



Zawiera CD

**Sergiusz Flanczewski**

# Access 2016 **PL**

W BIURZE I NIE TYLKO

**Helion** 

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Opieka redakcyjna: Ewelina Burska

Projekt okładki: Studio Gravite/Olsztyn

Obarek, Pokoński, Pazdrijowski, Zaprucki

Materiały graficzne na okładce zostały wykorzystane za zgodą Shutterstock.

Wydawnictwo HELION

ul. Kościuszki 1c, 44-100 GLIWICE

tel. 32 231 22 19, 32 230 98 63

e-mail: [helion@helion.pl](mailto:helion@helion.pl)

WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/a16biu>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-283-1737-6

Copyright © Helion 2016

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

# Spis treści

Wstęp .....	7
<b>Rozdział 1. Podstawowe informacje o obsłudze systemu zarządzania bazami danych Access 2016 .....</b>	<b>9</b>
Uruchomienie i zamykanie programu .....	11
Baza danych .....	13
Otwarcie istniejącej bazy danych .....	13
Otwarcie istniejącej bazy danych ze zmodyfikowanymi parametrami uruchomieniowymi .....	16
Tworzenie nowej bazy danych .....	20
Narzędzia bazy danych .....	21
Bazy danych a wersje programu Access .....	25
Otwieranie bazy danych z wcześniejszej wersji programu Access .....	26
<b>Rozdział 2. Tabele .....</b>	<b>27</b>
Tworzenie tabeli i jej struktura .....	27
Typy danych pola .....	28
Tworzenie tabel .....	28
Tworzenie tabeli przez wprowadzanie danych .....	30
Tworzenie tabeli w widoku projektu .....	35
Tworzenie tabeli — polecenie Importuj .....	43
Tworzenie tabeli — polecenie Połącz tabele .....	49
Klucze podstawowe tabeli .....	52
Obsługiwanie tabeli .....	55
Nawigowanie w tabeli .....	56
Zaznaczanie pól i rekordów .....	61
Prezentacja danych w tabeli .....	62
Aktualizacja danych w tabeli .....	74
Modyfikacja struktury tabeli .....	84
<b>Rozdział 3. Relacje i kwerendy .....</b>	<b>91</b>
Relacje — informacje podstawowe .....	91
Tworzenie, edytowanie i usuwanie relacji .....	94
Konstruowanie relacji „jeden do wielu” .....	94
Aktualizacja tabel pozostających w relacji „jeden do wielu” .....	99
Konstruowanie relacji „wiele do wielu” .....	102
Aktualizacja tabel pozostających w relacji „wiele do wielu” .....	113

Kwerendy — informacje podstawowe .....	116
Tworzenie kwerendy w widoku projektu .....	116
Tworzenie kwerendy za pomocą kreatora .....	122
Kwerenda wybierająca .....	125
Kwerenda parametryczna .....	135
Kwerenda krzyżowa .....	139
Kwerendy funkcjonalne .....	148
Kwerenda tworząca tabelę .....	149
Kwerenda aktualizująca .....	153
Kwerenda usuwająca .....	160
Kwerenda dołączająca .....	167
<b>Rozdział 4. Formularze, formanty i raporty .....</b>	<b>173</b>
Formularz — informacje podstawowe .....	173
Tworzenie formularza związanego .....	173
Formularz z wieloma elementami (tabelaryczny) .....	176
Formularz jako arkusz danych .....	178
Formularz dzielony .....	181
Tworzenie formularza za pomocą kreatora .....	183
Tworzenie formularza w widoku projektu .....	188
Zmiana właściwości formularza .....	194
Formularz — umieszczanie obrazu (grafiki) .....	196
Formularz jako okno dialogowe .....	199
Tworzenie podformularzy .....	202
Formanty — informacje podstawowe .....	209
Umieszczanie formantu na formularzu .....	210
Grupa narzędzi Formanty .....	213
Raporty — informacje podstawowe .....	215
Tworzenie raportu za pomocą polecenia Raport .....	216
Tworzenie raportu za pomocą kreatora .....	219
<b>Rozdział 5. Makra i edytor języka Visual Basic .....</b>	<b>225</b>
Makra — informacje podstawowe .....	226
Tworzenie makra .....	226
Uruchamianie makra .....	230
Edytor języka Visual Basic .....	236
Obsługa edytora VBA .....	236
<b>Rozdział 6. Jednoręki bandyta, czyli grafika i losowość w jednym .....</b>	<b>251</b>
Zadanie projektowe .....	251
Założenia szczegółowe do projektowanej bazy danych .....	251
Konstruowanie tabel bazy danych .....	256
Konstruowanie tabeli Tab_fot .....	257
Konstruowanie tabeli Wynik .....	264
Konstruowanie formularzy .....	265
Formularz Tabela wygranych .....	265
Formularz START .....	272
Formularz Logo .....	290
Makra .....	298
Makropolecenie Makro1 .....	300
Makropolecenie Makro2 .....	300
Makropolecenie Makro3 .....	301
Makropolecenie Makro4 .....	302

