helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

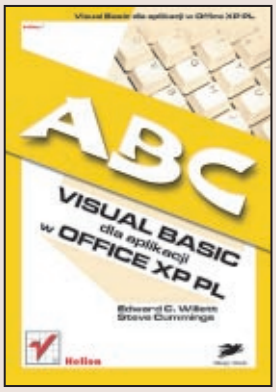
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

ABC Visual Basic dla aplikacji w Office XP PL



Autorzy: Edward C. Willett, Steve Cummings
Tłumaczenie: Bogdan Czogalik
ISBN: 83-7197-829-4
Format: B5, stron: 216

Jesteś znużony wykonywaniem powtarzających się czynności w Wordzie czy Excelu? Zmuś do nich swój komputer. Przejmij kontrolę nad MS Office i setkami innych programów. Dostosuj je do swoich potrzeb za pomocą Visual Basic for Applications (VBA) – prostego i skutecznego języka programowania, opartego na popularnym Visual Basicu. Jeśli często wprowadzasz dane do arkusza Excela możesz stworzyć specjalny formularz, który nie tylko ułatwi ich wpisywanie, ale także sprawdzi ich poprawność. Jeżeli w dokumencie Worda potrzebna jest funkcja dostępna tylko w Excelu, dzięki VBA możesz jej użyć. „ABC Visual Basic dla aplikacji w Office XP PL" jest przeznaczoną dla początkujących książką, dzięki której dowiesz się, jak można w pełni wykorzystać wszystkie możliwości pakietu Office XP.

Dzięki książce:

- dowiesz się, czym jest VBA
- nauczysz posługiwać się edytorem VBA
- poznasz zasady pisania programów w tym języku
- wejdziesz w świat programowania obiektowego
- nauczysz się usuwać błędy z programów
- stworzysz własne okna dialogowe i formularze
- poznasz sposoby tworzenia kompletnych aplikacji
- dowiesz się, jak rozpowszechniać własne programy

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Wprowadzenie 9
Kilka uwag o VBA 9
Krótka historia VBA 9
VBA jako standard 10

Rozdział 1. Visual Basic dla Aplikacji 11
Edytor języka Visual Basic dla Aplikacji 11
Moduł kodu 11
VBA — przegląd 13
VBA — pierwsze kroki 15
Makra to programy VBA 15
VBA to nie tylko makra 16
VBA — narzędzie do programowania obiektowego 17
Obiekty języka VBA 18
Modele obiektowe 18
Technologia ActiveX a VBA 18
Formanty ActiveX 18
Automatyzacja ActiveX 20
Serwery i kontrolery automatyzacji 20
Więcej informacji o VBA 20

Rozdział 2. Edytor języka VBA 23
Środowisko tworzenia programów 23
Menu edytora języka VBA 23
Praca z paskami narzędzi 24
Skróty klawiaturowe 25
Zarządzanie oknami 26
Moduły kodu, formularze oraz okna wspomagające wizualne programowanie 26
Wyświetlanie i ukrywanie okien 27
Okna, które można i których nie można zadokować 27
Zapisywanie układu okien 29
Zarządzanie projektami 29
Otwieranie okna Project 29
Okno Project 29
Właściwości projektu 31
Przeglądarka obiektów 33
Uruchamianie przeglądarki obiektów 33
Przeglądanie obiektów 34
Szukanie obiektów 35
Przeglądarka obiektów pomaga pisać kod 35

Tajemnice pisania kodu....36
Otwieranie modułu kodu....36
Tworzenie nowego modułu kodu....36
Pisanie kodu....36
Pomoc przy wpisywaniu kodu....39
Okno Properties....42
Wyświetlanie okna Properties....43
Zmiana nazwy projektu lub modułu....43
Okna służące do usuwania błędów z programów....44

Rozdział 3. Kod VBA....45

Części składowe programu....45
Przykładowy program....45
Instrukcje, procedury, moduły, projekty....47
Co to jest program?....47
Więcej o projektach....47
Praca z modułami....48
Pisanie procedur....50
Instrukcje VBA....55
Etykieta VBA....56
Zasady tworzenia nazw zmiennych i obiektów....57
Konwencje dotyczące tworzenia nazw zmiennych i obiektów....57
Czytelny kod....57
Komentarz....59
Zmienne....61
Deklarowanie zmiennych....61
Gdzie deklarować zmienne?....61
Kiedy deklarować zmienne?....62
Typy danych....62
Określanie zakresu zmiennej....64
Deklarowanie wielu zmiennych w jednym wierszu....65
Operacja przypisania wartości do zmiennej....66
Wyrażenia....67
Co znajduje się w zmiennej, zanim przypiszesz do niej wartość?....67
Praca ze stałymi....68
Deklarowanie stałych....68
Zalety stałych....69
Inne zastosowania stałych....69
Operatory....70
Kolejność wykonywania operacji w wyrażeniach....70
Porównywanie wartości....71
Konkatenacja łańcuchów znaków....73
Typy danych w szczegółach....74
Konwersja typów danych....74
Typ danych Variant....74
Numeryczne typy danych....75
Kiedy używać zmiennych typu Boolean?....75
Wartości walutowe....76
Praca z datami i godzinami....76
Łańcuchy znaków....78
Tablice....78
Elementy tablicy....79
Wymiary tablicy....79
Deklarowanie tablicy....79
Dostęp do elementów tablicy....81

