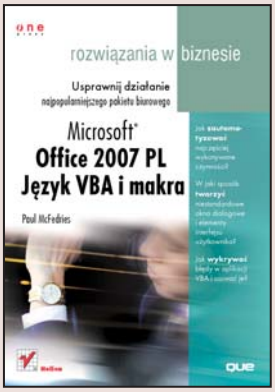
Microsoft Office 2007. Język VBA i makra. Rozwiązania w biznesie



Autor: Paul McFedries
Tłumaczenie: Piotr Pilch
ISBN: 978-83-246-1349-6
Tytuł oryginału: VBA for the 2007 Microsoft(R) Office System (Business Solutions)
Format: 170x230, stron: 472

Usprawnij działanie najpopularniejszego pakietu biurowego

- Jak zautomatyzować najczęściej wykonywane czynności?
- W jaki sposób tworzyć niestandardowe okna dialogowe i elementy interfejsu użytkownika?
- Jak wykrywać błędy w aplikacji VBA i usuwać je?

Możliwości i funkcjonalność pakietu Office 2007 są ogromne, jednak korzystanie z wielu z nich wymaga od użytkowników nieustannego powtarzania tych samych czynności. Ręczne wybieranie sekwencji poleceń z menu, zaznaczanie fragmentów tekstu lub zakresu komórek, przypisywanie reguł do filtrowania poczty – to wszystko jest pracochłonne i z czasem staje się uciążliwe dla osoby korzystającej z Office 2007. Na szczęście, twórcy pakietu zaimplementowali w nim niezwykle przydatną funkcję tworzenia i wykorzystywania makr, czyli zarejestrowanych sekwencji operacji, które można wywoływać tak samo jak zwykłe polecenia z menu. Każdy z programów wchodzących w skład Office'a posiada taką możliwość, a stosowanie jej leży w zakresie możliwości wszystkich użytkowników.

Książka „Office 2007. Język VBA i makra. Rozwiązania w biznesie" to podręcznik dla wszystkich, którzy chcą przyspieszyć i usprawnić pracę z tym pakietem biurowym. Czytając ją, nauczysz się tworzyć makra i korzystać z języka Visual Basic for Applications, w którym są pisane. Dowiesz się, z jakich elementów składa się makro, jak uruchomić edytor VBA i w jaki sposób wykorzystywać w makrach różne polecenia. Poznasz techniki budowania makr dla Worda, Excela, PowerPointa i Outlooka. Przeczytasz także o tworzeniu niestandardowych okien dialogowych, modyfikowaniu interfejsu użytkownika aplikacji z pakietu Office, wyszukiwaniu błędów w kodzie źródłowym i usuwaniu ich.

- Rejestrowanie i uruchamianie makr
- Korzystanie z edytora VBA
- Elementy języka VBA
- Obiekty Office w programach VBA
- Przetwarzanie tekstu za pomocą makr
- Obróbka zakresów komórek w Excelu
- Makra dla PowerPointa i Accessa
- Obsługa wiadomości pocztowych w Outlooku
- Tworzenie okien dialogowych i formularzy
- Modyfikowanie interfejsu Wstążka pakietu Office 2007
- Podział projektu na moduły
- Wykrywanie i usuwanie błędów

Korzystaj z makropoleceń i zwiększ efektywność swojej pracy!

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

Spis treści

II WYKORZYSTANIE JĘZYKA VBA PODCZAS PRACY

Tworzenie własnych makr

2

Umożliwienie interpreterowi języka VBA wykonania całości pracy przez zarejestrowanie makra jest prostą metodą automatyzowania zadań, z której będzie się często korzystać. Aby jednak w jak największym stopniu skorzystać z możliwości języka VBA, trzeba opanować umiejętność programowania w pełnym zakresie. Oznacza to pisanie własnych makr od podstaw lub na bazie zarejestrowanego makra.

Dlaczego warto sprawiać sobie tyle kłopotu? Oto tylko niektóre z korzyści, jakie się dzięki temu uzyska:

- Jeśli popełni się błąd podczas rejestrowania makra, zwłaszcza takiego, które wymaga dużej liczby kroków, do usunięcia pomyłki wystarczy prosta modyfikacja kodu VBA makra zamiast ponownego rejestrowania wszystkiego od początku.
- Uzyskuje się pełną kontrolę nad każdym makrem. Oznacza to, że zapewnia się, iż makra realizują dokładnie to, czego się od nich oczekuje.
- Można wykorzystać ukryte możliwości języka VBA do modyfikowania programów pakietu Office i zrealizowania kilku robiących wrażenie zadań programistycznych, które zwyczajnie nie są możliwe w ramach procesu rejestrowania makra.

Aby ułatwić Czytelnikowi urzeczywistnienie powyższych korzyści i wielu innych rzeczy, w niniejszym rozdziale zaprezentowano podstawy dotyczące pisania prostych procedur i funkcji, a także wyjaśniono, jak obsługiwać edytor VBA (ang. *Visual Basic Editor*), będący narzędziem, które język VBA oferuje do ręcznego tworzenia makr. Rozdział ten stanowi przygotowanie do kilku następnych rozdziałów, w których dokładniej omówiono szczegóły języka VBA.

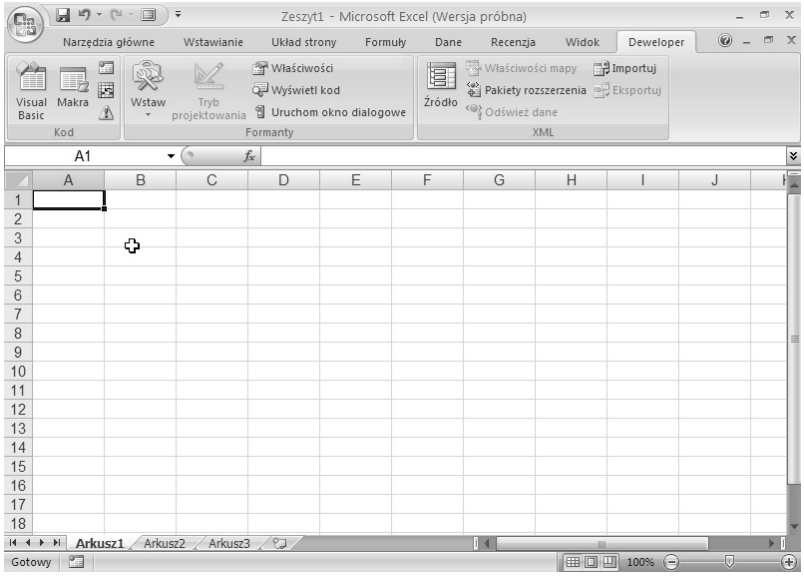
W TYM ROZDZIALE:

Uaktywnianie karty Deweloper interfejsu Wstążka

Jeśli kod VBA będzie tworzony regularnie, przez wyświetlenie karty *Deweloper* w obrębie interfejsu Wstążka pakietu Office 2007 można sprawić, że część zadań programistycznych stanie się trochę bardziej efektywna. W przypadku tej karty wystarczy jedno kliknięcie, żeby uzyskać dostęp do wielu elementów związanych z językiem VBA. A zatem kartę *Deweloper* warto uaktywnić. W tym celu należy wykonać następujące kroki:

1. Wybrać pozycję *Przycisk pakietu Office/Opcje programu* ***Aplikacja*** (w miejsce łańcucha ***Aplikacja*** należy wstawić nazwę używanego programu pakietu Office, czyli na przykład Word lub Excel).
2. W obrębie karty *Popularne* kliknąć opcję *Pokaż kartę Deweloper na Wstążce*.
3. Kliknąć przycisk *OK*.

