

Microsoft® Excel® 2016 PL Programowanie w VBA

Michael Alexander
Dick Kusleika



Helion 

Tytuł oryginału: Excel 2016 Power Programming with VBA

Tłumaczenie: Grzegorz Kowalczyk

ISBN: 978-83-283-2858-7

Copyright © 2016 by John Wiley & Sons, Inc., Indianapolis, Indiana

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

Translation copyright © 2017 by Helion SA

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise without either the prior written permission of the Publisher

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. Excel is a registered trademark 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.

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 Wydawnictwo HELION 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 Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION
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)

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
<ftp://ftp.helion.pl/przyklady/e26pvw.zip>

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
<http://helion.pl/user/opinie/e26pvw>
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

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

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

Spis treści

O autorach	19
O korektorze merytorycznym	19
Przedmowa	21
Zakres zagadnień	21
Co musisz wiedzieć?	22
Czym musisz dysponować?	22
Konwencje zastosowane w książce	23
Polecenia Excela	23
Polecenia edytora VBA	24
Konwencje związane z klawiaturą	24
Znaczenie ikon	25
Struktura książki	26
Część I: Wprowadzenie do języka Excel VBA	26
Część II: Zaawansowane techniki programowania	26
Część III: Praca z formularzami UserForm	26
Część IV: Tworzenie aplikacji	27
Część V: Dodatek	27
Przykłady	27
Narzędzie Power Utility Pak	27
Część I. Wprowadzenie do języka Excel VBA	29
<hr/>	
Rozdział 1. Podstawy projektowania aplikacji arkusza kalkulacyjnego	31
Czym jest aplikacja arkusza kalkulacyjnego?	31
Etapy projektowania aplikacji	33
Określanie wymagań użytkownika	33
Planowanie aplikacji spełniającej wymagania użytkownika	35
Wybieranie odpowiedniego interfejsu użytkownika	37
Dostosowywanie Wstążki do potrzeb użytkownika	38
Dostosowywanie menu podręcznego do potrzeb użytkownika	38
Tworzenie klawiszy skrótów	39
Tworzenie niestandardowych okien dialogowych	39
Zastosowanie formantów ActiveX w arkuszu	40
Rozpoczęcie prac projektowych	42
Zadania realizowane z myślą o końcowym użytkowniku	43
Testowanie aplikacji	43
Uodpornianie aplikacji na błędy popełniane przez użytkownika	45
Nadawanie aplikacji przyjaznego, intuicyjnego i estetycznego wyglądu	47

Tworzenie systemu pomocy i dokumentacji przeznaczonej dla użytkownika	48
Dokumentowanie prac projektowych	48
Przekazanie aplikacji użytkownikom	49
Aktualizacja aplikacji (kiedy to konieczne)	49
Inne kwestie dotyczące projektowania	50
Wersja Excela zainstalowana przez użytkownika	50
Wersje językowe	50
Wydajność systemu	51
Tryby karty graficznej	51
Rozdział 2. Wprowadzenie do języka VBA	53
Rejestrator makr Excela	53
Tworzenie pierwszego makra	54
Porównanie rejestrowania makr z odwołaniami względnymi i bezwzględnymi	58
Inne zagadnienia związane z makrami	63
Praca z edytorem Visual Basic Editor (VBE)	68
Podstawowe elementy edytora VBE	68
Tajemnice okna Project	70
Tajemnice okna Code	72
Dostosowywanie środowiska edytora Visual Basic	76
Karta Editor Format	78
Karta General	79
Karta Docking	80
Podstawowe informacje o języku VBA	80
Obiekty	81
Kolekcje	82
Właściwości	82
Tajemnice obiektów Range	87
Wyszukiwanie właściwości obiektów Range	87
Właściwość Range	87
Właściwość Cells	89
Właściwość Offset	92
Podstawowe zagadnienia, które należy zapamiętać	93
Nie panikuj — nie jesteś sam	95
Przeczytaj resztę książki	95
Pozwól Excelowi napisać makro za Ciebie	96
Korzystaj z systemu pomocy	96
Używaj przeglądarki obiektów	97
Szukaj kodu w internecie	98
Wykorzystuj fora dyskusyjne użytkowników Excela	99
Odwiedzaj blogi ekspertów	99
Poszukaj szkolenia wideo na YouTube	100
Ucz się z Microsoft Office Dev Center	100
Analizuj inne aplikacje Excela, które są używane w Twojej organizacji	101
Zapytaj lokalnego guru	101

Rozdział 3. Podstawy programowania w języku VBA	103
Przegląd elementów języka VBA	103
Komentarze	106
Zmienne, typy danych i stałe	107
Definiowanie typów danych	108
Deklarowanie zmiennych	110
Zasięg zmiennych	113
Zastosowanie stałych	116
Praca z łańcuchami tekstu	118
Przetwarzanie dat	118
Instrukcje przypisania	120
Tablice	122
Deklarowanie tablic	123
Deklarowanie tablic wielowymiarowych	123
Deklarowanie tablic dynamicznych	124
Zmienne obiektowe	124
Typy danych definiowane przez użytkownika	126
Wbudowane funkcje VBA	127
Praca z obiektami i kolekcjami	130
Polecenie With ... End With	130
Polecenie For Each ... Next	131
Sterowanie sposobem wykonywania procedur	133
Polecenie GoTo	134
Polecenie If ... Then	135
Polecenie Select Case	139
Wykonywanie bloku instrukcji w ramach pętli	143
Rozdział 4. Tworzenie procedur w języku VBA	153
Kilka słów o procedurach	153
Deklarowanie procedury Sub	154
Zasięg procedury	155
Wykonywanie procedur Sub	157
Uruchamianie procedury przy użyciu polecenia Run Sub/UserForm	158
Uruchamianie procedury z poziomu okna dialogowego Makro	158
Uruchamianie procedury przy użyciu skrótu z klawiszem Ctrl	159
Uruchamianie procedury za pomocą Wstążki	160
Uruchamianie procedur za pośrednictwem niestandardowego menu podręcznego	161
Wywoływanie procedury z poziomu innej procedury	161
Uruchamianie procedury poprzez kliknięcie obiektu	166
Wykonywanie procedury po wystąpieniu określonego zdarzenia	168
Uruchamianie procedury z poziomu okna Immediate	169
Przekazywanie argumentów procedurom	170
Metody obsługi błędów	173
Przechwytywanie błędów	174
Przykłady kodu źródłowego obsługującego błędy	175

Praktyczny przykład wykorzystujący procedury Sub	179
Cel	179
Wymagania projektowe	179
Co już wiesz	180
Podejście do zagadnienia	181
Wstępne rejestrowanie makr	181
Przygotowania	183
Tworzenie kodu źródłowego	184
Tworzenie procedury sortującej	185
Dodatkowe testy	190
Usuwanie problemów	191
Dostępność narzędzia	194
Ocena projektu	195
Rozdział 5. Tworzenie funkcji w języku VBA	197
Porównanie procedur Sub i Function	198
Dlaczego tworzymy funkcje niestandardowe?	198
Twoja pierwsza funkcja	199
Zastosowanie funkcji w arkuszu	200
Zastosowanie funkcji w procedurze języka VBA	201
Analiza funkcji niestandardowej	201
Procedury Function	204
Zasięg funkcji	205
Wywoływanie procedur Function	206
Argumenty funkcji	209
Przykłady funkcji	210
Funkcja bezargumentowa	210
Funkcje jednoargumentowe	213
Funkcje z dwoma argumentami	216
Funkcja pobierająca tablicę jako argument	217
Funkcje z argumentami opcjonalnymi	218
Funkcje zwracające tablicę VBA	220
Funkcje zwracające wartość błędu	223
Funkcje o nieokreślonej liczbie argumentów	224
Emulacja funkcji arkuszowej SUMA	226
Rozszerzone funkcje daty	229
Wykrywanie i usuwanie błędów w funkcjach	231
Okno dialogowe Wstawianie funkcji	233
Zastosowanie metody MacroOptions	233
Definiowanie kategorii funkcji	235
Dodawanie opisu funkcji	237
Zastosowanie dodatków do przechowywania funkcji niestandardowych	237
Korzystanie z Windows API	238
Przykłady zastosowania funkcji interfejsu API systemu Windows	239
Identyfikacja katalogu domowego systemu Windows	239
Wykrywanie wciśnięcia klawisza Shift	241
Dodatkowe informacje na temat funkcji interfejsu API	242

Rozdział 6. Obsługa zdarzeń	243
Co powinien wiedzieć o zdarzeniach	243
Sekwencje zdarzeń	245
Gdzie należy umieścić procedury obsługi zdarzeń?	245
Wyłączanie obsługi zdarzeń	247
Tworzenie kodu procedury obsługi zdarzeń	248
Procedury obsługi zdarzeń z argumentami	249
Zdarzenia poziomu skoroszytu	251
Zdarzenie Open	253
Zdarzenie Activate	254
Zdarzenie SheetActivate	254
Zdarzenie NewSheet	254
Zdarzenie BeforeSave	255
Zdarzenie Deactivate	255
Zdarzenie BeforePrint	256
Zdarzenie BeforeClose	257
Zdarzenia poziomu arkusza	259
Zdarzenie Change	260
Monitorowanie zmian w wybranym zakresie komórek	261
Zdarzenie SelectionChange	266
Zdarzenie BeforeDoubleClick	267
Zdarzenie BeforeRightClick	268
Zdarzenia dotyczące aplikacji	269
Włączenie obsługi zdarzeń poziomu aplikacji	271
Sprawdzanie, czy skoroszyt jest otwarty	271
Monitorowanie zdarzeń poziomu aplikacji	273
Zdarzenia niezwiązane z obiektami	273
Zdarzenie OnTime	274
Zdarzenie OnKey	275
Rozdział 7. Przykłady i techniki programowania w języku VBA	281
Nauka poprzez praktykę	281
Przetwarzanie zakresów	282
Kopiowanie zakresów	282
Przenoszenie zakresów	284
Kopiowanie zakresu o zmiennej wielkości	284
Zaznaczanie oraz identyfikacja różnego typu zakresów	286
Zmiana rozmiaru zakresu komórek	288
Wprowadzanie wartości do komórki	289
Wprowadzanie wartości do następnej pustej komórki	291
Wstrzymywanie działania makra w celu umożliwienia pobrania zakresu wyznaczonego przez użytkownika	292
Zliczanie zaznaczonych komórek	294
Określanie typu zaznaczonego zakresu	295
Wydajne przetwarzanie komórek zaznaczonego zakresu przy użyciu pętli	297
Usuwanie wszystkich pustych wierszy	300
Powielanie wierszy	301
Określanie, czy zakres zawiera się w innym zakresie	303
Określanie typu danych zawartych w komórce	303

Odczytywanie i zapisywanie zakresów	305
Lepsza metoda zapisywania danych do zakresu komórek	306
Przenoszenie zawartości tablic jednowymiarowych	309
Przenoszenie zawartości zakresu do tablicy typu Variant	309
Zaznaczanie komórek na podstawie wartości	310
Kopiowanie nieciągłego zakresu komórek	312
Przetwarzanie skoroszytów i arkuszy	314
Zapisywanie wszystkich skoroszytów	314
Zapisywanie i zamykanie wszystkich skoroszytów	315
Ukrywanie wszystkich komórek arkusza poza zaznaczonym zakresem	315
Tworzenie spisu treści zawierającego hiperłącza	317
Synchronizowanie arkuszy	318
Techniki programowania w języku VBA	319
Przełączanie wartości właściwości typu logicznego	319
Wyświetlanie daty i czasu	320
Wyświetlanie czasu w formie przyjaznej dla użytkownika	321
Pobieranie listy czcionek	323
Sortowanie tablicy	324
Przetwarzanie grupy plików	326
Ciekawe funkcje, których możesz użyć w swoich projektach	327
Funkcja FileExists	328
Funkcja FileNameOnly	328
Funkcja PathExists	329
Funkcja RangeNameExists	329
Funkcja SheetExists	330
Funkcja WorkbooksOpen	330
Pobieranie wartości z zamkniętego skoroszytu	331
Użyteczne, niestandardowe funkcje arkuszowe	333
Funkcje zwracające informacje o formatowaniu komórek	333
Wyświetlanie daty zapisania lub wydrukowania pliku	335
Obiekty nadrzędne	337
Zliczanie komórek, których wartości zawierają się pomiędzy dwoma wartościami	338
Wyznaczanie ostatniej niepustej komórki kolumny lub wiersza	339
Czy dany łańcuch tekstu jest zgodny ze wzorcem?	340
Wyznaczanie n-tego elementu łańcucha	342
Zamiana wartości na postać słowną	343
Funkcja wielofunkcyjna	344
Funkcja SHEETOFFSET	345
Zwracanie maksymalnej wartości ze wszystkich arkuszy	345
Zwracanie tablicy zawierającej unikatowe, losowo uporządkowane liczby całkowite	347
Porządkowanie zakresu w losowy sposób	348
Sortowanie zakresów	350
Wywołania funkcji interfejsu Windows API	351
Deklaracje API	351
Określanie skojarzeń plików	352
Pobieranie informacji dotyczących drukarki domyślnej	354
Pobieranie informacji o aktualnej rozdzielczości karty graficznej	355
Odczytywanie zawartości rejestru systemu Windows i zapisywanie w nim danych	356

Część II. Zaawansowane techniki programowania **359**

Rozdział 8. Tabele przestawne	361
Przykład prostej tabeli przestawnej	361
Tworzenie tabel przestawnych	362
Analiza zarejestrowanego kodu tworzącego tabelę przestawną	365
Optymalizacja wygenerowanego kodu tworzącego tabelę przestawną	365
Tworzenie złożonych tabel przestawnych	368
Kod tworzący tabelę przestawną	370
Jak działa złożona tabela przestawna?	371
Jednoczesne tworzenie wielu tabel przestawnych	373
Tworzenie odwróconych tabel przestawnych	376
Rozdział 9. Wykresy	379
Podstawowe wiadomości o wykresach	379
Lokalizacja wykresu	380
Rejestrator makr a wykresy	381
Model obiektu Chart	381
Tworzenie wykresów osadzonych na arkuszu danych	383
Tworzenie wykresu na arkuszu wykresu	384
Modyfikowanie wykresów	385
Wykorzystanie VBA do uaktywnienia wykresu	386
Przenoszenie wykresu	387
Wykorzystanie VBA do dezaktywacji wykresu	389
Sprawdzanie, czy wykres został uaktywniony	389
Usuwanie elementów z kolekcji ChartObjects lub Charts	390
Przetwarzanie wszystkich wykresów w pętli	391
Zmiana rozmiarów i wyrównywanie obiektów ChartObject	394
Tworzenie dużej liczby wykresów	395
Eksportowanie wykresów	398
Eksportowanie wszystkich obiektów graficznych	398
Zmiana danych prezentowanych na wykresie	400
Modyfikacja danych wykresu na podstawie aktywnej komórki	401
Zastosowanie języka VBA do identyfikacji zakresu danych prezentowanych na wykresie	403
Wykorzystanie VBA do wyświetlania dowolnych etykiet danych na wykresie	406
Wyświetlanie wykresu w oknie formularza UserForm	410
Zdarzenia związane z wykresami	412
Przykład wykorzystania zdarzeń związanych z wykresami	413
Obsługa zdarzeń dla wykresów osadzonych	416
Przykład zastosowania zdarzeń dla wykresów osadzonych	417
Jak ułatwić sobie pracę z wykresami przy użyciu VBA?	419
Drukowanie wykresów osadzonych na arkuszu	420
Tworzenie wykresów, które nie są połączone z danymi	420
Wykorzystanie zdarzenia MouseOver do wyświetlania tekstu	422
Przewijanie wykresów	425
Tworzenie wykresów przebiegu w czasie	427

Rozdział 10. Interakcje z innymi aplikacjami	431
Automatyzacja zadań w pakiecie Microsoft Office	431
Koncepcja wiązań	432
Przykład prostej automatyzacji	435
Sterowanie bazą danych Access z poziomu Excela	435
Uruchamianie zapytań bazy danych Access z poziomu Excela	436
Uruchamianie makr Accessa z poziomu Excela	437
Sterowanie edytorem Word z poziomu Excela	438
Przesyłanie danych z Excela do dokumentu Worda	438
Symulacja tworzenia korespondencji seryjnej z użyciem Worda	439
Sterowanie programem PowerPoint z poziomu Excela	442
Przesyłanie danych z Excela do prezentacji PowerPoint	442
Przesyłanie wszystkich wykresów z arkusza Excela do prezentacji PowerPoint	443
Zamiana skoroszytu na prezentację PowerPoint	445
Sterowanie programem Outlook z poziomu Excela	446
Wysyłanie aktywnego skoroszytu jako załącznika	446
Wysyłanie wybranego zakresu komórek jako załącznika wiadomości	447
Wysyłanie pojedynczego arkusza jako załącznika wiadomości	449
Wysyłanie wiadomości do wszystkich adresatów z listy kontaktów	450
Uruchamianie innych aplikacji z poziomu Excela	451
Zastosowanie funkcji Shell języka VBA	451
Zastosowanie funkcji ShellExecute interfejsu Windows API	453
Wykorzystanie instrukcji AppActivate	455
Uruchamianie okien dialogowych Panelu sterowania	456
Rozdział 11. Praca z danymi zewnętrznymi i plikami	457
Praca z danymi ze źródeł zewnętrznych	457
Ręczne tworzenie połączenia z zewnętrznymi źródłami danych	458
Ręczna modyfikacja połączeń z zewnętrznymi źródłami danych	461
Zastosowanie języka VBA do tworzenia dynamicznych połączeń danych	463
Przechodzenie w pętli przez wszystkie połączenia skoroszytu	465
Zastosowanie ADO i VBA do pobierania danych ze źródeł zewnętrznych	466
Ciąg połączenia	466
Deklarowanie zestawu rekordów	468
Odwołania do biblioteki obiektów ADO	469
Łączenie wszystkiego razem w kodzie procedury	470
Zastosowanie obiektów ADO w aktywnym skoroszytcie	472
Operacje z plikami tekstowymi	474
Otwieranie plików tekstowych	475
Odczytywanie plików tekstowych	476
Zapisywanie danych do plików tekstowych	476
Przydzielanie numeru pliku	476
Określanie lub ustawianie pozycji w pliku	477
Instrukcje pozwalające na odczytywanie i zapisywanie plików	478
Przykłady wykonywania operacji na plikach	478
Importowanie danych z pliku tekstowego	478
Eksportowanie zakresu do pliku tekstowego	479
Importowanie pliku tekstowego do zakresu	480
Logowanie wykorzystania Excela	481
Filtrowanie zawartości pliku tekstowego	482

Najczęściej wykonywane operacje na plikach	483
Zastosowanie poleceń języka VBA do wykonywania operacji na plikach	483
Zastosowanie obiektu FileSystemObject	488
Pakowanie i rozpakowywanie plików	491
Pakowanie plików do formatu ZIP	492
Rozpakowywanie plików ZIP	493

Część III. Praca z formularzami UserForm **495**

Rozdział 12. Tworzenie własnych okien dialogowych	497
Zanim rozpoczniesz tworzenie formularza UserForm	497
Okno wprowadzania danych	498
Funkcja InputBox języka VBA	498
Metoda Application.InputBox	500
Funkcja MsgBox języka VBA	504
Metoda GetOpenFilename programu Excel	509
Metoda GetSaveAsFilename programu Excel	513
Okno wybierania katalogu	514
Wyświetlanie wbudowanych okien dialogowych Excela	514
Wyświetlanie formularza danych	517
Wyświetlanie formularza wprowadzania danych	518
Wyświetlanie formularza wprowadzania danych za pomocą VBA	519
Rozdział 13. Wprowadzenie do formularzy UserForm	521
Jak Excel obsługuje niestandardowe okna dialogowe	521
Wstawianie nowego formularza UserForm	522
Dodawanie formantów do formularza UserForm	523
Formanty okna Toolbox	524
Formant CheckBox	524
Formant ComboBox	525
Formant CommandButton	525
Formant Frame	525
Formant Image	526
Formant Label	526
Formant ListBox	526
Formant MultiPage	526
Formant OptionButton	526
Formant RefEdit	527
Formant ScrollBar	527
Formant SpinButton	527
Formant TabStrip	527
Formant TextBox	527
Formant ToggleButton	528
Modyfikowanie formantów formularza UserForm	529
Modyfikowanie właściwości formantów	531
Zastosowanie okna Properties	531
Wspólne właściwości	533
Uwzględnienie wymagań użytkowników preferujących korzystanie z klawiatury	535

Wyświetlanie formularza UserForm	537
Zmiana położenia formularza na ekranie	538
Wyświetlanie niemodalnych okien formularzy UserForm	538
Wyświetlanie formularza UserForm na podstawie zmiennej	539
Ładowanie formularza UserForm	539
Procedury obsługi zdarzeń	539
Zamykanie formularza UserForm	540
Przykład tworzenia formularza UserForm	542
Tworzenie formularza UserForm	542
Tworzenie kodu procedury wyświetlającej okno dialogowe	545
Testowanie okna dialogowego	545
Dodawanie procedur obsługi zdarzeń	547
Zakończenie tworzenia okna dialogowego	548
Zdarzenia powiązane z formularzem UserForm	549
Zdobywanie informacji na temat zdarzeń	549
Zdarzenia formularza UserForm	550
Zdarzenia związane z formantem SpinButton	551
Współpraca formantu SpinButton z formantem TextBox	553
Odwoływanie się do formantów formularza UserForm	556
Dostosowywanie okna Toolbox do własnych wymagań	558
Dodawanie nowych kart	558
Dostosowywanie lub łączenie formantów	559
Dodawanie nowych formantów ActiveX	560
Tworzenie szablonów formularzy UserForm	561
Lista kontrolna tworzenia i testowania formularzy UserForm	563
Rozdział 14. Przykłady formularzy UserForm	565
Tworzenie formularza UserForm pełniącego funkcję menu	566
Zastosowanie przycisków CommandButton w formularzach UserForm	566
Zastosowanie formantów ListBox w formularzach UserForm	567
Zaznaczanie zakresów przy użyciu formularza UserForm	568
Tworzenie okna powitalnego	570
Wyłączanie przycisku Zamknij formularza UserForm	573
Zmiana wielkości formularza UserForm	574
Powiększanie i przewijanie arkusza przy użyciu formularza UserForm	575
Zastosowania formantu ListBox	577
Tworzenie listy elementów formantu ListBox	578
Identyfikowanie zaznaczonego elementu listy formantu ListBox	584
Identyfikowanie wielu zaznaczonych elementów listy formantu ListBox	585
Wiele list w jednym formancie ListBox	586
Przenoszenie elementów listy formantu ListBox	587
Zmiana kolejności elementów listy formantu ListBox	589
Wielokolumnowe formanty ListBox	591
Zastosowanie formantu ListBox do wybierania wierszy arkusza	593
Uaktywnianie arkusza za pomocą formantu ListBox	596
Filtrowanie zawartości listy za pomocą pola tekstowego	598
Zastosowanie formantu MultiPage na formularzach UserForm	601
Korzystanie z formantów zewnętrznych	602
Animowanie etykiet	605

Rozdział 15. Zaawansowane techniki korzystania z formularzy UserForm	609
Niemodalne okna dialogowe	610
Wyświetlanie wskaźnika postępu zadania	614
Tworzenie samodzielnego wskaźnika postępu zadania	615
Wyświetlanie wskaźnika postępu zintegrowanego z formularzem UserForm	619
Tworzenie innych, niegraficznych wskaźników postępu	623
Tworzenie kreatorów	626
Konfigurowanie formantu MultiPage w celu utworzenia kreatora	628
Dodawanie przycisków do formularza UserForm kreatora	628
Programowanie przycisków kreatora	629
Zależności programowe w kreatorach	631
Wykonywanie zadań za pomocą kreatorów	632
Emulacja funkcji MsgBox	633
Emulacja funkcji MsgBox: kod funkcji MyMsgBox	635
Jak działa funkcja MyMsgBox	636
Wykorzystanie funkcji MyMsgBox do emulacji funkcji MsgBox	638
Formularz UserForm z formantami, których położenie można zmieniać	638
Formularz UserForm bez paska tytułowego	640
Symulacja paska narzędzi za pomocą formularza UserForm	642
Emulowanie panelu zadań za pomocą formularza UserForm	644
Formularze UserForm z możliwością zmiany rozmiaru	646
Obsługa wielu przycisków formularza UserForm za pomocą jednej procedury obsługi zdarzeń	651
Wybór koloru za pomocą formularza UserForm	654
Wyświetlanie wykresów na formularzach UserForm	656
Zapisywanie wykresu w postaci pliku GIF	657
Modyfikacja właściwości Picture formantu Image	657
Tworzenie półprzezroczystych formularzy UserForm	657
Układanka na formularzu UserForm	660
Poker na formularzu UserForm	661

Część IV. Tworzenie aplikacji **663**

Rozdział 16. Tworzenie i wykorzystanie dodatków	665
Czym są dodatki?	665
Porównanie dodatku ze standardowym skoroszytem	666
Po co tworzy się dodatki?	667
Menedżer dodatków Excela	669
Tworzenie dodatków	671
Przykład tworzenia dodatku	672
Tworzenie opisu dla dodatku	674
Tworzenie dodatku	674
Instalowanie dodatku	676
Testowanie dodatków	677
Dystrybucja dodatków	677
Modyfikowanie dodatku	677

Porównanie plików XLAM i XLSM	679
Pliki XLAM — przynależność do kolekcji z poziomu VBA	679
Widoczność plików XLSM i XLAM	680
Arkusze i wykresy w plikach XLSM i XLAM	680
Dostęp do procedur VBA w dodatku	681
Przetwarzanie dodatków za pomocą kodu VBA	685
Dodawanie nowych elementów do kolekcji AddIns	685
Usuwanie elementów z kolekcji AddIns	687
Właściwości obiektu AddIn	687
Korzystanie z dodatku jak ze skoroszytu	691
Zdarzenia związane z obiektami AddIn	691
Optymalizacja wydajności dodatków	692
Problemy z dodatkami	693
Upewnij się, że dodatek został zainstalowany	693
Odwoływanie się do innych plików z poziomu dodatku	695
Wykrywanie właściwej wersji Excela dla dodatku	696
Rozdział 17. Praca ze Wstążką	697
Wprowadzenie do pracy ze Wstążką	697
Dostosowywanie Wstążki do własnych potrzeb	700
Dodawanie nowych przycisków do Wstążki	700
Dodawanie przycisków do paska narzędzi Szybki dostęp	703
Ograniczenia w dostosowywaniu Wstążki	703
Modyfikowanie Wstążki za pomocą kodu RibbonX	705
Dodawanie przycisków do istniejącej karty	705
Dodawanie pola wyboru do istniejącej karty	712
Demo formantów Wstążki	716
Przykład użycia formantu DynamicMenu	724
Więcej wskazówek dotyczących modyfikacji Wstążki	726
VBA i Wstążka	728
Dostęp do poleceń Wstążki	729
Praca ze Wstążką	730
Aktywowanie karty	732
Tworzenie pasków narzędzi w starym stylu	732
Ograniczenia funkcjonalności tradycyjnych pasków narzędzi w Excelu 2007 i nowszych wersjach	733
Kod tworzący pasek narzędzi	733
Rozdział 18. Praca z menu podręcznym	737
Obiekt CommandBar	737
Rodzaje obiektów CommandBar	738
Wyświetlanie menu podręcznych	738
Odwołania do elementów kolekcji CommandBars	740
Odwołania do formantów obiektu CommandBar	740
Właściwości formantów obiektu CommandBar	742
Wyświetlanie wszystkich elementów menu podręcznego	743
Wykorzystanie VBA do dostosowywania menu podręcznego	746
Menu podręczne w jednodokumentowym interfejsie Excela	746
Resetowanie menu podręcznego	748
Wyłączanie menu podręcznego	749

Wyłączanie wybranych elementów menu podręcznego	750
Dodawanie nowego elementu do menu podręcznego Cell	750
Dodawanie nowego podmenu do menu podręcznego	753
Ograniczanie zasięgu modyfikacji menu podręcznego do jednego skoroztytu	755
Menu podręczne i zdarzenia	756
Automatyczne tworzenie i usuwanie menu podręcznego	756
Wyłączanie lub ukrywanie elementów menu podręcznego	757
Tworzenie kontekstowych menu podręcznych	758
Rozdział 19. Tworzenie systemów pomocy w aplikacjach	761
Systemy pomocy w aplikacjach Excela	761
Systemy pomocy wykorzystujące komponenty Excela	764
Wykorzystanie komentarzy do tworzenia systemów pomocy	764
Wykorzystanie pól tekstowych do wyświetlania pomocy	766
Wykorzystanie arkusza do wyświetlania tekstu pomocy	767
Wyświetlanie pomocy w oknie formularza UserForm	768
Wyświetlanie pomocy w oknie przeglądarki sieciowej	772
Zastosowanie plików w formacie HTML	772
Zastosowanie plików w formacie MHTML	773
Wykorzystanie systemu HTML Help	775
Wykorzystanie metody Help do wyświetlania pomocy w formacie HTML Help	778
Łączenie pliku pomocy z aplikacją	779
Przypisanie tematów pomocy do funkcji VBA	779
Rozdział 20. Moduły klas	783
Czym jest moduł klasy?	783
Wbudowane moduły klas	784
Niestandardowe moduły klas	785
Tworzymy klasę NumLock	786
Wstawianie modułu klasy	786
Dodawanie kodu VBA do modułu klasy	787
Zastosowanie klasy NumLock	789
Programowanie właściwości, metod i zdarzeń	790
Programowanie właściwości obiektów	790
Programowanie metod obiektów	792
Zdarzenia modułu klasy	792
Zdarzenia obiektu QueryTable	793
Tworzenie klas przechowujących inne klasy	797
Tworzenie klas CSalesRep oraz CSalesReps	797
Tworzenie klas CInvoice oraz CInvoices	799
Wypełnianie klasy nadrzędnej obiektami	801
Obliczanie prowizji	802
Rozdział 21. Problem kompatybilności aplikacji	805
Co to jest kompatybilność?	805
Rodzaje problemów ze zgodnością	806
Unikaj używania nowych funkcji i mechanizmów	808
Czy aplikacja będzie działać na komputerach Macintosh?	810
Praca z 64-bitową wersją Excela	811

Tworzenie aplikacji dla wielu wersji narodowych	813
Aplikacje obsługujące wiele języków	813
Obsługa języka w kodzie VBA	816
Wykorzystanie właściwości lokalnych	816
Identyfikacja ustawień systemu	817
Ustawienia daty i godziny	819

Dodatki **821**

Dodatek A. Instrukcje i funkcje VBA	823
Skorowidz	833

Wprowadzenie do języka VBA

W TYM ROZDZIALE:

- ◆ Jak korzystać z rejestratora makr programu Excel
- ◆ Jak korzystać z edytora VBE?
- ◆ Model obiektowy Excela
- ◆ Przykłady zastosowań obiektów Range
- ◆ Gdzie szukać dodatkowych informacji

Rejestrator makr Excela

Makro to w zasadzie nic innego jak kod programu napisany w języku Visual Basic for Applications (VBA), który możesz uruchamiać w celu wykonania wielu różnych operacji. Makra w Excelu mogą być pisane bezpośrednio w edytorze lub rejestrowane za pomocą rejestratora makr.

Terminologia używana w programowaniu Excela może być nieco myląca. Zarejestrowane makro z technicznego punktu widzenia nie różni się niczym od procedury VBA, którą możesz napisać ręcznie. Określenia „makro” i „procedura VBA” są bardzo często używane wymiennie. Wielu użytkowników Excela wszystkie procedury języka VBA nazywa makrami. Z drugiej jednak strony, bardzo często użytkownicy Excela, mówiąc o makrach, mają na myśli procedury VBA zarejestrowane przy użyciu rejestratora makr.

Rejestrowanie makra przypomina nieco zapisywanie nowego numeru w pamięci telefonu komórkowego. Aby to zrobić, najpierw musisz ręcznie wybrać i zapisać numer w telefonie, dzięki czemu możesz później szybko się z nim połączyć za pomocą naciśnięcia jednego klawisza. Podobnie ma się sprawa z Excelem, gdzie możesz rejestrować różne operacje podczas ich wykonywania. Kiedy Ty jesteś zajęty działaniem, Excel przetwarza wszystkie wykonywane przez Ciebie operacje, naciśnięcia klawiszy i kliknięcia myszy na odpowiadające im polecenia VBA. Po zarejestrowaniu makra możesz w dowolnym momencie uruchomić i odtworzyć wykonywane wcześniej działania.

