i późniejszych wersjach).



Poprzednie wersje programu Excel obsługiwały również obiekt `FileSearch`, ale od wersji 2007 mechanizm ten został usunięty. W nowej wersji Excela próba uruchomienia makra wykorzystującego obiekt `FileSearch` zakończy się po prostu wyświetleniem komunikatu o błędzie.

Kolejne sekcje zawierają opis wymienionych metod wraz z odpowiednimi przykładami.

Zastosowanie poleceń języka VBA do wykonywania operacji na plikach

Zestawienie poleceń VBA, które można wykorzystać do wykonywania operacji na plikach, zostało zamieszczone w tabeli 11.1. Większość poleceń nie wymaga specjalnego komentarza, a wszystkie są opisane w systemie pomocy Excela.

W dalszej części tego rozdziału znajdziesz szereg przykładów ilustrujących zastosowanie niektórych poleceń.

TABELA 11.1. Polecenia operacji na plikach w języku VBA

Nazwa polecenia	Opis działania
ChDir	Zmienia bieżący katalog
ChDrive	Zmienia bieżący napęd
Dir	Zwraca nazwę pliku lub katalog pasujący do określonego wzorca lub atrybutu pliku
FileCopy	Kopiuje plik
FileDateTime	Zwraca datę i godzinę ostatniej modyfikacji pliku
FileLen	Zwraca rozmiar pliku w bajtach
GetAttr	Zwraca wartość reprezentującą atrybut pliku
Kill	Usuwa plik
MkDir	Tworzy nowy katalog
Name	Zmienia nazwę pliku lub katalogu
RmDir	Usuwa pusty katalog
SetAttr	Zmienia atrybut pliku

Funkcja VBA sprawdzająca, czy istnieje dany plik

Poniższa funkcja zwraca wartość `True`, jeżeli określony plik istnieje, a wartość `False`, jeżeli plik nie zostanie odnaleziony. Jeżeli funkcja `Dir` zwraca pusty ciąg znaków, oznacza to, że nie można odnaleźć żądanego pliku i funkcja `FileExists` zwraca wartość `False`.

```
Function FileExists(fname) As Boolean
    FileExists = Dir(fname) <> ""
End Function
```

Argumentem funkcji `FileExists` jest pełna ścieżka dostępu do pliku wraz z jego nazwą. Funkcję można wykorzystać w arkuszu lub wywołać z poziomu procedury VBA. A oto przykład wywołania takiej funkcji:

```
MyFile = "c:\Budżet\2013-propozycja budżetu.docx"
Msgbox FileExists(MyFile)
```

Funkcja VBA sprawdzająca, czy istnieje dany katalog

Poniższa funkcja zwraca wartość `True`, jeżeli określony katalog istnieje, a wartość `False`, jeżeli katalog nie zostanie odnaleziony:

```
Function PathExists(pname) As Boolean
    ' Zwraca wartość True, jeżeli katalog istnieje
    On Error Resume Next
    PathExists = (GetAttr(pname) And vbDirectory) = vbDirectory
End Function
```

Argument `pname` ma postać łańcucha tekstu, który zawiera ścieżkę katalogu (bez nazwy pliku). Znak ukośnika zamykający ścieżkę jest opcjonalny. Poniżej przedstawiamy przykład wywołania takiej funkcji.

```
MyFolder = "c:\użytkownicy\jan\pulpit\pobieranie\"
MsgBox PathExists(MyFolder)
```

FTP

Skoroszyt zawierający funkcje FileExists oraz PathExists (*Funkcje plikowe.xlsm*) znajdziesz na serwerze FTP wydawnictwa Helion (<ftp://ftp.helion.pl/przyklady/e19pww.zip>).

Procedura VBA wyświetlająca listę plików w katalogu

Poniższa procedura wyświetla w aktywnym arkuszu listę plików z określonego katalogu wraz z rozmiarem i datą modyfikacji pliku:

```
Sub ListFiles()
    Dim Directory As String
    Dim r As Long
    Dim f As String
    Dim FileSize As Double
    Directory = "F:\Excel\Budżet"
    r = 1
    ' Wstaw nagłówki
    Cells.ClearContents
    Cells(r, 1) = "Nazwa pliku"
    Cells(r, 2) = "Rozmiar"
    Cells(r, 3) = "Data/godzina"
    Range("A1:C1").Font.Bold = True
    ' Pobierz pierwszy plik
    f = Dir(Directory, vbReadOnly + vbHidden + vbSystem)
    Do While f <> ""
        r = r + 1
        Cells(r, 1) = f
        ' Poprawka na pliki o wielkości ponad 2 GB
        FileSize = FileLen(Directory & f)
        If FileSize < 0 Then FileSize = FileSize + 4294967296#
        Cells(r, 2) = FileSize
        Cells(r, 3) = FileDateTime(Directory & f)
        ' Pobierz następny plik
        f = Dir()
    Loop
End Sub
```



Funkcja FileLen języka VBA wykorzystuje dane typu Long, dlatego w przypadku plików większych niż 2 GB będzie zwracała niepoprawny rozmiar pliku (liczbę ujemną). Kod procedury sprawdza, czy funkcja FileLen zwróciła wartość ujemną, a jeżeli tak, dokonuje odpowiednich poprawek.

Zwróć uwagę, że procedura dwukrotnie wykorzystuje funkcję Dir. Za pierwszym razem (wywołanie z argumentem) funkcja pobiera pierwszą znalezioną nazwę pliku. Kolejne wywołania w pętli (bez argumentu) powodują pobranie nazw kolejnych plików. Jeżeli nie ma więcej plików, funkcja zwraca pusty ciąg znaków.

FTP

Skoroszyt z bardziej zaawansowaną wersją tej procedury, umożliwiającą wybranie katalogu za pomocą okna dialogowego (*Lista plików.xlsm*), znajdziesz na serwerze FTP wydawnictwa Helion (<ftp://ftp.helion.pl/przyklady/e19pww.zip>).

Pierwszym argumentem funkcji `Dir` może być nazwa pliku podana w postaci wzorca (przy użyciu symboli wieloznacznych). Na przykład: aby uzyskać listę plików programu Excel, możesz użyć polecenia przedstawionego poniżej.

```
f = Dir(Directory & "*.xl??", vbReadOnly + vbHidden + vbSystem)
```

Wykonanie tego polecenia spowoduje pobranie z podanego katalogu nazwy pierwszego pliku zgodnego z wzorcem `*.xl??`. Takie użycie symboli wieloznacznych powoduje, że zwracane są nazwy plików posiadających czteroznakowe rozszerzenie zaczynające się od liter XL. Mogą to być na przykład pliki o rozszerzeniach `.xlsx`, `.xltx` czy `.xlam`. Drugi argument funkcji `Dir` umożliwia wprowadzenie atrybutów plików (definiowanych w postaci wbudowanych stałych). W tym przykładzie funkcja `Dir` pobiera nazwy plików, które mają ustawione następujące atrybuty: bez atrybutów, tylko do odczytu, plik ukryty oraz plik systemowy.

Aby procedura wyświetlała również pliki Excela zapisane w starszych formatach (takich jak na przykład `.xls` czy `.xla`), powinieneś użyć następującego wzorca:

```
*.xl*
```

W tabeli 11.2 zamieszczono zestawienie stałych, które mogą być argumentami funkcji `Dir`.

TABELA 11.2. Zestawienie stałych, które mogą być argumentami funkcji `Dir`

Nazwa stałej	Wartość	Opis
<code>vbNormal</code>	0	Plik bez atrybutów, jest to domyślne ustawienie atrybutów dla tej funkcji
<code>vbReadOnly</code>	1	Pliki tylko do odczytu
<code>vbHidden</code>	2	Pliki ukryte
<code>vbSystem</code>	4	Pliki systemowe
<code>vbVolume</code>	8	Etykieta woluminu. Jeżeli w wywołaniu funkcji został użyty jakikolwiek inny atrybut, ten atrybut będzie ignorowany
<code>vbDirectory</code>	16	Katalogi. Co ciekawe, ten atrybut... po prostu nie działa. Wywołanie funkcji <code>Dir</code> z atrybutem <code>vbDirectory</code> nie zwraca nazw podkatalogów



Jeżeli używasz funkcji `Dir` do przechodzenia w pętli przez kolejne pliki i wywoływania innych procedur przetwarzających pliki, upewnij się, że nie zawierają one poleceń `Dir` — za każdym razem może być aktywna tylko jedna instancja polecenia `Dir`.

Rekurencyjna procedura VBA wyświetlająca listę plików w katalogu

Procedura przedstawiona w tym podrozdziale tworzy listę plików znajdujących się w danym katalogu oraz wszystkich jego podkatalogach. Sama procedura jest nieco nietypowa, ponieważ zawiera wywołania do samej siebie — takie rozwiązanie nazywamy *rekurencją*.

```
Public Sub RecursiveDir(ByVal CurrDir As String, Optional ByVal Level As Long)
```

```
    Dim Dirs() As String
    Dim NumDirs As Long
    Dim FileName As String
    Dim PathAndName As String
    Dim i As Long
    Dim FileSize As Double
```

```

' Upewnij się, że ścieżka kończy się znakiem lewego ukośnika
If Right(CurrDir, 1) <> "\" Then CurrDir = CurrDir & "\"

' Wstaw nagłówki kolumn do aktywnego arkusza
Cells(1, 1) = "Ścieżka"
Cells(1, 2) = "Nazwa pliku"
Cells(1, 3) = "Rozmiar"
Cells(1, 4) = "Data/godzina"
Range("A1:D1").Font.Bold = True

' Pobierz pliki
FileName = Dir(CurrDir & ".*", vbDirectory)
Do While Len(FileName) <> 0
    If Left(FileName, 1) <> "." Then 'Bieżący katalog
        PathAndName = CurrDir & FileName
        If (GetAttr(PathAndName) And vbDirectory) = vbDirectory Then
            'Zapamiętaj odnalezione katalogi
            ReDim Preserve Dirs(0 To NumDirs) As String
            Dirs(NumDirs) = PathAndName
            NumDirs = NumDirs + 1
        Else
            'Zapisz ścieżkę i nazwę pliku na arkuszu
            Cells(WorksheetFunction.CountA(Range("A:A")) + 1, 1) = _
                CurrDir
            Cells(WorksheetFunction.CountA(Range("B:B")) + 1, 2) = _
                FileName
            'Poprawka na pliki o rozmiarze powyżej 2 GB
            Filesize = FileLen(PathAndName)
            If Filesize < 0 Then Filesize = Filesize + 4294967296#
            Cells(WorksheetFunction.CountA(Range("C:C")) + 1, 3) = Filesize
            Cells(WorksheetFunction.CountA(Range("D:D")) + 1, 4) = _
                FileDateTime(PathAndName)
        End If
    End If
    FileName = Dir()
Loop
' Rekurencyjne przetwarzanie odnalezionych katalogów
For i = 0 To NumDirs - 1
    RecursiveDir Dirs(i), Level + 2
Next i
End Sub

```

Procedura pobiera tylko jeden argument, CurrDir, reprezentujący przetwarzany katalog. Informacja o poszczególnych odnalezionych plikach jest wyświetlana na aktywnym arkuszu. Nazwy podkatalogów odnalezionych podczas rekurencyjnego przetwarzania plików są zapamiętywane w tablicy o nazwie Dirs. Kiedy w bieżącym katalogu nie ma już więcej plików do przetwarzania, procedura wywołuje samą siebie, pobierając jako argument wywołania nazwę kolejnego podkatalogu z tablicy Dirs. Procedura kończy działanie po zakończeniu przetwarzania wszystkich podkatalogów zapisanych w tablicy Dirs.

Ponieważ procedura RecursiveDir wymaga podania odpowiedniego argumentu, musi być wywoływana z poziomu innej procedury, na przykład za pomocą następującego polecenia:

```
Call RecursiveDir("c:\nazwa_katalogu\")
```

FTP

Skoroszyt z tym przykładem (*Lista plików - rekurencja.xlsm*) znajdziesz na serwerze FTP wydawnictwa Helion (<ftp://ftp.helion.pl/przyklady/e19pww.zip>).

Zastosowanie obiektu FileSystemObject

Obiekt `FileSystemObject` należy do biblioteki *Windows Scripting Host* i zapewnia dostęp do systemu plików komputera. Obiekt ten często jest wykorzystywany na stronach WWW zawierających skrypty (np. VBScript lub JavaScript). Można z niego korzystać w Excelu 2000 i późniejszych wersjach.



Mechanizm *Windows Scripting Host* jest czasami wykorzystywany do rozpowszechniania wirusów komputerowych, dlatego funkcja ta w wielu systemach jest wyłączona. Projektując aplikacje, które będą wykorzystywane na wielu komputerach, należy o tym pamiętać i zachować szczególną ostrożność.