Warto zauważyć, że wyświetlenie karty *Deweloper* w jednym programie pakietu Office spowoduje pojawienie się jej w pozostałych aplikacjach pakietu. Rysunek 2.1 przedstawia kartę *Deweloper* wyświetloną w Excelu.

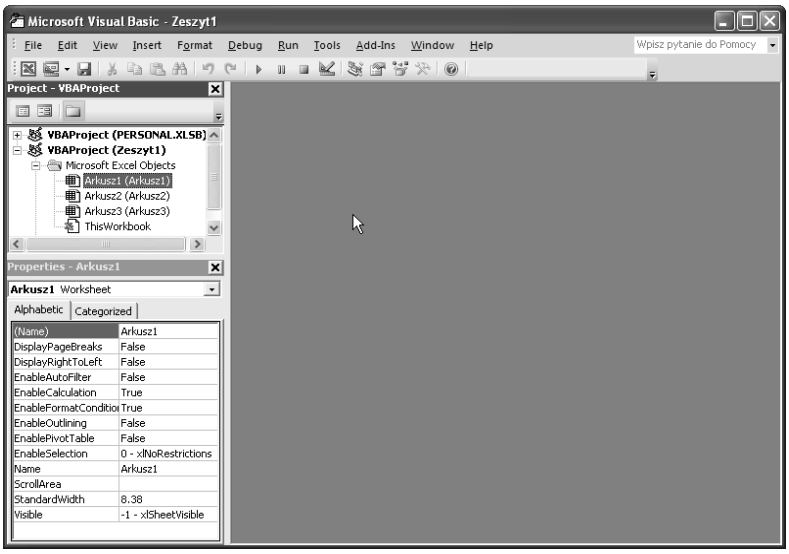


Rysunek 2.1. Wyświetlanie karty Deweloper w celu ułatwienia dostępu do niektórych elementów języka VBA

Uruchamianie edytora VBA

W celu wyświetlenia okna edytora VBA w obrębie dowolnego programu pakietu Office należy wybrać pozycję *Deweloper/Visual Basic* (warto zauważyć, że dla uproszczenia w przykładach zamieszczonych w rozdziale używana jest jedna aplikacja pakietu Office, a mia-

nowicie Excel). Rysunek 2.2 prezentuje nowe okno, które jest otwierane (trzeba jednak mieć świadomość, że okno, które się zobaczy, może się nieznacznie różnić od pokazanego poniżej).



Rysunek 2.2. W oknie edytora VBA można poprawiać i edytować makra

WSKAZÓWKA

Okno edytora VBA można również wyświetlić przez wciśnięcie kombinacji klawiszy *Alt+F11*. W rzeczywistości kombinacja ta powoduje przełączanie między edytorem VBA i aktualnie używanym programem pakietu Office.

Omówienie edytora VBA

Zasady funkcjonowania edytora VBA są proste. Jest to niezależny program, którego celem jest nic innego niż ułatwienie tworzenia i edytowania makr VBA (w kręgach zawodowych programistów edytor VBA jest nazywany *zintegrowanym środowiskiem programowania IDE* (ang. *Integrated Development Environment*)).

Po pierwszym uruchomieniu edytora VBA nie zobaczy się zbyt wiele. Po lewej stronie okna edytora znajdują się dwa okna — *Project* i *Properties*. Na tym etapie nie trzeba się przejmować drugim oknem (więcej na jego temat można znaleźć w rozdziale 5., „Praca z obiektami"). Okno *Project* (jego techniczna nazwa to *Project Explorer*) prezentuje zawartość bieżącego projektu VBA. Posługując się najprostszymi terminami, *projekt* można określić jako plik pakietu Office wraz ze wszystkimi powiązanymi z nim składnikami języka VBA, uwzględniającymi makra i formularze użytkownika (formularze omówiono w rozdziale 12., „Tworzenie niestandardowych okien dialogowych VBA").

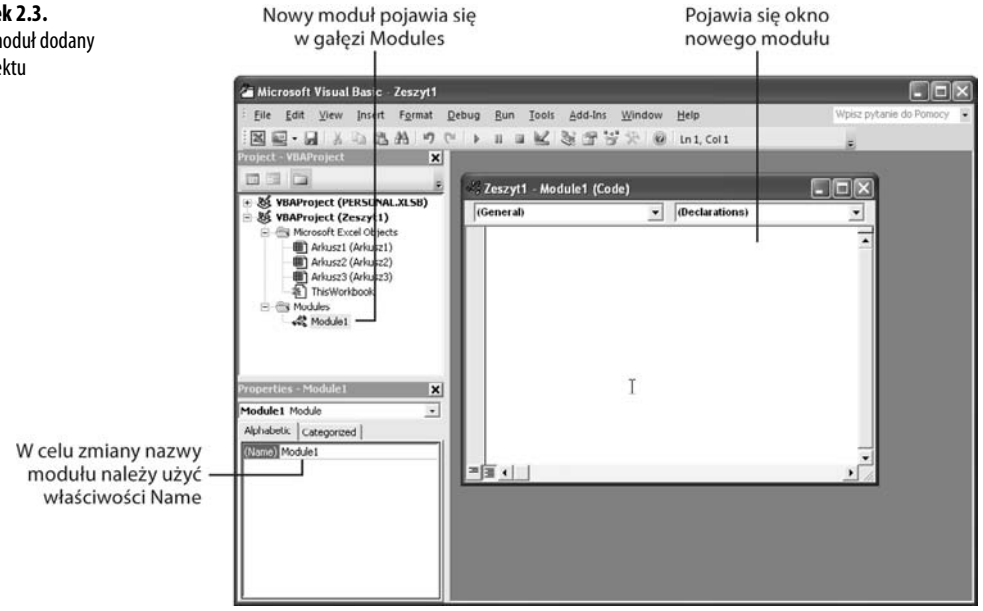
Tworzenie nowego modułu

Większość zadań realizuje się w obrębie jednego lub większej liczby modułów edytora VBA. *Moduły* są oknami służącymi do przechowywania kodu źródłowego. Jeśli w poprzednim rozdziale zarejestrowano jakieś makra, można już dysponować modułem. Na wszelki wypadek poniżej podano kroki potrzebne do utworzenia nowego modułu.

