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

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 któregokolwiek 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 komórki
- Ochrona przed zmianą nazwy arkusza i dynamiczne ukrywanie wierszy
- Hiperłącze 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

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.

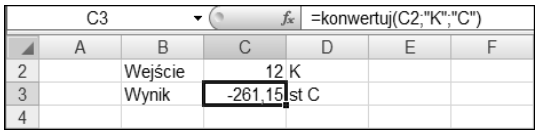
Przeliczenie według zależności:

```
1 st C = (1*1,8)+32 st F = 1+273,15 K
1 K = 1-273,15 st C = ((1-273,15)*1,8)+32 st F
1 st F = (1-32)/1,8 st C = ((1-32)/1,8)+273,15 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).

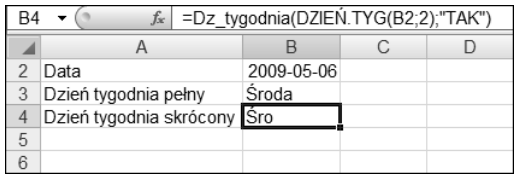


Rysunek 5.1.
Mimo wielkiego bogactwa funkcji wbudowanych każdy z nas może dodać do palety podstawowej garść własnych gadżetów. Zanim jednak opracujesz własną funkcję, dobrze poszukaj — prawdopodobnie ktoś już przygotował ją za Ciebie

```
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*

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)
```

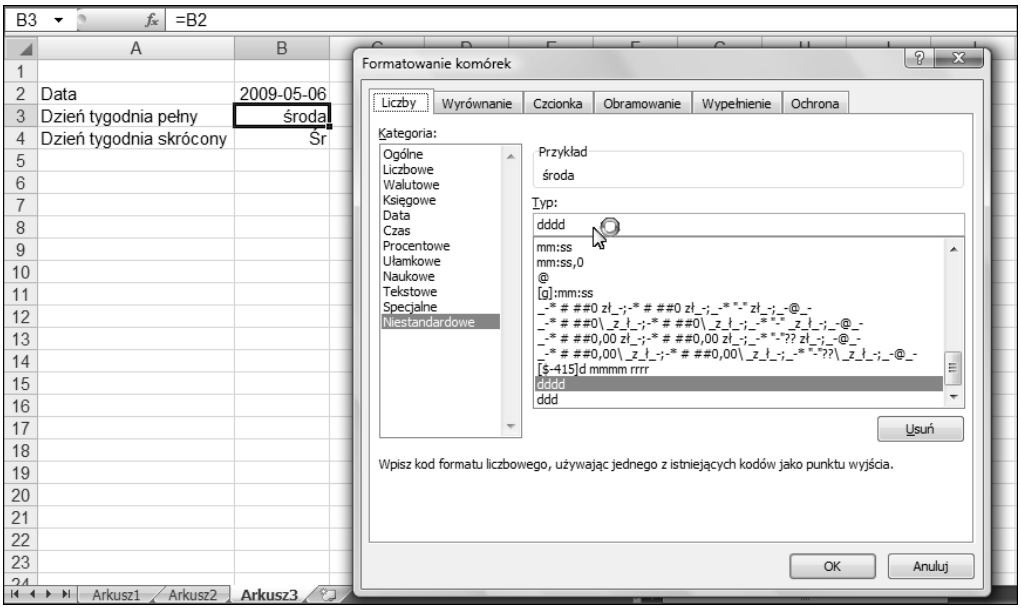
spowoduje nieprawidłowe działanie.

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+1) wybierz niestandardowe formaty komórek: dddd dla B3 i ddd dla B4*

```
Function SLOWNIE(x)

