

» Idź do

- Spis treści
- Przykładowy rozdział

» Katalog książek

- Katalog online
- Zamów drukowany katalog

» Twój koszyk

- Dodaj do koszyka

» Cennik i informacje

- Zamów informacje o nowościach
- Zamów cennik

» Czytelnia

- Fragmenty książek online

» Kontakt

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl
© Helion 1991-2010

Więcej niż Excel 2007. 166 gotowych rozwiązań i trików w języku VBA

Autor: Mirosław Lewandowski
ISBN: 978-83-246-1907-8
Format: 158×235, stron: 224



Dopasuj Excela do swoich wymagań!

- Narzędzia i zasady używania VBA – okno edytora, certyfikaty, zabezpieczenia
- Makra i kod VBA – tworzenie, uruchamianie, przykłady konkretnych działań
- Podstawy VBA – typy danych, operacje na łańcuchach, interakcje z użytkownikiem

Bez umiejętności posługiwania się arkuszem kalkulacyjnym Excela trudno dziś wyobrazić sobie jakąkolwiek pracę biurową. Jego wielofunkcyjność i elastyczność są imponujące, ale wciąż jeszcze można natknąć się na obszary nie do końca odpowiadające specyficznym potrzebom danej firmy, stanowiska czy pracownika. Owszem, ogólnie wiadomo, że wbudowany w program język VBA oferuje możliwość znacznego zmodyfikowania ustawień i zapisania dodatkowych funkcji albo sekwencji działań, jednak nie wszyscy potrafią sprawnie wykorzystać to narzędzie.

Książka „VBA dla Excela 2007. 166 praktycznych przykładów” to cenny poradnik dla tych, którym nie wystarcza znajomość podstawowych funkcji Excela. Znajdziesz w niej całą masę przykładów kodu napisanego w języku VBA – zastosowanie któregośkolwiek z nich pozwala poczuć różnicę w działaniu programu. Jeśli więc chciałbyś zarejestrować i uruchomić własne makro, zdefiniować funkcję, dodać przycisk, jednym ruchem zamknąć wszystkie skoroszyty, wykonać jakąś operację na tablicach albo wyświetlić określony komunikat, możesz zrobić to bez konieczności zagłębiania się w tajniki programowania. Prosto, szybko, skutecznie.

- Narzędzie VBA w Excelu 2007
- Zabezpieczenia Excela i Windows, certyfikaty cyfrowe
- Rejestrator makr – uruchamianie makr i funkcji
- Definiowanie własnej funkcji i określanie właściwości makra/funkcji
- Wymuszanie zezwolenia na makra przy uruchomieniu skoroszytu
- Dodawanie przycisku lub ikon do wstążki i komentarzy do komórek
- Ochrona przed zmianą nazwy arkusza i dynamiczne ukrywanie wierszy
- Hipertączę w formularzu i oknie komunikatu
- Odczytywanie danych o systemie
- Blokowanie wydruków i dostępu do makr
- Arkusz ofert
- Typy danych, zmienne, stałe i tablice
- Typy zmiennych i stałych, funkcje i operatory matematyczne
- Interakcja z użytkownikiem i operacje na łańcuchach
- Dodatki

Rozszerz potęgę Excela – wykorzystaj gotowe kody VBA

Spis treści

Wstęp	5
Rozdział 1. Excel 2007 — nowe przyzwyczajenie	7
Rozdział 2. Narzędzie VBA w Excelu 2007	9
Okno edytora VBA	9
Rozdział 3. Zanim zacniemy	15
Słowo o zabezpieczeniach	15
Zabezpieczenia Excela	17
Certyfikaty cyfrowe	20
Zabezpieczenia Windows	21
Rozdział 4. O VBA bez VBA	25
Rejestrator makr	25
Przykład 1. Rejestrowanie makr	26
Przykład 2. Modyfikacja kodu z rejestratora	27
Uruchamianie makr i funkcji	30
Przykład 3. Uruchamianie makr za pomocą okna dialogowego Makro	30
Przykład 4. Uruchamianie makr za pomocą kombinacji klawiszy	31
Przykład 5. Wstawianie przycisku makra na pasku narzędzi Szybki dostęp	32
Przykład 6. Uruchamianie makra za pomocą przycisku makra wewnątrz arkusza	34
Przykład 7. Uruchamianie makra za pomocą zdarzenia	34
Przykład 8. Zakładka Dodatki na wstążce	35
Przykład 9. Uruchamianie makra na podstawie analizy komórki	38
Przykład 10. Uruchamianie makr za pomocą formantu	39
Przykład 11. Uruchamianie makra na podstawie analizy komórki. Sposób drugi	39
Przykład 12. Uruchamianie makra na podstawie analizy wartości komórki. Sposób trzeci	41
Rozdział 5. Przykłady	43
Przykład 13. Definiowanie własnej funkcji. Konwertuj	43
Przykład 14. Definiowanie własnej funkcji. Dzień tygodnia	44
Przykład 15. Dzień tygodnia dużo łatwiej	45

Przykład 16. Funkcja Słownie	45
Przykład 17. Funkcja Słownie — wersja 2	51
Przykład 18. Funkcja Słownie z podprogramem	54
Przykład 19. Określenie właściwości makra/funkcji za pomocą edytora	58
Przykład 20. Określenie właściwości makra/funkcji za pomocą VBA	60
Przykład 21. Wstawianie funkcji arkuszowej do arkusza za pomocą VBA	61
Przykład 22. Wymuszenie włączenia dodatku	62
Przykład 23. Zamknięcie dodatku	62
Przykład 24. Zamknięcie dodatku uruchomionego przez nasz skoroszyt	62
Przykład 25. Rozwiązanie drugie — stabilniejsze	63
Przykład 26. Formatowanie warunkowe w zależności od wartości z innej komórki	65
Przykład 27. Ile mamy otwartych skoroszytów?	65
Przykład 28. Program w wersji trial	66
Przykład 29. Wymuszanie zezwolenia na makra przy uruchomieniu skoroszytu	67
Przykład 30. Dodanie przycisku do wstążki w zakładce Dodatki	68
Przykład 31. Dodawanie komentarzy do komórki w zależności od warunków	70
Przykład 32. Poszukiwanie ostatniego wiersza	71
Przykład 33. Ochrona przed zmianą nazwy arkusza	72
Przykład 34. Dynamiczne ukrywanie wierszy	72
Przykład 35. Jednorazowe losowanie — sposób 1	74
Przykład 36. Jednorazowe losowanie — sposób 2	75
Przykład 37. Automatyczne uruchamianie ostatnich skoroszytów	76
Przykład 38. Uruchamianie skoroszytów przy starcie programu	76
Przykład 39. Uruchamianie skoroszytów przy starcie programu — wersja non VBA	77
Przykład 40. Uruchamianie wielu skoroszytów jednocześnie	78
Przykład 41. „Pisanie” formularza	79
Przykład 42. Hiperłącze w oknie komunikatu	82
Przykład 43. Hiperłącze w formularzu	84
Przykład 44. Najprostszy komunikat	86
Przykład 45. Komunikat bardziej skomplikowany	86
Przykład 46. Okno dialogowe	87
Przykład 47. Porada dnia	87
Przykład 48. Porada dnia z opcjami w User form — część I	88
Przykład 49. Porada dnia z opcjami w User form — część II	89
Przykład 50. Czyszczenie formatowania w komórkach niezawierających danych	91
Przykład 51. Wyświetlanie postępu obliczeń na pasku stanu	92
Przykład 52. Wyświetlanie postępu obliczeń na formularzu	93
Przykład 53. Jeszcze trochę zabawy	95
Przykład 54. Odczytywanie danych o systemie — nazwa komputera	95
Przykład 55. Odczytywanie danych o systemie — nazwa użytkownika	96
Przykład 56. Odczytywanie danych o systemie — rozdzielczość karty graficznej	96
Przykład 57. Odczytywanie danych o systemie — wielkość okna aplikacji	97