Obsługa zdarzeń formularzy i formantów .....	302
Formularz START .....	304
Formularz Logo .....	314
<b>Rozdział 7. Kalendarze, czyli funkcje i kwerendy w jednym .....</b>	<b>315</b>
Zadanie projektowe .....	315
Założenia szczegółowe do projektowanej bazy danych .....	315
Konstruowanie tabel oraz kwerend bazy danych .....	319
Konstruowanie tabeli Rok .....	320
Konstruowanie tabeli Święta .....	321
Konstruowanie kwerendy aktualizującej ROK Kwerenda .....	323
Konstruowanie kwerendy aktualizującej Święta Kwerenda .....	327
Konstruowanie kwerendy aktualizującej Data_0_Kwerenda .....	328
Konstruowanie kwerendy aktualizującej Aktualizacja .....	329
Konstruowanie formularza .....	332
Konstruowanie formularza Kalendarz_rok .....	332
Makro .....	347
Makropolecenie Makro1 .....	349
Obsługa zdarzeń formularza i formantów .....	351
Formularz Kalendarz_rok .....	351
Filtrowanie formularza Kalendarz_rok .....	364
<b>Rozdział 8. Wykresy, czyli liczby w słupkach .....</b>	<b>371</b>
Zadanie projektowe .....	371
Założenia szczegółowe do projektowanej bazy danych .....	371
Konstruowanie tabel i kwerendy bazy danych .....	375
Konstruowanie tabeli Tab_osoby .....	376
Konstruowanie tabeli Tab_transakcje .....	377
Konstruowanie tabeli Parametry .....	379
Konstruowanie relacji pomiędzy tabelami Tab_osoby i Tab_transakcje .....	380
Konstruowanie kwerendy Suma_Kwerenda .....	384
Konstruowanie formularzy .....	388
Formularz START .....	388
Formularz Parametry .....	412
Formularz Wykres .....	420
Modyfikacja wykresu .....	428
Konstruowanie raportu .....	436
Konstruowanie makropoleceń .....	446
Makropolecenie Makro1 .....	448
Makropolecenie Makro2 .....	448
Makropolecenie Makro3 .....	449
Obsługa zdarzeń formularzy i formantów .....	449
<b>Skorowidz .....</b>	<b>453</b>



## Rozdział 5.

# Makra i edytor języka Visual Basic

W programie Microsoft Access obsługę zdarzeń zachodzących w obiektach (formularzach, formantach), wywołanych zachowaniem użytkownika (jak również reakcją na zdarzenia w innych obiektach), można zapewnić, używając makra lub korzystając z interfejsu użytkownika. Jednakże w wielu bazach danych przyjdzie nam się zmierzyć z „czystym programowaniem”, czyli użyciem procedur (składni) języka Visual Basic for Applications.