Funkcje i instrukcje wbudowane ... 82
Instrukcje, funkcje, metody ... 82
Kategorie wbudowanych instrukcji, funkcji i metod ... 83
Kontrola przepływu programu ... 84
Anatomia struktury kontrolującej przepływ programu ... 84
Zagnieżdżone struktury kontrolujące przepływ programu ... 84
Wyrażenia warunkowe ... 85
Instrukcja If…Then…Else ... 86
Postać podstawowa: If…Then ... 86
Jednowierszowa instrukcja If…Then ... 86
Instrukcja If…Then…Else w działaniu ... 87
Więcej warunków w instrukcji If…Then…Else ... 87
Struktura Select Case ... 89
Tworzenie struktury Select Case ... 89
Przykładowa struktura Select Case ... 89
Instrukcja Case Else ... 90
Więcej o instrukcjach Case struktury Select Case ... 90
Tworzenie pętli ... 91
Pętla Do…Loop w działaniu ... 91
Instrukcja Exit Do ... 94
Pętla Do bez słowa kluczowego While lub Until ... 95
Pętla For…Next ... 95
Pętla For Each…Next ... 98
Instrukcja GoTo ... 98
Instrukcja GoTo w działaniu ... 99
Uwagi o instrukcji GoTo ... 99

Rozdział 4. Programowanie obiektowe ... 101

Co to jest obiekt? ... 101
Komponenty aplikacji pakietu Office XP są obiektami ... 101
Przykłady obiektów ... 101
Praktyczna definicja obiektu ... 102
Klasy a obiekty ... 102
Co to jest model obiektowy? ... 103
Formularze są obiektami ... 104
Używanie obiektów w kodzie ... 105
Poznawanie obiektów ... 106
Właściwości obiektów ... 106
Metody ... 108
Zdarzenia ... 109
Praca z obiektami ... 110
Tworzenie zmiennych obiektowych ... 111
Tworzenie nowych obiektów ... 113
Instrukcja With ... 114
Porównywanie referencji obiektowych ... 115
Zarządzanie danymi za pomocą kolekcji ... 115
Moduły klas ... 118
Tworzenie modułu klasy ... 119
Składniki definicji klasy ... 119
Obiekt Termostat w działaniu ... 121

Rozdział 5. Tworzenie bezbłędnych programów ... 123

Typy błędów ... 123
Usuwanie błędów składniowych ... 123
Usuwanie błędów ... 124

Funkcja Auto Data Tips 131
Okno Immediate 132
Okno Locals 134
Praca z oknem Locals 134
Po co edytować wartości zmiennych? 135
Jak edytuje się wartości zmiennych? 135
Okno Watch 136
Różnice między oknem Locals a oknem Watch 137
Dodawanie wyrażeń do śledzenia 137
Praca z oknem Add Watch 138
Edytowanie śledzonych wyrażeń 138
Definiowanie punktów przerwania w oknie Add Watch 139
Instrukcja On Error oraz obiekt Err 139
Skąd się biorą błędy? 140
Jak działa kod odpowiedzialny za obsługę błędów? 140
Jak napisać kod odpowiedzialny za obsługę błędów? 141

Rozdział 6. Tworzenie okien dialogowych 145

Proste okna dialogowe 145
Okna komunikatów 145
Funkcja InputBox 148
Projektowanie formularzy 149
Uruchamianie formularzy 149
Ile formularzy powinien zawierać program? 150
Drukowanie formularzy 150
Praca z formularzami 150
Tworzenie formularza 150
Dodawanie formantów do formularza 151
Właściwości formularzy i formantów 151
Okno Properties 152
Pomoc na temat właściwości 153
Zmiana wartości właściwości 153
Wybierz właściwy obiekt 154
Najważniejsze właściwości formularzy i formantów 154
Edycja formantów 156
Praca z siatką 157
Formatowanie formantów 158
Menu Format 158
Pasek narzędzi UserForm 158
Grupowanie formantów 159
Kolejność formantów na stosie obiektów 159
Formatowanie zestawów formantów 160
Inne polecenia menu Format 161
Praca z formantami 162
Kolejność dostępu do formantów 162
Klawisze dostępu 163
Sekrety formantów 163
Etykiety 164
Pola tekstowe 165
Przyciski poleceń 167
Ramki 168
Tworzenie okien dialogowych z kartami 169
Grupy opcji 170
Pola wyboru oraz przełączniki 171