Przykład 58. Odczytywanie danych o systemie — prośba o zmianę wielkości okna	98
Przykład 59. Wyświetlanie okna w pełnym rozmiarze bez pytania	99
Przykład 60. Zamykanie wszystkich skoroszytów	99
Przykład 61. Blokowanie wydruków	100
Przykład 62. Blokada dostępu do makr	100
Przykład 63. Import tekstu z zewnętrznego pliku tekstowego	100
Przykład 64. Pobieranie danych z zamkniętego skoroszytu Excela	101
Przykład 65. Pobieranie danych z innego skoroszytu — wersja 2	102
Przykład 66. Dynamiczne tworzenie makra	102
Przykład 67. Usuwanie kodu z modułu	103
Przykład 68. Zastępowanie modułu innym modułem	103
Przykład 69. Przekazywanie modułu między plikami	104
Przykład 70. Sprawdzanie ustawień dostępu do VBProject	105
Przykład 71. Uruchamianie innych aplikacji Windows	105
Przykład 72. Uruchamianie okien panelu sterowania	105
Przykład 73. Zapis kopii zapasowej pliku z hasłem	105
Przykład 74. Ostatnie zapisanie arkusza	107
Przykład 75. Przygotowanie wydruku z danymi z arkusza innego niż widoczny	107
Przykład 76. Nie wszystko, co automatyczne, to VBA	108
Przykład 77. Automatyka formantów	110
Przykład 78. Wstążka w Office 2007. To także nie VBA, ale...	110
Przykład 79. Dodawanie ikon do wstążki	113
Przykład 80. Arkusz ofert — przykład, o który najczęściej pytają czytelnicy	114
Przykład 81. Arkusz ofert — otwieranie bazy	114
Przykład 82. Arkusz ofert — część 2	115
Przykład 83. Arkusz ofert — część 3	116
Przykład 84. Arkusz ofert — część 4	117
Przykład 85. Arkusz ofert — część 5	118
Przykład 86. Arkusz ofert — czynności końcowe	118

Rozdział 6. Trochę podstaw o VBA 119

Typy zmiennych i stałych	119
Przykład 87. Typy zmiennych	120
Przykład 88. Deklarowanie zmiennych	121
Własny typ zmiennej	121
Przykład 89. Deklaracja własnego typu zmiennej	121
Przykład 90. Deklarowanie zmiennych za pomocą Const, Dim i Static ...	123
Przykład 91. Słowa kluczowe Public i Private	124
Przykład 92. Opcje modułu	124
Przykład 93. Opcja Option Compare	125
Przykład 94. Konwersja typów danych	125
Przykład 95. Funkcje FormatNumber, FormatCurrency, FormatPercent	128
Przykład 96. Funkcja FormatDateTime	129
Przykład 97. Słowo kluczowe ReDim	129
Przykład 98. Funkcje LBound i UBound	130
Przykład 99. Operacje na tablicach	130

Funkcje i operatory matematyczne	131
Przykład 100. Funkcje trygonometryczne	131
Przykład 101. Funkcje Exp i Log	131
Przykład 102. Funkcja Sqr	132
Przykład 103. Funkcje Randomize i Rnd	132
Przykład 104. Funkcje Abs i Sgn	133
Przykład 105. Funkcje Fix, Int i Round	133
Przykład 106. Funkcje Hour, Minute, Second	134
Przykład 107. Funkcje Day, Month, Year	134
Przykład 108. Weekday	135
Przykład 109. TimeSerial i DateSerial	135
Przykład 110. TimeValue i DateValue	136
Przykład 111. Funkcja DateAdd	136
Przykład 112. Funkcja DateDiff	138
Przykład 113. Funkcja DatePart	138
Przykład 114. Funkcje Date, Now i Time	139
Przykład 115. Funkcja Timer	139
Przykład 116. Funkcja MonthName	139
Przykład 117. Funkcja WeekdayName	140
Interakcja z użytkownikiem	140
Przykład 118. Okno komunikatu MsgBox	140
Przykład 119. Okno dialogowe InputBox	141
Przykład 120. Funkcje logiczne	142
Operacje na łańcuchach	143
Przykład 121. Funkcja StrComp	143
Przykład 122. Funkcja StrConv	143
Przykład 123. Funkcje Lcase i Ucase	144
Przykład 124. Funkcje Space, String i Len	144
Przykład 125. Funkcja Format	144
Przykład 126. LSet, Rset	146
Przykład 127. InStr, InStrRev	146
Przykład 128. Left, Right	146
Przykład 129. LTrim, RTrim, Trim	147
Przykład 130. Funkcja Replace	147
Przykład 131. StrReverse	148
Przykład 132. Funkcja Mid	148
Przykład 133. Skok do podprogramu GoSub...Return,	148
Przykład 134. Skok bezwarunkowy GoTo	148
Przykład 135. Wykonuj aż... Do...Loop	149
Przykład 136. Pętla For...Step...Next, Exit For	149
Przykład 137. Pętla For Each...Next	150
Przykład 138. Wykonuj dopóki: While...Wend	150
Przykład 139. Reaguj na sytuację On...GoSub, On...GoTo	151
Przykład 140. Funkcja Choose	152
Przykład 141. Instrukcje warunkowe If...Then...Else	152
Przykład 142. Select Case...End Select	153
Przykład 143. Przełącznik Switch	153
Przykład 144. Struktura With...End With	154

Przykład 145. Przerwanie programu: End i Stop	154
Przykład 146. Przerwanie programu: Exit	155
Przykład 147. Funkcja IsArray	155
Przykład 148. Funkcja IsDate	155
Przykład 149. Funkcja IsEmpty	156
Przykład 150. Funkcja IsError	156
Przykład 151. Funkcja IsMissing	156
Przykład 152. Funkcja IsNull	156
Przykład 153. Funkcja IsNumeric	157
Przykład 154. Funkcja IsObject	157
Przykład 155. Funkcje TypeName, VarType	157
Przykład 156. Obiekty Err i Error	158
Przykład 157. Sam wygeneruj błąd! Raise	159
Przykład 158. Czujka błędu logicznego. On Error	159
Przykład 159. Wznowienie po błędzie. Resume	160
Przykład 160. Przydatne gadżety. Application.Volatile	160
Przykład 161. Przydatne gadżety. Application.DisplayAlerts	160
Przykład 162. Przydatne gadżety. Application.ScreenUpdating	161
Przykład 163. Przydatne gadżety. Application.EnableEvents	161
Przykład 164. Przydatne gadżety. Application.DoubleClick	162
Przykład 165. Przydatne gadżety. Application.WorksheetFunction	162
Przykład 166. Przydatne gadżety. Application.Calculate	162
Rozdział 7. Dodatki	163
Dodatek 1. Obiekt Application.WorksheetFunction	163
Dodatek 2. Skróty klawiaturowe używane w Excelu	171
Dodatek 3. Skróty klawiaturowe używane w edytorze VBA	174
Dodatek 4. Procedury zdarzeniowe	177
Skorowidz	215

Rozdział 5.

Przykłady

W tym rozdziale zaprezentuję przykłady zastosowania VBA w różnych dziedzinach życia. Mam nadzieję, że znajdziesz tu inspirację do tworzenia własnych projektów. Język programowania ma bowiem to do siebie, że jego ograniczeniem jest tylko Twoja wyobraźnia. I umiejętności. Do dzieła zatem.

Przykład 13. Definiowanie własnej funkcji. Konwertuj

Poniższa funkcja dokonuje konwersji jednostek temperatury pomiędzy stopniami Celsjusza, Fahrenheita i kelwinami. Algorytm jest następujący:

Pobranie danych i weryfikacja. W przypadku temperatury niższej niż 0 K wynikiem będzie komunikat o błędzie.

Przeliczenie według zależności:

$$1 \text{ st C} = (1 \cdot 1,8) + 32 \text{ st F} = 1 + 273,15 \text{ K}$$

$$1 \text{ K} = 1 - 273,15 \text{ st C} = ((1 - 273,15) \cdot 1,8) + 32 \text{ st F}$$

$$1 \text{ st F} = (1 - 32) / 1,8 \text{ st C} = ((1 - 32) / 1,8) + 273,15 \text{ K}$$

Jeżeli nastąpiło błędne wpisanie jednostki wejściowej lub wyjściowej, wynikiem będzie komunikat o błędzie.

Przyjmij obliczoną wartość.

Funkcji można używać jak każdej funkcji arkuszowej (rysunek 5.1).

Rysunek 5.1.