Dla osób niezajmujących się na co dzień programowaniem zastosowanie makra do tworzenia powiązań istniejących obiektów bazy danych jest łatwiejsze, ponieważ wymagana jest tylko niewielka wiedza na temat składni poleceń. Prostota ta pociąga jednak za sobą pewne ograniczenia „programistyczne”, do których trzeba zaliczyć:

- ◆ brak możliwości modyfikowania argumentu (użycia zmiennych jako argumentów) makra w czasie jego wykonywania,
- ◆ makra są obiektami zewnętrznymi w stosunku do formularzy i raportów (które z nich korzystają), dlatego też baza danych zawierająca dużą liczbę makr reagujących na zdarzenia występujące w formularzach i raportach staje się kłopotliwa w obsłudze,
- ◆ makra operują na całym zestawie rekordów jednocześnie.

Używając języka Visual Basic, można manipulować wszystkimi obiektami bazy łącznie z samą bazą, jak również wykonywać działania na poziomie systemu. Dostępne w języku VBA funkcje umożliwiają przeprowadzanie obliczeń bez konieczności tworzenia skomplikowanych wyrażeń. Można również tworzyć własne funkcje do wykonywania obliczeń przekraczających możliwości pojedynczego wyrażenia lub zastępujących skomplikowane wyrażenia. Ponadto utworzonych przez siebie funkcji można używać w stosunku do wielu obiektów (formularzy, formantów). Inne zalety zastosowania instrukcji i poleceń języka Visual Basic to:

- ◆ możliwość przetwarzania danych w pojedynczym rekordzie,
- ◆ możliwość generowania komunikatów o ewentualnych błędach,
- ◆ możliwość opracowania (zaprogramowania) procedur obsługujących błędy programu lub użytkownika.

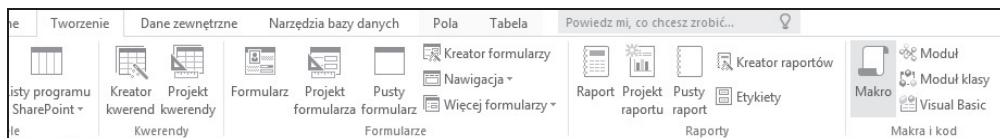
## Makra — informacje podstawowe

Instrukcja Accessa określa makro jako *akcję lub zestaw akcji, które można tworzyć w celu zautomatyzowania często wykonywanych zadań*. Mówiąc prościej, jeżeli chcemy na przykład otworzyć obiekt (formularz, raport, kwerendę itp.) lub przejść w tabeli do następnego rekordu w sposób „programowy”, to znaczy bez wykonywania kolejnych poleceń z menu i (lub) okien dialogowych, możemy wybrać odpowiednią akcję, czyli polecenie, i przypisać ją do procedury obsługi zdarzenia wybranego formantu (np. do zdarzenia, jakim jest kliknięcie formantu typu przycisk).

### Tworzenie makra

Aby utworzyć makro, które zostanie następnie związane z jakimś formantem lub formularzem, należy postępować według poniższych kroków:

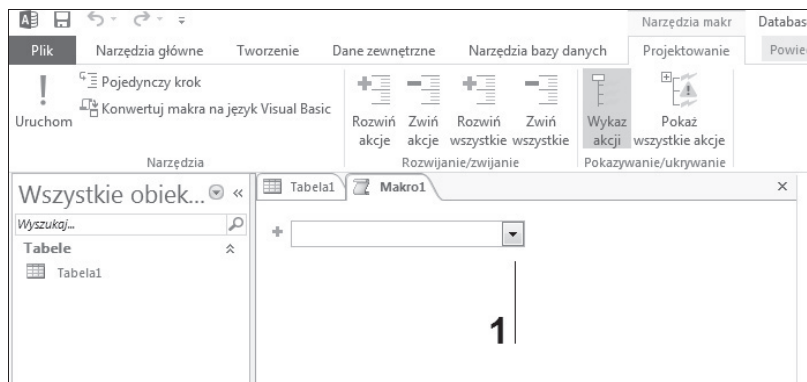
1. Z karty *Tworzenie* wybrać polecenie *Makro* (rysunek 5.1).