Nazwa `FileSystemObject` może być nieco myląca, ponieważ obiekt ten w rzeczywistości składa się z szeregu innych obiektów, z których każdy posiada swoje osobne, ściśle określone przeznaczenie:

- **Drive** — reprezentuje napęd dyskowy lub całą kolekcję napędów dyskowych.
- **File** — reprezentuje plik lub kolekcję plików.
- **Folder** — reprezentuje folder lub kolekcję folderów.
- **TextStream** — reprezentuje strumień tekstu odczytywany, zapisywany lub dołączany do pliku tekstowego.

Aby skorzystać z obiektu `FileSystemObject`, powinieneś najpierw utworzyć instancję tego obiektu. Możesz tego dokonać na dwa sposoby: za pomocą tzw. *metody wczesnego wiązania* (ang. *early binding*) lub *metody późnego wiązania* (ang. *late binding*).

Metoda późnego wiązania wykorzystuje sekwencję dwóch poleceń, na przykład:

```
Dim FileSys As Object
Set FileSys = CreateObject("Scripting.FileSystemObject")
```

Zwróć uwagę, że zmienna obiektowa `FileSys` została zadeklarowana jako ogólny typ `Object`, a nie jako konkretny typ obiektowy — rodzaj obiektu zostanie ustalony podczas działania programu.

Metoda wczesnego wiązania wymaga utworzenia odwołania do modelu obiektowego *Windows Scripting Host*. Aby to zrobić, powinieneś w edytorze VBE wybrać z menu głównego polecenie *Tools/References* (narzędzia/odwołania), a następnie w oknie dialogowym *References* zaznaczyć odpowiednią opcję. Po utworzeniu odwołania możesz utworzyć obiekt za pomocą następującej sekwencji poleceń:

```
Dim FileSys As Object
Set FileSys = CreateObject("Scripting.FileSystemObject")
```

Zastosowanie metody wczesnego wiązania pozwala na skorzystanie z mechanizmu *Auto List Members* (automatyczne wyświetlanie składowych obiektu) edytora VBE, który znakomicie ułatwia wpisywanie oraz identyfikację odpowiednich właściwości i metod obiektów. Co więcej, dzięki temu możesz również skorzystać z przeglądarki obiektów i sprawdzić informacje na temat danego obiektu. Aby to zrobić, wystarczy po wejściu do VBE nacisnąć klawisz *F2*.

W kolejnych przykładach poniżej przedstawimy zastosowanie obiektu `FileSystemObject` do wielu różnych zadań.

Zastosowanie obiektu FileSystemObject do sprawdzenia, czy dany plik istnieje

Poniższa funkcja pobiera jeden argument (ścieżkę wraz z nazwą pliku) i jeżeli plik istnieje, zwraca wartość `True`:

```
Function FileExists3(fname) As Boolean
    Dim FileSys As Object 'FileSystemObject
    Set FileSys = CreateObject("Scripting.FileSystemObject")
    FileExists3 = FileSys.FileExists(fname)
End Function
```

Funkcja tworzy nowy obiekt `FileSystemObject` o nazwie `FileSys`, a następnie sprawdza właściwość `FileExists` tego obiektu.

Zastosowanie obiektu `FileSystemObject` do sprawdzenia, czy istnieje dany katalog

Poniższa funkcja pobiera jeden argument (katalog) i zwraca wartość `True`, jeżeli ten katalog istnieje:

```
Function PathExists2(pname) As Boolean
    Dim FileSys As Object 'FileSystemObject
    Set FileSys = CreateObject("Scripting.FileSystemObject")
    PathExists2 = FileSys.FolderExists(path)
End Function
```

Wykorzystanie obiektu `FileSystemObject` do wyświetlenia informacji o wszystkich dostępnych napędach dysków

Procedura przedstawiona poniżej używa obiektu `FileSystemObject` do pobrania i wyświetlenia różnych informacji na temat dostępnych napędów dyskowych. Procedura przetwarza w pętli kolekcję `Drives` i zapisuje wartości różnych właściwości do arkusza.

FTP

Skoroszyt z tym przykładem (*Pokaż informację o napędach.xlsm*) znajdziesz na serwerze FTP wydawnictwa Helion (<ftp://ftp.helion.pl/przyklady/e19pww.zip>).

```
Sub ShowDriveInfo()
    Dim FileSys As FileSystemObject
    Dim Drv As Drive
    Dim Row As Long
    Set FileSys = CreateObject("Scripting.FileSystemObject")
    Cells.ClearContents
    Row = 1
    ' Nagłówki kolumn
    Range("A1:F1") = Array("Napęd", "Gotowy", "Typ", "Nazwa wolumenu", _
        "Rozmiar", "Dostępne")
    On Error Resume Next
    ' Pętla przetwarzająca kolejne napędy
    For Each Drv In FileSys.Drives
        Row = Row + 1
        Cells(Row, 1) = Drv.DriveLetter
        Cells(Row, 2) = Drv.IsReady
        Select Case Drv.DriveType
            Case 0: Cells(Row, 3) = "Nieznany"
            Case 1: Cells(Row, 3) = "Dysk wymienny"
            Case 2: Cells(Row, 3) = "Dysk twardy"
            Case 3: Cells(Row, 3) = "Dysk sieciowy"
            Case 4: Cells(Row, 3) = "Napęd CD-ROM"
            Case 5: Cells(Row, 3) = "RAM Disk"
        End Select
        Cells(Row, 4) = Drv.VolumeName
        Cells(Row, 5) = Drv.TotalSize
        Cells(Row, 6) = Drv.AvailableSpace
    Next Drv
    ' Utwórz tabelę
    ActiveSheet.ListObjects.Add xlSrcRange, _
        Range("A1").CurrentRegion, , xlYes
End Sub
```



W rozdziale 7., „Przykłady i techniki programowania w języku VBA”, znajdziesz opis innej metody pobierania informacji o napędach dyskowych, wykorzystującej funkcję API systemu Windows.

Pakowanie i rozpakowywanie plików

Prawdopodobnie najczęściej spotykanym formatem spakowanych plików jest popularny ZIP. Nawet dokumenty programu Excel 2007 (i wersji późniejszych) są zapisywane w tym formacie (choć nie używają rozszerzenia *.zip*). Archiwa w formacie ZIP mogą zawierać dowolną liczbę plików, a nawet całe struktury katalogów. Zawartość plików ma bezpośredni wpływ na stopień kompresji. Na przykład pliki graficzne w formacie JPG są już skompresowane, więc zapisanie ich dodatkowo w formacie ZIP w niewielkim stopniu wpłynie na ich rozmiar.

FTP

Skoroszyty z przykładami (*Pakowanie plików ZIP.xlsm* oraz *Rozpakowywanie plików ZIP.xlsm*) znajdziesz na serwerze FTP wydawnictwa Helion (<ftp://ftp.helion.pl/przyklady/e19pww.zip>).

Pakowanie plików do formatu ZIP

Kod zamieszczony poniżej ilustruje sposób tworzenia spakowanego archiwum w formacie ZIP zawierającego grupę plików wybranych przez użytkownika. Procedura `ZipFiles` wyświetla na ekranie okno dialogowe, za pomocą którego użytkownik może wybrać pliki przeznaczone do spakowania. Następnie w domyślnym katalogu programu Excel tworzone jest archiwum w formacie ZIP o nazwie *compressed.zip* zawierające spakowane pliki.

```
Sub ZipFiles()
    Dim ShellApp As Object
    Dim FileNameZip As Variant
    Dim FileNames As Variant
    Dim i As Long, FileCount As Long

    ' Pobierz nazwy plików
    FileNames = Application.GetOpenFilename _
        (FileFilter:="All Files (*.*)", "*.*", _
        FilterIndex:=1, _
        Title:="Zaznacz pliki przeznaczone do spakowania", _
        MultiSelect:=True)

    ' Zakończ, jeżeli operacja została anulowana
    If Not IsArray(FileNames) Then Exit Sub
    FileCount = UBound(FileNames)
    FileNameZip = Application.DefaultFilePath & "\compressed.zip"

    ' Utwórz pusty plik ZIP z nagłówkiem
    Open FileNameZip For Output As #1
    Print #1, Chr$(80) & Chr$(75) & Chr$(5) & Chr$(6) & String(18, 0)
    Close #1
    Set ShellApp = CreateObject("Shell.Application")

    ' Kopiuj pliki do skompresowanego archiwum
    For i = LBound(FileNames) To UBound(FileNames)
        ShellApp.Namespace(FileNameZip).CopyHere FileNames(i)
```

```

' Czekaj, dopóki pakowanie nie zostanie zakończone
On Error Resume Next
Do Until ShellApp.Namespace(FileNameZip).items.Count = i
    DoEvents
    Application.Wait (Now + TimeValue("0:00:01"))
Loop
Application.StatusBar = "Plik " & i & " z " & Ubound(Filenames)
Next i
If MsgBox(FileCount & " plików zostało spakowanych do pliku ZIP: " & _
vbNewLine & FileNameZip & vbNewLine & vbNewLine & _
"Wyświetlić plik ZIP?", vbQuestion + vbYesNo) = vbYes Then _
Shell "Explorer.exe /e," & FileNameZip, vbNormalFocus
End Sub

```

Procedura `ZipFiles` tworzy plik o nazwie *compressed.zip* i zapisuje w nim ciąg znaków tworzący standardowy nagłówek pliku archiwum w formacie ZIP. Następnie tworzony jest obiekt `Shell.Application` i procedura wykorzystuje jego metodę `CopyHere` do skopiowania wybranych przez użytkownika plików do archiwum ZIP. W następnej sekcji kodu znajdziemy pętlę `Do Until`, która co sekunda sprawdza liczbę plików w archiwum ZIP. Taka operacja jest konieczna, ponieważ kopiowanie plików do archiwum może zająć sporo czasu, a jeżeli procedura zakończyłaby działanie przed zakończeniem kopiowania plików, tworzone archiwum ZIP mogłoby być niekompletne (i prawdopodobnie uszkodzone).

Kiedy liczba plików w archiwum ZIP zgadza się z liczbą plików zaznaczonych przez użytkownika do spakowania, pętla kończy działanie i na ekranie zostaje wyświetlony odpowiedni komunikat. Naciśnięcie przycisku *Tak* powoduje uruchomienie programu Windows Explorer, w którym zostanie wyświetlona zawartość archiwum.



Procedura `ZipFiles` przedstawiona na przykładzie na poprzedniej stronie została maksymalnie uproszczona, aby ułatwić Czytelnikowi zrozumienie zasady jej działania. Kod procedury nie zawiera żadnych elementów sprawdzania błędów i nie jest zbyt uniwersalny. Na przykład nie ma tutaj opcji pozwalającej na wybranie nazwy i lokalizacji tworzonego archiwum ZIP, a domyślny plik *compressed.zip* jest za każdym razem po prostu nadpisywany bez żadnego ostrzeżenia. Z pewnością nie zastąpi to wbudowanych narzędzi do kompresowania, dostępnych w systemie Windows, ale jest ciekawą demonstracją tego, co możesz zrobić za pomocą języka VBA.

Rozpakowywanie plików ZIP

Procedura, którą przedstawiamy w tym podrozdziale, spełnia dokładnie przeciwną funkcję do procedury omawianej w poprzednim przykładzie. Nasz program prosi użytkownika o wskazanie pliku archiwum ZIP i następnie wypakowuje pliki z archiwum i umieszcza je w katalogu *Rozpakowane* zlokalizowanym w domyślnym katalogu programu Excel.

```

Sub UnzipAFile()
    Dim ShellApp As Object
    Dim TargetFile
    Dim ZipFolder

    ' Plik docelowy i katalog roboczy
    TargetFile = Application.GetOpenFilename _
        (FileFilter:="Zip Files (*.zip), *.zip")
    If TargetFile = False Then Exit Sub

    ZipFolder = Application.DefaultFilePath & "\Rozpakowane\"

```

```
' Utwórz katalog roboczy
On Error Resume Next
Rmdir ZipFolder
Mkdir ZipFolder
On Error GoTo 0

' Wypakuj skompresowane pliki i umieść je w utworzonym katalogu roboczym
Set ShellApp = CreateObject("Shell.Application")
ShellApp.Namespace(ZipFolder).CopyHere _
    ShellApp.Namespace(TargetFile).items _

If MsgBox("Rozpakowane pliki zostały umieszczone w folderze:" & _
    vbNewLine & ZipFolder & vbNewLine & vbNewLine & _
    "Wyświetlić zawartość tego foldera?", vbQuestion + vbYesNo) = vbYes Then _
    Shell "Explorer.exe /e," & ZipFolder, vbNormalFocus
End Sub
```

Procedura `UnzipAFile` do pobrania nazwy archiwum *ZIP* wykorzystuje metodę `GetOpenFilename`, a następnie tworzy nowy folder i używa obiektu `Shell.Application` do skopiowania zawartości archiwum *ZIP* do utworzonego wcześniej foldera. Po zakończeniu wypakowywania plików procedura pyta użytkownika, czy wyświetlić zawartość foldera docelowego.