Zdecydowanie najlepszym sposobem na szybkie zapoznanie się ze specyfiką i składnią poleceń języka VBA jest uruchomienie rejestratora makr, wykonanie kilku operacji w Excelu, a następnie uważne przeanalizowanie kodu utworzonego przez rejestrator.

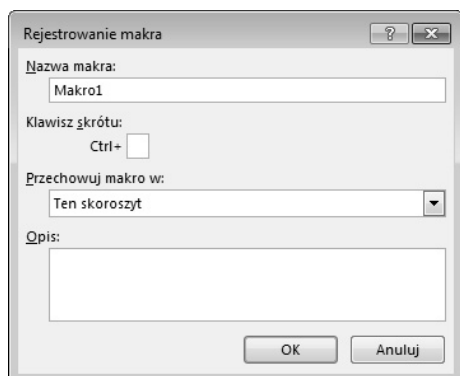
W tym podrozdziale będziesz pracował z różnymi makrami i dowiesz się, jak używać rejestratora makr do poznawania tajników programowania w języku VBA.

Tworzenie pierwszego makra

Aby móc rozpocząć rejestrowanie swojego pierwszego makra, musisz najpierw odszukać przycisk uruchamiający rejestrator makr, który znajduje się na karcie *Deweloper*. Niestety domyślnie po zainstalowaniu Excela karta *Deweloper* może nie być widoczna. Jeżeli chcesz pracować z makrami VBA, będziesz musiał ją samodzielnie włączyć. Aby to zrobić, powinieneś wykonać następujące operacje:

1. Przejdź na kartę *Plik* i naciśnij przycisk *Opcje*.
2. Na ekranie pojawi się okno dialogowe opcji Excela. Wybierz opcję *Dostosowywanie Wstążki*.
3. Na liście dostępnych kart, znajdującej się w prawej części okna, odszukaj i zaznacz kartę *Deweloper*.
4. Naciśnij przycisk *OK*, aby powrócić do głównego okna Excela.

Od tej chwili na Wstążce Excela karta *Deweloper* powinna być widoczna. Aby teraz uruchomić rejestrator makr, przejdź na kartę *Deweloper* i naciśnij przycisk *Zarejestruj makro*, znajdujący się w grupie opcji *Kod*. Na ekranie pojawi się okno dialogowe *Rejestrowanie makra*, pokazane na rysunku 2.1.



Rysunek 2.1. Okno dialogowe Rejestrowanie makra

W oknie dialogowym *Rejestrowanie makra* możemy wyróżnić cztery pola:

- **Nazwa makra.** Nazwa tego pola nie wymaga chyba specjalnych wyjaśnień. Excel przypisuje do nowego makra domyślną nazwę, składającą się ze słowa *Makro* i kolejnego numeru, ale oczywiście możesz (i w większości przypadków powinieneś) zmienić tę nazwę na coś bardziej opisowego. Na przykład makro, które będzie zmieniało wygląd tabeli, możesz nazwać `FormatowanieTabeli`.
- **Klawisz skrót.** Aby dane makro zostało uruchomione, musi zająć określone zdarzenie. Takim zdarzeniem może być naciśnięcie danej kombinacji klawiszy, naciśnięcie przycisku polecenia czy otwarcie lub zamknięcie skoroszytu. Jeżeli przypiszesz do makra określoną kombinację klawiszy, to makro zostanie uruchomione za każdym razem, kiedy użytkownik naciśnie taką kombinację. Pole *Klawisz skrót* jest opcjonalne.
- **Przechowuj makro w.** Domyślną wartością tego pola jest *Ten skoroszyt*. Wybranie tej opcji oznacza po prostu, że nasze makro zostanie zapisane w aktualnie otwartym skoroszycie. Dzięki temu następnym razem, kiedy otworzysz ten skoroszyt, będziesz mógł uruchomić zapisane w nim makro. Co więcej, jeżeli udostępnisz ten skoroszyt innemu użytkownikowi, to on również będzie mógł takie makro uruchomić (oczywiście przy założeniu, że będzie miał odpowiednio ustawione opcje bezpieczeństwa makr — więcej informacji na ten temat znajdziesz w dalszej części tego rozdziału).
- **Opis.** Jest to pole opcjonalne, w którym możesz zamieścić opis przeznaczenia i sposobu działania makra. Takie pole jest bardzo przydatne zwłaszcza w sytuacji, kiedy w skoroszycie zapisanych jest bardzo wiele makr lub chcesz przekazać innym użytkownikom dodatkowe informacje na temat danego makra.

Aby utworzyć proste makro, które będzie wpisywało Twoje imię do wybranej komórki arkusza, otwórz okno rejestratora makr i wykonaj polecenia opisane poniżej:

1. W polu *Nazwa makra* wpisz nową nazwę makra, zastępując domyślną nazwę *Makro1*. W naszym przykładzie wybraliśmy dla tego makra nazwę `MojeDane`.
2. Przypisz do makra kombinację klawiszy `Ctrl+Shift+N`. Aby to zrobić, w polu *Klawisz skrót* wpisz wielką literę *N*.
3. Naciśnij przycisk *OK*, aby zamknąć okno dialogowe *Rejestrowanie makra* i rozpocząć rejestrowanie makra.
4. Kliknij dowolnie wybraną komórkę arkusza, wpisz w niej swoje imię i nazwisko, a następnie naciśnij klawisz *Enter*.
5. Przejdź na kartę *Deweloper* i naciśnij przycisk *Zatrzymaj rejestrowanie*, znajdujący się w grupie poleceń *Kod* (zamiast tego możesz po prostu nacisnąć przycisk *Zatrzymaj rejestrowanie*, znajdujący się na pasku stanu w dolnej części okna programu Excel).

PRZEGLĄDANIE I SPRAWDZANIE UTWORZONEGO MAKRA

Nasze makro zostało zarejestrowane w nowym module kodu o nazwie `Modul1`. Aby wyświetlić kod VBA znajdujący się w tym module, musisz uruchomić edytor VBE. Możesz to zrobić na jeden z dwóch sposobów:

- Naciśnij kombinację klawiszy *Alt+F11*.
- Przejdź na kartę *Developer* i naciśnij przycisk *Visual Basic* znajdujący się w grupie opcji *Kod*.

W oknie edytora VBE znajdziesz panel o nazwie *Project* (projekt), w którym wyświetlana jest lista wszystkich aktualnie otwartych skoroszytów i dodatków. Każdy skoroszyt i dodatek stanowi osobny *projekt*, którego węzły składowe możesz dowolnie zwijać i rozwijać. Kod naszego zarejestrowanego przed chwilą makra znajduje się w module kodu o nazwie `Modul1` aktywnego skoroszytu. Kiedy dwukrotnie klikniesz lewym przyciskiem myszy ikonę tego modułu, kod makra pojawi się w oknie kodu w prawej części ekranu.



UWAGA

Jeżeli okno eksploratora projektów nie jest widoczne, możesz je otworzyć, wybierając z menu głównego edytora VBE polecenie *View/Project Explorer* (widok/eksplorator projektów). Ten sam efekt możesz uzyskać, naciskając po prostu kombinację klawiszy *Ctrl+R*.

Kod naszego zarejestrowanego makra powinien wyglądać mniej więcej tak:

```
Sub MojeDane()  
'  
' MojeDane Makro  
'  
' Klawisz skrótów: Ctrl+Shift+N  
'  
    ActiveCell.FormulaR1C1 = "Maksymilian Kowalski"  
End Sub
```

Nasze makro zostało zarejestrowane w postaci procedury `Sub` o nazwie `MojeDane`. Polecenia kodu VBA w ciele procedury informują Excela o tym, co powinien zrobić po uruchomieniu makra.

Zauważ, że na początku procedury Excel wstawił kilka wierszy komentarza, zawierających informacje, które wprowadziłeś w oknie *Rejestrowanie makra*. Wiersze komentarza (które rozpoczynają się od znaku apostrofu) nie są niezbędne do poprawnego działania makra i usunięcie ich nie będzie miało żadnego wpływu na jego działanie. Zatem jeżeli zignorujemy wszystkie komentarze, okaże się, że nasza procedura tak naprawdę składa się tylko z jednego wiersza kodu:

```
ActiveCell.FormulaR1C1 = "Maksymilian Kowalski"
```

Wykonanie powyższego polecenia powoduje wpisanie ciągu znaków *Maksymilian Kowalski* do aktywnej komórki arkusza.

TESTOWANIE DZIAŁANIA MAKRA

Przed rozpoczęciem rejestrowania przypisałeś do makra kombinację klawiszy *Ctrl+Shift+N*. Aby przetestować działanie makra, powinieneś najpierw powrócić do głównego okna Excela. Możesz to zrobić na jeden z dwóch sposobów:

- Naciśnij kombinację klawiszy *Alt+F11*.
- Naciśnij przycisk *View Microsoft Excel* (wyświetl okno programu Excel), znajdujący się na standardowym pasku narzędzi edytora VBE.

Po przejściu do okna Excela aktywuj skoroszyt (może to być skoroszyt zawierający nasz moduł kodu VBA, ale równie dobrze może to być dowolny inny skoroszyt). Kliknij wybraną komórkę arkusza i naciśnij kombinację klawiszy *Ctrl+Shift+N*. Excel uruchomi makro i wpisze w wybranej komórce ciąg znaków *Maksymilian Kowalski*.



UWAGA

Jak pamiętasz, przed rozpoczęciem rejestrowania makra zaznaczyłeś wybraną komórkę arkusza. Jest to bardzo ważny krok. Jeżeli zaznaczysz żądaną komórkę dopiero po uruchomieniu rejestratora, to komórka, którą wybierzesz, zostanie zapisana w kodzie generowanego makra. W takiej sytuacji makro zawsze będzie formatowało tylko tę jedną, zaznaczoną podczas rejestrowania komórkę i dlatego nie będzie makrem uniwersalnym.

EDYTOWANIE MAKRA

Po zarejestrowaniu makra możesz dowolnie modyfikować jego kod. Na przykład założmy, że chcesz, aby imię i nazwisko wpisywane do komórki były wyróżnione pogrubioną czcionką. W zasadzie mógłbyś usunąć stare makro i zarejestrować je na nowo, ale żądana zmiana sposobu działania makra jest bardzo prosta, dlatego znacznie lepszym rozwiązaniem będzie po prostu bezpośrednie dokonanie odpowiedniej modyfikacji kodu makra. Naciśnij kombinację klawiszy *Alt+F11*, aby aktywować edytor VBE. Następnie przejdź do modułu kodu `Module1` i wstaw polecenie **ActiveCell.Font.Bold = True**, tak jak to zostało pokazane w przykładzie poniżej:

```
Sub MojeDane()
'
' MojeDane Makro
'
' Klawisz skrót: Ctrl+Shift+N
'
    ActiveCell.Font.Bold = True
    ActiveCell.FormulaR1C1 = "Maksymilian Kowalski"
End Sub
```

Przetestuj nowe makro i sprawdź, czy działa zgodnie z Twoimi oczekiwaniami.

Porównanie rejestrowania makr z odwołaniami względnymi i bezwzględnymi

Poznałeś już podstawowe zasady posługiwania się interfejsem rejestratora makr, zatem nadszedł czas, aby zacząć rejestrowanie znacznie bardziej złożonych makr. Zanim jednak zaczniemy, musisz wiedzieć, że Excel posiada dwa tryby rejestrowania makr — rejestrowanie z odwołaniami bezwzględnymi i rejestrowanie z odwołaniami względnymi.

REJESTROWANIE MAKR W TRYBIE ODWOŁAŃ BEZWZGLĘDNYCH

Domyślnie Excel rejestruje makra z odwołaniami bezwzględnymi. Jak zapewne wiesz, odwołania bezwzględne są bardzo często używane w formułach arkuszy. Jeżeli w formule umieścimy bezwzględne odwołanie do danej komórki, to takie odwołanie nie zostanie automatycznie dostosowane po skopiowaniu formuły do nowej lokalizacji.

Najlepszym sposobem, aby zrozumieć, jak działają odwołania bezwzględne, jest ich wypróbowanie w praktyce. Otwórz skoroszyt z przykładowym zestawem danych, a następnie zarejestruj makro, które będzie zliczało liczbę wierszy (patrz rysunek 2.2).

	A	B	C	D	E	F	G	H	I	J
1		Region	Sklep	Kod oddziału			Region	Sklep	Kod oddziału	
2		Północ	Gdańsk	601419			Południe	Chorzów	173901	
3		Północ	Gdańsk	701407			Południe	Chorzów	301301	
4		Północ	Gdańsk	802202			Południe	Chorzów	302301	
5		Północ	Kołoźrzeg	910181			Południe	Chorzów	601306	
6		Północ	Kołoźrzeg	920681			Południe	Gliwice	202600	
7		Północ	Ustka	101419			Południe	Gliwice	490260	
8		Północ	Ustka	501405			Południe	Gliwice	490360	
9		Północ	Ustka	503405			Południe	Gliwice	490460	
10		Północ	Ustka	590140			Południe	Katowice	301316	
11		Północ	Szczecin	801211			Południe	Katowice	701309	
12		Północ	Szczecin	802211			Południe	Katowice	702309	
13		Północ	Szczecin	804211			Południe	Wrocław	601310	
14		Północ	Szczecin	805211			Południe	Wrocław	602310	
15		Północ	Szczecin	806211			Południe	Wrocław	801607	
16										

Rysunek 2.2. Przykładowy zestaw danych składa się z dwóch tabel

FTP NA FTP

Skoroszyt z tym przykładem (*Przykładowe dane.xlsm*) znajdziesz na serwerze FTP wydawnictwa Helion (<ftp://ftp.helion.pl/przyklady/e26pww.zip>).

Aby zarejestrować makro, wykonaj polecenia opisane poniżej:

1. Przed rozpoczęciem rejestrowania makra upewnij się, że zaznaczyłeś komórkę A1.
2. Przejdź na kartę *Developer* i wybierz polecenie *Zarejestruj makro*.

3. W polu *Nazwa makra* wpisz **DodajPodsumowanie**.
4. W polu *Przechowuj makro w* wybierz opcję *Ten skoroszyt*.
5. Naciśnij przycisk *OK*, aby rozpocząć rejestrowanie makra.
W tym momencie Excel rozpocznie rejestrowanie wykonywanych przez Ciebie operacji.
6. Zaznacz komórkę o adresie A16 i wpisz w niej etykietę **RAZEM**.
7. Zaznacz pierwszą wolną komórkę w kolumnie D (powinna to być komórka o adresie D16) i wpisz w niej następującą formułę: `=ILE.NIEPUSTYCH(D2:D15)`, która policzy liczbę oddziałów i umieści ją na dole kolumny D. Funkcja `ILE.NIEPUSTYCH` została tutaj użyta, ponieważ pozwala na uwzględnienie wszystkich kodów oddziałów (liczby), które ewentualnie zostały zapisane jako tekst.
8. Przejdź na kartę *Deweloper* i zatrzymaj rejestrowanie makra.

Gotowy arkusz powinien wyglądać tak, jak to zostało pokazane na rysunku 2.3.

	A	B	C	D	E	F	G	H	I	J
1		Region	Sklep	Kod oddziału			Region	Sklep	Kod oddziału	
2		Północ	Gdańsk	601419			Południe	Chorzów	173901	
3		Północ	Gdańsk	701407			Południe	Chorzów	301301	
4		Północ	Gdańsk	802202			Południe	Chorzów	302301	
5		Północ	Kołobrzeg	910181			Południe	Chorzów	601306	
6		Północ	Kołobrzeg	920681			Południe	Gliwice	202600	
7		Północ	Ustka	101419			Południe	Gliwice	490260	
8		Północ	Ustka	501405			Południe	Gliwice	490360	
9		Północ	Ustka	503405			Południe	Gliwice	490460	
10		Północ	Ustka	590140			Południe	Katowice	301316	
11		Północ	Szczecin	801211			Południe	Katowice	701309	
12		Północ	Szczecin	802211			Południe	Katowice	702309	
13		Północ	Szczecin	804211			Południe	Wrocław	601310	
14		Północ	Szczecin	805211			Południe	Wrocław	602310	
15		Północ	Szczecin	806211			Południe	Wrocław	801607	
16		RAZEM		14						
17										

Rysunek 2.3. Wygląd arkusza po zakończeniu rejestrowania makra

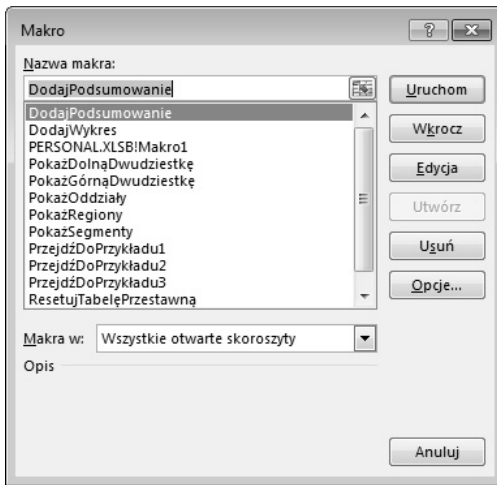
Aby teraz zobaczyć zarejestrowane makro w działaniu, usuń wiersz podsumowania, który przed chwilą dodałeś, i wykonaj makro. Aby uruchomić makro, wykonaj następujące polecenia:

1. Przejdź na kartę *Deweloper* i wybierz polecenie *Makra*.
2. Na liście dostępnych makr odszukaj i zaznacz zarejestrowane przed chwilą makro *DodajPodsumowanie*.
3. Naciśnij przycisk *Uruchom*.

Jeżeli wszystko poszło zgodnie z oczekiwaniami, makro zostanie wykonane bez przeszkód i na dole tabeli pojawi się odpowiednie podsumowanie. A teraz twardy orzech do zgryzienia: niezależnie od tego, jak wiele razy będziesz próbował, nie będziesz w stanie zmusić tego makra

do utworzenia podsumowania dla drugiej tabeli. Dlaczego? Ponieważ zostało zarejestrowane w trybie odwołań bezwzględnych.

Aby zrozumieć, co to znaczy, musisz przeanalizować kod VBA zarejestrowanego makra. Aby to zrobić, przejdź na kartę *Deweloper* i wybierz polecenie *Makra*. Na ekranie pojawi się okno dialogowe *Makra*, przedstawione na rysunku 2.4, w którym domyślnie wyświetlona jest lista makr dostępnych we wszystkich aktualnie otwartych skoroszytach Excela (wraz ze wszystkimi zainstalowanymi dodatkami). Aby ograniczyć liczbę wyświetlanych makr, możesz zmienić ustawienie listy rozwijanej *Makra w* na opcję *Ten skoroszyt*.



Rysunek 2.4. Okno dialogowe Makra

Zaznacz makro *DodajPodsumowanie* i naciśnij przycisk *Edycja*, co spowoduje uruchomienie edytora VBE i wyświetlenie kodu zarejestrowanego makra.

```
Sub DodajPodsumowanie()  
    Range("A16").Select  
    ActiveCell.FormulaR1C1 = "RAZEM"  
    Range("D16").Select  
    ActiveCell.FormulaR1C1 = "=COUNTA(R[-14]C:R[-1]C)"  
End Sub
```

Zwróć szczególną uwagę na drugi i czwarty wiersz kodu makra. Kiedy zaznaczałeś komórki o adresach A16 i D16, Excel zarejestrował dokładnie takie adresy komórek. Ponieważ makro było rejestrowane w trybie odwołań bezwzględnych, Excel zinterpretował odwołania do komórek jako bezwzględne. Innymi słowy, jeżeli w takim trybie zaznaczyłeś komórkę A16, to po uruchomieniu makra Excel zaznaczy komórkę A16. W kolejnym podrozdziale przekonasz się, jak wygląda takie makro zarejestrowane w trybie odwołań względnych.

REJESTROWANIE MAKR W TRYBIE ODWOŁAŃ WZGLĘDNYCH

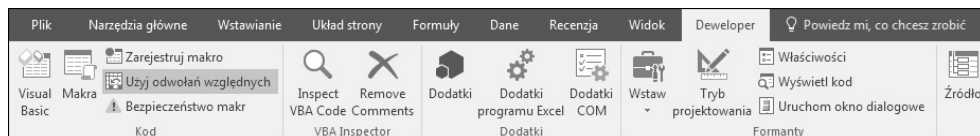
W terminologii makr Excela odwołanie względne oznacza odwołanie względem aktualnie aktywnej komórki arkusza. Z tego powodu powinieneś zawsze zachować pewną ostrożność, wybierając komórkę aktywną, zarówno podczas rejestrowania makra, jak i później, podczas jego uruchamiania.

Otwórz skoroszyt z przykładowym zestawem danych, z którego korzystaliśmy w poprzednim podrozdziale (plik skoroszytu znajdziesz na serwerze FTP wydawnictwa Helion — <ftp://ftp.helion.pl/przyklady/e26pvw.zip>), a następnie zarejestruj makro w trybie odwołań względnych. Aby to zrobić, wykonaj polecenia opisane poniżej:

1. Przejdź na kartę *Deweloper* i włącz opcję *Użyj odwołań względnych*, tak jak zostało to pokazane na rysunku 2.5.
2. Przed rozpoczęciem rejestrowania makra upewnij się, że zaznaczyłeś komórkę A1.
3. Przejdź na kartę *Deweloper* i wybierz polecenie *Zarejestruj makro*.
4. W polu *Nazwa makra* wpisz **DodajPodsumowanieWzględne**.
5. W polu *Przechowuj makro w* wybierz opcję *Ten skoroszyt*.
6. Naciśnij przycisk *OK*, aby zacząć rejestrowanie makra.

W tym momencie Excel rozpocznie rejestrowanie wykonywanych przez Ciebie operacji.

7. Zaznacz komórkę o adresie A16 i wpisz w niej etykietę **RAZEM**.
8. Zaznacz pierwszą wolną komórkę w kolumnie D (powinna to być komórka o adresie D16) i wpisz w niej następującą formułę: **=ILE.NIEPUSTYCH(D2:D15)**.
9. Przejdź na kartę *Deweloper* i zatrzymaj rejestrowanie makra.



Rysunek 2.5. Rejestrowanie makra w trybie odwołań względnych

Po zakończeniu będziemy mieć zarejestrowane dwa makra. Teraz uważnie przeanalizuj kod nowo utworzonego makra.

Aby to zrobić, przejdź na kartę *Deweloper* i wybierz polecenie *Makra*. Na ekranie pojawi się okno dialogowe *Makra*. Odszukaj i zaznacz makro **DodajPodsumowanieWzględne**, a następnie naciśnij przycisk *Edycja*.

Naciśnięcie tego przycisku spowoduje uruchomienie edytora VBE i wyświetlenie kodu zarejestrowanego makra. Tym razem kod naszego makra wygląda następująco:

```
Sub DodajPodsumowanieWzględne()
    ActiveCell.Offset(15, 0).Range("A1").Select
    ActiveCell.FormulaR1C1 = "RAZEM"
    ActiveCell.Offset(0, 3).Range("A1").Select
    ActiveCell.FormulaR1C1 = "=COUNTA(R[-14]C:R[-1]C)"
End Sub
```

Zauważ, że w kodzie procedury nie ma bezpośrednich odwołań do komórek, z wyjątkiem początkowej komórki A1. Przyjrzymy się teraz samej procedurze i wyjaśnimy, co znaczą poszczególne fragmenty jej kodu.

Zwróć uwagę, że w wierszu 2. Excel używa właściwości `Offset` aktywnej komórki, która wskazuje konieczność przeniesienia kursora o określoną liczbę komórek w górę lub w dół oraz o określoną liczbę komórek w lewo lub w prawo.

W naszym przypadku właściwość `Offset` nakazuje Excelowi przenieść kursor 15 wierszy w dół i 0 kolumn w prawo od aktywnej komórki (którą jest w tym przypadku komórka A1). Excel nie musi bezpośrednio zaznaczać komórki docelowej, tak jak to miało miejsce w przypadku makra z odwołaniami bezwzględnyymi.

Aby zobaczyć nasze nowe makro w działaniu, usuń wiersz podsumowania i wykonaj następujące polecenia:

1. Zaznacz komórkę A1.
2. Przejdź na kartę *Deweloper* i wybierz polecenie *Makra*.
3. Na liście wyszukaj makro *DodajPodsumowanieWzględne*.
4. Naciśnij przycisk *Uruchom*.
5. Zaznacz komórkę F1.
6. Ponownie przejdź na kartę *Deweloper* i wybierz polecenie *Makra*.
7. Na liście wyszukaj makro *DodajPodsumowanieWzględne*.
8. Naciśnij przycisk *Uruchom*.

Zauważ, że tym razem nasze nowe makro, w przeciwieństwie do makra z poprzedniego przykładu, działa poprawnie na obydwu zestawach danych. Ponieważ makro wykonuje operacje na komórkach wyznaczonych w odniesieniu do aktualnie aktywnej komórki, wszystkie etykiety i podsumowania są tworzone poprawnie.

Aby nasze makro działało poprawnie, musisz się po prostu upewnić, że:

- Przed uruchomieniem makra zaznaczyłeś odpowiednią komórkę bazową.
- Zestawy danych mają taką samą liczbę wierszy i kolumn jak zestaw danych użyty podczas rejestrowania makra.

Mamy nadzieję, że po uważnym przeanalizowaniu obu przytoczonych przykładów będziesz już wiedział, na czym polega różnica między rejestrowaniem makra w trybie odwołań bezwzględnych a rejestrowaniem makra w trybie odwołań względnych.

Inne zagadnienia związane z makrami

Na tym etapie rejestrowanie makr w Excelu nie powinno już Ci sprawiać najmniejszych trudności. W tym podrozdziale omówimy kilka dodatkowych zagadnień, o których powinienś pamiętać podczas samodzielnego tworzenia lub rejestrowania makr.

ROZSZERZENIA SKOROSZYTÓW Z OBSŁUGĄ MAKR

Począwszy od wersji 2007, skoroszyty Excela są domyślnie zapisywane z rozszerzeniem *.xlsx*. Niestety pliki w takim formacie nie mogą przechowywać żadnych makr. Jeżeli Twój skoroszyt zawiera jakieś makra i zapiszesz go w formacie *.xlsx*, to automatycznie cały kod VBA zostanie z takiego arkusza usunięty. Na szczęście wcześniej Excel wyświetli na ekranie odpowiednie ostrzeżenie i poprosi o potwierdzenie zamiaru wykonania takiej operacji.

Jeżeli chcesz zachować makra utworzone w skoroszycie, musisz zapisać go jako *Skoroszyt programu Excel z obsługą makr*. Takie pliki posiadają rozszerzenie *.xlsm*. Cała idea takiego rozgraniczenia formatów plików polega na tym, że wszystkie skoroszyty z rozszerzeniem *.xlsx* są automatycznie uznawane za bezpieczne, natomiast skoroszyty *.xlsm* mogą potencjalnie stanowić pewne zagrożenie.

BEZPIECZEŃSTWO MAKR W EXCELU

Począwszy od wersji 2010, firma Microsoft wprowadziła znaczące zmiany w modelu bezpieczeństwa pakietu Office. Jedną z najbardziej istotnych zmian było wdrożenie koncepcji dokumentów zaufanych. W skrócie i bez zbytniego zagłębiania się w szczegóły techniczne możemy powiedzieć, że dokument zaufany (ang. *trusted document*) to skoroszyt, który uznałeś za bezpieczny poprzez włączenie w nim obsługi makr.

Jeżeli otworzysz w Excelu skoroszyt zawierający makra, to na ekranie pojawi się okno ostrzeżenia z informacją, że makra (aktywna zawartość skoroszytu) zostały zablokowane.

Jeżeli naciśniesz przycisk *Włącz makra*, to automatycznie taki skoroszyt zostanie uznany za zaufany. Oznacza to, że Excel nie będzie już więcej pytał o włączenie makr dla tego pliku. Podstawowe założenie tego rozwiązania jest takie, że jeżeli raz pokazałeś Excelowi, że „ufasz” danemu skoroszytowi, włączając w nim możliwość wykonywania makr, to istnieje bardzo duże prawdopodobieństwo, że będziesz takie makra włączał za każdym razem, kiedy będziesz otwierał ten plik. Aby ułatwić Ci życie, Excel zapamiętuje, że już kiedyś włączyłeś makra w danym dokumencie, i zawiesza ponowne wyświetlanie komunikatów o włączeniu makr w tym dokumencie.

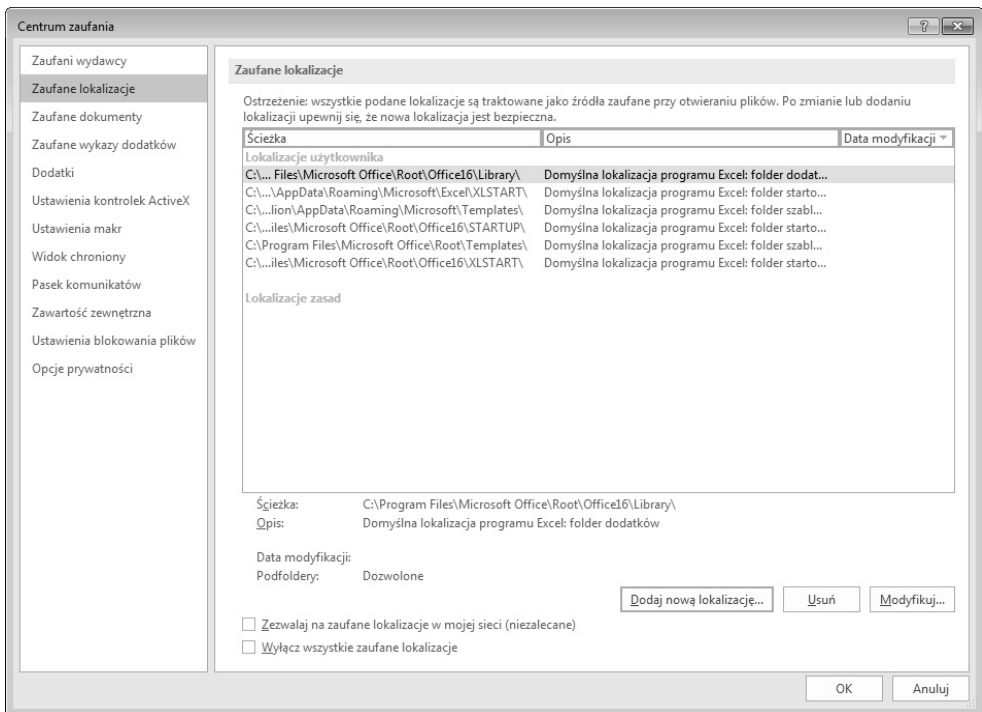
W sumie jest to całkiem niezła wiadomość zarówno dla Ciebie, jak i dla Twoich klientów, którzy po jednorazowym włączeniu makr nie będą już więcej monitowani komunikatami o włączaniu makr, a Ty nie będziesz musiał się martwić, że Twoja oparta na makrach aplikacja Excela przestanie działać z powodu zablokowanych makr.

ZAUFAANE LOKALIZACJE

Jeżeli irytuje Cię konieczność nawet jednorazowego włączenia obsługi makr, to możesz dla swoich skroszytów zdefiniować tak zwaną zaufaną lokalizację, czyli utworzyć dedykowany, bezpieczny katalog, w którym będziesz przechowywać wyłącznie zaufane, sprawdzone i bezpieczne skroszyty. Utworzenie współdzielonej zaufanej lokalizacji pozwoli zarówno Tobie, jak i Twoim klientom na uruchamianie przechowywanych tam skroszytów z obsługą makr bez żadnych ograniczeń.

Aby utworzyć taką zaufaną lokalizację, powinieneś wykonać następujące polecenia:

1. Przejdź na kartę *Deweloper* i wybierz polecenie *Bezpieczeństwo makr*. Na ekranie pojawi się okno dialogowe *Centrum zaufania*.
2. Wybierz opcję *Zaufane lokalizacje*. W prawej części okna dialogowego pojawi się panel *Zaufane lokalizacje* (patrz rysunek 2.6), w którym znajdziesz listę wszystkich katalogów uważanych za zaufane.