1. W oknie *Project Explorer* widocznym z lewej strony okna edytora VBA kliknąć projekt, w którym zamierza się umieścić nowy moduł. Oto kilka uwag, o których trzeba pamiętać:
   - Jeśli w Wordzie cały czas mają być dostępne makra nowego modułu, należy kliknąć projekt *Normal* (w efekcie moduł zostanie dodany do szablonu *Normal*).
   - Jeżeli w Excelu cały czas mają być dostępne makra nowego modułu, należy kliknąć projekt *PERSONAL.XLSB* (w efekcie moduł zostanie dodany do skoroszytu *PERSONAL*).

➔ Trzeba pamiętać, że nie zobaczy się projektu *PERSONAL.XLSB*, dopóki w skoroszycie *Skoroszyt makr osobistych* nie zapisze się co najmniej jednego zarejestrowanego makra (należy zapoznać się z punktem „Rejestrowanie makra Excela" na stronie 27).

   - Jeśli makra nowego modułu w dowolnym programie mają być dostępne tylko po otwarciu określonego dokumentu, należy kliknąć ten dokument lub jeden z jego obiektów.
2. Wybrać pozycję *Insert/Module*. Edytor VBA utworzy nowy moduł i otworzy go (rysunek 2.3).



Rysunek 2.3. Nowy moduł dodany do projektu

3. W oknie *Properties* za pomocą właściwości *Name* zmienić nazwę modułu, a następnie wcisnąć klawisz *Enter* (opcjonalny krok).

➔ Aby opanować umiejętność eksportowania, usuwania i zmieniania nazw modułów, należy zapoznać się z podrozdziałem „Praca z modułami" na stronie 357.

Otwieranie istniejącego modułu

Jeśli w oknie *Project* istnieje już moduł, w celu jego otwarcia należy wykonać następujące kroki:

1. W oknie *Project* otworzyć projekt przez kliknięcie znaku plusa (+) widocznego z lewej strony.
2. W obrębie właśnie otwartego projektu wyświetlić zawartość gałęzi *Modules* przez kliknięcie znaku plusa (+) widocznego z lewej strony.
3. Dwukrotnie kliknąć nazwę modułu, który ma zostać otwarty. Edytor VBA otworzy okno modułu i wyświetli jego kod VBA.

Procedury VBA

Zanim Czytelnik zacznie pisać własne makra, powinien poświęcić chwilę na zrozumienie tego, co właściwie będzie tworzył. Podczas ręcznego tworzenia kodu VBA (lub rejestrowania makra zgodnie z opisem zamieszczonym w rozdziale 1.) to, co się uzyskuje, jest określane mianem *procedury*. Ogólnie rzecz biorąc, w języku VBA procedura jest zbiorem powiązanych ze sobą instrukcji, które tworzą moduł i realizują określonego typu zadanie (*instrukcja* jest poleceniem nakazującym interpreterowi języka VBA wykonanie konkretnego zadania). Na potrzeby książki użyto dwóch odmian procedury VBA — makr poleceń i funkcji użytkownika. Oto podsumowanie różnic:

- *Makro poleceń* to najczęściej używany typ procedury. Zwykle makro to zawiera instrukcje odpowiadające opcjom interfejsu Wstążka i innym poleceniom programu. Cechą wyróżniającą makra poleceń jest to, że podobnie do zwykłych poleceń aplikacji, oddziałują na swoje otoczenie (przykładowo, w przypadku Worda oznacza to, że makro ma wpływ na bieżący dokument, fragment tekstu itp.). Niezależnie od tego, czy otwiera się dokument, formatuje tekst lub wstawia akapit, makra poleceń *wprowadzają zmiany* (należy zapoznać się z poniższym podrozdziałem „Tworzenie makra poleceń").
- *Funkcje użytkownika* działają tak jak wbudowane funkcje programu. Ich cechą charakterystyczną jest to, że akceptują wartości wejściowe, a następnie przetwarzają je i zwracają wynik (należy zapoznać się z podrozdziałem „Tworzenie funkcji użytkownika").

Tworzenie makra poleceń

Jak wspomniano na początku rozdziału, rejestrowanie makra ma ograniczone możliwości, ponieważ istnieje mnóstwo elementów makra, do których nie można uzyskać dostępu za pomocą myszy lub klawiatury, bądź przez wybranie pozycji menu. Przykładowo, w przypadku Excela język VBA oferuje kilkadziesiąt informacyjnych funkcji makr, które zwracają dane dotyczące między innymi komórek, arkuszy i obszarów roboczych. Ponadto funkcje sterujące VBA umożliwiają dodanie prawdziwych struktur programowania, takich jak pętla, rozgałęzienie i instrukcja decyzyjna (należy zajrzeć do rozdziału 6., „Kontrolowanie kodu VBA").

W celu uzyskania dostępu do tych elementów makra trzeba od podstaw napisać własne procedury VBA. Jest to prostsze, niż mogłoby się wydawać, gdyż wszystko, co trzeba zrobić, to wprowadzić do modułu zestaw instrukcji.

UWAGA

Choć w niniejszym podrozdziale omówiono tworzenie makr VBA, zdaję sobie sprawę z oczywistego paradoksu — jak można pisać makra, jeśli jeszcze nie zdobyło się na ich temat żadnej wiedzy? Celem następnych czterech rozdziałów jest zaznajomienie Czytelnika z instrukcjami i funkcjami języka VBA. Informacje zamieszczone w tym podrozdziale i w następnych rozdziałach będą służyć jako baza do poszerzania umiejętności programowania w języku VBA.

Pisanie makra poleceń

Po otwarciu i uaktywnieniu okna modułu, w celu utworzenia własnego makra poleceń należy wykonać następujące kroki:

1. Otworzyć moduł, który ma być zastosowany dla funkcji.
2. Znak kursora umieścić w miejscu, w którym rozpocznie się pisanie makra (trzeba zadbać o to, żeby znak kursora nie znajdował się w obrębie istniejącego makra).
3. Z menu *Insert* wybrać pozycję *Procedure*. Edytor VBA wyświetli okno dialogowe *Add Procedure* (rysunek 2.4).