Lista wyboru i lista rozwijana ... 172
Formanty, dzięki którym można wybierać wartość ... 174
Dodawanie kodu do formularza ... 175
Wczytywanie i wyświetlanie formularzy ... 176
Formularz modalny i niemodalny ... 178
Odwoływanie się do formularza za pomocą zmiennej ... 178
Ukrywanie formularza ... 179
Usuwanie formularza z pamięci operacyjnej ... 179
Zdarzenia formularza i formantów ... 179
Podstawowe zdarzenia ... 179
Dodawanie kodu do zdarzeń ... 180
Składnia procedury zdarzenia ... 181
Zdarzenie Click ... 182
Zdarzenia Change, BeforeUpdate oraz AfterUpdate ... 183
Zdarzenia KeyPress, KeyDown oraz KeyUp ... 184
Projektowanie okien dialogowych ... 184
Przycisk Zamknij lub Anuluj ... 184
Przycisk OK ... 185
Sprawdzanie poprawności danych ... 186
Modyfikowanie właściwości formularzy i formantów ... 188
Formanty ActiveX ... 189
Praca z formantami ActiveX ... 189
Formanty ActiveX w programach VBA ... 191
Niewidoczne formanty ActiveX ... 191

Rozdział 7. Tworzenie aplikacji ... 193
Tworzenie aplikacji a technologia COM ... 193
Tworzenie programu ... 195
Automatyzacja pakietu Office XP ... 196
Budowanie dodatków ... 198
Tworzenie dodatku dla konkretnej aplikacji ... 199
Tworzenie dodatku COM ... 199
Właściwość niestandardowa w roli zmiennej ... 200
Certyfikat cyfrowy ... 201
Ochrona kodu projektu ... 202
Office XP Developer Edition ... 202
Rozpowszechnianie programów ... 202
Rozpowszechnianie baz danych aplikacji Access ... 203

Skorowidz ... 205

Rozdział 1.

Visual Basic dla Aplikacji

Pakiet Office XP, choć wyposażony w wiele narzędzi i funkcji, nie może zaspokoić potrzeb wszystkich użytkowników – stąd popularność języka VBA. Jeżeli chcesz dostosować do swoich potrzeb, na przykład, aplikację Word, możesz skorzystać z narzędzi do programowania w języku Visual Basic dla Aplikacji (VBA). VBA pozwala tworzyć zarówno proste, jak i złożone programy wykorzystujące ukryte możliwości pakietu Office XP.

Edytor języka Visual Basic dla Aplikacji

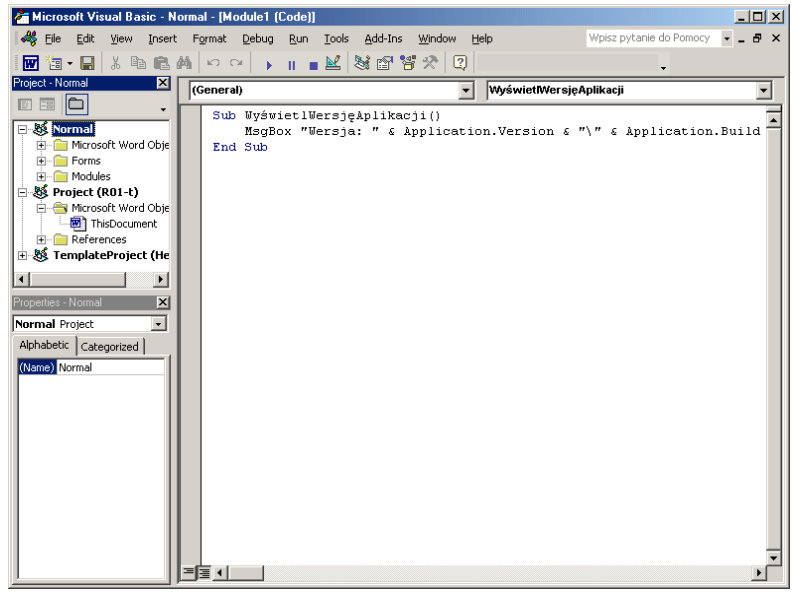
Rysunek 1.1 przedstawia edytor języka Visual Basic dla Aplikacji. Praca z tym edytorem zajmuje większość czasu, gdy tworzymy programy usprawniające działanie pakietu Office XP. Jedną z zalet VBA jest to, że umiejętności zdobyte podczas nauki programowania w jednej aplikacji można wykorzystać podczas programowania w innej.

Rozdział 2. omawia edytor języka VBA szczegółowo. Przed uruchomieniem edytora należy najpierw uruchomić jedną z aplikacji pakietu Office XP lub inną aplikację, do której dołączono VBA — edytora nie można uruchomić bez uruchomienia jego aplikacji macierzystej. Po uruchomieniu edytor języka VBA działa jak osobna aplikacja. Kiedy jednak uruchomisz procedurę napisaną w edytorze języka VBA, efekty jej działania są natychmiast widoczne w aplikacji macierzystej.