Skorowidz

A

- Abs, 666
- Accelerator, 437
- Activate, 155, 201, 207, 308, 329, 447, 464
- ActiveCell, 72, 229
 - Cells, 77
 - ClearContents, 73
 - Offset, 79
 - Range, 76
- ActiveChart, 73, 309, 311
 - Name, 309
- ActivePrinter, 282
- ActiveSheet, 73
 - Cells, 77
- ActiveWindow, 73, 468
 - DisplayGridlines, 102
 - DisplayHeadings, 255
- ActiveWorkbook, 73
 - FullName, 73
 - Path, 363
- ActiveX, 37
- AddChart, 306
- AddIn, 554
 - Comments, 555
 - FullName, 555
 - Installed, 556
 - Name, 554
 - Path, 555
 - Title, 555
 - zdarzenia, 558
- AddInInstall, 201, 558, 560
- AddIns, 538, 545, 548, 553
 - Add, 553
- AddIns2, 553
- AddInUninstall, 201, 558
- AddItem, 470
- Additional Controls, 455
- Address, 74
- AfterCalculate, 216
- AfterPrint, 205
- AfterSave, 201
- AfterUpdate, 449
- aktualizacja
 - aplikacji, 45
 - nagłówka, 204
 - stopki, 204
- aktualna rozdzielczość karty graficznej, 282
- aktywacja wykresu, 308
- aktywna komórka, 72
- aktywne okno, 73
- aktywny
 - arkusz, 73
 - arkusz wykresu, 73
 - skoroszyt, 73
- algorytm sortowania, 259
- AllBold, 266
- Ambiguous name detected, 133
- And, 101, 102
- animacja etykiet, 490
- aplikacje
 - arkusza kalkulacyjnego, 31
 - ustawienia międzynarodowe, 655
 - Application.International, 658
 - data i czas, 660
 - identyfikacja ustawień systemu, 658
 - język aplikacji, 656
 - kody języków, 655
 - obsługa języka w kodzie VBA, 657
 - problemy, 656
 - właściwości lokalne, 658
- AppActivate, 365, 663
- Application, 106, 134, 170
 - ActiveCell, 72
 - ActiveChart, 73
 - ActivePrinter, 282
 - ActiveSheet, 73
 - ActiveWindow, 73

Application

- ActiveWorkbook, 73
- Calculation, 559
- CommandBars, 421, 598
- Dialogs, 419
- DisplayCommentIndicator, 618
- EnableCancelKey, 155, 465
- EnableEvents, 197
- GetOpenFilename, 415
- GetSaveAsFilename, 418
- Goto, 419
- Help, 628
- International, 655, 658
- MacroOptions, 187
- PathSeparator, 262, 653
- Run, 131, 550
- ScreenUpdating, 154
- Selection, 73
- SendKeys, 592
- StatusBar, 497
- ThisWorkbook, 73
- Transpose, 471
- Volatile, 170, 265
- WorksheetFunction, 106, 666

Areas, 236**AreaType, 236****argumenty, 125, 138**

- funkcji, 168, 188
- literały, 139
- nieokreślona liczba, 180
- opcjonalne, 175
- procedur, 126
- przekazywanie, 138
- tablice, 174

arkusze

- synchronizacja, 254
- system pomocy, 620
- ukrywanie komórek, 252

arkusze wykresu, 303

- tworzenie wykresu, 307

Array, 176, 666**ArrayFillRange, 245****As, 96, 140, 162****Asc, 666****Atn, 666****Auto**

- Data Tips, 67
- Indent, 67
- List Members, 67, 98, 106, 637
- Quick Info, 67
- Syntax Check, 67, 89

automatyczne menu podręczne, 611**automatyzacja**

- CreateObject, 347
- GetObject, 347
- odwołanie do obiektu, 347

B**BatchProcess, 260****Beep, 663****BeforeClose, 201, 205****BeforeDoubleClick, 207, 214, 329****BeforeDragOver, 449****BeforeDropOrPaste, 449****BeforePrint, 200, 201, 204****BeforeRightClick, 207, 214****BeforeSave, 195, 201, 203****BeforeUpdate, 449****BeginGroup, 601****bezpieczeństwo, 35**

- hasła, 43

Biblioteka funkcji, 161, 188**biblioteki**

- DLL, 190, 279

- obiektów, 134

bieżąca data, 122**blokowanie**

- obiektów arkusza, 42
- wybranych komórek, 41

błędy, 40

- #ARG!, 185

- #DZIEL/0!, 183

- #N/D!, 179

- #NAZWA?, 160

- składni, 67, 141

- wykonania, 141

Boolean, 92**BoundColumn, 480****Break on All Errors, 141****Break on Unhandled Errors, 141****BUBBLESIZE_FROM_SERIES, 322****Button_Click, 193****ByRef, 140****Byte, 92****ByVal, 139****C****Calculate, 195, 207, 329****Calculation, 559****Call, 131, 134, 139, 663****callback procedures, 570****CallByName, 666**

- Caption, 466, 498, 599, 601
- Case, 116
- CBool, 666
- CByte, 666
- CCur, 666
- CDate, 666
- CDbl, 666
- CDec, 92, 666
- Cell, 598–601, 604, 606
- Cells, 76, 77
- Change, 207–209, 447–451, 510
- Chart, 304, 305, 309, 332, 530
 - Export, 317
- ChartObject, 73, 304, 305, 314
- ChartObjects, 311
 - Delete, 311
- Charts, 307, 311
 - Add, 307
- ChartTitle, 305
- ChDir, 663
- ChDrive, 663
- CheckBox, 38, 427, 576
- CHM, 615
- Choose, 666
- Chr, 666
- Chronienie arkusza, 41
- Chroni skoroszyt, 42
- CInt, 666
- Class Module, 331, 526
- Click, 508
- CLng, 666
- Close, 663
- CloseAllWorkbooks, 251
- Code, 63, 64
- Code Colors, 68
- Collection, 473
 - NoDups, 473
- ColorNegative, 237
- ColorNegative2, 238
- ColorNegative3, 239
- ColumnCount, 469
- ColumnHeads, 469
- ComboBox, 428
- Comma Separated Values, 216
- CommandBar, 420, 421, 597
 - BeginGroup, 601
 - BuiltIn, 601
 - Caption, 601
 - Enabled, 601
 - FaceID, 601
 - ID, 601
 - odwołania do formantów, 599
 - OnAction, 601
 - Picture, 601
 - Reset, 603
 - ToolTipText, 601
 - Type, 597, 601
 - Visible, 601
 - właściwości formantów, 601
- CommandBars, 594, 598
 - ExecuteMso, 420, 589, 590
 - FindControl, 600
 - GetEnabledMso, 589, 591
 - GetImageMso, 590, 591
 - GetLabelMso, 590
 - GetPressedMso, 590
 - GetScreenTipMso, 590
 - GetSupertipMso, 590
 - Name, 598, 599
- CommandButton, 38, 137, 428, 459, 465, 466, 533
- Comment Block, 90
- Comments, 555
- Component Object Model, 540
- Const, 97, 663
- Controls, 453
- ControlSource, 469
- ControlTipText, 519, 622
- Copy, 226
- CopyMultipleSelection, 249
- CopyRange, 226
- CopyTable, 228
- Cos, 666
- Count, 235
- COUNTA, 240, 446
- CountBetween, 269
- CreateChartSheet, 307
- CreateObject, 347, 666
- CreatePivotTable, 292, 296
- CSng, 666
- CStr, 666
- CSV, Comma Separated Values, 216
- Ctrl, 130
- Ctrl+Break, 465
- CurDir, 666
- Currency, 78, 92
- CurrentRegion, 228
 - Select, 230
- CustomUI, 588
- Cut, 227
- CVar, 666
- CVDate, 666
- CVErr, 179, 667
- czas, 656
 - aplikacje dla wersji narodowych, 660
 - wyświetlanie, 255

czcionki, 258
CZY.BŁĄD, 242
CZY.LICZBA, 174
CZY.LOGICZNA, 242
CZY.TEKST, 242

D

DATA, 99
data, 99, 183, 666
 aplikacje dla wersji narodowych, 660
 zapisania pliku, 267
 wyświetlanie, 255
Date, 92, 99, 122, 256, 663, 667
DateAdd, 667
DateAndTime, 255, 256
DateDiff, 667
DatePart, 667
DateSerial, 99, 122, 660, 667
DateValue, 667
Day, 667
DDB, 667
Deactivate, 201, 204, 207, 329, 448
deaktywacja wykresu, 310
Debug, 185
 /Compile, 544
 Print, 137, 150, 168, 186
debugging, 67
Decimal, 92
Declare, 191, 663
Default to Full Module View, 68
DefaultPrinterInfo, 281
DefBool, 663
DefByte, 663
DefCur, 663
DefDate, 663
DefDbl, 663
DefDec, 663
definiowanie
 kategorii funkcji, 188
 typów danych, 91
DefInt, 663
DefLng, 663
DefObj, 664
DefSng, 664
DefStr, 664
DefVar, 664
deklaracja, 62
 funkcji, 164
 funkcji interfejsu API, 191
 procedury Sub, 126
 stałych, 97
 tablic, 102
 dynamicznych, 103
 wielowymiarowych, 103
 zmiennych, 87, 93, 94
Delete, 311, 312
DeleteEmptyRows, 240
DeleteSetting, 664
Dialogs, 419
Dim, 95, 96, 104, 105, 139, 664
Dir, 262, 667
DisplayCommentIndicator, 618
DisplayDataForm, 423
DisplayGridlines, 102
DisplayVideoInfo, 282
DLL, 190, 279
Do Until, 123, 124, 492
Do While, 122
Do-Loop, 664
dodatki, 166, 537, 540, 569
 AddIns, 538
 dodawanie elementów, 553
 usuwanie elementów, 554
 arkusze, 549
 dostęp do procedur VBA, 550
 dystrybucja, 546
 instalacja, 545
 lista kontrolna tworzenia, 547
 menedżer, 540, 545
 modyfikacja, 546
 nazwy, 555
 odwołania do plików, 561
 opis, 544
 optymalizacja wydajności, 558
 otwieranie, 557
 podglądanie zabezpieczonego dodatku, 551
 problemy, 559
 przechowywanie funkcji niestandardowych, 190
 przetwarzanie, 553
 ścieżka dostępu do pliku dodatku, 555
 testowanie, 546
 tworzenie, 541, 542, 544
 wykresy, 549
 zastosowanie, 538
 zdarzenia, 558
dodawanie
 elementu do menu podręcznego Cell, 606
 formantów do formularza UserForm, 426
 modułu, 62
 odwołania do pliku biblioteki obiektów, 347
 podmenu do menu podręcznego, 608
DoEvents, 492, 667
dokumentowanie prac projektowych, 44

dostęp
 do poleceń Wstążki, 589
 do rejestru systemu Windows, 285

dostosowywanie
 edytora VBE, 66
 menu podręcznego, 36, 603
 okna Toolbox, 454
 Wstążki, 36, 565

Dostosuj pasek narzędzi Szybki dostęp, 423

Double, 92

Drag-and-Drop Text Editing, 68

DrawMenuBar, 518

drukarka domyślna, 281

drukowanie
 aktualizacja nagłówka lub stopki, 204
 ukrywanie kolumn przed wydrukiem, 205
 wykresy osadzone na arkuszu, 334