Rysunek 2.6. Panel *Zaufane lokalizacje* pozwala na dodawanie nowych katalogów uważanych za bezpieczne

3. Naciśnij przycisk *Dodaj nową lokalizację*.
4. Na ekranie pojawi się okno dialogowe *Zaufana lokalizacja pakietu Microsoft Office*. Naciśnij przycisk *Przeglądaj* i wybierz katalog, który chcesz dodać do listy zaufanych lokalizacji.

Po dodaniu zaufanej lokalizacji wszystkie przechowywane w niej skoroszyty po otwarciu w Excelu będą miały automatycznie włączoną obsługę makr.

ZAPISYWANIE MAKR W SKOROSZYCIE MAKR OSOBISTYCH

Większość makr tworzonych przez użytkowników jest przeznaczona do działania w określonym skoroszycie, ale od czasu do czasu możesz napisać makro, które powinno być dostępne we wszystkich skoroszytach. Takie makra ogólnego przeznaczenia możesz zapisywać w tzw. skoroszycie makr osobistych (ang. *Personal Macro Workbook*), dzięki czemu będziesz je miał zawsze pod ręką. Skoroszyt makr osobistych jest automatycznie ładowany za każdym razem, kiedy uruchamiasz Excela. Plik tego skoroszytu nosi nazwę *personal.xlsm* i jest automatycznie tworzony dopiero w momencie, kiedy po raz pierwszy zarejestrujesz makro, którego docelową lokalizacją będzie ten skoroszyt.

Aby zapisać rejestrowane makro w skoroszycie makr osobistych, powinieneś w oknie *Rejestrowanie makra* wybrać z listy *Przechowuj makro w* opcję *Skoroszyt makr osobistych* (patrz rysunek 2.1 we wcześniejszej części tego rozdziału).

Jeżeli przechowujesz swoje makra w skoroszycie makr osobistych, to nie musisz już pamiętać o jego otwieraniu, ponieważ Excel będzie to robił automatycznie. Jeżeli dokonasz w skoroszycie makr osobistych jakichś zmian, to przed zakończeniem działania Excel zapyta Cię, czy chcesz zachować wprowadzone zmiany.



UWAGA

Skoroszyt makr osobistych jest domyślnie ukryty, tak aby nie przeszkadzał Ci w codziennej pracy z Excelem.

PRZYPISYWANIE MAKR DO PRZYCISKÓW I INNYCH FORMANTÓW

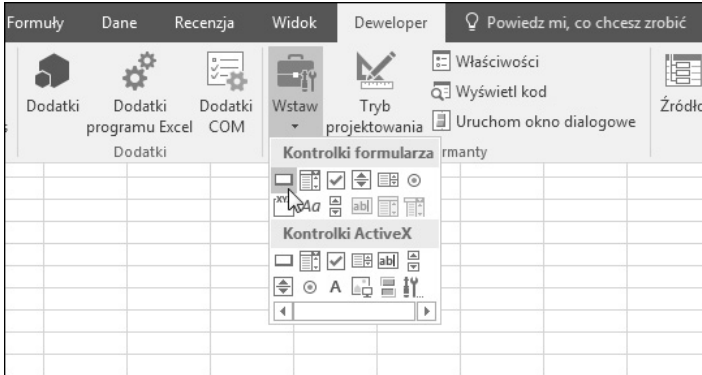
Pracując z makrami, bardzo często będziesz chciał, aby można je było uruchamiać w prosty i wygodny sposób. Jednym z najlepszych rozwiązań jest zazwyczaj utworzenie w skoroszycie przycisku, którego naciśnięcie będzie uruchamiało wybrane makro.

Na szczęście Excel oferuje cały szereg różnego rodzaju formantów pozwalających na tworzenie interfejsów użytkownika bezpośrednio na arkuszu. Istnieje wiele różnych formantów, począwszy od prostych przycisków (to chyba najczęściej wykorzystywany formant), aż do pasków przewijania.

Koncepcja używania formantów jest bardzo prosta. Umieszczasz wybrany formant bezpośrednio na arkuszu, a następnie przypisujesz do niego odpowiednie makro, które zostanie automatycznie uruchomione po kliknięciu lub naciśnięciu takiego formantu.

Aby to wypróbować, utworzymy teraz na arkuszu przycisk i następnie przypiszemy do niego utworzone wcześniej makro *DodajPodsumowanie* i ewzględne. Wykonaj następujące polecenia:

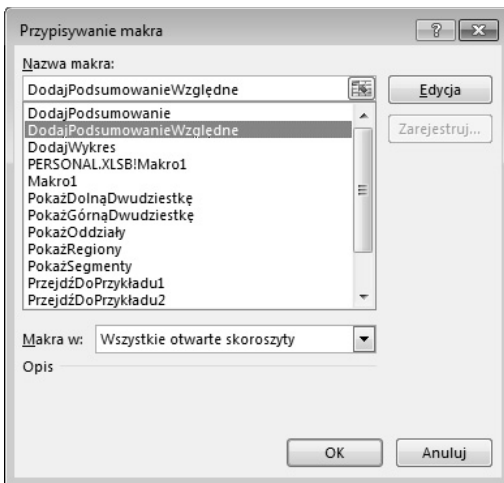
1. Przejdź na kartę *Developer* i naciśnij przycisk *Wstaw* (patrz rysunek 2.7).



Rysunek 2.7. Listę dostępnych formantów znajdziesz na karcie *Developer*

2. Na liście dostępnych kontrolerek odszukaj i naciśnij ikonę formantu *Przycisk* (formant formularza).
3. Kliknij na arkuszu miejsce, w którym chcesz umieścić przycisk.

Kiedy umieścisz przycisk na arkuszu, na ekranie pojawi się okno dialogowe *Przypisywanie makra*, za pomocą którego możesz przypisać do przycisku wybrane makro (patrz rysunek 2.8).



Rysunek 2.8. Przypisywanie wybranego makra do przycisku

4. Wybierz z listy makro, które chcesz przypisać do przycisku, i naciśnij przycisk *OK*.

Od tej chwili masz już gotowy przycisk, za pomocą którego możesz uruchamiać zarejestrowane wcześniej makro. Pamiętaj, że wszystkie formanty dostępne w grupie *Kontrolki formularza* działają podobnie — możesz do nich przypisywać dowolnie wybrane makra, które zostaną uruchomione po kliknięciu formantu lewym przyciskiem myszy.



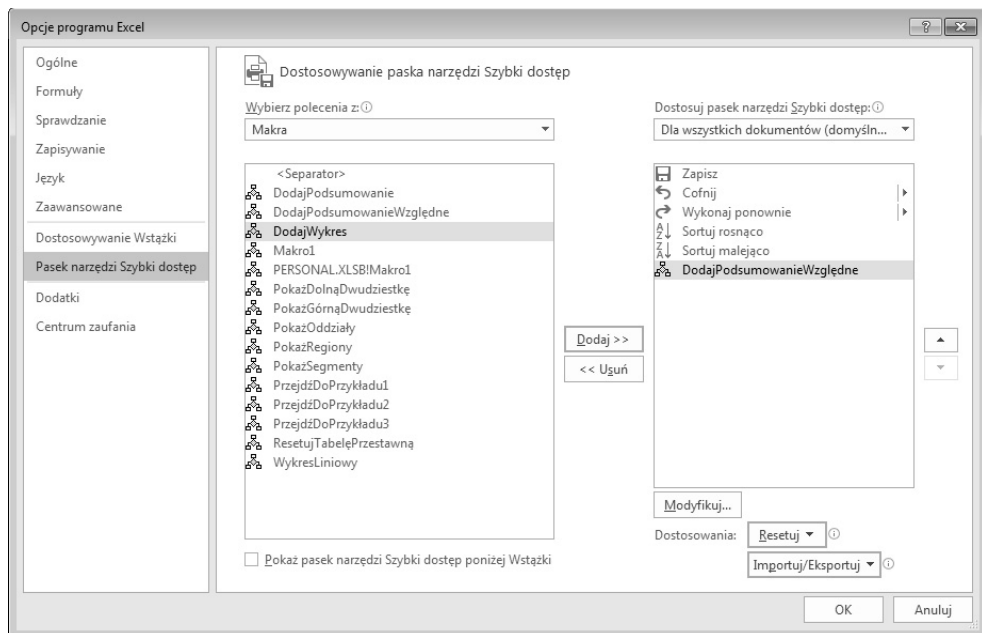
UWAGA

Zwróć uwagę, że formanty widoczne na rysunku 2.7 podzielone są na dwie grupy: *Kontrolki formularza* i *Kontrolki ActiveX*. Kontrolki formularza zostały zaprojektowane specjalnie do użycia bezpośrednio na arkuszach, podczas gdy kontrolki ActiveX są zazwyczaj używane na formularzach *UserForm*. Pracując na arkuszach, w zdecydowanej większości przypadków powinieneś używać kontrolki formularza. Dlaczego? Kontrolki formularza działają bardziej efektywnie, a ich konfiguracja jest zdecydowanie łatwiejsza niż ich odpowiedników — kontrolki ActiveX.

UMIESZCZANIE MAKRA NA PASKU NARZĘDZI SZYBKIEGO DOSTĘPU

Makra możesz również przypisywać do przycisków umieszczanych na pasku narzędzi *Szybki dostęp*, który w zależności od ustawień użytkownika znajduje się powyżej lub poniżej *Wstążki*. Aby umieścić na tym pasku nowy przycisk i przypisać do niego własne makro, powinieneś wykonać polecenia pokazane poniżej:

1. Kliknij pasek narzędzi *Szybki dostęp* prawym przyciskiem myszy i z menu podręcznego, które pojawi się na ekranie, wybierz polecenie *Dostosuj pasek narzędzi Szybki dostęp*. Na ekranie pojawi się okno opcji programu Excel, przedstawione na rysunku 2.9.



Rysunek 2.9. Dodawanie makra do paska narzędzi Szybkiego dostępu

2. W panelu po lewej stronie okna wybierz opcję *Pasek narzędzi Szybki dostęp*.
3. Rozwiń listę *Wybierz polecenia z:* i wybierz z niej opcję *Makra*.
4. Odszukaj i zaznacz wybrane makro, a następnie naciśnij przycisk *Dodaj*.
5. Naciśnij przycisk *Modyfikuj* i wybierz dla makra odpowiednią ikonę przycisku.
6. Naciśnij przycisk *OK*.

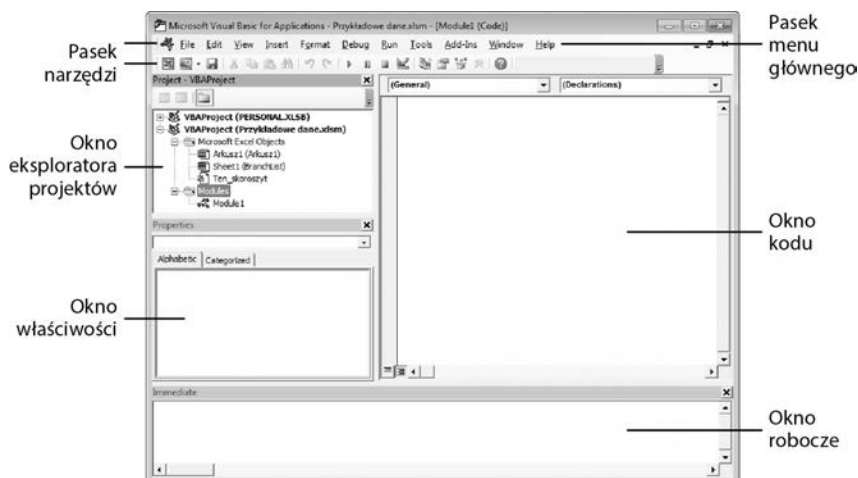
Praca z edytorem Visual Basic Editor (VBE)

Tworzenie i modyfikacja kodu języka VBA odbywa się przy użyciu edytora VBE (ang. *Visual Basic Editor*). Edytor Visual Basic jest co prawda aplikacją oddzielną, ale bardzo ściśle zintegrowaną z Excelem. Nie możesz uruchomić edytora Visual Basic bez uprzedniego uruchomienia programu Excel. Aby uruchomić edytor VBE, naciśnij kombinację klawiszy *Alt+F11*. Aby powrócić do Excela, ponownie naciśnij kombinację klawiszy *Alt+F11*.

Edytor VBE możesz również uruchomić, przechodząc na kartę *Developer* i naciskając przycisk *Visual Basic* znajdujący się w grupie opcji *Kod*.

Podstawowe elementy edytora VBE

Na rysunku 2.10 przedstawiono wygląd głównego okna edytora VBE. Oczywiście w Twoim przypadku okno edytora VBE może wyglądać nieco inaczej. Użytkownik ma duże możliwości dopasowywania wyglądu tego okna do własnych potrzeb — możesz ukrywać poszczególne okna składowe, zmieniać ich rozmiary, zmieniać układ okien na ekranie itd.



Rysunek 2.10. Okno edytora VBE

PASEK MENU GŁÓWNEGO EDYTORA VBE

Pasek menu edytora VBE działa jak każdy inny tradycyjny pasek menu, z którym się do tej pory spotkałeś. Znajdziesz tam szereg poleceń wykorzystywanych w pracy z różnymi elementami składowymi edytora. Wiele poleceń posiada powiązane z nimi skróty klawiaturowe.

Edytor VBE oferuje również menu podręczne, w którym znajdziesz często używane polecenia. Aby przywołać menu na ekran, kliknij prawym przyciskiem myszy w dowolnym miejscu okna edytora VBE.

PASKI NARZĘDZI EDYTORA VBE

Standardowy pasek narzędzi, domyślnie znajdujący się bezpośrednio pod paskiem menu, jest jednym z kilku dostępnych pasków narzędzi edytora VBE (pasek menu jest traktowany jako pasek narzędzi). Paski narzędzi edytora VBE możesz dostosowywać do własnych wymagań, przemieszczać w inne miejsca, wyświetlać i ukrywać w zależności od potrzeb. Aby wyświetlić lub ukryć wybrany pasek narzędzi, z menu głównego wybierz polecenie *View/Toolbars*.

OKNO EKSPLOATORA PROJEKTÓW

W oknie *Project* znajdziesz diagram zawierający wszystkie skoroszyty aktualnie otwarte w Excelu (w tym także dodatki i skoroszyty ukryte), które możesz dowolnie związać i rozwijać według potrzeb. Każdy skoroszyt jest określany mianem *projektu*. Okno eksploratora projektów bardziej szczegółowo zostanie omówione w podrozdziale zatytułowanym „Tajemnice okna Project” w dalszej części tego rozdziału.

Jeżeli okno *Project* nie jest widoczne, naciśnij kombinację klawiszy *Ctrl+R* lub z menu głównego edytora VBE wybierz polecenie *View/Project Explorer* (widok/eksplorator projektów). Aby ukryć okno *Project*, naciśnij przycisk *Close* (zamknij), znajdujący się w prawej części paska tytułowego tego okna. Zamiast tego możesz również kliknąć prawym przyciskiem myszy dowolny obszar okna *Project* i z menu podręcznego, które pojawi się na ekranie, wybrać polecenie *Hide* (ukryj).

OKNO KODU

Okno *Code* (czasem nazywane oknem *Module*) przechowuje kod źródłowy języka VBA. Aby otworzyć okno kodu dla wybranego obiektu, należy dwukrotnie kliknąć taki obiekt w oknie *Project*. Na przykład: aby otworzyć okno *Code* dla obiektu *Arkusz1*, powinieneś po prostu dwukrotnie kliknąć ten arkusz w oknie *Project*. Jeżeli dany obiekt nie posiada przypisanego kodu VBA, okno *Code* będzie puste.

Okno *Code* zostanie bardziej szczegółowo omówione w podrozdziale „Tajemnice okna Code” w dalszej części tego rozdziału.

OKNO ROBOCZE (OKNO IMMEDIATE)

Okno *Immediate* jest bardzo użyteczne w przypadku bezpośredniego wykonywania i testowania poleceń języka VBA oraz usuwania błędów z kodu źródłowego. Okno może być widoczne lub ukryte. Jeżeli okno *Immediate* jest niewidoczne, powinieneś nacisnąć kombinację

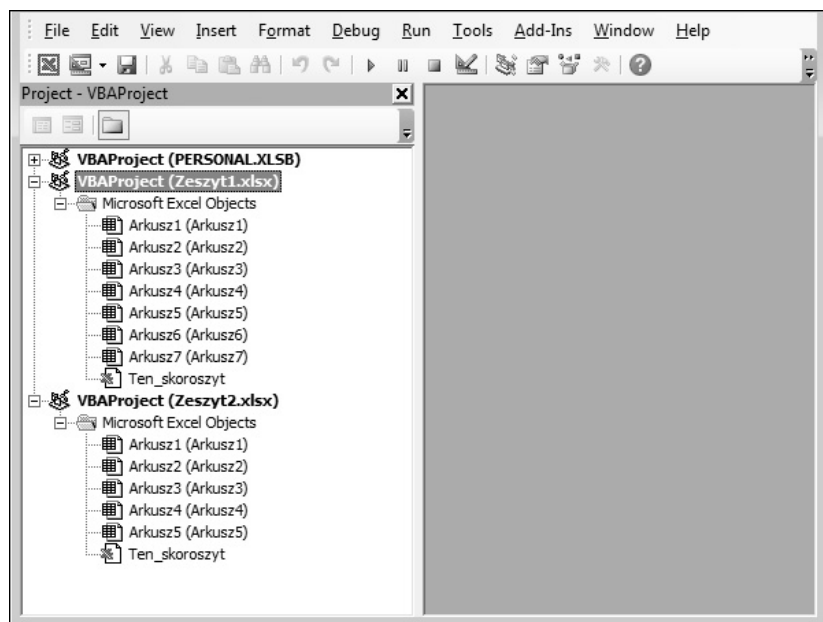
klawiszy *Ctrl+G*. Aby je zamknąć, kliknij przycisk *Zamknij* na pasku tytułowym. Zamiast tego możesz również prawym przyciskiem myszy kliknąć w dowolnym miejscu okna *Immediate* i z menu podręcznego wybrać polecenie *Hide*.

Okno *Immediate* jest bardzo użyteczne — nadaje się zwłaszcza do bezpośredniego wykonywania poleceń VBA oraz testowania i wyszukiwania błędów w kodzie. Jeżeli dopiero zaczynasz swoją przygodę z językiem VBA, okno *Immediate* nie będzie Ci zbyt potrzebne, więc możesz je po prostu ukryć.

Tajemnice okna Project

Kiedy pracujesz z edytorem VBE, każdy otwarty skoroszyt i dodatek Excela jest traktowany jako projekt. *Projekt* można potraktować jako zbiór obiektów uporządkowanych na wzór konceptu. Aby rozwinąć zawartość projektu, należy kliknąć ikonę ze znakiem plus, znajdującą się z lewej strony nazwy projektu w oknie *Project*. W celu zwinienia zawartości projektu należy kliknąć ikonę ze znakiem minus, widoczną z lewej strony nazwy projektu. Jeżeli spróbujesz rozwinąć zawartość projektu chronionego hasłem, zostaniesz poproszony o wpisanie hasła.

Na rysunku 2.11 pokazano okno *Project* zawierające trzy projekty (skoroszyt makr osobistych, skoroszyt o nazwie *Zeszyt1* oraz skoroszyt o nazwie *Zeszyt2*).



Rysunek 2.11. W oknie Project widoczne są trzy projekty; dwa z nich mają rozwiniętą listę obiektów

W każdym projekcie rozwiniętym w oknie *Project* wyświetlany jest przynajmniej jeden węzeł o nazwie *Microsoft Excel Objects*. Po jego rozwinięciu będą widoczne poszczególne arkusze i arkusze wykresu zawarte w danym skoroszycie (każdy arkusz jest traktowany jak osobny obiekt). Dodatkowo wyświetlany jest również obiekt o nazwie *ThisWorkbook*, który reprezentuje obiekt *Workbook*. Jeżeli projekt zawiera moduły VBA, na liście pojawi się też węzeł *Modules* reprezentujący moduły kodu VBA.

DODAWANIE NOWEGO MODUŁU KODU VBA

Kiedy rejestrujesz makro, Excel automatycznie wstawia odpowiedni moduł VBA zawierający wygenerowany kod źródłowy. Skoroszyt, w którym przechowywany jest rejestrowany kod, zależy od docelowej lokalizacji makra, którą wybrałeś w oknie dialogowym przed rozpoczęciem rejestrowania.

Ogólnie rzecz biorąc, w oknie kodu możesz przechowywać trzy rodzaje kodu języka VBA:

- **Deklaracje.** *Deklaracja* to informacja na temat zmiennej zastosowanej w kodzie źródłowym języka VBA. Możesz na przykład zadeklarować typ danych dla zmiennych, których planujesz użyć w procedurze czy funkcji.
- **Procedury Sub.** Jak pamiętasz, *procedura* to inaczej zbiór poleceń, które wykonują określone operacje. Wszystkie makra są rejestrowane w postaci procedur Sub.
- **Procedury Function (funkcje).** *Funkcja* jest zbiorem instrukcji zwracających pojedynczą wartość lub tablicę (pod względem działania przypominają funkcje arkusza, na przykład funkcję SUMA).

W pojedynczym module VBA możesz przechowywać dowolną liczbę procedur, funkcji oraz deklaracji. Sposób zorganizowania modułu VBA jest całkowicie zależny od Ciebie. Niektórzy programiści umieszczają cały kod VBA aplikacji w jednym module, z kolei inni preferują podział i przechowanie poszczególnych części kodu w kilku różnych modułach.

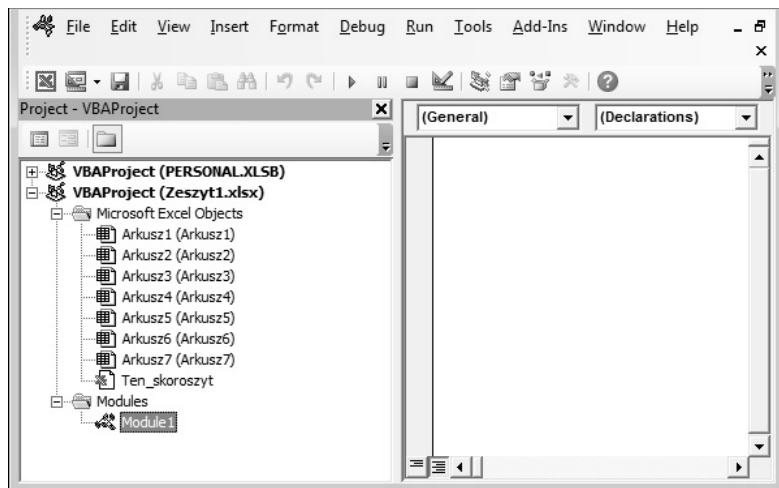
Aby dodać nowy moduł kodu VBA do projektu, powinieneś wykonać następujące polecenia:

1. Zaznacz w oknie *Project* nazwę wybranego projektu.
2. Z menu głównego edytora VBE wybierz polecenie *Insert/Module* (wstaw/moduł).

Inny sposób:

1. Kliknij prawym przyciskiem myszy nazwę wybranego projektu.
2. Z menu podręcznego wybierz polecenie *Insert/Module*.

Nowy moduł jest dodawany w folderze *Modules* projektu (patrz rysunek 2.12). Wszystkie inne moduły kodu VBA będą również umieszczane w tym folderze.



Rysunek 2.12. Moduły kodu umieszczane są w folderze Modules

USUWANIE MODUŁU KODU VBA

Jeżeli chcesz usunąć z danego projektu moduł kodu VBA, który nie jest już potrzebny, powinienś wykonać polecenia opisane poniżej:

1. Zaznacz w oknie *Project* nazwę wybranego projektu.
2. Z menu głównego edytora VBE wybierz polecenie *File/Remove xxx* (plik/usuń xxx), gdzie xxx to nazwa usuwanego modułu.

Inny sposób:

1. Kliknij prawym przyciskiem myszy nazwę modułu, który chcesz usunąć.
2. Z menu podręcznego wybierz polecenie *Remove xxx*.



UWAGA

Możesz usuwać moduły kodu VBA, ale nie masz możliwości usunięcia modułów kodu źródłowego, które są powiązane ze skoroszytem (np. moduł kodu *ThisWorkbook*) lub arkuszem (np. moduł kodu dla obiektu *Arkusz1*).

Tajemnice okna Code

Gdy już nabierzesz wprawy w korzystaniu z języka VBA, będziesz spędzał *mnóstwo* czasu na pracy z oknami *Code* (okno kodu źródłowego). W modułach kodu zapisywane są rejestrowane makra, ale możesz również ręcznie wpisywać w nich kod procedur.

MINIMALIZACJA I MAKSYMALIZACJA OKIEN KODU

Okna kodu zachowują się bardzo podobnie jak okna arkuszy Excela. Możesz je minimalizować, maksymalizować, ukrywać, zmieniać ich położenie itd. Wiele osób twierdzi, że najefektywniej pracuje się z oknami kodu rozciągniętymi do maksymalnych rozmiarów. Dzięki takiemu rozwiązaniu możesz zobaczyć większą ilość kodu źródłowego i skupić się na wykonywanym zadaniu.

Aby zmaksymalizować okno *Code*, naciśnij przycisk *Maksymalizuj* znajdujący się na pasku tytułowym okna lub dwukrotnie kliknij pasek tytułu. Aby przywrócić poprzednią wielkość okna *Code*, naciśnij przycisk *Przywróć okno* znajdujący się poniżej paska tytułowego okna aplikacji.

Czasem jednak bardzo korzystne może być jednoczesne wyświetlenie dwóch lub większej liczby okien *Code*, na przykład kiedy chcesz porównać kod źródłowy zawarty w dwóch modułach lub przekopiować kod z jednego modułu do drugiego. Aby to zrobić, możesz zmienić wielkość i położenie okien ręcznie albo skorzystać z automatycznej zmiany układu, wybierając z menu głównego edytora VBE polecenie *Window/Tile Horizontally* lub *Window/Tile Vertically*.

Aby szybko przechodzić między kolejnymi oknami kodu, powinieneś naciskać kombinację klawiszy *Ctrl+Tab*. Aby przechodzić między oknami kodu w odwrotnej kolejności, naciskaj kombinację klawiszy *Ctrl+Shift+Tab*.

Minimalizacja okna *Code* spowoduje jego ukrycie. Aby całkowicie zamknąć okno *Code*, naciśnij przycisk *Zamknij* znajdujący się na pasku tytułowym okna. Zamknięcie okna powoduje wyłączenie jego ukrycie — zamykając okno, nie utracisz w żaden sposób przechowywanego w nim kodu. Aby ponownie otworzyć okno kodu, wystarczy dwukrotnie kliknąć odpowiedni obiekt w oknie *Project*.

WPROWADZANIE KODU JĘZYKA VBA DO MODUŁU

Zanim będziesz mógł czegokolwiek dokonać za pomocą języka VBA, musisz w oknie *Code* umieścić odpowiedni kod programu, mający postać procedury. Kod programu możesz umieścić w module VBA na trzy sposoby:

- Użyj rejestratora makr programu Excel, który umożliwi zarejestrowanie wykonywanych operacji i automatyczną zamianę na odpowiedni kod źródłowy języka VBA.
- Wprowadź kod źródłowy bezpośrednio przy użyciu klawiatury.
- Skopiuj kod z innego modułu i wklej go do modułu, nad którym pracujesz.

Zapoznałeś się już z rewelacyjnymi możliwościami, jakie daje tworzenie kodu VBA za pomocą rejestratora makr. Niestety istnieją operacje, których nie można przekształcić na kod VBA za pomocą rejestratora makr. W takiej sytuacji jedynym rozwiązaniem pozostaje ręczne utworzenie odpowiedniego kodu. Innymi słowy, musisz po prostu ręcznie „wklepać” odpowiedni kod programu lub po prostu skopiować i wkleić kod pochodzący z innego źródła (na przykład ze strony internetowej).

Wprowadzanie i edytowanie kodu źródłowego modułu VBA wygląda dokładnie tak, jak mogłeś się tego spodziewać — kod programu możesz zaznaczać, kopiować lub wycinać, a następnie wklejać w inne miejsce.

Pojedynczy wiersz kodu VBA może być tak długi, jak tylko będzie to potrzebne, aczkolwiek dla zachowania czytelności kodu długą instrukcję warto podzielić na dwie lub większą liczbę wierszy. Aby to zrobić, na końcu wiersza po spacji powinieneś wstawić znak podkreślenia, nacisnąć klawisz *Enter* i ciąg dalszy instrukcji umieścić w następnym wierszu. Oto przykład pojedynczego polecenia umieszczonego w trzech wierszach:

```
Selection.Sort Key1:=Range("A1"), _  
    Order1:=xlAscending, Header:=xlGuess, _  
    Orientation:=xlTopToBottom
```

Tak podzielone polecenie będzie działać w dokładnie taki sam sposób jak identyczne polecenie umieszczone w jednym, długim wierszu (oczywiście w takim wypadku z pominięciem znaków kontynuacji wiersza). Zwróć uwagę, że drugi i trzeci wiersz polecenia zostały wcięte. Wcięcia są opcjonalne, ale pozwalają szybko zorientować się, że te wszystkie wiersze rzeczywiście stanowią jedno wyrażenie.

Edytor VBE posiada efektywny mechanizm wycofywania wprowadzonych zmian, pozwalający na wycofanie lub ponowne wykonanie wielu kolejnych poleceń. Jeżeli przypadkowo usuniesz jakiś wiersz polecenia, który jest Ci potrzebny, możesz nacisnąć przycisk *Undo* (wycofaj), znajdujący się na pasku narzędzi edytora VBE, lub po prostu nacisnąć kombinację klawiszy *Ctrl+Z*, co spowoduje przywrócenie skasowanego wiersza. Po wycofaniu dowolnego polecenia lub serii poleceń aktywuje się przycisk *Redo* (wykonaj ponownie), który pozwala na ponowne wykonanie wycofanych poleceń.

Aby przekonać się, jak w praktyce wygląda tworzenie kodu procedury VBA, wykonaj polecenia opisane poniżej:

1. Utwórz w Excelu nowy skoroszyt.
2. Przejdź do edytora VBE, naciskając kombinację klawiszy *Alt+F11*.
3. W oknie *Project* kliknij nazwę nowo utworzonego skoroszytu.
4. Wstaw do projektu nowy moduł VBA, wybierając z menu głównego edytora polecenie *Insert/Module* (wstaw/moduł).

5. W oknie kodu wpisz procedurę VBA przedstawioną poniżej:

```
Sub GuessName()  
    Dim Msg as String  
    Dim Ans As Long  
    Msg = "Czy nazywasz się " & Application.UserName & "?"  
    Ans = MsgBox(Msg, vbYesNo)  
    If Ans = vbNo Then MsgBox "No cóż, każdy może się pomylić."  
    If Ans = vbYes Then MsgBox "Jestem jasnowidzem!"  
End Sub
```

6. Upewnij się, że kursor znajduje się w dowolnym miejscu kodu, i uruchom procedurę, naciskając klawisz *F5*.



WSKAZÓWKA

Klawisz *F5* spełnia rolę skrótu klawiszowego dla polecenia *Run/Run Sub/UserForm* (uruchom/uruchom procedurę lub formularz *UserForm*) z menu głównego edytora VBE.

Wprowadzając kod procedury, z pewnością zauważyłeś, że edytor VBE automatycznie dokonuje drobnych poprawek i korekt wpisywanego kodu. Kiedy na przykład wpiszesz polecenie *Sub* rozpoczynające procedurę, edytor VBE automatycznie je dopełni, wstawiając polecenie *End Sub* kończące procedurę. Podobnie jeżeli pominiiesz spację przed lub po znaku równości, edytor VBE automatycznie wstawi pominiętą spację. Oprócz tego edytor VBE zmienia kolor czcionki słów kluczowych języka VBA oraz zmienia wielkość liter wybranych fragmentów tekstu. Jest to całkowicie normalne i pożądane zachowanie, dzięki któremu VBE ułatwia Ci utrzymywanie porządku i przejrzystości wprowadzanego kodu.