Moduł kodu

Aby zapoznać się z językiem VBA, napiszemy bardzo prostą krótką procedurę. Mimo swojej prostoty, procedura ta nie jest zupełnie bezużyteczna. Wykonaj następujące czynności:

1. Uruchom program Word.
2. Aby uruchomić edytor języka VBA, naciśnij kombinację klawiszy *Alt+F11*.



Rysunek 1.1. *Edytor języka Visual Basic dla Aplikacji zawiera okno Project, okno Properties, moduł kodu, pasek menu oraz pasek narzędzi Standard*

3. W oknie *Project* zaznacz (kliknij) element *Normal* odpowiadający szablonowi programu Word *Normal.dot*.

4. Aby wstawić moduł kodu, z menu *Insert* wybierz pozycję *Module*. Moduł ten otrzyma nazwę *Module1*.

5. W module kodu *Module1* wpisz wiersz kodu `Sub WyświetlWersjęAplikacji` i naciśnij klawisz *Enter*. W ten sposób „poinformujesz" VBA, że chcesz napisać podprogram.

6. Zwróć uwagę, że VBA dodał nawiasy do pierwszego wiersza kodu, a także dopisał wiersz kodu `End Sub`. Kursor został umieszczony między pierwszym a ostatnim wierszem procedury. Wpisz poniższy wiersz kodu między pierwszym (`Sub WyświetlWersjęAplikacji`) a ostatnim wierszem (`End Sub`) procedury:

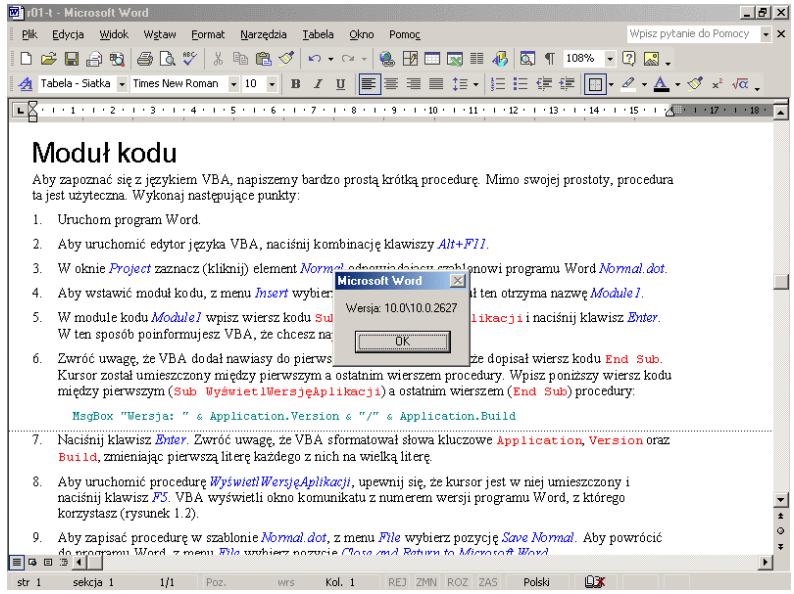
```
MsgBox "Wersja: " & Application.Version & "/" & Application.Build
```

7. Naciśnij klawisz *Enter*. Zwróć uwagę, że VBA sformatował słowa kluczowe `Application`, `Version` oraz `Build`, zmieniając pierwszą literę każdego z nich na wielką literę.

8. Aby uruchomić procedurę *WyświetlWersjęAplikacji*, upewnij się, że kursor jest w niej umieszczony, i naciśnij klawisz *F5*. VBA wyświetli okno komunikatu z numerem wersji programu Word, z którego korzystasz (rysunek 1.2).

9. Aby zapisać procedurę w szablonie *Normal.dot*, z menu *File* wybierz pozycję *Save Normal*. Aby powrócić do programu Word, z menu *File* wybierz pozycję *Close and Return to Microsoft Word*.

Procedurę zapisaną w szablonie *Normal.dot* można uruchomić w dowolnej chwili. W tym celu z menu *Narzędzia* wybierz pozycję *Makro*, a następnie — pozycję *Makra*. Word otworzy okno dialogowe *Makra*. Zaznacz nazwę procedury (makra), którą chcesz uruchomić, i kliknij przycisk *Uruchom*.



Rysunek 1.2. *Efekt działania procedury VBA (okno komunikatu z numerem wersji programu Word)*

VBA — przegląd

Oto krótki przegląd głównych cech języka VBA stosowanego w pakiecie Office XP:

- **Każda aplikacja pakietu Office XP posiada edytor języka VBA.** Oznacza to, że kod można przenosić z jednej aplikacji do drugiej.
- **Każda aplikacja pakietu Office XP posiada i udostępnia własny model obiektowy.** Aplikację Word można kontrolować, na przykład, z poziomu aplikacji Excel (obiekty, modele obiektowe i manipulowanie nimi omawiamy w dalszej części tego rozdziału). Niektóre obiekty, jak *CommandBar* i *Assistant*, nie należą do modelu obiektowego pojedynczej aplikacji, ale do modelu obiektowego pakietu Office. VBA jest serwerem automatyzacji umożliwiającym tworzenie dodatków usprawniających pracę w środowisku tworzenia programów.
- **Edytor języka VBA jest zintegrowanym środowiskiem tworzenia programów.** Jest to miejsce, w którym tworzy się i testuje moduły i formularze. Edytor języka VBA jest bardzo podobny do środowiska tworzenia programów języka VB.
- **Każda aplikacja pakietu Office XP udostępnia prawie jednakowe narzędzia do tworzenia programów.** Aplikacje, do których dołączono VBA, w tym wszystkie aplikacje pakietu Office XP z wyjątkiem aplikacji Access, korzystają ze wspólnego systemu projektowania okien dialogowych (w terminologii VBA okno dialogowe nosi nazwę UserForm). Oznacza to, że bez względu na to, w której aplikacji pracujesz, okna dialogowe tworzy się za pomocą tych samych narzędzi oraz formantów. Ponadto okno dialogowe zaprojektowane na przykład w Wordzie można wykorzystać w Excelu, często *bez konieczności dokonywania*

zmian. Dzięki formantom dostarczanym z VBA tworzy się efektowne i funkcjonalne okna dialogowe (rysunek 1.3). Poza tym można korzystać z formantów sprzedawanych przez innych producentów (patrz podrozdział na temat formantów ActiveX w dalszej części tego rozdziału).

Rysunek 1.3. *Okno dialogowe ze standardowymi formantami*

VBA w różnych aplikacjach

Mimo wysiłków firmy Microsoft zmierzających do ujednolicenia języka VBA we wszystkich aplikacjach pakietu Office, nadal istnieją różnice między odmianami VBA stosowanymi w poszczególnych aplikacjach. Aby na przykład za pomocą kodu wywołać wbudowane okno dialogowe *Zapisz jako* w programie Word, należy użyć stałej *wdDialogFileSaveAs*. Aby podobne okno dialogowe wywołać w programie Excel, należy użyć stałej *xlDialogSaveAs*. Zwróć uwagę, że stała stosowana w Excelu nie zawiera członu *File*. Warto również zauważyć, że spośród wszystkich aplikacji pakietu Office żadna nie pozwala kontrolować wbudowanych poleceń za pomocą kodu VBA tak bardzo jak Word.

- **Można definiować własne obiekty w modułach klas.** Dzięki zdefiniowaniu obiektów można korzystać z dobrodziejstw programowania obiektowego. Ten sam obiekt można wykorzystywać w różnych projektach, przez co skraca się czas tworzenia programów.
- **Technologia ADO (ActiveX Data Objects) sprawia, że dokumenty pakietu Office XP mogą pobierać dane z różnych źródeł.** Dzięki ADO można pobierać i przechowywać dane bez względu na to, czy znajdują się one na dysku twardym, serwerze grupy roboczej lub w relacyjnej bazie danych. Technologia ADO jest łatwiejsza w użyciu i szybsza niż technologie odpowiedzialne za dostęp do danych wykorzystywane w poprzednich wersjach pakietu Office, w tym Data Access Objects (DAO) oraz Open Database Connectivity (ODBC).

Więcej informacji na temat technologii ADO można znaleźć w „ABC Access 2002/XP PL" (Helion, 2002).

- **Warunkowa kompilacja kodu ułatwia tworzenie programów.** Dzięki warunkowej kompilacji kodu można stworzyć programy w różnych wersjach językowych, na przykład w wersji polskiej lub angielskiej. Warunkową kompilację kodu można również stosować do wskazania fragmentów kodu, które mają być wykonane podczas testowania programu.

- **Funkcje API zwiększają możliwości języka VBA.** Jeżeli biblioteki języka VBA nie zawierają funkcji, której potrzebujesz, możesz skorzystać z funkcji API (Application Programming Interface) przechowywanych w systemie Windows w dynamicznych bibliotekach DLL (Dynamic Linked Library).

VBA — pierwsze kroki

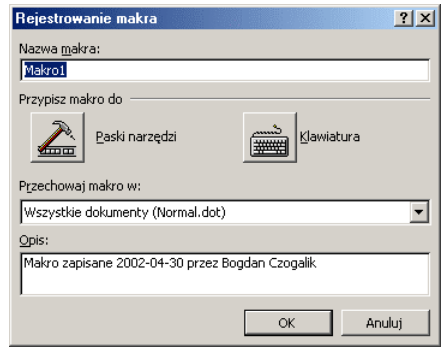
VBA jest wystarczająco bogatym środowiskiem programowania, by zaspokoić potrzeby profesjonalnych programistów. Naukę języka VBA najlepiej rozpocząć od próby zautomatyzowania często powtarzanych czynności.

Makra to programy VBA

Naukę programowania w języku VBA ułatwia *Rejestrator makr* dostępny w programach Word, PowerPoint i Excel. Makra służą do automatyzacji często powtarzanych czynności.