Rysunek 2.4. Za pomocą okna dialogowego Add Procedure można nadać nazwę nowej procedurze i kreślić typ procedury, którą zamierza się zastosować

4. Przy użyciu pola tekstowego *Name* określić nazwę makra. Oto kilka ogólnych wytycznych, które muszą być przestrzegane:
   - Nazwa musi liczyć nie więcej niż 255 znaków.
   - Pierwszy znak musi być literą lub podkreśleniem (_).
   - Nie można używać spacji ani kropki.
5. W grupie *Type* sprawdzić, czy jest aktywna opcja *Sub*.
6. Kliknąć przycisk *OK*. Interpreter języka VBA umieści w module poniższy kod (w miejscu *NazwaProcedury* będzie nazwa wprowadzona w kroku 3.).

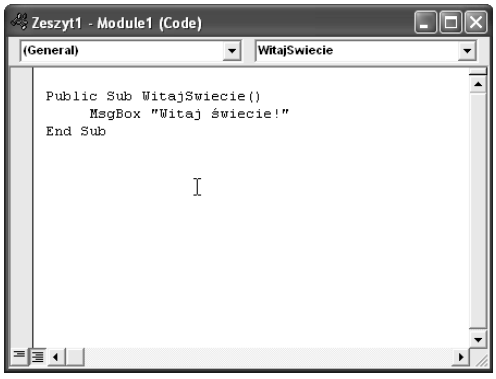
```
Public Sub NazwaProcedury()

End Sub
```

7. Między wierszami `Public Sub` i `End Sub` wprowadzić instrukcje VBA, które mają znaleźć się w makrze. Aby wstawić nowy wiersz, po wpisaniu każdej instrukcji należy wcisnąć klawisz *Enter*.

Rysunek 2.5 prezentuje prosty przykład, w którym zastosowano tylko jedną instrukcję VBA. Oto ona:

```
MsgBox "Witaj świecie!"
```



Rysunek 2.5. Przykładowe makro gotowe do wykonania

Instrukcja w przykładzie zawiera funkcję języka VBA o nazwie `MsgBox`, która służy do wyświetlania użytkownikowi prostego okna dialogowego (nazwa `MsgBox` jest skrótem od słów *Message Box*). Aby poprawić czytelność, przed wpisaniem instrukcji jednokrotnie wcisnąłem klawisz *Tab*. Więcej informacji na temat wcięć instrukcji można znaleźć w dalszej części rozdziału.

UWAGA

Kod źródłowy zamieszczony w przykładach tego rozdziału można znaleźć pod adresem *ftp://ftp.helion.pl/przyklady/ofjvba.zip*

➔ Aby uzyskać szczegóły dotyczące funkcji `MsgBox`, należy zapoznać się z punktem „Pobieranie danych wejściowych za pomocą funkcji MsgBox" na stronie 66.

Uruchamianie makra poleceń

Choć aplikacje pakietu Office oferują kilka metod uruchamiania makr poleceń VBA, najczęściej używane będą następujące dwie:

- W module należy kliknąć w dowolnym miejscu w obrębie makra, a następnie z menu *Run* wybrać pozycję *Run Sub/UserForm* lub wcisnąć klawisz *F5*.
- W oknie aplikacji pakietu Office z menu *Deweloper* należy wybrać pozycję *Makra* (lub wcisnąć kombinację klawiszy *Alt+F8*), żeby otworzyć okno dialogowe *Makro*. W razie potrzeby za pomocą listy *Makra w* należy zaznaczyć dokument zawierający makro, którego zamierza się użyć. W dalszej kolejności z listy *Nazwa makra* należy wybrać makro i kliknąć przycisk *Uruchom*.

> **UWAGA**
>
> Aby przejść bezpośrednio do dowolnego makra poleceń, które zamierza się zmodyfikować za pomocą edytora VBA, można również skorzystać z okna dialogowego *Makro*. Z listy *Nazwa makra* należy wybrać makro, a następnie kliknąć przycisk *Edycja*.

Jeśli jedną z powyższych metod zastosuje się dla przykładowego makra pokazanego na rysunku 2.5, pojawi się okno dialogowe widoczne na rysunku 2.6. W celu zamknięcia okna należy kliknąć przycisk *OK*.

Rysunek 2.6. Po uruchomieniu makra pokazanego na rysunku 2.5 zostanie wyświetlone to okno dialogowe

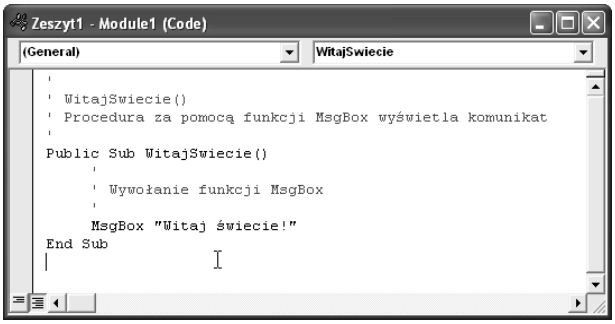
Wprowadzanie instrukcji VBA

Jak wcześniej wspomniano, wprowadzanie instrukcji VBA jest prostą sprawą. Wpisuje się kod, a następnie na końcu każdego wiersza wciska klawisz *Enter*. Dodatkowo przy rozpoczynaniu tworzenia makra poleceń zalecane jest wciśnięcie klawisza *Tab* przed wprowadzeniem pierwszego wiersza. W ten sposób uzyska się wcięcie kodu, ułatwiające jego czytanie (operacji wcięcia nie należy wykonywać dla wierszy `Public Sub` i `End Sub`, a jedynie dla instrukcji znajdujących się między nimi). Dla wygody użytkownika interpreter języka VBA zachowuje wcięcie w kolejnych wierszach. Dzięki temu wcięcie trzeba zastosować tylko w pierwszym wierszu.

Komentarz jest specjalnego typu instrukcją języka VBA, która służy do opisania czegoś związanego z procedurą. Przykładowo, wiele osób umieszcza przed procedurą kilka wierszy komentarza, aby opisać przeznaczenie procedury. Większość programistów (w tym *wszyscy* dobrzy programiści) kod procedury poszerza o komentarze objaśniające, co robią poszczególne instrukcje, a także opisujące logiczny przebieg realizowania procedury itp. Interpreter

języka VBA nie przetwarza komentarzy, które są dodawane przez programistę, żeby sobie lub innym osobom ułatwić analizowanie kodu lub umożliwić prześledzenie działania procedury. Rysunek 2.7 przedstawia prosty przykład procedury z komentarzami. Komentarze są instrukcjami rozpoczynającymi się od znaku ′.