Drukuj, 334

DupeRows, 241

Dynamic Link Library, 279

DynamicMenu, 585

dynamiczna zmiana położenia formantów, 515

E

Each...Next, 109

edytor VBE, 25, 48, 66
 Auto
 Data Tips, 67
 Indent, 67
 List Members, 67
 Quick Info, 67
 Syntax Check, 67

błędy składni, 67

Code, 63

Code Colors, 68

czcionki, 68

Default to Full Module View, 68

Docking, 69

dostosowywanie, 66

Drag-and-Drop Text Editing, 68

Editor Format, 68

Immediate, 61

informacje o argumentach funkcji, 67

karta Editor, 66

kod źródłowy języka VBA, 61

kopiowanie i przenoszenie tekstu, 68

lista funkcji, 106

Margin Indicator Bar, 69

okno kodu źródłowego, 63

pasek menu, 60

paski narzędzi, 60

pomoc w trakcie wprowadzania kodu, 67

Procedure Separator, 68

Project Explorer, 61

Properties, 433

Require Variable Declaration, 67

rozmiar czcionki, 69

wcięcie, 67

wprowadzanie kodu źródłowego, 64

efekt podświetlenia okna dialogowego, 532

eksportowanie
 obiektów graficznych, 318
 wykresów, 317

elementy języka VBA, 87

Else, 113

ElseIf, 114

emulowanie
 funkcji MsgBox, 512
 MyMsgBox, 515
 okien dialogowych Excela, 457

EnableCancelKey, 155, 465

EnableEvents, 197

End, 97, 270, 340, 664
 Function, 160, 163, 164
 Select, 118
 Sub, 81, 126, 139
 Type, 105
 With, 104, 108

Enter, 449, 450

EntryIsValid, 211

Enum, 664

Environ, 667

EOF, 667

Eqv, 101

Erase, 664

Err, 141, 142, 143
 Number, 141

Error, 141, 449, 664, 667

etykiety, 429

Event, 664

event handler procedure, 137

ExecuteMso, 420, 589, 590

Exit, 449
 Do, 664
 For, 110, 119, 121, 664
 Function, 664
 Property, 664
 Sub, 126, 144, 233, 664

Exp, 667

EXTRACTELEMENT, 272

F

FaceID, 595, 601, 611

False, 255

FileAttr, 667

- FileCopy, 664
- FileDateTime, 667
- FileDialog, 419
- FileExists, 261
- FileLen, 667
- FileNameOnly, 261, 262
- FILLCOLOR, 266
- Filter, 667
- FilterName, 317
- FindControl, 600
- FindExecutableA, 280
- FindWindowA, 518
- Fix, 667
- fmListStyleOption, 481
- fmMultiSelectExtended, 474
- fmMultiSelectMulti, 474, 481
- fmMultiSelectSingle, 474
- fmTabStyleButtons, 488
- fmTabStyleNone, 488, 507
- FollowHyperlink, 207
- Font, 434
- For, 119
- For Each...Next, 109, 144, 239, 664
- For...Next, 87, 119, 152, 664
 - Step, 120
- formant, 37, 427
 - CheckBox, pole wyboru, 427
 - ComboBox, pole kombi, 428
 - CommandButton, przycisk polecenia, 428, 443, 459
 - DynamicMenu, 585
 - Frame, pole grupy, 428, 442
 - Image, obraz, 327, 428
 - Label, etykieta, 429, 442, 490, 523
 - ListBox, pole listy, 429, 460, 461, 469
 - MultiPage, 429, 487, 501, 503, 507
 - OptionButton, przycisk opcji, 429, 443
 - RefEdit, 429, 461
 - ScrollBar, pasek przewijania, 429, 467, 528
 - SpinButton, pokrętło, 429, 430, 450
 - TabStrip, 430, 487
 - TextBox, pole tekstowe, 430, 442, 491
 - ToggleButton, przycisk przełącznika, 430
 - ViewCustomViews, 589
 - Windows Media Player, 488
- formanty
 - ActiveX, 37–39, 134, 135, 444, 455
 - definiowanie klawiszy skrótu, 437
 - dodawanie do formularza, 426
 - formularza, 39, 135
 - kolejność tabulacji, 436
 - nazwy, 435
 - modyfikacja właściwości, 431, 433
 - okna ToolBox, 427
 - stosowanie w arkuszu, 430
 - wielokolumnowe ListBox, 479
 - wspólne właściwości, 434
 - wstawianie, 426
 - Wstążki, 578
 - wyrównanie, 432
 - wyświetlanie kolekcji, 453
 - zewnętrzne, 488
- Format, 667
- format
 - CSV, 216
 - HTML, 624
 - JPEG, 317
 - MHTML, 625
- FormatCurrency, 667
- FormatDateTime, 667
- FormatNumber, 667
- FormatPercent, 667
- formatowanie komórek, 265, 421
- Formuła, 78, 79, 321
- FormułaArray, 78
- FormułaLocal, 78
- FormułaR1C1, 78, 79
- formularze
 - UserForm, *Patrz* UserForm
 - wprowadzania danych, 422, 423
- formuły formatowania warunkowego, 166
- Frame, 428, 442, 462, 498
- FreeFile, 667
- FullName, 73, 555
- Function, 62, 159, 160, 164, 187, 664
- funkcja, 26, 62, 100
 - Abs, 666
 - Array, 176, 666
 - ArrayFillRange, 245
 - Asc, 666
 - Atn, 666
 - BUBBLESIZE_FROM_SERIES, 322
 - CallByName, 666
 - CBool, 666
 - CByte, 666
 - CCur, 666
 - CDate, 666
 - CDbl, 666
 - CDec, 92, 666
 - Choose, 666
 - Chr, 666
 - CLng, 666
 - CLng, 666
 - Cos, 666
 - COUNTA, 240, 446
 - CreateObject, 347, 666
 - CSng, 666

CStr, 666
CurDir, 666
CVar, 666
CVDate, 666
CVer, 179, 667
CZY.BŁĄD, 242
CZY.LICZBA, 174
CZY.LOGICZNA, 242
CZY.TEKST, 242
DATA, 99
Date, 122, 256, 667
DateAdd, 667
DateDiff, 667
DatePart, 667
DateSerial, 99, 122, 660, 667
DateValue, 667
Day, 667
DDB, 667
Dir, 262, 667
DoEvents, 667
Environ, 667
EOF, 667
Error, 141, 667
Exp, 667
FileAttr, 667
FileDateTime, 667
FileDialog, 419
FileExists, 261
FileLen, 667
FileNameOnly, 261
Filter, 667
Fix, 667
Format, 667
FormatCurrency, 667
FormatDateTime, 667
FormatNumber, 667
FormatPercent, 667
FreeFile, 667
FV, 667
GetAllSettings, 667
GetAttr, 667
GetExitCodeProcess, 363
GetObject, 347, 667
GetSetting, 529, 667
Hex, 667
Hour, 667
IIf, 115, 667
ILE.NIEPUSTYCH, 240, 446
Input, 667
InputBox, 231, 233, 406, 408, 667
InStr, 667
InStrRev, 667
Int, 667

IPmt, 668
IRR, 668
IsArray, 668
IsDate, 242, 668
IsEmpty, 242, 668
IsError, 668
IsMissing, 175, 177, 668
IsNull, 668
ISNUMBER, 174
IsNumeric, 242, 668
IsObject, 668
IsText, 179
JEŻELI, 115
Join, 668
LBound, 668
LCase, 668
Left, 668
Len, 668
LEWY, 175
LICZ.JEŻELI, 138
LICZ.WARUNKI, 269
LITERY.WIELKIE, 105
LoadPicture, 529
Loc, 668
LOF, 668
Log, 668
LoopFillRange, 245
LOS, 138, 170
LTrim, 668
MAX, 275
Mid, 668
Minute, 668
MIRR, 668
Month, 122, 668
MonthName, 668
MsgBox, 87, 102, 149, 411, 668
Now, 668
NPer, 668
NPV, 668
Oct, 668
Partition, 668
PathExists, 262
PIERWIASTEK, 106
Pmt, 668
Ppmt, 668
PV, 668
QBColor, 668
RangeNameExists, 262
Rate, 668
Replace, 668
RGB, 668
Right, 668
Rnd, 668

funkcja

Round, 669
RTrim, 669
RZYMSKIE, 106
SaveAllWorkbooks, 251
SaveSetting, 529
Second, 669
Seek, 669
SERIE, 319, 320, 322, 340
SERIESNAME_FROM_SERIES, 322
Sgn, 669
SheetExists, 263
SheetOffset, 274
Shell, 361, 362, 669
Sin, 669
SLN, 669
Space, 669
Spc, 669
SPELLDOLLARS, 272, 273
Split, 262, 669
Sqr, 106, 669
Str, 669
StrComp, 669
StrConv, 669
String, 669
StrReverse, 669
SUMA, 138, 181, 183
Switch, 669
SYD, 669
Tab, 669
Tan, 669
Time, 669
Timer, 244, 669
TimeSerial, 669
TimeValue, 669
TRANSPONUJ, 177, 247
TRANSPOSE, 247
Trim, 669
TypeName, 669
UBound, 669
UCase, 105, 154, 262, 669
Val, 669
VALUES_FROM_SERIES, 322, 323
VarType, 669
Weekday, 116, 669
WeekdayName, 669
WorkbookIsOpen, 263
WriteReadRange, 244
XDATE, 184
XDATEADD, 184
XDATEDAY, 184
XDATEDIF, 184
XDATEMONTH, 184

XDATEYEAR, 184
XDATEYEARDIF, 184
XVALUE_FROM_SERIES, 323
XVALUES_FROM_SERIES, 322
Year, 669
ZDATEDOW, 184
funkcje, 26, 62, 100
argumenty, 168, 173
argumenty opcjonalne, 175
bezargumentowe, 168
deklaracja, 164
informacje o formatowaniu komórki, 265
jednoargumentowe, 171
kategorie, 189
nieokreślona liczba argumentów, 180
niestandardowe, 190
 arkuszowe, 265
 przechowywanie, 190
opis, 189
osłonowe, 169, 192
pobieranie tablic, 174
ponowne przeliczanie, 170
przypisanie tematów pomocy, 630
rozszerzone funkcje daty, 183
stosowanie, 161
tworzenie, 99, 126, 159
usuwanie błędów, 185
wartość zwracana, 162
wielofunkcyjne, 273
wstawianie, 186
wywołanie, 165
z wieloma argumentami, 173
zasięg, 165
zwracanie
 tablic VBA, 176
 wartości błędu, 178
funkcje Windows API, 190
DrawMenuBar, 518
FindExecutableA, 280
FindWindowA, 518
GetKeyboardState, 635
GetKeyState, 193
GetProfileStringA, 281, 282
GetRegistry, 283
GetSetting, 285
GetSystemDirectory, 561
GetSystemMetrics, 282
GetWindowLong, 518
GetWindowsDirectoryA, 192
RegCloseKey, 283
RegCreateKeyA, 283
RegOpenKeyA, 283
RegQueryValueExA, 283

RegSetValueExA, 283
 SaveSetting, 285
 SetWindowLong, 518
 ShellExecute, 363
 WriteRegistry, 284
 wywołanie, 279
 FV, 667

G

Get, 664
 GetAColor, 528
 GetAFolder, 419
 GetAllSettings, 667
 GetAttr, 667
 GetData, 232
 GetEnabled, 576
 GetEnabledMso, 589, 591
 GetExecutable, 280
 GetExitCodeProcess, 363
 GetImageMso, 590, 591
 GetImportFileName, 416
 GetImportFileName2, 417
 GetKeyboardState, 635
 GetKeyState, 193
 GetLabelMso, 590
 GetObject, 347, 667
 GetOpenFilename, 415
 GetPressed, 576
 GetPressedMso, 590
 GetProfileStringA, 281, 282
 GetRegistry, 283
 GetSaveAsFilename, 418
 GetScreentipMso, 590
 GetSetting, 285, 529, 667
 GetSupertipMso, 590
 GetSystemDirectory, 561
 GetSystemMetrics, 282
 GetUserRange, 234
 GetValue, 231, 264
 GetWindowLong, 518
 GetWindowsDirectoryA, 191, 192, 654
 GIF, 317, 530
 GoSub, 664
 GoTo, 111, 118, 664
 gra
 poker, 534
 układanka, 532

H

hasła, 42, 43
 Height, 466, 504, 524
 Help, 628

HelpFile, 629
 Hex, 667
 Hide, 441
 HideRowsAndColumns, 252
 HKEY_CLASSES_ROOT, 283
 HKEY_CURRENT_CONFIG, 283
 HKEY_CURRENT_USER, 283
 HKEY_LOCAL_MACHINE, 283
 HKEY_USERS, 283
 Hour, 667
 HTML, 318, 624
 Help, 626–628
 Help Viewer, 627