Mimo wielkiego bogactwa funkcji wbudowanych każdy z nas może dodać do palety podstawowej garść własnych gadżetów. Zanim jednak opracujesz własną funkcję, dobrze poszukaj — prawdopodobnie ktoś już przygotował ją za Ciebie

	A	B	C	D	E	F
2		Wejście	12 K			
3		Wynik	-261,15	st C		
4						

```

Function konwertuj(liczba As Single, jed_wejściowa, jed_wyjściowa As String)
↳As Variant
    Dim c, k, f, wynik As Variant, błąd As String
    'konwersja na wielkie litery
        jed_wejściowa = UCase(jed_wejściowa)
        jed_wyjściowa = UCase(jed_wyjściowa)
    'czy nie jest niższa od zera bezwzględnego
        If jed_wejściowa = "K" And liczba < 0 Then _
błąd = "Temperatura poniżej 0 bezwzględnego"
        If jed_wejściowa = "C" And liczba < -273.15 Then _
błąd = "Temperatura poniżej 0 bezwzględnego"
        If jed_wejściowa = "F" And liczba < -459.67 Then _
błąd = "Temperatura poniżej 0 bezwzględnego"
    'zamiana danej wejściowej na stopnie Celsjusza
        Select Case jed_wejściowa
            Case "C"
                c = liczba
            Case "K"
                c = liczba - 273.15
            Case "F"
                c = (liczba - 32) / 1.8
            Case Else
                błąd = "Błędna jednostka wejściowa"
        End Select
    'zamiana stopni Celsjusza na jednostkę wyjściową
        Select Case jed_wyjściowa
            Case "C"
                wynik = c
            Case "K"
                wynik = c + 273.15
            Case "F"
                wynik = (c * 1.8) + 32
            Case Else
                błąd = "Błędna jednostka wyjściowa"
        End Select
        If błąd <> "" Then wynik = błąd
    konwertuj = wynik
End Function

```

Przykład 14. Definiowanie własnej funkcji. Dzień tygodnia

Poniższa funkcja wyświetla nazwę dnia tygodnia z podanej daty. Dodatkowo zawiera argument opcjonalny. Jeżeli go nie pominiemy, wynikiem funkcji będzie trzyliterowy skrót dnia tygodnia — rysunek 5.2.

Rysunek 5.2.

Określanie nazwy
dnia tygodnia
za pomocą VBA

	A	B	C	D
2	Data	2009-05-06		
3	Dzień tygodnia pełny	Środa		
4	Dzień tygodnia skrócony	Śro		
5				
6				

Następny przykład pokazuje, że niekiedy warto poszukać, zanim stworzysz kod, bo rozwiązanie może być bliżej, niż myślisz.

```
Function Dz_tygodnia(dzień As Single, Optional skrócona)
Dim tabela(1 To 7) As String
    tabela(7) = "Niedziela"
    tabela(1) = "Poniedziałek"
    tabela(2) = "Wtorek"
    tabela(3) = "Środa"
    tabela(4) = "Czwartek"
    tabela(5) = "Piątek"
    tabela(6) = "Sobota"
Dz_tygodnia = tabela(dzień)
    If IsMissing(skrócona) = False Then Dz_tygodnia = Left(Dz_tygodnia, 3)
End Function
```

Funkcja nie jest skomplikowana. Jej działanie polega na przypisaniu pozycjom tabeli jednowymiarowej nazw kolejnych dni tygodnia, a następnie wybraniu odpowiedniej wartości z obliczonej w komórce. Dodatkowo, jeżeli podasz drugi argument (może to być cokolwiek, bowiem kod sprawdza tylko jego obecność lub brak — `If IsMissing(skrócona) = False Then...`), funkcja zwróci tylko trzy pierwsze litery nazwy dnia tygodnia (`Left(Dz_tygodnia, 3)`).

Bardzo ważne jest, aby argument opcjonalny badany funkcją `IsMissing` miał określony typ `Variant`. Nasza funkcja nie definiuje typu argumentu, zatem `Variant` zostaje do niego przyporządkowany automatycznie. Zadeklarowanie innego typu argumentu, na przykład:

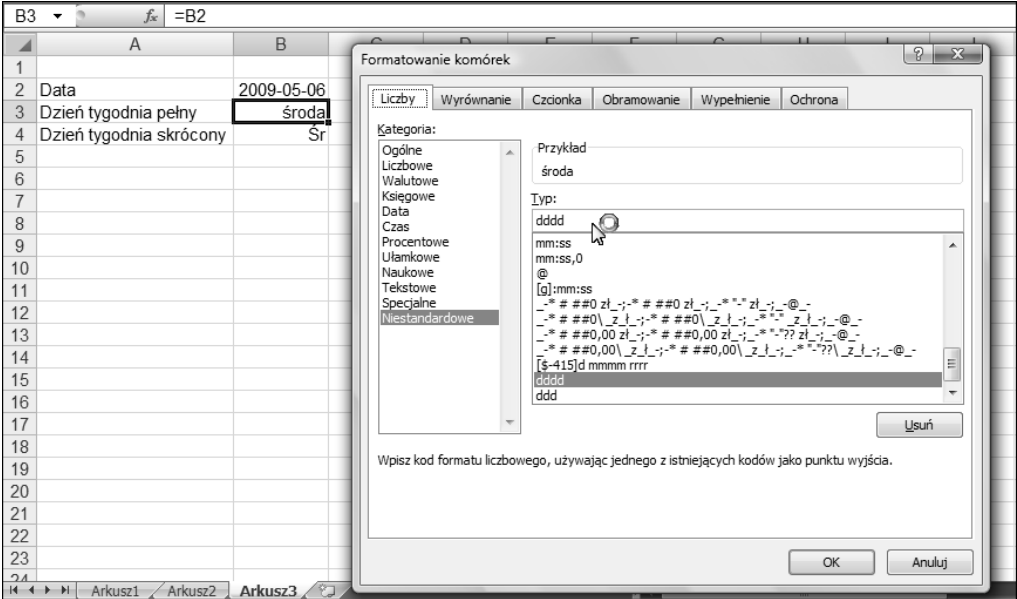
```
Function Dz_tygodnia(dzień As Single, Optional skrócona As String)
    spowoduje nieprawidłowe działanie.
```

Przykład 15. Dzień tygodnia dużo łatwiej

Być może miałeś już okazję przekonać się o tym, że narzędzia wbudowane w Excel bywają zaskakujące. Niekiedy przemyślane ich użycie może prowadzić do takich samych efektów jak używanie VBA. Rysunek 5.3 pokazuje, jak możemy otrzymać wynik z przykładu 14 bez używania VBA.

Przykład 16. Funkcja Słownie

To pewnie najbardziej pożądana z funkcji, które trzeba samemu napisać. Nie wiadomo, dlaczego Microsoft w kolejnych edycjach pakietu Office nie dołącza tej funkcji w standardzie. Nieważne. Mamy przynajmniej możliwość, aby rozwinąć skrzydła. Ten przykład jest dość rozbudowany, bowiem zachowuje polską gramatykę słów. Przykład 17 rezygnuje z niej, dając w zamian możliwość wyboru waluty.



Rysunek 5.3. Po prostu skopiuj datę do komórek B3 i B4, a następnie z okna Formatowanie komórek (uruchamianego kombinacją `Ctrl+I`) wybierz niestandardowe formaty komórek: `dddd` dla B3 i `ddd` dla B4

Function SLOWNIE(x)

```
Dim Setki(10) As String
Dim dziesiątki(10) As String
Dim jednostki(20) As String
```

```
Setki(0) = ""
Setki(1) = "sto"
Setki(2) = "dwieście"
Setki(3) = "trzysta"
Setki(4) = "czterysta"
Setki(5) = "pięćset"
Setki(6) = "sześćset"
Setki(7) = "siedemset"
Setki(8) = "osiemset"
Setki(9) = "dziewięćset"
dziesiątki(0) = ""
dziesiątki(1) = "dziesięć"
dziesiątki(2) = "dwadzieścia"
dziesiątki(3) = "trzydzieści"
dziesiątki(4) = "czterdzieści"
dziesiątki(5) = "pięćdziesiąt"
dziesiątki(6) = "sześćdziesiąt"
dziesiątki(7) = "siedemdziesiąt"
dziesiątki(8) = "osiemdziesiąt"
dziesiątki(9) = "dziewięćdziesiąt"
```

```

jednostki(0) = ""
jednostki(1) = "jeden"
jednostki(2) = "dwa"
jednostki(3) = "trzy"
jednostki(4) = "cztery"
jednostki(5) = "pięć"
jednostki(6) = "sześć"
jednostki(7) = "siedem"
jednostki(8) = "osiem"
jednostki(9) = "dziewięć"
jednostki(10) = "dziesięć"
jednostki(11) = "jedenaście"
jednostki(12) = "dwanaście"
jednostki(13) = "trzynaście"
jednostki(14) = "czternaście"
jednostki(15) = "piętnaście"
jednostki(16) = "szesnaście"
jednostki(17) = "siedemnaście"
jednostki(18) = "osiemnaście"
jednostki(19) = "dziewiętnaście"