Rysunek 5.1. Rozpoczęcie tworzenia makropolecenia

2. Działanie z punktu 1. spowoduje aktywację okna projektu makropolecenia (rysunek 5.2).

Rysunek 5.2.  
Okno projektu makropolecenia

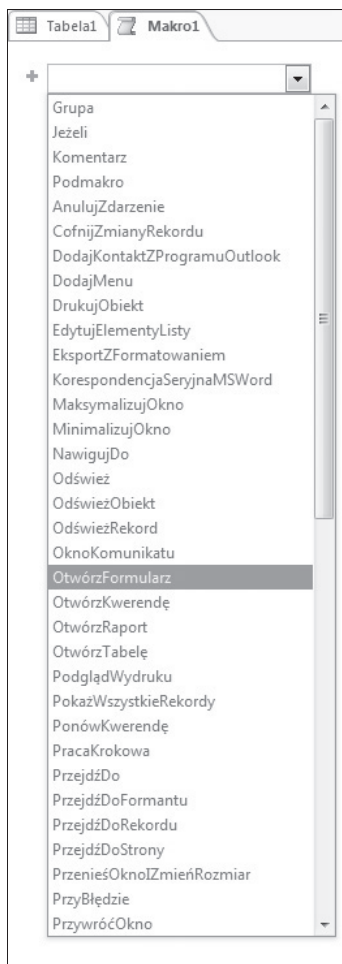




3. Za pomocą przycisku oznaczonego symbolem 1 na rysunku 5.2 rozwinąć listę z dostępnymi akcjami, a następnie dokonać wyboru żądanej akcji (rysunek 5.3).

**Rysunek 5.3.**

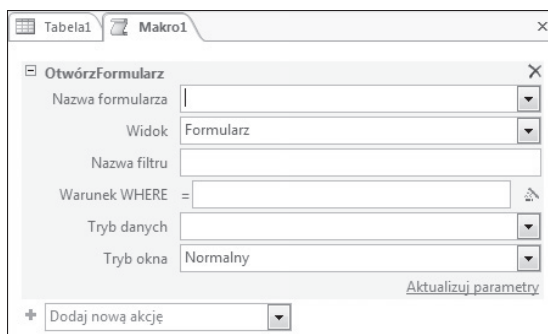
*Lista rozwijana akcji dostępnych dla makropolecenia*



4. Po dokonaniu wyboru akcji w dolnej części okna projektu makra zostaną wyświetlone (jeżeli występują) *Argumenty akcji* (rysunek 5.4).

**Rysunek 5.4.**

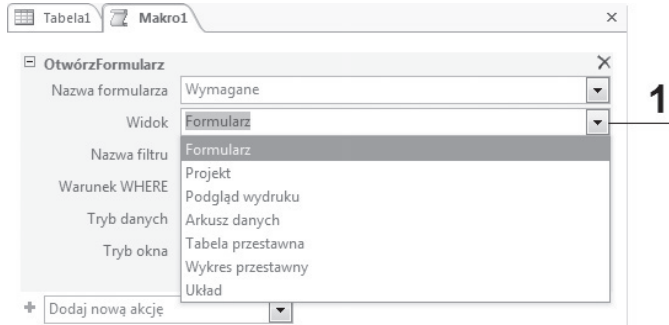
*Sekcja Argumenty akcji dla akcji OtwórzFormularz*



5. Określenie argumentu akcji możemy wykonać przez:

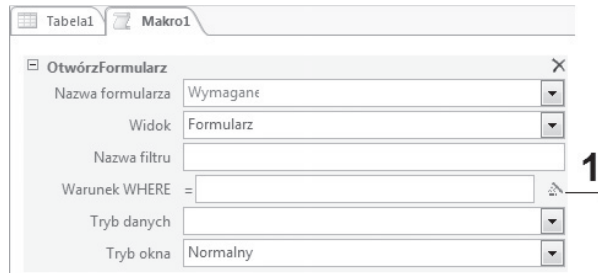
- ◆ kliknięcie przycisku (jeżeli jest dostępny) ze strzałką (rysunek 5.5, oznaczenie 1), a następnie dokonanie wyboru z listy rozwijanej wyświetlającej dostępne opcje argumentu akcji,

