Microsoft

Microsoft Excel 2019

VBA i makra

Bill Jelen i Tracy Syrstad

Przykładowe pliki na stronie Web

Bill Jelen
Tracy Syrstad

Microsoft Excel 2019 VBA i makra

Przekład: Leszek Biolik, Marek Włodarz

APN Promise, Warszawa 2020

Microsoft Excel 2019. VBA i makra

Authorized translation from the English language edition, entitled: Microsoft Excel 2019 VBA and Macros, ISBN: 978-1-5093-0611-4, by Bill Jelen and Tracy Syrstad, published by Pearson Education, Inc, publishing as Microsoft Press, a Division of Microsoft Corporation.

Polish language edition published by APN PROMISE S.A., Copyright © 2020

Autoryzowany przekład z wydania w języku angielskim, zatytułowanego: Microsoft Excel 2019 VBA and Macros, ISBN: 978-1-5093-0611-4, by Bill Jelen and Tracy Syrstad, opublikowanego przez Pearson Education, Inc, publikującego jako Microsoft Press, oddział Microsoft Corporation.

APN PROMISE SA, ul. Domaniewska 44a, 02-672 Warszawa
tel. +48 22 35 51 600, fax +48 22 35 51 699
e-mail: mspress@promise.pl

ISBN: 978-83-7541-424-0 (druk), 978-83-7541-425-7 (ebook)

Przekład: Leszek Biolik, Marek Włodarz
Redakcja: Marek Włodarz
Korekta: Ewa Swędrowska
Skład i łamanie: MAWart Marek Włodarz

Dla Chipa Pearsona.

Witryna Chipa na temat VBA pomogła dziesiątkom tysięcy ludzi na całym świecie. Zasmuciła nas wieść o jego śmierci w wypadku samochodowym w tym roku.

To wydanie książki dedykujemy pamięci Chipa.

– Bill Jelen i Tracy Syrstad

Spis treści

Podziękowania xix
O autorach xx
Wprowadzenie xxi

1 Zwiększanie możliwości programu Excel za pomocą języka VBA 1

Początkowe przeszkody 1
Rejestrator makr nie działa! 2
Nikt w zespole programu Excel nie poświęca wiele uwagi rejestratorowi makr 2
Visual Basic nie przypomina języka BASIC 2
Dobra wiadomość: z poznawanie języka VBA nie jest trudne 3
Dobra wiadomość: Excel wraz z językiem VBA wart jest włożonego wysiłku 3
Poznawanie narzędzi: karta Deweloper 4
Typy plików, dla których dopuszczane są makra 5
Bezpieczeństwo makr 7
Dodawanie zaufanej lokalizacji 7
Zastosowanie ustawień makr w celu włączenia obsługi makr poza zaufanymi lokalizacjami 8
Stosowanie opcji Wyłącz wszystkie makra i wyświetl powiadomienie 9
Przegląd informacji dotyczących rejestrowania, zapisywania i uruchamiania makr 10
Wypełnianie okna dialogowego Rejestrowanie makra 10
Uruchamianie makra 12
Tworzenie przycisku makra na wstążce 12
Tworzenie przycisku makra na pasku narzędzi Szybki dostęp 13
Przypisywanie makra do kontrolki formularza, pola tekstowego lub kształtu 14
Działanie edytora Visual Basic 16
Ustawienia narzędzia VB Editor 17
Eksplorator projektu 17
Okno Properties 18
Mankamenty rejestratora makr 19
Rejestrowanie makra 21
Analiza kodu w oknie programowania 21
Uruchomienie tego samego makra innego dnia generuje nieoczekiwane wyniki 23
Możliwe rozwiązanie: wykorzystywanie odwołań względnych podczas rejestrowania 25

Podczas rejestrowania nigdy nie używaj przycisku Autosumowanie lub Szybka analiza 30
Cztery wskazówki dotyczące używania rejestratora makr 31

2 Skoro nazywa się BASIC, dlaczego nie wygląda znajomo? 33

„Części mowy" języka VBA 34
Język VBA naprawdę nie jest trudny 38
Pliki pomocy VBA: Stosowanie klawisza F1 do wyszukiwania potrzebnych informacji 38
Korzystanie z pomocy 39
Analiza kodu zarejestrowanego makra: korzystanie z edytora VB i tematów pomocy . 40
Parametry opcjonalne 41
Zdefiniowane stałe 42
Właściwości mogą zwracać obiekty 45
Stosowanie narzędzi debugowania do analizy zarejestrowanego kodu 46
Krokowe wykonywanie kodu 46
Inne opcje debugowania: punkty przerwania 48
Poruszanie się w kodzie w przód lub w tył 49
Uruchamianie fragmentu kodu bez trybu krokowego 49
Tworzenie zapytań podczas krokowego wykonywania kodu 49
Wykorzystywanie czujek do ustawiania punktów przerwań 52
Stosowanie czujek do obiektów 53
Narzędzie Object Browser: ostateczne źródło 54
Siedem wskazówek poprawiania zarejestrowanego kodu 55
Wskazówka 1: Niczego nie zaznaczaj 55
Wskazówka 2: Używaj Cells(2,5), ponieważ jest wygodniejsze od Range("E2") . . 56
Wskazówka 3: Używaj bardziej niezawodnych sposobów wyszukiwania ostatniego wiersza 57
Wskazówka 4: Stosuj zmienne, by unikać „sztywnego" kodowania wierszy i formuł 59
Wskazówka 5: Używaj formuł typu R1C1, które ułatwiają życie 59
Wskazówka 6: Kopiuj i wklejaj w pojedynczej instrukcji 59
Wskazówka 7: Używaj konstrukcji With...End With do wykonywania wielu działań 60

3 Odwoływanie się do zakresów 65

Obiekt Range 65
Składnia specyfikowania zakresu 66
Zakresy nazwane 67
Skróty stosowane przy odwołaniach do zakresów 67
Odwoływanie się do zakresów w innych arkuszach 67

Odwoływanie się do zakresu względem innego zakresu 68
Stosowanie właściwości Cells do zaznaczania zakresu 69
Stosowanie właściwości Offset do odwoływania się do zakresu 71
Używanie właściwości Resize do zmiany rozmiaru zakresu 72
Stosowanie właściwości Columns i Rows do określania zakresu 74
Stosowanie metody Union do łączenia wielu zakresów 75
Stosowanie metody Intersect do tworzenia nowego zakresu na podstawie nakładających się zakresów 75
Stosowanie funkcji IsEmpty do sprawdzenia, czy komórka jest pusta 75
Stosowanie właściwości CurrentRegion do zaznaczania zakresu danych Data . . 76
Stosowanie kolekcji Areas do zwracania nieciągłego zakresu 80
Odwołania do tabel 80

4 Pętle i sterowanie przepływem 83

Pętle For...Next 84
Stosowanie zmiennych w instrukcji For 86
Warianty pętli For...Next 87
Wcześniejsze przerywanie pętli po spełnieniu warunku 88
Zagnieżdżanie pętli 89
Pętle Do 90
Stosowanie klauzul While lub Until w pętlach Do 92
Pętla VBA: For Each 95
Zmienne obiektowe 95
Sterowanie przepływem: stosowanie konstrukcji If...Then...Else i Select Case 97
Podstawowe sterowanie przepływem: If...Then...Else 97
Stosowanie konstrukcji Select Case...End Select dla wielu warunków 99

5 Formuły w stylu R1C1 103

Zmiana odwołań na styl R1C1 104
Magia formuł programu Excel 105
Wprowadź formułę raz i skopiuj ją 1000 razy! 105
Nie ma w tym niczego nadzwyczajnego 106
Istota stylu odwołań R1C1 108
Używanie stylu R1C1 dla odwołań względnych 108
Stosowanie stylu R1C1 dla odwołań bezwzględnych 109
Stosowanie notacji R1C1 przy odwołaniach mieszanych 109
Odwoływanie się do całych kolumn lub wierszy 110
Zastępowanie wielu formuł A1 pojedynczą formułą R1C1 110
Pamiętanie numerów kolumn powiązanych z literą kolumn 113
Stosowanie formuł R1C1 w formułach tablicowych 113

6 Tworzenie nazw i operacje na nazwach w VBA 115

Porównanie nazw globalnych i lokalnych 115
Dodawanie nazw 116
Usuwanie nazw 118
Dodawanie komentarzy 118
Typy nazw 119
Formuły 119
Ciągi 120
Liczby 121
Tabele 122
Używanie tablic w nazwach 123
Nazwy zastrzeżone 124
Ukrywanie nazw 125
Sprawdzanie istnienia nazwy 125

7 Programowanie zdarzeń 129

Poziomy zdarzeń 129
Stosowanie zdarzeń 130
Parametry zdarzenia 131
Włączanie zdarzeń 131
Zdarzenia dotyczące skoroszytu 132
Zdarzenia dotyczące arkusza i wykresu na poziomie skoroszytu 134
Zdarzenia dotyczące arkusza 136
Zdarzenia dotyczące wykresów 139
Wykresy osadzone 139
Zdarzenia dotyczące wykresu osadzonego i arkusza wykresu 140
Zdarzenia na poziomie aplikacji 141

8 Tablice 149

Deklarowanie tablicy 149
Deklarowanie wielowymiarowej tablicy 150
Wypełnianie tablicy 151
Pobieranie danych z tablicy 153
Wykorzystywanie tablic do przyspieszenia działania kodu 154
Wykorzystywanie tablic dynamicznych 155
Przekazywanie tablicy 157

9 Tworzenie klas, rekordów i kolekcji 159

Wstawianie modułu klasy 160
Śledzenie zdarzeń dotyczących aplikacji i wykresów osadzonych 160
Zdarzenia dotyczące aplikacji 160

Zdarzenia dotyczące wykresów osadzonych 162
Tworzenie obiektu niestandardowego 164
Stosowanie obiektu niestandardowego 166
Wykorzystywanie kolekcji 167
Tworzenie kolekcji 167
Tworzenie kolekcji w module standardowym 168
Tworzenie kolekcji w module klasy 170
Stosowanie słowników 172
Stosowanie typów zdefiniowanych przez użytkownika do tworzenia właściwości niestandardowych 177

10 Obiekty UserForm – wprowadzenie 181

Pola wprowadzania danych 182
Pola komunikatów 182
Tworzenie formularza użytkownika 183
Wywoływanie i ukrywanie formularza użytkownika 185
Programowanie formularzy użytkownika 185
Zdarzenia związane z formularzami użytkownika 185
Programowanie kontrolek 187
Stosowanie podstawowych kontrolek formularza 189
Stosowanie etykiet, pól tekstowych i przycisków poleceń 189
Wybór pomiędzy polami list a polami kombi w formularzach 191
Stosowanie właściwości MultiSelect pola listy 193
Dodawanie przycisków opcji do formularza użytkownika 195
Dodawanie grafiki do formularza użytkownika 197
Stosowanie przycisku pokrętła na formularzu użytkownika 198
Stosowanie kontrolki MultiPage do łączenia formularzy 200
Weryfikowanie wpisów w polach 203
Nieprawidłowe zamknięcie okna 203
Uzyskiwanie nazwy pliku 204

11 Analiza danych za pomocą funkcji Filtr zaawansowany 207

Zastępowanie pętli funkcją Autofiltr 207
Wykorzystywanie funkcji Autofiltr 210
Zaznaczanie tylko widocznych komórek 215
Filtr zaawansowany – łatwiej w VBA niż w programie Excel 217
Korzystanie z interfejsu użytkownika do budowania filtru zaawansowanego . . 217
Stosowanie filtru zaawansowanego do uzyskiwania listy unikatowych wartości 218
Uzyskiwanie listy unikatowych wartości za pomocą interfejsu użytkownika 219
Wyodrębnianie listy unikatowych wartości za pomocą kodu VBA 220
Uzyskiwanie unikatowej kombinacji dwóch lub większej liczby pól 224

Stosowanie filtru zaawansowanego z zakresami kryteriów 226
Łączenie wielu kryteriów za pomocą alternatywy (OR) 227
Łączenie dwóch kryteriów za pomocą iloczynu logicznego AND 228
Inne trochę bardziej złożone zakresy kryteriów 228
Najbardziej złożone kryteria: Zastępowanie listy wartości warunkiem utworzonym jako wynik formuły 230
Stosowanie warunków opartych na formule w kodzie VBA 232
Stosowanie funkcji Filtr zamiast Filtr zaawansowany 237
Brak rekordów w wyniku przy użyciu opcji Filtruj listę na miejscu 238
Wyświetlanie wszystkich rekordów po uruchomieniu filtrowania listy na miejscu 239
Prawdziwy koń pociągowy: xlFilterCopy dla wszystkich rekordów, a nie tylko unikatowych. 239
Kopiowanie wszystkich kolumn 239
Kopiowanie podzestawu kolumn i zmiana ich kolejności 240
Program Excel w praktyce: Wyłączanie kilku list rozwijanych funkcji Autofiltr . . 247

12 Wykorzystywanie VBA do tworzenia tabel przestawnych 249

Ewolucja tabel przestawnych w różnych wersjach programu Excel 250
Tworzenie tabel przestawnych w języku Excel VBA 251
Definiowanie bufora tabeli przestawnej 251
Tworzenie i konfigurowanie tabeli przestawnej 252
Dodawanie pól do obszaru danych 253
Powody, dla których nie można przenosić lub zmieniać fragmentów raportu przestawnego 256
Określanie rozmiaru gotowej tabeli przestawnej w celu przekształcenia jej na wartości. 256
Stosowanie zaawansowanych funkcji tabel przestawnych. 259
Używanie wielu pól wartości 259
Grupowanie dat poszczególnych dni według miesięcy, kwartałów lub lat 261
Zmiana obliczeń w celu prezentowania wartości procentowych 263
Eliminowanie pustych komórek w obszarze wartości. 265
Kontrolowanie kolejności sortowania za pomocą opcji autosortowania (AutoSort) 266
Powielanie raportu dla każdego produktu 266
Filtrowanie zestawu danych. 269
Ręczne filtrowanie dwóch lub kilku elementów w polu tabeli przestawnej. 269
Stosowanie filtrów pojęciowych 271
Stosowanie filtrów wyszukiwania 275
Konfigurowanie fragmentatorów w celu filtrowania tabeli przestawnej 278
Konfigurowanie osi czasu tabeli przestawnej programu Excel 2019 283

Wykorzystywanie modelu danych w programie Excel 2019 ... 286
Dodanie obu tabel do Modelu danych ... 286
Tworzenie relacji pomiędzy dwoma tabelami ... 287
Definiowanie bufora tabeli przestawnej i tworzenie tabeli przestawnej ... 287
Dodawanie pól modelu do tabeli przestawnej ... 288
Dodawanie pól numerycznych do obszaru wartości ... 288
Zebranie wszystkiego razem ... 289
Stosowanie innych funkcji tabel przestawnych ... 291
Obliczeniowe pola danych ... 291
Elementy obliczeniowe ... 292
Używanie właściwości ShowDetail do filtrowania zestawu rekordów ... 292
Zmiana układu na karcie Projektowanie ... 293
Ustawienia układu raportu ... 294
Wyłączanie sum częściowych dla wielu pól wierszy ... 294

13 Zaawansowane możliwości programu Excel ... 297

Operacje na plikach ... 297
Tworzenie listy plików w katalogu ... 298
Importowanie i usuwanie pliku CSV ... 300
Wczytanie pliku tekstowego do pamięci i jego analiza ... 301
Łączenie i rozdzielanie skoroszytów ... 302
Rozdzielanie arkuszy w oddzielnych skoroszytach ... 302
Łączenie skoroszytów ... 303
Filtrowanie i kopiowanie danych do oddzielnych arkuszy ... 304
Kopiowanie danych do oddzielnych arkuszy bez użycia filtru ... 305
Eksportowanie danych do pliku XML ... 306
Komentarze komórek ... 307
Zmiana rozmiaru komentarzy ... 307
Umieszczanie wykresu w komentarzu ... 309
Śledzenie zmian użytkownika ... 311
Metody dla profesjonalistów języka VBA ... 312
Tworzenie w programie Excel modułu klasy stanu ... 312
Pogłębione analizy tabel przestawnych ... 314
Filtrowanie tabeli przestawnej OLAP według listy elementów ... 315
Tworzenie niestandardowej kolejności sortowania ... 317
Tworzenie wskaźnika postępu ... 318
Stosowanie chronionych pól haseł ... 320
Zmiana wielkości liter ... 322
Zaznaczanie za pomocą SpecialCells ... 324
Resetowanie formatu tabeli ... 324

Używanie możliwości rozbudowy VBA w celu dodawania kodu do nowych skoroszytów 325

14 Przykłady funkcji definiowanych przez użytkownika 329

Tworzenie funkcji definiowanych przez użytkownika 329
Budowanie prostej funkcji użytkownika 330
Udostępnianie funkcji UDF 332
Użyteczne niestandardowe funkcje programu Excel 332
Ustawianie nazwy bieżącego skoroszytu w komórce 332
Ustawianie w komórce nazwy bieżącego skoroszytu i ścieżki pliku 333
Sprawdzenie, czy skoroszyt jest otwarty 333
Sprawdzenie, czy istnieje arkusz w otwartym skoroszycie 334
Określenie liczby skoroszytów w katalogu 335
Pobieranie ID użytkownika 336
Pobieranie informacji o dacie i godzinie ostatniego zapisu 337
Pobieranie informacji o niezmieniającej się dacie i godzinie 338
Sprawdzanie poprawności adresu e-mail 338
Sumowanie komórek w oparciu o kolor ich wypełnienia 340
Zliczanie unikatowych wartości 341
Usuwanie duplikatów z zakresu 342
Wyszukiwanie w zakresie pierwszej komórki o niezerowej długości 344
Zastępowanie wielu znaków 345
Uzyskiwanie liczb z mieszanego tekstu 346
Przekształcenie numeru tygodnia na datę 347
Wyodrębnianie pojedynczego elementu z ograniczonego ciągu 347
Sortowanie i łączenie 348
Sortowanie liczb i znaków alfanumerycznych 350
Wyszukiwanie ciągu wewnątrz tekstu 352
Odwracanie zawartości komórki 352
Zwracanie adresów duplikatów wartości maksymalnych 353
Zwracanie adresu hiperłącza 354
Zwracanie litery kolumny na podstawie adresu komórki 355
Stosowanie statycznej funkcji losowej 355
Używanie `Select Case` na arkuszu 355

15 Tworzenie wykresów 357

Stosowanie metody `.AddChart2` do tworzenia wykresu 358
Style wykresu 359
Formatowanie wykresu 362
Odwoływanie się do określonego wykresu 363
Specyfikowanie tytułu wykresu 364

Stosowanie koloru wykresu. 364
Filtrowanie wykresu. 366
Używanie metody `SetElement` do emulowania zmian dostępnych w menu ikony Plus 367
Stosowanie metody `Format` do zarządzania formatowaniem 372
Zmiana wypełnienia obiektu 373
Formatowanie ustawień linii. 375
Tworzenie wykresów kombi. 376
Tworzenie wykresów kartogramowych 379
Tworzenie wykresów kaskadowych. 380
Eksportowanie wykresu jako grafiki 382
Rozważanie kompatybilności wstecznej 382