Jeżeli poprawnie wykonałeś wszystkie opisane wyżej polecenia, utworzyłeś właśnie swoją pierwszą procedurę *Sub* języka VBA (nazywaną też *makrem*). Po wykonaniu polecenia uaktywniającego makro (czyli naciśnięciu klawisza *F5*), edytor Visual Basic szybko skompilował kod źródłowy i uruchomił procedurę. Innymi słowy, poszczególne polecenia zostały przetworzone i Excel po prostu wykonał to, co mu nakazano. Nasze przykładowe makro możesz uruchomić dowolną liczbę razy, chociaż po jakimś czasie cała zabawa z pewnością nie będzie już tak atrakcyjna...

Nasza prosta procedura składa się z następujących elementów:

- Deklaracja procedury *Sub* (pierwszy wiersz).
- Deklaracje zmiennych (polecenia *Dim*).
- Przypisywanie wartości do zmiennych (zmiennie *Msg* i *Ans*).
- Wykonanie operacji łączenia łańcuchów tekstu (przy użyciu operatora *&*).
- Korzystanie z wbudowanych funkcji języka VBA (*MsgBox*).
- Używanie wbudowanych stałych języka VBA (*vbYesNo*, *vbNo* oraz *vbYes*).
- Używanie instrukcji warunkowej *If-Then-Else* (dwukrotnie).
- Zakończenie procedury *Sub* (ostatni wiersz).

Jak już wspominaliśmy wcześniej, inną metodą umieszczania kodu w module VBA jest skopiowanie kodu z zewnętrznego źródła i wklejenie go do modułu. Wyobraź sobie, że w ramach jednego projektu utworzyłeś procedurę, która może być przydatna również w innym projekcie. Zamiast tracić czas na ponowne, mozolne wprowadzanie takiego samego kodu źródłowego, możesz po prostu otworzyć odpowiedni skoroszyt, uaktywnić wybrany moduł, skopiować kod wybranej procedury (naciskając kombinację klawiszy *Ctrl+C*) i następnie wkleić go do aktualnie tworzonego modułu VBA (za pomocą kombinacji klawiszy *Ctrl+V*). W razie konieczności po wklejeniu kodu można go zmodyfikować i dopasować do wymagań bieżącego projektu.

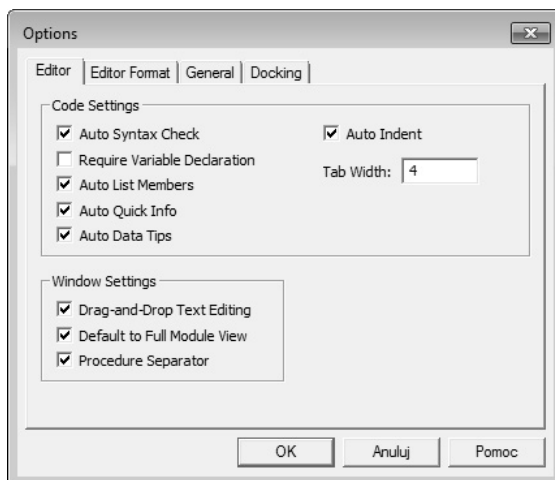
Dostosowywanie środowiska edytora Visual Basic

Jeżeli naprawdę chcesz zostać programistą tworzącym aplikacje Excela, musisz przyzwycząić się do myśli, że będziesz spędzał mnóstwo czasu pracując z edytorem Visual Basic. Aby ułatwić życie programisty, edytor VBE posiada całkiem sporo opcji pozwalających na dopasowanie go do indywidualnych wymagań użytkownika.

Po uruchomieniu edytora Visual Basic z menu *Tools* (narzędzia) wybierz polecenie *Options* (opcje). Na ekranie pojawi się okno dialogowe zawierające cztery karty — *Editor* (ustawienia edytora), *Editor Format* (ustawienia formatowania), *General* (opcje ogólne) oraz *Docking* (ustawienia dokowania okien). W kolejnych podrozdziałach omówimy kilka najważniejszych opcji znajdujących się na tych kartach.

KARTA EDITOR

Na rysunku 2.13 przedstawiono opcje znajdujące się na karcie *Editor* okna dialogowego *Options*, za pomocą których możesz zmieniać sposób działania wybranych mechanizmów edytora VBE.



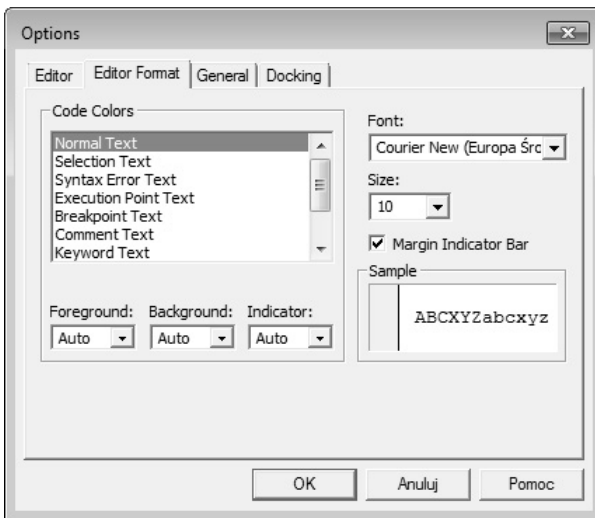
Rysunek 2.13. Karta Editor okna dialogowego Options

- **Opcja *Auto Syntax Check*** — opcja *Auto Syntax Check* określa, czy edytor Visual Basic wyświetli okno dialogowe po wykryciu błędu składni w trakcie wprowadzania kodu źródłowego języka VBA. Okno dialogowe zawiera przybliżoną informację o zaistniałym problemie. Jeżeli opcja *Auto Syntax Check* zostanie wyłączona, edytor Visual Basic oznaczy niepoprawny kod innym kolorem niż pozostałą część kodu, a okno dialogowe z komunikatem o błędzie nie będzie pojawiało się na ekranie. Jeżeli dopiero zaczynasz uczyć się języka VBA, opcja *Auto Syntax Check* może okazać się bardzo pomocna.
- **Opcja *Require Variable Declaration*** — jeżeli opcja *Require Variable Declaration* zostanie włączona, edytor Visual Basic na początku każdego tworzonego modułu VBA wstawi następujące polecenie: `Option Explicit`. Włączenie tej opcji nie ma wpływu na istniejące moduły kodu, ale polecenie będzie wstawiane we wszystkich nowych modułach, które utworzysz od momentu włączenia tej opcji. Jeżeli takie polecenie pojawi się w module, będziesz musiał jawnie zadeklarować wszystkie zmienne używane w tym module. Jednym ze sposobów deklarowania zmiennych jest używanie polecenia `Dim`.
- **Opcja *Auto List Members*** — po włączeniu opcji *Auto List Members* edytor Visual Basic będzie oferował pomoc w trakcie wprowadzania kodu źródłowego języka VBA. Pomoc edytora polega na wyświetlaniu listy elementów powiązanych z obiektem, takich jak metody i właściwości użytego obiektu. Jest to jeden z najbardziej przydatnych mechanizmów edytora VBE.
- **Opcja *Auto Quick Info*** — po włączeniu opcji *Auto Quick Info* edytor Visual Basic wyświetla informacje o argumentach wprowadzanych funkcji, właściwości i metod. Sposób działania tego mechanizmu jest bardzo podobny do sposobu, w jaki Excel podpowiada składnię argumentów funkcji podczas wpisywania formuł w komórkach arkusza.
- **Opcja *Auto Data Tips*** — po włączeniu opcji *Auto Data Tips* możesz ustawić wskaźnik myszy nad wybraną zmienną, a edytor VBE wyświetli na ekranie jej wartość. Pamiętaj jednak, że mechanizm ten działa tylko wtedy, kiedy wystąpi błąd, działanie procedury zostanie wstrzymane i wejdiesz w tryb wyszukiwania i usuwania błędów (ang. *debugging*). Jest to bardzo użyteczna opcja i naprawę nie ma żadnego powodu, aby ją kiedykolwiek wyłączać.
- **Opcja *Auto Indent*** — opcja *Auto Indent* określa, czy edytor Visual Basic automatycznie wstawi na początku każdego nowego wiersza identyczne wcięcie jak w przypadku wcześniej wprowadzonego. Zdecydowana większość programistów VBA jest wielkimi zwolennikami stosowania wcięć w kodzie źródłowym, stąd opcja ta zazwyczaj bywa włączona.
- **Opcja *Drag-and-Drop Text Editing*** — włączenie opcji *Drag-and-Drop Text Editing* umożliwia kopiowanie i przenoszenie tekstu myszą za pomocą metody przeciągnij i upuść.

- Opcja *Default to Full Module View* — opcja *Default to Full Module View* określa sposób przeglądania procedur. Po jej włączeniu procedury zawarte w oknie kodu będą wyświetlane w pojedynczym, przewijanym oknie. Po wyłączeniu tej opcji w danej chwili widoczna będzie tylko jedna procedura.
- Opcja *Procedure Separator* — kiedy opcja *Procedure Separator* jest aktywna, na końcu każdej procedury zawartej w oknie *Code* wyświetlany będzie pasek separatora (przy założeniu oczywiście, że włączona jest również opcja *Default to Full Module View*). Zwykle dobrze jest widzieć, gdzie kończy się kod danej procedury, dlatego zawsze warto pozostawić tę opcję włączoną.

Karta Editor Format

Na rysunku 2.14 przedstawiono kartę *Editor Format* okna dialogowego *Options*. Opcje zamieszczone na tej karcie odpowiadają za wygląd edytora VBE.



Rysunek 2.14. Karta Editor Format okna dialogowego Options

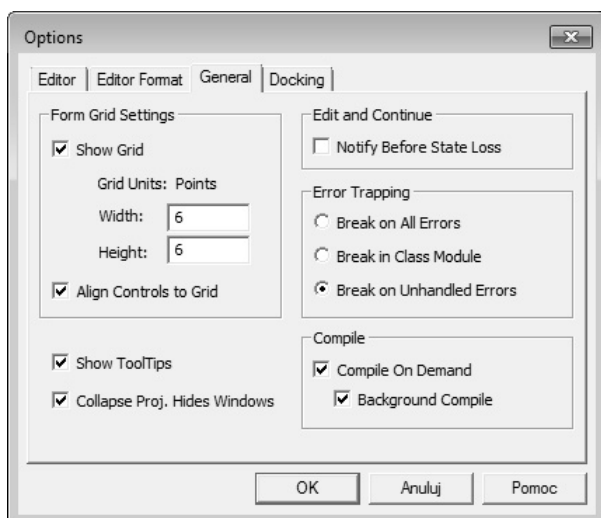
- Opcja *Code Colors* — opcja *Code Colors* umożliwia wybór koloru tekstu (tła i pierwszego planu) oraz kolorów powiązanych z różnymi elementami kodu źródłowego języka VBA. W tym przypadku główną rolę odgrywają indywidualne upodobania każdego użytkownika. Ze swojej strony uważam, że kolory domyślne są całkiem odpowiednie, ale pomimo to od czasu do czasu, chcąc zmienić scenериę, możesz zmienić ustawienia tej opcji.
- Opcja *Font* — opcja *Font* umożliwia wybranie czcionki, która zostanie użyta w modułach VBA. Aby uzyskać jak najlepsze wyniki, powinieneś raczej pozostać przy *czcionce o stałej szerokości*, takiej jak na przykład Courier New.

W takich czcionkach wszystkie znaki mają dokładnie taką samą szerokość. Takie rozwiązanie powoduje, że kod źródłowy jest bardziej czytelny, ponieważ znaki są ładnie wyrównane w pionie i bardzo łatwo można odnaleźć powielane spacje (co w pewnych sytuacjach może być bardzo użyteczne).

- **Pole *Size*** — pole *Size* pozwala określić rozmiar czcionki stosowanej w modułach VBA. Wielkość czcionki zależy zwykle od indywidualnych upodobań użytkownika oraz używanej rozdzielczości obrazu.
- **Opcja *Margin Indicator Bar*** — opcja *Margin Indicator Bar* decyduje o tym, czy będzie wyświetlany pasek wskaźnika pionowego marginesu. Opcja powinna być włączona. W przeciwnym razie podczas pracy w trybie wykrywania i usuwania błędów w kodzie źródłowym nie zobaczysz przydatnych graficznych wskaźników szerokości marginesu.

Karta General

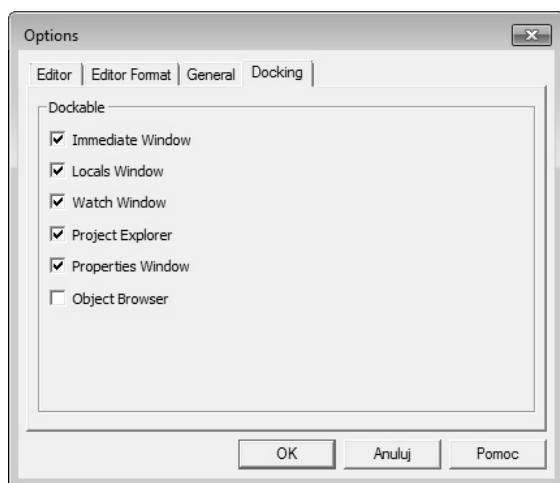
Na rysunku 2.15 przedstawiono opcje znajdujące się na karcie *General* okna dialogowego *Options*. W zdecydowanej większości przypadków ustawienia domyśle będą w zupełności wystarczające. Najważniejszymi opcjami na tej karcie są opcje grupy *Error Trapping* (przechwytywanie i obsługa błędów). Jeżeli dopiero rozpoczynasz swoją karierę jako programista makr Excela, to najlepszym rozwiązaniem będzie wybranie opcji *Break on Unhandled Errors*, która powoduje, że Excel będzie przerywał działanie procedury i wyświetlał odpowiedni komunikat po napotkaniu błędu nieposiadającego w kodzie swojej procedury obsługi.



Rysunek 2.15. Zakładka General okna dialogowego Options

Karta Docking

Na rysunku 2.16 przedstawiono kartę *Docking* okna dialogowego *Options*. Opcje znajdujące się na tej karcie pozwalają określić zachowanie różnych okien edytora Visual Basic. Po zadokowaniu okno znajduje się przy jednej z krawędzi głównego okna edytora Visual Basic, dzięki czemu znacznie łatwiej można je zidentyfikować i zlokalizować. Gdy wszystkie opcje znajdujące się na karcie *Docking* zostaną wyłączone, okna będą nieuporządkowane i na ekranie powstanie niezłe zamieszanie. Zazwyczaj domyślne ustawienia są uznawane za odpowiednie.



Rysunek 2.16. Karta Docking okna dialogowego Options

Podstawowe informacje o języku VBA

Visual Basic for Applications (VBA) jest językiem programowania zorientowanym obiektowo. Podstawowa koncepcja programowania zorientowanego obiektowo polega na tym, że każda aplikacja (w naszym przypadku Excel) składa się z szeregu różnych obiektów, z których każdy posiada swój własny zestaw cech i zastosowań. Excel składa się ze skoroszytów, arkuszy, komórek, wykresów, tabel przestawnych, kształtów i wielu innych obiektów. Każdy z obiektów posiada swój zestaw cech nazywanych **właściami** oraz zestaw zastosowań nazywanych **metodami**.

Taką koncepcję łatwo sobie wyobrazić, ponieważ podobnymi cechami wykazują się wszystkie obiekty, z którymi spotykasz się w codziennym życiu, takie jak Twój komputer, samochód czy lodówka. Każdy z tych obiektów posiada swoje cechy pozwalające na jego identyfikację, takie jak wysokość, waga czy kolor. Każdy z obiektów ma również swoje zastosowanie:

komputera używasz do pracy z Excelem, samochodu używasz do przenoszenia się w inne miejsca, a lodówki do przechowywania żywności w warunkach chłodniczych.

Obiekty VBA również posiadają swoje charakterystyczne właściwości i metody. Komórka arkusza jest obiektem i posiada takie właściwości jak adres, wysokość, kolor wypełnienia i tak dalej. Skoroszyt również jest obiektem VBA, a wśród metod można wymienić otwieranie, zamykanie czy możliwość dodania do skoroszytu wykresu lub tabeli przestawnej.

Pracując w Excelu, codziennie będziesz się stykał ze skoroszytami, arkuszami czy zakresami komórek. Najprawdopodobniej traktujesz te wszystkie wymienione „obiekty” po prostu jako część Excela, tak naprawdę bez separowania ich od siebie. Excel traktuje wszystkie obiekty jako część hierarchicznego modelu, nazywanego **modelem obiektowym Excela**. Model obiektowy Excela jest zdefiniowany jako zestaw obiektów, które są zorganizowane według relacji między nimi.

Obiekty

W rzeczywistym świecie wszystko, co widzisz, możesz opisać jako obiekt. Kiedy patrzysz na swój dom, widzisz obiekt. Twój dom posiada pokoje, a każdy z nich to osobny obiekt. W pokojach stoją szafy, każda z nich to osobny obiekt. Jeżeli teraz pomyślisz o domu, jego pokojach i stojących w nich szafach, zobaczysz gotowy, hierarchiczny model relacji między tymi obiektami. Excel działa dokładnie w taki sam sposób.

W Excelu obiektem nadrzędnym dla wszystkich innych jest obiekt `Application`, będący w Excelu odpowiednikiem Twojego domu. Wewnątrz obiektu `Application` Excel posiada skoroszyty. Wewnątrz każdego skoroszytu znajdują się arkusze, a w każdym arkuszu — zakresy i komórki. Wszystkie wymienione obiekty znajdują się w strukturze hierarchicznej.

Aby w języku VBA wskazać wybrany obiekt Excela, musisz wymienić kolejno wszystkie poziomy hierarchii modelu obiektowego. Aby na przykład zaznaczyć komórkę A1 na arkuszu `Arkusz1`, możesz użyć następującego wiersza kodu:

```
Application.ThisWorkbook.Sheets("Sheet1").Range("A1").Select
```

W większości przypadków hierarchia modelu obiektowego jest na tyle jednoznaczna, że nie musisz podawać nazw wszystkich kolejnych poziomów modelu. Przykładowo wykonanie polecenia przedstawionego poniżej również spowoduje zaznaczenie komórki o adresie A1, ponieważ Excel domyślnie założył, że miałeś na myśli aktywny arkusz bieżącego skoroszytu.

```
Range("A1").Select
```

Jeśli jednak kursor znajdował się już wcześniej w komórce A1, to mogłeś po prostu użyć obiektu `ActiveCell`, całkowicie eliminując konieczność definiowania adresu komórki.

```
Activecell.Select
```

Kolekcje

Wiele obiektów Excela należy do kolekcji. Twój dom stoi zapewne w otoczeniu innych domów, które stanowią kolekcję domów o nazwie *osiedle*. Każde osiedle jest częścią kolekcji *osiedli* tworzących kolejną kolekcję o nazwie *miasto*. Excel traktuje kolekcje jako osobne obiekty.

W każdym skoroszycie (obiekt *Workbook*) znajduje się kolekcja arkuszy (kolekcja o nazwie *Worksheets*). Kolekcja *Worksheets* również jest obiektem, do którego możesz się odwoływać za pomocą kodu VBA. Każdy arkusz w skoroszycie jest częścią kolekcji *Worksheets*.

Jeżeli chcesz odwołać się do danego arkusza w kolekcji *Worksheets*, możesz to zrobić za pomocą pozycji tego arkusza w kolekcji, podając jego numer indeksu (począwszy od 1), lub za pomocą nazwy arkusza ujętej w znaki cudzysłowu. Jeżeli w danym skoroszycie znajduje się tylko jeden arkusz i nosi on nazwę *MójArkusz*, to oba polecenia VBA przedstawione poniżej będą miały taki sam efekt:

```
Worksheets(1).Select  
Worksheets("MójArkusz").Select
```

Jeżeli w danym skoroszycie znajdują się dwa arkusze, jeden o nazwie *MójArkusz*, a drugi o nazwie *TwójArkusz*, i są umieszczone w tej właśnie kolejności, to do drugiego z tych arkuszy możesz odwołać się za pomocą dowolnego z poleceń przedstawionych poniżej:

```
Worksheets(2).Select  
Worksheets("YourSheet").Select
```

Jeżeli chciałbyś się odwołać do arkusza o nazwie *MójArkusz*, znajdującego się w nieaktywnym skoroszycie o nazwie *MojeDane.xlsx*, to będziesz musiał użyć pełnego, kwalifikowanego odwołania do skoroszytu i arkusza, tak jak to zostało pokazane poniżej:

```
Workbooks("MojeDane.xlsx").Worksheets("MójArkusz").Select
```

Właściwości

Każdy obiekt ma swoje właściwości, które określają jego charakterystykę. Twój dom ma określony kolor, powierzchnię, wiek i tak dalej. Niektóre właściwości można zmieniać, tak jak na przykład kolor domu. Inne właściwości nie mogą być zmieniane, na przykład rok, w którym Twój dom został wybudowany.

Podobnie sprawy mają się w Excelu. Obiekt Excela, taki jak *Worksheet* (arkusz), posiada właściwość *Name* (nazwa), którą możesz dowolnie zmieniać, ale już właściwość *Rows.Count* (liczba wierszy) nie może być zmieniana.

Do wybranej właściwości obiektu odwołujemy się, podając najpierw nazwę obiektu, a następnie nazwę właściwości. Na przykład nazwę wybranego arkusza możesz zmienić poprzez zmianę wartości jego właściwości `Name`.

W przykładzie przedstawionym poniżej zmienimy nazwę arkusza z `Arkusz1` na `MójArkusz`:

```
Sheets("Arkusz1").Name = "MójArkusz"
```

Niektóre właściwości są przeznaczone tylko do odczytu, co oznacza, że nie możesz bezpośrednio przypisywać do nich żadnych wartości. Przykładem takiej sytuacji może być właściwość `Text` komórki (czyli obiektu `Cell`). Właściwość `Text` udostępnia Ci sformatowaną wartość danej komórki, której jednak nie możesz w żaden sposób zmieniać ani nadpisywać.

Niektóre właściwości posiadają argumenty, które jeszcze dokładniej określają charakter wartości tej właściwości. Na przykład w wierszu kodu zamieszczonym poniżej używamy argumentów `RowAbsolute` i `ColumnAbsolute` do pobrania wartości adresu komórki `A1` w postaci adresu bezwzględnego (`A1`).

```
MsgBox Range("A1").Address(RowAbsolute:=True, ColumnAbsolute:=True)
```

OKREŚLANIE WŁAŚCIWOŚCI AKTYWNEGO OBIEKTU

Podczas pracy z Excelem za każdym razem może być aktywny tylko jeden skoroszyt. W takim skoroszycie w dowolnej chwili aktywny może być tylko jeden arkusz. Jeżeli arkusz aktywny jest arkuszem komórek, to w danej chwili aktywna może być tylko jedna komórka (nawet jeżeli zaznaczysz obszar składający się z wielu komórek). VBA zna hierarchiczną strukturę skoroszytów, arkuszy i komórek i dzięki temu pozwala na używanie uproszczonego formatu odwołań do aktywnych obiektów.

Takie rozwiązanie często się przydaje, ponieważ nie zawsze będziesz dokładnie wiedział, który konkretnie skoroszyt, arkusz lub zakres zostanie użyty. Język VBA znakomicie ułatwia takie odwołania dzięki wykorzystaniu właściwości obiektu `Application`. Przykładowo obiekt `Application` posiada właściwość `ActiveCell` zwracającą odwołanie do aktywnej komórki. Poniższa instrukcja przypisuje aktywnej komórce wartość 1:

```
ActiveCell.Value = 1
```

Zwróć uwagę, że w powyższym przykładzie pominęliśmy odwołanie do obiektu `Application` oraz aktywnego arkusza. Pamiętaj jednak, że próba wykonania takiego polecenia zakończy się niepowodzeniem, jeżeli arkusz aktywny nie będzie arkuszem komórek. Jeżeli takie polecenie zostanie wykonane w chwili, kiedy aktywny będzie arkusz wykresu, to działanie procedury zostanie zatrzymane i zakończy się wyświetleniem odpowiedniego komunikatu o wystąpieniu błędu.

Jeżeli w danym arkuszu zaznaczymy zakres komórek, komórką aktywną będzie zawsze komórka należąca do zaznaczonego zakresu. Inaczej mówiąc, komórką aktywną jest zawsze pojedyncza komórka arkusza (a nigdy wielokomórkowy zakres).

Obiekt `Application` posiada również właściwość o nazwie `Selection`, która zwraca odwołanie do aktualnie zaznaczonego elementu, którym może być pojedyncza komórka (aktywna komórka), zakres komórek czy obiekt taki jak `ChartObject`, `TextBox` lub `Shape`.

W tabeli 2.1 zamieszczamy listę innych właściwości obiektu `Application`, które mogą być użyteczne podczas pracy z komórkami i zakresami komórek.

TABELA 2.1. NIEKTÓRE PRZYDATNE WŁAŚCIWOŚCI OBIEKTU APPLICATION

Właściwość	Zwracany obiekt
<code>ActiveCell</code>	Aktywna komórka
<code>ActiveChart</code>	Aktywny arkusz wykresu lub wykres powiązany z obiektem <code>ChartObject</code> znajdującym się w arkuszu. Jeżeli wykres nie jest aktywny, właściwość będzie miała wartość <code>Nothing</code>
<code>ActiveSheet</code>	Aktywny arkusz (zwykły lub wykresu)
<code>ActiveWindow</code>	Aktywne okno
<code>ActiveWorkbook</code>	Aktywny skoroszyt
<code>Selection</code>	Zaznaczony obiekt (mogą to być takie obiekty jak <code>Range</code> , <code>Shape</code> , <code>ChartObject</code> itp.).
<code>ThisWorkbook</code>	Skoroszyt zawierający wykonywaną procedurę VBA. Ten obiekt może (ale nie musi) być taki sam jak obiekt <code>ActiveWorkbook</code> .

Zaletą stosowania wymienionych właściwości zwracających obiekty jest to, że nie musisz wiedzieć, która komórka, arkusz lub skoroszyt są aktualnie aktywne, ani też nie musisz tworzyć do nich odwołania. Dzięki temu możesz tworzyć kod źródłowy języka VBA, który nie jest ściśle powiązany z określonym skoroszytem, arkuszem lub zakresem. Na przykład poniższa instrukcja czyści zawartość aktywnej komórki, pomimo że nie jest znany jej adres:

```
ActiveCell.ClearContents
```

Kolejna instrukcja wyświetla komunikat zawierający nazwę aktywnego arkusza:

```
MsgBox ActiveSheet.Name
```

Aby poznać nazwę i ścieżkę aktywnego skoroszytu, należy użyć następującej instrukcji:

```
MsgBox ActiveWorkbook.FullName
```


Gdy zaznaczysz w arkuszu zakres komórek, za pomocą jednego polecenia możesz go w całości wypełnić wybraną wartością. W poniższym przykładzie właściwość `Selection` obiektu `Application` zwraca obiekt `Range` odpowiadający zaznaczonym komórkom. Polecenie po prostu modyfikuje właściwość `Value` obiektu `Range`, a wynikiem takiej operacji jest zakres komórek wypełnionych daną wartością:

```
Selection.Value = 12
```

Jeżeli zostanie zaznaczone coś innego niż zakres (np. obiekt `ChartObject` lub `Shape`), próba wykonania takiego polecenia spowoduje wygenerowanie błędu, ponieważ obiekty `ChartObject` i `Shape` nie posiadają właściwości `Value`.

Zwróć jednak uwagę na fakt, że kolejne polecenie wpisuje wartość 12 do komórek reprezentowanych przez obiekt `Range`, który został zaznaczony jeszcze przed zaznaczeniem innego obiektu, niebędącego obiektem typu `Range`. Jeżeli przyjrzyysz się opisowi właściwości `RangeSelection` w pomocy systemowej, przekonasz się, że właściwość ta ma zastosowanie wyłącznie do obiektu `Window`.

```
ActiveWindow.RangeSelection.Value = 12
```

Aby dowiedzieć się, ile komórek zostało zaznaczonych w aktywnym oknie, należy użyć właściwości `Count`. Oto przykład:

```
MsgBox ActiveWindow.RangeSelection.Count
```

METODY

Oprócz wspomnianych wcześniej właściwości obiekty posiadają również swoje metody. Metoda to inaczej operacja wykonywana na obiekcie. Wracając do naszej analogii „domowej”, w każdej chwili możesz pomalować swój dom na inny kolor, co w języku VBA przekładałoby się na coś w stylu metody `dom.pomaluj`.

Prostym przykładem jest użycie metody `Select` obiektu `Range`:

```
Range("A1").Select
```

Innym przykładem może być użycie metody `Copy` obiektu `Range`:

```
Range("A1").Copy
```

Niektóre metody posiadają argumenty pozwalające na doprecyzowanie sposobu działania metody. Na przykład, metoda `Paste` może działać w sposób bardziej efektywny po zdefiniowaniu argumentu `Destination`:

```
ActiveSheet.Paste Destination:=Range("B1")
```

Kilka słów o argumentach

Jednym z zagadnień, które często wywołują zamieszanie wśród niedoświadczonych programistów używających języka VBA, są argumenty metod i właściwości. Niektóre metody korzystają z argumentów w celu dokładniejszego zdefiniowania operacji. Z kolei niektóre właściwości korzystają z argumentów w celu uzyskania możliwości dodatkowego określenia wartości. W niektórych przypadkach argumenty są opcjonalne.

Weźmy dla przykładu metodę `Protect` obiektu `Workbook`. Jeżeli sprawdzisz w systemie pomocy opis tej metody, to zobaczysz, że metoda `Protect` pobiera trzy argumenty, reprezentujące odpowiednio hasło, status ochrony struktury skoroszytu i status ochrony okna. Argumenty odpowiadają opcjom zawartym w oknie dialogowym *Chronienie struktury i systemu Windows*.

Jeżeli na przykład zależy Ci na ochronie skoroszytu o nazwie *MójSkoroszyt.xlsx*, możesz użyć następującego polecenia:

```
Workbooks("MójSkoroszyt.xlsx").Protect "xyzyzy", True, False
```

W tym przypadku skoroszyt jest chroniony hasłem (argument 1). Dodatkowo jest chroniona jego struktura (argument 2), ale okna skoroszytu już nie (argument 3).

Jeżeli nie chcesz przypisywać hasła, możesz użyć następującej instrukcji:

```
Workbooks("MójSkoroszyt.xlsx").Protect , True, False
```

Pierwszy argument został pominięty, a zamiast niego pozostawiliśmy puste miejsce identyfikowane przy użyciu przecinka.

Kolejna metoda zwiększająca czytelność kodu źródłowego polega na zastosowaniu argumentów, którym nadano nazwy. Oto przykład demonstrujący, w jaki sposób w poprzedniej instrukcji zastosować nazwane argumenty:

```
Workbooks("MójSkoroszyt.xlsx").Protect Structure:=True, Windows:=False
```

Zastosowanie nazwanych argumentów jest dobrym pomysłem, zwłaszcza w przypadku metod mających wiele argumentów opcjonalnych, a także w sytuacji, gdy konieczne jest użycie jedynie kilku z nich. Przy korzystaniu z nazwanych argumentów nie ma potrzeby pozostawiania pustych miejsc identyfikujących pominięte argumenty.

W przypadku właściwości i metod zwracających wartość konieczne jest umieszczenie argumentów w nawiasach okrągłych. Przykładowo właściwość `Address` obiektu `Range` pobiera pięć argumentów, z których wszystkie są opcjonalne. Ze względu na to, że właściwość `Address` zwraca wartość, poniższa instrukcja nie jest poprawna, ponieważ pominięto nawiasy okrągłe:

```
MsgBox Range("A1").Address False ' nieprawidłowa instrukcja
```

Aby składnia takiej instrukcji była poprawna, konieczne jest zastosowanie nawiasów okrągłych:

```
MsgBox Range("A1").Address(False)
```

Instrukcja może też zostać zapisana przy użyciu nazwanego argumentu w następujący sposób:

```
MsgBox Range("A1").Address(rowAbsolute:=False)
```

Tego typu niuanse stają się bardziej zrozumiałe w miarę zdobywania doświadczenia z zakresu języka VBA.

Tajemnice obiektów Range

Wiele zadań realizowanych przy użyciu instrukcji języka VBA jest powiązanych z komórkami i zakresami arkuszy — w końcu w tym właśnie celu stworzono arkusze kalkulacyjne. Z tego względu poświęcimy teraz trochę czasu na bardziej szczegółowe zapoznanie się z obiektem Range.