Dim Setki(10) As String
Dim dziesiatki(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"
dziesiatki(0) = ""
dziesiatki(1) = "dziesięć"
dziesiatki(2) = "dwadzieścia"
dziesiatki(3) = "trzydzieści"
dziesiatki(4) = "czterdzieści"
dziesiatki(5) = "pięćdziesiąt"
dziesiatki(6) = "sześćdziesiąt"
dziesiatki(7) = "siedemdziesiąt"
dziesiatki(8) = "osiemdziesiąt"
dziesiatki(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
    liczbamld = Int(liczba / 1000000000)
'określa ile mamy milionów
    liczbamil = Int((liczba - liczbamld * 1000000000) / 1000000)
'określa ile mamy tysięcy
    liczbatys = Int((liczba - liczbamld * 1000000000 - liczbamil *
    ↪1000000) / 1000)
'określa ile mamy setek
    liczbaset = Int(liczba - liczbamld * 1000000000 - liczbamil * 1000000
    ↪- liczbatys * 1000)

miliardy:

    liczba = liczbamld
    If liczba = 0 Then GoTo miliony

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

```
    liczba = liczba - (L * 100)
    If liczba < 20 Then slowniezl = slowniezl + " " + jednostki(liczba):
    ↪GoTo piszmiliardy
    L = Int(liczba / 10)
    slowniezl = slowniezl + " " + dziesiatki(L)

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

piszmiliardy:

    If liczbamld = 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 liczbamld >= 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 piszmiliony
    L = Int(liczba / 10)
    slowniezl = slowniezl + " " + dziesiatki(L)

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

piszmiliony:

    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)
    slowniezl = slowniezl + Setki(L)

    liczba = liczba - (L * 100)
    If liczba < 20 Then slowniezl = slowniezl + " " + jednostki(liczba):
    ↪GoTo pisztysiace
    L = Int(liczba / 10)
    slowniezl = slowniezl + " " + dziesiatki(L)

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

pisztysiace:

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

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

jednostki:

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

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

    liczba = liczba - (L * 100)
    If liczba < 20 Then slowniezl = slowniezl + " " + jednostki(liczba):
    ↪GoTo piszjednostki
    L = Int(liczba / 10)
    slowniezl = slowniezl + " " + dziesiatki(L)

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

```
piszjednostki:

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

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

grosze:

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

    L = Int(liczba / 10)
    slowniegr = slowniegr + " " + dziesiatki(L)

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

piszgrosze:

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

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


wynik:

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

'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 + " " + slowniegr)

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 „slowniezl", 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 USUŃ.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 & zł
```

```
'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 dziesiatki(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"
dziesiatki(0) = ""
dziesiatki(1) = "dziesięć"
dziesiatki(2) = "dwadzieścia"
dziesiatki(3) = "trzydzieści"
dziesiatki(4) = "czterdzieści"
dziesiatki(5) = "pięćdziesiąt"
dziesiatki(6) = "sześćdziesiąt"
dziesiatki(7) = "siedemdziesiąt"
dziesiatki(8) = "osiemdziesiąt"
dziesiatki(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
    liczbamld = Int(liczba / 1000000000)
 'określa ile mamy milionów
    liczbamil = Int((liczba - liczbamld * 1000000000) / 1000000)
'określa ile mamy tysięcy
    liczbatys = Int((liczba - liczbamld * 1000000000 - liczbamil *
    ↪1000000) / 1000)
 'określa ile mamy setek
    liczbaset = Int(liczba - liczbamld * 1000000000 - liczbamil *
    ↪1000000 - liczbatys * 1000)

miliardy:

    liczba = liczbamld
    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 piszmiliardy
    L = Int(liczba / 10)
    slownie_zlotych = slownie_zlotych + " " + dziesiatki(L)
    liczba = liczba - (L * 10)
    slownie_zlotych = slownie_zlotych + " " + jednostki(liczba)

piszmiliardy:

    If liczbamld = 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 ", liczbamld)
```

```
miliony:

    liczba = liczbamil
    If liczba = 0 Then GoTo tysiace

    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 piszmiliony
    L = Int(liczba / 10)
    slownie_zlotych = slownie_zlotych + " " + dziesiatki(L)

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

piszmiliony:

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

slownie_zlotych = slownie_zlotych + przypisz(aa, bb, " milionów ",
↪" miliony ", liczbamil)


tysiace:

    liczba = liczbatys
    If liczba = 0 Then GoTo jednostki

    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 pisztysiace
    L = Int(liczba / 10)
    slownie_zlotych = slownie_zlotych + " " + dziesiatki(L)

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

pisztysiace:

    If liczbatys = 1 Then slownie_zlotych = slownie_zlotych + " tysiąc ":
    ↪GoTo jednostki
    aa = Str(liczba)
    bb = Right(aa, 1)
```

```
slownie_zlotych = slownie_zlotych + przypisz(aa, bb, " tysięcy ", " tysiące ",
↪liczbatys)

jednostki:

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

    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 piszjednostki
    L = Int(liczba / 10)
    slownie_zlotych = slownie_zlotych + " " + dziesiatki(L)

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

piszjednostki:

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

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

grosze:

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

    L = Int(liczba / 10)
    slowniegr = slowniegr + " " + dziesiatki(L)

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

piszgrosze:

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

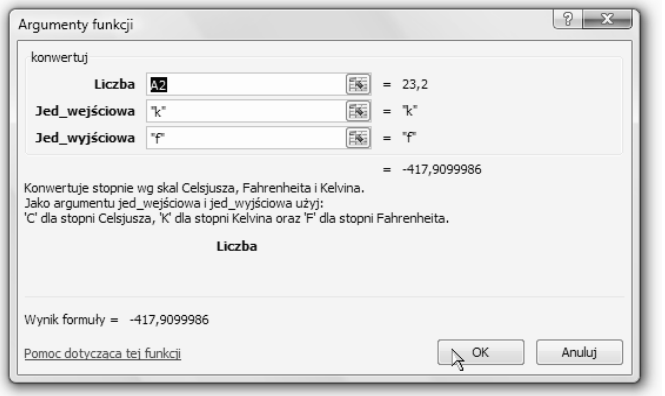
```
    If aa >= 5 And aa < 20 Then slowniegr = slowniegr + " groszy": GoTo wynik
    If bb >= 2 And bb < 5 Then slowniegr = slowniegr + " grosze": GoTo wynik
    If aa = 1 Then slowniegr = slowniegr + " grosz": GoTo wynik
    If aa >= 5 Then slowniegr = slowniegr + " groszy"


wynik:

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

'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 +
➥" " + slowniegr)

End Function
```

W powtarzalnych miejscach programu wywoływana jest funkcja *przypisz*, której wynik doklejany jest do naszej zmiennej `slownie_zlotych`. 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

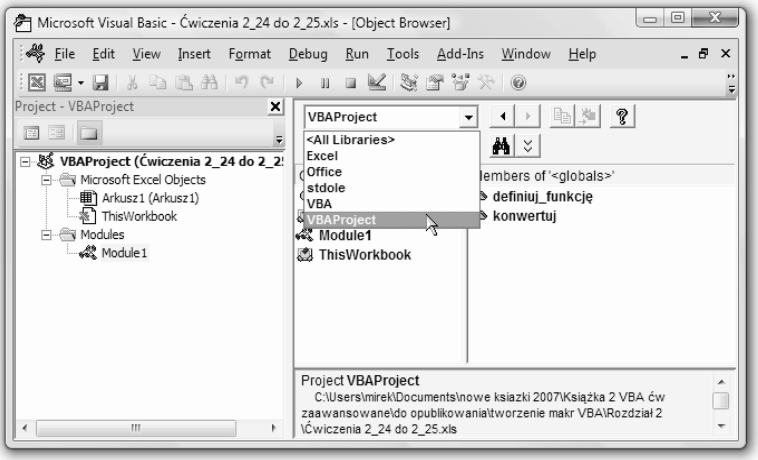
Pisząc funkcję użytkownika, możesz zadbać o jej opis i inne właściwości. To bardzo ułatwia korzystanie z niej. Przykład 13 będzie tu dobrą ilustracją. Funkcja *Konwertuj* posiada trzy parametry. Jaką jednak powinny mieć postać? Wie o tym tylko jej twórca i może jeszcze ten, kto będzie miał dość czasu 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 tworzysz 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*