**Rysunek 5.5.**  
*Lista rozwijana dostępnych opcji argumentu akcji*

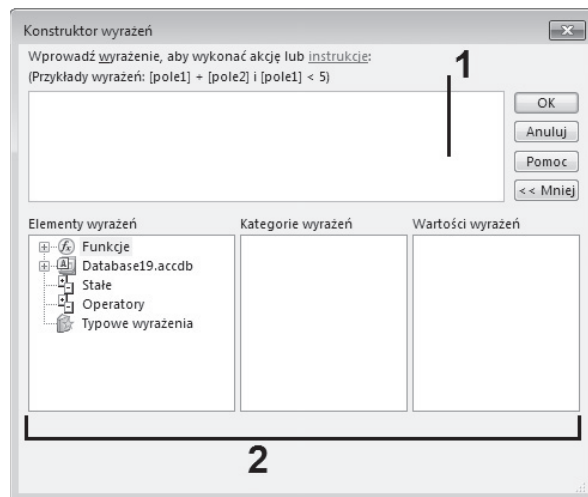


- ◆ kliknięcie przycisku (jeżeli jest dostępny) z wielokropkiem (rysunek 5.6, oznaczenie 1), powodując tym samym uruchomienie okna dialogowego *Konstruktor wyrażeń* (rysunek 5.7),

**Rysunek 5.6.**  
*Przycisk uruchomienia okna Konstruktor wyrażeń*



**Rysunek 5.7.**  
*Okno dialogowe Konstruktor wyrażeń*



- ♦ ręczne wpisanie w polu argumentużądanego wyrażenia.



Okno *Konstruktor wyrażeń* składa się z dwóch podstawowych sekcji:

1 — pole wyrażenia — służy do utworzenia zapisu wyrażenia za pomocą wklejenia wybranych elementów z sekcji 2. W polu wyrażenia można również bezpośrednio wpisywać poszczególne części wyrażenia.

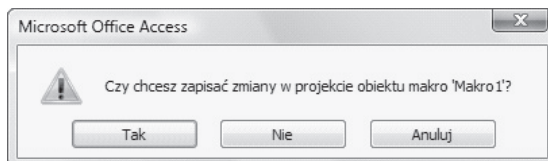
2 — elementy wyrażenia — to trzy pola w dolnej części okna *Konstruktor wyrażeń*, służące do określenia (wyboru):

- ♦ Lewe pole zawiera foldery z listami obiektów bazy danych, takich jak tabele, kwerendy, formularze i raporty, oraz listami funkcji wbudowanych i zdefiniowanych przez użytkownika, stałych, operatorów i typowych wyrażeń.
- ♦ Środkowe pole zawiera listę elementów lub kategorii elementów określonych dla folderu zaznaczonego w lewym polu.
- ♦ W prawym polu wyświetlana jest lista wartości (jeśli takie istnieją) dla elementów zaznaczonych w polu lewym i środkowym.

6. Aby zakończyć projektowanie makropolecenia, należy kliknąć ikonę oznaczoną symbolem *X*, znajdującą się w prawym górnym rogu okna projektu makra, co spowoduje pojawienie się okna z komunikatem dotyczącym potwierdzenia zapisu zmian w projekcie makropolecenia (rysunek 5.8).

#### Rysunek 5.8.

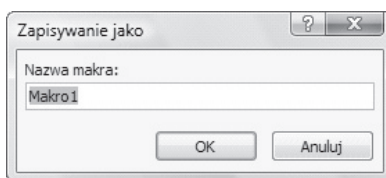
*Okno z komunikatem dotyczącym potwierdzenia zapisu zmian w projekcie makropolecenia*



7. Kliknięcie przycisku *Tak* spowoduje pojawienie się okna dialogowego *Zapisywanie jako*, w którym należy wpisać nazwę makra lub pozostawić nazwę domyślną, po czym kliknąć przycisk *OK* (rysunek 5.9).