```

' jeżeli moduł liczby jest większy od podanej wartości to zakończ i wyświetl komunikat

```

If Abs(x) >= 1000000000000# Then
    SLOWNIE = "Przekroczony zakres od -1.000.000.000.000
    ↪do 1.000.000.000.000"
    Exit Function
End If
If x < 0 Then minus = "Minus " Else minus = ""
x = Abs(x)
liczba = Int(x)
L = (x - liczba) * 100
liczbagr = Int(L)
If (L - liczbagr) * 10 >= 5 Then liczbagr = liczbagr + 1
'określa ile mamy miliardów
liczbamd = Int(liczba / 1000000000)
'określa ile mamy milionów
liczbamil = Int((liczba - liczbamd * 1000000000) / 1000000)
'określa ile mamy tysięcy
liczbatys = Int((liczba - liczbamd * 1000000000 - liczbamil *
    ↪1000000) / 1000)
'określa ile mamy setek
liczbaset = Int(liczba - liczbamd * 1000000000 - liczbamil * 1000000
    ↪- liczbatys * 1000)

```

miliardy:

```

liczba = liczbamd
If liczba = 0 Then GoTo miliony

L = Int(liczba / 100)
słowniezl = słowniezl + Setki(L)

```

```

liczba = liczba - (L * 100)
If liczba < 20 Then slowniezl = slowniezl + " " + jednostki(liczba):
↳GoTo pizmiliardy
L = Int(liczba / 10)
slowniezl = slowniezl + " " + dziesiątki(L)

liczba = liczba - (L * 10)
slowniezl = slowniezl + " " + jednostki(liczba)

```

pizmiliardy:

```

If liczbamd = 1 Then slowniezl = slowniezl + " miliard ": GoTo miliony
aa = Str(liczba)
bb = Right(aa, 1)

If aa >= 5 And aa < 20 Then slowniezl = slowniezl + " miliardów ":
↳GoTo miliony
If bb > 1 And bb < 5 Then slowniezl = slowniezl + " miliardy ":
↳GoTo miliony
If liczbamd >= 5 Then slowniezl = slowniezl + " miliardów "

```

miliony:

```

liczba = liczbamil
If liczba = 0 Then GoTo tysiace

L = Int(liczba / 100)
slowniezl = slowniezl + Setki(L)

liczba = liczba - (L * 100)
If liczba < 20 Then slowniezl = slowniezl + " " + jednostki(liczba):
↳GoTo pizmiliony
L = Int(liczba / 10)
slowniezl = slowniezl + " " + dziesiątki(L)

liczba = liczba - (L * 10)
slowniezl = slowniezl + " " + jednostki(liczba)

```

pizmiliony:

```

If liczbamil = 1 Then slowniezl = slowniezl + " milion ": GoTo tysiace
aa = Str(liczba)
bb = Right(aa, 1)

If aa >= 5 And aa < 20 Then slowniezl = slowniezl + " milionów ":
↳GoTo tysiace
If bb > 1 And bb < 5 Then slowniezl = slowniezl + " miliony ":
↳GoTo tysiace
If liczbamil >= 5 Then slowniezl = slowniezl + " milionów "

```

tysiace:

```

liczba = liczbatys
If liczba = 0 Then GoTo jednostki

L = Int(liczba / 100)
sloowniezl = sloowniezl + Setki(L)

liczba = liczba - (L * 100)
If liczba < 20 Then sloowniezl = sloowniezl + " " + jednostki(liczba):
↳GoTo pisztyysiace
L = Int(liczba / 10)
sloowniezl = sloowniezl + " " + dziesiątki(L)

liczba = liczba - (L * 10)
sloowniezl = sloowniezl + " " + jednostki(liczba)

```

pisztysiace:

```

If liczbatys = 1 Then sloowniezl = sloowniezl + " tysiąc ":
↳GoTo jednostki
aa = Str(liczba)
bb = Right(aa, 1)

If aa >= 5 And aa < 20 Then sloowniezl = sloowniezl + " tysięcy ":
↳GoTo jednostki
If bb > 1 And bb < 5 Then sloowniezl = sloowniezl + " tysiące ":
↳GoTo jednostki
If liczbatys >= 5 Then sloowniezl = sloowniezl + " tysięcy "

```

jednostki:

```

liczba = liczbaset
If Int(x) = 0 Then sloowniezl = sloowniezl + " zero złotych":
↳GoTo grosze
If liczba = 0 Then sloowniezl = sloowniezl + " złotych": GoTo grosze

L = Int(liczba / 100)
sloowniezl = sloowniezl + Setki(L)

liczba = liczba - (L * 100)
If liczba < 20 Then sloowniezl = sloowniezl + " " + jednostki(liczba):
↳GoTo piszjednostki
L = Int(liczba / 10)
sloowniezl = sloowniezl + " " + dziesiątki(L)

liczbaa = liczba - (L * 10)
sloowniezl = sloowniezl + " " + jednostki(liczbaa)

```

piszjednostki:

```
aa = Str(liczba)
bb = Right(aa, 1)

If aa >= 5 And aa < 20 Then słowniezl = słowniezl + " złotych": GoTo grosze
If bb >= 2 And bb < 5 Then słowniezl = słowniezl + " złote": GoTo grosze
If aa = 1 Then słowniezl = słowniezl + " złoty"
If liczbaset >= 5 Then słowniezl = słowniezl + " złotych"
```

grosze:

```
liczba = liczbagr
If liczba = 0 Then słowniegr = "zero groszy": GoTo wynik
If liczba < 20 Then słowniegr = jednostki(liczba): GoTo pizsgrosze

L = Int(liczba / 10)
słowniegr = słowniegr + " " + dziesiątki(L)

liczba = liczba - (L * 10)
słowniegr = słowniegr + " " + jednostki(liczba)
```

pizsgrosze:

```
aa = Str(liczbagr)
bb = Right(aa, 1)

If aa >= 5 And aa < 20 Then słowniegr = słowniegr + " groszy": GoTo wynik
If bb >= 2 And bb < 5 Then słowniegr = słowniegr + " grosze": GoTo wynik
If aa = 1 Then słowniegr = słowniegr + " grosz": GoTo wynik
If aa >= 5 Then słowniegr = słowniegr + " groszy"
```

wynik:

'obcina zbędne spacje na początku i końcu słowa

```
bez_spacji = Trim(minus + słowniezl)
```

'pierwsza litera wyrażenia

```
wielka_litera = UCase(Left(bez_spacji, 1))
```

'liczy ile liter jest w wyrażeniu

```
ile_mamy_liter = Len(bez_spacji)
```

'wstawia wielką literę na początku wyrażenia

```
wstaw_wielka = wielka_litera + Right(bez_spacji, ile_mamy_liter - 1)
```

'przypisuje wartość do funkcji

```
SLOWNIE = Application.WorksheetFunction.Trim(wstaw_wielka + " " + słowniegr)
```

End Function

Wyjaśnienia do działania tej funkcji zawarte są w komentarzach w kodzie (czyli w tekście po apostrofie — taki tekst nie jest analizowany przez VBA). Funkcja jest sekwencją operacji powtarzających się dla każdego z rzędów wielkości: miliardów, milionów, tysięcy itd. Wynikiem kolejnej sekwencji (nazwijmy ją podprogramem) jest zmienna „słownieZ1”, która jest wydłużana (czyli doklejane są do niej kolejne słowa) w miarę schodzenia kodu coraz niżej. W sekcji „wynik” dodawana jest jeszcze wielka litera na początku wyrażenia, znak minus, jeżeli liczba jest ujemna, oraz usuwane są zbędne spacje między kolejnymi słowami.

Do usunięcia zbędnych spacji postanowiłem użyć funkcji arkuszowej Trim (w polskim Excelu jest to USUN.ZBĘDNE.ODSTĘPY). Co prawda VBA także oferuje funkcję Trim, jednak podczas prób okazało się, że odcina ona tylko spacje na początku i końcu wyniku, nie ingerując w odstępy pomiędzy słowami.