I

ID, 601
 identyfikacja
 formantów, 454
 katalogu domowego, 191
 typu zakresów, 229
 zakresu danych na wykresie, 321
 identyfikator języka, 656
 If, 110, 112, 664
 If...Then...Else, 113
 If...Then...ElseIf, 114
 If, 115, 667
 ikona polecenia menu podręcznego, 611
 ILE.NIEPUSTYCH, 240, 446
 Image, 327, 428, 575
 Picture, 434, 530
 imageMso, 574, 575, 595
 Immediate, 61, 81, 128, 150, 152
 uruchamianie procedur Sub, 137
 wywołanie funkcji, 168
 Imp, 101
 Implements, 664
 implikacja logiczna, 101
 Index, 599
 informacje
 o aktualnej rozdzielczości karty graficznej, 282
 o argumentach funkcji, 67
 o drukarce domyślnej, 281
 o formatowaniu komórki, 265
 InitialFilename, 419
 Initialize, 196, 447, 448, 453, 460
 Input, 664, 667
 InputBox, 231, 233, 406, 408, 667
 Insert/Class Module, 634
 instalacja dodatku, 545
 Installed, 556
 InStr, 667
 InStrRev, 667

Int, 667
Integer, 92
interakcja z innymi aplikacjami, 345
 AppActivate, 365
 Shell, 361
 ShellExecute, 363
 uruchamianie
 aplikacji z poziomu Excela, 361
 okien dialogowych Panelu sterowania, 365
 wyświetlanie okna folderu, 363
interfejs użytkownika, 25, 35
 okna dialogowe, 405
International, 655, 658–660
IPmt, 668
IRibbonControl, 574
IRibbonUI, 586
IRR, 668
IsAddin, 537, 552
IsArray, 668
IsBold, 266
IsDate, 242, 668
IsEmpty, 242, 270, 668
IsError, 668
IsInCollection, 263
IsItalic, 266
ISLIKE, 271
IsMissing, 175, 177, 668
IsNull, 668
ISNUMBER, 174
IsNumeric, 242, 668
IsObject, 668
IsText, 179

J

Jednoplikowa strona sieci Web, 625
język
 aplikacji, 656
 HTML, 318, 624
 MHTML, 625
 programowania ściśle deklarowany, 91
 strukturalny, 119
 VBA, 23, 47, 405, 512
Join, 668
JPEG, 317

K

karty, 25, 454
 aktywacja, 592
 graficzne, 46
katalogi, 419
kategorie funkcji, 188, 189

KeyDown, 449, 450
KeyPress, 449
KeyUp, 449, 450
Kill, 664
klasy, 631
 metody, 638
 stosowanie, 635
 tworzenie, 633
 właściwości, 636
klawiatura, 25, 450
klawisze, 193, 220
klawisze skrótu, 37
kliknięcie obiektu, 135
kod RibbonX, 570
 CustomUI, 577
 IRibbonControl, 574
 Office 2007 Custom UI Part, 574
 Office 2010 Custom UI Part, 574
 procedury zwrotne, 574, 576
 tworzenie
 formantów, 579
 grupy, 579
 karty, 579
 wyświetlanie błędów, 570
kody
 języków, 655
 klawiszy, 193, 221, 222
 spaghetti, 119
 źródłowe VBA, 25, 61, 88
kolejność operatorów, 101
kolekcje, 71, 108
 AddIns, 538, 548, 553
 AddIns2, 553
 Charts, 307
 CommandBars, 598
 Controls, 453
 Dialogs, 419
 Each...Next, 109
 PivotFields, 294
 Shapes, 306
 SparklineGroups, 341
 With...End With, 108
 Workbooks, 548
kolory, 528
komentarze, 87, 89, 163
 do zawartości komórek, 617
kompatybilność aplikacji
 aplikacje dla wielu wersji narodowych, 655
 data i czas, 656
 język aplikacji, 656
 kreator sprawdzania zgodności, 652
 Macintosh, 653
 nazwy plików, 653

kompatybilność aplikacji
 obsługa języka w kodzie VBA, 657
 problemy ze zgodnością, 650
 wersje Excela, 651, 654
 komunikaty błędów, 141
 koniunkcja logiczna, 101
 kontekstowe menu podręczne, 612
 kontekstowy interfejs użytkownika, 25
 kontener formantów, 428
 kontynuacja polecenia w kolejnym wierszu, 88
 konwersja typów danych, 94
 kopiowanie zakresu, 226
 nieciągłego komórek, 249
 o zmiennej wielkości, 227
 kreatory, 506
 Dalej, 508
 MultiPage, 507
 programowanie przycisków, 508
 sprawdzanie zgodności, 652
 Wstecz, 508
 wykonywanie zadań, 511
 zależności programowe, 509

L

Label, 429, 442, 490, 523
 LanguageID, 656
 LastInColumn, 269, 270
 LastInRow, 269, 270
 LastPrinted, 267, 268
 LastSaved, 267, 268
 LastSaved2, 268
 LBound, 668
 LCase, 668
 Left, 668
 Len, 668
 Let, 664
 LEWY, 175
 LICZ.JEŻELI, 138
 LICZ.WARUNKI, 269
 Like, 271
 Line Input, 664
 linie siatki, 102, 255, 307, 431, 564
 LinkedCell, 430
 lista
 czcionek, 258
 pól tabeli przestawnej, 290, 297
 ListBox, 429, 460, 469
 AddItem, 470
 aktywacja arkusza, 483
 BoundColumn, 480
 ColumnCount, 469
 ColumnHeads, 469

ControlSource, 469
 identyfikacja zaznaczonych elementów, 473, 474
 ListIndex, 461, 474
 MultiSelect, 473, 474
 przenoszenie elementów listy, 476
 RowSource, 475, 480
 Selected, 474
 tworzenie listy elementów, 469–472
 Value, 474
 wiele list w jednej kontrolce, 475
 wielokolumnowe formanty, 479
 wybieranie wierszy arkusza, 481
 zmiana kolejności elementów listy, 477
 ListIndex, 461, 474
 listy danych, 422
 literały, 139
 LITERY.WIELKIE, 105
 Load, 440, 664
 LoadPicture, 529
 Loc, 668
 Lock, 664
 LOF, 668
 Log, 668
 lokalizacja wykresu, 303
 Long, 92
 Loop, 123
 LoopFillRange, 245
 LOS, 138, 170
 losowo uporządkowane liczby, 275
 Lset, 664
 LTrim, 668

Ł

ładowanie formularza UserForm, 440
 łańcuchy znaków, 98
 łączenie, 100
 o stałej długości, 99
 o zmiennej długości, 99
 łączenie pliku pomocy z aplikacją, 629

M

Macintosh, 653
 MacroOptions, 187, 189, 630
 Main, 138
 makra, 66
 Makro, 129
 uruchamianie procedur Sub, 129
 Maksymalizuj, 64
 maksymalna wartość we wszystkich arkuszach, 274
 MAX, 275
 MAXALLSHEETS, 275

Me, 440
menedżer dodatków, 540, 545
menu, 459
menu Excela 2003, 604
menu podręczne, 214, 229, 541, 597, 650
 automatyczne
 tworzenie, 611
 usuwanie, 611
Cell, 598–606
 dodawanie elementu, 606
 usuwanie elementu, 608
CommandBar, 597
 odwołania do formantów, 599
 właściwości formantów, 601
CommandBars, 598
dostosowywanie, 36, 603
elementy
 ukrywanie, 612
 wyłączanie, 222, 606, 612
 wyświetlanie, 598, 601
FaceID, 611
ikony poleceń, 611
podmenu
 dodawanie, 608
 usuwanie, 610
resetowanie, 603
VBA, 603
 zdarzenia, 611
metody, 70, 75, 638
MHTML, MIME HyperText Markup Language, 625
Microsoft
 Excel 16.0 Object Library, 134
 Forms 2.0 Object Library, 134
 HTML Help Workshop, 615
 Office 16.0 Object Library, 134
 Office Code Compatibility Inspector, 652
 Office Compatibility Pack, 651
 Microsoft Visual Studio Tools for Office, 24
Mid, 664, 668
Minute, 668
MIRR, 668
MkDir, 664
Mod, 100
modalne okna dialogowe, 493
moduły, 196
 wywołanie procedury, 132
 zmiennne, 96
moduły klas, 331, 495, 631
 dodawanie kodu VBA, 634
 programowanie
 metod, 638
 właściwości obiektów, 636

 tworzenie, 633
 wstawianie, 634
 zdarzenia, 638
modyfikacja
 danych wykresu, 319
 dodatku, 546
 kodu VBA, 59
 komórki, 208
 właściwości formantów, 431–433
 Wstążki, 576, 587
monitorowanie
 zdarzeń poziomu aplikacji, 218
 zmian w wybranym zakresie komórek, 209
Month, 122, 668
MonthName, 668
MouseDown, 329
MouseMove, 329
MouseOver, 337
MouseUp, 329
MoveRange1, 227
MP3, 490
MsgBox, 81, 87, 102, 149, 411, 668
 emulacja funkcji, 512
 stałe przycisków, 412
 wartości zwracane, 413
msoBarTypeMenuBar, 597
msoBarTypeNormal, 597
msoBarTypePopUp, 597
msoLanguageIDUI, 656
MultiPage, 429, 462, 487, 501, 503, 506
 karty, 488
 kreatory, 507
 Style, 488
 Value, 488
MultiSelect, 469, 473, 474
mysz
 zdarzenia, 449

N

Name, 310, 664
natychmiastowe zakończenie procedury, 126
nazwa
 aktywnego skoroszytu, 73
 arkusza, 73, 147
 pliku, 26, 653
 procedury, 127
 zakresu, 26
 zmiennej, 90, 94
nazwane argumenty, 74
negacja logiczna, 101
New Page, 454

NewSheet, 195, 201, 203
 NewWorkbook, 196, 216
 Next, 109, 119
 niemodalne okna
 dialogowe, 493
 formularzy UserForm, 439
 nieoficjalne systemy pomocy, 615
 nierównoważność logiczna, 101
 niestandardowe
 funkcje arkusza, 163
 menu podręczne, 36, 229
 uruchamianie procedur Sub, 131
 okna dialogowe, 37, 425
 paski narzędzi, 592
 procedury Function, 164
 typy danych, 105
 NoDups, 473
 Not, 101, 102, 255
 Nothing, 311, 639
 Now, 668
 NPer, 668
 NPV, 668
 Number, 141
 NumLock, 633

0

obiekty, 70, 103, 108
 AddIn, 554
 Chart, 304, 332
 ChartObject, 304
 CommandBar, 420, 597
 Err, 141
 metody, 80
 nadrzędne, 268
 przypisanie do zmiennej, 104
 Range, 75
 Sparkline, 341
 SparklineGroup, 341
 With...End With, 108
 właściwości, 80
 Object, 92
 Object Browser, 83, 215
 obrazy, 428
 FaceID, 611
 imageMso, 574, 575
 obsługa
 błędów, 34, 141, 235
 Err, 141, 143
 Error, 141
 On Error, 141
 On Error GoTo ErrorHandler, 142

 On Error Resume Next, 141, 144
 języka w kodzie VBA, 657
 przycisków formularza UserForm, 525
 skoroszytów trójwymiarowych, 274
 zdarzeń, 137, 195, 440, 447, 558, 574, 611
 dla wykresów, 329, 331
 wyłączanie obsługi zdarzeń, 197
 ochrona skoroszytu, 42, 155
 Oct, 668
 odczytywanie
 zakresów, 243
 zawartości rejestru systemu Windows, 283
 Odkrywanie, 420
 odwołania
 do innego skoroszytu, 133
 do innych plików z poziomu dodatku, 561
 do formantów formularza UserForm, 452
 do komórek, 173
 do obiektów, 80
 do zakresu, 76
 odwrócone tabele przestawne, 300
 Office 2007 Custom UI Part, 574
 Office 2010 Custom UI Part, 574
 Offset, 78, 79
 okna dialogowe, 405, 512
 formatowanie komórek, 421
 formularze UserForm, 425
 modalne, 493
 niemodalne, 493
 odkrywanie, 420
 Panelu sterowania, 365
 wprowadzania danych, 405–408
 wyświetlanie, 363, 419
 okno powitalne, 463
 określanie
 numeru wersji Excela, 651
 skojarzeń plików, 279, 280
 typu danych, 93, 242
 typu zaznaczonego zakresu, 235
 wymagań użytkownika, 33
 OLE Automation, 134
 On Error, 141, 144, 239, 664
 On Error GoTo, 235
 On Error GoTo ErrorHandler, 142
 On Error Resume Next, 141, 144, 235, 473
 On...GoSub, 665
 On...GoTo, 665
 OnAction, 578, 601
 OnKey, 196, 220–222
 OnTime, 196, 219, 464
 Open, 195, 201, 202, 665
 OperatingSystem, 653

operator, 88, 100
 And, 101, 102
 Eqv, 101
 Imp, 101
 Like, 271
 Mod, 100
 Not, 101, 102, 255
 Or, 101
 Xor, 101
 operatory
 kolejność wykonywania, 101
 logiczne, 101
 porównania, 101
 przypisanie, 100
 opis funkcji, 189
 Option
 Base, 102, 178, 665
 Compare, 665
 Explicit, 94, 558, 665
 Private, 665
 Private Module, 126, 127
 Optional, 175
 OptionButton, 38, 429, 433, 441, 443
 optymalizacja wydajności dodatków, 558
 Or, 101
 Otwieranie, 415