#### Rysunek 5.9.

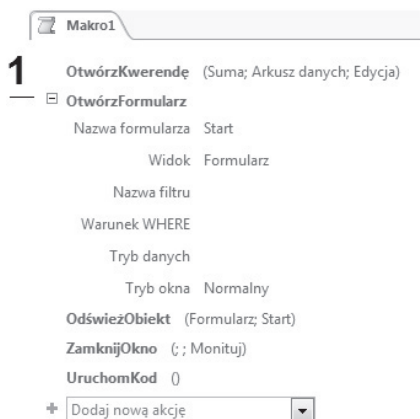
*Okno dialogowe Zapisywanie jako*



Aby utworzyć zestaw akcji wchodzących w skład makra, które zostanie następnie związane z jakimś formantem lub formularzem, należy w kolejnych liniach okna projektu makropolecenia umieścić polecenia wykonaniażądaney akcji (rysunek 5.10, oznaczenie 1).

**Rysunek 5.10.**

Wygląd zestawu akcji wchodzących w skład przykładowego makra o nazwie *Makro\_MC*



## Uruchamianie makra

Skonstruowane makro (bez względu na jego strukturę: czy pojedyncza akcja, czy zestaw akcji) możemy uruchomić w jeden z podanych poniżej sposobów:

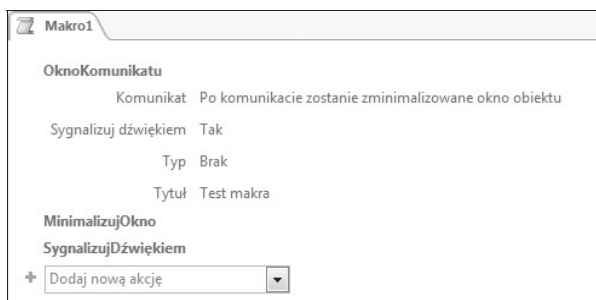
- ◆ bezpośrednio w okienku nawigacji,
- ◆ w odpowiedzi na zdarzenie związane z obiektem bazy danych (np. kliknięcie przycisku znajdującego się na formularzu),
- ◆ z karty *Narzędzia bazy danych* w grupie *Makro* — polecenie *Uruchom makro*,
- ◆ z procedury języka VBA.

Określone powyżej sposoby uruchamiania makra prześledzimy na specjalnie do tego skonstruowanym zestawie akcji (rysunek 5.11) o nazwie *Makro1*, w którego skład wchodzi kolejno:

1. Akcja *OknoKomunikatu* — wyświetla komunikat z treścią określoną przez użytkownika.
2. Akcja *Minimalizuj* — minimalizuje aktywne okno.
3. Akcja *SygnalizujDźwiękiem* — emituje sygnał dźwiękowy za pomocą głośnika komputera.

**Rysunek 5.11.**

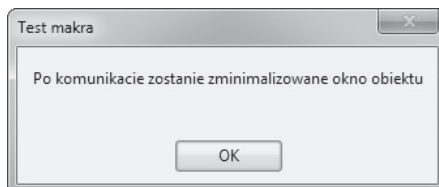
Okno projektu przykładowego makra



Tak skonstruowane makro spowoduje następujące działanie:

1. Bezpośrednio po uruchomieniu w oknie dialogowym o nazwie *Test makra* (argument *Tytuł* w akcji *Okno komunikatu* został uzupełniony tekstem *Test makra*) wyświetli się komunikat o treści: *Po komunikacie zostanie zminimalizowane okno obiektu* (rysunek 5.12) oraz usłyszymy sygnał dźwiękowy (argument *Sygnalizuj dźwiękiem* akcji *Okno komunikatu* został ustawiony na wartość *Tak*). Działanie makra zostanie wstrzymane do chwili kliknięcia przez użytkownika przycisku *OK* znajdującego się w oknie z komunikatem.