16 Wizualizacje danych i formatowanie warunkowe 383

Metody i ich właściwości w VBA do wizualizacji danych 385
Dodawanie pasków danych do zakresów 386
Dodawanie do zakresu skali kolorów 391
Dodawanie do zakresu zestawów ikon 392
Specyfikowanie zestawu ikon. 393
Specyfikowanie zakresów dla każdej ikony 395
Triki wizualizacji danych 395
Tworzenie zestawu ikon dla podzbioru zakresu. 396
Używanie w zakresie dwóch kolorów dla pasków danych. 398
Inne metody formatowania warunkowego 400
Formatowanie komórek, których wartości są powyżej lub poniżej średniej 400
Formatowanie komórek za pomocą reguły Pierwsze 10 lub Ostatnie 5 401
Formatowanie unikatowych lub duplikowanych komórek. 402
Formatowanie komórek w oparciu o ich wartość 404
Formatowanie komórek zawierających tekst. 404
Formatowanie komórek, które zawierają daty 405
Formatowanie komórek, które są puste lub zawierają błędy. 405
Używanie formuły do określenia, które komórki mają być formatowane 406
Stosowanie właściwości `NumberFormat` 407

17 Tworzenie pulpitów nawigacyjnych za pomocą wykresów przebiegu w czasie 409

Tworzenie wykresów przebiegu w czasie 410
Skalowanie wykresów przebiegu w czasie 412
Formatowanie wykresów przebiegu w czasie 416
Stosowanie kolorów motywu. 416
Stosowanie kolorów RGB 419

Formatowanie elementów wykresów przebiegu w czasie 421
Formatowanie wykresów Zysk/strata 424
Tworzenie pulpitu nawigacyjnego 425
Uwagi dotyczące wykresów przebiegu w czasie 426
Tworzenie setek indywidualnych wykresów przebiegu w czasie na pulpicie nawigacyjnym 426

18 Odczytywanie i zapisywanie na stronach sieci Web 433

Uzyskiwanie danych z sieci Web 433
Tworzenie wielu zapytań za pomocą VBA 435
Wyszukiwanie wyników w pobranych danych 436
Zebranie wszystkiego razem 438
Przykłady pobierania danych z witryn sieci Web za pomocą kwerend 439
Używanie metody `Application.OnTime` do okresowej analizy danych 440
Używanie trybu Ready w zaplanowanych procedurach 441
Specyfikowanie okna czasowego dla aktualizacji 441
Anulowanie poprzednio zaplanowanego makra 441
Zamknięcie programu Excel anuluje wszystkie oczekujące zaplanowane makra 442
Planowanie uruchamiania makra x minut w przyszłości 442
Planowanie przypomnienia słownego 443
Zaplanowanie uruchamiania makra co dwie minuty 444
Publikowanie danych na stronie sieci Web 445
Stosowanie VBA do tworzenia niestandardowych stron sieci Web 447
Stosowanie programu Excel jako systemu zarządzania zawartością 447
Bonus: FTP w programie Excel 451

19 Przetwarzanie plików tekstowych 453

Importowanie z plików tekstowych 453
Importowanie plików tekstowych, które mają mniej niż 1 048 576 wierszy 453
Obsługa plików tekstowych zawierających więcej niż 1 048 576 wierszy 461
Zapisywanie plików tekstowych 466

20 Automatyzowanie programu Word 467

Stosowanie wiązania wczesnego do odwoływania się do obiektów programu Word 468
Stosowanie wiązania późnego do odwołań do obiektów programu Word 471
Stosowanie słowa kluczowego `New` w odwołaniach do aplikacji Word 472
Stosowanie funkcji `CreateObject` do tworzenia nowej instancji obiektu 472
Stosowanie funkcji `GetObject` do odwoływania się do istniejącej instancji programu Word 472
Używanie wartości stałych 474

Używanie okna czujek do uzyskiwania rzeczywistych wartości stałych. 474
Stosowanie wyszukiwarki obiektów do uzyskania rzeczywistych wartości stałych 475
Działanie obiektów programu Word 476
Obiekt `Document` 477
Obiekt `Selection` 479
Obiekt `Range` 480
Zakładki 484
Kontrolowanie pól formularzy w programie Word 486

21 Stosowanie bazy danych Access dla usprawnienia dostępu do danych dla wielu użytkowników 489

Porównanie ADO i DAO 490
Narzędzia obiektów ADO. 493
Dodawanie rekordów do bazy danych. 495
Pobieranie rekordów z bazy danych. 496
Aktualizowanie istniejącego rekordu 499
Usuwanie rekordów poprzez obiekt ADO. 501
Sumowanie rekordów za pośrednictwem obiektów ADO 501
Inne narzędzia za pośrednictwem ADO. 503
Sprawdzanie, czy tabela istnieje 503
Sprawdzenie, czy pole istnieje 504
Dodawanie tabeli w locie 505
Dodawanie pola w locie. 506
Przykłady dla SQL Server 506

22 Zaawansowane techniki stosowania obiektów UserForm 509

Stosowanie paska narzędzi UserForm w projektowaniu kontrolek na formularzach. . 509
Dodatkowe kontrolki użytkownika 510
Kontrolki `Checkbox` 510
Kontrolki `TabStrip` 512
Kontrolki `RefEdit` 514
Kontrolki `ToggleButton` 516
Stosowanie paska przewijania jako suwaka wyboru wartości 517
Kontrolki i kolekcje. 519
Niemodalne formularze użytkownika 521
Stosowanie hiperłączy w formularzach użytkownika 522
Dodawanie kontrolek w czasie działania. 523
Zmiana rozmiaru formularza użytkownika w locie 524
Dodawanie kontrolek na bieżąco 525
Zmiana rozmiaru na bieżąco 525

Dodawanie innych kontrolek 526
Dodawanie obrazu na bieżąco 526
Podsumowanie 527
Dodawanie pomocy do formularza użytkownika 530
Wyświetlanie klawiszy akceleratorów 530
Dodawanie kontrolki porady tekstowej 530
Określanie kolejności kart 531
Kolorowanie kontrolki aktywnej 531
Tworzenie przezroczystych formularzy 534

23 Interfejs programowania aplikacji (API) 537

Deklaracje interfejsu API 538
Używanie deklaracji API 539
Tworzenie deklaracji API zgodnych z systemami 32- i 64-bitowymi 540
Przykłady funkcji API 541
Uzyskiwanie nazwy komputera 541
Sprawdzenie, czy plik programu Excel jest otwarty w sieci 542
Uzyskiwanie informacji dotyczących rozdzielczości ekranu 543
Dostosowywanie okna dialogowego Windows – informacje 544
Blokowanie przycisku *X* do zamykania formularza użytkownika 545
Tworzenie czasomierza 546
Odtwarzanie dźwięków 547

24 Obsługa błędów 549

Co dzieje się, kiedy pojawia się błąd? 549
Mylące błędy debugowania kodu formularza użytkownika 551
Podstawowa obsługa błędów za pomocą składni `On Error GoTo` 553
Ogólne programy obsługi błędów 555
Obsługa błędów poprzez ich ignorowanie 555
Blokowanie ostrzeżeń programu Excel 557
Celowe wywoływanie błędu 558
Szkolenia klientów 559
Błędy, które nie ujawniają się w trybie debugowania 559
Błędy podczas projektowania w porównaniu z błędami występującymi kilka miesięcy później 560
Runtime Error 9: Indeks poza zakresem 561
Runtime Error 1004: Niepowodzenie metody Range dla obiektu _Global 562
Dolegliwości kodu chronionego 563
Więcej problemów dotyczących haseł 564
Błędy powodowane przez zmiany wersji 565

25 Dostosowywanie wstążki w celu uruchamiania makr 567

Gdzie dodawać kod: Folder i plik customui 568
Tworzenie kart i grup 570
Dodawanie kontrolki do wstążki 570
Uzyskiwanie dostępu do struktury plików 577
Działanie pliku RELS 578
Zmiana nazwy pliku Excel i otwieranie skoroszytu 579
Używanie obrazów na przyciskach 580
Stosowanie na wstążce ikon pakietu Microsoft Office 580
Dodawanie obrazów ikon do wstążki 581
Rozwiązywanie problemów dotyczących komunikatów o błędach 583
Atrybut `"Attribute Name"` dla elementu `"customui Ribbon"` nie jest zdefiniowany w schemacie lub definicji DTD 583
Niedozwolony znak w nazwie kwalifikowanej 584
Element znacznika `"customui"` nie jest poprawny względem zawartości elementu nadrzędnego 584
Problemy z pewnymi zawartościami 585
Nieprawidłowa liczba argumentów lub nieprawidłowe przypisanie właściwości 586
Nieprawidłowy format lub rozszerzenie pliku 586
Nic się nie stało 587
Inne sposoby uruchamiania makr 587
Stosowanie skrótów klawiszowych do uruchamiania makra 587
Dołączanie makra do przycisku polecenia 588
Dołączanie makra do kształtu 589
Dołączanie makra do kontrolki ActiveX 590
Uruchamianie makra za pomocą hiperłącza 591

26 Tworzenie dodatków 593

Cechy standardowych dodatków 593
Przekształcanie skoroszytu programu Excel na dodatek 594
Używanie funkcji Zapisywanie jako do przekształcenia pliku na dodatek 595
Stosowanie edytora VB do konwersji pliku na dodatek 596
Instalowanie dodatku w systemie klienckim 597
Zamykanie dodatków 600
Usuwanie dodatków 600
Ukryty arkusz jako alternatywa dla dodatku 600

27 Sposoby tworzenia dodatków pakietu Office 603

Tworzenie pierwszego dodatku pakietu Office – Hello World 604
Dodawanie mechanizmów interakcji do dodatku pakietu Office 609

Wstęp do języka HTML 612
Stosowanie znaczników 612
Dodawanie przycisków 613
Wykorzystanie plików CSS 614
Stosowanie XML do definiowania dodatku pakietu Office 614
Wykorzystanie kodu JavaScript w celu dodania interakcji do dodatku pakietu Office 615
Struktura funkcji 616
Zmienne 617
Ciągi 618
Tablice 618
Pętle `for` w kodzie JavaScript 620
Działanie instrukcji `if` w kodzie JavaScript 620
Działanie instrukcji `Select..Case` w kodzie JavaScript 621
Działanie instrukcji `For each..next` w kodzie JavaScript 622
Operatory matematyczne, logiczne i używane do przypisywania 623
Funkcje matematyczne w JavaScript 625
Zapisywanie w okienku zawartości lub okienku zadań 626
Modyfikacje kodu JavaScript, by działał w dodatku pakietu Office 626

28 Nowości i zmiany w programie Excel 2019 629

Subskrypcja Office 365 czy „wieczna" licencja Excel 2019? 629
Jeśli coś zmieniło się w interfejsie użytkownika, zmieniło się też w VBA 630
Wstążka 630
Interfejs SDI (Single Document Interface) 630
Nowoczesne formuły tablicowe 632
Narzędzie Szybka analiza 632
Wykresy 632
Tabele przestawne 633
Fragmentatory 633
Ikony 634
Modele 3D 634
Grafiki SmartArt 634
Poznawanie nowych obiektów i metod 635
Tryb zgodności 635
Stosowanie właściwości `Version` 636
Stosowanie właściwości `Excel8CompatibilityMode` 637
Indeks 639

Podziękowania

Przede wszystkim dziękuję Tracy Syrstad za to, że jest wspaniałym współautorem.

Bob Umlas jest jednym z najinteligentniejszych specjalistów od Excela, jakich znam i jest wspaniałym redaktorem technicznym.

Loretta Yates z wydawnictwa Pearson jest świetnym redaktorem treści. Dziękuję Kughens za kierowanie produkcją tej ksiażki. Nad tym wydaniem pracowałem między innymi w Kola Mi Writing Camp. Dziękuję serdecznie całemu personelowi za dopilnowanie, abym nie zboczył z drogi podczas pracy.

Niemal wszystko, czego nauczyłem się na temat programowania VBA, zawdzięczam cudownej społeczności forum MrExcel.com. VoG, Richard Schollar i Jon von der Heyden szczególnie zasługują na wspomnienie, gdyż ich posty wniosły wiele pomysłów do tej książki. Dziękuję Pam Gensel za pierwszą lekcję na temat makr w Excelu. Mala Singh nauczyła mnie tworzenia wykresów VBA.

Moja rodzina była niezwykle pomocna i cierpliwa w tym czasie. Dziękuję wam, Mary Ellen, Robert F., Barbara i Robert K.!

– *Bill*

Dziękuję wszystkim moderatorom forum MrExcel, którzy dbają o porządek pomimo wielkich wysiłków ze strony spamerów. Programowanie to ciągła nauka i dziękuję tym wszystkim, którzy przychodzą z problemami i pozwalają odkrywać nowe zagadnienia i rozwiązania, z których inaczej w ogóle nie zdawałabym sobie sprawy.

Programowanie to ciągłe uczenie się i naprawdę doceniam klientów, którzy zachęcili mnie do wyjścia poza moją strefę komfortu i rozpoczęcie pracy w nieznanej wcześniej dziedzinie, przez co moje umiejętności i wiedza powiększyły się tak bardzo.

Tym, czego używam dla odstresowania, jest World of Warcraft. Chciałabym przekazać szczególne podziękowania moim przyjaciołom z gry, dzięki czemu daje ona dużo więcej frajdy: Louisiv (za nauczenie mnie, jak używać czołgów), War (za najlepszego czołgistę „obok"), Amabeast (za wypchnięcie mnie ze strefy komfortu), Chraz (za pozostawienie mnie przy życiu) i Jagdeule (za pokazanie, jak świetny może być MM).

Na koniec chcę podziękować Billowi. Jego witryna, MrExcel.com, jest tym miejscem, do którego tysiące przychodzą po pomoc. Jest to również miejsce, w którym ja i inni podobni do mnie znajdują okazję, aby uczyć się i pomagać uczyć się innym.

– *Tracy*

O autorach

Bill Jelen, posiadacz tytułu Excel MVP i gospodarz witryny MrExcel.com, posługuje się arkuszami kalkulacyjnymi od roku 1985, zaś witrynę MrExcel.com utworzył w roku 1998. Jest regularnym gościem wideobloga *Call for Help with Leo Laporte* i wyprodukował ponad 2200 odcinków swojego codziennego podcastu wideo, *Learn Excel from MrExcel*. Jest autorem 57 książek na temat programu Microsoft Excel i comiesięcznego felietonu dla *Strategic Finance*. Przed utworzeniem witryny MrExcel.com Bill Jelen spędził 12 lat na froncie, pracując w działach analiz finansowych, marketingu, księgowości i zarządzania publicznej spółki o kapitale przekraczającym 500 milionów dolarów. Mieszka w Merritt Island na Florydzie ze swą żoną Mary Ellen.

Tracy Syrstad jest programistką i projektantką rozwiązań dla Microsoft Excel oraz autorką dziewięciu książek o tej tematyce. Zajmuje się wsparciem w zakresie Microsoft Office od roku 1997, gdy odkryła istnienie publicznych forum online, w których każdy mógł zadać pytanie lub udzielić odpowiedzi. Tracy odkryła, że uczenie innych sprawia jej przyjemność i gdy zaczęła pracować jako projektant, połączyła radość uczenia z codzienną pracą. Mieszka w odludnym rejonie Południowej Dakoty w towarzystwie męża, psa, dwóch kotów, jednego konia oraz rozlicznych dzikich lisów, wiewiórek i królików.

Wprowadzenie

- Czy JavaScript jest zagrożeniem dla VBA?
- Co zawiera niniejsza książka?
- Wersje programu Excel
- Elementy specjalne i konwencje typograficzne
- Pliki kodu
- Errata i aktualizacje

Kiedy okazało się, że czasy realizowania żądań przez działy IT ciągle się wydłużają, użytkownicy programu Excel odkryli, że przy użyciu języka makr *Visual Basic for Applications* (VBA) sami mogą tworzyć raporty potrzebne do prowadzenia przedsiębiorstwa. VBA umożliwia nam uzyskanie niewiarygodnej efektywności w codziennym użytkowaniu programu Excel. VBA pomaga nam znaleźć sposób na zaimportowanie danych i wygenerowanie raportów w programie Excel, dzięki czemu nie musimy czekać, by w tym zadaniu pomógł nam dział IT.

Czy JavaScript jest zagrożeniem dla VBA?

Pierwsze pytania, jakie zapewne stawia sobie każdy Czytelnik, to: „Czy powinienem poświęcić czas na naukę VBA? Jak długo Microsoft będzie wspierać VBA? Czy obsługa języka JavaScript zapowiedziana w maju 2018 roku zastąpi VBA?"

Odpowiedź brzmi – inwestycja w VBA będzie służyć nam dobrze co najmniej do roku 2046.

Ostatnia zmiana języka makr – z XLM na VBA – nastąpiła w roku 1993. Tym niemniej, XLM nadal jest obsługiwany w Excelu. Mamy do czynienia z sytuacją, gdy VBA jest pod każdym względem *lepszy* niż XLM, ale XLM nadal jest wspierany, 26 lat później. Jeśli Microsoft kiedykolwiek zdecyduje się na przejście z VBA na JavaScript, możemy oczekiwać, że wsparcie dla VBA będzie kontynuowane w wersjach Excela dla Windows i Maca przez (co najmniej) kolejne 26 lat.

W maju roku 2018 firma Microsoft zapowiedziała nowe, wykorzystujące JavaScript funkcje zdefiniowane przez użytkownika (*user-defined function* – UDF), które pozwolą

na tworzenie kodu makr działających zarówno w klienckiej wersji programu Excel, jak i w Excel Online. Ta międzyplatformowość jest interesująca.

W dzisiejszym uniwersum Excela mamy wersje działające w systemach Windows, w MacOS, na telefonach komórkowych opartych na systemach Android i iOS, a także w nowoczesnych przeglądarkach z wykorzystaniem Excel Online. W moim świecie przez 99% czasu używam Excela na komputerze Windows. Zapewne w 1% przypadków otwieram skoroszyty Excela na iPadzie. Jeśli jednak ktoś pracuje w środowisku mobilnym, w którym używa Excela w przeglądarce, funkcje UDF w języku JavaScript mogą być odpowiednie.

Jako wprowadzenie do funkcji UDF JavaScript dla Excela można przeczytać książkę Suata M. Ozgura *Excel JavaScript UDFs Straight to the Point* (ISBN 978-1-61547-247-5).

Niemniej jednak wydajność JavaScript jest nadal fatalna. Jeśli nie potrzebujemy makr, które da się uruchamiać w Excel Online, wersja VBA naszych makr będzie działać (co najmniej) osiem razy szybciej, niż wersja JavaScript. Dla tych, którzy zamierzają uruchamiać Excela wyłącznie na platformach Mac lub Windows, VBA będzie językiem wyboru co najmniej przez następną dekadę.

Zagrożeniem dla Excel VBA mogą być natomiast nowe narzędzia Excel Power Query, które można znaleźć w sekcji Get & Transform (Pobieranie i przekształcanie danych) karty Data (Dane) w programie Excel dla Windows. Jeśli ktoś tworzy makra w celu czyszczenia importowanych danych, powinien rozważyć sprzątanie ich jednorazowo za pomocą Power Query, a następnie odświeżać zapytanie każdego dnia. Mam już mnóstwo opracowanych przebiegów Power Query, które wcześniej wymagały VBA. Jako wprowadzenie do Power Query, warto zapoznać się z książką *Master Your Data with Excel and Power BI: Leveraging Power Query to Get & Transform Your Task Flow* autorstwa Kena Pulsa i Miguela Escobara (ISBN 978-1-61547-058-7).

Co zawiera niniejsza książka?

Zakup tej książki to dobra decyzja – pozwoli skrócić i ułatwić proces opanowania umiejętności pozwalających samemu napisać makra VBA, a w konsekwencji zlikwidować obciążenia związane z ręcznym generowaniem raportów.

Skrócenie procesu przyswajania wiedzy

We Wprowadzeniu przedstawiono analizę przypadku, która ilustruje możliwości makr. W rozdziale 1 „Zwiększanie możliwości programu Excel za pomocą języka VBA" znajdziemy omówienie narzędzi i potwierdzenie znanego faktu: rejestrator makr nie działa stabilnie. Rozdział 2 „Skoro nazywa się BASIC, dlaczego nie wygląda znajomo?" ułatwia zrozumienie szalonej składni języka VBA. Rozdział 3 „Odwoływanie się do zakresów" wyjaśnia, jak sprawnie pracować z zakresami i komórkami.