Wyszukiwanie właściwości obiektów Range

Uruchom edytor VBE i następnie wybierz z jego menu głównego polecenie *Help/Microsoft Visual Basic for Applications Help* (pomoc/pomoc dla języka Microsoft VBA). Zostaniesz przeniesiony na stronę internetową Microsoft Developer Network (MSDN), gdzie w wyszukiwarce możesz wpisać słowo kluczowe Range i zapoznać się z tematami pomocy dostępnymi dla obiektu Range. Dowiesz się tam, że obiekt Range posiada trzy właściwości, których możesz używać podczas pracy z arkuszami. Są to:

- właściwość Range obiektu klasy Worksheet lub Range,
- właściwość Cells obiektu Worksheet,
- właściwość Offset obiektu Range.

Właściwość Range

Właściwość Range zwraca obiekt Range. Jeżeli poszukasz w pomocy systemowej, to dowiesz się, że ta właściwość posiada dwie wersje składni:

```
obiekt.Range(komórka1)  
obiekt.Range(komórka1, komórka2)
```

Właściwość `Range` jest powiązana z dwoma typami obiektów — `Worksheet` lub `Range`. Słowa `komórka1` i `komórka2` są odpowiednikami terminów, które Excel traktuje jako identyfikator zakresu (w pierwszym przypadku) lub jego granice (drugi przypadek). Poniżej zamieszczono kilka przykładów użycia właściwości `Range`.

Do tej pory poznałeś przykłady podobne do zamieszczonego poniżej. Polecenie takie po prostu wpisuje wybraną wartość do określonej komórki arkusza. W tym konkretnym przypadku w komórce `A1` arkusza `Arkusz1` jest umieszczana wartość `12.3`.

```
Worksheets("Arkusz1").Range("A1").Value = 12.3
```

Właściwość `Range` rozpoznaje też nazwy zdefiniowane w skoroszytach. A zatem jeżeli komórce nadano nazwę `Input`, można wpisać do niej wartość przy użyciu następującego polecenia:

```
Worksheets("Arkusz1").Range("Input").Value = 100
```

W kolejnym przykładzie do zakresu złożonego z 20 komórek aktywnego arkusza jest wstawiana taka sama wartość (jeżeli aktywna karta Excela nie jest zwykłym arkuszem, zostanie wygenerowany komunikat błędu):

```
ActiveSheet.Range("A1:B10").Value = 2
```

Kolejne polecenie daje takie same wyniki, jak poprzednio:

```
Range("A1", "B10") = 2
```

Co prawda w powyższym poleceniu pominięto odwołanie do aktywnego arkusza, ale jest ono używane domyślnie. Poza tym pominięto właściwość `Value` obiektu `Range`, ponieważ jest to właściwość domyślna. W powyższym przykładzie użyto drugiej składni właściwości `Range`. W jej przypadku pierwszym argumentem jest górna lewa komórka, natomiast drugim — dolna prawa komórka zakresu.

W następnym przykładzie zastosowano operator przecięcia zakresów Excela (znak spacji), który zwraca przecięcie dwóch zakresów. W tym przypadku w miejscu przecięcia zakresów znajduje się jedna komórka, `C6`, stąd następująca instrukcja wpisuje do tej komórki wartość `3`:

```
Range("C1:C10 A6:E6") = 3
```

Na koniec kolejne polecenie wpisuje wartość `4` do pięciu komórek tworzących nieciągły zakres. Przecinek spełnia funkcję operatora złączenia (zwróć uwagę, że ciąg odwołań do poszczególnych komórek, oddzielonych od siebie przecinkami, został ujęty w znaki cudzysłowu):

```
Range("A1, A3, A5, A7, A9") = 4
```

Dotychczas we wszystkich przykładach używaliśmy właściwości `Range` obiektu `Worksheet`. Jak już wspominaliśmy wcześniej, możesz też używać właściwości `Range` obiektu `Range`.

Poniżej zamieszczono przykład takiego użycia właściwości Range obiektu Range. W tym przypadku obiektem Range jest aktywna komórka. Poniższe polecenie traktuje obiekt Range tak, jakby komórka znajdowała się w górnym lewym narożniku arkusza, i wstawia wartość 5 do komórki, którą *będzie* B2. Innymi słowy, zwrócone odwołanie jest względne w stosunku do górnego lewego narożnika obiektu Range. A zatem instrukcja wpisze wartość 5 bezpośrednio do komórki znajdującej się na prawo od aktywnej komórki i jeden wiersz poniżej:

```
ActiveCell.Range("B2") = 5
```

Jest to trochę zmatwane, ale na szczęście istnieje o wiele bardziej zrozumiała metoda uzyskania dostępu do komórki względnej w stosunku do zakresu. Jest to właściwość Offset, którą omówimy w jednym z kolejnych podrozdziałów.

Właściwość Cells

Kolejna metoda odwoływania się do zakresu polega na użyciu właściwości Cells. Podobnie jak właściwość Range, tak i właściwość Cells jest używana w stosunku do obiektów Worksheet i Range. Kiedy zajrzysz do pomocy systemowej programu Excel, przekonasz się, że właściwość Cells ma trzy warianty składni:

```
obiekt.Cells(rowIndex, columnIndex)  
obiekt.Cells(rowIndex)  
obiekt.Cells
```

Poniżej podano kilka przykładów zastosowania właściwości Cells. W pierwszym z nich do komórki A1 arkusza Arkusz1 zostanie wpisana wartość 9. W tym przypadku posłużę się pierwszą składnią akceptującą numer indeksu wiersza (wartości z przedziału od 1 do 1048576) i numer indeksu kolumny (wartości z przedziału od 1 do 16384):

```
Worksheets("Arkusz1").Cells(1, 1) = 9
```

Kolejny przykład to instrukcja wpisująca do komórki D3 (czyli komórki znajdującej się na przecięciu wiersza numer 3 i kolumny numer 4 aktywnego arkusza) wartość 7:

```
ActiveSheet.Cells(3, 4) = 7
```

Można też użyć właściwości Cells obiektu Range. W tym przypadku obiekt Range zwracany przez właściwość Cells jest względny w stosunku do górnej lewej komórki obiektu Range, do którego jest wykonywane odwołanie. Zmatwane? Raczej tak. Kolejny przykład może sprawić, że stanie się to bardziej przejrzyste. Instrukcja wpisuje do aktywnej komórki wartość 5. Pamiętaj, że aktywna komórka jest traktowana tak, jakby była komórką A1 arkusza.

```
ActiveCell.Cells(1, 1) = 5
```

**UWAGA**

Prawdziwa korzyść wynikająca z zastosowania tego typu odwołania do komórek stanie się widoczna przy omawianiu zmiennych i pętli (patrz rozdział 3.). W większości przypadków zamiast faktycznych wartości argumentów będziesz raczej używał zmiennych.

Aby do komórki znajdującej się bezpośrednio poniżej aktywnej komórki wpisać wartość 5, należy użyć następującej instrukcji:

```
ActiveCell.Cells(2, 1) = 5
```

Instrukcję tę można opisać za pomocą następującego zdania: „Rozpocznij od aktywnej komórki i traktuj ją tak, jakby to była komórka A1. Następnie wpisz wartość 5 w komórce znajdującej się w drugim wierszu i pierwszej kolumnie”.

W drugim wariancie składni właściwości `Cells` jest używany jeden argument, który może przyjmować wartości z przedziału od 1 do 17 179 869 184. Wartość ta jest równa liczbie komórek arkusza programu Excel 2013. Komórki są numerowane, począwszy od A1, kolejno w prawą stronę, a następnie od początku następnego wiersza. Na przykład komórka numer 16 384 ma adres `XFD1`, natomiast komórka o numerze 16 385 ma już adres `A2`.

W następnym przykładzie do komórki `SZ1` aktywnego arkusza (która jest 520. komórką tego arkusza) wprowadzono wartość 2.

```
ActiveSheet.Cells(520) = 2
```

Aby wyświetlić wartość ostatniej komórki arkusza (o adresie `XFD1048576`), należy użyć następującej instrukcji:

```
MsgBox ActiveSheet.Cells(17179869184)
```

Taka składnia może też zostać użyta w przypadku obiektu `Range`. W tym przypadku zwrócona komórka jest względna w stosunku do obiektu `Range`, do którego się odwołano. Jeżeli na przykład obiekt `Range` reprezentuje zakres `A1:D10` (40 komórek), właściwość `Cells` może pobrać argument o wartości z przedziału od 1 do 40 i zwrócić jedną z komórek obiektu `Range`. W poniższym przykładzie wartość 2000 jest wpisywana do komórki `A2`, ponieważ jest to piąta komórka zakresu (licząc od góry i w prawą stronę, a następnie od początku następnego wiersza), do którego się odwołano:

```
Range("A1:D10").Cells(5) = 2000
```

**UWAGA**

W poprzednim przykładzie argument właściwości `Cells` nie jest ograniczony do wartości z przedziału od 1 do 40. Jeżeli wartość argumentu przekracza liczbę komórek zakresu, zliczanie jest kontynuowane, tak jakby zakres był większy niż w rzeczywistości. A zatem polecenie podobne do powyższego może zmienić wartość komórki spoza zakresu `A1:D10`. Na przykład polecenie przedstawione poniżej zmienia wartość komórki `A11`:

```
Range("A1:D10").Cells(41) = 2000
```

Trzeci wariant składni właściwości `Cells` po prostu zwraca wszystkie komórki arkusza, do którego jest wykonywane odwołanie. W przeciwieństwie do dwóch pozostałych składni w tym przypadku nie jest zwracana pojedyncza komórka. Poniższa instrukcja używa metody `ClearContents` w stosunku do zakresu zwróconego przez właściwość `Cells` aktywnego arkusza. W wyniku wykonania instrukcji czyszczona jest zawartość wszystkich komórek arkusza.

```
ActiveSheet.Cells.ClearContents
```

Pobieranie informacji z komórki

Jeżeli chcesz odczytać zawartość wybranej komórki, to VBA udostępni Ci kilka właściwości. Poniżej zamieszczamy krótkie zestawienie najczęściej używanych właściwości:

- Właściwość `Formula` zwraca formułę zapisaną w pojedynczej komórce (pod warunkiem oczywiście, że taka formuła istnieje). Jeżeli dana komórka nie posiada formuły, zwracana jest wartość komórki. Właściwość `Formula` jest właściwością przeznaczoną do odczytu i zapisu i posiada kilka podobnych odmian, takich jak `FormulaR1C1`, `FormulaLocal` czy `FormulaArray` (więcej szczegółowych informacji na ich temat znajdziesz w systemie pomocy).
- Właściwość `Value` zwraca nieformatowaną wartość komórki. Jest to właściwość przeznaczona do odczytu i zapisu.
- Właściwość `Text` zwraca tekst, który jest wyświetlany w komórce. Jeżeli w danej komórce znajduje się wartość numeryczna, właściwość zwraca wartość razem z formatowaniem, takim jak przecinki czy symbole walut. Właściwość `Text` jest przeznaczona tylko do odczytu.
- Właściwość `Value2` jest nieco podobna do właściwości `Value`, z wyjątkiem tego, że nie korzysta z danych typu `Date` oraz `Currency`. W przypadku napotkania takich typów danych, właściwość dokonuje ich konwersji na zmienne typu `Variant` zawierające wartości typu `Double`. Na przykład, jeżeli w komórce została zapisana data 2016-03-16, to właściwość `Value` zwróci ją jako wartość typu `Date`, podczas gdy właściwość `Value2` zwróci wartość typu `Double` (w tym wypadku 42445).

Właściwość Offset

Właściwość `Offset` (podobnie jak właściwości `Range` i `Cells`) również zwraca obiekt typu `Range`. Jednak w przeciwieństwie do dwóch poprzednio omawianych metod, właściwość `Offset` jest powiązana tylko i wyłącznie z obiektem `Range`. Oto jej składnia:

```
obiekt.Offset(rowOffset, columnOffset)
```

Właściwość `Offset` pobiera dwa argumenty odpowiadające względnemu położeniu odniesionemu do górnej lewej komórki określonego obiektu `Range`. Argumentami mogą być wartości dodatnie (przesunięcie w dół lub w prawo), ujemne (przesunięcie w górę lub w lewo) lub zero. Poniższa instrukcja wpisuje wartość 12 do komórki znajdującej się bezpośrednio poniżej komórki aktywnej:

```
ActiveCell.Offset(1, 0).Value = 12
```

Kolejna instrukcja wstawia wartość 15 do komórki znajdującej się bezpośrednio powyżej aktywnej komórki:

```
ActiveCell.Offset(-1, 0).Value = 15
```

Nawiasem mówiąc, jeżeli aktywna komórka znajduje się w wierszu 1, właściwość `Offset` z powyższej instrukcji spowoduje wygenerowanie błędu, ponieważ nie będzie mogła zwrócić obiektu `Range`, który nie istnieje.

Właściwość `Offset` jest dość przydatna, zwłaszcza w przypadku stosowania zmiennych w procedurach wykonujących pętlę. Więcej szczegółowych informacji na ten temat znajdziesz w kolejnym rozdziale.

Po zarejestrowaniu makra w trybie odwołań względnych, Excel użyje właściwości `Offset` w celu odwołania się do komórek względnych w stosunku do pozycji początkowej (komórki, która jest aktywna w momencie rozpoczęcia rejestrowania makra). Na przykład do wygenerowania poniższego kodu użyłem rejestratora makr. Najpierw umieściłem kursor w komórce B1, a następnie wprowadziłem wartości do komórek zakresu B1:B3 i powróciłem do komórki B1.

```
Sub Makro1()  
    ActiveCell.FormulaR1C1 = "1"  
    ActiveCell.Offset(1, 0).Range("A1").Select  
    ActiveCell.FormulaR1C1 = "2"  
    ActiveCell.Offset(1, 0).Range("A1").Select  
    ActiveCell.FormulaR1C1 = "3"  
    ActiveCell.Offset(-2, 0).Range("A1").Select  
End Sub
```

Zwróć uwagę, że rejestrator makr użył właściwości `FormulaR1C1`. Zazwyczaj w celu wprowadzenia wartości do komórki będziemy używać właściwości `Value`, co nie zmienia jednak faktu, że po zastosowaniu właściwości `FormulaR1C1` lub nawet `Formula` uzyskasz takie same wyniki.

Warto również zauważyć, że wygenerowany kod odwołuje się do komórki A1, co może wydawać się trochę dziwne, ponieważ komórka ta nie była nawet użyta w trakcie rejestrowania makra. Jest to nietypowy element procedury rejestrowania, który sprawia, że kod źródłowy jest niepotrzebnie aż tak złożony. Możesz śmiało usunąć wszystkie odwołania do Range("A1"), a makro nadal będzie znakomicie działać:

```
Sub ZmodyfikowaneMakro1()  
    ActiveCell.FormulaR1C1 = "1"  
    ActiveCell.Offset(1, 0).Select  
    ActiveCell.FormulaR1C1 = "2"  
    ActiveCell.Offset(1, 0).Select  
    ActiveCell.FormulaR1C1 = "3"  
    ActiveCell.Offset(-2, 0).Select  
End Sub
```

Poniżej zamieszczamy jeszcze bardziej efektywną wersję makra, która nie używa metody Select:

```
Sub Makro1()  
    ActiveCell = 1  
    ActiveCell.Offset(1, 0) = 2  
    ActiveCell.Offset(2, 0) = 3  
End Sub
```

Podstawowe zagadnienia, które należy zapamiętać

W tym podrozdziale przedstawimy kilka dodatkowych zagadnień, których opanowanie jest niezbędne dla każdego, kto chce na poważnie zajmować się programowaniem w języku VBA. Poszczególne zagadnienia staną się bardziej zrozumiałe w miarę zdobywania doświadczenia i czytania kolejnych rozdziałów.

- **Obiekty posiadają unikatowe właściwości i metody** — Każdy obiekt dysponuje własnym zestawem właściwości i metod. Jednak niektóre obiekty mają wspólne niektóre właściwości (np. Name) i metody (np. Delete).
- **Możesz manipulować obiektami bez ich zaznaczania** — Może to być niezgodne ze standardowym wyobrażeniem przetwarzania obiektów w Excelu. W praktyce jednak wykonywanie operacji na obiektach bez ich uprzedniego zaznaczenia jest zazwyczaj efektywniejsze. Kiedy rejestrujesz nowe makro, Excel najpierw zaznacza obiekt, a dopiero potem wykonuje na nim operacje. Nie jest to konieczne i właściwie spowalnia działanie makra.

- **Powinieneś dokładnie zrozumieć koncepcję i zasady pracy z kolekcjami** — W większości przypadków odwołujesz się do obiektu w sposób pośredni za pośrednictwem kolekcji, w której dany obiekt się znajduje. Aby odwołać się na przykład do obiektu klasy `Workbook` o nazwie `Mój_plik`, powinieneś użyć następującego odwołania do kolekcji `Workbooks`:

```
Workbooks("Mój_plik.xlsx")
```

Powyższe odwołanie zwraca obiekt, który jest żądanym przez Ciebie skoroszytem.

- **Właściwości mogą zwracać odwołanie do innego obiektu** — Przykładowo w poniższej instrukcji właściwość `Font` zwraca obiekt `Font` zawarty w obiekcie `Range`. `Bold` to właściwość obiektu `Font`, a nie obiektu `Range`.

```
Range("A1").Font.Bold = True
```

- **Istnieje wiele różnych metod odwoływania się do tego samego obiektu** — Załóżmy, że dysponujesz skoroszytem o nazwie `Sprzedaż` i jest to jedyny otwarty w danej chwili skoroszyt. Przyjmijmy też, że skoroszyt zawiera jeden arkusz o nazwie `Zestawienie`. Do arkusza można odwołać się przy użyciu następujących sposobów:

```
Workbooks("Sprzedaż.xlsx").Worksheets("Zestawienie")
Workbooks(1).Worksheets(1)
Workbooks(1).Sheets(1)
Application.ActiveWorkbook.ActiveSheet
ActiveWorkbook.ActiveSheet
ActiveSheet
```

To, która metoda zostanie wybrana, przeważnie zależy od wiedzy użytkownika na temat obszaru roboczego. Jeżeli na przykład otwartych jest więcej skoroszytów niż jeden, nieodpowiednia będzie druga lub trzecia metoda. Jeżeli używasz aktywnego arkusza (dowolnego typu), każda z trzech ostatnich metod będzie skuteczna. Aby mieć całkowitą pewność, że odwołujesz się do właściwego arkusza określonego skoroszytu, zastosuj pierwszą metodę.

Kilka słów o przykładach kodu

W książce znajdziesz wiele fragmentów kodu źródłowego języka VBA, które mają na celu zilustrowanie danego zagadnienia czy problemu. Bardzo często takie przykłady kodu mogą składać się tylko z jednego polecenia. W niektórych przypadkach cytowany przykład składa się tylko z pojedynczego wyrażenia, które samo w sobie nie jest jeszcze poprawnym poleceniem języka VBA.

Na przykład kod przedstawiony poniżej jest wyrażeniem:

```
Range("A1").Value
```

Aby przetestować wyrażenie, musisz je obliczyć (wykonać). W tym przypadku dobrym rozwiązaniem będzie użycie funkcji `MsgBox`:

```
MsgBox Range("A1").Value
```

Jeżeli chcesz samodzielnie wypróbować podane przykłady, powinieneś umieścić polecenie w procedurze utworzonej w module kodu VBA, na przykład:

```
Sub Test()  
    ' tutaj wstaw testowane polecenie  
End Sub
```

Następnie umieść kursor w dowolnym miejscu kodu procedury i naciśnij klawisz *F5* aby ją wykonać. Dodatkowo powinieneś się upewnić, czy kod procedury jest wykonywany we właściwym kontekście. Na przykład: jeżeli testowane polecenie odnosi się do arkusza *Arkusz1*, powinieneś upewnić się, czy aktywny skoroszyt faktycznie zawiera arkusz o takiej nazwie.

Jeżeli kod źródłowy składa się tylko z jednej instrukcji, możesz użyć okna *Immediate* edytora Visual Basic. Okno to jest bardzo przydatne w przypadku natychmiastowego wykonywania instrukcji bez konieczności tworzenia procedury. Jeżeli okno *Immediate* nie jest widoczne, w edytorze Visual Basic powinieneś nacisnąć kombinację klawiszy *Ctrl+G*.

Po wprowadzeniu instrukcji w oknie *Immediate* wystarczy nacisnąć klawisz *Enter*. Aby obliczyć (wykonać) dane wyrażenie, powinieneś na jego początku wstawić znak zapytania (?), który pełni funkcję odpowiednika metody *Print*. Na przykład w oknie *Immediate* możesz wprowadzić następujące wyrażenie:

```
? Range("A1").Value
```

Wynik wyrażenia zostanie wyświetlony w następnym wierszu okna *Immediate*.

Nie panikuj — nie jesteś sam

Jeżeli po raz pierwszy masz styczność z językiem VBA, prawdopodobnie będziesz trochę przytłoczony mnogością obiektów, właściwości i metod. To zupełnie normalne. Nikt nie staje się ekspertem programowania w języku VBA w ciągu jednego dnia. Nabieranie doświadczenia w programowaniu VBA to kwestia czasu i praktyki. Dobra wiadomość jest taka, że nie jesteś pierwszą osobą, która przez to przechodzi. Do dyspozycji masz ogromną ilość różnych źródeł, w których z pewnością znajdziesz odpowiedzi na nurtujące Cię pytania. W tym podrozdziale przedstawimy kilka dobrych sposobów zdobycia informacji, które pomogą Ci skierować się we właściwą stronę.

Przeczytaj resztę książki

Nie zapomnij o tym, że tytuł tego rozdziału brzmi „Wprowadzenie do języka VBA”. W pozostałej części książki znajdziesz wiele dodatkowych szczegółów oraz przydatnych i pouczających przykładów.

Pozwól Excelowi napisać makro za Ciebie

Bez wątpienia najlepszą metodą zaznajomienia się z językiem VBA jest po prostu uaktywnienie rejestratora makr i zapisanie kilku operacji wykonywanych w Excelu. Po zakończeniu rejestrowania makra możesz przeanalizować wygenerowany kod i spróbować jeszcze lepiej dostosować go do swoich potrzeb.

Na przykład założmy, że potrzebujesz makra, które odświeża wszystkie tabele przestawne w skoroszycie i resetuje ustawienia wszystkich filtrów tabel przestawnych. Napisanie takiego makra od zera byłoby dosyć trudnym zadaniem. Zamiast tego możesz po prostu uruchomić rejestrator makr i za jego pomocą zarejestrować, jak ręcznie odświeżasz wszystkie tabele przestawne i resetujesz ich filtry. Po zatrzymaniu rejestratora makr możesz przeanalizować otrzymany kod VBA i ewentualnie dokonać niezbędnych modyfikacji.

Korzystaj z systemu pomocy

Początkującemu użytkownikowi system pomocy Excela może się wydawać nieprzyjaznym mechanizmem, który w odpowiedzi na pytania użytkownika zwraca oszałamiającą liczbę tematów wydających się nie mieć nic wspólnego z oryginalnym zapytaniem. Prawda jest jednak taka, że kiedy nauczysz się korzystać z tego systemu pomocy, to stanie się on najlepszym i najszybszym źródłem szczegółowych informacji o niuansach programowania w VBA.

Pamiętaj o dwóch podstawowych założeniach systemu pomocy Excela: podczas zadawania pytań lokalizacja kursora w kodzie ma ogromne znaczenie oraz aby skorzystać z systemu pomocy, musisz mieć aktywne połączenie z internetem.

PODCZAS ZADAWANIA PYTAŃ LOKALIZACJA MA ZNACZENIE

Excel tak naprawdę posiada dwa systemy pomocy: pierwszy, który dostarcza informacje na temat funkcji i sposobu działania samego Excela, i drugi, który oferuje pomoc na temat programowania w języku VBA. Zamiast wykonywać globalne wyszukiwanie tematów pasujących do zadanego pytania, Excel stara się ograniczyć kryteria wyszukiwania tylko do takich, które mają sens w kontekście aktualnego położenia kursora. W praktyce oznacza to, że pomoc wyświetlana przez Excela jest kontekstowa i zależy od tego, co w danej chwili robisz. Wynika stąd po prostu, że jeżeli poszukujesz pomocy na temat jakiegoś zagadnienia związanego z makrami i programowaniem w języku VBA, musisz pracować w edytorze VBE, dzięki czemu wyszukiwanie tematów pasujących do zapytania będzie się odbywało we właściwym systemie pomocy.

MUSISZ MIEĆ POŁĄCZENIE Z INTERNETEM

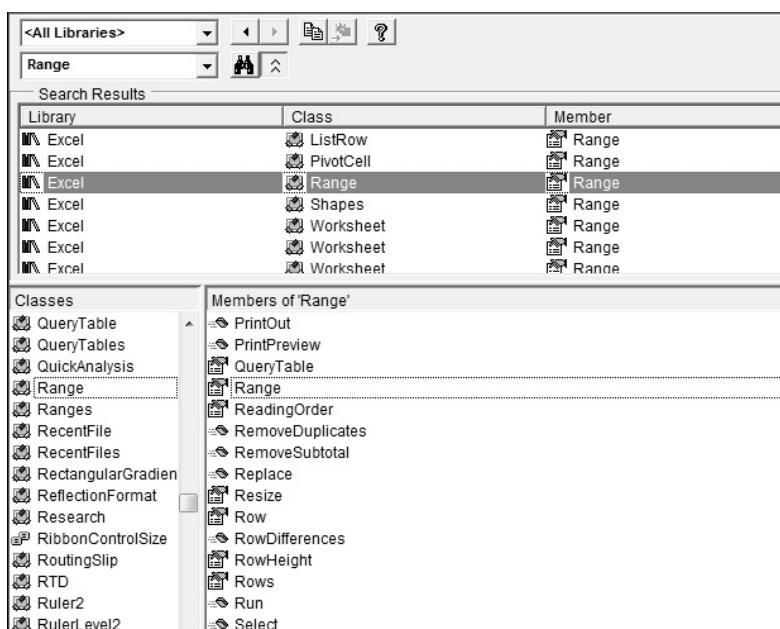
Kiedy poszukujesz pomocy na określony temat, Excel sprawdza, czy jesteś podłączony do internetu. Jeżeli tak, Excel zabiera Cię na stronę sieciową MSDN (Microsoft Developer Network), gdzie możesz przeszukiwać system pomocy online. Jeżeli nie jesteś połączony z internetem, Excel wyświetli na ekranie okno dialogowe z informacją, że aby uzyskać pomoc, musisz się połączyć.

Używaj przeglądarki obiektów

Object Browser, czyli przeglądarka obiektów, jest bardzo przydatnym narzędziem wyświetlającym wszystkie właściwości i metody dostępne dla danego obiektu. Po uaktywnieniu edytora Visual Basic przeglądarkę *Object Browser* można uruchomić na jeden z trzech sposobów:

- Naciśnij klawisz *F2*.
- Wybierz z menu *View* (widok) polecenie *Object Browser* (przeglądarka obiektów).
- Kliknij przycisk *Object Browser* znajdujący się na pasku narzędzi *Standard*.

Okno przeglądarki *Object Browser* zostało przedstawione na rysunku 2.17.



Rysunek 2.17. Przeglądarka Object Browser jest znakomitym źródłem informacji o obiektach

Lista rozwijana w górnym lewym narożniku przeglądarki *Object Browser* zawiera wszystkie dostępne biblioteki obiektów:

- Program Excel.
- *MSForms* (używana do tworzenia niestandardowych okien dialogowych).
- *Office* (obiekty wspólne dla wszystkich aplikacji pakietu Microsoft Office).
- *Stdole* (obiekty automatyzacji OLE).
- VBA.
- Bieżący projekt (wybrany w oknie *Project*) i wszystkie skoroszyty, do których się odwołuje.

Zawartość okna *Classes* zależy od elementu wybranego z listy rozwijanej. Z kolei zawartość okna *Members of* zależy od elementu, który zostanie zaznaczony w oknie *Classes*.

Jeśli po wybraniu biblioteki chcesz uzyskać listę właściwości i metod zawierających określony tekst, możesz go wprowadzić w polu szukania *Search Text* drugiej listy rozwijanej, a następnie kliknąć ikonę *Search* (lornetka).

1. Zaznacz interesującą Cię bibliotekę.

Jeżeli nie jesteś pewien, która biblioteka obiektów jest właściwa, wybierz pozycję *All Libraries*.

2. W polu listy rozwijanej znajdującej się poniżej listy bibliotek wpisz nazwę poszukiwanego obiektu.
3. Aby rozpocząć wyszukiwanie, kliknij ikonę lornetki.

Wyniki wyszukiwania zostaną wyświetlone w oknie *Search Results*. Aby w oknie *Classes* wyświetlić klasy obiektu, zaznacz go na liście. Aby uzyskać listę składników klasy (właściwości, metody i stałe), kliknij nazwę klasy. Zwróć uwagę na dolny panel wyświetlający dodatkowe informacje o obiekcie. Aby bezpośrednio przejść do właściwego tematu pomocy, naciśnij klawisz *F1*.

Na pierwszy rzut oka korzystanie z przeglądarki *Object Browser* może się wydawać trudne, ale z czasem się do niej przyzwyczaisz i docenisz jej przydatność.

Szukaj kodu w internecie

Składnia wszystkich poleceń VBA, których kiedykolwiek będziesz potrzebował, z pewnością już dawno została opublikowana gdzieś w sieci. Pod wieloma względami współczesne programowanie nie polega już tylko na tworzeniu potrzebnego kodu od zera, ale coraz częściej sprowadza się do wyszukiwania potrzebnego kodu w sieci i dostosowywania go do potrzeb określonego zadania.

Jeżeli utknąłeś podczas pisania makra wykonującego takie czy inne zadanie, po prostu uruchom swoją ulubioną przeglądarkę sieciową i w polu wyszukiwania krótko opisz zadanie, które sprawia Ci problem. Aby osiągnąć lepsze rezultaty, możesz przed opisem problemu umieścić frazę *Excel VBA*.

Jeżeli próbujesz napisać na przykład makro, które będzie usuwało wszystkie puste wiersze w arkuszu, możesz wpisać w wyszukiwarce następujące zapytanie: **excel vba usuwanie pustych wierszy**. Możesz śmiało założyć się o swoją dwumiesięczną pensję, że ktoś w internecie borykał się już kiedyś z takim samym problemem. Co więcej, w dziewięciu przypadkach na dziesięć znajdziesz również gotowy kod rozwiązujący takie zadanie; może on być dla Ciebie znakomitym punktem wyjścia do zastosowania w swoim programie.

Wykorzystuj fora dyskusyjne użytkowników Excela

Jeżeli znajdziesz się w naprawdę poważnych tarapatach, możesz zamieścić swoje pytanie na wybranym forum dyskusyjnym użytkowników Excela, i to nie bez nadziei na szybką, profesjonalną odpowiedź.

Fora dyskusyjne to internetowe grupy użytkowników zajmujących się określonymi zagadnieniami. W takich miejscach możesz zamieścić swoje pytania i oczekiwać, że znajdzie się ekspert lub chociaż bardziej doświadczony użytkownik, który przedstawi propozycję rozwiązania opisanego problemu. Użytkownicy odpowiadający na takich forach to zazwyczaj pasjonaci, którzy na ochotnika podejmują się pomagania innym.

Istnieje wiele forów internetowych zajmujących się tematyką Excela i programowania w VBA. Aby znaleźć takie miejsce, po prostu wpisz w wyszukiwarce sieciowej hasło **Excel forum**.

Poniżej zamieszczamy kilka najważniejszych zasad korzystania z większości forów internetowych:

- Przed rozpoczęciem korzystania z danego forum zawsze przeczytaj i przestrzegaj jego regulaminu. W takich regulaminach często zamieszczane są odpowiedzi na najczęściej zadawane pytania i reguły etykiety danego forum.
- Zamieszczając posty staraj się nadawać im krótkie tematy związane z problemem. Nigdy nie umieszczaj na forach postów zatytułowanych *Potrzebna pomoc* czy *Mam pytanie*.
- Staraj się zadawać precyzyjnie sformułowane pytania, dotyczące konkretnego problemu. Nie zadawaj pytań w stylu *Jak mam napisać makro podliczające faktury?*
- Bądź cierpliwy. Pamiętaj, że użytkownicy udzielający odpowiedzi na forach to pasjonaci, którzy oprócz działania na formach mają swoje normalne życie, pracę, rodzinę i dzieci. Daj im trochę czasu na odpowiedź.
- Często zagłądaj na forum. Po zadaniu pytania możesz otrzymać wiele dodatkowych pytań z prośbą o doprecyzowanie problemu bądź dodatkowe informacje. Staraj się ułatwiać innym życie i szybko odpowiadaj na zadawane pytania.
- Kiedy otrzymasz odpowiedź, podziękuj osobie, która Ci ją przesłała oraz wszystkim innym osobom, które były zaangażowane w rozwiązywanie Twojego problemu.