Przykład 17. Funkcja Słownie — wersja 2

Poniższa funkcja jest prostsza od poprzedniej, bowiem nie uwzględnia polskiej gramatyki. Mamy za to do wyboru zapis w języku polskim lub angielskim i waluty: euro, dolary i złote.

```
Function SŁOWNIE(ByVal liczba, Optional waluta, Optional język)
On Error Resume Next
'deklaracja tablicy
Dim słowa(9) As String
'deklaracja zmiennych
Dim przecinek, a, x, całkowita As Single, po_przecinku As Boolean
Dim wynik, gr, zł, znak As String, jedność
'zmiana liter w zmiennych na duże
If waluta = Null Then waluta = " "
If język = Null Then język = " "
waluta = UCase(waluta)
język = UCase(język)
'Nadanie wartości początkowych zmiennym
wynik = ""
po_przecinku = False
Select Case waluta
Case "E"
gr = "Cent"
zł = "Euro"
Case "D"
gr = "Cent"
zł = "Dolar"
Case Else
gr = "gr"
zł = "zł"
End Select
Select Case język
Case "ANG"
```

```

słowa(0) = "zero"
słowa(1) = "one"
słowa(2) = "two"
słowa(3) = "three"
słowa(4) = "four"
słowa(5) = "five"
słowa(6) = "six"
słowa(7) = "seven"
słowa(8) = "eight"
słowa(9) = "nine"
Case Else
słowa(0) = "zero"
słowa(1) = "jeden"
słowa(2) = "dwa"
słowa(3) = "trzy"
słowa(4) = "cztery"
słowa(5) = "pięć"
słowa(6) = "sześć"
słowa(7) = "siedem"
słowa(8) = "osiem"
słowa(9) = "dziewięć"
End Select

' ustalenie znaku plus/minus
znak = ""
If liczba < 0 Then
znak = "minus "
End If
liczba = Abs(liczba)
całkowita = Int(liczba)
liczba = Trim(Str(liczba))

'szukanie przecinka
przecinek = InStr(liczba, ".")
If przecinek <> 0 Then a = przecinek - 1: po_przecinku = True Else a =
↳Len(liczba)

'rozpatrywanie liczb z lewej strony przecinka
If całkowita > 0 Then
For x = 1 To a
jedność = Mid(liczba, x, 1)
jedność = Val(jedność)
wynik = wynik & słowa(jedność) & "-"
Next x
Else
wynik = słowa(0) & "-"
End If

wynik = wynik & z1

```



```

'rozpatrywanie dwóch liczb z prawej strony przecinka (jeżeli występują)
If po_przecinku Then
    wynik = wynik & "-"
    For x = 2 To 3
        jedność = Mid(liczba, a + x, 1)
        jedność = Val(jedność)
        wynik = wynik & słowa(jedność) & "-"
    Next x
    wynik = wynik & gr
Else
    wynik = wynik & "-" & słowa(0) & "-" & gr
End If
'zwrócenie wartości funkcji, obcięcie niepotrzebnych spacji
SŁOWNIE = Trim(znak & wynik)
End Function

```

Kilka słów wyjaśnienia, jak działa powyższy kod:

1. Rezerwuje tablicę jednowymiarową na jednostki.
2. Pobiera informacje na temat języka i waluty. Odpowiednie wartości zostają przypisane zmiennym waluta i język.
3. Wprowadza do tablicy słowa po polsku lub angielsku, w zależności od wskazanego języka, oraz przypisuje zmiennym zł i gr odpowiednie wartości, stosownie do wybranej waluty.
4. W przypadku liczby ujemnej nadaje zmiennej znak wartość minus.
5. W analizowanej liczbie znajduje przecinek.
6. Obcina część ułamkową w przypadku, gdy są więcej niż dwie cyfry po przecinku.
7. Analizuje kolejno cyfry z lewej strony przecinka, przyporządkowuje im właściwe słowa, rozdziela je myślnikiem.
8. Na zakończenie wstawiona zostaje nazwa jednostki głównej waluty.
9. Analizuje dwie pierwsze cyfry z prawej strony przecinka i postępuje jak wyżej.
10. Ostatnią czynnością jest przypisanie funkcji SŁOWNIE obliczonej wartości ciągu.

W przeciwieństwie do poprzedniej funkcji tutaj nie mamy ograniczenia dotyczącego wielkości wprowadzanej liczby. Nie jest ono potrzebne, bowiem nie używamy tu słów „miliardy”, „miliony” itp.

Przykład 18. Funkcja Słownie z podprogramem

Jak zauważyłeś, w przykładzie 16 część kodu jest powtarzana. Te same czynności są wykonywane dla każdej grupy cyfr. Jeżeli tworzysz duże programy, warto zamykać powtarzalne elementy kodu w osobne procedury. Zmniejszają one objętość kodu i tylko minimalnie spowalniają jego działanie.

```
Function SLOWNIE_Z_PODPROGRAMEM(x)
```

```
Dim Setki(10) As String
```

```
Dim dziesiątki(10) As String
```

```
Dim jednostki(20) As String
```

```
Setki(0) = ""
```

```
Setki(1) = "sto"
```

```
Setki(2) = "dwieście"
```

```
Setki(3) = "trzysta"
```

```
Setki(4) = "czteryście"
```

```
Setki(5) = "pięćset"
```

```
Setki(6) = "sześćset"
```

```
Setki(7) = "siedemset"
```

```
Setki(8) = "osiemset"
```

```
Setki(9) = "dziewięćset"
```

```
dziesiątki(0) = ""
```

```
dziesiątki(1) = "dziesięć"
```

```
dziesiątki(2) = "dwadzieścia"
```

```
dziesiątki(3) = "trzydzieści"
```

```
dziesiątki(4) = "czterdzieści"
```

```
dziesiątki(5) = "pięćdziesiąt"
```

```
dziesiątki(6) = "sześćdziesiąt"
```

```
dziesiątki(7) = "siedemdziesiąt"
```

```
dziesiątki(8) = "osiemdziesiąt"
```

```
dziesiątki(9) = "dziewięćdziesiąt"
```

```
jednostki(0) = ""
```

```
jednostki(1) = "jeden"
```

```
jednostki(2) = "dwa"
```

```
jednostki(3) = "trzy"
```

```
jednostki(4) = "cztery"
```

```
jednostki(5) = "pięć"
```

```
jednostki(6) = "sześć"
```

```
jednostki(7) = "siedem"
```

```
jednostki(8) = "osiem"
```

```
jednostki(9) = "dziewięć"
```

```
jednostki(10) = "dziesięć"
```

```
jednostki(11) = "jedenaście"
```

```
jednostki(12) = "dwanaście"
```

```
jednostki(13) = "trzynaście"
```

```
jednostki(14) = "czternaście"
```

```
jednostki(15) = "piętnaście"
```

```
jednostki(16) = "szesnaście"
```

```

jednostki(17) = "siedemnaście"
jednostki(18) = "osiemnaście"
jednostki(19) = "dziewiętnaście"

' jeżeli moduł liczby jest większy od podanej wartości to zakończ i wyświetl komunikat
  If Abs(x) >= 1000000000000# Then
    SLOWNIE_Z_PODPROGRAMEM = "Przekroczony zakres od -1.000.000.000.000
    ↪do 1.000.000.000.000"
    Exit Function
  End If
If x < 0 Then minus = "Minus " Else minus = ""
x = Abs(x)
  liczba = Int(x)
  L = (x - liczba) * 100
  liczbagr = Int(L)
  If (L - liczbagr) * 10 >= 5 Then liczbagr = liczbagr + 1
'określa ile mamy miliardów
  liczbamd = Int(liczba / 1000000000)
'określa ile mamy milionów
  liczbamil = Int((liczba - liczbamd * 1000000000) / 1000000)
'określa ile mamy tysięcy
  liczbatys = Int((liczba - liczbamd * 1000000000 - liczbamil *
  ↪1000000) / 1000)
'określa ile mamy setek
  liczbaset = Int(liczba - liczbamd * 1000000000 - liczbamil *
  ↪1000000 - liczbatys * 1000)

miliardy:

  liczba = liczbamd
  If liczba = 0 Then GoTo miliony

  L = Int(liczba / 100)
  slownie_zlotych = slownie_zlotych + Setki(L)

  liczba = liczba - (L * 100)
  If liczba < 20 Then slownie_zlotych = slownie_zlotych + " " +
  ↪jednostki(liczba): GoTo pismiliardy
  L = Int(liczba / 10)
  slownie_zlotych = slownie_zlotych + " " + dziesiątki(L)
  liczba = liczba - (L * 10)
  slownie_zlotych = slownie_zlotych + " " + jednostki(liczba)

pismiliardy:

  If liczbamd = 1 Then slownie_zlotych = slownie_zlotych + " miliard ":
  ↪GoTo miliony
  aa = Str(liczba)
  bb = Right(aa, 1)
  slownie_zlotych = slownie_zlotych + przypisz(aa, bb, " miliardów ",
  ↪" miliardy ", liczbamd)