Rysunek 2.7. Procedura VBA z komentarzami opisującymi ją i jej kod



UWAGA

Na tym początkowym etapie kariery programisty używającego języka VBA Czytelnik powinien sobie uzmysłowić korzyści wynikające ze stosowania starannego i uporządkowanego stylu programowania. Choć ludzie potrafią radzić sobie w chaotycznych środowiskach i faktycznie mają z nimi do czynienia, są o wiele mądrzejsi i bezgranicznie bardziej intuicyjni niż jakiekolwiek makro. Procedury przynależą do świata cechującego się prostą i niezmienną logiką. Programowanie zawsze będzie znacznie prostsze, jeśli logice tej będzie towarzyszyć porządek. Na szczęście są tylko dwie rzeczy, które trzeba zrobić, żeby uzyskać porządek wystarczający do bycia dobrym programistą. Pierwszą rzeczą jest wykonywanie wcięć w kodzie, a drugą stosowanie komentarzy w odpowiedniej ilości. Kwestia komentarzy jest szczególnie istotna. Każda procedura będzie znacznie łatwiejsza do odczytania (zwłaszcza wtedy, gdy kodu nie przeglądano przez kilka miesięcy), jeśli w jej kodzie swobodnie będzie się umieszczać komentarze. Ponadto dodawanie komentarzy na bieżąco podczas tworzenia kodu jest znakomitym sposobem kontrolowania własnych myśli i przeskoków logicznych.

Każdorazowo po wciśnięciu klawisza *Enter* interpreter języka VBA analizuje właśnie wprowadzony wiersz i realizuje następujące trzy zadania:

- Formatowanie za pomocą koloru każdego słowa wiersza. Domyślnie słowa kluczowe języka VBA są niebieskie, komentarze zielone, błędy czerwone, a reszta tekstu czarna.
- Wielkość znaków słów kluczowych języka VBA jest odpowiednio ustawiana. Jeśli na przykład wprowadzi się **`msgbox "Witaj świecie!"`**, po wciśnięciu klawisza *Enter* interpreter języka VBA dokona konwersji instrukcji do postaci `MsgBox "Witaj świecie!"`.
- Sprawdzane są *błędy składni*, czyli błędy występujące, gdy słowo wprowadzi się z literówką, niepoprawnie wpisze się nazwę funkcji itp. Interpreter języka VBA wyróżnia błąd składni przez wyświetlenie okna dialogowego informującego o szczegółach problemu lub przez niedokonywanie konwersji słowa (zmiana wielkości znaków lub formatowanie przy użyciu koloru).

WSKAZÓWKA

Jeżeli słowa kluczowe języka VBA zawsze wprowadza się przy zastosowaniu małych znaków, literówki będzie można wykryć, szukając tych słów, które nie zostały rozpoznane przez interpreter (inaczej mówiąc, są to słowa, w których wielkość znaków nie została zmieniona).

2

Tworzenie funkcji użytkownika

Aplikacje pakietu Office dysponują wieloma wbudowanymi funkcjami. Przykładowo, Excel ma setki funkcji (jedna z największych bibliotek funkcji spotykanych w pakietach arkuszy kalkulacyjnych). Jednak nawet pomimo tak obszernego zbioru, w dalszym ciągu napotyka się mnóstwo sytuacji, w których funkcje te nie są przydatne. Przykładowo może być konieczne obliczenie powierzchni koła o określonym promieniu lub siły grawitacji występującej między dwoma obiektami. Oczywiście obliczenia te można z łatwością przeprowadzić w arkuszu. Jeśli jednak trzeba będzie często je wykonywać, sensowne staje się utworzenie własnych funkcji, z których można w dowolnej chwili skorzystać. W następnych trzech punktach wyjaśniono, jak utworzyć funkcje użytkownika.

Ogólne informacje na temat funkcji użytkownika

Jak wcześniej wspomniano, cechą charakterystyczną funkcji użytkownika jest to, że zwracają wynik. Są one w stanie wykonywać dowolną liczbę obliczeń na liczbach, tekście, wartościach logicznych lub innych. Jednak zwykle nie mają wpływu na swoje otoczenie. Przykładowo, w arkuszu zazwyczaj nie przemieszczają aktywnej komórki, nie formatują zakresu ani nie zmieniają ustawień obszaru roboczego.

A zatem, co *można* umieścić w funkcji użytkownika? Większość tego typu funkcji składa się z jednego lub większej liczby *wyrażeń*. Wyrażenie jest określoną kombinacją wartości (na przykład liczb), operatorów (np. + i *), zmiennych (więcej o nich w rozdziale 3.), funkcji języka VBA lub funkcji aplikacji. Kombinacja ta generuje wynik (szczegółowo wyrażenia omówiono w rozdziale 4., „Tworzenie wyrażeń języka VBA”).

Wszystkie funkcje użytkownika mają taką samą podstawową strukturę. Oto ona:

```
Function NazwaProcedury (argument1, argument2, ...)
      [Instrukcje VBA]
      NazwaProcedury = zwracanaWartość
End Function
```

Oto podsumowanie różnych składników funkcji użytkownika:

- **`Function`** — słowo kluczowe identyfikujące procedurę jako funkcję użytkownika. Słowo to jest powodem, dla którego funkcje użytkownika są też nazywane *procedurami funkcyjnymi*.

- ***NazwaProcedury*** — unikatowa nazwa funkcji.
- ***argument1, argument2, ...*** — podobnie do wielu funkcji aplikacji, funkcje użytkownika pobierają argumenty. Argumenty (czasami określane mianem parametrów) zwykle mają postać jednej lub wielu wartości wykorzystywanych na wejściu przez funkcję przeprowadzającą obliczenia. Argumenty zawsze podaje się w nawiasach okrągłych za nazwą funkcji. Wiele argumentów oddziela się przecinkami (jeśli funkcja nie wymaga argumentów, niezależnie od tego za jej nazwą trzeba wstawić nawiasy).
- ***Instrukcje VBA*** — jest to kod faktycznie realizujący obliczenia. Zazwyczaj ma on postać zestawu instrukcji języka VBA i wyrażeń, które doprowadzają do uzyskania końcowego wyniku zwracanego przez funkcję.
- ***zwracanaWartość*** — ostateczny wynik obliczony przez funkcję.
- **End Function** — słowa kluczowe identyfikujące koniec funkcji.

Wszystkie funkcje użytkownika będą miały taką podstawową strukturę. W związku z tym, tworząc tego typu makra, na uwadze trzeba mieć następujące trzy rzeczy:

- Jakie argumenty funkcja pobierze?
- Jakie wyrażenia zostaną użyte w obrębie funkcji?
- Jaka wartość zostanie zwrócona?