Odwiedzaj blogi ekspertów

Jest wielu ekspertów w dziedzinie Excela, którzy dzielą się swoją wiedzą z innymi za pośrednictwem blogów internetowych. Takie miejsca to prawdziwe kopalnie wiedzy, wskazówek i trików, które może eksplorować każdy, kto chce poszerzyć swoją wiedzę. A co najlepsze, wszystko za darmo!

Choć na takich bloga nie zawsze będzie rozwiązanie Twojego konkretnego problemu, to jednak z całą pewnością znajdziesz tam wiele interesujących artykułów i porad, które pozwolą Ci nabierać nowych doświadczeń i używać Excela w nowych zastosowaniach.

Poniżej zamieszczamy listę kilku najlepszych blogów internetowych, zajmujących się zagadnieniami Excela:

<http://chandoo.org>

<http://www.contextures.com>

<http://www.datapigtechnologies.com/blog>

<http://www.dailydoseofexcel.com>

<http://www.excelguru.ca/blog>

<http://www.mrexcel.com>

Poszukaj szkolenia wideo na YouTube

Niektórzy z nas uczą się lepiej i szybciej, kiedy oglądają film ilustrujący sposób rozwiązania danego problemu. Jeżeli należysz do takich osób, powinieneś poszukać odpowiedniego szkolenia wideo na popularnym serwisie YouTube. Znajdziesz tam dziesiątki kanałów prowadzonych przez pasjonatów chcących podzielić się swoją wiedzą i doświadczeniem. Z pewnością będziesz zaskoczony ilością i jakością zamieszczanych tam materiałów.

Wejdź na stronę <http://youtube.com> i w pasku wyszukiwania wpisz **Excel VBA**.

Ucz się z Microsoft Office Dev Center

Microsoft Office Dev Center to serwis internetowy pomagającym nowym deweloperom rozpocząć karierę w programowaniu produktów pakietu Office. Część tego serwisu, poświęconą Excelowi znajdziesz pod adresem <https://msdn.microsoft.com/en-us/library/office/fp179694.aspx>.

Choć poruszanie się po stronach Microsoft Office Dev Center może na pierwszy rzut oka nie być takie proste, to jednak warto poświęcić chwilę na zapoznanie się z nim. Znajdziesz tam bardzo wiele różnych zasobów, włącznie z gotowymi fragmentami kodu, narzędziami, instrukcjami krok po kroku i innymi.

Analizuj inne aplikacje Excela, które są używane w Twojej organizacji

Inne aplikacje Excela używane w Twojej organizacji mogą być dla Ciebie prawdziwą kopalnią wiedzy, pomysłów i rozwiązań. Jeżeli znajdziesz taki plik, postaraj się zajrzeć mu „pod maskę” i przeanalizować zawarte w nim makra. Przekonaj się, jak inni użytkownicy piszą swoje makra. Spróbuj przeanalizować ich kod wiersz po wierszu i postaraj się wyłapać różne interesujące sztuczki i techniki programowania. Być może natkniesz się nawet na całe fragmenty kodu, które będziesz mógł wykorzystać w swoich zastosowaniach.

Zapytaj lokalnego guru

Czy masz w swojej organizacji jakiegoś lokalnego guru zajmującego się Excelem? Postaraj się z nim poznać i zaprzyjaźnić. Większość ekspertów uwielbia dzielić się swoją wiedzą z innymi. Nie bój się podejść lub zadzwonić do takiej osoby i poprosić o pomoc w rozwiązaniu trapiącego Cię problemu.

Skorowidz

A

- Accelerator, 537
- Access, 435
- Activate, 192, 252, 386, 413, 572
- ActiveCell, 83
 - ClearContents, 84
 - Offset, 92
- ActiveChart, 84, 387, 389
 - Name, 387
- ActivePrinter, 354
- ActiveSheet, 84
 - Cells, 89
- ActiveWindow, 84, 577
 - DisplayGridlines, 122
 - DisplayHeadings, 319
- ActiveWorkbook, 84
 - FullName, 84
 - Path, 453
- ActiveX, 40
- AddChart, 383
- AddIn, 687
 - Comments, 688
 - FullName, 688
 - Installed, 689
 - Name, 687
 - Path, 687
 - Title, 688
 - zdarzenia, 691
- AddInInstall, 252, 691, 694
- AddIns, 666, 675, 679, 685
 - Add, 686
- AddIns2, 685
- AddInUninstall, 252, 691
- AddItem, 580
- Additional Controls, 560
- Address, 86
- ADO, ActiveX Data Objects, 466
 - pobieranie danych zewnętrznych, 466
- AfterCalculate, 270
- AfterPrint, 257
- AfterSave, 252
- AfterUpdate, 552
- aktualizacja
 - aplikacji, 49
 - nagłówka, 256
 - stopki, 256
- aktualna rozdzielczość karty graficznej, 355
- aktywacja wykresu, 386
- aktywna komórka, 83
- aktywne okno, 84
- aktywny
 - arkusz, 84
 - arkusz wykresu, 84
 - skoroszyt, 84
- algorytm sortowania, 325
- AllBold, 335
- analiza funkcji niestandardowej, 201
- animacja etykiet, 605
- API, Application Programming Interface, 238
- API call, 351
- aplikacje
 - arkusza kalkulacyjnego, 31
 - dla wielu wersji narodowych, 813
 - Application.International, 817
 - daty i czas, 819
 - identyfikacja ustawień systemu, 817
 - język aplikacji, 813
 - kody języków, 814
 - obsługa języka w kodzie VBA, 816
 - problemy, 813
 - właściwości lokalne, 816
- AppActivate, 455
- Application, 128, 165
 - ActiveCell, 83
 - ActiveChart, 84
 - ActivePrinter, 354
 - ActiveSheet, 84
 - ActiveWindow, 84
 - ActiveWorkbook, 84
 - Calculation, 692
 - CommandBars, 740
 - Dialogs, 515
 - DisplayCommentIndicator, 765
 - EnableCancelKey, 192, 573
 - EnableEvents, 247
 - GetOpenFilename, 510

Application

- GetSaveAsFilename, 513
- Goto, 514
- Help, 778
- International, 813, 817
- MacroOptions, 233
- PathSeparator, 328, 811
- Run, 161, 682
- ScreenUpdating, 191
- Selection, 84
- SendKeys, 733
- StatusBar, 615
- ThisWorkbook, 84
- Transpose, 581
- Version, 696
- WorksheetFunction, 827
- Application.CommandBars, 517
- Application.WorksheetFunction, 128
- Areas, 295
- argumenty, 154, 170
 - funkcje, 209
 - literały, 171
 - nieokreślona liczba argumentów, 224
 - opcjonalne, 218
 - przekazywanie, 170
 - tablice, 217
- Argumenty funkcji, 235
- arkusze
 - synchronizacja arkuszy, 318
 - system pomocy w aplikacjach, 767
 - ukrywanie wszystkich komórek arkusza poza zaznaczonym zakresem, 315
 - wykresu, 380, 384
- Array, 220
- ArrayFillRange, 307
- As, 114, 172, 202
- Asc, 827
- Atn, 827
- Auto Data Tips, 77
- Auto Indent, 77
- Auto List Members, 77, 117, 127, 790
- Auto Quick Info, 77
- Auto Syntax Check, 77, 105
- automatyczne
 - sprawdzanie składni, 105
 - tworzenie menu podręcznego, 756
 - usuwanie menu podręcznego, 756
- automatyzacja, 435
 - CreateObject, 434
 - GetObject, 434
 - odwołanie do obiektu, 434
 - zadań, 431

B

- baza danych Access, 435, 461
- BeforeClose, 252, 257
- BeforeDoubleClick, 260, 267, 413
- BeforeDragOver, 552
- BeforeDropOrPaste, 552
- BeforePrint, 250, 252, 256
- BeforeRightClick, 260, 268
- BeforeSave, 244, 252, 255
- BeforeUpdate, 552
- BeginGroup, 743
- bezpieczeństwo, 36
 - hasła, 46
- biblioteka
 - funkcji, 200, 235
 - obiektów, 432
 - obiektów ADO, 469
- biblioteki
 - DLL, 238, 351
 - obiektów, 164
- bieżąca data, 149
- blokowanie
 - obiektów arkusza, 46
 - wybranych komórek, 45
- błąd, 43
 - #ARG!, 231
 - #DZIEL/0!, 229
 - #N/D!, 223
 - #NAZWA?, 200
- błędy
 - formuł Excela, 223
 - składni, 77, 173
 - użytkownika, 45
 - wykonania, 173
- Boolean, 109
- BoundColumn, 592
- Break on All Errors, 174
- Break on Unhandled Errors, 174
- Button_Click, 241
- ByRef, 172
- Byte, 109
- ByVal, 172

C

- Calculate, 244, 260, 414
- Calculation, 692
- Call, 161, 165
- callback procedures, 705
- CallByName, 827
- Caption, 575, 616, 740, 742
- Case, 140
- CBool, 827

- CByte, 827
 - CCur, 827
 - CDate, 827
 - CDBl, 827
 - CDec, 110, 827
 - Cell, 739–741, 748, 750
 - Cells, 89
 - Change, 260, 552–554, 631
 - Chart, 381, 387, 417, 657
 - Export, 398
 - ChartObject, 85, 381, 394
 - ChartObjects, 390
 - Delete, 390
 - Charts, 384, 390
 - Add, 384
 - ChartTitle, 382
 - CheckBox, 40, 524, 713
 - CHM, 762
 - Choose, 827
 - Chr, 827
 - Chronienie arkusza, 45
 - Chroń skoroszyt, 46
 - ciąg połączenia, 466
 - Data Source, 467
 - Extended Properties, 467
 - Password, 467
 - Provider, 467
 - User ID, 467
 - CInt, 827
 - Class Module, 417, 652
 - Click, 629
 - CLng, 827
 - CloseAllWorkbooks, 315
 - Code, 72
 - maksymalizacja okien, 73
 - minimalizacja okien, 73
 - wprowadzanie kodu źródłowego, 73
 - Code Colors, 78
 - Collection
 - NoDupes, 583
 - ColorNegative2, 298
 - COM, Component Object Model, 669
 - ComboBox, 525
 - CommandBar, 515, 517, 737
 - BeginGroup, 743
 - BuiltIn, 743
 - Caption, 742
 - Enabled, 743
 - FacelID, 743
 - ID, 742
 - OnAction, 743
 - Picture, 743
 - Reset, 748
 - ToolTipText, 743
 - Type, 738, 743
 - Visible, 743
 - właściwości formantów, 742
 - CommandBars, 736, 740
 - ExecuteMso, 515, 731
 - FindControl, 742
 - GetEnabledMso, 730, 731, 732
 - GetImageMso, 731, 732
 - GetLabelMso, 731
 - GetPressedMso, 731
 - GetScreentipMso, 731
 - GetSupertipMso, 731
 - Name, 740
 - CommandButton, 40, 167, 525, 566, 573
 - Comment Block, 107
 - Comments, 688
 - Const, 116
 - Controls, 558
 - ControlTipText, 643, 769
 - Copy, 283
 - CopyRange, 283
 - Cos, 827
 - Count, 294
 - COUNTA, 301, 548
 - CreateChartSheet, 385
 - CreateObject, 434, 827
 - CreatePivotTable, 365, 371
 - CSng, 827
 - CStr, 827
 - CSV, Comma Separated Values, 271
 - CurDir, 828
 - Currency, 91, 109
 - CurrentRegion, 286
 - Select, 287
 - CustomUI, 729
 - Cut, 284
 - CVar, 828
 - CVDate, 828
 - CVErr, 223, 828
 - czas, 813
 - aplikacje dla wielu wersji narodowych, 819
 - wyświetlanie, 320
 - czcionki, 323
 - CZY.BŁĄD, 303
 - CZY.LICZBA, 217
 - CZY.LOGICZNA, 303
 - CZY.TEKST, 303
- ## D
- dane
 - zaimportowane, 460
 - zewnętrzne, 457
 - DAO, Data Access Object, 436
 - DATA, 119

- data
 - wydrukowania pliku, 335
 - zapisania pliku, 335
- DataPivotField.Orientation, 371
- Date, 109, 118, 149, 320, 828
- DateAdd, 828
- DateAndTime, 320
- DateDiff, 828
- DatePart, 828
- DateSerial, 119, 149, 819, 828
- DateValue, 828
- daty, 118, 229, 813
 - wyświetlanie, 320
- Day, 828
- DDB, 828
- Deactivate, 252, 255, 260, 414
- deaktywacja wykresu, 389
- Debug, 232
- Debug.Print, 185, 232
- Debug/Compile, 674
- debugging, 77
- Decimal, 109, 110
- Declare, 238
- Default to Full Module View, 78
- DefaultPrinterInfo, 354
- definiowanie
 - kategorii funkcji, 235
 - typów danych, 108
- deklaracja, 71
 - funkcji, 204
 - funkcji interfejsu API, 239, 351
 - procedury Sub, 154
 - stałych, 116
 - tablic, 123
 - dynamicznych, 124
 - wielowymiarowych, 123
 - zestawu rekordów, 468
 - zmiennych, 104, 110, 112
- Delete, 390, 391
- DeleteEmptyRows, 300
- Dialogs, 515
- Dim, 113, 114, 124, 126
- Dir, 329, 828
- DisplayCommentIndicator, 765
- DisplayGridlines, 122
- DisplayVideoInfo, 355
- DLL, 238, 351
- Do Loop, 150
- Do Until, 150, 151, 607
- Do While, 148
- dodatki, 665, 669
 - AddIn, 687
 - arkusze, 680
 - dodawanie elementów do kolekcji AddIns, 685
 - dostęp do procedur VBA, 681
 - dystrybucja, 677
 - instalacja, 676
 - korzystanie z dodatku jak ze skoroszytu, 691
 - lista kontrolna tworzenia, 677
 - menedżer dodatków, 669, 675
 - modyfikacja, 677
 - nazwa pliku, 687
 - nazwy, 688
 - odwołania do plików, 695
 - opis, 674
 - optymalizacja wydajności, 692
 - podglądanie zabezpieczonego dodatku, 683
 - problemy, 693
 - przechowywanie funkcji niestandardowych, 237
 - przetwarzanie dodatków za pomocą kodu VBA, 685
 - ścieżka dostępu do pliku dodatku, 687
 - testowanie, 677
 - tworzenie, 671, 672, 674
 - usuwanie elementów z kolekcji AddIns, 687
 - wykresy, 680
 - wykrywanie właściwej wersji Excela, 696
 - zastosowanie, 667
 - zdarzenia, 691
- dodawanie
 - elementu do menu podręcznego Cell, 750
 - formantów do formularza UserForm, 523
 - makra do paska narzędzi, 67
 - modułu, 71
 - obiektów do klasy, 801
 - odwołania do pliku biblioteki obiektów, 433
 - podmenu do menu podręcznego, 753
 - poła wyboru do karty, 713
 - przycisków do karty, 705
 - przycisków do paska narzędzi, 703
- DoEvents, 607, 828
- dokumentacja, 48
 - prac projektowych, 48
- dołączanie rekordów, 473
- dostęp
 - do pliku
 - binarny, 474
 - losowy, 474
 - sekwencyjny, 474
 - do poleceń Wstążki, 730
 - do rejestru systemu Windows, 358
- dostosowywanie
 - edytora VBE, 76
 - menu podręcznego, 38, 746
 - okna Toolbox, 558
 - Wstążki, 38, 700
- Dostosuj pasek narzędzi Szybki dostęp, 518
- Double, 109
- Drag-and-Drop Text Editing, 77

DrawMenuBar, 641
 drukarka domyślna, 354
 drukowanie
 aktualizacja nagłówka lub stopki, 256
 ukrywanie kolumn przed wydrukiem, 256
 wykresów osadzonych, 421
 DupeRows, 302
 Dynamic Link Library, 351
 dynamicMenu, 724
 dynamiczna zmiana położenia formantów formularza
 UserForm, 638
 dynamiczne połączenia danych, 463
 dystrybucja dodatków, 677

E

Each...Next, 131
 edytor VBE, 54, 68, 76
 Auto Data Tips, 77
 Auto Indent, 77
 Auto List Members, 77
 Auto Quick Info, 77
 Auto Syntax Check, 77
 błędy składni, 77
 Code, 72
 Code Colors, 78
 Custom UI, 708
 czcionki, 78
 Default to Full Module View, 78
 Docking, 80
 dostosowywanie, 76
 Drag-and-Drop Text Editing, 77
 Editor Format, 78
 Immediate, 69
 informacje o argumentach funkcji, 77
 karta Editor, 76
 kod źródłowy języka VBA, 69
 kopiowanie i przenoszenie tekstu, 77
 lista funkcji VBA, 127
 Margin Indicator Bar, 79
 okno kodu źródłowego, 72
 pasek menu, 69
 paski narzędzi, 69
 pomoc w trakcie wprowadzania kodu źródłowego, 77
 Procedure Separator, 78
 Project Explorer, 69, 70
 Properties, 531
 Require Variable Declaration, 77
 rozmiar czcionki, 79
 VBA, 24
 wcięcia, 77
 edytor Word, 438
 wprowadzanie kodu źródłowego, 73
 efekt podświetlenia okna dialogowego, 659

eksportowanie
 obiektów graficznych, 398
 wykresów, 398
 zakresu do pliku tekstowego, 479
 elementy języka VBA, 103
 Else, 136–138
 emulowanie
 funkcji MsgBox, 633
 kod funkcji, 635
 MyMsgBox, 638
 okien dialogowych Excela, 562
 EnableCancelKey, 192, 573
 EnableEvents, 247
 End, 115, 428
 End Function, 199, 202, 205
 End Select, 142
 End Sub, 94, 154, 155
 End Type, 126
 Enter, 552, 553
 Environ, 828
 EOF, 828
 Eqv, 122
 Err, 174–177
 Number, 174
 Error, 174, 552, 828
 etykiety, 526
 danych, 407
 event handler procedure, 168
 Excel
 eksport plików tekstowych, 477
 import plików tekstowych, 477
 przesyłanie wykresów, 443
 sterowanie bazą danych Access, 435
 sterowanie edytorem Word, 438
 sterowanie programem Outlook, 446
 sterowanie programem PowerPoint, 442
 uruchamianie innych aplikacji, 451
 ExecuteMso, 515, 731
 Exit, 552
 Exit For, 133, 145, 147
 Exit Sub, 155, 177, 292
 Exp, 828
 EXTRACTELEMNT, 342

F

FaceID, 737, 743, 755
 False, 319
 FileAttr, 828
 FileDateTime, 828
 FileDialog, 514
 FileExists, 328
 FileLen, 828
 FileNameOnly, 328
 FileNameOnly2, 328

- FileSystemObject, 488
 - Drive, 489
 - File, 489
 - Folder, 489
 - TextStream, 489
- FILLCOLOR, 335
- Filter, 828
- filtrowanie zawartości
 - listy, 598
 - pliku tekstowego, 482
- FindControl, 742
- FindExecutableA, 353
- FindWindowA, 641
- Fix, 828
- fmListStyleOption, 593
- fmMultiSelectMulti, 593
- fmTabStyleButtons, 602
- fmTabStyleNone, 602, 628
- folder Modules, 72
- FollowHyperlink, 260
- Font, 532
- For Each...Next, 132, 178, 298
- For...Next, 104, 144, 188
 - Step, 145
- formant, 40, 524
 - ActiveX, 40, 164, 167, 528
 - CheckBox, 524, 550
 - ComboBox, 525
 - CommandButton, 525, 544, 566
 - Controls, 558
 - DynamicMenu, 724
 - etykiety, 526
 - Frame, 525, 543
 - Image, 410, 526, 712
 - kontener, 525
 - Label, 526, 543, 605, 648
 - ListBox, 526, 567, 568, 577, 598, 625
 - MHTML, 773
 - MultiPage, 526, 601, 620, 622, 628
 - obrazy, 526
 - OptionButton, 526, 544
 - pasek przewijania, 527
 - pokrętko, 527
 - pole grupy, 525
 - pole kombi, 525
 - pole listy, 526
 - pole tekstowe, 527
 - pole wyboru, 524
 - przycisk opcji, 526
 - przycisk polecenia, 525
 - przycisk przełącznika, 528
 - RefEdit, 527, 568
 - ScrollBar, 527, 575, 654
 - SpinButton, 527, 553
 - stosowanie w arkuszu, 528
 - TabStrip, 527, 601
 - TextBox, 527, 543, 605
 - ToggleButton, 528
 - ViewCustomViews, 730
 - wspólne właściwości, 533
 - wstawianie, 523
 - wyrównanie formantów, 530
- formanty
 - definiowanie klawiszy skrótu, 537
 - formularza, 41, 167, 528
 - kolejność tabulacji, 535
 - modyfikacja właściwości, 531
 - Wstążki, 716
 - Windows Media Player, 602
 - zewnętrzne, 602
- format
 - CSV, 271, 477
 - HTML, 772
 - PRN, 477
 - TXT, 477
 - ZIP, 491
- Format, 828
- FormatCurrency, 828
- FormatDateTime, 828
- FormatNumber, 828
- formatowanie komórki, 333, 517
- FormatPercent, 828
- Formuła, 91, 92
- FormulaArray, 91
- FormulaLocal, 91
- FormulaR1C1, 91, 92
- formularze danych, 517
- formularze UserForm, 40, 410, 497, 521, 565
 - Activate, 549
 - animacja etykiet, 605
 - automatyczna aktualizacja, 613
 - Caption, 616
 - ControlTipText, 769
 - definiowanie klawiszy skrótu, 537
 - dodawanie formantów, 523
 - dodawanie procedur obsługi zdarzeń, 547
 - formanty, 523
 - formanty zewnętrzne, 602
 - formularze bez paska tytułowego, 640
 - Initialize, 549
 - kod procedury wyświetlającej okno dialogowe, 545
 - kreatory, 626
 - ładowanie formularza, 539
 - menu, 566
 - modyfikacja formantów, 529
 - możliwość zmiany rozmiaru, 646
 - MultiPage, 601
 - Niemodalne okna dialogowe, 610

- obsługa wielu przycisków, 651
- odwołanie do formantów, 556
- okno powitalne, 570
- poker, 661
- powiększanie arkusza, 575
- półprzezroczyste formularze UserForm, 657
- procedury obsługi zdarzeń, 539, 547
- przewijanie arkusza, 575
- samodzielny wskaźnik postępu zadania, 616
- symulacja paska narzędzi, 642
- system pomocy w aplikacjach, 768
- szablony, 561
- Tag, 556
- testowanie, 537, 545, 563
- tworzenie, 522, 542, 563
- układanka, 660
- ukrywanie, 541
- wskaźnik postępu zadania, 614
- wstawianie formularza, 522
- wybór koloru, 654
- wykresy, 410, 656
- wyłączanie przycisku Zamknij, 573
- wyświetlanie formularza, 537
- wyświetlanie formularza na podstawie zmiennej, 539
- wyświetlanie niemożliwych okien, 538
- wyświetlanie wykresu, 656
- zamykanie formularza, 540
- zaznaczanie zakresów, 568
- zdarzenia, 539, 549, 550
- zmiana położenia formantów, 638
- zmiana wielkości formularza, 574
- formuła
 - SERIE, 400, 404
 - tablicowa, 220
- formuły formatowania warunkowego, 207
- Frame, 525, 543, 569, 617
- FreeFile, 828
- FullName, 84, 688
- Function, 71, 198, 199, 204, 233
- funkcja, 25, 71, 98
- funkcja
 - Abs, 827
 - ALLBOLD, 335
 - AreaType, 295
 - Array, 220, 827
 - ArrayFillRange, 307
 - Asc, 827
 - Atn, 827
 - BUBBLESIZE_FROM_SERIES, 405
 - CallByName, 827
 - CBool, 827
 - CByte, 827
 - CCur, 827
 - CDate, 827
 - CDbl, 827
 - CDec, 110, 827
 - Choose, 827
 - Chr, 827
 - CInt, 827
 - CLng, 827
 - Commission, 215
 - Cos, 827
 - COUNTA, 301, 548
 - COUNTBETWEEN, 338
 - CreateObject, 434, 827
 - CSng, 827
 - CStr, 827
 - CurDir, 828
 - CVar, 828
 - CVDate, 828
 - CVErr, 223, 828
 - CZY.BŁĄD, 303
 - CZY.LICZBA, 217
 - CZY.LOGICZNA, 303
 - CZY.TEKST, 303
 - DATA, 119
 - Date, 149, 320, 828
 - DateAdd, 828
 - DateDiff, 828
 - DatePart, 828
 - DateSerial, 119, 149, 819, 828
 - DateValue, 828
 - Day, 828
 - DDB, 828
 - Dir, 329, 484, 486, 828
 - DoEvents, 828
 - Environ, 828
 - EOF, 828
 - Error, 174, 828
 - Exp, 828
 - EXTRACTELEMENT, 342
 - FileAttr, 828
 - FileDateTime, 828
 - FileDialog, 514
 - FileExists, 328, 484
 - FileLen, 828
 - FileNameOnly, 328
 - FILLCOLOR, 335
 - Filter, 828
 - Fix, 828
 - Format, 828
 - FormatCurrency, 828
 - FormatDateTime, 828
 - FormatNumber, 828
 - FormatPercent, 828
 - FT, 321, 322
 - FreeFile, 828
 - FV, 828

- funkcja
- GetAllSettings, 829
- GetAttr, 829
- GetExitCodeProcess, 453
- GetExecutable, 353
- GetObject, 434, 829
- GetRegistry, 357
- GetSetting, 655, 829
- GetValue, 331, 332
- Hex, 829
- Hour, 829
- If, 139, 829
- ILE.NIEPUSTYCH, 301, 548
- informacje o formatowaniu komórki, 333
- Input, 829
- InputBox, 289, 498, 829
- InStr, 829
- InStrRev, 829
- Int, 829
- IPmt, 829
- IRR, 829
- IsArray, 829
- IsBold, 334
- IsDate, 303, 829
- IsEmpty, 303, 340, 829
- ISLIKE, 340
- IsError, 829
- IsInCollection, 330
- IsMissing, 219, 221, 829
- IsNull, 829
- ISNUMBER, 217
- IsNumeric, 303, 829
- IsObject, 829
- IsText, 224
- Join, 829
- LASTINCOLUMN, 339, 340
- LASTINROW, 339
- LastSaved2, 336
- LBound, 829
- LCase, 829
- Left, 829
- Len, 829
- LEWY, 218
- LICZ.JEŻELI, 170
- LICZ.WARUNKI, 338
- LITERY.WIELKIE, 127
- LoadPicture, 656
- Loc, 829
- LOF, 830
- Log, 830
- LoopFillRange, 307
- LOS, 170, 212
- LTrim, 830
- MAX, 346
- MAXALLSHEETS, 346
- Mid, 830
- Minute, 830
- MIRR, 830
- Month, 149, 830
- MonthName, 830
- MONTHNAMES, 220
- MsgBox, 104, 122, 184, 504, 830
- MyMsgBox, 636
- MySum, 226
- nieokreślona liczba argumentów, 224
- Now, 830
- NPer, 830
- NPV, 830
- Oct, 830
- OpenProcess Lib, 452
- Partition, 830
- PathExists, 329
- PIERWIASTEK, 128
- Pmt, 830
- pobieranie tablic, 217
- Ppmt, 830
- PV, 830
- QBColor, 830
- RANDOMINTEGERS, 347
- RangeNameExists, 329
- RANGERANDOMIZE, 348
- Rate, 830
- REMOVEVOWELS, 200
- Replace, 830
- RGB, 830
- Right, 830
- Rnd, 830
- Round, 830
- RTrim, 830
- RZYMSKIE, 128
- SaveAllWorkbooks, 314
- SaveSetting, 655
- Second, 830
- Seek, 830
- SERIE, 400, 427
- SERIESNAME_FROM_SERIES, 404
- Shell, 451
- Sgn, 830
- SheetExists, 330
- SheetOffset, 345
- Shell, 451, 830
- Sin, 830
- SLN, 831
- SORTED, 350
- Space, 831
- Spc, 831
- SPELLDOLLARS, 343
- Split, 328, 831

- Sqr, 128, 831
- STATFUNCTION, 344
- Str, 831
- StrComp, 831
- StrConv, 831
- String, 831
- StrReverse, 831
- SUMA, 170, 226, 229
- SUMARRAY, 217
- Switch, 831
- SYD, 831
- Tab, 831
- Tan, 831
- Time, 831
- Timer, 305, 831
- TimeSerial, 831
- TimeValue, 831
- TRANSPONUJ, 220, 309
- TRANSPOSE, 309
- Trim, 831
- tworzenie, 197
- TypeName, 831
- UBound, 831
- UCase, 127, 191, 329, 831
- usuwanie błędów, 231
- Val, 831
- VALUES_FROM_SERIES, 405
- VarType, 831
- wartość zwracana, 202
- Weekday, 140, 831
- WeekdayName, 831
- WorkbookIsOpen, 330
- WriteReadRange, 305
- Wstawianie funkcji, 233
- wywołanie, 206
- XDATE, 230
- XDATEADD, 230
- XDATEDAY, 230
- XDATEDIF, 230
- XDATEMONTH, 230
- XDATEYEAR, 230
- XDATEYEARDIF, 230
- XVALUE_FROM_SERIES, 405
- XVALUES_FROM_SERIES, 405
- Year, 831
- zasięg funkcji, 205
- ZDATEDOW, 230
- zwracanie tablic VBA, 220
- zwracanie wartości błędu, 223
- funkcje
 - argumenty, 209, 215
 - argumenty opcjonalne, 218
 - API, 351
 - arkuszowe, 333
 - bezargumentowe, 210
 - deklaracja, 204
 - Excela, 827
 - interfejsu API, 242
 - jednoargumentowe, 213
 - kategorie, 236
 - niestandardowe, 198
 - opis funkcji, 237
 - osłone, 211, 240
 - przechowywanie funkcji niestandardowych, 237
 - przypisanie tematów pomocy, 779
 - rozszerzone funkcje daty, 229
 - VBA, 827–831
 - w arkuszu, 200
 - w procedurze, 201
 - wielofunkcyjne, 344
 - Windows API, 238
 - DrawMenuBar, 641
 - FindExecutableA, 353
 - FindWindowA, 641
 - GetKeyboardState, 788
 - GetKeyState, 241
 - GetProfileStringA, 354
 - GetRegistry, 357
 - GetSetting, 358
 - GetSystemDirectory, 696
 - GetSystemMetrics, 355
 - GetWindowLong, 641
 - GetWindowsDirectoryA, 240
 - RegCloseKey, 356
 - RegCreateKeyA, 356
 - RegOpenKeyA, 356
 - RegQueryValueExA, 356
 - RegSetValueExA, 356
 - SaveSetting, 358
 - SetWindowLong, 641
 - ShellExecute, 453
 - WriteRegistry, 357
 - wywołanie, 351
 - z wieloma argumentami, 216

G

- galeria zdjęć, 723
- generowanie liczby losowej, 605
- GetAColor, 654, 655
- GetAllSettings, 829
- GetAttr, 829
- getContent, 726
- GetEnabled, 714
- GetEnabledMso, 730, 731, 732
- GetExitCodeProcess, 453
- GetImageMso, 731, 732
- GetKeyboardState, 788
- GetKeyState, 241

GetLabelMso, 731
 getName(), 499
 GetObject, 434, 829
 GetOpenFilename, 509
 GetPressed, 714
 GetPressedMso, 731
 GetProfileStringA, 354
 GetRegistry, 357
 GetSaveAsFilename, 513
 GetScreentipMso, 731
 GetSetting, 358, 655, 829
 GetSupertipMso, 731
 GetSystemDirectory, 696
 GetSystemMetrics, 355
 GetValue1, 289
 GetWindowLong, 641
 GetWindowsDirectoryA, 238, 240, 812
 GIF, 657
 GoTo, 134, 144, 514