W rozdziale 4 „Pętle i sterowanie przepływem” omówiono możliwości mechanizmu pętli w języku VBA. Analiza przypadku w tym rozdziale prezentuje proces tworzenia programu generowania raportu dla działu, a następnie „opakowanie” go procedurą, która za pomocą pętli generuje 46 raportów.

W rozdziale 5 „Formuły w stylu R1C1” rzecz jasna opisano działanie tych formuł, a w rozdziale 6 „Tworzenie nazw i operacje na nazwach w VBA” omówiono nazwy. Rozdział 7 „Programowanie zdarzeń” przedstawia niektóre sztuczki używane w programowaniu zdarzeń. Rozdział 8 „Tablice” i rozdział 9 „Tworzenie klas, rekordów i kolekcji” to omówienie tablic, klas i kolekcji, a w rozdziale 10 „Obiekty UserForm – wprowadzenie” opisano niestandardowe okna dialogowe, których można używać do zbierania informacji od użytkowników za pomocą programu Excel.

Możliwości programu Excel VBA

Rozdział 11 „Analiza danych za pomocą funkcji Filtr zaawansowany” i rozdział 12 „Wykorzystywanie VBA do tworzenia tabel przestawnych” to dokładne przedstawienie narzędzi Filtr, Filtr zaawansowany i tabele przestawne. Narzędzia automatyzacji raportów rzadko wykorzystują te koncepcje. W rozdziale 13 „Zaawansowane możliwości programu Excel” i w rozdziale 14 „Przykłady funkcji zdefiniowanych przez użytkownika” omówiono dziesiątki przykładów kodu opracowanego w celu zaprezentowania możliwości programu Excel VBA i funkcji niestandardowych.

W rozdziałach od 15 „Tworzenie wykresów” do 20 „Automatyzowanie programu Word” zapoznajemy się z procesami tworzenia wykresów, wizualizacji danych, generowania zapytań do stron sieci Web i automatyzowania działań w programie Word.

Materiały techniczne potrzebne do tworzenia aplikacji

W rozdziale 21 „Stosowanie programu Access jako wewnętrznej bazy danych w celu usprawnienia dostępu do danych dla wielu użytkowników” omówiono obsługę odczytywania i zapisywania baz danych programu Access i SQL Server. Metody wykorzystywania baz danych Access umożliwiają tworzenie aplikacji z funkcjami obsługi wielu użytkowników, które są dostępne w programie Access, przy jednoczesnym zachowaniu przyjaznego interfejsu użytkownika programu Excel.

W rozdziale 22 „Zaawansowane techniki stosowania obiektów UserForm” rozwinięta została tematyka stosowania formularzy użytkowników. Rozdział 23 „Interfejs programowania aplikacji (API)” zapoznaje nas z niektórymi sposobami realizacji zadań za pomocą interfejsu Windows API. W rozdziałach od 24 „Obsługa błędów” do 26 „Tworzenie dodatków” omówiono obsługę błędów oraz działanie niestandardowych menu i dodatków. Rozdział 27 „Omówienie sposobów tworzenia dodatków pakietu Office” to krótkie wprowadzenie w proces konstruowania własnych aplikacji JavaScript

w programie Excel, a rozdział 28 „Nowości i zmiany w programie Excel 2016” to podsumowanie zmian wprowadzonych w wersji Excel 2016.

Czy książka ta uczy programu Excel?

Firma Microsoft ocenia, że typowi użytkownicy pakietu Office korzystają z 10% funkcji tego oprogramowania. Mamy świadomość, że Czytelnicy tej książki znacznie wykraczają ponad tę średnią, a witryna MrExcel.com ma bardzo inteligentnych użytkowników. Pomimo tego, ankieta przeprowadzona wśród 8000 czytelników witryny MrExcel.com pokazała, że tylko 42% użytkowników (a jest to grupa ponadprzeciętnych użytkowników!) korzysta z przynajmniej jednej spośród 10 zaawansowanych funkcji programu Excel.

Regularnie prowadzę seminaria Power Excel dla księgowych. Są to podstawowi użytkownicy programu Excel, którzy poświęcają 30 do 40 godzin tygodniowo na pracę z tym programem. Na każdym seminarium pojawiają się dwie kwestie. Pierwsza kwestia to mocne zaskoczenie co najmniej połowy uczestników, jak szybko można zrealizować zadanie za pomocą poszczególnych funkcji, na przykład automatycznego podsumowania czy tabel przestawnych. Druga kwestia to fakt, że zawsze ktoś z uczestników mnie „przebija”. Przykładowo, ktoś zadaje pytanie, odpowiadam na nie, a inna osoba w drugim rzędzie podnosi rękę i daje lepszą odpowiedź.

Wniosek? Sporo wiemy na temat programu Excel. Jednakże oceniam, że we wszystkich rozdziałach, może 58% osób nie stosowało wcześniej tabel przestawnych i może nawet jeszcze mniej używało funkcji filtru „10 pierwszych” w tabeli przestawnej. Mając to na uwadze zdecydowałem, że zanim pokażę, jak zautomatyzować cokolwiek w VBA, krótko omawiam wykonanie tego samego zadania w interfejsie programu Excel. Niniejsza książka nie uczy nas, jak tworzyć tabele przestawne, ale jedynie zwraca uwagę na tematykę, z którą warto zapoznać się dokładniej.

Studium przypadku: Miesięczne raporty księgowe

To jest prawdziwa historia. Valerie jest analitykiem biznesowym w dziale księgowości średniej wielkości korporacji. W jej firmie niedawno zainstalowano system ERP (enterprise resource planning), który przekroczył budżet o 16 milionów dolarów. Pod koniec projektu okazało się, że zabrakło funduszy w budżecie IT na generowanie miesięcznych raportów wykorzystywanych w firmie do podsumowywania działań każdego działu.

Jednak Valerie była już bliska podjęcia decyzji o tworzeniu raportów własnymi siłami. Wiedziała, że w tym celu konieczne będzie wyeksportowanie danych księgi głównej z systemu ERP do pliku tekstowego o wartościach rozdzielanych

przecinkami. Za pomocą programu Excel, Valerie potrafiła zaimportować dane księgi głównej z systemu ERP do programu Excel.

Tworzenie raportu nie było proste. Podobnie jak w wielu innych firmach, w danych pojawiały się wyjątki. Valerie wiedziała, że niektóre konta w jednym ze źródeł kosztów powinny być przeklasyfikowane jako wydatki oraz że inne konta trzeba całkowicie usunąć z raportu. Pracując uważnie w programie Excel, Valerie wykonała te dostosowania. Utworzyła tabelę przestawną, by wygenerować pierwszą część podsumowania raportu. Wycięła wyniki tabeli przestawnej i wkleiła je do pustego arkusza. Następnie utworzyła nowy raport tabeli przestawnej dla drugiej części podsumowania. Po około trzech godzinach miała zaimportowane dane, utworzone i poukładane do podsumowania pięć tabel przestawnych oraz starannie sformatowany raport.

Zostać bohaterem

Valerie zaniosła raport menedżerowi, który właśnie usłyszał od działu IT, że generowanie tego zawiłego raportu mogą zabrać dwa, trzy miesiące. Po utworzeniu raportu w programie Excel, Valerie natychmiast stała się bohaterem dnia. W trzy godziny zrobiła coś niemożliwego do wykonania. Valerie była w siódmym niebie, kiedy usłyszała zasłużone „zuch dziewczyna".

Kolejne zachwyty

Następnego dnia, menedżer Valerie uczestniczył w comiesięcznym spotkaniu działów. Kiedy inni menedżerowie działów zaczęli narzekać, że nie mogą uzyskać raportów z systemu ERP, menedżer Valerie położył swój raport na stół. Pozostali byli bardzo zdumieni, jak można było wygenerować ten raport? Każdy był zadowolony, że udało się komuś „złamać kod". Szef firmy poprosił menedżera Valerie, czy nie mógłby przygotowywać raportu dla każdego działu.

Radość przemienia się w strach

Łatwo domyślić się, co się stało. W tej konkretnej firmie było 46 działów. Oznacza to 46 jednostronicowych podsumowań, które trzeba generować co miesiąc. Każdy raport wymaga importowania danych z systemu ERP, usunięcia pewnych kont, utworzenia pięciu tabel przestawnych i sformatowania raportu w kolorze. Utworzenie pierwszego raportu zajęło Valerie trzy godziny, a po nabyciu wprawy, można oszacować, że wygenerowanie 46 raportów zajmie jej około 40. Nawet jeśli Valerie jeszcze bardziej skróci ten czas, i tak jest to horror. Valerie miała wcześniej inne zadania do wykonania, zanim stała się odpowiedzialna za „spędzenie" 40 godzin na generowaniu tych raportów.

Ratunkiem jest VBA

Valerie znalazła moją firmę, MrExcel Consulting i wyjaśniła na czym polega problem. W ciągu tygodnia napisałem kilka makr w VBA, które realizowały wszystkie prozaiczne zadania. Na przykład, makra importowały dane, usuwały zbędne konta, tworzyły 5 tabel przestawnych i formatowały raporty. Inaczej mówiąc, 40-to godzinny ręczny proces został zredukowany do kliknięcia dwóch przycisków i skrócony do około 4 minut.

Niewątpliwie w każdej firmie znajdzie się ktoś przyzwyczajony do ręcznego wykonywania zadań, które można zautomatyzować za pomocą VBA. Jestem przekonany, że mogę pójść do każdej firmy, w której jest ponad 20 użytkowników programu Excel i znajdę podobnie zdumiewający przypadek, jaki spotkał Valerie.

Wersje programu Excel

To szóste już wydanie książki *Język VBA i makra* zostało opracowane do współpracy z programem Excel 2019. Poprzednie wydania tej książki uwzględniały kod dla wersji od Excel 97 po Excel 2016. W 80% w rozdziałach kod dla programu Excel 2019 jest identyczny z kodem dla poprzednich wersji.

Różnice dla użytkowników systemu Mac

Pomimo że program Excel dla systemu Windows i program Excel dla systemu Mac są podobne w kontekście interfejsu, istnieje jednak szereg różnic, jeśli będziemy porównywać środowisko VBA. Rzecz jasna, nic z tego, co w rozdziale 23 wykorzystuje interfejs Windows API, nie będzie działać w systemie Mac. Można powiedzieć, że ogólne koncepcje omawiane w tej książce stosują się także do systemu Mac. Ogólną listę różnic w odniesieniu do systemu Mac znaleźć można pod adresem *http://www.mrexcel.com/macvba.html*. Projektowanie za pomocą VBA dla programu Mac Excel 2019 jest dużo trudniejsze, niż w systemie Windows, ze względu na bardzo uproszczone narzędzia edytowania VBA. Firma Microsoft faktycznie zaleca pisanie kodu VBA w programie Excel 2019 dla systemu Windows, a następnie używanie tego kodu VBA w systemie Mac.

Elementy specjalne i konwencje typograficzne

W książce używane są następujące konwencje typograficzne:

- *Kursywa* – Wskazuje nowe pojęcie podczas jego definiowania, specjalne wyróżnienie, obce słowa czy frazy oraz litery czy słowa używane jako słowa.
- `Czcionka o stałej szerokości` – Wskazuje część kodu VBA, na przykład nazwę obiektu czy metody.
- **`Czcionka wytłuszczona o stałej szerokości`** – Wskazuje na informacje wprowadzane przez użytkownika.

Oprócz tych konwencji typograficznych, używanych jest też kilka elementów specjalnych. W każdym rozdziale jest co najmniej jedno Studium przypadku, które opisuje rzeczywiste rozwiązania typowych problemów. Analizy przypadków pokazują także praktyczne zastosowania tematyki omawianej w rozdziale.

Oprócz ramek Studium przypadku, zobaczymy również ramki Uwaga, Wskazówka i Ostrzeżenie:

Uwaga Uwagi udostępniają dodatkowe, przydatne informacje spoza głównego wątku omawianych kwestii.

Wskazówka Wskazówki opisują szybkie inne sposoby rozwiązywania omawianego problemu, które oszczędzają czas i umożliwiają bardziej efektywną pracę.

Ostrzeżenie Ostrzeżenia wskazują na potencjalne problemy i pułapki, z którymi możemy się zetknąć. Warto zapoznać się z tymi informacjami – mogą oszczędzić nam wielu godzin pracy i zdenerwowania.

Pliki kodu

W podziękowaniu za zakup tej książki, dołączyliśmy zestaw blisko 50 arkuszy programu Excel, które ilustrują koncepcje omawiane w książce. Ten zestaw plików obejmuje cały kod omówiony w książce, dane przykładowe, dodatkowe uwagi od autorów oraz 25 dodatkowych makr. Pliki kodu można pobrać ze strony dostępnej pod adresem *https://microsoftpressstore.com/Excel2019VBAMacros/downloads*.

Errata i aktualizacje

Dołożyliśmy wszelkich starań, aby zapewnić dokładność tej książki i dołączonych do niej materiałów dodatkowych. Aktualizacje – w formie listy zgłoszonych błędów i poprawek – są dostępne pod adresem:

microsoftpressstore.com/Excel2019VBAMacros/errata

Jeśli zauważysz błąd, którego nie ma na tej liście, zgłoś go korzystając z tej samej strony.

Dodatkowe informacje i wsparcie są dostępne pod adresem

http://www.MicrosoftPressStore.com/Support.

Należy zauważyć, że wsparcie dla oprogramowania i sprzętu oferowanego przez firmę Microsoft nie jest dostępne za pośrednictwem wymienionych adresów. Pomoc dotyczącą oprogramowania lub sprzętu firmy Microsoft można uzyskać pod adresem *http://support.microsoft.com*.

ROZDZIAŁ 1

Zwiększanie możliwości programu Excel za pomocą języka VBA

W tym rozdziale:

- Początkowe przeszkody
- Poznawanie narzędzi: karta Deweloper
- Typy plików, dla których dopuszczane są makra
- Bezpieczeństwo makr
- Przegląd informacji dotyczących rejestrowania, zapisywania i uruchamiania makr
- Uruchamianie makra
- Działanie edytora Visual Basic
- Mankamenty rejestratora makr

Visual Basic for Applications (VBA) w połączeniu z Microsoft Excel jest zapewne najbardziej wydajnym spośród dostępnych narzędzi. VBA jest dostępny w komputerach 850 milionów użytkowników Microsoft Office, ale większość z nich nigdy nie spróbowała nawet wykorzystać mocy VBA w Excelu. Przy użyciu VBA można przyśpieszyć niemal każde zadanie wykonywane w Excelu. Jeśli regularnie tworzysz w Excelu comiesięczne wykresy, możesz sprawić, aby VBA wykonał to zadanie w ciągu kilku sekund.

Początkowe przeszkody

Istnieją dwie przeszkody utrudniające opanowanie programowania w VBA. Po pierwsze, rejestrator makr w Excelu nie jest daleki od doskonałości i nie generuje działającego kodu, który możnaby wykorzystywać jako model. Po drugie, dla wielu osób, które wcześniej poznały inny język programowania, na przykład BASIC, składnia języka VBA jest okropnie frustrująca.

Rejestrator makr nie działa!

Firma Microsoft zaczęła dominować na rynku arkuszy kalkulacyjnych w połowie lat 90-tych. Wprawdzie zadanie stworzenia złożonego arkusza kalkulacyjnego, na który mogliby przejść użytkownicy programu Lotus 1-2-3, w zasadzie zakończyło się całkowicie pomyślnie, jednak w odniesieniu do makr było zupełnie inaczej. Dla osób, które sprawnie tworzyły makra 1-2-3, próby rejestrowania makr w Excelu zazwyczaj kończyły się niepowodzeniem. Choć język Microsoft VBA ma znacznie większe możliwości niż język makr programu Lotus 1-2-3, podstawową wadą było niewłaściwe działanie rejestratora makr przy domyślnych ustawieniach.

W programie Lotus 1-2-3 możemy zarejestrować makro jednego dnia, odtworzyć je następnego i zazwyczaj nie będziemy mieli problemów z jego działaniem. Przy próbie wykonania tych samych czynności w programie Microsoft Excel makro może działać dzisiaj, ale jutro już nie. W 1995 roku, kiedy spróbowałem zarejestrować moje pierwsze makro w programie Excel, byłem mocno sfrustrowany takim działaniem programu. W tej książce pokażę trzy reguły, które pozwalają najefektywniej wykorzystywać rejestrator makr.

Nikt w zespole programu Excel nie poświęca wiele uwagi rejestratorowi makr

Kiedy w firmie Microsoft dodawane są nowe funkcje w programie Excel, poszczególni menedżerowie projektu danej funkcji zakładają, że rejestrator makr podczas wykonywania polecenia zarejestruje odpowiednią sekwencję. W poprzedniej dekadzie, rejestrowany kod mógł działać w niektórych sytuacjach, ale najczęściej nie działał we wszystkich sytuacjach. O ile w firmie Microsoft były osoby zajmujące się tworzeniem przydatnego rejestratora makr, rejestrowany kod mógłby być trochę bardziej ogólny, niż jest obecnie.

Kiedyś zazwyczaj można było zarejestrować polecenie jedną z pięciu metod, a zarejestrowany kod przeważnie działał. Niestety obecnie, jeśli chcemy zastosować rejestrator makr, musimy często próbować rejestrowania makra siedmioma różnymi metodami, aż znajdziemy taki zestaw kroków, które rejestrują stabilnie działający kod.

Visual Basic nie przypomina języka BASIC

Dwie dekady wcześniej kod wygenerowany przez rejestrator makr nie przypominał niczego, z czym można było się spotkać do tej pory. Nazywano to „Visual Basic” (VB). Miałem przyjemność nauczenia się tuzina języków programowania w różnych okresach, ale ten dziwacznie wyglądający język był zupełnie nieintuicyjny i zdecydownie nie przypominał języka BASIC poznanego jeszcze w szkole średniej.

Na domiar złego, już w 1995 roku byłem uważany za eksperta w dziedzinie arkuszy kalkulacyjnych. W mojej firmie, w tym właśnie czasie nakazano wszystkim przejście z Lotusa 1-2-3 do Excela, co oznaczało, że musiałem się teraz zmagać z rejestratorem makr, który nie działał i z językiem, którego nie rozumiałem. Trudno to nazwać korzystnym splotem wydarzeń.

Jednym z moich założeń podczas pisania tej książki jest to, że Czytelnicy są sprawnymi użytkownikami arkuszy kalkulacyjnych oraz że Czytelnik książki prawdopodobnie zna ponad 90% osób w swoim biurze. Ponadto, założyłem, że nawet jeśli Czytelnik nie jest programistą, zna podstawy języka BASIC. Jednakże, znajomość tego języka nie jest wymaganiem, a prawdę mówiąc w istocie jest przeszkodą w osiągnięciu celu, jakim jest sprawne programowanie w języku VBA. Istnieje również spora szansa, że Czytelnik próbował rejestrować makro w Excelu i także istnieje podobna szansa, że nie był zadowolony z uzyskanych wyników.

Dobra wiadomość: z poznawanie języka VBA nie jest trudne

Jeśli nawet próba zastosowania rejestratora makr okazała się frustrująca, można to uznać jedynie za niewielką przeszkodę na drodze do pisania złożonych programów w programie Excel. Z tej książki dowiemy się nie tylko, dlaczego nie działa rejestrator makr, ale także, jak zmodyfikować zarejestrowany kod, by stał się użyteczny. Wszystkim Czytelnikom, którzy programowali już w języku BASIC, opiszę język VBA w taki sposób, by mogli łatwo przeanalizować zarejestrowany kod i zrozumieć jego działanie.