P

pakiet zgodności formatu plików, 651
 Panel sterowania, 365
 ParamArray, 180
 Parent, 268
 Partition, 668
 pasek
 narzędzi, 518, 592
 edytora VBE, 60
 Szybki dostęp, 422
 kod VBA, 593
 tworzenie, 592
 przewijania, 429
 stanu, 497
 Path, 363
 PathExists, 262
 PathSeparator, 262
 pętla, 87, 118
 Do Until, 123, 124
 Do While, 122
 For Each...Next, 109, 144
 For...Next, 119
 While Wend, 124
 zła, 118

Picture, 434, 529, 530, 595, 611
 PIERWIASTEK, 106
 PivotCache, 296
 PivotCaches, 292
 PivotFields, 292, 294
 PivotItems, 292
 PivotTables, 292
 PivotTableUpdate, 207
 planowanie
 aplikacji, 34
 zdarzeń, 219
 plik arkusza, 31
 pliki
 CHM, 615
 GIF, 530
 HTML, 624
 MHTML, 625
 MP3, 490
 przetwarzanie grupy plików, 259
 skojarzenia plików, 279, 280
 ścieżka pliku, 261
 XLAM, 548
 XLSM, 548
 Pmt, 668
 PNG, 317
 pobieranie
 listy czcionek, 258
 nazwy pliku, 415
 wartości z zamkniętego skorysytu, 264
 zakresu wyznaczonego przez użytkownika, 233
 podglądanie zabezpieczonego dodatku, 551
 Pokaż błędy interfejsu użytkownika dodatku, 570
 pokrętko, 430
 pola
 danych, 291
 kategorii, 291
 pole
 grupy, 428
 kombi, 428
 listy, 429
 tekstowe, 337, 430, 618
 wyboru, 427
 polecenia, 25
 Wstążki, 589
 porównania, 101
 potęgowanie, 101
 powielanie wierszy, 240
 powiększanie arkusza, 467
 półprzezroczyste formularze UserForm, 531
 Ppmt, 668
 Print, 150, 665
 PrintEmbeddedCharts, 335

- Private, 95, 126, 127, 164, 191, 665
- problemy ze zgodnością aplikacji, 650
- Procedure Separator, 68
- procedury, 111, 125, 164, 221
 - argumenty, 125
 - Main, 138
 - prywatne, 127
 - publiczne, 127
 - przekazywanie argumentów, 138
 - wywołanie, 125
 - w innym module, 132
 - w innym skoroszytcie, 133
 - Function, 62, 159, 162, 164
 - obsługi zdarzeń, 137, 196, 440, 445
 - argumenty, 199
 - tworzenie, 198
 - Property, 635
 - Get, 634, 637, 665
 - Let, 637, 665
 - Set, 637, 665
 - Sub, 62, 87, 126, 145
 - deklaracja, 126
 - natychmiastowe zakończenie, 126
 - nazwy, 127
 - tworzenie, 150
 - uruchamianie, 128
 - zasięg, 126
 - zwrotne, 570, 574
- programowanie
 - metod, 638
 - strukturalne, 119
 - w języku VBA, 77, 87, 140, 254
 - właściwości obiektów, 636
- Project Explorer, 61
 - dodawanie modułu, 62
 - Modules, 62
 - projekt, 61
- projekt, 49, 61
- projektowanie aplikacji arkusza kalkulacyjnego, 31
 - bezpieczeństwo, 35
 - dokumentacja, 44
 - dostosowywanie menu podręcznego, 36
 - etapy projektowania, 32
 - formanty ActiveX, 38
 - interfejs użytkownika, 35
 - klawisze skrótu, 37
 - niestandardowe okna dialogowe, 37
 - obsługa błędów, 34
 - określanie wymagań użytkownika, 33
 - struktura danych, 34
 - system pomocy, 44
 - wersja Excela, 34, 45
 - wersje językowe, 46
 - wydajność, 35
 - wygląd aplikacji, 43
- Properties, 433
- Property
 - Get, 634, 637, 665
 - Let, 637, 665
 - Set, 637, 665
- Protect, 74
- ProtectStructure, 155
- przechwytywanie błędów, 141
- przeglądarka obiektów, 83
- Przejdź do — specjalnie, 142
- przekazywanie argumentów, 138
 - przez odwołanie, 139
 - przez wartość, 139
- przekształcanie dodatku w skoroszyt, 547
- przenoszenie
 - wykresu, 309
 - zakresów, 227
 - zawartości tablic jednowymiarowych, 246
 - zawartości zakresu do tablicy typu Variant, 247
- przetwarzanie
 - arkuszy, 251
 - dat, 99
 - dodatków za pomocą kodu VBA, 553
 - grupy plików, 259
 - kolekcji, 109
 - komórek zaznaczonego zakresu, 237
 - skoroszytów, 251
 - wykresów, 312
 - zakresów, 226, 228
- przewijane etykiety, 622
- przewijanie
 - arkusza, 467
 - wykresów, 339
- przezroczystość okna, 531
- przycisk, 508
 - dzielony, 564
 - opcji, 429
 - polecenia, 428, 444
 - przełącznika, 430
- przypisanie, 87, 100
 - obiektu do zmiennej, 104
 - tematów pomocy do funkcji VBA, 630
- Przypisz makro, 136
- PtrSafe, 191
- Public, 95, 97, 104, 126, 127, 140, 164, 665
- Public WithEvents, 526
- pułapki, 185
- puste wiersze, 240
- Put, 665
- PV, 668

Q

QBColor, 668
QueryClose, 441, 448, 465
quick-sort, 259

R

RaiseEvent, 665
RandomIntegers, 276
Randomize, 665
Range, 75, 76, 81, 104, 242
 Address, 74
 Cells, 76, 77
RangeNameExists, 262
RangeNameExists2, 262
RangeRandomize, 277
RangeToVariant, 247
Rate, 668
ReDim, 103, 665
RefEdit, 429, 461, 462
References, 133, 134
RegCloseKey, 283
RegCreateKeyA, 283
RegOpenKeyA, 283
RegQueryValueExA, 283
RegSetValueExA, 283
rejestr systemu Windows, 283
 odczytywanie zawartości, 283
 zapisywanie zawartości, 283
Rejestrator makr, 47, 147
 wykresy, 304
Rem, 89, 665
Replace, 668
Require Variable Declaration, 67
Reset, 665
Resize, 329
Resume, 665
Return, 664
ReversePivot, 301, 302
RGB, 668
Ribbon, 25
RibbonX, 570
Right, 668
Rmdir, 665
Rnd, 668
Round, 669
RowSource, 470, 475, 480
rozdzielczość karty graficznej, 282
rozszerzone funkcje daty, 183
równoważność logiczna, 101
RSet, 665
RTrim, 669

Run, 128, 131, 134, 166, 550
rundll32.exe, 366
RZYMSKIE, 106

S

SaveAllGraphics, 318
SaveAllWorkbooks, 251
SaveSetting, 285, 529, 665
Schowek, 564
ScreenUpdating, 154, 328, 441, 558
ScrollBar, 38, 429, 467, 468, 528
ScrollBarZoom, 468
ScrollColumns, 468
ScrollRow, 468
Second, 669
Seek, 665, 669
sekwencje zdarzeń, 196
Select, 115, 195, 229, 230, 329, 665
 Case, 115
 Case Else, 115
SelectByValue, 248
SelectCurrentRegion, 230
Selected, 474
Selection, 73, 235, 236
SelectionChange, 195, 207, 213, 321
SendKeys, 665
separator, 272, 658
seria danych, 320
SERIE, 319–322, 340
Series, 319
 Values, 321
SeriesChange, 195, 329
SeriesCollection, 323
SERIESNAME_FROM_SERIES, 322
Set, 104, 665
SetAttr, 665
SetWindowLong, 518
Sgn, 669
Shape, 73, 306, 309, 337
Shapes, 306
 AddChart, 306
SheetActivate, 196, 199–203, 216, 494
SheetBeforeDoubleClick, 201, 216
SheetBeforeRightClick, 201, 216
SheetCalculate, 201, 216
SheetChange, 196, 201, 216
SheetDeactivate, 196, 201, 216
SheetExists, 263
SheetFollowHyperlink, 201, 216
SheetOffset, 274
SheetPivotTableUpdate, 201, 216
SheetSelectionChange, 201, 216, 494, 495
Shell, 361, 362, 669

- ShellExecute, 363, 364
- Shift, 193
- Show, 438, 439, 463
- ShowDataForm, 423
- ShowInstalledFonts, 258
- Sin, 669
- Single, 92
- skojarzenia plików, 279, 280
- skoroszyt
 - przetwarzanie, 251
 - zamykanie, 251
 - zapisywanie, 251
 - zdarzenia, 201
- SLN, 669
- słowo kluczowe, 88, 91
 - As, 96, 140
 - ByRef, 140
 - ByVal, 139
 - Call, 131
 - Case, 116
 - Const, 97
 - Declare, 191
 - Dim, 95, 105
 - Do, 122, 123
 - Else, 113
 - ElseIf, 114
 - End, 97
 - For, 119
 - Function, 159, 160, 164
 - GoTo, 111, 118
 - If, 110, 112
 - Loop, 123
 - Me, 440
 - Next, 109, 119
 - Optional, 175
 - ParamArray, 180
 - Private, 95, 126, 164, 191
 - PtrSafe, 191
 - Public, 97, 127, 140, 164
 - ReDim, 103
 - Rem, 89
 - Select, 115
 - Set, 104
 - Static, 95, 97, 126, 164
 - Step, 120
 - Sub, 126
 - Then, 110, 112
 - To, 102
 - Type, 105
 - Until, 123
 - While, 122
 - With, 104, 108
 - WithEvents, 215
- SmartArt, 136
- sortowanie, 145
 - arkuszowe, 259
 - bąbelkowe, 150, 259
 - szybkie, quick-sort, 259
 - tablicy, 259
 - zliczające, 259
- SortSheets, 156, 157
- SourceData, 319
- Space, 669
- spaghetti, 119
- Sparkline, 341
- SparklineGroup, 341
- SparklineGroups, 341
- sparklines, 340
- Spc, 669
- SpecialCells, 142
- SpecialEffect, 499, 520
- SPELLDOLLARS, 272, 273
- SpinButton, 429, 430, 447, 450
 - TextBox, 450
 - zdarzenia, 449
- SpinDown, 449, 450
- SpinUp, 447, 449, 450
- splash screen, 463
- Split, 262, 669
- sprawdzanie
 - poprawności danych, 210
 - przynależności obiektu do kolekcji, 263
 - stanu skoroszytu, 216
 - stanu wykresu, 311
 - zgodności, 293, 652
- Sqr, 106, 669
- stałe, 90, 97
 - deklaracja, 97
 - predefiniowane, 98
- startFromScratch, 579
- StatFunction, 273
- Static, 95, 97, 126, 164, 665
- StatusBar, 497
- Step, 120
- sterowanie
 - ponownym przeliczaniem funkcji, 170
 - wykonywaniem kodu, 111
- Stop, 665
- stosowanie formantów w arkuszu, 430
- Str, 669
- StrComp, 669
- StrConv, 669
- String, 92, 99, 669
- StrReverse, 669

struktura
 danych, 34
 plików aplikacji, 34
 Sub, 62, 81, 87, 112, 126, 139, 665
 SUMA, 138, 181, 183
 suma logiczna, 101
 Switch, 669
 SYD, 669
 symulacja paska narzędzi, 518
 synchronizacja arkuszy, 254
 SynchSheets, 254
 system pomocy w aplikacjach, 44, 82, 615
 arkusz, 620
 etykiety, 621
 formularze UserForm, 621
 Help, 628
 HTML Help, 626, 627
 kategorie, 615
 komentarze do zawartości komórek, 617
 komponenty Excela, 617
 łączenie pliku pomocy z aplikacją, 629
 nieoficjalny, 615
 pliki HTML, 624
 pliki MHTML, 625
 pola tekstowe, 618
 pole kombi, 623
 przewijane etykiety, 622
 przypisanie tematów pomocy do funkcji VBA, 630
 wybór tematów pomocy, 623
 wyświetlanie tekstu pomocy, 621, 624
 szablon formularzy UserForm, 456
 Szybki dostęp, 422