H

hasła, 46, 672
 Height, 575, 623
 Hex, 829
 hiperłącza, 317
 HKEY_CLASSES_ROOT, 357
 HKEY_CURRENT_CONFIG, 357
 HKEY_CURRENT_USER, 357
 HKEY_LOCAL_MACHINE, 357
 HKEY_USERS, 357
 Hour, 829
 HTML, 398
 HTML Help, 775
 HTML Help Viewer, 776

I

ID, 742
 identyfikacja
 formantów, 559
 katalogu domowego systemu Windows, 239
 typu zakresów, 286
 zakresu wykresu, 403
 identyfikator
 języka, 813
 tematu pomocy, 777
 If, 132, 135
 If...Then...Else, 136
 If...Then...ElseIf, 138
 ikona
 polecenia menu podręcznego, 755
 przycisku makra, 702
 ILE.NIEPUSTYCH, 301, 548

Image, 410, 526
 Picture, 533, 657
 imageMso, 710, 712, 737
 Immediate, 69, 94, 157, 185, 188
 uruchamianie procedur Sub, 169
 wywołanie funkcji, 209
 Imp, 122
 implikacja logiczna, 122
 importowane
 danych, 458
 danych z pliku tekstowego, 478
 pliku tekstowego do zakresu, 480
 Index, 740
 informacje
 na temat zdarzeń, 549
 o argumentach funkcji, 77
 o drukarce domyślnej, 354
 o formatowaniu komórki, 333
 o języku VBA, 80
 o napędach dysków, 490
 o rozdzielczości karty graficznej, 355
 InitialFilename, 514
 Initialize, 244, 557, 567
 Input, 829
 InputBox, 289, 292, 498, 829
 Insert/Class Module, 786
 instalacja dodatku, 676
 Installed, 689
 InStr, 829
 InStrRev, 829
 instrukcja
 Input #, 478
 Print #, 478
 Write #, 478
 instrukcja
 AppActivate, 455
 Const, 116
 Debug.Print, 232
 Dim, 124
 End Function, 205
 Exit For, 147
 Exit Sub, 292
 GoTo, 144
 Input, 478
 Line Input #, 478
 On Error, 300
 Open, 475
 Option Explicit, 112
 instrukcje
 przypisania, 120
 VBA, 823–826
 Int, 829
 Integer, 109

interakcja z innymi aplikacjami
uruchamianie okien dialogowych

Panelu sterowania, 456

interfejs użytkownika, 23, 37

okna dialogowe, 497

International, 813, 817

IPmt, 829

IRibbonControl, 711

IRibbonUI, 726

IRR, 829

IsAddin, 683

IsArray, 829

IsDate, 303, 829

IsEmpty, 303, 340, 829

IsError, 829

IsItalic, 334

ISLIKE, 340

IsMissing, 219, 221, 829

IsNull, 829

ISNUMBER, 217

IsNumeric, 303, 829

IsObject, 829

IsText, 224

J

jednoczesne tworzenie tabel przestawnych, 373

język

aplikacji, 813

programowania ściśle deklarowany, 108

strukturalny, 144

VBA, 21, 53

Join, 829

K

karta

Docking, 80

Editor Format, 78

General, 79

karty, 23, 558

aktywacja, 733

graficzne, 51

katalogi, 514

kategorie funkcji, 235, 236

KeyDown, 552, 553

KeyPress, 552

KeyUp, 552, 553

klasa, 783, 785

CInvoice, 799

CInvoices, 799

Collection, 583

CSalesRep, 797

CSalesReps, 797

klasy

metody, 792

nadrzędne, 801

stosowanie, 789

tworzenie, 786

właściwości, 790

klawiatura, 24, 553

klawisze skrótu, 39, 241, 275

kliknięcie obiektu, 166

klucze rejestru, 357

kod

„spaghetti”, 144

klawisza, 241, 277, 278

procedur zwrotnych, 708

RibbonX, 705, 709, 721

CustomUI, 714

IRibbonControl, 711

Office 2007 Custom UI Part, 711

Office 2010 Custom UI Part, 711

procedury zwrotne, 711, 714

tworzenie formantów, 717

tworzenie grupy, 717

tworzenie karty, 716

wyświetlanie błędów, 706

XML, 707

źródłowy języka VBA, 24, 69, 104, 534

kolejność operatorów, 121

kolekcja, 81, 82, 130

AddIns, 685, 687

AddIns2, 685

ChartObjects, 390

Charts, 384, 390

CommandBars, 740

Controls, 558

Dialogs, 515

Each...Next, 131

PivotFields, 366

przetwarzanie, 131

Shapes, 383

SparklineGroups, 429

With...End With, 130

Workbooks, 679

kolory, 654

komentarze, 104, 106, 203

kompatybilność aplikacji

64-bitowa wersja Excela, 811

aplikacje dla wielu wersji narodowych, 813

daty i czas, 813

język aplikacji, 813

kreator sprawdzania zgodności, 809

Macintosh, 810

nazwy plików, 811

obsługa języka w kodzie VBA, 816

kompatybilność aplikacji
 problemy ze zgodnością, 806
 tabel przestawnych, 367
 wersje Excela, 808
 komunikaty błędów, 175
 koniunkcja logiczna, 122
 kontekstowe menu podręczne, 758
 kontekstowy interfejs użytkownika, 23
 kontener formantów, 525
 kontrolki
 ActiveX, 42
 formularza, 42
 kontynuacja polecenia w kolejnym wierszu, 104
 konwersja typów danych, 111
 kopiowanie
 nieciągłego zakresu komórek, 312
 zakresów, 282
 zakresu o zmiennej wielkości, 284
 korespondencja seryjna, 439
 kreator, 626
 Dalej, 630
 MultiPage, 628
 programowanie przycisków, 629
 przyciski, 628
 sprawdzania zgodności, 809
 Wstecz, 630
 wykonywanie zadań, 632
 zależności programowe, 631

L

Label, 526, 543, 605, 648
 LanguageID, 813
 LastPrinted, 336
 LastSaved, 335, 336
 LBound, 829
 LCase, 829
 Left, 829
 Len, 829
 LEWY, 218
 LICZ.JEŻELI, 170
 LICZ.WARUNKI, 338
 liczba wierszy arkusza, 339
 Like, 341
 Linie siatki, 122, 699
 lista
 czcionek, 323
 obiektów, 70
 pól tabeli przestawnej, 362
 ListBox, 526, 567, 568, 577, 578
 AddItem, 580
 aktywacja arkusza, 596
 BoundColumn, 592
 ColumnCount, 578

ColumnHeads, 578
 ControlSource, 578
 identyfikacja wielu zaznaczonych elementów, 585
 identyfikacja zaznaczonego elementu listy, 584
 ListIndex, 568, 584
 MultiSelect, 584, 585
 przenoszenie elementów listy, 587
 RowSource, 586, 592
 Selected, 585
 tworzenie listy elementów, 578–582
 Value, 584
 wiele list w jednej kontrolce, 586
 wielokolumnowe formanty, 591
 wybieranie wierszy arkusza, 593
 zmiana kolejności elementów listy, 589
 ListIndex, 568, 584
 listy danych, 517
 literały, 171
 LITERY.WIELKIE, 127
 Load, 539
 LoadPicture, 656
 Loc, 829
 LOF, 830
 Log, 830
 lokalizacja wykresu, 380
 Long, 109
 Loop, 150
 LoopFillRange, 307
 LOS, 170, 212
 losowanie liczb, 347
 LTrim, 830

Ł

ładowanie formularza UserForm, 539
 łańcuchy
 o stałej długości, 118
 o zmiennej długości, 118
 znaków, 118
 łączenie
 łańcuchów, 120
 pliku pomocy z aplikacją, 779

M

Macintosh, 810
 MacroOptions, 233, 236, 780
 Main, 170
 makra, 75
 Accessa, 437
 bezpieczeństwo, 63
 Edytowanie, 57
 Przypisywanie makr, 65
 rejestrator, 53
 rozszerzenia skoroszytów, 63

- Testowanie, 57
- tryb odwołań bezwzględnych, 58
- tryb odwołań względnych, 61
- tworzenie, 54
- Umieszczanie na pasku narzędzi, 67
- wstępne rejestrowanie, 181
- Zapisywanie, 65
- Zaufane lokalizacje, 64
- makro
 - DodajPodsumowanie, 60
 - ReturnToMain, 419
 - SelectCurrentRegion, 287
 - SortSheets, 194
- maksymalna wartość we wszystkich arkuszach, 345
- MAX, 346
- MAXALLSHEETS, 346
- Me, 540
- mechanizm Auto List Members, 534
- menedżer dodatków, 669, 675
- menu, 566
 - Excela 2003, 745, 747
 - podręczne, 287, 737
 - automatyczne tworzenie, 756
 - automatyczne usuwanie, 756
 - Cell, 739, 740, 741, 748, 750
 - CommandBar, 737, 738
 - CommandBars, 740
 - dodawanie elementu, 750
 - dodawanie podmenu, 753
 - dostosowywanie, 38, 746
 - FaceID, 755
 - ikony poleceń, 755
 - jednego skoroszytu, 755
 - odwołania do formantów, 740
 - podmenu, 753
 - resetowanie, 748
 - ukrywanie elementów, 757
 - usuwanie elementu menu podręcznego Cell, 752
 - usuwanie podmenu, 754
 - właściwości formantów obiektu CommandBar, 742
 - wyłączanie, 278, 749
 - wyłączanie elementów, 750, 757
 - wyświetlanie, 738
 - wyświetlanie wszystkich elementów, 743
 - zdarzenia, 756
- metoda, 82, 85, 792
 - Application.InputBox, 500
 - CopyFromRecordset, 471
 - End, 340
 - GetEnabledMso, 731
 - GetOpenFilename, 509
 - GetSaveAsFilename, 513
 - Help, 778
 - Hide, 541
 - InputBox, 292
 - MacroOptions, 233
 - SetElement, 385
- metody
 - Protect, 86
 - obsługi błędów, 173
- MHTML, 773
- Microsoft
 - Excel 16.0 Object Library, 164
 - Forms 2.0 Object Library, 164
 - HTML Help Workshop, 762
 - Office 16.0 Object Library, 164
 - Office Code Compatibility Inspector, 809
 - Office Compatibility Pack, 807
 - Visual Studio Tools for Office, 22
- Mid, 830
- Minute, 830
- MIRR, 830
- Mod, 120
- modalne okna dialogowe, 610
- moduły klas, 246, 417, 783
 - dodawanie kodu VBA, 787
 - niestandardowe, 785
 - programowanie metod, 792
 - programowanie właściwości obiektów, 790
 - tworzenie, 786
 - wbudowane, 784
 - wstawianie modułu klasy, 786
 - wywołanie procedury, 163
 - zdarzenia, 792
 - zmienne, 115
- moduły VBA, 246
- modyfikacja
 - danych wykresu, 402
 - dodatku, 677
 - formantów formularza UserForm, 529
 - kodu języka VBA, 68
 - komórki, 260
 - połączeń, 461
 - właściwości formantów, 531
 - Properties, 531
 - wyrównanie formantów, 530
 - Wstążki, 713, 727
- monitorowanie
 - zdarzeń poziomu aplikacji, 273
 - zmian w wybranym zakresie komórek, 261
- Month, 149, 830
- MonthName, 830
- MouseDown, 414
- MouseMove, 414
- MouseOver, 423
- MouseUp, 414
- MoveRange1, 284
- MP3, 604

MsgBox, 94, 104, 122, 184, 504, 830
 emulacja funkcji, 633
 stałe przycisków, 505
 wartości zwracane, 506
 msoBarTypeMenuBar, 738
 msoBarTypeNormal, 738
 msoBarTypePopUp, 738
 msoLanguageIDUI, 813
 MultiPage, 526, 569, 622
 karty, 602
 kreatory, 628
 Style, 602
 Value, 602
 MultiSelect, 578, 584
 mysz
 zdarzenia, 553

N

Name, 388
 narzędzie PUP, 27
 natychmiastowe zakończenie procedury, 155
 nazwane argumenty, 86
 nazwy
 arkuszy, 84, 182
 plików, 25, 811
 procedur, 155
 skoroszytów, 84
 zakresów, 25
 zmiennych, 107, 112
 negacja logiczna, 122
 New Page, 558
 NewSheet, 244, 252, 254
 NewWorkbook, 244, 270
 Next, 131, 144
 niemodalne
 okna dialogowe, 610
 okna formularzy UserForm, 538
 nieoficjalne systemy pomocy, 762
 nieokreślona liczba argumentów, 224
 nierównoważność logiczna, 122
 niestandardowe
 funkcje arkusza, 203
 funkcje arkuszowe, 333
 menu podręczne, 38, 287
 okna dialogowe, 39, 521
 paski narzędzi, 734
 procedury Function, 204
 typy danych, 126
 NoDupes, 583
 Not, 122, 319
 Nothing, 389, 793
 Now, 830
 NPer, 830

NPV, 830
 Number, 174
 numer pliku, 476
 NumLock, 786

O

obiekt, 81, 124, 130, 785
 AddIn, 687
 Application, 272
 Chart, 381, 417
 ChartObject, 381
 CommandBar, 515, 737
 Err, 174
 FileSystemObject, 488
 QueryTable, 793
 Range, 87
 Recordset, 436, 468
 Sparkline, 429
 SparklineGroup, 429
 ThisWorkbook, 246
 WorksheetFunction, 827
 obiekty
 nadrzędne, 337
 metody, 93
 przypisanie do zmiennej, 124
 typu Chart, 246
 typu Sheet, 246
 typu UserForm, 246
 With...End With, 130
 właściwości, 93
 Object, 109
 Object Browser, 97, 269
 obrazy, 526
 FaceID, 755
 imageMso, 710, 712
 obsługa
 błędów, 36, 173, 175, 294
 Err, 174, 177
 Error, 174
 komunikat błędu, 175
 On Error, 174
 On Error GoTo ErrorHandler, 175
 On Error Resume Next, 174, 178
 przechwytywanie błędów, 174
 języka w kodzie VBA, 816
 trójwymiarowych skoroszytów, 345
 wielu przycisków formularza UserForm, 651
 zdarzenia Button_Click, 241
 zdarzenia Change, 600
 zdarzenia Click, 546
 zdarzeń, 168, 243, 539
 wersje Excela, 246
 ochrona skoroszytu, 46, 191
 Oct, 830

- odczytywanie
 - plików, 478
 - plików tekstowych, 476
 - zakresów, 305
 - zawartości rejestru systemu Windows, 356
 - Odkrywanie, 515
 - odwołania
 - bezwzględne, 58
 - do biblioteki obiektów ADO, 469
 - do formantów formularza UserForm, 556
 - do innego skoroszytu, 163
 - do komórek, 215
 - do obiektów, 94
 - do plików z poziomem dodatku, 695
 - do skoroszytu, 207
 - do zakresów komórek, 288
 - do zakresu, 89
 - względne, 58
 - odwrócone tabele przestawne, 376
 - Office 2007 Custom UI Part, 711
 - Office 2010 Custom UI Part, 711
 - Offset, 92
 - ograniczenia funkcjonalności pasków narzędzi, 734
 - okna dialogowe, 39, 497
 - Formatowanie komórek, 517
 - formularze UserForm, 521
 - modalne okna dialogowe, 610
 - niemodalne okna dialogowe, 610
 - Odkrywanie, 515
 - wyświetlanie, 514
 - okno
 - Code, 72
 - Dodatki, 670
 - eksploratora projektów, 69
 - Immediate, 69, 94, 169
 - kodu, 69
 - Makro, 158, 159
 - mechanizmu Auto List Members, 534
 - Options, 76
 - powitalne, 570
 - Project, 70
 - Properties, 523
 - Toolbox, 523, 524, 558
 - właściwości formantu, 532
 - wprowadzania danych, 498
 - funkcja InputBox, 498
 - metoda InputBox, 500
 - Wstawianie funkcji, 233
 - wybierania katalogu, 514
 - Zmienianie nazwy, 702
 - określanie
 - numeru wersji Excela, 808
 - skojarzeń plików, 352
 - typu danych, 111, 303
 - typu zaznaczonego zakresu, 295
 - wymagań użytkownika, 33
 - OLE Automation, 164
 - On Error, 174, 177, 300
 - On Error GoTo, 294
 - On Error GoTo ErrorHandler, 175
 - On Error Resume Next, 174, 178, 294, 583
 - OnAction, 715, 743
 - OnEntry, 247
 - OnKey, 244, 275–277
 - kody klawiszy, 278
 - OnTime, 244, 274, 572
 - Open, 244, 252
 - operacje na plikach, 478, 483
 - OperatingSystem, 810
 - operatory, 105, 120
 - And, 122
 - Eqv, 122
 - Imp, 122
 - kolejność wykonywania, 121
 - Like, 341
 - logiczne, 122
 - Mod, 120
 - Not, 122, 319
 - Or, 122
 - porównania, 121
 - przypisanie, 120
 - Xor, 122
 - opis funkcji, 237
 - Option Base, 123, 222
 - Option Explicit, 112, 692
 - Option Private Module, 154, 156
 - Optional, 218
 - OptionButton, 40, 526, 531, 544
 - optymalizacja wydajności dodatków, 692
 - Or, 122
 - osobisty arkusz makr, 481
 - ostatnia niepusta komórka, 339
 - Otwieranie, 509
 - otwieranie plików tekstowych, 475
 - Outlook, 446
- P**
- pakowanie plików, 491
 - Panel sterowania, 456
 - ParamArray, 225
 - Parent, 337
 - Partition, 830
 - pasek
 - menu edytora VBE, 69
 - narzędzi Szybki dostęp, 518
 - przewijania, 527
 - stanu, 615

- paski narzędzi, 642, 733
 - edytora VBE, 69
- Path, 453
- PathExists, 329
- PathSeparator, 328
- Personal Macro Workbook, 65
- pętla, 104, 143
 - Do Loop, 150
 - Do Until, 150
 - Do While, 148
 - For Each...Next, 131, 178
 - For ... Next, 144, 148
 - While Wend, 151
- Picture, 532, 656, 737, 755
- PIERWIASTEK, 128
- PivotCache, 371
- PivotCaches, 365
- PivotFields, 365, 366
- PivotItems, 365
- PivotTables, 365
- PivotTableUpdate, 260
- planowanie
 - aplikacji, 35
 - zdarzeń, 274
- plik
 - arkusza, 31
 - excelusage.txt, 482
 - MyDatabaseName.accdb, 471
 - Myfuncs.xlsm, 238
 - personal.xlsb, 65
- pliki
 - CHM, 762
 - CSV, 477
 - GIF, 657
 - HLP, 775
 - HTML, 772
 - MHTML, 773
 - MP3, 604
 - PRN, 477
 - przetwarzanie grupy plików, 326
 - TXT, 477
 - skojarzenia plików, 352
 - ścieżka pliku, 328
 - XLAM, 679
 - XLSM, 679
- pliki tekstowe, 474
 - eksportowanie zakresu, 479
 - filtrowanie zawartości, 482
 - importowanie danych, 478
 - odczytywanie, 476
 - otwieranie, 475
 - zapisywanie danych, 476
- Pmt, 830
- pobieranie
 - danych ze skoroszytów, 472
 - informacji z komórki, 91
 - listy czcionek, 323
 - nazwy pliku, 509
 - wartości z zamkniętego skoroszytu, 331
 - wyznaczonego zakresu, 292
- podglądanie zabezpieczonego dodatku, 683
- pogrubienie zawartości komórek, 262
- Poker, 661
- pokrętło, 527
- pola
 - danych, 364
 - kategorii, 364
 - tabeli przestawnej, 372
- pole
 - grupy, 525
 - kombi, 525
 - listy, 526
 - tekstowe, 424, 527
 - wyboru, 524, 713
- polecenia, 23
 - operacji na plikach, 484
 - starego menu, 517
 - Wstążki, 730
- polecenie
 - End Function, 202
 - For Each ... Next, 131
 - GoTo, 134
 - If, 135
 - Run Sub/UserForm, 158
 - Select Case, 140
 - With ... End With, 125, 130
- połączenia
 - dynamiczne, 463
 - ręczna modyfikacja, 461
 - ręczne tworzenie, 458
 - skoroszytu, 465
 - z bazą danych Access, 466
 - zewnętrzne źródła danych, 458
- pomoc, 96
- ponowne przeliczanie funkcji, 212
- porównania, 121
- porządkowanie zakresu w losowy sposób, 348
- potęgowanie, 121
- Power Utility Pak, 27
- PowerPoint, 442
- powielanie wierszy, 301
- powiększanie arkusza, 575
- półprzezroczyste formularze UserForm, 657
- późne wiązanie, 432, 434
- Ppmt, 830
- Print, 185

- PrintEmbeddedCharts, 421
- Private, 113, 154, 204, 239
- problemy ze zgodnością aplikacji, 806
- procedura, 153
 - AddSubmenu(), 753
 - AddToShortCut(), 750
 - AnimateChart(), 427
 - Append_Results(), 473
 - BatchProcess, 326
 - BubbleSort, 186
 - CalculateCommission(), 802
 - cmdAdd_Click(), 588
 - cmdFinish_Click(), 632
 - cmdOK_Click(), 547
 - cmdOptions_Click(), 575
 - cmdStartStop_Click(), 606
 - cmdUp_Click(), 590
 - ColorNegative, 297
 - CopyMultipleSelection, 312
 - CommanButton1_Click(), 541
 - CommandButton1_Click(), 566
 - ContractAllSeries(), 405
 - ConvertChartToPicture(), 422
 - CopyAllChartsToPresentation(), 444
 - CopyRangeToPresentation (), 442
 - CreateChart, 396, 411
 - CreatePivotTable(), 363, 365, 370
 - CreateShortcut(), 758
 - CreateTOC, 317
 - CreateUnlinkedChart(), 422
 - DataLabelsFromRange(), 409
 - DataLabelsFromRange, 409
 - DateAndTime, 320
 - DescribeFunction, 235
 - DisplayDataForm, 519
 - EmailContactList(), 450
 - EmailRange(), 448
 - EmailWorksheet(), 449
 - EmptyCount(), 556
 - EntryIsValid, 265
 - EraseRange(), 502
 - ExecuteButton_Click(), 567
 - ExportRange(), 479
 - FillContacts(), 599
 - FillInvoices(), 801
 - FillSalesReps(), 801
 - FilterFile(), 482
 - FormatAllCharts, 391–393
 - GenerateRandomNumbers(), 615, 618
 - GetAccessData(), 470
 - GetAFolder, 514
 - GetAnswer(), 506
 - GetData_From_Excel_Sheet(), 472
 - GetImportFileName, 511
 - GetImportFileName2, 512
 - GetValue(), 500
 - GetValue2(), 503
 - GetWord(), 500
 - GetWordVersion(), 435
 - HideRowsAndColumns, 315
 - ImportData(), 479
 - ImportRange(), 480
 - lbxFrom_Change(), 589
 - ListAllAddins(), 689
 - ListConnections(), 465
 - ListFiles(), 485
 - Make50Charts(), 396
 - MakePivotTables(), 374
 - MultiPage1_Change(), 631
 - myChartClass_MouseDown, 420
 - obsługi zdarzenia, 168
 - OKButton_Click(), 555, 585, 598
 - OpenTextFile(), 454
 - ProcessFiles(), 327, 626
 - RecursiveDir, 487
 - RefreshQuery(), 464
 - RemoveDuplicates1(), 583
 - ResetAllShortcutMenus2(), 748
 - RunAccessMacro(), 437
 - RunAccessQuery(), 436
 - SaveAllGraphics, 399
 - SaveAllWorkbooks2, 314
 - scbZoom_Change(), 577
 - SendDataToWord(), 439
 - SendWorkbookToPowerPoint(), 445
 - Separator, 78
 - SetDefaultButton(), 637
 - SetOptions(), 781
 - SheetSort, 188
 - ShowCaptions(), 741
 - ShowChart(), 411
 - ShowDriveInfo(), 490
 - ShowInstalledFonts, 323
 - ShowRange(), 508
 - ShowShortcutMenuItems(), 743
 - ShowShortcutMenuNames(), 738
 - ShowUserForm2(), 580
 - SizeAndAlignCharts(), 394
 - SortSheets, 193
 - SparklineReport(), 429
 - StartEmail(), 455
 - Sub UserForm_Initialize(), 596
 - SynchSheets, 318
 - TextBox1_Change(), 554
 - UnzipAFile(), 493
 - UpdateBox(), 611
 - UpdateChart(), 403
 - UpdateDynamicRibbon(), 726

- procedura
 - UpdateLogFile, 272
 - UserForm_Initialize(), 412, 576, 592, 594
 - WordMailMerge(), 440
 - Workbook_BeforeClose, 258
 - Workbook_Open(), 481, 694, 735
 - Worksheet_Change, 263
 - ZipFiles(), 492
- procedury, 153
 - argumenty, 154
 - Function, 71, 198, 201, 204
 - Main, 170
 - obsługa zdarzeń, 245, 248, 539, 547
 - prywatne, 156
 - Property, 788
 - Property Get, 787, 790
 - Property Let, 788
 - przekazywanie argumentów, 170
 - publiczne, 156
 - Sub, 71, 103, 154, 179
 - deklaracja, 154
 - natychniastowe zakończenie, 155
 - nazwy, 155
 - uruchamianie, 157
 - zasięg, 155
 - wykonywanie po wystąpieniu określonego zdarzenia, 168
 - wywołanie procedury zawartej w innym module, 163
 - wywołanie procedury zawartej w innym skoroszycie, 163
 - zwrotne, 705, 711
- programowanie
 - metod, 792
 - strukturalne, 144
 - w języku VBA, 103, 319
 - właściwości obiektów, 790
 - zaawansowane, 359
- Project Explorer, 69, 70
 - dodawanie modułu, 71
 - Modules, 71
- projekt, 56, 69, 70
- projektowanie aplikacji arkusza kalkulacyjnego, 31
 - bezpieczeństwo, 36
 - dokumentacja, 48
 - dostosowywanie menu podręcznego, 38
 - etapy projektowania, 33
 - formanty ActiveX, 40
 - interfejs użytkownika, 37
 - klawisze skrótu, 39
 - niestandardowe okna dialogowe, 39
 - obsługa błędów, 36
 - określanie wymagań użytkownika, 33
 - planowanie aplikacji, 35
 - struktura danych, 35
 - struktura plików, 35
 - system pomocy, 48
 - wersja Excela, 36, 60
 - wersje językowe, 50
 - wydajność, 36
 - wygląd aplikacji, 47
- Properties, 531
 - modyfikacja właściwości formantów, 531
- Property Get, 787, 790, 791
- Property Let, 791
- Property Set, 791
- Protect, 86
- ProtectStructure, 191
- przechodzenie pomiędzy formantami okna dialogowego, 535
- przechowywanie funkcji niestandardowych, 237
- przechwytywanie błędów, 174
- przeglądarka obiektów, 97
- Przejdź do — specjalnie, 176
- przekazanie aplikacji użytkownikom, 49
- przekazywanie argumentów, 170
 - przez odwołanie, 171
 - przez wartość, 171
- przekształcanie dodatku w skoroszyt, 678
- przeliczanie funkcji, 212
- przełączanie wartości właściwości typu logicznego, 319
- przeniesienie
 - wykresu, 387
 - zakresów, 284
 - zawartości tablic jednowymiarowych, 309
 - zawartości zakresu do tablicy typu Variant, 309
- przetwarzanie
 - arkuszy, 314
 - dat, 118
 - grupy plików, 326
 - kolekcji, 131
 - komórek zaznaczonego zakresu, 297
 - skoroszytów, 314
 - wykresów, 391
 - zakresów, 282, 285
 - znaku kropki, 125
- przewijane
 - etykiety, 770
 - arkusza, 575
 - wykresów, 426
- przezroczystość okna, 658
- przycisk, 628
 - Generate Callbacks, 707
 - opcji, 526
 - polecenia, 525, 545
 - przełącznika, 528
- przyciski dzielone, 698

przypisanie, 104, 120
 obiektu do zmiennej, 124
 Przypisz makro, 167
 Przywróć okno, 73
 Public, 113, 124, 154, 173, 204
 Public WithEvents, 652
 pułapki, 232
 PUP, 27
 puste wiersze, 300
 PV, 830

Q

QBColor, 830
 QueryClose, 551, 573
 quick-sort, 325

R

Range, 87, 88, 94, 124, 303
 Address, 86
 Cells, 89
 RangeNameExists, 329
 RangeNameExists2, 329
 RangeToVariant, 309
 Rate, 830
 Recordset, 468
 adOpenDynamic, 469
 adOpenForwardOnly, 468
 adOpenStatic, 469
 ReDim, 124
 RefEdit, 527, 568, 569
 References, 163, 164
 RegCloseKey, 356
 RegCreateKeyA, 356
 RegOpenKeyA, 356
 RegQueryValueExA, 356
 RegSetValueExA, 356
 rejestr systemu Windows, 356
 odczytywanie zawartości, 356
 zapisywanie zawartości, 356
 rejestrator makr, 53, 181
 wykresy, 381
 rekurencja, 487
 Rem, 106
 Replace, 830
 Require Variable Declaration, 77
 resetowanie menu podręcznego, 748
 Resize, 414
 ReversePivot, 377, 378
 ręczna modyfikacja połączeń, 461
 ręczne tworzenie połączeń, 458
 RGB, 830
 Ribbon, 23
 RibbonX, 705

Right, 830
 Rnd, 830
 Round, 830
 RowSource, 579, 586, 592
 rozdzielczość karty graficznej, 355
 rozpakowywanie plików, 491, 493
 rozszerzenie
 XLA, 666, 675
 XLAM, 675, 679, 680
 XLL, 675
 XLSM, 666, 679, 680
 rozszerzone funkcje daty, 229
 równoważność logiczna, 122
 RTrim, 830
 Run, 161, 165, 682
 Run Sub/UserForm, 157, 158, 207
 rundll32.exe, 456
 RZYMSKIE, 128

S

samodzielny wskaźnik postępu zadania, 615
 formularze UserForm, 616
 SaveAllWorkbooks, 314
 SaveSetting, 358, 655
 Schowek, 699
 ScreenUpdating, 191, 412, 541, 692
 ScrollBar, 40, 527, 575, 577, 654
 ScrollBarZoom, 577
 ScrollColumns, 577
 ScrollRow, 577
 SearchHelp, 777
 Second, 830
 Seek, 830
 sekcja CUSTOM UI, 711
 sekwencje zdarzeń, 245
 Select Case, 139, 244, 287, 321
 SelectByValue, 310
 SelectCurrentRegion, 287
 Selected, 585
 Selection, 84
 Selection.Areas.Count, 295
 Selection.Columns.Count, 295
 Selection.Value, 85
 SelectionChange, 244, 260, 266
 seria danych, 400
 SERIE, 400, 427
 Series, 400
 Values, 403
 SeriesChange, 244, 414
 SeriesCollection, 405
 Set, 124
 SetWindowLong, 641
 Sgn, 830
 Shape, 85, 383, 386, 424