Dobra wiadomość: Excel wraz z językiem VBA wart jest włożonego wysiłku

Chociaż pewnie brak możliwości zarejestrowania makr w programie Excel spowodował, że wiele osób poczuło się rozczarowanych firmą Microsoft, trzeba jednak podkreślić, że możliwości języka VBA w programie Excel są ogromne. Absolutnie wszystko, co można wykonać za pomocą interfejsu programu Excel, można też niezwykle szybko osiągnąć używając języka VBA. Jeśli ktoś codziennie czy co tydzień ręcznie tworzy takie same raporty, język VBA w programie Excel znacznie uprości wykonywanie takich zadań.

Autorzy tej książki pracują dla firmy MrExcel Consulting i w ramach tej pracy zautomatyzowali tworzenie raportów dla setek klientów. Te historie zazwyczaj są do siebie podobne: lista zleceń dla działu IT jest tak długa, że ich realizacja zajmuje miesiące. Ktoś z działu księgowego lub konstrukcyjnego odkrywa, że można zaimportować pewne dane do programu Excel i utworzyć raporty potrzebne do działania przedsiębiorstwa.

„Odkrycie” to pozwala nie czekać już przez wiele miesięcy, aż dział IT napisze program. Problem polega jednak na tym, że po zaimportowaniu danych do programu Excel i zdobyciu uznania u przełożonego za utworzenie raportu, najprawdopodobniej czekać będzie nas dodatkowa i uciążliwa praca związana z tygodniowym czy comiesięcznym generowaniem tego raportu.

I ponownie, doskonałą w tej sytuacji wiadomością jest to, że wystarczy poświęcić kilka godzin na programowanie w języku VBA, by zautomatyzować proces tworzenia raportu i sprowadzić go do kilku kliknięć myszą. Satysfakcja jest ogromna, tak więc pozostańcie ze mną – wyjaśnię kilka podstawowych zasad.

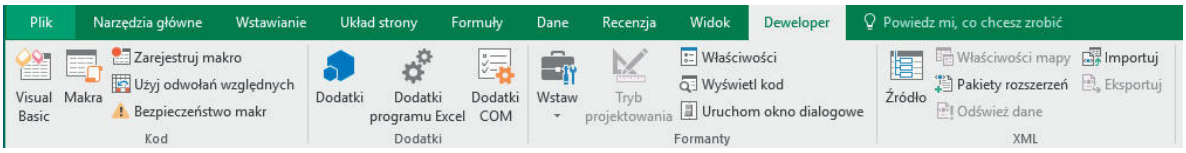
W niniejszym rozdziale wyjaśnimy, dlaczego rejestrator makr nie działa. Przeanalizujemy także przykład zarejestrowanego kodu i pokażemy, dlaczego jednego dnia kod działa, a następnego już nie. W rozdziale znajdują się fragmenty kodu. Zdaję sobie sprawę, że prezentowany w rozdziale kod może nie być zrozumiały dla Czytelnika, ale nie trzeba się tym przejmować. Celem tego rozdziału jest przedstawienie podstawowego problemu dotyczącego rejestratora makr. W rozdziale zaprezentowano również podstawowe elementy środowiska języka Visual Basic.

Poznawanie narzędzi: karta Deweloper

Rozpoczniemy od przeglądu podstawowych narzędzi potrzebnych do korzystania z języka VBA. Domyślnie firma Microsoft ukrywa narzędzia VBA. Aby uzyskać dostęp do karty Deweloper, należy wykonać następujące czynności:

1. Kliknij wstążkę prawym przyciskiem myszy i wybierz polecenie Dostosuj Wstążkę.
2. W okienku z prawej strony zaznacz pole wyboru Deweloper.
3. Kliknij OK, by powrócić do programu Excel.

W programie Excel wyświetlona zostanie karta Deweloper (rysunek 1.1).



Rysunek 1.1 Karta Deweloper dostępnia interfejs do uruchamiania i rejestrowania makr.

W grupie Kod na karcie Deweloper znajdują się ikony używane do rejestrowania i odtwarzania makr VBA:

- **Visual Basic** Ikona otwiera narzędzie Visual Basic Editor.
- **Makra** Kliknięcie ikony powoduje wyświetlenie okna dialogowego Makro, gdzie można uruchomić lub edytować makra wymienione na liście.

- **Zarejestruj makro** Kliknięcie ikony rozpoczyna proces rejestrowania makra.
- **Użyj odwołań względnych** Funkcja ta przełącza pomiędzy trybami rejestrowania względnego i bezwzględnego. W przypadku rejestrowania względnego program Excel rejestruje przemieszczenie w dół o trzy komórki. Dla rejestrowania bezwzględnego program Excel zarejestruje wybranie komórki A4.
- **Bezpieczeństwo makr** Ikona umożliwia dostęp do modułu Centrum zaufania, gdzie dla używanego komputera można zezwolić na uruchamianie makr lub tego zabronić.

W grupie Dodatki udostępniono narzędzia zarządzania zwykłymi dodatkami i dodatkami COM.

Grupa Formanty (na karcie Deweloper) zawiera menu Wstaw, gdzie znajduje się szereg kontrolek, które można umieszczać na arkuszach. Więcej informacji na ten temat można znaleźć w dalszej części, w podrozdziale „Przypisywanie makra do kontrolki formularza, pola tekstowego lub kształtu". Przycisk Uruchom okno dialogowe umożliwia wyświetlenie okna dialogowego zdefiniowanego przez użytkownika lub formularza zaprojektowanego za pomocą języka VBA. Więcej informacji na temat formularzy użytkownika znaleźć można w rozdziale 10 „Obiekty UserForm – wprowadzenie".

W grupie XML na karcie Deweloper umieszczono narzędzia do importowania i eksportowania dokumentów XML.

Grupa Modyfikowanie pozwala określić, czy dla nowych dokumentów zawsze będzie wyświetlane narzędzie Panel dokumentów. W panelu dokumentów użytkownicy mogą wprowadzać słowa kluczowe i opisy dokumentów. Jeśli mamy zainstalowane aplikacje SharePoint i InfoPath, możemy definiować niestandardowe pola, które widoczne będą w panelu dokumentów.

Typy plików, dla których dopuszczane są makra

Program Excel 2019 obsługuje cztery typy plików. Makra nie mogą być przechowywane w plikach .xlsx, a jak wiadomo, ten rodzaj plików jest typem domyślnym! Musimy używać funkcji Zapisz jako dla wszystkich arkuszy zawierających makra, ale można też zmienić domyślny typ pliku używany przez program Excel 2019.

Poniżej wymieniono dostępne typy plików:

- **Skoroszyt programu Excel (.xlsx)** Pliki są przechowywane jako szereg obiektów XML, a następnie są pakowane w postaci pojedynczego pliku. Dzięki temu powstają pliki o znacznie mniejszych rozmiarach, a także możliwa jest edycja lub tworzenie skoroszytów programu Excel w innych aplikacjach (nawet w Notatniku!). Niestety makra nie mogą być przechowywane w plikach z rozszerzeniem .xlsx.

- **Skoroszyt programu Excel z obsługą makr (.xlsm)** Jest to podobny typ pliku do domyślnego .xlsx, przy czym włączona jest obsługa makr. Podstawowe założenie polega na tym, żeby użytkownik mając plik .xlsx nie musiał obawiać się złośliwych makr. Jeśli jednak użytkownik widzi plik .xlsm, powinien zwrócić uwagę na to, że do pliku mogą być dołączone makra.
- **Skoroszyt binarny programu Excel (.xlsb)** Jest to binarny format opracowany w celu obsługi tabel zawierających ponad milion wierszy (możliwość wprowadzona w programie Excel 2007). Starsze wersje programu Excel przechowują swoje pliki we własnych formatach binarnych. Pomimo że formaty binarne mogą ładować się szybciej, są mniej odporne na uszkodzenia, a utrata kilku bitów może zniszczyć cały plik. W tym formacie makra są obsługiwane.
- **Skoroszyt programu Excel 97-2003 (.xls)** Format ten tworzy pliki, które mogą być odczytywane przez użytkowników starszych wersji programu Excel. W tym formacie binarnym dopuszczona jest obsługa makr; jeśli jednak zapiszemy plik w tym formacie, utracimy dostęp do komórek spoza zakresu A1:IV65536. Ponadto, jeśli otworzymy plik w programie Excel 2003, utracimy dostęp do elementów, korzystających z funkcji wprowadzonych w programie Excel 2007 lub nowszym.

Aby uniknąć konieczności wybierania typu skoroszytu z włączoną obsługa makr (w oknie dialogowym Zapisywanie jako), możemy dostosować naszą kopię programu Excel tak, by nowe pliki były zawsze zapisywane w formacie .xlsm. W tym celu:

1. Kliknij menu Plik i wybierz menu Opcje.
2. W oknie dialogowym Opcje programu Excel, w okienku z lewej strony zaznacz kategorię Zapisywanie.
3. Wyświetl listę rozwijaną Zapisz pliki w następującym formacie i wybierz opcje Skoroszyt programu Excel z obsługą makr. Kliknij OK.

Uwaga Chociaż zwykle nie boimy się używać makr, spotykałem osoby, które denerwują się, kiedy widzą plik z rozszerzeniem .xlsm, a naprawdę byli zirytowani, kiedy wysłałem im plik .xlsm, który nie miał żadnego makra. Ich reakcja przypomina gniewny okrzyk króla Artura w filmie „Monty Python i Święty Grail": „You got me all worked up!"[1]. Poczta Gmail portalu Google dołączyła do tego poglądu i odrzuciła prezentowanie podglądu załączników wysyłanych w formacie .xlsm.

Jeśli spotkamy kogoś, kto obawia się plików typu .xlsm, warto przypomnieć, że:

- Każdy skoroszyt utworzony w ciągu ostatnich 30 lat mógł mieć makra, ale w rzeczywistości większość ich nie ma.

[1] „Zmusiłeś mnie do mnóstwa niepotrzebnej pracy".

- Jeśli ktoś chce uniknąć plików z makrami, powinien stosować ustawienia zabezpieczeń, by zapobiec uruchamianiu makr. Osoba taka będzie nadal mogła otwierać plik .xlsm, by uzyskać dane zawarte w arkuszu.

Mam nadzieję, że dzięki tym argumentom pokonamy wszystkie obawy dotyczące plików .xlsm i staną się one domyślnym typem.

Bezpieczeństwo makr

Po tym, jak makra VBA w programie Word wykorzystano jako metodę przesyłania wirusa Melissa, firma Microsoft zmieniła domyślne ustawienia zabezpieczeń tak, by blokować uruchamianie makr. Z tego powodu, zanim rozpoczniemy omawianie sposobów rejestrowania makr, warto przyjrzeć się, jak dostosować ustawienia domyślne.

W programie Excel 2019 można globalnie zmodyfikować ustawienia zabezpieczeń lub kontrolować ustawienia makr dla wskazanych skoroszytów poprzez przechowywanie ich w zaufanej lokalizacji. Makra zostaną automatycznie włączone dla wszystkich skoroszytów zapisanych w lokalizacji oznaczonej jako zaufana.

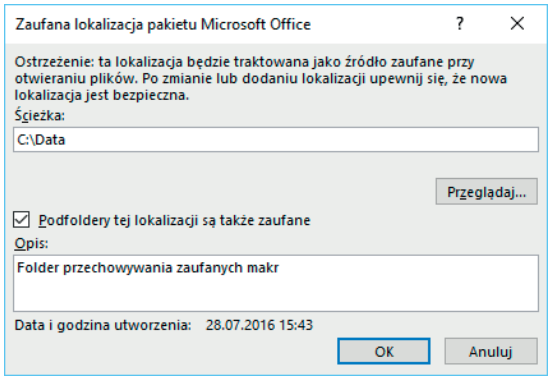
Ustawienia bezpieczeństwa makr znajdziemy, klikając ikonę Bezpieczeństwo makr na karcie Deweloper, co spowoduje wyświetlenie kategorii Ustawienia makr w oknie Centrum zaufania. Okienko nawigacji z lewej strony pozwala uzyskać dostęp do listy Zaufane lokalizacje.

Dodawanie zaufanej lokalizacji

Skoroszyty zawierające makra możemy przechowywać w folderze oznaczonym jako lokalizacja zaufana. Wszystkie skoroszyty zapisane w zaufanym folderze będą miały włączoną obsługę makr. Firma Microsoft zaleca, by zaufane lokalizacje były definiowane na lokalnych dyskach twardych. Zgodnie z ustawieniami domyślnymi nie możemy ufać lokalizacjom na dyskach sieciowych.

Aby zdefiniować zaufaną lokalizację, wykonujemy poniższą procedurę:

1. Kliknij przycisk Bezpieczeństwo na karcie Deweloper.
2. W okienku nawigacji z lewej strony okna Centrum zaufania, kliknij Zaufane lokalizacje.
3. Jeśli zaufane lokalizacje mają znajdować się na dyskach sieciowych, zaznacz opcję Zezwalaj na zaufane lokalizacje w mojej sieci (niezalecane).
4. Kliknij przycisk Dodaj nową lokalizację. Wyświetlone zostaje okno dialogowe Zaufana lokalizacja pakietu Microsoft Office (rysunek 1.2).



Rysunek 1.2 Zarządzanie zaufanymi folderami w oknie Centrum zaufania.

5. Kliknij przycisk Przeglądaj. Program Excel wyświetli okno dialogowe Przeglądaj.
6. Przejdź do folderu nadrzędnego dla folderu, który ma stać się lokalizacją zaufaną. Kliknij zaufany folder. Chociaż nazwa folderu nie jest wyświetlana w polu Nazwa folderu, kliknij OK. Prawidłowa nazwa folderu będzie widoczna w oknie dialogowym Przeglądaj.
7. Aby podfoldery wybranego folderu również były lokalizacjami zaufanymi, zaznacz pole wyboru Podfoldery tej lokalizacji są także zaufane.
8. Kliknij OK, by dodać folder do listy Zaufane lokalizacje.

Ostrzeżenie Podczas wybierania zaufanych lokalizacji należy zachować ostrożność. Po kliknięciu załącznika wiadomości e-mail, będącego plikiem programu Excel, program Outlook zapisze ten plik w folderze tymczasowym na dysku C:. Raczej nie będziemy chcieli dodawać całego dysku C:\ i wszystkich jego podfolderów do listy Zaufane lokalizacje.

Zastosowanie ustawień makr w celu włączenia obsługi makr poza zaufanymi lokalizacjami

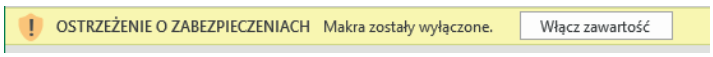
Do wszystkich makr zapisanych poza zaufanymi lokalizacjami program Excel stosuje ustawienia makr. Zmienione zostały nazwy ustawień Niskie, Średnie, Wysokie i Bardzo wysokie, które znamy z programu Excela 2003.

Aby uzyskać dostęp do ustawień makr, należy kliknąć polecenie Bezpieczeństwo makr (na karcie Deweloper). Excel wyświetli kategorię Ustawienia makr okna dialogowego Centrum zaufania. Należy zaznaczyć drugą opcję: Wyłącz wszystkie makra i wyświetl powiadomienie. Poniżej przedstawiono opis każdej opcji:

- **Wyłącz wszystkie makra bez powiadomienia** Ustawienie to blokuje uruchamianie wszystkich makr i jest przeznaczone dla osób, które nigdy nie zamierzają

korzystać z makr. Ponieważ książka uczy, jak używać makr, nie będziemy używać tej opcji. Ustawienie to, ogólnie rzecz biorąc, jest odpowiednikiem opcji Bardzo wysokie w programie Excel 2003. W przypadku tego ustawienia mogą działać jedynie makra zapisane w zaufanych lokalizacjach.

- **Wyłącz wszystkie makra i wyświetl powiadomienie** W opisie działania ustawienia dołączono słowa „wyświetl powiadomienie". Oznacza to, że użytkownikowi wyświetlone zostanie powiadomienie, kiedy otworzy plik zawierający makra i użytkownik zdecyduje, czy chce włączyć zawartość. Jeśli powiadomienie zostanie zignorowane, makra pozostaną wyłączone. Jest to ustawienie podobne do opcji Średnie w programie Excel 2003 i jest to zalecane ustawienie. W programie Excel 2019, w obszarze komunikatów wyświetlane jest powiadomienie, informujące o tym, że makra zostały wyłączone. Klikając tę opcję, użytkownik może włączyć zawartość (rysunek 1.3).
- **Wyłącz wszystkie makra oprócz makr podpisanych cyfrowo** Ustawienie to wymaga użycia narzędzia do tworzenia podpisów cyfrowych firmy VeriSign lub innego dostawcy. Jest to właściwy wybór dla osób zamierzających sprzedawać dodatki innym użytkownikom, ale jest trochę uciążliwym rozwiązaniem w przypadku pisania makr na własny użytek.
- **Włącz wszystkie makra (niezalecane, może zostać uruchomiony niebezpieczny kod)** Ustawienie to jest podobne do ustawienia Niskie w programie Excel 2003. Chociaż ustawienie sprawia najmniej kłopotów, jednak naraża komputer na ataki złośliwych wirusów, podobnych w działaniu do wirusa Melissa. Firma Microsoft nie zaleca używania tego ustawienia.



Rysunek 1.3. Przycisk Włącz zawartość widoczny w przypadku użycia opcji Wyłącz wszystkie makra i wyświetl powiadomienie.

Stosowanie opcji Wyłącz wszystkie makra i wyświetl powiadomienie

Zalecane jest stosowanie ustawienia Wyłącz wszystkie makra i wyświetl powiadomienie. W przypadku użycia tej opcji po otwarciu skoroszytu zawierającego makra, bezpośrednio nad paskiem formuły wyświetli się ostrzeżenie o zabezpieczeniach. Jeśli spodziewamy się, że ten skoroszyt zawiera makra, należy kliknąć przycisk Włącz zawartość. Jeśli nie chcemy włączać makr w otwieranym skoroszycie, możemy zamknąć ostrzeżenie o zabezpieczeniach poprzez kliknięcie ikonki X z prawej strony paska tytułu.

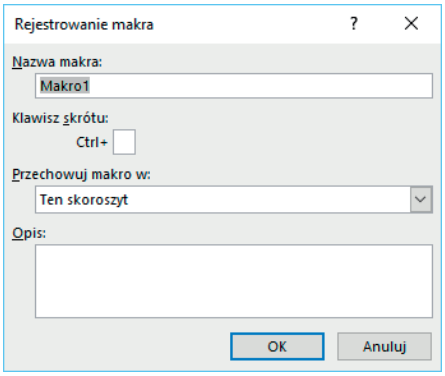
Jeśli zapomnimy włączyć makra i spróbujemy je uruchomić, program Excel poinformuje nas, że nie można uruchomić makra, ponieważ wszystkie zostały wyłączone. W takiej sytuacji najlepiej jest zamknąć skoroszyt i otworzyć go jeszcze raz.

Ostrzeżenie Jeśli włączymy makra w skoroszycie przechowywanym na lokalnym dysku twardym, a następnie go zapiszemy, program Excel zapamięta, że w tym skoroszycie włączaliśmy makra. Przy kolejnym otwieraniu tego skoroszytu obsługa makr będzie automatycznie włączona.

Przegląd informacji dotyczących rejestrowania, zapisywania i uruchamiania makr

Rejestrowanie makr jest przydatne dla osób, które nie mają odpowiedniego doświadczenia, by samodzielnie napisać ich kod. Po zdobyciu potrzebnej wiedzy i doświadczenia coraz rzadziej korzysta się z możliwości rejestrowania makr.

Aby rozpocząć rejestrację makra, na karcie Deweloper wybieramy polecenie Zarejestruj makro. Przed rozpoczęciem rejestrowania program Excel wyświetla okno dialogowe Rejestrowanie makra (rysunek 1.4).