Ś

ścieżka pliku, 261

T

Tab, 436, 669
 Tab Order, 437
 tabele bazy danych, 291
 tabele przestawne, 289
 CreatePivotTable, 292, 296
 dane źródłowe, 291
 jednoczesne tworzenie, 298
 kompatybilność, 293
 Narzędzia tabel przestawnych, 297
 odwrócone, 300
 optymalizacja wygenerowanego kodu, 292
 PivotCache, 296

 PivotCaches, 292
 PivotFields, 292, 294
 PivotItems, 292
 PivotTables, 292
 Pola tabeli przestawnej, 290
 ReversePivot, 301, 302
 tworzenie, 290
 tabele znormalizowane, 291
 TabIndex, 437
 tablice, 102
 deklaracja, 102
 dynamiczne, 103
 Option Base, 102
 sortowanie, 259
 wielowymiarowe, 103
 TabStrip, 430, 487
 Tag, 452
 Tan, 669
 techniki programowania, 254
 Terminate, 441, 448
 TestGetValue2, 264
 testowanie, 150, 154
 aplikacji, 40
 dodatków, 546
 formularzy UserForm, 438, 457
 okna dialogowego, 445
 polecen języka VBA, 61
 wersji beta, 41, 566, 629, 640
 Text, 78
 TextBox, 430, 442, 491
 SpinButton, 450
 Then, 110, 112
 ThisWorkbook, 73, 197, 199, 463, 537
 Time, 665, 669
 Timer, 244, 669
 TimeSerial, 669
 TimeValue, 669
 To, 102
 ToggleButton, 430
 TogglePageBreakDisplay, 576
 ToggleWrapText, 255
 Toolbox
 Additional Controls, 488
 dodawanie
 formantów ActiveX, 455
 kart, 454
 dostosowywanie formantów, 454
 łączenie formantów, 454
 ToolTipText, 601
 TRANSPONUJ, 177, 247
 Transpose, 247, 471
 TRANSPOSE, 247

Trim, 669
 trójwymiarowe skoroszyty, 274
 True, 255
 tryb

- karty graficznej, 46
- projektowania, 431
- tylko do odczytu, 42

tworzenie

- dodatków, 541
- formularzy UserForm, 425, 441, 457
 - półprzezroczystych, 531
- funkcji, 99, 126, 159
- kart, 579
- klas, 633
- klawiszy skrótu, 37
- kodu języka VBA, 59, 149
- menu podręcznego, 612
- kreatorów, 506
- listy elementów formantu ListBox, 469
- modułów klas, 331, 633
- odwołań, 133
- okien dialogowych, 37
- okna powitalnego, 463
- pasków narzędzi, 592
- procedur, 111, 125, 150, 164, 221
- procedur obsługi zdarzeń, 198, 445
- wskaźnika postępu zadania, 498
- systemu pomocy, 44
- szablonów formularzy UserForm, 456
- tabel przestawnych, 290
 - jednoczesne, 298
 - odwróconych, 300
 - złożonych, 294
- wykresów, 306, 335, 340

typ danych, 90–93

- Boolean, 92
- Byte, 92
- Currency, 92
- Date, 92, 99, 100
- Decimal, 92
- Double, 92
- Integer, 92
- Long, 92
- Object, 92
- Single, 92
- String, 92
- Variant, 92

Type, 105, 665

TypeName, 669

typy

- danych użytkownika, 105
- zaznaczeń zakresów, 235

U

uaktualnianie zawartości ekranu, 154

UBound, 669

UCase, 105, 154, 262, 669

udostępnienie skoroszytu, 42

układanka, 532

ukrywanie

- arkuszy, 42
- dokumentów, 42
- elementów menu podręcznego, 612
- formularza UserForm, 441
- formuł, 42
- kolumn, 42
- kolumn przed wydrukiem, 205
- komórek arkusza, 252
- linii siatki, 102
- wierszy, 42

Uncomment Block, 90

unikatowe liczby całkowite, 275

Unload, 440, 665

Unlock, 664

Until, 123

uodpornianie aplikacji na błędy, 41

UpdateDynamicRibbon, 586

uruchamianie

- aplikacji z poziomu Excela, 361
- edytora VBE, 50
- okien dialogowych Panelu sterowania, 365
- procedur Sub, 128
 - Immediate, 137
 - klawisz skrótu Ctrl, 130
 - kliknięcie obiektu, 135
 - Makro, 129
 - Run Sub/UserForm, 128
 - Wstążka, 131
 - wywołanie, 131

UsedRange, 240

UserForm, 37, 405, 415, 425, 459, 517

- Activate, 447
- animacja etykiet, 490
- automatyczna aktualizacja, 495
- Caption, 498
- ControlTipText, 622
- definiowanie klawiszy skrótu, 437
- dodawanie
 - formantów, 426
 - procedur obsługi zdarzeń, 445
- formanty, 426
 - kolejność przechodzenie, 436
 - modyfikacja, 431
 - odwołania, 452
 - zewnętrzne, 488
 - zmiana położenia, 515

UserForm

- formularze, 440, 517
 - półprzezroczyste, 531
 - wstawianie, 426
 - wyświetlanie, 438, 439
 - zamykanie, 440
 - zmiana wielkości, 465
- Initialize, 447
- kreatory, 506
- menu, 459
- MultiPage, 487
- niemodalne okna dialogowe, 493
- obsługa
 - wielu przycisków, 525
 - zdarzeń, 440, 445
- okno powitalne, 463
- powiększanie arkusza, 467
- przewijanie arkusza, 467
- samodzielny wskaźnik postępu zadania, 498
- symulacja paska narzędzi, 518
- system pomocy w aplikacjach, 621
- szablony, 456
- Tag, 452
- testowanie, 438, 445, 457
- tworzenie, 425, 441, 457
- ukrywanie, 441
- wskaźnik postępu zadania, 497
- wybór koloru, 528
- wykresy, 327, 529
- zaznaczanie zakresów, 461
- ustawienia międzynarodowe, 655
- usuwanie
 - błędów, 61, 67, 185
 - elementu podręcznego Cell, 608
 - elementu z kolekcji ChartObjects, 311
 - podmenu, 610
 - problemów, 154
 - pustych wierszy, 240

V

- Val, 669
- Value, 78, 435
- Values, 319, 322
- VALUES_FROM_SERIES, 322, 323
- Variant, 92, 114, 162, 176, 247
- VarType, 669
- VBA, Visual Basic for Applications, 23, 47, 134, 405, 512
 - błędy, 141
 - definiowanie typów danych, 91
 - deklaracja zmiennych, 87
 - długie polecenia, 88

- funkcje, 100, 105
- kolejność operatorów, 101
- kolekcje, 70, 108
- komentarze, 87, 89
- konwersja typów danych, 94
- łańcuchy znaków, 98
- metody, 72
- moduły klas, 631
- nazwy zmiennych, 90
- obiekty, 70, 103, 108
- obsługa błędów, 141
- obsługa języka aplikacji, 657
- operatory, 88, 100
 - logiczne, 101
 - porównania, 101
- Option Explicit, 94
- pętle, 118–124
- procedury, 111, 125, 164, 221
 - Sub, 87
 - wywołanie, 131
 - zwrotne, 570
- projekt, 49, 61
- przetwarzanie dat, 99
- przypisanie, 100
- słowa kluczowe, 91
- stałe, 90, 97, 412
- sterowanie wykonywaniem kodu, 111
- tablice, 102
- typy danych, 90, 92
 - użytkownika, 105
- właściwości, 72
- wyrażenia, 100
- zmiennie, 90, 95
 - globalne, 97
 - lokalne, 95
 - obiektywne, 103
 - statyczne, 97
- VBA7, 654
- vbAbort, 413
- vbAbortRetryIgnore, 412
- vbCancel, 413
- vbCritical, 412
- vbCrLf, 256
- vbDefaultButton1, 412
- vbDefaultButton2, 412
- vbDefaultButton3, 412
- vbDefaultButton4, 412
- VBE, 48
- vbExclamation, 412
- vbFormControlMenu, 465
- vbIgnore, 413
- vbInformation, 412
- vbModeless, 464, 493

vbMsgBoxHelpButton, 412
 vbNo, 107, 413
 vbOK, 413
 vbOKCancel, 412
 vbOKOnly, 412
 vbQuestion, 107, 412
 vbRetry, 413
 vbRetryCancel, 412
 vbSystemModal, 412
 vbYes, 107, 413
 vbYesNo, 107, 412
 vbYesNoCancel, 412
 ViewCustomViews, 589
 Volatile, 265
 VSTO, 24

W

warstwa rysunkowa, 38
 wartości logiczne, 255
 wbudowane funkcje VBA, 105
 wcięcia, 67
 Weekday, 116, 669
 WeekdayName, 669
 wersje
 Excels, 45, 651
 językowe, 46
 While Wend, 124, 665
 widoczność plików
 XLAM, 548
 XLSM, 548
 Width, 524
 Width #, 665
 Win64, 654
 WindowActivate, 201, 216
 WindowDeactivate, 201, 216
 WindowResize, 201, 216
 Windows API, 190, 279
 64-bitowa wersja Excels, 191
 funkcje, 191
 Windows Explorer, 363
 Windows Media Player, 488
 tryb niemodalny, 489
 URL, 489
 With, 104, 108, 665
 WithEvents, 215
 Wklej, 335
 właściwości, 72, 80, 636
 Accelerator, 438
 ActiveCell, 72
 do odczytu i zapisu, 637
 Formuła, 78
 lokalne, 658
 Offset, 78
 Parent, 268
 Range, 75
 TabIndex, 437
 Tag, 452
 tylko do odczytu, 637
 typu logicznego, 255
 włączenie obsługi zdarzeń poziomu aplikacji, 215
 WordArt, 136
 Workbook, 304
 Workbook_Open, 611
 WorkbookActivate, 216
 WorkbookAddinInstall, 216
 WorkbookAddinUninstall, 216
 WorkbookBeforeClose, 196, 216
 WorkbookBeforePrint, 216
 WorkbookBeforeSave, 216
 WorkbookDeactivate, 216
 WorkbookIsOpen, 263
 WorkbookNewSheet, 196, 216
 WorkbookOpen, 216
 Workbooks, 548
 Worksheet, 75
 Range, 76
 UsedRange, 240
 WorksheetFunction, 106
 Worksheets, 76
 wprowadzanie
 danych, 25, 405
 kodu źródłowego języka VBA, 64, 430, 435, 438, 452, 457
 wartości do komórki, 231, 232
 wrapper function, 169, 192
 Write #, 665
 WriteReadRange, 244
 WriteRegistry, 284
 wskaźnik postępu zadania, 497
 formularz UserForm, 501
 MultiPage, 503
 procedura startowa, 501
 UpdateProgress, 502
 wyświetlanie, 503
 Wstaw funkcję, 161
 wstawianie
 formularza UserForm, 426
 funkcji, 166, 186
 kategorie funkcji, 188
 opis funkcji, 189
 modułu klasy, 634
 Wstążka, 25, 563
 aktywacja karty, 592
 błędy kodu RibbonX, 570
 dostosowywanie, 36, 565

Wstążka

- dynamicMenu, 585
- formanty, 578
- karty, 25
- modyfikacja, 587
- procedury Sub, 131
- przyciski dzielone, 564
- tworzenie
 - grupy, 579
 - karty, 579
- VBA, 589
- wstępne rejestrowanie makr, 147
- wybór
 - katalogu, 419
 - koloru, 528
- wydajność, 35
 - dodatków, 558
 - systemu, 46
- wygląd aplikacji, 43
- wykonywanie
 - poleceń ze starego menu, 421, 423
 - procedur Sub, 128, 137
- wykresy, 303, 306
 - aktywacja, 308
 - animowane, 339
 - arkusze wykresu, 307
 - Chart, 304, 332
 - ChartObject, 304, 314
 - deaktywacja, 310
 - drukowanie, 334
 - eksportowanie, 317, 318
 - formularze UserForm, 327, 529
 - identyfikacja zakresu danych, 321
 - lokalizacja wykresu, 303
 - modyfikacja danych, 319
 - procedury obsługi zdarzeń, 331, 332
 - przebiegu w czasie, 340, 341
 - przenoszenie, 309
 - przetwarzanie wszystkich wykresów, 312
 - Rejestrator makr, 304
 - seria danych, 320
 - sprawdzanie stanu aktywacji, 311
 - statyczne, 335
 - usuwanie elementów, 311
 - wyrównywanie, 314
 - wyświetlanie etykiet danych, 324
 - wyświetlanie tekstu, 337
 - zapisywanie do pliku GIF, 317, 530
 - zdarzenia, 329
 - zmiana danych, 319
 - zmiana rozmiarów wykresu, 314
- wykrywanie
 - błędów, 185
 - wciśnięcia klawisza Shift, 193
- wyłączanie
 - menu podręcznego, 222, 606, 612
 - obsługi zdarzeń, 197
 - przycisku Zamknij, 465
- wymagania użytkownika, 33
- wymuszanie deklarowania wszystkich zmiennych, 94
- wyrażenia, 100
- wyrównanie
 - formantów, 432
 - wykresów, 314
- wyszukiwanie
 - formantu, 600
 - obrazów FaceID, 611
 - zdarzeń, 215
- wyświetlanie
 - błędów kodu RibbonX, 570
 - czasu, 255
 - daty, 255
 - wydrukowania pliku, 267
 - zapisania pliku, 267
 - etykiet danych na wykresie, 324
 - formularza
 - UserForm, 438
 - wprowadzania danych, 422, 423
 - komunikatów, 107, 411
 - komunikatów o błędach, 155
 - linii siatki, 102
 - menu podręcznych, 598
 - okien
 - dialogowych, 419, 444
 - folderu, 363
 - niemodalnych, 439
 - pomocy
 - w formacie HTML Help, 628
 - w oknie przeglądarki sieciowej, 624
 - wskaźnika postępu zadania, 497
 - wykresów w formularzu UserForm, 327, 529
- wywołania funkcji Windows API, 279
- wywołanie procedury
 - Sub, 125, 135
 - zawartej w innym module, 132
 - zawartej w innym skoroszytcie, 133
 - Function z poziomu, 165
 - formuły arkusza, 166
 - formuły formatowania warunkowego, 166
 - innej procedury, 131, 165
 - okna Immediate, 168
- wyznaczanie ostatniej niepustej komórki, 269