- Shapes, 383
 - AddChart, 383
- SheetActivate, 245, 270, 611, 613
- SheetBeforeDoubleClick, 252, 270
- SheetBeforeRightClick, 252, 270
- SheetCalculate, 252, 270
- SheetChange, 244, 252, 270
- SheetDeactivate, 245, 252, 270
- SheetExists, 330
- SheetFollowHyperlink, 252, 270
- SheetOffset, 345
- SheetPivotTableUpdate, 252, 270
- SheetSelectionChange, 252, 270, 611, 613
- Shell, 451, 830
- ShellExecute, 453, 454
- Show, 537, 539, 571
- ShowDataForm, 519
- Sin, 830
- Single, 109
- skojarzenia plików, 352
- skoroszyt
 - makr osobistych, 65
 - przetwarzanie, 314
 - sprawdzanie połączeń, 465
 - zamykanie wszystkich skoroszytów, 315
 - zapisywanie wszystkich skoroszytów, 314
 - zastosowanie obiektów ADO, 472
 - zdarzenia, 251
- skrót klawiaturowy Ctrl+Shift+Enter, 220
- SLN, 831
- słowa kluczowe, 105
 - As, 114, 172
 - ByRef, 172
 - ByVal, 172
 - Call, 161
 - Case, 140
 - Const, 116
 - Declare, 238
 - Dim, 113, 114, 126
 - Do, 148, 150
 - Else, 136
 - Elseif, 138
 - End, 115
 - For, 144
 - Function, 198, 199, 204
 - GoTo, 134, 144
 - If, 132, 135
 - Loop, 150
 - Me, 534, 540
 - Next, 131, 144
 - Optional, 218
 - ParamArray, 225
 - Private, 113, 154, 156, 204, 239
 - Public, 113, 115, 154, 156, 173, 204
 - ReDim, 124
 - Rem, 106
 - Select, 139
 - Set, 124
 - Static, 113, 116, 154, 204
 - Step, 145
 - Sub, 154
 - Then, 132, 135
 - To, 123
 - Type, 126
 - Until, 150
 - While, 148
 - With, 130
 - WithEvents, 271
- słowa kluczowe zastrzeżone, 108
- SmartArt, 167
- sortowanie, 179, 185
 - arkuszowe, 324
 - bębelkowe, 185, 324
 - metodą quick-sort, 325
 - szybkie, 325
 - tablicy, 324
 - zakresów, 350
 - zliczające, 325
- SortSheets, 193
- SourceData, 400
- Space, 831
- spaghetti, 144
- Sparkline, 429
- SparklineGroup, 429
- sparklines, 428
- Spc, 831
- SpecialCells, 176
- SpecialEffect, 617, 644
- SpinButton, 527, 553
 - TextBox, 553
 - zdarzenia, 551
- SpinDown, 552, 553
- SpinUp, 552, 553
- splash screen, 570
- Split, 328, 831
- sprawdzanie
 - połączeń skoroszytu, 465
 - poprawności danych, 263
 - przynależności obiektu do kolekcji, 330
 - uaktywnienia wykresu, 389
 - występowania katalogu, 484, 490
 - występowania pliku, 484, 490
 - zgodności, 367, 809
- Sqr, 128, 831
- stała, 107, 116
 - fmMultiSelectExtended, 585
 - fmMultiSelectMulti, 585
 - fmMultiSelectSingle, 585

- vbAbort, 506
 - vbAbortRetryIgnore, 505
 - vbCancel, 506
 - vbCritical, 505
 - vbCrLf, 215
 - vbDefaultButton1, 505
 - vbDefaultButton2, 505
 - vbDefaultButton3, 505
 - vbDefaultButton4, 505
 - vbExclamation, 505
 - vbIgnore, 506
 - vbInformation, 505
 - vbMsgBoxHelpButton, 505
 - vbNo, 129, 506
 - vbOK, 506
 - vbOKCancel, 505
 - vbOKOnly, 505
 - vbQuestion, 129, 505
 - vbRetry, 506
 - vbRetryCancel, 505
 - vbSystemModal, 505
 - vbTab, 215
 - vbYes, 129, 506
 - vbYesNo, 129, 505
 - vbYesNoCancel, 505
 - xlToLeft, 286
 - xlToRight, 286
 - xlUp, 286
 - stałe
 - deklaracja, 116
 - predefiniowane, 117
 - xlLandscape, 117
 - funkcji Dir, 486
 - startFromScratch, 717
 - StatFunction, 344
 - Static, 113, 116, 154, 204
 - StatusBar, 615
 - Step, 145
 - sterowanie
 - bazą danych Access, 435
 - edytorem Word, 438
 - programem Outlook, 446
 - programem PowerPoint, 442
 - wykonywaniem kodu, 133
 - formantów w arkuszu, 528
 - Str, 831
 - StrComp, 831
 - StrConv, 831
 - String, 109, 831
 - StrReverse, 831
 - struktura danych, 35
 - struktura plików aplikacji, 35
 - Sub, 71, 94, 103, 135, 154
 - SUMA, 170, 226, 229
 - suma logiczna, 122
 - Switch, 831
 - SYD, 831
 - symulacja paska narzędzi, 642
 - synchronizacja arkuszy, 318
 - system pomocy w aplikacjach, 48, 96, 761
 - arkusz, 767
 - etykiety, 768
 - formularze UserForm, 768
 - HTML Help, 775
 - kategorie, 762
 - komentarze do zawartości komórek, 764
 - komponenty Excela, 764
 - łączenie pliku pomocy z aplikacją, 779
 - metoda Help, 778
 - nieoficjalne systemy pomocy, 762
 - pliki HLP, 775
 - pliki HTML, 772
 - pliki MHTML, 773
 - pola tekstowe, 766
 - pole kombi, 771
 - przewijane etykiety, 770
 - przypisanie tematów pomocy, 779
 - wybór tematów pomocy, 771
 - wyświetlanie pomocy w oknie przeglądarki sieciowej, 772
 - wyświetlanie tekstu pomocy, 768
 - szablony formularzy UserForm, 561
 - Szybki dostęp, 518
 - dodawanie makr, 704
 - dodawanie przycisków, 703
- ## Ś
- ścieżka pliku, 328
- ## T
- Tab, 535, 831
 - Tab Order, 536
 - tabele
 - bazy danych, 364
 - przetawne, 361
 - CreatePivotTable, 365, 371
 - dane źródłowe, 364
 - jednoczesne tworzenie wielu tabel, 373
 - kompatybilność, 367
 - Lista pól tabeli przestawnej, 362
 - odwrócone, 376
 - optymalizacja wygenerowanego kodu, 365
 - PivotCache, 371
 - PivotCaches, 365
 - PivotFields, 365, 366
 - PivotItems, 365
 - PivotTables, 365

- pola, 372
- ReversePivot, 377, 378
- tworzenie, 362
- znormalizowane, 364
- TabIndex, 536
- tablice, 122
 - deklaracja, 123
 - dynamiczne, 124
 - Option Base, 123
 - sortowanie, 324
 - wielowymiarowe, 123
- TabStrip, 527, 601
- Tag, 556
- Tan, 831
- techniki programowania, 319
- Terminate, 551
- testowanie, 185, 190
 - aplikacji, 43
 - dodatków, 677
 - formularzy UserForm, 537, 563
 - okna dialogowego, 545
 - poleczeń języka VBA, 69
 - wersji beta, 44
- Text, 91
- TextBox, 527, 543, 605
 - SpinButton, 553
- Then, 132, 135
- ThisWorkbook, 84, 246, 249, 571
- Time, 831
- Timer, 305, 831
- TimeSerial, 831
- TimeValue, 831
- To, 123
- ToggleButton, 528
- TogglePageBreakDisplay, 714
- ToggleWrapText, 319
- Toolbox
 - Additional Controls, 602
 - dodawanie formantów ActiveX, 560
 - dodawanie kart, 558
 - dostosowywanie, 558
 - dostosowywanie formantów, 559
 - łączenie formantów, 559
- ToolTipText, 743
- Topic ID, 777
- TRANSPONUJ, 220, 309
- Transpose, 309, 581
- TRANSPOSE, 309
- Trim, 831
- trójwymiarowe skoroszyty, 345
- True, 319
- tryb
 - dostępu
 - Append, 475
 - Binary, 475
 - Input, 475
 - Output, 475
 - Random, 475
 - projektowania, 528
- tryby karty graficznej, 51
- tworzenie
 - aplikacji, 663
 - dodatku, 207, 671
 - dużej liczby wykresów, 395
 - formularze UserForm, 522, 542, 563
 - funkcji, 197
 - karty, 716
 - klasy, 786
 - klawisze skrótów, 39
 - kodu źródłowego, 184
 - kontekstowe menu podręczne, 758
 - korespondencji seryjnej, 439
 - kreatory, 626
 - lista elementów formantu ListBox, 578
 - makra, 54
 - moduły klas, 417, 786
 - niegraficznych wskaźników postępu, 623
 - niestandardowe okna dialogowe, 39
 - odwołania do innego skoroszytu, 163
 - odwrócone tabele przestawne, 376
 - okien dialogowych, 497, 548
 - okno powitalne, 570
 - paski narzędzi, 733
 - połączeń, 458
 - półprzezroczyste formularze UserForm, 657
 - procedur, 153
 - procedury obsługi zdarzeń, 248, 547
 - procedury sortującej, 185
 - samodzielny wskaźnik postępu zadania, 615
 - spisu treści, 317
 - system pomocy, 48
 - szablony formularzy UserForm, 561
 - tabele przestawne, 362
 - typy danych definiowane przez użytkownika, 126
 - unikatowe elementy listy formantu ListBox, 582
 - wykresy na arkuszu wykresu, 384
 - wykresy osadzone na arkuszu danych, 383
 - wykresy przebiegu w czasie, 428
 - wykresy statyczne, 421
 - złożone tabele przestawne, 368
- Type, 126
- TypeName, 831
- typy danych, 107, 108, 109
 - Boolean, 109
 - Byte, 109
 - Currency, 109
 - Date, 109, 118, 119
 - Decimal, 109, 110

definiowane przez użytkownika, 126
 Double, 109
 Integer, 109
 Long, 109
 Object, 109
 określanie typu, 111
 Single, 109
 String, 109, 118
 Variant, 109, 110, 202
 typy zaznaczeń zakresów, 295

U

uaktualnianie zawartości ekranu, 191
 UBound, 831
 UCase, 127, 191, 329, 831
 udostępnienie skoroszytu w trybie tylko do odczytu, 46
 układanka, 660
 ukrywanie
 arkuszy, 46
 dokumentów, 46
 elementów menu podręcznego, 757
 formularza UserForm, 541
 formuł, 45
 kolumn, 46
 kolumn przed wydrukiem, 256
 linii siatki, 122
 wierszy, 46
 wszystkich komórek arkusza, 315
 Uncomment Block, 107
 Unload, 540
 uodpornianie aplikacji na błędy, 45
 UpdateDynamicRibbon, 726
 uruchamianie
 edytora VBE, 57
 okien dialogowych Panelu sterowania, 456
 procedur Sub, 157, 158
 Immediate, 169
 kliknięcie obiektu, 166
 Makro, 158
 niestandardowe menu podręczne, 161
 Run Sub/UserForm, 158
 skrót z klawiszem Ctrl, 159
 Wstążka, 160
 wywołanie, 161
 zapytań, 436
 UserForm, 497, 521
 ustawianie pozycji w pliku, 477
 ustawienia międzynarodowe, 813
 usuwanie
 elementu podręcznego Cell, 752
 elementu z kolekcji ChartObjects, 390
 podmenu menu podręczne, 754
 błędów, 69, 77

modułu, 72
 problemów, 191
 pustych wierszy, 300

V

Val, 831
 ValidateEntry, 247
 Value, 91, 533
 Value2, 91
 VALUES_FROM_SERIES, 405
 Variant, 109, 110, 138, 202, 220, 309
 VarType, 831
 VBA, 21, 53
 biblioteka obiektów ADO, 469
 błędy, 173
 ciąg połączenia, 466
 definiowanie typów danych, 108
 deklaracja zmiennych, 104
 Dim, 114
 długie polecenia, 104
 Do Loop, 150
 Do Until, 150
 Do While, 148
 For...Next, 144
 formularze wprowadzania danych, 519
 funkcje, 120, 127, 827–31
 GoTo, 134
 If, 135
 instrukcje, 823–26
 kolejność operatorów, 121
 kolekcje, 81, 130
 komentarze, 104, 106
 konwersja typów danych, 111
 łańcuchy znaków, 118
 metody, 82
 moduły klas, 783
 nazwy zmiennych, 107
 obiekty, 81, 124, 130
 obsługa błędów, 173
 obsługa języka aplikacji, 816
 obsługa menu podręczne, 746
 obsługa wykresów, 420
 operacje na plikach, 483
 operatory, 120
 operatory logiczne, 122
 operatory porównania, 121
 Option Explicit, 112
 pętle, 143
 pobieranie danych zewnętrznych, 466
 procedury, 153
 procedury Sub, 103
 procedury zwrotne, 705
 projekt, 56, 69

VBA

- przetwarzanie dat, 118
- przetwarzanie dodatków, 685
- przypisanie, 120
- przypisanie tematów pomocy, 779
- Select Case, 139
- stałe, 107, 116
- sterowanie wykonywaniem kodu, 133
- tablice, 122
- Then, 135
- tworzenie dynamicznych połączeń danych, 463
- tworzenie paska narzędzi, 734
- typy danych, 107, 109
- typy danych definiowane przez użytkownika, 126
- While Wend, 151
- właściwości, 82
- Wstążka, 729
- wyrażenia, 120
- wywołanie funkcji Excela, 827
- wywołanie procedury, 161
- zapytania SQL, 463
- zasięg zmiennych, 113
- zastrzeżone słowa kluczowe, 108
- zmienne, 107
- zmienne globalne, 115
- zmienne lokalne, 113
- zmienne obiektowe, 124
- zmienne statyczne, 115
- VBA7, 812
- vbCrLf, 321
- VBE, 54
- VBE, Visual Basic Editor, 68
- vbFormControlMenu, 573
- vbModeless, 572, 610
- vbNo, 129
- vbYes, 129
- Version, 696
- ViewCustomViews, 730
- Visual Basic for Applications, 21, 164
- VSTO, 22

W

- warstwa rysunkowa, 40
- wartości
 - błędów formuł Excela, 223
 - logiczne, 319
- wbudowane funkcje VBA, 127
- wcięcia, 77
- wczesne wiązanie, 432
- Weekday, 140, 831
- WeekdayName, 831
- wersja Excela, 50
 - określanie, 808

- wersje językowe, 50
- While Wend, 151
- wiązanie, 432
- widoczność
 - plików XLAM, 680
 - plików XLSM, 680
- wielkość
 - formularza UserForm, 574
 - znaków w słowach kluczowych, 105
- wielokolumnowe formanty ListBox, 591
- Win64, 812
- WindowActivate, 252, 270
- WindowDeactivate, 252, 270
- WindowResize, 252, 270
- Windows API, 238, 351
 - 64-bitowa wersja Excela, 238
 - funkcje, 239
- Windows Explorer, 453
- Windows Media Player, 602
 - tryb niemodalny, 603
 - URL, 603
- WithEvents, 271
- Wklej, 421
- właściwości, 82, 94, 790
 - do odczytu i zapisu, 791
 - lokalne, 816
 - obiektu AddIn, 687
 - obiektu Application, 84
 - tylko do odczytu, 790
 - tylko do zapisu, 791
- właściwość
 - Accelerator, 537
 - ActiveCellm 83, 287
 - Cells, 89
 - ChartColor, 386
 - ColumnCount, 578
 - ColumnHeads, 578
 - ControlSource, 578
 - CountLarge, 296
 - CurrentRegion, 285, 286
 - Formula, 91, 403
 - IsAddin, 666
 - Height, 650
 - LinkedCell, 528
 - Multiselect, 585
 - Offset, 92
 - Parent, 337
 - Picture, 657
 - Range, 87
 - Resize, 288
 - rng.Column, 340
 - rng.Parent, 340
 - Row, 292
 - Rows.Count, 339

- TablIndex, 536
- Tag, 556
- Text, 91
- typu logicznego, 319
- UsedRange, 300
- Value, 91
- Value2, 91
- Values, 400, 404
- Width, 650
- Xvalues, 400
- XValues, 404
- włączenie obsługi zdarzeń poziomu aplikacji, 271
- WordArt, 167
- Workbook, 381
- Workbook_Open, 246, 756
- WorkbookActivate, 270
- WorkbookAddinInstall, 270
- WorkbookAddinUninstall, 270
- WorkbookBeforeClose, 244, 270
- WorkbookBeforePrint, 270
- WorkbookBeforeSave, 270
- WorkbookDeactivate, 270
- WorkbookIsOpen, 330
- WorkbookNewSheet, 245, 270
- WorkbookOpen, 270
- Workbooks, 679
- Worksheet, 88
 - Range, 88
 - UsedRange, 300
- Worksheet_Change, 247
- WorksheetFunction, 128, 827
- Worksheets
 - Range, 88
- wprowadzanie
 - danych, 24, 498
 - kodu źródłowego języka VBA, 73, 104, 534
 - wartości do komórki, 289
 - wartości do następnej pustej komórki, 291
- wrapper function, 211, 240
- WriteReadRange, 305
- WriteRegistry, 357
- wskaźnik postępu
 - niegraficzny, 623
 - zadania, 614
 - formularz UserForm, 620
 - MultiPage, 622
 - procedura startowa, 619
 - UpdateProgress, 622
 - wyświetlanie, 623
- Wstaw funkcję, 200
- wstawianie
 - formularza UserForm, 522
 - funkcji, 207, 233
 - katégorie funkcji, 235
 - opis funkcji, 237
 - modułu klasy, 786
- Wstążka, 23, 697
 - aktywacja karty, 733
 - dodawanie przycisków, 700
 - dostosowywanie, 38, 700
 - dynamicMenu, 724
 - formanty, 716
 - grupy poleceń, 698
 - karty, 23
 - karty poleceń, 698
 - kod RibbonX, 705
 - kontrolki poleceń, 698
 - modyfikacja, 727
 - ograniczenia w dostosowywaniu, 703
 - poła kombi, 699
 - poła wyboru, 699
 - przyciski dzielone, 698
 - przyciski menu, 699
 - przyciski poleceń, 698
 - tworzenie grupy, 717
 - tworzenie karty, 716
 - uruchamianie procedur Sub, 160
 - VBA, 729
 - wyświetlanie błędów kodu RibbonX, 706
- wstępne rejestrowanie makr, 181
- wybór
 - ikony, 702
 - katalogu, 514
 - koloru, 654
- wydajność, 36
 - dodatków, 692
 - systemu, 51
- wygląd aplikacji, 47
- wykonywanie
 - procedur Sub, 157
 - procedury po wystąpieniu określonego zdarzenia, 168
- wykresy, 379
 - aktywacja wykresu, 386
 - aktywne komórki, 402
 - animowane, 426
 - arkusze wykresu, 384
 - Chart, 381, 417
 - ChartObject, 381, 394
 - deaktywacja wykresu, 389
 - drukowanie wykresów osadzonych na arkuszu, 421
 - eksportowanie, 398
 - eksportowanie obiektów graficznych, 398
 - formularze UserForm, 656
 - identyfikacja zakresu danych, 403
 - lokalizacja wykresu, 380
 - obsługa zdarzeń, 417
 - osadzone na arkuszu danych, 383
 - procedury obsługi zdarzeń, 417

- wykresy
 - przebiegu w czasie, 428
 - przenoszenie wykresu, 387
 - przetwarzanie, 391
 - rejestrator makr, 381
 - seria danych, 400
 - Sparkline, 429
 - SparklineGroup, 429
 - SparklineGroups, 429
 - uaktywnianie, 389
 - układ obiektów ChartObject, 394
 - UserForm, 410
 - usuwanie elementów, 390
 - wykresy statyczne, 421
 - wyrównywanie, 394
 - wyświetlanie etykiet danych, 406
 - wyświetlanie tekstu, 423
 - wykres XY, 408
 - zapisywanie do pliku GIF, 398, 657
 - zdarzenia, 413, 414
 - zmiana danych, 400
 - zmiana rozmiarów, 394
- wykrzywanie
 - błędów, 231
 - wciśnięcia klawisza Shift, 241
- wyłączanie
 - elementów menu podręcznego, 750, 757
 - menu podręcznego, 278, 749
 - obsługi zdarzeń, 247
 - przycisku Zamknij formularza UserForm, 573
- wymagania użytkownika, 33
- wymuszanie deklarowania wszystkich zmiennych, 112
- wyrażenia, 120
- wyrównanie
 - formantów, 530
 - wykresów, 394
- wysyłanie
 - arkusza jako załącznika, 449
 - skoroszytu jako załącznika, 446
 - wiadomości, 450
- wyszukiwanie
 - formantu, 742
 - kodu, 98
 - obrazów FaceID, 755
 - zdarzeń, 269
- wyświetlanie
 - błędów kodu RibbonX, 706
 - czasu, 320, 321
 - daty, 320
 - daty wydrukowania pliku, 335
 - daty zapisania pliku, 335
 - etykiet danych na wykresie, 406
 - formularza danych, 517
 - formularza UserForm, 537, 538
 - formularza wprowadzania danych, 518
 - komunikatów, 129, 504
 - komunikatów o błędach, 192
 - linii siatki, 122
 - listy plików, 485
 - menu podręcznych, 738
 - niemodalnych okien formularzy UserForm, 538
 - okna dialogowego, 514, 545
 - okna folderu, 453
 - pomocy, 773
 - pomocy w formacie HTML Help, 778
 - pomocy w oknie przeglądarki sieciowej, 772
 - tematów pomocy programu Excel, 777
 - wskaźnika postępu, 619
 - wskaźnika postępu zadania, 614
 - wykresów na formularzach UserForm, 656
 - wykresów w formularzu UserForm, 410
 - zaimportowanych danych, 460
- wywołanie
 - funkcji, 206
 - funkcji Excela, 827
 - funkcji Windows API, 351
 - funkcji z poziomu formuły arkusza, 206
 - funkcji z poziomu formuły formatowania warunkowego, 207
 - funkcji z poziomu innej procedury, 206
 - funkcji z poziomu okna Immediate, 209
 - procedury, 153, 165
 - procedury Function, 206
 - procedury z poziomu innej procedury, 161
 - procedury zawartej w innym module, 163
 - procedury zawartej w innym skoroszytcie, 163
- wyznaczanie
 - n-tego elementu łańcucha, 342
 - ostatniej niepustej komórki, 379

X

- XDATE, 230
- XDATEADD, 230
- XDATEDAY, 230
- XDATEDIF, 230
- XDATEMONTH, 230
- XDATEYEAR, 230
- XDATEYEARDIF, 230
- xl4DigitYears, 819
- xlAlternateArraySeparator, 818
- xlColumnSeparator, 818
- xlCommentIndicatorOnly, 765
- xlCountryCode, 813, 817
- xlCountrySetting, 817
- xlCurrencyBefore, 819
- xlCurrencyLeadingZeros, 819
- xlCurrencyMinusSign, 819
- xlCurrencySpaceBefore, 819

xlCurrencyTrailingZeros, 819
 xlDateSeparator, 818
 xlDayCode, 818
 xlDayLeadingZero, 819
 xlDecimalSeparator, 817
 xlDisabled, 192
 xlErrDiv0, 223
 xlErrNA, 223
 xlErrName, 223
 xlErrNull, 223
 xlErrNum, 223
 xlErrRef, 223
 xlErrValue, 223
 XLFile, 791
 xlHourCode, 818
 xlLandscape, 117
 xlLeftBrace, 817
 xlLeftBracket, 817
 xlListSeparator, 817
 xlLowerCaseColumnLetter, 817
 xlLowerCaseRowLetter, 817
 xlMDY, 819
 xlMetric, 819
 xlMinuteCode, 818
 xlMonthCode, 818
 xlMonthLeadingZero, 819
 xlNonEnglishFunctions, 819
 xlRightBrace, 818
 xlRightBracket, 817
 xlRowField, 371
 xlRowSeparator, 818
 xlSecondCode, 818
 xlSeries, 425
 XLSM, 674, 676
 xlThousandsSeparator, 817
 xlTimeLeadingZero, 819
 xlTimeSeparator, 818
 xlUp, 340
 xlUpperCaseColumnLetter, 817
 xlUpperCaseRowLetter, 817
 xlYearCode, 818
 XMLcode, 727
 XOR, 122
 XVALUE_FROM_SERIES, 405
 XValues, 400, 421
 XVALUES_FROM_SERIES, 405

Y

Year, 831

Z

zakresy, 282
 identyfikacja typu zakresów, 286
 kopiowanie, 282
 kopiowanie nieciągłego zakresu komórek, 312
 kopiowanie zakresu o zmiennej wielkości, 284
 odczytywanie zakresów, 305
 określanie typu danych, 303
 określanie typu zaznaczonego zakresu, 295
 określanie zawierania się zakresu w innym zakresie, 303
 pobieranie, 292
 powielanie wierszy, 301
 przenoszenie, 284
 przenoszenie zawartości tablic, 309
 przenoszenie zawartości zakresu do tablicy typu Variant, 309
 przetwarzanie, 285
 przetwarzanie komórek zaznaczonego zakresu, 297
 typy zaznaczeń zakresów, 295
 usuwanie pustych wierszy, 300
 zapisywanie zakresów, 305, 306
 zaznaczanie komórek na podstawie wartości, 310
 zaznaczanie, 286
 zliczanie zaznaczonych komórek, 294

zamiana
 skoroszytu na prezentację, 445
 wartości na słowa, 343

Zamknij, 573

zamykanie
 formularza UserForm, 540
 wszystkich skoroszytów, 315

zapisywanie
 do plików tekstowych, 476
 plików, 478
 wszystkich skoroszytów, 314
 wykresów do pliku GIF, 398, 657
 zakresów, 305, 306
 zawartości rejestru systemu Windows, 356

zapytania, 436
 SQL, 463, 472

zasieg
 funkcji, 205
 procedury, 155
 zmiennych, 113

zastosowanie
 modułów klas, 784
 obiektów ADO, 472
 obiektu FileSystemObject, 488

zaznaczanie
 komórek na podstawie wartości, 310
 zakresów, 286, 568

- zdarzenie, 168, 243, 539
 - Activate, 252, 260, 413, 549, 550
 - AddInInstall, 252, 691, 694
 - AddInUninstall, 252, 691
 - AfterCalculate, 270
 - AfterPrint, 257
 - AfterSave, 252
 - AfterUpdate, 552
 - aplikacje, 244, 269
 - arkusze, 244
 - BeforeClose, 252, 257
 - BeforeDoubleClick, 260, 267, 413
 - BeforeDragOver, 552
 - BeforeDropOrPaste, 552
 - BeforePrint, 252, 256
 - BeforeRightClick, 260, 268
 - BeforeSave, 244, 252, 255
 - BeforeUpdate, 552
 - Button_Click, 241
 - Calculate, 244, 260, 414
 - Change, 260, 261, 549, 552, 553, 554
 - Click, 629
 - Deactivate, 252, 260, 414, 551
 - Enter, 552, 553
 - Error, 552
 - Exit, 552
 - FollowHyperlink, 260
 - formularze UserForm, 244
 - Initialize, 244, 549, 550, 557
 - KeyDown, 552, 553
 - KeyPress, 552
 - KeyUp, 552, 553
 - klawiatura, 275, 553
 - menu podręczne, 756
 - moduły klas, 792
 - monitorowanie zdarzeń poziomu aplikacji, 273
 - monitorowanie zmian w wybranym zakresie
 - komórek, 261
 - MouseDown, 414, 420
 - MouseMove, 414
 - MouseOver, 423
 - MouseUp, 414
 - mysz, 553
 - NewSheet, 244, 252, 254
 - NewWorkbook, 244, 270
 - OnKey, 244, 276
 - OnTime, 244, 274
 - Open, 244, 252, 253
 - PivotTableUpdate, 260
 - procedury obsługi zdarzeń, 245, 539
 - QueryClose, 541, 551, 573
 - Resize, 414
 - sekwencje zdarzeń, 245
 - Select, 244, 414
 - SelectionChange, 244, 260, 266, 402
 - SeriesChange, 244, 414
 - SheetActivate, 245, 252, 270, 611, 613
 - SheetBeforeDoubleClick, 252, 270
 - SheetBeforeRightClick, 252, 270
 - SheetCalculate, 252, 270
 - SheetChange, 244, 252, 270
 - SheetDeactivate, 245, 252, 270
 - SheetFollowHyperlink, 252, 270
 - SheetPivotTableUpdate, 252, 270
 - SheetSelectionChange, 252, 270, 611, 613
 - skoroszyt, 244, 251
 - SpinDown, 552, 553
 - SpinUp, 549, 552, 553
 - Terminate, 541, 551
 - WindowActivate, 252, 270
 - WindowDeactivate, 252, 270
 - WindowResize, 252, 270
 - włączenie obsługi zdarzeń poziomu aplikacji, 271
 - Workbook_Open, 756
 - WorkbookActivate, 270
 - WorkbookAddInInstall, 270
 - WorkbookAddInUninstall, 270
 - WorkbookBeforeClose, 244, 270
 - WorkbookBeforePrint, 270
 - WorkbookBeforeSave, 270
 - WorkbookDeactivate, 270
 - WorkbookNewSheet, 245, 270
 - WorkbookOpen, 270
 - wykonywanie procedury, 168
 - wykresy, 244
 - wyłączanie obsługi zdarzeń, 247
 - wyszukiwanie zdarzeń, 269
- zdarzenia
 - dotyczące obiektu Chart, 417
 - niezwiązane z obiektami, 273
 - obiektu QueryTable, 793
 - poziomu arkusza, 259
- ZDATEDOW, 230
- zestaw rekordów, 468
- zgodność aplikacji, 805
- zliczanie
 - komórek, 338
 - zaznaczonych komórek, 294
- złe pętle, 143
- złożone tabele przestawne, 371
- zmiana
 - danych wykresu, 400
 - kolejności tabulacji formantów, 535
 - rozmiaru zakresu komórek, 288
 - wielkości formularza UserForm, 574

- zmienne, 77, 104, 107
 - deklaracja, 104, 110
 - globalne, 115
 - konwersja typów danych, 111
 - lokalne, 113
 - nazwy, 107
 - obiektywne, 124
 - przypisanie obiektu do zmiennej, 124
 - publiczne, 173
 - statyczne, 115
 - wymuszanie deklarowania wszystkich zmiennych, 112
 - zasięg, 113
- znacznik CustomUI, 714
- Znajdowanie i zamienianie, 574
- znak
 - , 120
 - &, 120
 - *, 120
 - /, 120
 - ^, 120, 121
 - +, 120
 - <, 121
 - <=, 121
 - <>, 121
 - =, 120, 121
 - >, 121
 - >=, 121
 - podkreślenia, 104
- Zoom, 577



PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ 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>

Połącz VBA z Excelem i napisz najlepszą aplikację!

Wielu z nas nie wyobraża sobie pracy bez Excela. Studenci, naukowcy, biznesmeni, analitycy, księgowi, a nawet gospodynie domowe znajdują dla tego potężnego arkusza kalkulacyjnego najróżniejsze zastosowania. Mimo że możliwości Excela są imponujące, można je jeszcze rozszerzać i budować aplikacje szczególnego przeznaczenia za pomocą dostarczonego przez Microsoft narzędzia — języka Visual Basic for Applications (VBA). Wystarczy tylko poznać składnię i zasady programowania w VBA!

Niniejsza książka jest zaktualizowanym wydaniem unikalnego przewodnika po języku VBA. W przystępny sposób wyjaśniono tu wiele złożonych zagadnień, dzięki czemu szybko zaczniesz pisać programy, które automatyzują wykonywanie różnych zadań w Excelu. Będą to zarówno proste makra, jak i wyrafinowane aplikacje i narzędzia, pozwalające na zaawansowaną interakcję z użytkownikiem.

Układ treści w książce pozwala zarówno na systematyczną naukę VBA, jak i na szybkie wyszukanie porad i wskazówek umożliwiających rozwiązanie konkretnego zadania. Nie zabrakło również wielu praktycznych przykładów gotowego do użycia kodu.

Michael Alexander to Microsoft Certified Application Developer (MCAD) i laureat prestiżowego tytułu Microsoft MVP, który otrzymał za bezustanne wspieranie użytkowników Excela. Jest również autorem kilku książek o zaawansowanych technikach analitycznych dla biznesu w MS Excel i Access.

Dick Kusleika od ponad dwudziestu lat pracuje z pakietem Office, a od dwunastu lat zdobywa tytuł Microsoft MVP. Ma ogromne doświadczenia w opracowywaniu rozwiązań bazujących na MS Excel i Access. Jest cenionym szkoleniowcem w dziedzinie obsługi pakietu Office w Australii i w USA.

Dzięki tej książce poznasz:

- podstawy programowania w VBA, w tym tworzenie funkcji i procedur
- projektowanie aplikacji arkusza kalkulacyjnego
- automatyzację operacji na tabelach przestawnych i wykresach
- integrację aplikacji Excela z innymi aplikacjami, takimi jak Word czy Outlook
- pracę z zewnętrznymi źródłami danych

Helion

księgarnia internetowa

<http://helion.pl>

zamówienia telefoniczne

☎ 0 801 339900

☎ 0 601 339900

Informatyka w najlepszym wydaniu

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
<http://helion.pl>

Sprawdź najnowsze promocje:
☉ <http://helion.pl/promocje>
Książki najchętniej czytane:
☉ <http://helion.pl/bestsellery>
Zamów informacje o nowościach:
☉ <http://helion.pl/nowosci>

WILEY

ISBN 978-83-283-2858-7



9 788328 328587

ceną: 99,00 zł

sięgnij po WIĘCEJ



KOD KORZYŚCI