Rysunek 1.4. W oknie dialogowym Rejestrowanie makra, do rejestrowanego makra przypisujemy nazwę i klawisz skrótu

Wypełnianie okna dialogowego Rejestrowanie makra

W polu Nazwa makra wpisujemy jego nazwę. Trzeba pamiętać, by nie używać spacji, przykładowo należy użyć nazwy Makro1, a nie Makro 1. Ponieważ makr może być bardzo dużo, warto stosować nazwy opisowe, na przykład nazwa FormatowanieRaportu jest znacznie bardziej przydatna niż Makro1.

W drugim polu okna dialogowego Rejestrowanie makra definiujemy klawisz skrótu. Jeśli w tym polu wpiszemy małą literę j, a następnie naciśniemy klawisze Ctrl+J, makro to zostanie uruchomione. Warto pamiętać jednak, że większość skrótów z literami, począwszy od Ctrl+A, a skończywszy na Ctrl+Z (za wyjątkiem Ctrl+J) jest już w programie Excel wykorzystywanych do innych zadań. Jednym z rozwiązań jest

przypisywanie do makra połączenia klawiszy Ctrl+Shift+A aż do Ctrl+Shift+Z. Aby przypisać do makra skrót Ctrl+Shift+A, w polu skrótu wpisujemy Shift+A.

Ostrzeżenie Przypisanie skrótów do makr zmienia dotychczasowe przypisania klawiszy. Przykładowo, jeśli przypiszemy do makra kombinację Ctrl+C, program Excel zamiast wykonać standardowe kopiowanie, uruchomi makro.

W oknie dialogowym Rejestrowanie makra można zdecydować o tym, gdzie makro ma być zapisane po zarejestrowaniu. Dostępne opcje to: Skoroszyt makr osobistych, Nowy skoroszyt oraz Ten skoroszyt. Zaleca się, by zapisywać makra związane z danym skoroszytem przy użyciu opcji Ten skoroszyt.

Skoroszyt makr osobistych (Personal.xlsm) nie jest widocznym skoroszytem – skoroszyt jest tworzony, jeśli do zapisu wybierzemy opcję Skoroszyt makr osobistych. Skoroszyt ten jest używany do zapisywania makra w skoroszycie otwieranym automatycznie podczas uruchamiania programu Excel i dzięki temu od razu będzie można korzystać z makra. Po uruchomieniu programu Excel skoroszyt jest ukrywany. Aby wyświetlić ten skoroszyt, należy wybrać opcję Odkryj okno na karcie Widok.

Wskazówka Nie zaleca się używania skoroszytu makr osobistych dla wszystkich zapisywanych makr. Należy zapisywać w nim tylko te makra, które pomagają w wykonywaniu zadań ogólnego przeznaczenia, a nie zadań, które wykonywane są dla określonego arkusza lub skoroszytu.

W czwartym polu okna dialogowego Rejestrowanie makra wpisujemy opis. Opis ten zostanie dodany w formie komentarza na początku makra.

Po wybraniu lokalizacji, w której ma być zapisane makro, należy kliknąć OK. Teraz trzeba zarejestrować makro. Na przykład, w aktywnej komórce wpisujemy Hello World i naciskamy Ctrl+Enter, by zaakceptować wpis i pozostać w tej samej komórce. Po zakończeniu rejestrowania makra klikamy ikonę Zatrzymaj rejestrowanie (na karcie Deweloper).

Wskazówka Ikona Zatrzymaj rejestrowanie jest również dostępna w lewym dolnym rogu okna programu Excel i wygląda jak niewielki niebieski kwadrat. Znajduje się z prawej strony słowa *Gotowy* na pasku stanu. Użycie tego przycisku Zatrzymaj może być czasami wygodniejsze niż powrót karty Deweloper. Po zarejestrowaniu pierwszego makra w tym obszarze zazwyczaj widoczna jest ikona Rejestruj makro.

Uruchamianie makra

Jeśli przypisaliśmy klawisz skrótu do makra, możemy je uruchomić poprzez wciśnięcie tej kombinacji klawiszy. Makra można także przypisywać do przycisków na wstążce lub na pasku narzędzi Szybki dostęp, do kontrolek formularza, do obiektów graficznych lub też możemy je uruchamiać na pasku narzędzi programu Visual Basic.

Tworzenie przycisku makra na wstążce

W celu uruchomienia makra możemy dodać ikonę do nowej grupy na wstążce. Jest to odpowiednie działanie w przypadku makr przechowywanych w skoroszycie makr osobistych. Ikony dodane do wstążki pozostają dostępne, nawet jeśli skoroszyt z makrem nie jest otwarty. Jeśli klikniemy ikonę, kiedy skoroszyt nie jest otwarty, program Excel otworzy skoroszyt i uruchomi makro. Aby dodać przycisk makra na wstążce, wykonujemy następujące instrukcje:

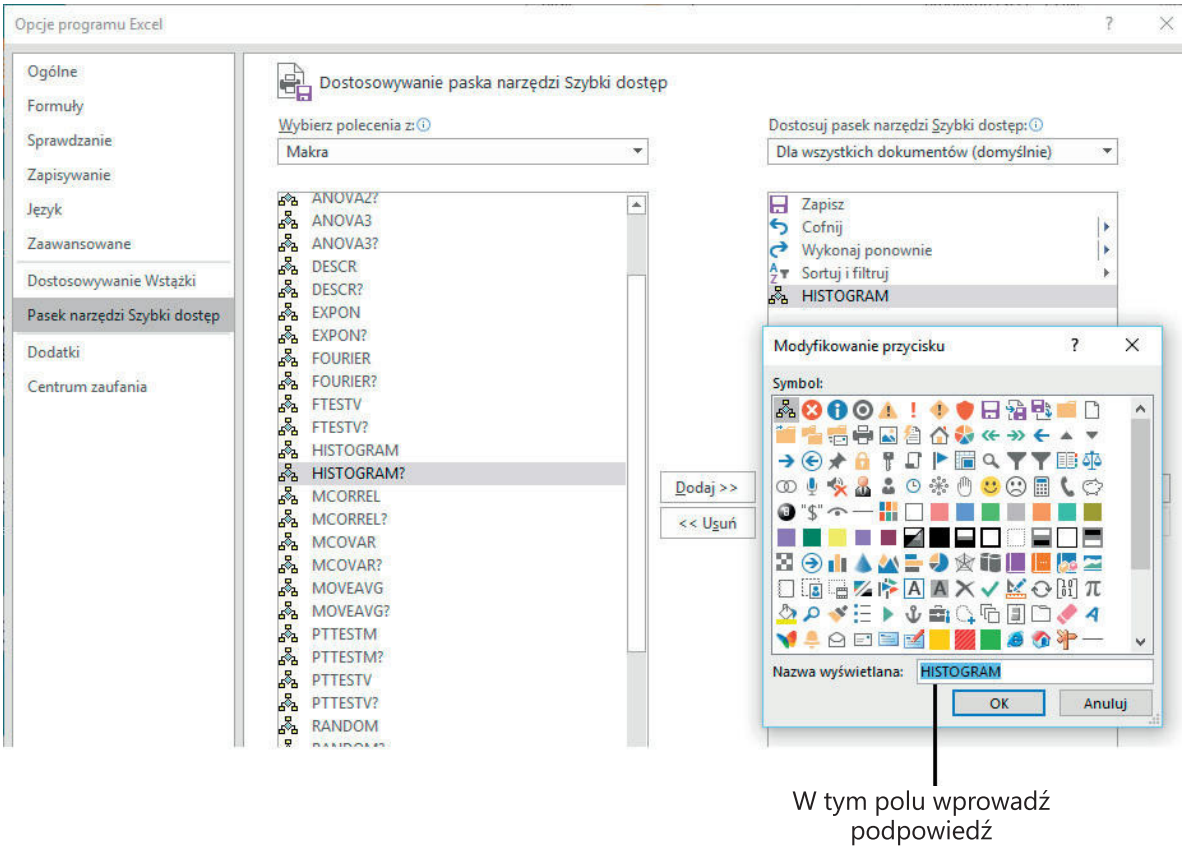
1. Kliknij wstążkę prawym przyciskiem myszy, a następnie wybierz polecenie Dostosuj Wstążkę.
2. Na liście z prawej strony wybierz nazwę karty, do której ma być dodana ikona.
3. Kliknij przycisk Nowa grupa poniżej listy z prawej strony. Program Excel doda nową pozycję nazwaną Nowa grupa (Niestandardowa) na końcu listy grup wybranej karty.
4. Aby przenieść grupę na karcie wstążki w lewo, kilkakrotnie kliknij ikonę strzałki w górę (z prawej strony okna dialogowego).
5. Aby zmienić nazwę grupy, kliknij przycisk Zmień nazwę. Wpisz nową nazwę, na przykład Makra raportów. Kliknij OK. Excel wyświetli grupę na liście jako Makra raportów (Niestandardowa). Należy zwrócić uwagę, że słowo *Niestandardowa* nie pojawi się na wstążce.
6. Otwórz listę rozwijaną w lewym górnym narożniku i wybierz opcję Makra (czwarta kategoria na liście). Excel wyświetli listę dostępnych makr.
7. Wybierz makro z listy po lewej stronie. Kliknij przycisk Dodaj znajdujący się po środku okna dialogowego. Program Excel przeniesie makro na listę z prawej strony do wybranej grupy. Dla wszystkich makr stosowana jest ogólna ikona VBA.
8. Kliknij makro na liście z prawej strony. Kliknij przycisk Zmień nazwę, poniżej tej listy. Program Excel wyświetli listę 180 ikon do wyboru. Wybierz ikonę. Możesz również wpisać opisową etykietę dla tej ikony, na przykład Formatuj raport.
9. Grupę Makra raportów możesz przenieść w inne miejsce na karcie wstążki. W tym celu kliknij pozycję Makra raportów (Niestandardowa) i użyj strzałek w górę i w dół z prawej strony okna dialogowego.

10. Kliknij OK, by zamknąć okno Opcje programu Excel. Na wybranej karcie wstążki pojawi się nowy przycisk.

Tworzenie przycisku makra na pasku narzędzi Szybki dostęp

W celu uruchomienia makra można dodać ikonę do paska narzędzi Szybki dostęp. Jeśli makro zostało zapisane w skoroszycie makr osobistych, do makra można przypisać przycisk, który będzie się stale wyświetlał na pasku narzędzi Szybki dostęp. Jeśli makro jest zapisane w bieżącym skoroszycie, można określić, że ikona będzie wyświetlana tylko wtedy, gdy skoroszyt jest otwarty. Aby dodać przycisk makra do paska narzędzi Szybki dostęp, wykonujemy poniższe czynności:

1. Kliknij prawym przyciskiem myszy pasek narzędzi Szybki dostęp, a następnie wybierz polecenie Dostosuj pasek narzędzi Szybki dostęp.
2. Jeśli makro ma być dostępne tylko wtedy, gdy jest otwarty bieżący skoroszyt, rozwiń listę w prawym górnym narożniku i zmień opcję Dla wszystkich dokumentów (Domyślnie) na opcję Dla *NazwaPliku.xlsm*. Ikony powiązane z bieżącym skoroszytem wyświetlane są na końcu paska narzędzi Szybki dostęp.
3. Rozwiń listę w górnej lewej części okna i wybierz pozycję Makra (czwarta kategoria na liście). Program Excel wyświetli listę dostępnych makr.
4. W oknie z lewej strony wybierz makro z listy. Kliknij przycisk Dodaj znajdujący się na środku okna dialogowego. Program Excel przeniesie makro na listę po prawej stronie. Dla wszystkich makr w programie Excel stosowana jest ogólna ikona VBA.
5. Kliknij makro na liście z prawej strony. Kliknij przycisk Modyfikuj (poniżej okienka listy z prawej strony). Excel wyświetli listę 180 dostępnych ikon (rysunek 1.5). Wybierz ikonę z listy. W polu Nazwa wyświetlania zastąp nazwę makra nazwą skrótową, która widoczna będzie w etykietce ekranowej ikony.
6. Kliknij OK, by zamknąć okno dialogowe Modyfikowanie przycisku.
7. Kliknij OK, by zamknąć okno dialogowe Opcje programu Excel. Na pasku narzędzi Szybki dostęp wyświetli się nowy przycisk.



Rysunek 1.5 Dodawanie przycisku makra do paska narzędzi Szybki dostęp

Przypisywanie makra do kontrolki formularza, pola tekstowego lub kształtu

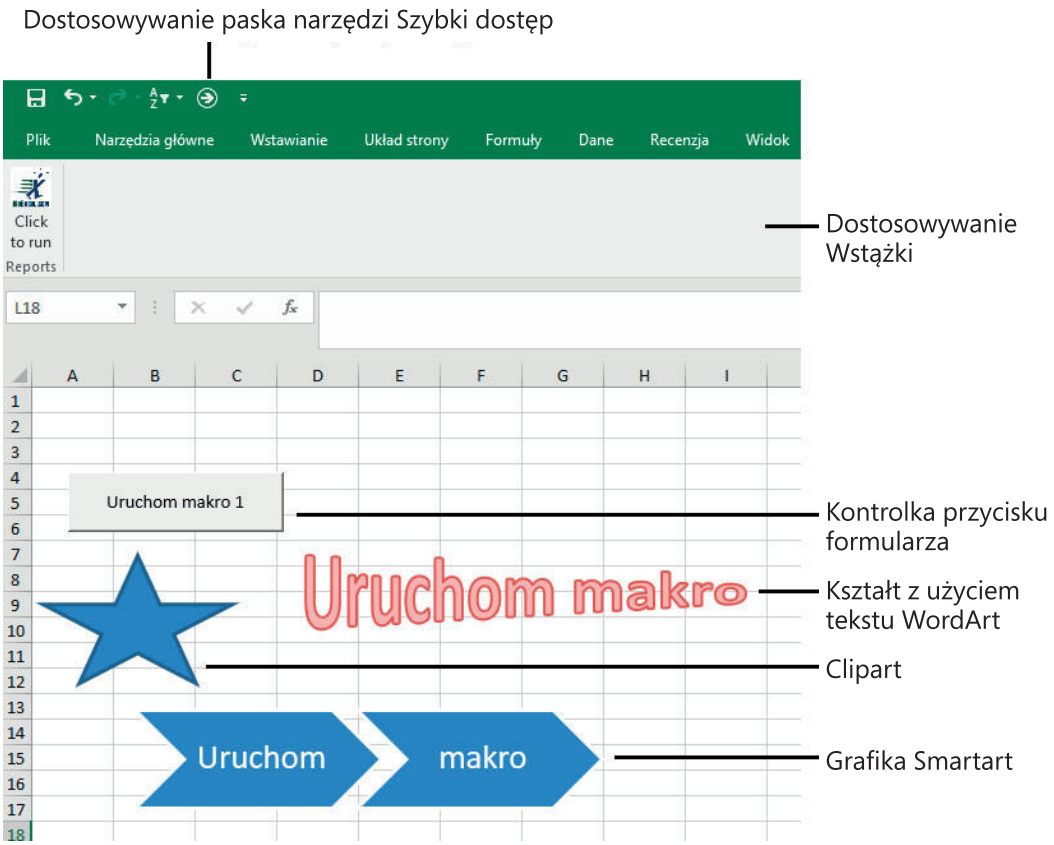
Jeśli chcemy utworzyć makro specyficzne dla danego skoroszytu, zapisujemy je w skoroszycie i dołączamy do kontrolki formularza lub dowolnego obiektu arkusza.

Aby dołączyć makro do kontrolki formularza na arkuszu:

1. Na karcie Deweloper kliknij przycisk Wstaw, by wyświetlić listę rozwijaną tego przycisku. Na tej liście Excel udostępnia 12 kontrolek formularza i 12 kontrolek ActiveX. Na początku prezentowane są kontrolki formularza, a u dołu kontrolki ActiveX. Większość ikon na tej liście w części dotyczącej kontrolek ActiveX wygląda identycznie jak ikonki kontrolek formularza. Kliknij ikonę Przycisk (formant formularza) w lewym górnym narożniku listy rozwijanej Wstaw.
2. Przenieś kursor nad arkusz; kursor zmieni się na znak plus.
3. Narysuj przycisk na arkuszu. W tym celu kliknij i przytrzymaj lewy przycisk myszy, rysując jednocześnie kształt pola. Po zakończeniu rysowania zwolnij przycisk myszy.

4. Wybierz makro w oknie dialogowym Przypisywanie makra i kliknij OK. Spowoduje to utworzenie przycisku wraz z ogólnym opisem typu Przycisk 1.
5. Wpisz nową etykietę przycisku. Podczas pisania ramka zaznaczenia wokół przycisku zmienia się z kropek na skośne linie oznaczające tryb edycji tekstu. W trybie edycji tekstu nie można zmienić koloru przycisku. Aby wyjść z trybu edycji tekstu, należy kliknąć skośne linie (co spowoduje, że zmienią się w kropki) lub ponownie nacisnąć klawisz Ctrl i kliknąć przycisk. Jeśli przypadkowo klikniemy poza przyciskiem, w celu ponownego zaznaczenia przycisku, klikamy przycisk przy naciśniętym klawiszu Ctrl. Następnie należy przeciągnąć kursor nad tekstem, aby go zaznaczyć.
6. Kliknij prawym przyciskiem myszy ramkę w postaci kropek wokół przycisku i wybierz polecenie Formatuj formant. Wyświetlone zostaje okno dialogowe Formatowanie formantu, składające się z siedmiu zakładek. Jeśli okno dialogowe Formatowanie formantu zawiera tylko zakładkę Czcionka, oznacza to, że nadal aktywny jest tryb edycji tekstu. W takim przypadku zamknij okno dialogowe, naciśnij Ctrl i kliknij przycisk, a następnie powtórz ten krok.
7. Użyj opcji w oknie dialogowym Formatowanie formantu, by zmienić rozmiar czcionki, kolor czcionki, marginesy oraz inne ustawienia kontrolki. Po zakończeniu formatowania kliknij OK, by zamknąć okno dialogowe. Kliknij komórkę, by usunąć zaznaczenie przycisku.
8. Kliknij nowy przycisk, aby uruchomić makro.

Makra można przypisywać do dowolnych obiektów arkusza, takich jak grafiki clipart, kształty, obiekty SmartArt czy pola tekstowe. Na rysunku 1.6 górny przycisk ma kształt standardowego przycisku kontrolki formularza. Pozostałe obrazy to obiekt clipart, tekst WordArt i grafika SmartArt. W celu przypisania makra do dowolnego obiektu klikamy obiekt prawym przyciskiem myszy i wybieramy polecenie Przypisz makro.