```

miliony:

```

liczba = liczbamil
If liczba = 0 Then GoTo tysiace

L = Int(liczba / 100)
sloownie_zlotych = sloownie_zlotych + Setki(L)

liczba = liczba - (L * 100)
If liczba < 20 Then sloownie_zlotych = sloownie_zlotych + " " +
↳jednostki(liczba): GoTo pismiliony
L = Int(liczba / 10)
sloownie_zlotych = sloownie_zlotych + " " + dziesiatki(L)

liczba = liczba - (L * 10)
sloownie_zlotych = sloownie_zlotych + " " + jednostki(liczba)

```

pismiliony:

```

If liczbamil = 1 Then sloownie_zlotych = sloownie_zlotych + " milion ":
↳GoTo tysiace
aa = Str(liczba)
bb = Right(aa, 1)

```

```

sloownie_zlotych = sloownie_zlotych + przypisz(aa, bb, " milionów ",
↳" miliony ", liczbamil)

```

tysiace:

```

liczba = liczbatys
If liczba = 0 Then GoTo jednostki

L = Int(liczba / 100)
sloownie_zlotych = sloownie_zlotych + Setki(L)

liczba = liczba - (L * 100)
If liczba < 20 Then sloownie_zlotych = sloownie_zlotych + " " +
↳jednostki(liczba): GoTo pisztysiacie
L = Int(liczba / 10)
sloownie_zlotych = sloownie_zlotych + " " + dziesiatki(L)

liczba = liczba - (L * 10)
sloownie_zlotych = sloownie_zlotych + " " + jednostki(liczba)

```

pisztysiacie:

```

If liczbatys = 1 Then sloownie_zlotych = sloownie_zlotych + " tysiąc ":
↳GoTo jednostki
aa = Str(liczba)
bb = Right(aa, 1)

```

```
słownie_zlotych = słownie_zlotych + przypisz(aa, bb, " tysięcy ", " tysiące ",
↳liczbatys)
```

jednostki:

```
liczba = liczbaset
If Int(x) = 0 Then słownie_zlotych = słownie_zlotych + " zero złotych":
↳GoTo grosze
If liczba = 0 Then słownie_zlotych = słownie_zlotych + " złotych":
↳GoTo grosze
```

```
L = Int(liczba / 100)
słownie_zlotych = słownie_zlotych + Setki(L)
```

```
liczba = liczba - (L * 100)
If liczba < 20 Then słownie_zlotych = słownie_zlotych + " " +
↳jednostki(liczba): GoTo piszjednostki
L = Int(liczba / 10)
słownie_zlotych = słownie_zlotych + " " + dziesiątki(L)
```

```
liczbaa = liczba - (L * 10)
słownie_zlotych = słownie_zlotych + " " + jednostki(liczbaa)
```

piszjednostki:

```
aa = Str(liczba)
bb = Right(aa, 1)

If aa >= 5 And aa < 20 Then słownie_zlotych = słownie_zlotych + "
↳złotych": GoTo grosze
If bb >= 2 And bb < 5 Then słownie_zlotych = słownie_zlotych + " złote":
↳GoTo grosze
If aa = 1 Then słownie_zlotych = słownie_zlotych + " złoty"
If liczbaset >= 5 Then słownie_zlotych = słownie_zlotych + " złotych"
```

grosze:

```
liczba = liczbagr
If liczba = 0 Then słowniegr = "zero groszy": GoTo wynik
If liczba < 20 Then słowniegr = jednostki(liczba): GoTo piszgrosze
```

```
L = Int(liczba / 10)
słowniegr = słowniegr + " " + dziesiątki(L)
```

```
liczba = liczba - (L * 10)
słowniegr = słowniegr + " " + jednostki(liczba)
```

piszgrosze:

```
aa = Str(liczbagr)
bb = Right(aa, 1)
```

```

If aa >= 5 And aa < 20 Then słowniegr = słowniegr + " groszy": GoTo wynik
If bb >= 2 And bb < 5 Then słowniegr = słowniegr + " grosze": GoTo wynik
If aa = 1 Then słowniegr = słowniegr + " grosz": GoTo wynik
If aa >= 5 Then słowniegr = słowniegr + " groszy"

```

wynik:

```

' obcina zbędne spacje na początku i końcu słowa
bez_spacji = Trim(minus + słownie_złotych)

```

```

' pierwsza litera wyrażenia
wielka_litera = UCase(Left(bez_spacji, 1))

```

```

' liczy ile liter jest w wyrażeniu
ile_mamy_liter = Len(bez_spacji)

```

```

' wstawia wielką literę na początku wyrażenia
wstaw_wielka = wielka_litera + Right(bez_spacji, ile_mamy_liter - 1)

```

```

' przypisuje wartość do funkcji
SLOWNIE_Z_PODPROGRAMEM = Application.WorksheetFunction.Trim(wstaw_wielka +
↳ " " + słowniegr)

```

```
End Function
```

W powtarzalnych miejscach programu wywoływana jest funkcja *przypisz*, której wynik doklejany jest do naszej zmiennej *słownie_złotych*. Kluczem działania funkcji *przypisz* są wartości przekazywane przy jej wywoływaniu jako argumenty. Poniżej przedstawiona jest jej treść:

```

Function przypisz(aa, bb, cc, dd, ee)
    If aa >= 5 And aa < 20 Then przypisz = cc: Exit Function
    If bb > 1 And bb < 5 Then przypisz = dd: Exit Function
    If ee >= 5 Then przypisz = cc
End Function

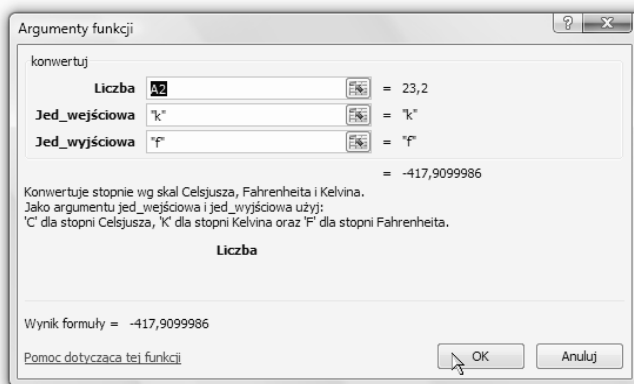
```

Przykład 19. Określenie właściwości makra/funkcji za pomocą edytora

Pisząc funkcję użytkownika, możesz zadbać o jej opis i inne właściwości. To bardzo ułatwia korzystanie z niej. Przykład 13 będzie tu dobrą ilustracją. Funkcja *Konwertuj* posiada trzy parametry. Jaką jednak powinny mieć postać? Wie o tym tylko jej twórca i może jeszcze ten, kto będzie miał dość czasu i umiejętności, aby rzucić okiem na jej kod. Marna to informacja dla potencjalnego użytkownika. Oczekuje on bowiem od Ciebie użytecznego, łatwego w obsłudze narzędzia i informacji podanych na talerzu (rysunek 5.4), a nie kolejnej łamigłówki. Gdyby szukał łamigłówki, pewnie sięgnąłby po odpowiednie czasopismo. My nie zakładamy, że stworzysz makra w VBA do czasopisma z zagadkami, zatem spróbujemy do utworzonej w przykładzie 13 funkcji dodać informacje umożliwiające użytkownikowi prawidłowe jej zredagowanie.

Rysunek 5.4.