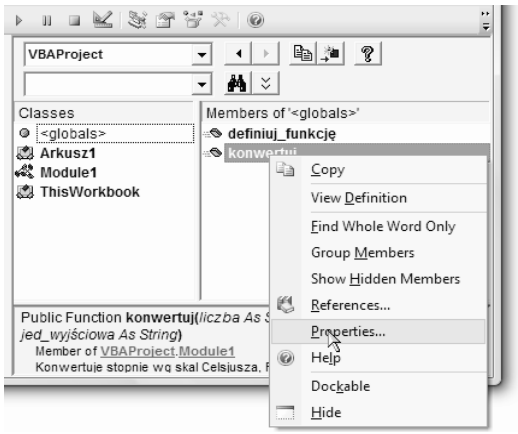
W tym przykładzie nie będziemy pisać kodów VBA, mimo że edytor będzie nam potrzebny.

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*.

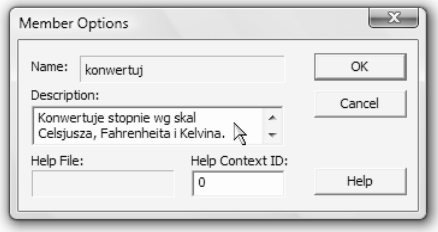


Rysunek 5.5. *Przeglądarka obiektów. Użyteczne narzędzie dla deweloperów VBA. Naprawdę warto się z nim zaprzyjaźnić*

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 zapadniesz użytkownikom w pamięć. Tym błyskotliwym ruchem być może przysłużysz się czyimś 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.

Mając powyższą wiedzę, pomyśl, jak łatwo byłoby stworzyć „samopiszący się" arkusz Excela. Możesz w taki sposób wymusić na użytkowniku 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
```

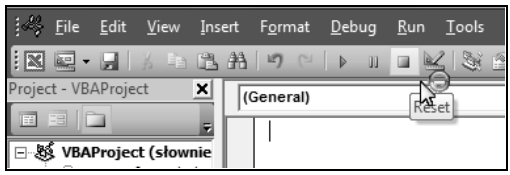
Przykład 24. Zamknięcie dodatku uruchomionego przez nasz skoroszyt

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 (pozostańmy 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
```

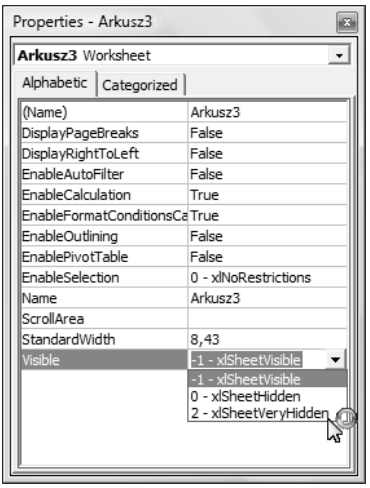
... i przy zamknięciu.

```
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

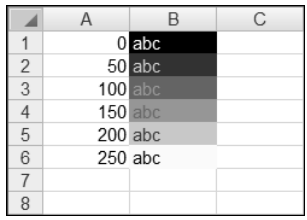
Dlaczego trudna nazwa? Bowiem arkusz ten nie będzie widoczny dla użytkownika (właściwość `xlSheetVeryHidden` sprawia, że arkusza nie widać na liście arkuszy do odkrycia), jednak dla Excela arkusz ten oczywiście istnieje i program obrazi się, jeżeli niczego nieświadomy użytkownik wybierze dla swojego 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
```



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

Rysunek 5.10. *Zależnie od wartości wpisanej do kolumny A otrzymujemy formatowanie komórek w kolumnie B. Absurdalnie proste*