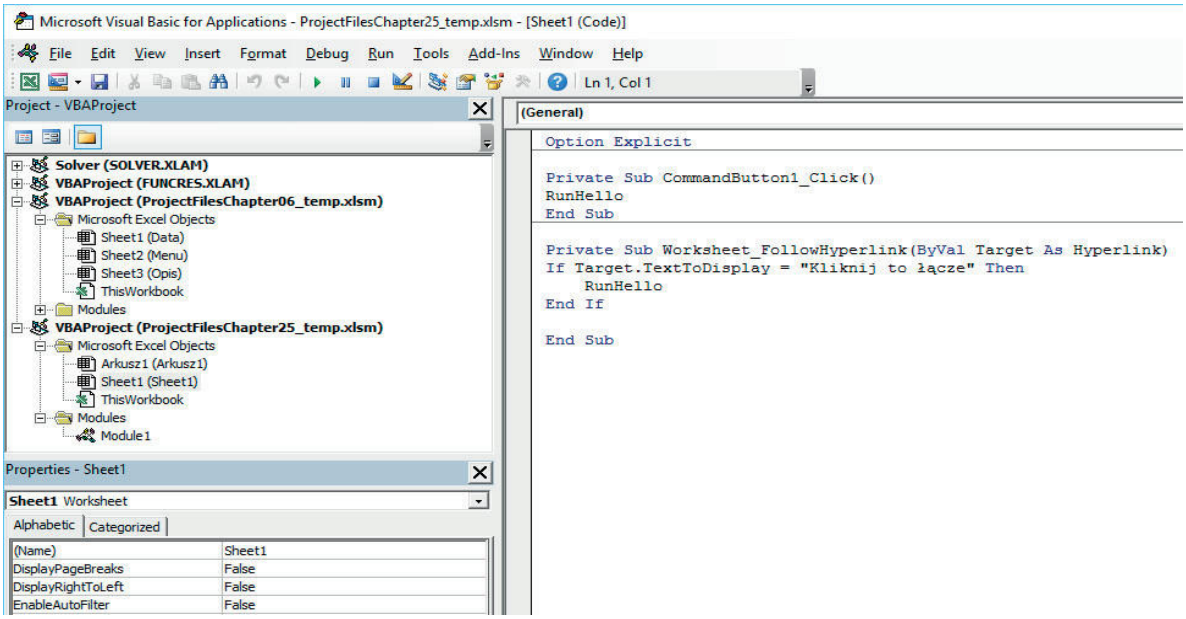


Rysunek 1.6. Przypisywanie makra do kontrolki formularza lub obiektu zapisanego w tym samym skoroszycie co makro. Makro można przypisać do dowolnego spośród tych obiektów

Działanie edytora Visual Basic

Za pomocą narzędzia VB Editor przeprowadzamy edycję zarejestrowanego makra. Aby uruchomić edytor, naciskamy klawisze Alt+F11 lub używamy ikony Visual Basic na karcie Deweloper.

Na rysunku 1.7 pokazano przykład typowego ekranu edytora VB, na którym widoczne są trzy okna: Project Explorer, Properties i Programming (okno eksploratora projektu, właściwości i kodu). Nie należy się przejmować tym, że na komputerze mamy trochę inny ekran, niż okno prezentowane na rysunku, ponieważ podczas omawiania edytora dowiemy się, jak wyświetlać potrzebne okna.



Rysunek 1.7 Okno VB Editor

Ustawienia narzędzia VB Editor

Szereg opcji edytora VB umożliwia dostosowanie narzędzia, by łatwiej było nam pisać makra.

Klika przydatnych ustawień udostępnionych jest na karcie Tools/Options/Editor. Wszystkie opcje, poza jedną, mają odpowiednie domyślne ustawienie. Jedno ustawienie wymaga podjęcia decyzji ze strony użytkownika. Chodzi o opcję Require Variable Declaration (wymagaj deklarowania zmiennych). Zgodnie z ustawieniami domyślnymi program Excel nie wymaga deklarowania zmiennych. Osobiście preferuję to ustawienie, ponieważ pozwala ono zaoszczędzić czas podczas tworzenia programów. Współautorka tej książki jest jednak odmiennego zdania. Zmiana ta powoduje zatrzymanie kompilatora, jeśli napotka zmienną, której nie rozpoznaje, co zmniejsza liczbę pomyłek w nazwach zmiennych. Włączenie lub wyłączenie tej opcji zależy od osobistych preferencji użytkownika.

Eksplorator projektu

W oknie eksploratora projektu wymienione są wszystkie otwarte skoroszyty i załadowane dodatki. Jeśli klikniemy ikonę +, obok pozycji VBAProject, zobaczymy folder Microsoft Excel Objects. Mogą tam być również foldery formularzy, modułów klas i modułów standardowych. Każdy folder zawiera jeden lub kilka komponentów.

Kliknięcie komponentu prawym przyciskiem myszy i wybranie polecenia View Code lub dwukrotne kliknięcie komponentu powoduje wyświetlenie kodu w oknie

Programming. Wyjątek stanowią formularze użytkowników – w tym przypadku dwukrotne kliknięcie powoduje wyświetlenie formularza w widoku projektu (Design).

Aby wyświetlić okno Project Explorer, należy z menu wybrać polecenie View/Project Explorer, nacisnąć klawisze Ctrl+R lub znaleźć i kliknąć dziwną ikonę eksploratora projektu na pasku narzędzi, poniżej menu Tools, znajdującą się pomiędzy ikonami Design Mode i Properties Window.

Aby wstawić moduł, klikamy projekt prawym przyciskiem myszy, wybieramy polecenie Insert, a następnie wskazujemy typ modułu. Poniżej wymieniono dostępne moduły:

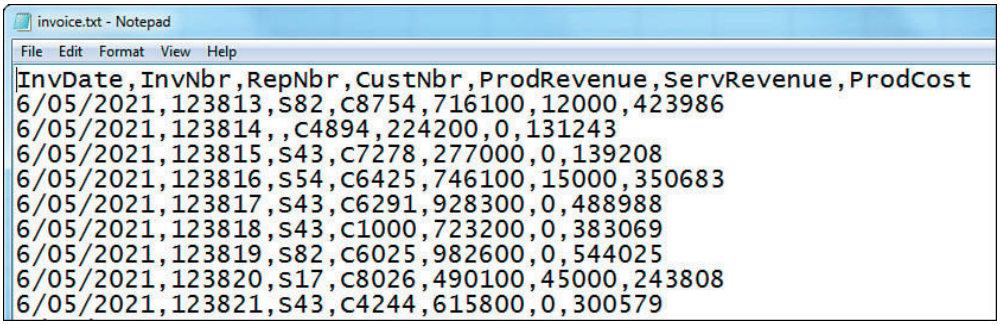
- **Obiekty Microsoft Excel** Domyślnie projekt zawiera moduły arkuszy dla każdego arkusza w skoroszycie oraz jeden moduł ThisWorkbook (Ten skoroszyt). Kod specyficzny dla arkusza, taki jak kontrolki czy zdarzenia związane z arkuszem, umieszczany jest w odpowiadającym mu arkuszu. Kod obsługi zdarzeń skoroszytu umieszczony jest w module ThisWorkbook. Więcej informacji na temat zdarzeń można znaleźć w rozdziale 7 „Programowanie zdarzeń”.
- **Formularze** Program Excel pozwala zaprojektować własne formularze do komunikowania się z użytkownikami. Więcej informacji na temat formularzy można znaleźć w rozdziale 10 „Obiekty UserForm – wprowadzenie”.
- **Moduły** Podczas rejestrowania makra program Excel automatycznie tworzy moduł, w którym umieszcza kod. Większość tworzonego przez nas kodu umieszczana jest w modułach tego typu.
- **Moduły klas** Moduły klas to w programie Excel metoda tworzenia własnych obiektów. Moduły klas pozwalają również na współdzielenie fragmentów kodu pomiędzy programistami bez konieczności poznania sposobu działania kodu. Więcej informacji na temat modułów klas znaleźć można w rozdziale 9 „Tworzenie klas, rekordów i kolekcji”.

Okno Properties

Okno Properties umożliwia edycję właściwości różnych komponentów, takich jak arkusze, skoroszyty, moduły i kontrolki formularzy. Lista właściwości może być różna, w zależności od wybranego komponentu. Aby wyświetlić to okno, z menu wybieramy polecenie View/Properties Window, wciskamy klawisz F4 lub klikamy ikonę okna Properties na pasku narzędzi.

Mankamenty rejestratora makr

Załóżmy, że pracujemy w dziale księgowości. Każdego dnia otrzymujemy plik tekstowy z systemu firmy z listą wszystkich faktur wystawionych poprzedniego dnia. W tym pliku tekstowym poszczególne pola są rozdzielone przecinkami. Kolumny w pliku to DataFaktury, NumerFaktury, NumerSprzedawcy, NumerKlienta, DochódProdukt, DochódUsługa oraz KosztProduktu (rysunek 1.8).



Rysunek 1.8 Plik Invoice.txt

Każdego ranka ręcznie importujemy ten plik do programu Excel. Dodajemy wiersz podsumowania, pogrubiamy nagłówki, a następnie drukujemy raport i przekazujemy go kilku menedżerom.

Wydaje się, że jest to prosty proces, idealnie nadający się do wykorzystania rejestratora makr. Jednak z powodu pewnych problemów z rejestratorem makr pierwsze próby wykonania tego procesu nie muszą zakończyć się sukcesem. W zamieszczonej poniżej analizie przypadku wyjaśniamy, jak pokonać te problemy.

Studium przypadku: Przygotowanie do rejestracji makra

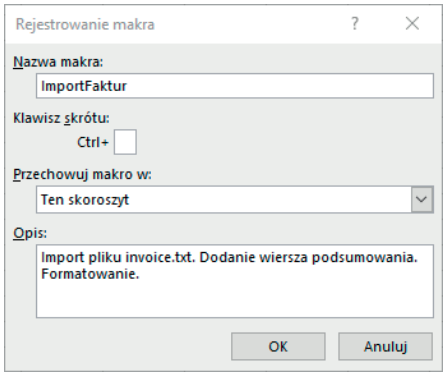
Zadanie opisane w poprzednim fragmencie idealnie nadaje się do utworzenia makra. Przed zarejestrowaniem makra należy jednak przeanalizować wykonywane czynności. W naszym przypadku są to następujące operacje:

1. Kliknięcie menu Plik i wybranie polecenia Otwórz.
2. Przejście do folderu, w którym jest zapisany plik invoice.txt.
3. Z listy rozwijanej Pliki typu wybranie opcji Wszystkie pliki.
4. Wybranie pliku invoice.txt.
5. Kliknięcie przycisku Otwórz.
6. W oknie Kreator importu tekstu – krok 1 z 3, w sekcji Typ danych źródłowych zaznaczenie opcji Rozdzielany.

7. Kliknięcie przycisku Dalej.
8. W oknie Kreator importu tekstu – krok 2 z 3, w ramce Ograniczniki usunięcie zaznaczenia pola wyboru Tabulator i zaznaczenie pola wyboru Przecinek.
9. Kliknięcie Dalej.
10. W oknie Kreator importu tekstu – krok 3 z 3, w ramce Format danych w kolumnie wybranie opcji Data: MDR.
11. Klikniecie przycisku Zakończ, aby zaimportować plik.
12. Naciśnięcie klawisza End i strzałka w dół, by przejść do ostatniego wiersza danych.
13. Ponownie wciśnięcie strzałki w dół, by przejść do wiersza podsumowania.
14. Wpisanie słowa Razem.
15. Cztery razy naciśnięcie klawisza strzałka w prawo, by przejść do kolumny E wiersza podsumowania.
16. Kliknięcie przycisku Autosumowanie, a następnie naciśnięcie klawiszy Ctrl+Enter, by dodać podsumowanie do kolumny DochódProdukt i jednocześnie pozostać w tej komórce.
17. Przeciągnięcie uchwytu automatycznego wypełniania z kolumny E do kolumny G w celu skopiowania formuły podsumowującej do kolumn F i G.
18. Zaznaczenie pierwszego wiersza i kliknięcie ikony Pogrubienie na karcie Narzędzia główne, by wyróżnić czcionkę nagłówków.
19. Zaznaczenie wiersza podsumowania i kliknięcie ikony Pogrubienie, by wyróżnić zawartość tego wiersza.
20. Wciśnięcie Ctrl+A, by zaznaczyć bieżący obszar.
21. Na karcie Narzędzia główne wybieranie narzędzia Formatuj i opcji Autodopasowanie szerokości kolumn.

Po zebraniu czynności, które należy wykonać, możemy przystąpić do rejestracji pierwszego makra. Otwieramy pusty skoroszyt i zapisujemy go na przykład pod nazwą MacroToImportInvoices.xlsm. Na karcie Deweloper klikamy teraz przycisk Zarejestruj makro.

W oknie Rejestrowanie makra domyślna nazwa makra to Makro1. Zmieniamy tę nazwę na bardziej opisową, przykładowo ImportFaktur. Upewniamy się, że do zapisu makra wybrana jest opcja Ten skoroszyt. Ponieważ później przyda się łatwy sposób uruchamiania tego makra, w polu Klawisz skrótu wpisujemy literę i. W polu Opis dodajemy krótki opis działania makra (rysunek 1.9). Po wykonaniu tych czynności klikamy OK.



Rysunek 1.9 Przed zarejestrowaniem makra należy wypełnić pola w oknie dialogowym Rejestrowanie makra.

Rejestrowanie makra

Teraz rejestrator makr zarejestruje każdy nasz ruch. Z tego względu należy starać się wykonywać wszystkie czynności po kolei, bez żadnych dodatkowych działań. Jeśli przypadkowo przejdziemy do kolumny F, a następnie z powrotem do kolumny E w celu wprowadzenia pierwszej sumy, zarejestrowane makro będzie codziennie „ślepo" powielało tę samą pomyłkę. Zarejestrowane makra działają szybko, ale nie warto przyglądać się, jak każdorazowo odtwarzane są wszystkich nasze pomyłki.

Należy zatem uważnie wykonać wszystkie działania niezbędne do utworzenia raportu. Po wykonaniu ostatniej czynności klikamy przycisk Zatrzymaj rejestrowanie (na karcie Deweloper).

Analiza kodu w oknie programowania

Spróbujmy przyjrzeć się kodowi, który przed chwilą zarejestrowaliśmy w analizie przypadku. Nic nie szkodzi, jeśli na razie kod ten nie jest zrozumiały.

Aby otworzyć edytor VB, naciskamy klawisze Alt+F11. Dla pozycji VBA Project (Makro-DoImportowaniaFaktur.xlsm) odnajdujemy komponent Module1. Trzeba go kliknąć prawym przyciskiem myszy i wybrać polecenie View Code. Warto zwrócić uwagę, że niektóre wiersze rozpoczynają się od apostrofu; są to komentarze, które są ignorowane przez program. Rejestrator makr rozpoczyna makra od kilku komentarzy. Do tego celu wykorzystuje opis, który wprowadziliśmy w oknie dialogowym Rejestrowanie makra. Komentarz dotyczący klawisza skrótu ma przypominać, jaki klawisz skrótu wybraliśmy dla makra.

Uwaga Komentarz *nie* powoduje przypisania klawisza skrótu. Jeśli zmodyfikujemy komentarz i wprowadzimy tam ciąg Ctrl+J, nie spowoduje to zmiany klawisza skrótu przypisanego do makra. W celu zmiany klawisza skrótu należy zmienić ustawienie w oknie dialogowym Makro w Excelu lub uruchomić poniższy kod:

```
Application.MacroOptions Macro:="ImportFaktur", _
    Description:="", ShortcutKey:="j"
```

Zarejestrowane makro jest zazwyczaj dość staranne (rysunek 1.10). W każdym wierszu kodu (poza wierszami komentarza) jest wcięcie o szerokości czterech znaków. W celu kontynuacji wiersza kodu w następnym wierszu należy na końcu wiersza umieścić spację i znak podkreślenia. Nie zapominajmy o spacji przez znakiem podkreślania – jej brak spowoduje błąd.

Uwaga Ze względu na fizyczne rozmiary tej książki w pojedynczym wierszu nie zmieści się 100 znaków. Dlatego wiersze będą dzielone przy szerokości około 80 znaków, by mogły zmieścić się na stronie. Z tych też powodów makro, które zarejestrowaliśmy na komputerze, może się nieco różnić od prezentowanego w książce.

```
Sub ImportInvoice()
'
' ImportInvoice Macro
' Import Invoice.txt. Add Total Row. Format.
'
' Keyboard Shortcut: Ctrl+i
'
    Workbooks.OpenText Filename:="G:\2016VBA\SampleFiles\invoice.txt", Origin:= _
        437, StartRow:=1, DataType:=xlDelimited, TextQualifier:=xlDoubleQuote, _
        ConsecutiveDelimiter:=False, Tab:=False, Semicolon:=False, Comma:=True _
        , Space:=False, Other:=False, FieldInfo:=Array(Array(1, 3), Array(2, 1), _
        Array(3, 1), Array(4, 1), Array(5, 1), Array(6, 1), Array(7, 1)), TrailingMinusNumbers _
        :=True
    Selection.End(xlDown).Select
    Range("A11").Select
    ActiveCell.FormulaR1C1 = "Total"
    Range("E11").Select
    Selection.FormulaR1C1 = "=SUM(R[-9]C:R[-1]C)"
    Selection.AutoFill Destination:=Range("E11:G11"), Type:=xlFillDefault
    Range("E11:G11").Select
    Rows("1:1").Select
    Selection.Font.Bold = True
    Rows("11:11").Select
    Selection.Font.Bold = True
    Selection.CurrentRegion.Select
    Selection.Columns.AutoFit
End Sub
```

Rysunek 1.10 Zarejestrowane makro ma schludny wygląd i estetyczne wcięcia

Zwróćmy uwagę, że poniższe siedem wierszy zarejestrowanego kodu to w rzeczywistości tylko jeden wiersz, który został podzielony w celu poprawy czytelności:

```
Workbooks.OpenText Filename:= "C:\ścieżka\invoice.txt", _
    Origin:=437, StartRow:=1, DataType:=xlDelimited, _
    TextQualifier:=xlDoubleQuote, ConsecutiveDelimiter:=False, _
    Tab:=False, Semicolon:=False, Comma:=True, Space:=False, _
```

```
Other:=False, FieldInfo:=Array(Array(1, 3), Array(2, 1), _
Array(3, 1), Array(4, 1), Array(5, 1), Array(6, 1), Array(7, 1)), _
TrailingMinusNumbers:=True
```

Uwzględniając, że powyższy kod to jeden wiersz, widzimy, że rejestrator makr zarejestrował proces składający się z 21 kroków w 14 wierszach kodu. Robi wrażenie.

Uwaga Każde działanie wykonywane za pomocą interfejsu użytkownika programu Excel może odpowiadać jednemu wierszowi lub kilku wierszom zarejestrowanego kodu. Niektóre działania mogą generować kilkanaście wierszy kodu.

Sprawdzanie każdego makra

Zawsze warto sprawdzać makra. W celu przetestowania nowego makra powrócimy do standardowego interfejsu programu Excel, naciskając klawisze Alt+F11. Zamykamy plik invoice.txt bez zapisywania zmian. Arkusz MacroToImportInvoices.xlsm jest w dalszym ciągu otwarty.

Naciskamy klawisze Ctrl+I, by uruchomić zarejestrowane makro. Makro powinno zadziałać dobrze, jeśli prawidłowo wykonaliśmy poszczególne kroki procedury. Dane zostały zaimportowane, podsumowania dodane, formatowanie zastosowane, a szerokość kolumn powiększona. Wydaje się, że to doskonałe rozwiązanie (rysunek 1.11).

	A	B	C	D	E	F	G	H
1	InvDate	InvNbr	RepNbr	CustNbr	ProdRevenue	ServRevenue	ProdCost	
2	6/5/2021	123813	S82	C8754	716100	12000	423986	
3	6/5/2021	123814		C4894	224200	0	131243	
4	6/5/2021	123815	S43	C7278	277000	0	139208	
5	6/5/2021	123816	S54	C6425	746100	15000	350683	
6	6/5/2021	123817	S43	C6291	928300	0	488988	
7	6/5/2021	123818	S43	C1000	723200	0	383069	
8	6/5/2021	123819	S82	C6025	982600	0	544025	
9	6/5/2021	123820	S17	C8026	490100	45000	243808	
10	6/5/2021	123821	S43	C4244	615800	0	300579	
11	Total				5703400	72000	3005589	
12								

Rysunek 1.11 Makro formatuje dane w arkuszu

Uruchomienie tego samego makra innego dnia generuje nieoczekiwane wyniki

Po przetestowaniu makra należy je zapisać, by można go było użyć następnego dnia. Następnego dnia po przyjściu do pracy otrzymaliśmy nowy plik invoice.txt z systemu. Otwieramy makro, wciskamy Ctrl+I, by je uruchomić i… mamy katastrofę. W pliku z 5 czerwca jest 9 faktur, a 6 czerwca faktur jest 17. Zarejestrowane makro „ślepo”

dodało wiersz z podsumowaniem w wierszu 11, ponieważ właśnie w tym wierszu umieściliśmy podsumowanie w momencie, gdy było ono rejestrowane (rysunek 1.12).