Pisanie funkcji użytkownika

Gdy rejestruje się makro, interpreter języka VBA zawsze umieszcza kod w obrębie makra poleceń. Niestety, nie istnieje metoda rejestrowania funkcji użytkownika. Trzeba ją napisać ręcznie. Okazuje się, że zadanie to bardzo przypomina tworzenie makra poleceń od podstaw. Oto ogólne kroki, które trzeba wykonać w celu napisania funkcji użytkownika:

1. Otworzyć moduł, który ma być zastosowany dla funkcji.
2. Znak kursora umieścić w miejscu, w którym rozpocznie się pisanie funkcji (trzeba zadbać o to, żeby znak kursora nie znajdował się w obrębie istniejącego makra).
3. Z menu *Insert* wybrać pozycję *Procedure*, żeby wyświetlić okno dialogowe *Add Procedure*.
4. Przy użyciu pola tekstowego *Name* określić nazwę funkcji. Wytyczne, które muszą być przestrzegane, są takie same jak w przypadku makra poleceń. Nazwa musi liczyć nie więcej niż 255 znaków. Pierwszy znak musi być literą lub podkreśleniem (_). Nie można używać spacji ani kropki.
5. W grupie *Type* sprawdzić, czy aktywna jest opcja *Function*.
6. Kliknąć przycisk *OK*. Interpreter języka VBA umieści w module poniższy kod (w miejscu *NazwaProcedury* będzie nazwa określona w kroku 3.).

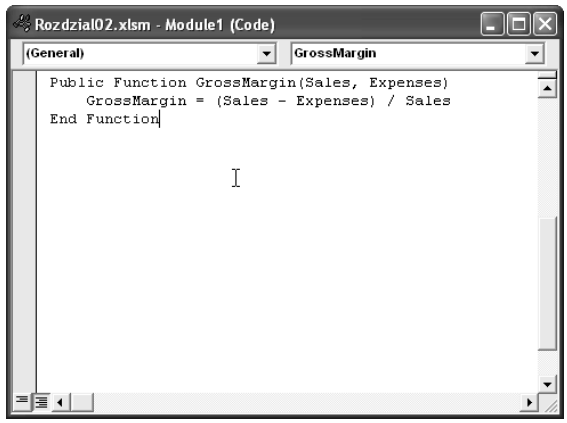
```
Public Function NazwaProcedury()

End Function
```

7. Między wierszami `Public Function` i `End Function` umieścić instrukcje VBA, które mają znaleźć się w funkcji. Aby wstawić nowy wiersz, po wpisaniu każdej instrukcji należy wcisnąć klawisz *Enter*.
8. Trzeba pamiętać o uwzględnieniu instrukcji, która definiuje zwracaną wartość. Instrukcja powinna zawierać nazwę funkcji, a za nią znak równości i zwracaną wartość.

Rysunek 2.8 prezentuje przykładową funkcję użytkownika, która oblicza i zwraca wynik. W tym celu używa następującej instrukcji VBA:

```
GrossMargin = (Sales - Expenses) / Sales
```



Rysunek 2.8. Przykładowa funkcja gotowa do zastosowania w innych procedurach

`Sales` i `Expenses` to argumenty przekazywane funkcji. Funkcja od wartości `Sales` odejmuje wartość `Expenses`, a następnie wynik dzieli przez wartość `Sales`, żeby zwrócić wartość marży brutto (`GrossMargin`).

Zastosowanie funkcji

Funkcji użytkownika nie można uruchamiać w taki sam sposób jak makra poleceń. Funkcji takiej należy użyć jako części składowej makra poleceń (a nawet innej funkcji) lub aplikacji.

W celu zastosowania funkcji w makrze poleceń należy utworzyć oddzielną instrukcję VBA, która zawiera nazwę funkcji i wszelkie wymagane przez nią argumenty (jest to określane mianem *wywoływania* funkcji). Oto prosty przykład:

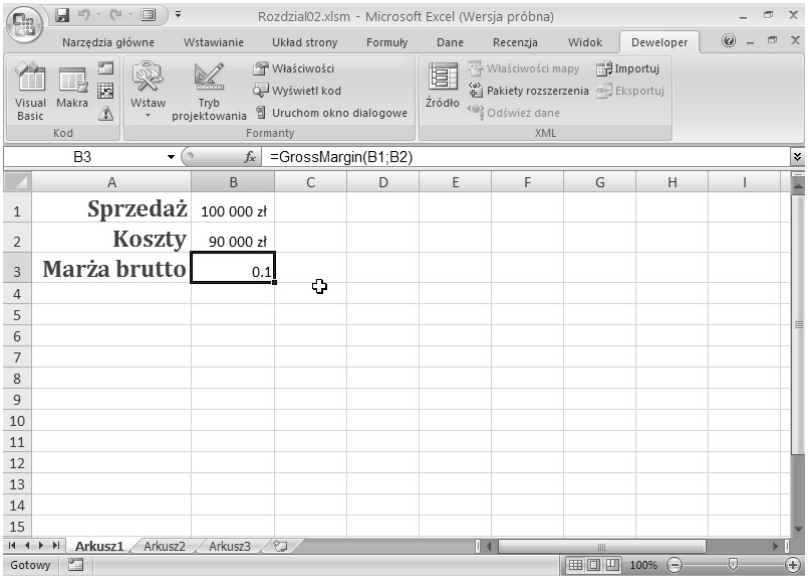
```
Public Sub GrossMarginTest1()
      MsgBox GrossMargin (100000, 90000)
End Sub
```

Procedura `Sub` wywołuje funkcję `GrossMargin` i przekazuje jej wartości `100000` i `90000` odpowiednio dla argumentów `Sales` i `Expenses`. Funkcja `MsgBox` wyświetla wynik w oknie dialogowym.

Aby użyć funkcji w aplikacji, należy odwołać się do niej. Jest to najbardziej przydatne w przypadku Excela, w którym funkcję użytkownika można zastosować w obrębie formuły arkuszowej.

W tym celu najprościej jest wstawić funkcję w komórce w taki sam sposób jak dowolną wbudowaną funkcję Excela. Inaczej mówiąc, najpierw należy wpisać nazwę funkcji, a następnie w nawiasach okrągłych wymagane argumenty. Poniżej zamieszczono przykładową formułę, która wykorzystuje funkcję `GrossMargin` i zakłada, że wartości argumentów `Sales` i `Expenses` znajdują się odpowiednio w komórkach *B1* i *B2* (rysunek 2.9).