Oprócz przyjaznych nazw parametrów możesz dodać trzy linijki informacji dla użytkownika, aby zachęcić go do korzystania z Twojej funkcji

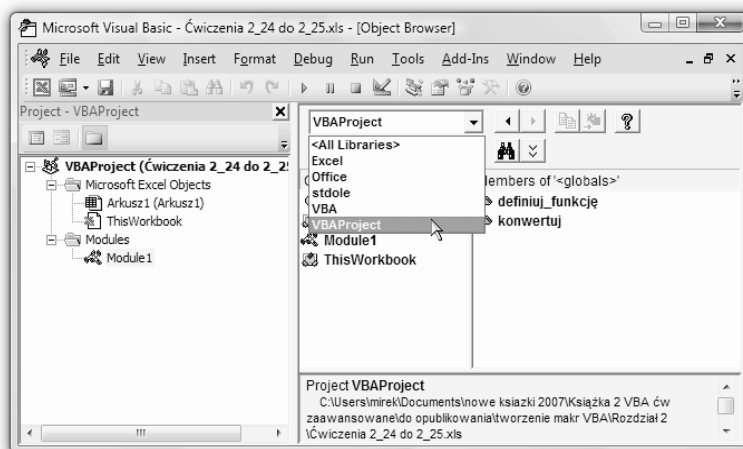


W tym przykładzie nie będziemy pisać kodów VBA, mimo że edytor będzie nam potrzebny.

1. W oknie edytora VBA otwórz okno przeglądarki obiektu (F2, lub *View/Object Browser*).
2. Z listy (patrz rysunek 5.5) wybierz kategorię *VBA Project*.

Rysunek 5.5.

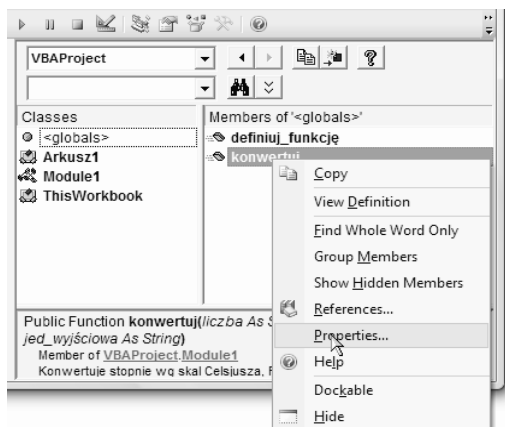
Przeglądarka obiektów. Użyteczne narzędzie dla deweloperów VBA. Naprawdę warto się z nim zaprzyjaźnić



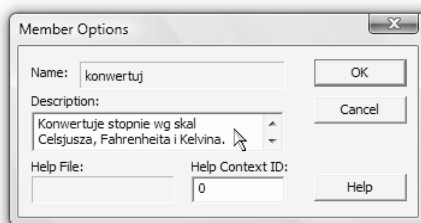
3. W polu *Classes* (rysunek 5.6) wskaż moduł, w którym znajduje się opisywana funkcja, kliknij prawym przyciskiem myszy jej nazwę w przeglądarce i z podręcznego menu wybierz polecenie *Properties*....
4. W oknie dialogowym *Member Options* (rysunek 5.7) wpisz opis funkcji. Oczywiście! Możesz tam wpisać dowcip z cyklu „przychodzi baba do lekarza”, lecz jeżeli będzie dłuższy niż trzy linijki, nie wyświetli się w całości, a jeżeli dodatkowo nie będzie śmieszny, to gwarantuję ci, że mocno zapadnieś użytkownikom w pamięć. Tym błyskotliwym ruchem być może przysłużysz się czymś badaniom na temat wpływu inteligencji krzemowej na białkową.

Rysunek 5.6.

Dodanie opisu do funkcji użytkownika nie jest trudne. Zaufaj mi

**Rysunek 5.7.**

Kto powiedział, że w okno Member Options należy wpisać pomoc do funkcji użytkownika? Możemy co najwyżej założyć, że jest to zalecane

**Przykład 20. Określenie właściwości makra/funkcji za pomocą VBA**

Wróćmy już do VBA. Nareszcie.

Skorzystaj z metody MacroOptions. Dla obiektu Application ma ona wiele argumentów. Część z nich jest pomijana. Najważniejsze dla nas argumenty są użyte w poniższej procedurze:

Macro — podaje nazwę funkcji (makra), do którego odnosi się Twoje działanie.

Description — wyświetla tekst wprowadzony przez Ciebie w oknie *Member Options*.

Category — przypisuje Twoją funkcję do odpowiedniej kategorii funkcji, zgodnie z tabelą 5.1.

```
Sub definiuj_funkcję()
    tekst = "Konwertuje stopnie wg skal Celsjusza, Fahrenheita i Kelvina."
    tekst = tekst & vbCrLf & "Jako argumentu jed_wejściowa i jed_wyjściowa
    użyj:"
    tekst = tekst & vbCrLf & "'C' dla stopni Celsjusza."
    tekst = tekst & " 'K' dla stopni Kelvina oraz"
    tekst = tekst & " 'F' dla stopni Fahrenheita."
    Application.MacroOptions Macro:="konwertuj", _
    Description:=tekst, Category:=3
End Sub
```


Tabela 5.1. Kategorie funkcji i wartości argumentu Category metody MacroOptions

Kategoria funkcji	Wartość argumentu Category
Finansowe	1
Daty i czasu	2
Matematyczne	3
Statystyczne	4
Wyszukiwania i adresu	5
Bazy danych	6
Tekstowe	7
Logiczne	8
Informacyjne	9
Użytkownika	Argument należy pominąć

Przykład 21. Wstawianie funkcji arkuszowej do arkusza za pomocą VBA

Poniższy przykład spowoduje wstawienie funkcji ZŁĄCZ.TEKSTY do komórki C7.

```
Range("C7").Formula = "=CONCATENATE(A1,Arkusz2!A1,Arkusz3!B3)"
```

Jak widać, nie możemy wprowadzać tu polskich nazw funkcji. Angielskie odpowiedniki znajdziesz w pliku *funcs.xls*, który jest instalowany w komputerze razem z Excelem. W dalszej części książki zamieściłem tabelę metod zawierających się w obiekcie `Application.Worksheetfunction`. Tabela ta zawiera polskie nazwy funkcji i odpowiadające im metody będące w zbiorze tego obiektu. Nazwy metod odpowiadają angielskim nazwom funkcji. Niestety, funkcja ZŁĄCZ.TEKSTY — jak i kilkanaście innych — nie jest reprezentowana w tym zbiorze i nie można użyć jej w połączeniu z obiektem `Worksheetfunction`. Jedynym więc sposobem na odnalezienie jej angielskiego odpowiednika jest wspomniany wyżej plik.

Możesz także użyć rejestratora makr w celu wygenerowania kodu wpisującego formułę do komórki. To najpewniejszy sposób, aby nie popełnić błędu, choć rejestrator używa słowa kluczowego `FormulaR1C1`, które wymusza nieco inny sposób adresowania komórek. Poza tym rejestrator wstawia osobny wiersz kodu mówiący o zaznaczeniu komórki.

```
Range("C7").Select
ActiveCell.FormulaR1C1 =
    "=CONCATENATE(R[-14]C[-2],Arkusz2!R[-14]C[-2],Arkusz3!R[-12]C[-1])"
```

Poza tymi sprawami technicznymi kod ma takie samo działanie, więc jeżeli nie musisz w niego ingerować, skorzystaj z rejestratora i ciesz się szybko osiągniętym wynikiem.



Wskazówka

Mając powyższą wiedzę, pomyśl, jak łatwo byłoby stworzyć „samopiszący się” arkusz Excela. Możesz w taki sposób wymusić na użytkownikowi zgodę na uruchomienie makr. Jeżeli nie będzie zgody, nie będzie w nim nawet zwykłych excelowskich formuł!

Przykład 22. Wymuszenie włączenia dodatku

Jeżeli Twój skoroszyt korzysta z formuł lub funkcjonalności któregoś ze standardowych dodatków, na przykład *AnalysisToolPak* zawierającego szereg funkcji inżynierskich, możesz spowodować jego załadowanie przy każdym otwarciu Twojego pliku. Poniższa procedura zostanie wykonana po każdym otwarciu skoroszytu. Należy ją umieścić w jego module.

```
Private Sub Workbook_Open()
    AddIns("Analysis ToolPak").Installed = True
End Sub
```

Przykład 23. Zamknięcie dodatku

Poniższa procedura została przypisana do zdarzenia wykonywanego przed zamknięciem skoroszytu `Workbook_BeforeClose`. Użyłem tego zdarzenia, niejako kontynuując poprzedni przykład. Jednak oczywiście nie jest to konieczne. Być może Twój skoroszyt nie powinien pracować, gdy jakiś dodatek jest załadowany, i musisz umieścić procedurę, która go wyłączy zaraz na początku. W tym przypadku powinieneś użyć zdarzenia `Workbook_Open`. A może dodatek powinien ładować się po wejściu/wyjściu z określonego arkusza? Twój wybór! Na końcu książki zamieściłem listę zdarzeń, z których możesz do woli korzystać.