Dla tych Czytelników, którzy w trakcie lektury korzystają z plików przykładowych dla tej książki, oto kroki, które należy wykonać, aby spróbować zaimportować dane innego dnia:

1. Zamknij plik Invoice.txt w programie Excel.
2. W Eksploratorze Windows zmień nazwę Invoice.txt na Invoice1.txt.
3. Przemianuj plik Invoice2.txt na Invoice.txt.
4. Wróć do Excela i arkusza MacroToImportInvoices.xlsm.
5. Naciśnij Ctrl+I, aby uruchomić makro dla większego zbioru danych.

Problem wynika stąd, że rejestrator makr zapisał wszystkie działania w trybie odwołań bezwzględnych (domyślnie). Jako alternatywę dla stosowania domyślnego stanu rejestratora w kolejnym rozdziale omówię rejestrowanie względnych odwołań, co powinno nas zbliżyć do pożądanego rozwiązania.

	A	B	C	D	E	F	G	H
1	InvDate	InvNbr	RepNbr	CustNbr	ProdRevenue	ServRevenue	ProdCost	
2	6/6/2021	123829	S21	C8754	21000	0	9875	
3	6/6/2021	123830	S45	C3390	188100	0	85083	
4	6/6/2021	123831	S54	C2523	510600	0	281158	
5	6/6/2021	123832	S21	C5519	86200	0	49967	
6	6/6/2021	123833	S45	C3245	800100	0	388277	
7	6/6/2021	123834	S54	C7796	339000	0	195298	
8	6/6/2021	123835	S21	C1654	161000	0	90761	
9	6/6/2021	123836	S45	C6460	275500	10000	146341	
10	6/6/2021	123837	S54	C5143	925400	0	473515	
11	**Total**	**123838**	**S21**	**C7868**	**3306900**	**10000**	**1720275**	
12	6/6/2021	123839	S45	C3310	890200	0	468333	
13	6/6/2021	123840	S54	C2959	986000	0	528980	
14	6/6/2021	123841	S21	C8361	94400	0	53180	
15	6/6/2021	123842	S45	C1842	36500	55000	20696	
16	6/6/2021	123843	S54	C4107	599700	0	276718	
17	6/6/2021	123844	S21	C5205	244900	0	143393	
18	6/6/2021	123845	S45	C7745	63000	0	35102	
19	6/6/2021	123846	S54	C1730	212600	0	117787	
20	6/6/2021	123847	S21	C6292	974700	0	478731	
21	6/6/2021	123848	S45	C2008	327700	0	170968	
22	6/6/2021	123849	S54	C4096	30700	0	18056	
23								

Rysunek 1.12 Makro miało dodawać podsumowanie na końcu pliku, ale rejestrator tak zarejestrował makro, że podsumowanie zawsze wyświetla się w wierszu 11.

Możliwe rozwiązanie: wykorzystywanie odwołań względnych podczas rejestrowania

Domyślnie rejestrator makr rejestruje wszystkie działania jako odwołania *bezwzględne*. Jeśli podczas rejestrowania przechodzimy do wiersza 11, w momencie uruchamiania makro zawsze będzie przechodziło do wiersza 11. Bardzo rzadko takie działanie będzie odpowiednie w przypadku zmiennej liczby wierszy danych. Lepszym rozwiązaniem podczas rejestrowania jest wykorzystanie odwołań względnych.

W makrach zarejestrowanych w trybie odwołań bezwzględnych są zapisywane rzeczywiste adresy wskaźnika komórki (przykładowo A11). W makrach zarejestrowanych w trybie odwołań względnych zapisane są informacje o tym, że wskaźnik komórki powinien przemieścić się o określoną liczbę wierszy i kolumn w odniesieniu do bieżącego położenia. Na przykład, jeśli wskaźnik komórki znajduje się w komórce A1, kod `ActiveCell.Offset(16,1).Select` przeniesie wskaźnik do komórki B17, czyli do komórki, która znajduje się o 16 wierszy w dół i jedną kolumnę w prawo.

Pomimo że rejestrowanie w trybie odwołań względnych jest odpowiednie dla większości sytuacji, czasami podczas rejestrowania może zachodzić jednak potrzeba użycia odwołania bezwzględnego. Ten przykład jest dobrą ilustracją takiej sytuacji: Po dodaniu wiersza podsumowania do zestawu danych trzeba powrócić do wiersza 1. Jeśli w trybie odwołań względnych po prostu klikniemy wiersz 1, program Excel zarejestruje operację zaznaczenia wiersza, który znajduje się 10 wierszy powyżej wiersza bieżącego. Taka operacja zadziała w przypadku pierwszego pliku invoice.txt, ale nie będzie działać w przypadku dłuższego lub krótszego pliku. Poniżej wymieniono dwie metody obejścia problemu:

- Wyłączenie rejestrowania w trybie odwołań względnych, kliknięcie wiersza 1, a następnie ponowne włączenie rejestrowania względnego.
- Zachowanie włączonego rejestrowania w trybie odwołań względnych. Wyświetlenie okna dialogowego Przechodzenie do, poprzez naciśnięcie klawisza F5. Wpisanie A1 i kliknięcie OK. Okno dialogowe Przechodzenie do rejestrowane jest w typowy sposób i następuje przejście do adresu bezwzględnego, nawet jeśli włączone jest rejestrowanie w trybie odwołań względnych. Odmiana tej metody wykorzystana została w poniższej analizie przypadku.

Spróbujmy ponownie przeprowadzić analizę tego samego przypadku – tym razem z wykorzystaniem odwołań względnych. Rozwiązanie będzie znacznie zbliżone do rozwiązania poprawnego.

Studium przypadku: Rejestrowanie makra przy użyciu odwołań względnych

Spróbujmy zarejestrować makro jeszcze raz, tym razem z wykorzystaniem odwołań względnych.

Uwaga: jeśli stosowaliśmy pliki przykładów, należy wykonać następujące działania:

1. Zamknięcie pliku invoice.txt w programie Excel.
2. Zmiana nazwy pliku invoice.txt na invoice2.txt.
3. Zmiana nazwy pliku invoice1.txt na invoice.txt.
4. Powrót do skoroszytu MacroToImportInvoices.xlsm.

Na karcie Deweloper wybieramy przycisk Użyj odwołań względnych, by zmienić tryb rejestrowania. Ustawienie to jest aktywne dopóki nie zostanie wyłączone lub do zamknięcia programu Excel.

W skoroszycie MacroToImportInvoices.xlsm rejestrujemy nowe makro poprzez wybranie polecenia Zarejestruj makro na karcie Deweloper. Nadajemy mu nazwę ImportFakturOdwWzgl i przypisujemy inny klawisz skrótu, np.: Ctrl+Shift+J.

Należy powtórzyć kroki 1 do 11 w poprzednim przykładzie analizy, by zaimportować plik, a następnie wykonać poniższe instrukcje:

1. Naciśnięcie Ctrl+strzałka w dół, by przejść do ostatniego wiersza danych.
2. Ponownie naciśnięcie klawisza strzałki w dół, by przejść do wiersza podsumowania.
3. Wpisanie słowa Razem.
4. Cztery razy naciśnięcie strzałki w prawo, by przejść do kolumny E wiersza podsumowania.
5. Przytrzymanie klawisza Shift naciskając jednocześnie dwa razy strzałkę w prawo, by zaznaczyć zakres E11:G11.
6. Kliknięcie przycisku Autosumowanie.
7. Naciśnięcie klawiszy Shift+spacja, by zaznaczyć cały wiersz. Naciśnięcie klawiszy Ctrl+B, by do wiersza zastosować formatowanie: pogrubienie.
8. Naciśnięcie F5, by wyświetlić okno dialogowe Przechodzenie do.
9. W oknie dialogowym Przechodzenie do, wpisz A1:G1 i kliknij przycisk OK. Nawet jeśli włączone jest rejestrowanie względne, każda nawigacja wykonywana za pośrednictwem okna dialogowego Przechodzenie do zostanie zarejestrowana jako odwołanie bezwzględne. Naciśnięcie klawiszy Ctrl+Home, by przejść do komórki A1.

10. Kliknięcie ikony Pogrubienie, by nagłówki były sformatowane za pomocą pogrubienia.
11. Naciśnięcie Ctrl+*, by zaznaczyć wszystkie dane w bieżącym regionie.
12. Na karcie Narzędzia główne, wybranie opcji Autodopasowanie szerokości kolumn (w menu Formatuj).
13. Zatrzymanie rejestrowania.

Naciskamy klawisze Alt+F11, by przejść do edytora VB i przejrzeć kod. Nowe makro widoczne jest w Module1, poniżej poprzedniego makra.

Jeśli zamkniemy program Excel pomiędzy rejestrowaniem pierwszego i drugiego mara, dla nowo zarejestrowanego modułu program Excel wstawi nowy moduł nazwany Module2.

```
Sub ImportInvoicesRelative()
' ImportInvoicesRelative Macro
' Import. Total Row. Format.
' Keyboard Shortcut: Ctrl+J
Workbooks.OpenText Filename:="C:\data\invoice.txt", _
  Origin:= 437, StartRow:=1, DataType:=xlDelimited, _
  TextQualifier:=xlDoubleQuote, ConsecutiveDelimiter:=False, _
  Tab:=False, Semicolon:=False, Comma:=True, Space:=False, _
  Other:=False, FieldInfo:=Array(Array(1, 3), Array(2, 1), _
  Array(3, 1), Array(4, 1), Array(5, 1), Array(6, 1), _
  Array(7, 1)), TrailingMinusNumbers:=True
Selection.End(xlDown).Select
ActiveCell.Offset(1, 0).Range("A1").Select
ActiveCell.FormulaR1C1 = "Total"
ActiveCell.Offset(0, 4).Range("A1:C1").Select
Selection.FormulaR1C1 = "=SUM(R[-9]C:R[-1]C)"
ActiveCell.Rows("1:1").EntireRow.Select
ActiveCell.Activate
Selection.Font.Bold = True
Application.Goto Reference:="R1C1:R1C7"
Selection.Font.Bold = True
Selection.CurrentRegion.Select
Selection.Columns.AutoFitSelection.Font.Bold = True
End Sub
```

W celu przetestowania makra zamykamy plik invoice.txt bez zapisywania zmian, a następnie uruchamiamy makro za pomocą klawiszy Ctrl+Shift+J. Wszystko wygląda poprawnie – uzyskaliśmy te same wyniki za pomocą makra utworzonego przez rejestrator.

Następny test polega na sprawdzeniu, czy program zadziała następnego dnia, w którym może być więcej wierszy. Jeśli wykorzystywaliśmy pliki przykładów,

zamykamy plik invoice.txt w programie Excel. Zmieniamy nazwę pliku invoice.txt na invoice2.txt i zmieniamy nazwę pliku invoice4.txt na invoice.txt.

Otwieramy plik MacroToImportInvoices i uruchamiamy makro za pomocą skrótu Ctrl+Shift+J. Tym razem wszystko wygląda dobrze, a sumy umieszczone są na właściwych miejscach. Spójrzmy na rysunek 1.13. Czy jest tam coś niezwykłego?

Jeśli nie jesteśmy uważni, możemy wydrukować ten raport dla menedżera. I w takiej sytuacji możemy mieć kłopoty. Kiedy spojrzymy na komórkę E19, widzimy, że program Excel wstawił zielony trójkąt, by zwrócić uwagę na komórkę. Jeśli spróbowalibyśmy sprawdzać to w programie Excel 95 lub Excel 97, kiedy nie działały jeszcze funkcje znaczników inteligentnych (SmartTag), nic nie wskazywałoby, że coś nie jest w porządku.

Kiedy przeniesiemy wskaźnik położenia do komórki E19, w pobliżu komórki pojawi się wskaźnik ostrzeżenia, który informuje, że w formule nie powiodło się dołączenie przylegających komórek. Przyglądając się paskowi formuły, zauważymy, że makro posumowało tylko wiersze od 10 do 18. Ani rejestrowanie z użyciem odwołań względnych, ani z użyciem odwołań bezwzględnych nie jest na tyle „sprytne", by powielić logikę przycisku Autosumowanie.

Załóżmy, że w danym dniu mamy mniej rekordów faktur. Program Excel „nagrodziłby" nas nielogiczną formułą =SUM(E6:E1048574), jak ilustruje to rysunek 1.14. Ponieważ formuła ta byłaby w komórce E7, w pasku stanu pojawiłoby się ostrzeżenie o odwołaniu cyklicznym.

Uwaga Aby wypróbować to samemu, zamykamy plik invoice.txt w programie Excel. Zmieniamy nazwę pliku invoice.txt na invoice2.txt i zmieniamy nazwę pliku invoice4.txt na invoice.txt.

Jeśli ktoś spróbował używać rejestratora makr, prawdopodobnie napotkał problemy podobne do tych, które opisano w analizie ostatnich dwóch przypadków. Chociaż jest to frustrujące, pozytywne jest to, że rejestrator makr pozwala zrealizować 95% zadania związanego z utworzeniem poprawnego makra.

Naszym zadaniem jest rozpoznanie miejsc, w których działanie rejestratora makr może zawieść, a następnie zmodyfikowanie kodu VBA w celu poprawienia jednego czy kilku wierszy i uzyskania poprawnie działającego makra. Korzystając z własnej inwencji, możemy tworzyć ciekawe makra, które przyspieszą nasze codzienne działania.

	A	B	C	D	E	F	G	H
1	InvDate	InvNbr	RepNbr	CustNbr	ProdRevenue	ServRevenue	ProdCost	
2	6/6/2021	123829	S21	C8754	21000	0	9875	
3	6/6/2021	123830	S45	C3390	188100	0	85083	
4	6/6/2021	123831	S54	C2523	510600	0	281158	
5	6/6/2021	123832	S21	C5519	86200	0	49967	
6	6/6/2021	123833	S45	C3245	800100	0	388277	
7	6/6/2021	123834	S54	C7796	339000	0	195298	
8	6/6/2021	123835	S21	C1654	161000	0	90761	
9	6/6/2021	123836	S45	C6460	275500	10000	146341	
10	6/6/2021	123837	S54	C5143	925400	0	473515	
11	6/6/2021	123838	S21	C7868	148200	0	75700	
12	6/6/2021	123839	S45	C3310	890200	0	468333	
13	6/6/2021	123840	S54	C2959	986000	0	528980	
14	6/6/2021	123841	S21	C8361	94400	0	53180	
15	6/6/2021	123842	S45	C1842	36500	55000	20696	
16	6/6/2021	123843	S54	C4107	599700	0	276718	
17	6/6/2021	123844	S21	C5205	244900	0	143393	
18	6/6/2021	123845	S45	C7745	63000	0	35102	
19	6/6/2021	123846	S54	C1730	212600	0	117787	
20	6/6/2021	123847	S21	C6292	974700	0	478731	
21	6/6/2021	123848	S45	C2008	327700	0	170968	
22	6/6/2021	123849	S54	C4096	30700	0	18056	
23	Total				2584200	55000	1314631	

Rysunek 1.13 Rezultat uzyskany po uruchomieniu makra w trybie odwołań względnych.

E7 =SUM(E6:E1048574)

	A	B	C	D	E	F	G
1	InvDate	InvNbr	RepNbr	CustNbr	ProdRevenue	ServRevenue	ProdCost
2	6/7/2021	123850		C1654	161000	0	90761
3	6/7/2021	123851		C6460	275500	10000	146341
4	6/7/2021	123852		C5143	925400	0	473515
5	6/7/2021	123853		C7868	148200	0	75700
6	6/7/2021	123854		C3310	890200	0	468333
7	Total				0	0	0

Rysunek 1.14 Wynik działania makro w trybie odwołań względnych przy mniejszej liczbie rekordów faktur.

Jeśli ktoś jest podobny do mnie, to pewnie teraz w duchu przeklina firmę Microsoft. Zmarnowaliśmy sporo czasu w ciągu ostatnich kilku dni i żadne makro nie zadziałało. Na domiar złego, z tego typu sytuacjami bez trudu radził sobie rejestrator makr Lotus 1-2-3 powstały w 1983 roku. 33 lata temu Mitch Kapor rozwiązał problem, z którym firma Microsoft nie może sobie poradzić dzisiaj.

Czy wiecie, że do wersji Excel 97 program Microsoft Excel „potajemnie" obsługiwał makra Lotusa? Fakt ten odkryłem zaraz po tym, jak firma Microsoft zaprzestała

obsługi programu Excel 97. Wiele firm przeszło wówczas na Excel XP, który już nie obsługiwał makr Lotus 1-2-3. Sporo firm zleciło nam wtedy zadanie konwersji starych makr Lotusa 1-2-3 na Excel VBA. Interesujące jest to, że w wersjach Excel 5, Excel 95 i Excel 97 firma Microsoft oferowała interpreter, który potrafił poprawnie obsłużyć makra programu Lotus, które bezbłędnie rozwiązują ten problem, podczas gdy ich własny rejestrator tego nie robił (i nadal nie potrafi).

Podczas rejestrowania nigdy nie używaj przycisku Autosumowanie lub Szybka analiza

Opisany problem dotyczący rejestrowania funkcji Autosumowanie można jednak rozwiązać, przy czym należy zapamiętać, że rejestrator makr nigdy poprawnie nie zarejestruje działania przycisku Autosumowanie.

Jeśli rozpoczniemy działanie w komórce E99 i klikniemy przycisk Autosumowanie, program Excel zacznie skanowanie od komórki E98 w górę do czasu, aż odnajdzie komórkę tekstową, pustą komórkę lub formułę. Następnie zaproponuje formułę, która sumuje wszystkie komórki pomiędzy komórką bieżącą a tą znalezioną.

Rejestrator makr rejestruje jednak określony wynik tego wyszukiwania w momencie, w którym zarejestrowano makro. Zamiast zarejestrować coś w stylu „wykonaj zwykłą logikę przycisku Autosumowanie", rejestrator makr wstawia pojedynczy wiersz kodu sumujący poprzednie 98 komórek.

W programie Excel 2013 wprowadzono funkcję Szybka analiza. Zaznaczamy zakres E2:G99; otwieramy ikonę Szybka analiza, która wyświetlana jest poniżej prostokątnego zaznaczenia z prawej jego strony; wybieramy opcję Suma; i uzyskujemy poprawne sumy w wierszu 100. Rejestrator makr na sztywno zakoduje formuły, by zawsze wyświetlane były w wierszu 100 i zawsze sumowały wiersze od 2 do 99.

Dość dziwacznym obejściem tego problemu jest wpisanie funkcji SUMA, która wykorzystuje połączenie odwołań względnych i bezwzględnych. Jeśli wpiszemy `=SUMA(E$2:E10)` w czasie działania rejestratora makr, program Excel poprawnie doda kod, który zawsze będzie sumował komórki, począwszy od ustalonego wiersza 2 w dół do względnego odwołania, które jest bezpośrednio nad komórką bieżącą.