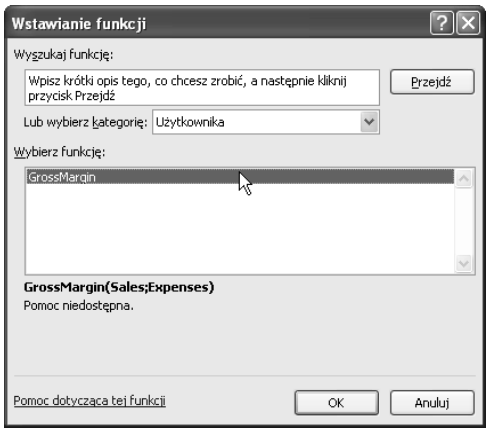
```
=GrossMargin(B1, B2)
```



Rysunek 2.9. Funkcja GrossMargin użyta w formule arkuszowej Excela

W celu wstawienia funkcji użytkownika można również posłużyć się kreatorem funkcji. Oto kroki, które trzeba wykonać:

1. Kliknąć komórkę, w której zamierza się umieścić funkcję użytkownika.
2. Z menu *Formuły* wybrać pozycję *Wstaw funkcję*, żeby otworzyć okno dialogowe *Wstawianie funkcji*.
3. Z listy *Lub wybierz kategorię* wybrać pozycję *Użytkownika*. Excel wyświetla listę funkcji użytkownika (rysunek 2.10).
4. Zaznaczyć funkcję, która ma zostać wstawiona, i kliknąć przycisk *OK*. Pojawi się okno dialogowe *Argumenty funkcji*.
5. Określić wartości lub adresy komórek przechowujących wartości argumentów funkcji, a następnie kliknąć przycisk *OK*. Excel wstawi funkcję.



Rysunek 2.10. W oknie dialogowym Wstawianie funkcji należy wybrać kategorię Użytkownika, żeby ujrzeć listę istniejących funkcji użytkownika

Wykorzystanie technologii IntelliSense

Technologię IntelliSense języka VBA można porównać do mini wersji systemu pomocy VBA Help. Oferuje ona pomoc dotyczącą składni języka VBA, która jest uaktywniana dynamicznie lub na żądanie. Powinno się sięgnąć po to niebywale przydatne narzędzie, ponieważ w trakcie lektury rozdziałów tej książki okaże się, że język VBA zawiera dziesiątki instrukcji i funkcji, a ponadto programy obsługujące ten język oferują setki obiektów, z którymi można pracować. Niewiele osób jest w stanie wszystko to zapamiętać. Poza tym niemiłym doświadczeniem będzie ciągłe szukanie poprawnej składni. W czasie wprowadzania kodu technologia IntelliSense pomaga przez oferowanie wskazówek i alternatyw. Aby Czytelnik zrozumiał, co mam na myśli, przyjrzymy się czterem najbardziej przydatnym typom pomocy oferowanej przez technologię IntelliSense.

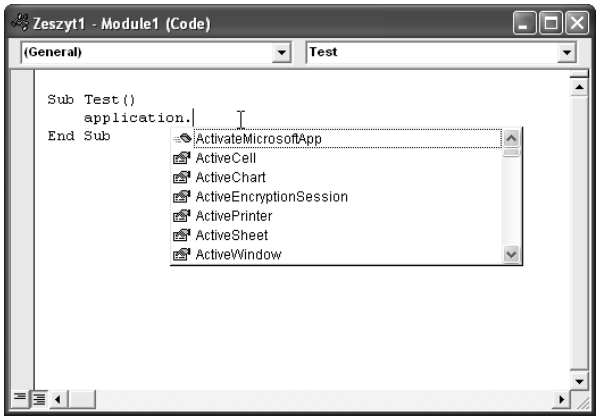
Lista właściwości lub metod

W rozdziale 4. wyjaśniono, jak pracować z obiektami udostępnianymi przez każdą aplikację obsługującą język VBA. W szczególności Czytelnik dowie się o *właściwościach* i *metodach*, które po prostu definiują cechy charakterystyczne każdego obiektu (w bardziej ogólnym sensie właściwości opisują wygląd lub działanie obiektu, a metody to, co można zrobić w przypadku określonego obiektu).

Jednak w dalszej części książki okaże się, że każdy obiekt może mieć dziesiątki właściwości i metod. W celu ułatwienia użytkownikowi poprawnego tworzenia kodu procedury, podczas wprowadzania instrukcji VBA technologia IntelliSense może wyświetlić listę dostępnych właściwości i metod. Aby to wypróbować, w obrębie edytora VBA należy uaktywnić moduł, a następnie wpisać łańcuch **`application`**, a za nim kropkę. Jak widać na

rysunku 2.11, interpreter języka VBA udostępnia menu podręczne. Jego pozycjami są właściwości i metody powiązane z obiektem `Application`. Oto metody umożliwiające korzystanie z tego menu:

- W celu wyświetlenia różnych pozycji menu należy kontynuować wprowadzanie znaków. Jeśli na przykład w Excelu wpisze się **`cap`**, interpreter języka VBA wyróżni na liście pozycję *Caption*.
- Aby umieścić pozycję w kodzie, należy ją dwukrotnie kliknąć.
- Najpierw należy zaznaczyć pozycję (przez kliknięcie lub przy użyciu klawiszy strzałek skierowanych w górę i w dół), a następnie wcisnąć klawisz *Tab*, żeby ją wstawić i kontynuować edycję tej samej instrukcji.
- Najpierw należy zaznaczyć pozycję, po czym wcisnąć klawisz *Enter*, aby ją wstawić i przejść do nowego wiersza.
- W celu ukrycia menu bez wstawiania jego pozycji należy wcisnąć klawisz *Esc*.



Rysunek 2.11. Podczas wprowadzania znaków technologia IntelliSense wyświetla dostępne właściwości i metody

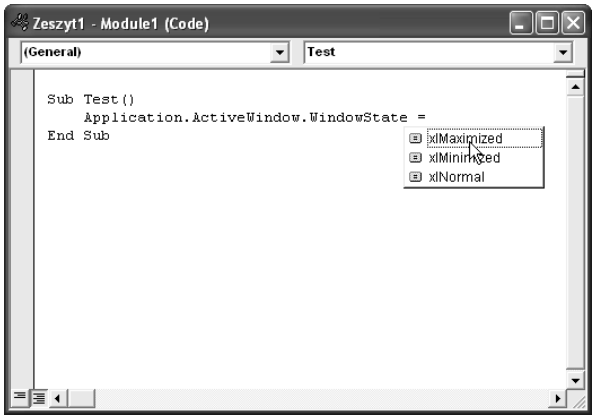
Warto zauważyć, że jeśli w celu ukrycia menu podręcznego wciśnie się klawisz *Esc*, interpreter języka VBA nie wyświetli go ponownie dla tego samego obiektu. Jeżeli zamierza się jeszcze raz skorzystać z tego menu, z menu *Edit* należy wybrać pozycję *List Properties/Methods* (lub wcisnąć kombinację klawiszy *Ctrl+J*).