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
    AddIns("Analysis ToolPak").Installed = False
End Sub
```

Przykład 24. Zamknięcie dodatku uruchomionego przez nasz skoroszyt

Praktyka mówi, że choć 90 procent użytkowników nigdy nie dokonało większej modyfikacji narzędzi używanych w swoim pakiecie Office, to właśnie z tymi 10 procentami, które czegoś dokonały, programiści mają największy kłopot. Trudno bowiem przewidzieć, na jaki grunt trafi Twój produkt. Na etapie projektowania musisz zatem założyć różne sytuacje. Im więcej ich przewidzisz, tym stabilniejszy będzie Twój program. Jednym z oczywistych założeń wydaje się, że użytkownik mógł samodzielnie załadować jakiś dodatek (pozostaliśmy już przy naszym przykładowym *AnalysisToolPak*), i nie ma potrzeby, aby po zamknięciu naszego skoroszytu dodatek ten był dezaktywowany. Na początku musimy tylko odczytać, czy dodatek jest już otwarty, i informację o jego stanie przechować w zmiennej, która zostanie odczytana przy zamykaniu skoroszytu.

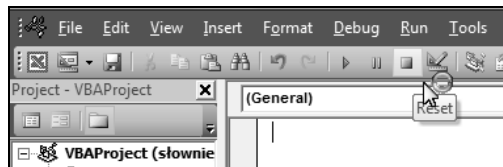
Na poziomie modułu wpisz więc deklarację zmiennej.

```
Dim dodatek As Boolean
```

To deklaracja zmiennej logicznej, która może przyjmować wartości Prawda lub Fałsz. Zadeklarowanie jej na poziomie modułu (przed pierwszą procedurą sub lub function) zapewnia nam, że jej wartość będzie przechowywana do zamknięcia skoroszytu, pod warunkiem że VBA nie będzie wymagał zresetowania (rysunek 5.8).

Rysunek 5.8.

Gdy nastąpi zawieszenie wykonywania makr (mimo dołożenia wszelkich starań zdarza się to częściej, niż przypuszczasz), konieczne jest zresetowanie VBA. Powoduje to, niestety, wyzerowanie zmiennych



W module skoroszytu wpisz zdarzenie przy otwarciu.

```
Private Sub Workbook_Open()
dodatek = False
If AddIns("Analysis ToolPak").Installed = True Then
' jeżeli zachodzi powyższy warunek to zmiana wartości zmiennej dodatek...
dodatek = True
' ... i wyjście z procedury
Exit Sub
' w przeciwnym wypadku załadowanie dodatku
Else
AddIns("Analysis ToolPak").Installed = True
End If
End Sub
```

... i przy zamknięciu.

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
' wyjdź jeżeli zmienna dodatek ma wartość True
If dodatek = True Then Exit Sub
AddIns("Analysis ToolPak").Installed = False
End Sub
```

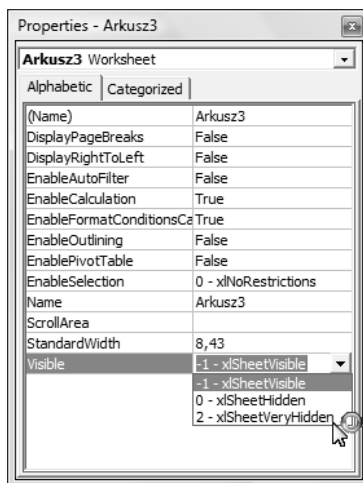
Przykład 25. Rozwiązanie drugie — stabilniejsze

Jak już wiesz z ćwiczenia powyżej, zdarzają się sytuacje, gdy wartość przechowywana przez zmienną w VBA może zostać — z różnych przyczyn — zresetowana. Doprowadzi to do złego działania procedury uruchamianej przy zamykaniu skoroszytu. Nie wiadomo bowiem, czy informacja pobrana na początku pracy jest jeszcze przechowywana.

Przyjmij sobie jeden arkusz do przechowywania danych. Daj mu trudną nazwę, której użytkownik się nie spodziewa, i nadaj mu właściwość `Visible = 2` (rysunek 5.9).

Rysunek 5.9.

W naszym skoroszycie z poziomu Excela możemy mieć arkusze widoczne `xlSheetVisible` (wartość liczbowa -1) lub ukryte `xlSheetHidden` (wartość liczbowa 0). VBA dorzuca do tej kolekcji wartość `xlSheetVeryHidden` (liczbowo 2). Arkusz jest i można w nim przechowywać dane, ale nie jest dostępny z poziomu „zwykłego” Excela



Dlaczego trudna nazwa? Bowiem arkusz ten nie będzie widoczny dla użytkownika (właściwość `xlSheetVeryHidden` sprawia, że arkusza nie widać na liście arkuszy do odkrycia), jednak dla Excela arkusz ten oczywiście istnieje i program obrazi się, jeżeli niczego nieświadomy użytkownik wybierze dla swojego nowego arkusza taką samą nazwę. Potem zmodyfikuj procedury z poprzedniego przykładu. Tym razem, zamiast przechowywać wartość w zmiennej dodatek, umieścimy ją w komórce `A1` ukrytego arkusza. Stamtąd nic jej nie ruszy.

```
Private Sub Workbook_Open()
    'zerowanie komórki A1
    ThisWorkbook.Sheets("ustawienia_a").Cells(1, 1).ClearContents
    'zmiana wartości komórki A1 jeżeli dodatek jest uruchomiony i wyjście z procedury lub...
    If AddIns("Analysis ToolPak").Installed = True Then
        ThisWorkbook.Sheets("ustawienia_a").Cells(1, 1) = 1
    Exit Sub
    Else
    '... załadowanie dodatku
    AddIns("Analysis ToolPak").Installed = True
    End If
End Sub
```

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
    ' odczytanie wartości komórki A1 i wyjście jeżeli dodatek
    ' był załadowany przed uruchomieniem tego skoroszytu...

    If Sheets("ustawienia_a").Cells(1, 1) = 1 Then Exit Sub

    '... lub odinstalowanie dodatku
    AddIns("Analysis ToolPak").Installed = False
End Sub
```

Przykład 26. Formatowanie warunkowe w zależności od wartości z innej komórki

Przykład ten jest uzupełnieniem formatowania warunkowego dostępnego w Excelu. Za jego pomocą możemy wpływać na formatowanie komórki zależnie od wartości innej komórki (rysunek 5.10). Wykorzystamy w tym celu zdarzenie Worksheet_Change, które — jak już wiesz — należy umieścić w module arkusza, którego dotyczy.

```
Private Sub Worksheet_Change(ByVal Target As Range)
```

```
    rzad = Target.Row
    kolumna = Target.Column
```

```
    'procedura działa tylko dla kolumny A
```

```
    If kolumna > 1 Then Exit Sub
```

```
    kolor_czcionki = Int(255 - Target.Value)
    If kolor_czcionki > 255 Then kolor_czcionki = 255
    If kolor_czcionki < 0 Then kolor_czcionki = 0
```

```
    kolor_komorki = Int(Target.Value)
    If kolor_komorki > 255 Then kolor_komorki = 255
    If kolor_komorki < 0 Then kolor_komorki = 0
```

```
    'Procedura zmienia kolory czcionki i wypełnienie
    'w komórce w kolumnie B w tym samym wierszu
```

```
    With Cells(rzad, kolumna + 1)
        .Font.Color = RGB(kolor_czcionki, kolor_czcionki, kolor_czcionki)
        .Interior.Color = RGB(kolor_komorki, kolor_komorki, kolor_komorki)
    End With
End Sub
```

Rysunek 5.10.

Zależnie od wartości wpisanej do kolumny A otrzymujemy formatowanie komórek w kolumnie B. Absurdalnie proste

	A	B	C
1	0	abc	
2	50	abc	
3	100	abc	
4	150	abc	
5	200	abc	
6	250	abc	
7			
8			