Rejestrację makra rozpoczyna się od wyświetlania okna dialogowego *Rejestrowanie makra*, w którym można podać nazwę dla nowego makra i wybrać inne opcje związane z jego rejestracją. Wygląd i dokładna nazwa tego okna zależy od aplikacji. Rysunek 1.4 przedstawia okno dialogowe *Rejestrowanie makra* programu Word.

Rysunek 1.4. *Okno dialogowe Rejestrowanie makra programu Word*



Aby zarejestrować makro, które ustawia skalę wyświetlania dokumentu programu Word na 200%, wykonaj następujące czynności:

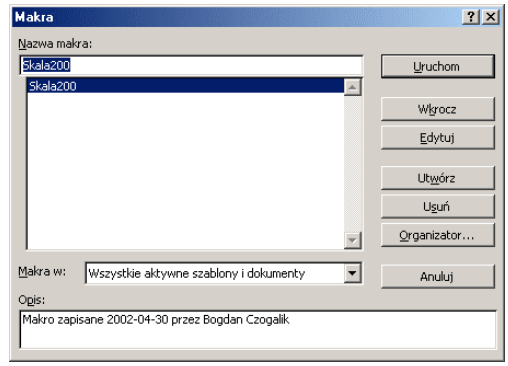
1. Z menu *Narzędzia* wybierz pozycję *Makro*, a następnie — pozycję *Zarejestruj nowe makro*. Word otworzy okno dialogowe *Rejestrowanie makra*.
2. W polu *Nazwa makra* wpisz nazwę dla nowego makra, na przykład *Skala200*.
3. Kliknij przycisk *OK*. Word wyświetli pasek narzędzi *Zatrzymaj rejestrowanie* z dwiema ikonami *Kończenie rejestracji makr* oraz *Wstrzymywanie rejestratora*.
4. Z menu *Widok* wybierz pozycję *Powiększenie*, w oknie dialogowym *Powiększenie* wybierz opcję *200%* i kliknij przycisk *OK*.

5. Kliknij ikonę *Kończenie rejestracji makr* na pasku narzędzi *Zatrzymaj rejestrowanie*. Po wyłączeniu rejestratora makr ustaw skalę wyświetlania dokumentu na wartość 100% (w tym celu w polu *Powiększenie widoku* na pasku narzędzi *Standardowy* wpisz wartość 100 i naciśnij klawisz *Enter*).

Aby uruchomić makro *Skala200*, które ustawia skalę wyświetlania dokumentu programu Word na 200%, wykonaj następujące czynności:

1. Z menu *Narzędzia* wybierz pozycję *Makro*, a następnie — pozycję *Makra*. Word otworzy okno dialogowe *Makra* (rysunek 1.5).

Rysunek 1.5. *Okno dialogowe Makra*



2. Zaznacz (kliknij) nazwę makra (w tym przypadku *Skala200*) i kliknij przycisk *Uruchom*. Skala wyświetlania dokumentu zmieni się na 200%.

Zarejestrowane makra są przechowywane w postaci kodu VBA — nie ma różnicy między kodem zarejestrowanego makra a kodem wpisanym przez programistę. Aby obejrzeć kod zarejestrowanego makra, z menu *Narzędzia* wybierz pozycję *Makro*, a następnie — pozycję *Makra*. W oknie dialogowym *Makra* zaznacz nazwę makra, którego kod chcesz obejrzeć, i kliknij przycisk *Edytuj*.

Zwróć uwagę, że makro to procedura VBA (ściśle mówiąc, makro to podprogram — procedura zadeklarowana ze słowem kluczowym *Sub* — bez parametrów). Standardowo aplikacja pakietu Office umieszcza zarejestrowane makra w module kodu związanym z bieżącym dokumentem. Moduł kodu może zawierać więcej niż jedno makro (procedurę).

Procedury i moduły omawiamy w rozdziale 4.

VBA to nie tylko makra

Makra radzą sobie znakomicie z automatycznym wykonywaniem często powtarzanych poleceń w określonym porządku. Makro nie jest jednak w stanie zareagować we właściwy sposób na zmieniające się warunki. Aby utworzyć procedurę dostosowującą swoje zachowanie do bieżących warunków, programista musi napisać kod procedury w module kodu.

Warto pisać procedury w języku VBA m.in., po to, by móc:

- **Tworzyć okna dialogowe.** VBA pozwala projektować zarówno proste okna komunikatów z przyciskami *Tak*, *Nie* i *Anuluj*, jak i skomplikowane okna dialogowe z wieloma kartami. VBA jest narzędziem do programowania wizualnego, ponieważ okna dialogowe tworzy się, rysując na formularzach różne elementy.

Aby okna dialogowe zaczęły pracować, musisz napisać kod VBA.