Lista stałych

Technologia IntelliSense oferuje polecenie *List Constants*, które przypomina polecenie *List Properties/Methods*. W przypadku tej funkcji pojawia się menu podręczne zawierające listę dostępnych stałych powiązanych z właściwością lub metodą (*stała* jest niezmienną wartością odpowiadającą określonemu stanowi lub wynikowi; więcej na temat stałych można znaleźć w rozdziale 3.). W module można umieścić następujący przykładowy wiersz:

```
Application.ActiveWindow.WindowState=
```

Rysunek 2.12 pokazuje menu podręczne pojawiające się w Excelu. Menu zawiera listę stałych odpowiadających różnym ustawieniom właściwości `WindowState` okna. Przykładowo, w celu maksymalizacji okna należy zastosować stałą `xlMaximized`. Korzystając z tej listy, należy użyć tych samych metod co w przypadku polecenia *List Properties/Methods*.



Rysunek 2.12. Polecenie List Constants w akcji

Jeśli zamierza się ręcznie wyświetlić listę, z menu *Edit* należy wybrać pozycję *List Constants* (lub wcisnąć kombinację klawiszy *Ctrl+Shift+J*).

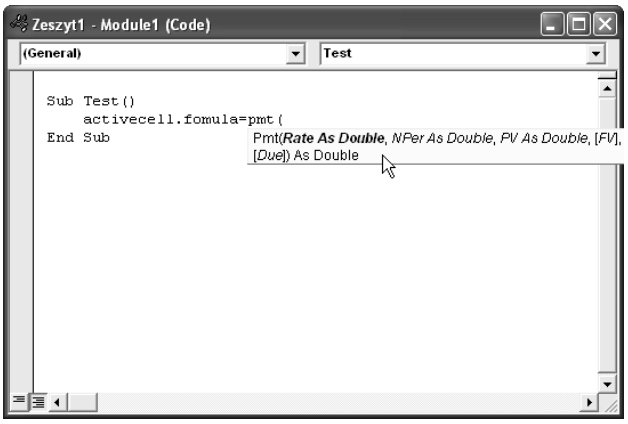
Informacja o parametrach

Wcześniej wspomniano, że na potrzeby wewnętrznych obliczeń funkcja użytkownika zwykle pobiera jeden lub więcej argumentów (parametrów). Wiele wbudowanych funkcji i instrukcji języka VBA również używa parametrów. Część tych funkcji i instrukcji stosuje nawet tuzin niezależnych argumentów! Ponieważ oczywiście składnia takich instrukcji jest bardzo złożona, łatwo popełnić pomyłkę. Aby pomóc podczas wprowadzania funkcji użytkownika lub jednej z wbudowanych funkcji czy instrukcji języka VBA, technologia IntelliSense oferuje polecenie *Parameter Info*. Jak nazwa wskazuje, funkcja ta wyświetla informacje dotyczące parametrów, które mogą być użyte w funkcji. W celu zapoznania się z przykładem w dowolnym module Excela należy wpisać następujący tekst:

```
activecell.fomula=pmt(
```

Od razu po wprowadzeniu lewego nawiasu pojawi się okienko informujące o argumentach dostępnych dla funkcji `Pmt` języka VBA (rysunek 2.13). Oto cechy charakterystyczne okienka:

- Bieżący argument jest pogrubiany. Gdy wprowadzi się argument, a następnie przecinek, interpreter języka VBA pogrubi następny argument.
- Opcjonalne argumenty są umieszczone w nawiasach kwadratowych (*[ ]*).



Rysunek 2.13. Polecenie Parameter Info pokazuje argumenty bieżącej funkcji lub instrukcji

- Różne instrukcje `As` (na przykład `As Double`) określają *typ danych* każdego argumentu. Typy danych zostaną omówione w następnym rozdziale. Na tym etapie należy wiedzieć, że identyfikują rodzaj danych powiązany z każdym argumentem (tekst, wartości liczbowe itp.).
- W celu ukrycia okienka należy wcisnąć klawisz *Esc*.

Oczywiście technologia IntelliSense umożliwia też ręczne wyświetlenie informacji o parametrach. Wystarczy z menu *Edit* wybrać pozycję *Parameter Info* (lub wcisnąć kombinację klawiszy *Ctrl+Shift+I*).

Uzupełnianie słów kluczowych

Ostatnie z omawianych poleceń technologii IntelliSense nosi nazwę *Complete Word*. Powoduje ono, że interpreter języka VBA uzupełnia słowo kluczowe, które zaczęto wpisywać. Dzięki temu można trochę odciążyć palce. Aby skorzystać z polecenia *Complete Word*, najpierw należy wprowadzić kilka liter słowa kluczowego, a następnie z menu *Edit* wybrać pozycję *Complete Word* (lub wcisnąć kombinację klawisza *Ctrl* i spacji).

Jeżeli wprowadzono liczbę liter wystarczającą do identyfikacji unikatowego słowa kluczowego, technologia IntelliSense uzupełni je. Jeśli na przykład wpisano **`appl`** i wybrano pozycję *Complete Word*, technologia IntelliSense zamieni ten łańcuch na słowo *Application*. Gdy jednak istnieje wiele słów kluczowych rozpoczynających się od wprowadzonych liter, technologia IntelliSense wyświetli menu podręczne, z którego można wybrać żądane słowo.

Zamykanie edytora Visual Basic Editor

Po zakończeniu prac związanych z kodem języka VBA można zamknąć edytor VBA. W tym celu należy skorzystać z jednej z następujących metod:

- Z menu *File* należy wybrać polecenie *Close and Return to Aplikacja*. W miejscu łańcucha *Aplikacja* będzie widoczna nazwa używanego programu (na przykład *Microsoft Excel*).
- Wciśnięcie kombinacji klawiszy *Alt+Q*.

Odnośniki

- Aby dowiedzieć się więcej na temat stosowania zmiennych w kodzie VBA, należy zajrzeć do rozdziału 3., „Zmienne programów" na stronie 53.
- Szczegółowe omówienie funkcji `MsgBox` można znaleźć w punkcie „Pobieranie danych wejściowych za pomocą funkcji MsgBox" na stronie 66.
- W celu uzyskania dokładnych informacji na temat wyrażeń należy przejść do rozdziału 4., „Tworzenie wyrażeń języka VBA" na stronie 73.
- Informacje dotyczące właściwości obiektów zamieszczono w podrozdziale „Stosowanie właściwości obiektów" na stronie 74.
- Informacje na temat metod obiektów zawarto w podrozdziale „Stosowanie metod obiektów" na stronie 98.
- W celu zapoznania się z takimi operacjami jak zmiana nazwy, eksportowanie i usuwanie modułów należy przejść do podrozdziału „Praca z modułami" na stronie 357.