Poniżej zaprezentowano uzyskany kod:

```
Sub FormatInvoice3()
Sub FormatInvoice3()
' Makro FormatInvoice3
' Importowanie. Podsumowanie. Formatowanie.
' Skrót klawiszowy: Ctrl+K
Workbooks.OpenText Filename:="C:\Data\invoice.txt", _
Origin:=437, StartRow:=1, DataType:=xlDelimited, _
```

```
    TextQualifier:=xlDoubleQuote, ConsecutiveDelimiter:=False, _
    Tab:=False, Semicolon:=False, Comma:=True , Space:=False, _
    Other:=False, FieldInfo:=Array(Array(1, 3), Array(2, 1), _
    Array(3, 1), Array(4, 1), Array(5, 1), Array(6, 1), _
    Array(7, 1)), TrailingMinusNumbers:=True
Selection.End(xlDown).Select
ActiveCell.Offset(1, 0).Range("A1").Select
ActiveCell.FormulaR1C1 = "Total"
ActiveCell.Offset(0, 4).Range("A1").Select
Selection.FormulaR1C1 = "=SUM(R2C:R[-1]C)"
Selection.AutoFill Destination:=ActiveCell.Range("A1:C1"), _
    Type:=xlFillDefault
ActiveCell.Range("A1:C1").Select
ActiveCell.Rows("1:1").EntireRow.Select
ActiveCell.Activate
Selection.Font.Bold = True
Application.Goto Reference:="R1C1:R1C7"
Selection.Font.Bold = True
Selection.CurrentRegion.Select
Selection.Columns.AutoFit
End Sub
```

To trzecie makro działa poprawnie dla zbioru danych dowolnego rozmiaru.

Cztery wskazówki dotyczące używania rejestratora makr

Bardzo rzadko zdarza się sytuacja, by 100% z zarejestrowanych makr zadziałało. Będziemy jednak znacznie bliżej tego celu, jeśli zastosujemy się do czterech wskazówek prezentowanych poniżej.

Wskazówka 1: Włączaj ustawienie Użyj odwołań względnych

Microsoft powinien nadać temu ustawieniu status ustawienia domyślnego. Włącz ustawienie i nie wyłączaj go podczas rejestrowania makr.

Wskazówka 2: Używaj specjalnych klawiszy nawigacyjnych do przechodzenia na koniec zbioru danych

Jeśli jesteś na początku zbioru danych i chcesz przejść do ostatniej komórki z danymi, naciśnij Ctrl+strzałka w dół lub naciśnij klawisz End, a następnie strzałkę w dół.

Aby przejść do ostatniej kolumny w bieżącym wierszu zbioru danych, naciśnij klawisze Ctrl+strzałka w prawo lub naciśnij klawisz End, a następnie strzałkę w prawo. Dzięki użyciu tych klawiszy nawigacyjnych można przechodzić na koniec zbioru danych niezależnie od tego, z ilu wierszy lub kolumn składa się zbiór danych określonego dnia.

Używaj klawiszy Ctrl+*, by zaznaczać bieżący region wokół komórki aktywnej. Zakładając, że dane nie zawierają pustych wierszy lub kolumn, ta kombinacja klawiszy powoduje zaznaczenie całego zestawu danych.

Wskazówka 3: Podczas rejestrowania makra nigdy nie używaj przycisku Autosumowanie

Rejestrator makr nie potrafi zarejestrować „istoty" działania przycisku Autosumowanie. Zamiast tego koduje „na sztywno" formułę, wynikającą z naciśnięcia tego przycisku. Formuła ta nie zadziała poprawnie, jeśli zbiór danych będzie zawierał mniej lub więcej rekordów.

Zamiast tego należy wpisać formułę z pojedynczym znakiem dolara, na przykład `=SUMA(E$2:E10)`. W takim przypadku rejestrator makr zarejestruje pierwsze odwołanie E$2 jako ustalone i rozpocznie określać zakres sumowania bezpośrednio pod nagłówkami w pierwszym wierszu. Zakładając, że bieżącą komórką jest E11, rejestrator makr rozpozna E10 jako odwołanie względne wskazujące bezpośrednio powyżej bieżącej komórki.

Wskazówka 4: Próbuj rejestrować makra innymi metodami, jeśli zastosowana metoda nie działa

W programie Excel najczęściej istnieje wiele sposobów wykonywania tych samych zadań. Jeśli napotkamy błędny kod generowany przez jeden sposób, warto spróbować innego. Ponieważ w zespole menedżerów projektów programu Excel jest 16 osób, prawdopodobnie każda metoda była programowana przez inną grupę. W jednej z analiz przypadków w tym rozdziale, jedno zadanie wiązało się z zastosowaniem do wszystkich komórek funkcji Autodopasowanie szerokości kolumn. Niektóre osoby mogą nacisnąć klawisze Ctrl+A, by zaznaczyć wszystkie komórki, a inne mogą nacisnąć klawisze Ctrl+*. Począwszy od wersji Excel 2007, nie działa kod generowany przez naciśnięcie klawiszy Ctrl+A przy włączonym trybie odwołań względnych. Kod związany z sekwencją Ctrl+* jest bardzo stary i nadal działa we wszystkich przypadkach.

Następne kroki

W rozdziale 2 „Skoro nazywa się BASIC, dlaczego nie wygląda znajomo?", przeanalizujemy trzy makra, które zarejestrowaliśmy w tym rozdziale i postaramy się je zrozumieć. Kiedy już nauczymy się czytać kod VBA, naturalnym krokiem będzie poprawienie zarejestrowanego kodu lub napisanie poprawnego od początku. Warto przeczytać kolejny rozdział – wkrótce przekonamy się, że VBA to dobre rozwiązanie i nauczymy się pisać dobrze działający kod.

ROZDZIAŁ 2

Skoro nazywa się BASIC, dlaczego nie wygląda znajomo?

W tym rozdziale:

- „Części mowy" języka VBA
- Język VBA naprawdę nie jest trudny
- Analiza kodu zarejestrowanego makra: korzystanie z edytora VB i tematów pomocy
- Stosowanie narzędzi debugowania do analizy zarejestrowanego kodu
- Narzędzie Object Browser: ostateczne źródło
- Siedem wskazówek poprawiania zarejestrowanego kodu

Jak wspomniałem w rozdziale 1 „Zwiększanie możliwości programu Excel za pomocą języka VBA", jeśli ktoś zapoznał się z językiem proceduralnym, takim jak BASIC czy COBOL, może być trochę zdezorientowany przy przeglądaniu kodu w języku VBA. Wprawdzie skrót VBA rozwija się jako *Visual Basic for Applications*, jest to jednak obiektowa wersja języka BASIC. Poniżej zamieszczono przykład kodu w języku VBA:

```
Selection.End(xlDown).Select
Range("A11").Select
ActiveCell.FormulaR1C1 = "Razem"
Range("E11").Select
Selection.FormulaR1C1 = _
    "=SUM(R[-9]C:R[-1]C)"
Selection.AutoFill _
Destination:=Range("E11:G11"), _
    Type:=xlFillDefault
```

Prawdopodobnie kod ten nie ma żadnego sensu dla osób, które poznały jedynie języki proceduralne. Niestety, najczęściej pierwszym wprowadzeniem do programowania w szkole (zakładając, że ktoś ma ponad 40 lat) był język proceduralny. Oto przykład kodu napisanego w języku BASIC:

```
For x = 1 to 10
    Print Rpt$(" ",x);
```

```
    Print "*"
Next x
```

Jeśli uruchomimy ten kod, na ekranie uzyskamy piramidę gwiazdek:

```
*
  *
    *
      *
        *
          *
            *
```

Jeśli ktoś poznał programowanie proceduralne, prawdopodobnie przygląda się wcześniejszemu kodowi i zastanawia się, co tu się dzieje, ponieważ języki proceduralne bardziej przypominają język angielski niż jest to w przypadku języków obiektowych. Instrukcja `Print "Hello World"` jest zgodna z formatem czasownik-obiekt, który jest zgodny z ogólną zasadą mówienia. Oderwijmy się na chwilę od programowania i przyjrzyjmy się konkretnemu przykładowi.

„Części mowy" języka VBA

Jeśli chcielibyśmy napisać kod dla instrukcji gry w piłkę za pomocą języka BASIC, instrukcja kopnięcia piłki mogłaby wyglądać następująco:

```
"Kick the Ball"
```

To tak samo, jak mówimy! I coś takiego ma sens. Mamy czasownik kopnij (kick), a następnie rzeczownik piłka (ball). Kod języka BASIC w poprzednim podrozdziale miał czasownik drukuj (Print) i rzeczownik gwiazdka (*). Życie jest piękne.

I tu mamy problem: język VBA nie działa w ten sposób. W rzeczywistości nie działa w ten sposób żaden język obiektowy. W języku obiektowym, obiekty (rzeczowniki) są najważniejsze, i stąd nazwa tych języków: obiektowe. Jeśli zamierzaliśmy napisać kod instrukcji do gry w piłkę za pomocą VBA, podstawowa struktura mogłaby mieć taką składnię:

```
Ball.Kick
```

Mamy rzeczownik piłka `(Ball)`, który jest na początku. W języku VBA jest to *obiekt*. Następnie mamy czasownik kopnij `(Kick)`. W języku VBA jest to *metoda*.

Podstawowa struktura języka VBA to seria wierszy kodu o poniższej składni:

```
Obiekt.Metoda
```

Nie trzeba mówić, że nie jest to język angielski. Jeśli ktoś w szkole uczył się języka romańskiego, pamięta, że korzystają one z konstrukcji „rzeczownik-przymiotnik", ale nikt nie używał składni „rzeczownik-czasownik", prosząc kogoś, by coś zrobił:

```
woda.pić
jedzenie.jeść
dziewczyna.całować
```

I to jest przyczyna, dlaczego język VBA jest niezrozumiały dla osób, które poprzednio uczyły się programowania proceduralnego.

Rozwińmy analogię trochę bardziej. Wyobraźmy sobie, że chodzimy po trawiastym boisku, a przed nami widzimy pięć piłek. Jest to piłka do piłki nożnej (soccer), koszykówki, baseballa, kula do gry w kręgle i piłka do tenisa. Chcemy poinstruować dzieciaka w naszym zespole piłki nożnej, by kopnął piłkę (do piłki nożnej) „kick the soccer ball".

Jeśli powiemy, aby kopnął piłkę (czyli `ball.kick`), nie będziemy mieli pewności, która z pięciu piłek zostanie kopnięta przez dziecko. Być może kopnie piłkę, która jest najbliżej, co może być problematyczne, jeśli będzie stał naprzeciw kuli do kręgli.

W przypadku większości rzeczowników, czyli obiektów w języku VBA, istnieje zbiór tego obiektu. Pomyślmy o programie Excel. Jeśli mamy jeden wiersz (*row*), możemy mieć kilka wierszy (*rows*). Jeśli mamy jedną komórkę (*cell*), możemy mieć grupę komórek (*cells*). Jeśli mamy jeden arkusz (*worksheet*), również możemy mieć wiele arkuszy (*worksheets*).

W języku angielskim jedyna różnica pomiędzy obiektem a zbiorem tych obiektów polega na dodaniu litery *s* do nazwy obiektu:

`Row` zmienia się na `Rows`

`Cell` zmienia się na `Cells`

`Ball` zmienia się na `Balls`

Jeśli odwołujemy się do czegoś, co jest zbiorem (czy inaczej kolekcją), musimy poinformować język programowania, do którego elementu się odnosimy. Istnieje kilka metod zrealizowania tego zadania. Możemy odnosić się do elementu za pomocą liczb. Przykładowo, jeśli piłka nożna jest drugą w kolejności piłką, możemy „powiedzieć":

```
Balls(2).Kick
```

Metoda działa dobrze, ale może to być niebezpieczne dla programu. Na przykład, to dobrze zadziała we wtorek, ale jeśli przyjdziemy na boisko w środę i ktoś zmieni kolejność piłek, polecenie Balls(2).Kick może mieć bolesne konsekwencje.

Znacznie bezpieczniejsza metoda polega na używaniu nazw obiektów w kolekcji, która korzysta z takiej składni:

```
Balls("Soccer").Kick
```

Za pomocą tej metody zawsze wiemy, że kopnięta będzie piłka nożna.

Jak dotąd jest dobrze. Wiemy, że piłka będzie kopnięta i wiemy, że kopnięta będzie piłka nożna. Dla większości czasowników, czyli metod w języku Excel VBA, istnieją parametry, określające, jak ma być wykonane to działanie. Parametry te spełniają rolę

przysłówków. Przykładowo, może chcieć, by piłka nożna była kopnięta silnie i w lewą stronę. W tym przypadku, metoda będzie miała kilka parametrów, które poinformują, w jaki sposób program powinien wykonać metodę (w poniższym przykładzie Direction oznacza kierunek, Force siłę, a Hard mocno):

```
Balls("Soccer").Kick Direction:=Left, Force:=Hard
```

W kodzie VBA, połączenie dwukropka i znaku równości (:=) wskazuje, że patrzymy na parametry, określające, jak powinna być wykonywana czynność (czyli czasownik).

Czasami metoda będzie miała listę 10 parametrów, a niektóre z nich są opcjonalne. Przykładowo, jeśli metoda `Kick` posiada parametr `Elevation` (wysokość), otrzymamy taki wiersz kodu:

```
Balls("Soccer").Kick Direction:=Left, Force:=Hard, Elevation:=High
```

W tym miejscu mamy pewne zagmatwanie: Każda metoda posiada domyślną kolejność swoich parametrów. Jeśli nie jesteśmy skrupulatnym programistą i znamy kolejność parametrów, możemy opuścić nazwy parametrów, jak w poniższym przykładzie, który jest równoważny poprzedniej linii kodu:

```
Balls("Soccer").Kick Left, Hard, High
```

Sposób ten nie ułatwia zrozumienia kodu. Bez znaków `:=` nie jest oczywiste, że są to parametry. Jeśli nie znamy kolejności parametrów, możemy nie zrozumieć tego zapisu. Dość prosto zrozumieć zapis w przypadku `Left`, `Hard` i `High`, ale jeśli są to następujące parametry:

```
ActiveSheet.Shapes.AddShape type:=1, Left:=10, Top:=20, _
    Width:=100, Height:=200
```

poniższy „skrótowy” zapis nie będzie zrozumiały:

```
ActiveSheet.Shapes.AddShape 1, 10, 20, 100, 200
```

Poprzedni wiersz kodu jest poprawny. O ile jednak nie znamy domyślnej kolejności parametrów tej metody `Add`, czyli `Type`, `Left`, `Top`, `Width` i `Height`, kod ten nie ma sensu. Domyślna kolejność parametrów danej metody to kolejność pokazywana w temacie Pomocy opisującym tę metodę.

Żeby jeszcze bardziej skomplikować życie, możemy rozpocząć specyfikowanie parametrów w ich porządku domyślnym bez nazywania parametrów, a następnie możemy rozpocząć nazywanie parametrów, jeśli akurat użyjemy parametru, który nie spełnia domyślnej kolejności. Przykładowo, jeśli chcemy kopnąć piłkę w lewo (left) i wysoko (high), ale nie ma znaczenia dla nas siła kopnięcia (czyli chcemy użyć wartości domyślnej dla parametru siły), poniższe dwie instrukcje mają identyczne działanie:

```
Balls("Soccer").Kick Direction:=Left, Elevation:=High
Balls("Soccer").Kick Left, Elevation:=High
```

Warto jednak pamiętać, że jeśli rozpoczniemy nazywanie parametrów, muszą one być również nazywane w pozostałej części wiersza.

Niektóre metody działają na swój własny sposób. Aby zasymulować naciśnięcie klawisza F9, używamy tego kodu:

```
Application.Calculate
```

Inne metody wykonują działanie i coś tworzą. Na przykład, za pomocą poniższego kodu możemy dodać arkusz:

```
Worksheets.Add Before:=Worksheets(1)
```

Ponieważ jednak metoda `Worksheets.Add` tworzy nowy obiekt, możemy do zmiennej przypisać wynik działania tej metody. W tym przypadku, musimy ująć parametry nawiasami:

```
Set MyWorksheet = Worksheets.Add(Before:=Worksheets(1))
```

Potrzebne jest jeszcze poznanie ostatniego fragmentu gramatyki: przymiotników. Podobnie jak przymiotnik opisuje rzeczownik, *właściwości* opisują obiekt. Ponieważ jesteśmy fanami programu Excel, przejdziemy z analogii piłkarskich do analogii z programem Excel. Istnieje obiekt do opisywania komórki aktywnej i na szczęście ma intuicyjną nazwę:

```
ActiveCell
```

Załóżmy, że chcemy zmienić kolor komórki aktywnej na czerwony. Istnieje właściwość nazwana `Interior.Color` dla komórki, która używa złożonego kodu. Możemy jednak zmienić kolor komórki na czerwony za pomocą poniższego kodu:

```
ActiveCell.Interior.Color = 255
```

Widzimy, jak może to być mylące. Mamy poprzednią konstrukcję *rzeczownik*-kropka-*coś*, ale tym razem jest to raczej `Obiekt.Właściwość`, a nie `Obiekt.Metoda`. Sposób rozróżniania tego jest dość subtelny: Przed znakiem równości brak jest dwukropka. Prawie zawsze wartość właściwości jest ustawiana tak samo jak coś innego lub inaczej mówiąc wartości właściwości przypisywana jest inna wartość (stąd znak równości).

Aby kolor tej komórki był taki sam, jak dla komórki A1, możemy „powiedzieć":

```
ActiveCell.Interior.Color = Range("A1").Interior.Color
```

`Interior.Color` to właściwość. Poprzez zmianę wartości właściwości zmieniamy wygląd rzeczy. Trochę to dziwne: Zmieniając przymiotnik, tak naprawdę modyfikujemy komórkę. Ludzie powiedzieliby „Pokoloruj komórkę na czerwono", natomiast w języku VBA zdanie takie ma postać:

```
ActiveCell.Interior.Color = 255
```

W tabeli 2.1 zamieszczono podsumowanie „części mowy" języka VBA.

Tabela 2.1 Części języka programowania VBA

Składnik VBA	Analogia	Uwagi
Obiekt	Rzeczownik	Przykład: komórka (Cell) lub arkusz (Sheet).
Kolekcja	Rzeczownik w liczbie mnogiej	Zazwyczaj określa, który z obiektów: `Sheets(1)`.
Metoda	Czasownik	Występuje w postaci: `Obiekt.Metoda`.
Parametr	Przysłówek	Lista parametrów po metodzie. Nazwa parametru oddzielana jest od wartości za pomocą znaków `:=`
Właściwość	Przymiotnik	Można ustawić właściwość (na przykład activecell.height=10) lub zapisać wartość właściwości (na przykład x = activecell. height).

Język VBA naprawdę nie jest trudny

Informacja o tym, czy mamy do czynienia z właściwościami czy z metodami ułatwia określenie prawidłowej składni kodu. Nie trzeba przejmować się, że w tym momencie wygląda to trochę zagmatwanie. Kiedy piszemy kod VBA od początku, wystarczy wiedzieć, czy proces zmieniający kolor komórki na żółty wymaga czasownika, czy przymiotnika. Innymi słowy, czy jest to metoda, czy właściwość?

I to jest moment, w którym rejestrator makr jest szczególnie przydatny. Jeśli nie wiem, jak zakodować jakieś działanie, rejestrujemy krótkie makro, przyglądamy się zarejestrowanemu kodowi i domyślamy się jego działania.

Pliki pomocy VBA: Stosowanie klawisza F1 do wyszukiwania potrzebnych informacji

Moduł pomocy programu Excel VBA to zdumiewająca funkcja, która zakłada, że jesteśmy połączeni z Internetem. Jeśli planujemy napisanie makr VBA, bezwzględnie *musimy* mieć dostęp do zainstalowanych plików pomocy VBA. Poniższa procedura opisuje, jak prosto jest uzyskać pomoc w programie VBA:

1. Otwórz program Excel i przejdź do edytora VB, naciskając klawisze Alt+F11. W menu Insert wybierz opcję Module.
2. Wpisz poniższe trzy wiersze kodu:

```
Sub Test()
MsgBox "Hello World!"
End Sub
```