- **Pobierać dane z baz danych Access, dBase, FoxPro, a także z relacyjnych baz danych dostępnych za pomocą interfejsu OLE DB lub ODBC.** Aby przeglądać dane pochodzące z wymienionych źródeł, należy korzystać z programu Microsoft Query. Aby natomiast manipulować nimi, należy napisać procedury w języku VBA.
- **Dodać do procedury kod uruchamiany, ilekroć w procedurze wystąpi błąd.** Aby nie doszło do zawieszenia makra lub procedury napisanej przez programistę w przypadku wystąpienia błędu, można dodać kod odpowiedzialny za obsługę takich sytuacji. Kod obsługujący błędy jest szczególnie ważny, jeżeli programy VBA są używane nie tylko przez ich autora, ale i przez innych użytkowników. Autor nie może kontrolować warunków, w jakich inni użytkownicy uruchamiają jego programy. VBA zawiera kilka instrukcji służących do tworzenia kodu obsługującego błędy.
- **Utworzyć system pomocy.** Można stworzyć pomoc kontekstową i wyświetlać komunikaty, kiedy użytkownik popełni błąd. Można nawet programować Asystenta pakietu Office, tworząc zestaw wskazówek pomagających użytkownikowi wykonać złożone zadanie.
- **Ochronić program przed nieautoryzowanym modyfikowaniem kodu lub kopiowaniem.** Kod VBA można zabezpieczyć hasłem, aby nieuprawnione osoby nie miały do niego dostępu.

VBA — narzędzie do programowania obiektowego

Chociaż Visual Basic dla Aplikacji nie spełnia wszystkich warunków, jakie są stawiane językowi programowania obiektowego, obiekty odrywają fundamentalną rolę w języku VBA. Ten podrozdział jest wprowadzeniem do programowania obiektowego.

Obiekty VBA są omówione szczegółowo w rozdziale 4.

Obiekty języka VBA

Obiekty są elementami tworzącymi aplikację macierzystą (aplikacją macierzystą jest na przykład Word, Excel i PowerPoint) oraz dokument utworzony za pomocą tej aplikacji (na przykład dokument programu Word, skoroszyt programu Excel lub prezentacja programu PowerPoint). W programie Excel obiektem jest nie tylko skoroszyt, ale i arkusz, komórka arkusza, zakres komórek oraz wykres.

Obiekty mogą być również bardziej abstrakcyjne. Obiekt *WidokNiestandardowy* na przykład może reprezentować niestandardowy widok skoroszytu programu Excel. Widok niestandardowy nie jest czymś, co można zobaczyć na ekranie, ale zbiorem ustawień definiujących wygląd skoroszytu oraz jego opcje drukowania. We wszystkich aplikacjach, do których dołączono VBA, obiekt *Collection* reprezentuje kolekcję obiektów traktowanych jako jedna całość.

Modele obiektowe

Każdy obiekt należy do hierarchii obiektów zdefiniowanej przez aplikację macierzystą i przez VBA. Obiekt umieszczony na samym szczycie hierarchii posiada właściwości, metody i zdarzenia. Jest on również kontenerem innych obiektów (tych, które znajdują się niżej w hierarchii). Obiekty umieszczone poniżej obiektu bazowego również posiadają właściwości, metody i zdarzenia. *Model obiektowy* aplikacji, do której dołączono VBA, definiuje hierarchię obiektów, ich właściwości, metody i zdarzenia oraz relacje między obiektami.

VBA posiada własny model obiektowy. Zatem, pracując na przykład w aplikacji Word, możesz korzystać z modelu obiektowego *Word* oraz modelu obiektowego *VBA*. Część modelu obiektowego aplikacji Word przedstawia rysunek 1.6.

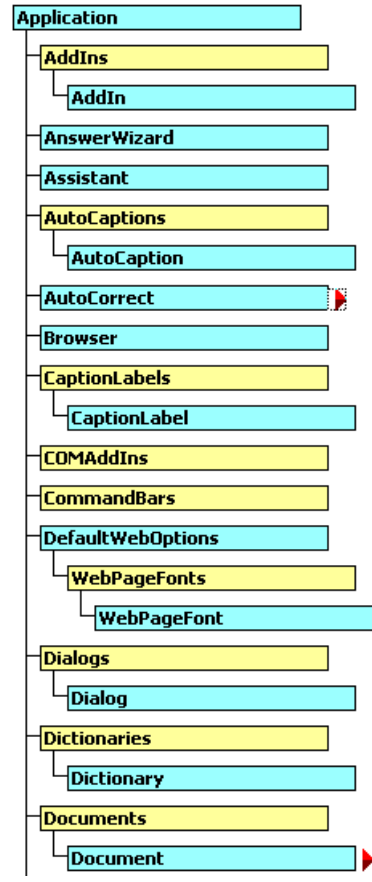
Obiektem bazowym modelu obiektowego aplikacji Word jest obiekt *Application*. Poniżej znajdują się m.in. obiekty *Documents*, *Paragraphs*, *Characters*, *Bookmarks* oraz *Dialogs*.

Technologia ActiveX a VBA

Termin ActiveX, wprowadzony przez Microsoft, służy do określania zbioru technologii stosowanych w produkcji oraz automatyzacji aplikacji. Jest to standard stosowany przez twórców komponentów ActiveX (formantów) oraz programistów wykorzystujących te komponenty do tworzenia aplikacji.