X

XDATE, 184
 XDATEADD, 184
 XDATEDAY, 184
 XDATEDIF, 184
 XDATEMONTH, 184
 XDATEYEAR, 184
 XDATEYEARDIF, 184
 xl4DigitYears, 660
 XLA, 538, 545
 xlAlternateArraySeparator, 659
 XLAM, 538, 545, 548
 xlColumnSeparator, 659
 xlCommentIndicatorOnly, 618
 xlCountryCode, 655, 659
 xlCountrySetting, 659
 xlCurrencyBefore, 660
 xlCurrencyLeadingZeros, 660
 xlCurrencyMinusSign, 660
 xlCurrencySpaceBefore, 660
 xlCurrencyTrailingZeros, 660
 xlDateSeparator, 659
 xlDayCode, 659
 xlDayLeadingZero, 660
 xlDecimalSeparator, 658, 659
 xlDisabled, 155
 xlErrDiv0, 179
 xlErrNA, 179
 xlErrName, 179
 xlErrNull, 179
 xlErrNum, 179
 xlErrRef, 179
 xlErrValue, 179
 XLFile, 638
 xlHourCode, 659
 XLL, 545
 xlLandscape, 98
 xlLeftBrace, 659
 xlLeftBracket, 659
 xlListSeparator, 659
 xlLowerCaseColumnLetter, 659
 xlLowerCaseRowLetter, 659
 xlMDY, 660
 xlMetric, 660
 xlMinuteCode, 659
 xlMonthCode, 659
 xlMonthLeadingZero, 660
 xlNonEnglishFunctions, 660
 xlRightBrace, 659
 xlRightBracket, 659
 xlRowField, 296
 xlRowSeparator, 659

xlSecondCode, 659
 xlSeries, 338
 XLSM, 537, 544, 545, 548
 xlThousandsSeparator, 659
 xlTimeLeadingZero, 660
 xlTimeSeparator, 659
 xlToLeft, 229
 xlToRight, 229
 xlUp, 229, 270
 xlUpperCaseColumnLetter, 659
 xlUpperCaseRowLetter, 659
 xlYearCode, 659
 XMLcode, 587
 Xor, 101
 XVALUE_FROM_SERIES, 323
 Xvalues, 319
 XValues, 319, 322, 335
 XVALUES_FROM_SERIES, 322

Y

Year, 669

Z

zakresy, 226

- identyfikacja typu, 229, 235, 242
- kopiowanie zakresu, 226
 - nieciągłego, 249
 - o zmiennej wielkości, 227
- odczytywanie, 243
- określanie zawierania się zakresów, 242
- pobieranie zakresu wyznaczonego przez użytkownika, 233
- przenoszenie zawartości, 227
 - tablic jednowymiarowych, 246
 - do tablic typu Variant, 247
- przetwarzanie, 228, 237
- usuwanie pustych wierszy, 240
- zapisywanie, 243, 245
- zaznaczanie, 229, 235
 - na podstawie wartości, 248
- zliczanie komórek, 235

 zamiana wartości na słowa, 272
 Zamknij, 465
 zamykanie

- formularza UserForm, 440
- wszystkich skoroszytów, 251

 zapisywanie

- wszystkich skoroszytów, 251
- wykresu do pliku GIF, 317, 530

 zakresów, 243, 245
 zawartości rejestru systemu Windows, 283

- zasięg
 - funkcji, 165
 - procedury, 126
 - zmiennych, 95
- zastrzeżone słowa kluczowe, 91
- zaznaczanie
 - komórek na podstawie wartości, 248
 - zakresów, 229, 461
- zdarzenia, 137, 195, 440, 611
 - aplikacji, 196, 215
 - arkuszy, 195
 - formularzy UserForm, 196
 - klawiatury, 220, 450
 - menu podręczne, 611
 - moduły klas, 638
 - monitorowanie, 209, 218
 - myszy, 449
 - obsługa zdarzeń, 137, 195, 215, 329, 440, 447, 494, 558, 574, 611
 - sekwencje zdarzeń, 196
 - skoroszytu, 195, 201
 - wykonywanie procedury, 137
 - wykresu, 195
 - wyłączanie obsługi zdarzeń, 197
 - wyszukiwanie, 215
- zdarzenie, 137, 195, 440, 611
 - Activate, 201, 202, 207, 329, 447, 448
 - AddInInstall, 201, 558, 560
 - AddInUninstall, 201, 558
 - AfterCalculate, 216
 - AfterPrint, 205
 - AfterSave, 201
 - AfterUpdate, 449
 - BeforeClose, 201, 205
 - BeforeDoubleClick, 207, 214, 329
 - BeforeDragOver, 449
 - BeforeDropOrPaste, 449
 - BeforePrint, 201, 204
 - BeforeRightClick, 207, 214
 - BeforeSave, 195, 201, 203
 - BeforeUpdate, 449
 - Button_Click, 193
 - Calculate, 195, 207, 329
 - Change, 207, 208, 209, 447–451
 - Click, 508
 - Deactivate, 201, 204, 207, 329, 448
 - Enter, 449, 450
 - Error, 449
 - Exit, 449
 - FollowHyperlink, 207
 - Initialize, 196, 447, 448, 453
 - KeyDown, 449, 450
 - KeyPress, 449
 - KeyUp, 449, 450
 - MouseDown, 329
 - MouseMove, 329
 - MouseOver, 337
 - MouseUp, 329
 - NewSheet, 195, 201, 203
 - NewWorkbook, 196, 216
 - OnKey, 196, 220
 - OnTime, 196, 219
 - Open, 195, 201, 202
 - PivotTableUpdate, 207
 - QueryClose, 441, 448, 465
 - Resize, 329
 - Select, 195, 329
 - SelectionChange, 195, 207, 213
 - SeriesChange, 195, 329
 - SheetActivate, 196, 201, 203, 216, 494, 495
 - SheetBeforeDoubleClick, 201, 216
 - SheetBeforeRightClick, 201, 216
 - SheetCalculate, 201, 216
 - SheetChange, 196, 201, 216
 - SheetDeactivate, 196, 201, 216
 - SheetFollowHyperlink, 201, 216
 - SheetPivotTableUpdate, 201, 216
 - SheetSelectionChange, 201, 216, 494, 495
 - SpinDown, 449, 450
 - SpinUp, 447, 449, 450
 - Terminate, 441, 448
 - WindowActivate, 201, 216
 - WindowDeactivate, 201, 216
 - WindowResize, 201, 216
 - Workbook_Open, 611
 - WorkbookActivate, 216
 - WorkbookAddinInstall, 216
 - WorkbookAddinUninstall, 216
 - WorkbookBeforeClose, 196, 216
 - WorkbookBeforePrint, 216
 - WorkbookBeforeSave, 216
 - WorkbookDeactivate, 216
 - WorkbookNewSheet, 196, 216
 - WorkbookOpen, 216
- ZDATEDOW, 184
- zgodność aplikacji, 649
- zliczanie komórek, 235, 269
- złe pętle, 118
- złożone tabele przestawne, 296
- zmiana
 - danych prezentowanych na wykresie, 319
 - kolejności tabulacji formantów, 436
 - wielkości formularza UserForm, 465
- zmiennie, 67, 87, 90
 - deklaracja, 87, 93
 - globalne, 97

konwersja typów danych, 94	?, 81, 168
lokalne, 95	\, 100, 393
nazwy, 90	^, 100, 101
obiektywne, 103, 104	_, 26, 64, 88
obowiązujące na obszarze całego modułu, 96	+, 100
publiczne, 140	<, 101
statyczne, 97	<=, 101
wymuszanie deklarowania, 94	<>, 101
zasięg, 95	=, 100, 101
Znajdowanie i zamienianie, 465	>, 101
znaki	>=, 101
#, 280	apostrofu, 90
&, 100, 600	kropki, 104
*, 100	spacji, 76
-, 100	tabulacji, 414
/, 100	Zoom, 468

PROGRAM PARTNERSKI

— GRUPY HELION —

- 
1. ZAREJESTRUJ SIĘ
 2. PREZENTUJ KSIĄŻKI
 3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion 

VBA dla Excela: niemożliwe staje się proste!

Możliwości arkusza MS Excel są imponujące. Uważa się, że nikt na świecie nie zna jego wszystkich funkcji i narzędzi. Mimo to wiele osób odczuwa potrzebę wykroczenia poza standardowo dostępne opcje Excela. Często jest to automatyzacja jakiegoś żmudnego zadania albo utworzenie narzędzia do specjalnych celów. I właśnie dla takich użytkowników przygotowano VBA — język, w którym można napisać prosty program do przetwarzania danych, własny dodatek do Excela albo nawet profesjonalną aplikację. A do tego programowania w VBA można nauczyć się błyskawicznie!

Ta książka jest jedynym w swoim rodzaju źródłem wiedzy o VBA i jego wykorzystywaniu do najróżniejszych zadań. Bardzo złożone zagadnienia zaprezentowano tu w prosty i przystępny sposób, koncentrując się na praktycznym wykorzystywaniu kodu VBA. Znalazło się tutaj wiele wskazówek, porad i ciekawych rozwiązań, które w połączeniu z pełnymi kodami programów i przykładowymi skoroszytami umożliwiają szybkie zrozumienie poszczególnych kwestii. Pokazano krok po kroku, jak pisać programy do automatyzacji wielu zadań w Excelu. Książka zawiera wszystkie wiadomości potrzebne do nauki rejestrowania prostych makr, pisania kodu, a także do tworzenia wyrafinowanych narzędzi i aplikacji.

W tej książce między innymi:

- dynamiczna praca ze skoroszytami i z arkuszami
- automatyzacja operacji na tabelach przestawnych i wykresach
- przetwarzanie danych z plików i innych źródeł
- seryjna korespondencja elektroniczna bezpośrednio z Excela
- projektowanie elementów interfejsu i korzystanie ze Wstążki
- tworzenie i udostępnianie własnych dodatków do Excela

MICHAEL ALEXANDER specjalizuje się w zaawansowanej analizie biznesowej z użyciem MS Access i MS Excel. Napisał kilka książek o tej tematyce. Uzyskał certyfikat MCAD (Microsoft Certified Application Developer) oraz tytuł MVP (Most Valuable Professional).

DICK KUSLEIKA jest wielokrotnym zdobywcą tytułu MVP. Od ponad 20 lat zajmuje się pakietem Microsoft Office, a swoją ogromną wiedzę dzieli się na forach internetowych, konferencjach i w książkach. Opracowuje systemy bazujące na programach Access i Excel oraz szkoli w zakresie zaawansowanej obsługi Office.

  helion.pl  HELION SA ul. Kościuski 1c 44-100 Gliwice tel.: 32 230 98 63 helion@helion.pl	<i>Sprawdź nasze szkolenia!</i>  AKADEMIA IT & BUSINESS HELIONSZKOLENIA.PL	KOD KORZYŚCI <i>Sięgnij po więcej!</i>  ISBN 978-83-283-6634-3  9 788328 366343
INFORMATYKA W NAJLEPSZYM WYDANIU		Cena: 99,00 zł

WILEY