Formanty ActiveX

Formanty ActiveX to elementy okien dialogowych i pasków narzędzi, za pomocą których użytkownik kieruje aplikacją. Przykładem formantu jest przycisk polecenia, który użytkownik może kliknąć, grupa przycisków opcji, z której użytkownik może wybrać jedną opcję, oraz pole tekstowe, gdzie użytkownik może wpisać i edytować tekst.



Rysunek 1.6. *Część modelu obiektowego aplikacji Word*

Warto zwrócić uwagę, że można korzystać nie tylko z formantów dostarczanych z VBA, ale również z formantów sprzedawanych przez niezależnych producentów. Formant jest komponentem, który można dodać do formularza i dostosować do potrzeb konkretnego projektu. Formanty ActiveX można zastosować w aplikacjach pisanych za pomocą języka Visual Basic, C++, Java i oczywiście VBA. Jeżeli żaden z formantów dostarczanych z VBA nie odpowiada potrzebom danego projektu, możesz nabyć odpowiedni formant od jednej z wielu firm specjalizujących się w produkcji formantów ActiveX. Formant ActiveX można również utworzyć za pomocą pakietu programistycznego Visual Basic 6.0 (wersja Professional lub Enterprise). Rysunek 1.7 przedstawia okno dialogowe z formantami ActiveX.

Niektóre formanty są widoczne podczas działania aplikacji, inne — nie. Mogą one wykonywać wyspecjalizowanie zadania, na przykład dokonywać obliczeń finansowych.

Formanty ActiveX można również wykorzystać do tworzenia stron internetowych wyświetlanych w przeglądarkach internetowych. Więcej informacji na ten temat zawiera rozdział 6.

Rysunek 1.7. *Okno dialogowe z formantami ActiveX*

Posiadacze przeglądarek internetowych obsługujących standard ActiveX mogą oglądać i używać witryny internetowe zawierające formanty ActiveX. W ten sposób technologia ActiveX staje się dostępna dla użytkowników korzystających z różnych systemów operacyjnych.

Automatyzacja ActiveX

Pisząc program w języku VBA, programista może korzystać nie tylko z obiektów aplikacji macierzystej, ale także z każdej aplikacji lub komponentu stosujących standard Component Object Model (COM). COM określa, w jaki sposób należy definiować i udostępniać obiekty aplikacji, aby mogły z nich korzystać inne aplikacje. Termin *automatyzacja* oznacza możliwość aktywowania i kontrolowania obiektów aplikacji COM.

Automatyzacja oraz standard COM otwierają olbrzymie możliwości przed programistami tworzącymi aplikacje w języku VBA. Można na przykład stworzyć aplikację przetwarzającą informacje pochodzące z dokumentów aplikacji Word, skoroszytów aplikacji Excel, baz danych aplikacji Access i Outlook, prezentacji aplikacji PowerPoint, diagramów aplikacji Visio, rysunków technicznych aplikacji AutoCAD itd. Informacje po przetworzeniu można wyświetlić w oknach dialogowych. Technologia DCOM (Distributed Component Object Model) pozwala rozpowszechniać komponenty ActiveX w sieciach internetowych lub intranetowych.

Serwery i kontrolery automatyzacji

Serwer automatyzacji to aplikacja, która udostępnia obiekty innym aplikacjom. *Kontroler automatyzacji* to aplikacja, która kontroluje obiekty innej aplikacji. Każda aplikacja pakietu Office XP może być zarówno serwerem, jak i kontrolerem automatyzacji.

Więcej informacji o VBA

Aby dowiedzieć się więcej o programowaniu w języku Visual Basic, możesz nabyć jedną z książek poświęconych tej tematyce wydanych przez Wydawnictwo Helion.

Informacje na temat VBA można również znaleźć w Internecie. Oto adresy dwóch stron producenta technologii VBA:

- *msdn.microsoft.com/vba*,
- *msdn.microsoft.com/vbasic*.

Informacji warto też szukać na stronach internetowych firmy Microsoft poświęconych konkretnym aplikacjom pakietu Office oraz na stronach Microsoft Knowledge Base.

Oto kilka innych stron internetowych wartych uwagi:

- *http://download.com.com/2001-2206-0.html?tag=dir*
- *www.geocities.com*,
- *www.cgvb.com*,
- *www.mvps.org/vbnet*,
- *www.vbip.com*,
- *http://searchvb.techtarget.com*,
- *www.codehound.com/vb*.

Oto lista wybranych czasopism poświęconych językowi VBA:

- Microsoft Developer's Network (MSDN)

  http://msdn.microsoft.com,

- Elements K Journals

  www.elementkjournals.com,

- Microsoft Office & Visual Basic for Applications Developer

  Informant Communications Group, Inc.
  10519 E. Stockton Blvd.
  Suite 100
  Elk Grove, CA 95624-9703
  www.informant.com,

- Visual Basic Programmer's Journal,

  Fawcette Technical Publications, Inc
  913 Emerson Ave.
  Palo Alto
  CA 94301
  www.windx.com.