**Rysunek 5.12.**  
*Okno dialogowe  
Test makra*



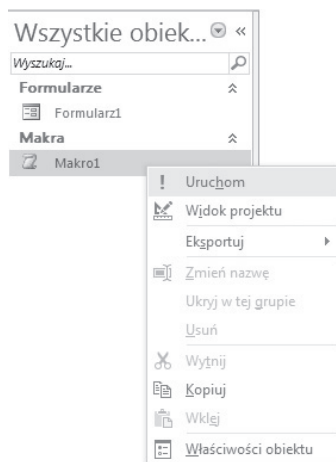
2. Po kliknięciu przycisku *OK*:
  - ♦ zniknie okno z komunikatem,
  - ♦ okno aktywne w chwili działania makra (np. okno formularza) zostanie zmniejszone do postaci małego paska tytułu.
3. Po zminimalizowaniu okna usłyszymy sygnał dźwiękowy (efekt działania akcji *SygnalizujDźwiękiem*).

## Uruchamianie makra w okienku nawigacji

Aby uruchomić makro w okienku nawigacji, należy wykonać jedną z czynności:

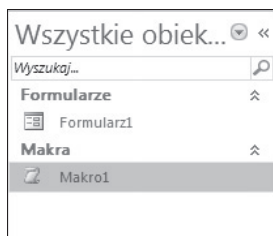
- ♦ kliknąć prawym przyciskiem myszy selektor formularza, po czym z menu podręcznego wybrać polecenie *Uruchom* (rysunek 5.13),

**Rysunek 5.13.**  
*Menu podręczne  
obiektu Makro*



- ◆ w okienku nawigacji kliknąć dwukrotnie nazwę żądanego makra (rysunek 5.14).

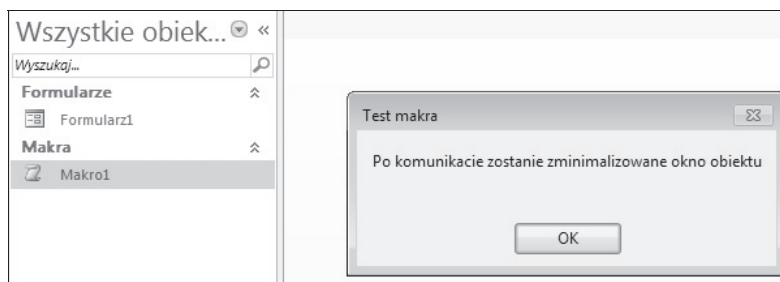
**Rysunek 5.14.**  
Uruchomienie makra  
przez dwukrotne  
kliknięcie



Wykonanie jednej z opisanych powyżej czynności spowoduje w przypadku uruchomienia przykładowego makropolecenia o nazwie *Makro1*:

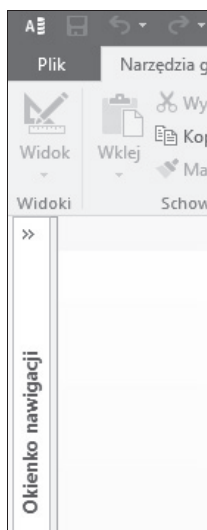
- ◆ wyświetlenie komunikatu w postaci pokazanej na rysunku 5.15 wraz z emisją sygnału dźwiękowego,

**Rysunek 5.15.**  
Okno z komunikatem  
określonym w akcji  
OknoKomunikatu



- ◆ po kliknięciu przycisku *OK* — przyjęcie przez okienko nawigacji postaci pokazanej na rysunku 5.16,

**Rysunek 5.16.**  
Wygląd okienka  
nawigacji  
po wykonaniu akcji  
Minimalizuj



- ◆ wyemitowanie sygnału dźwiękowego przez głośnik komputera.

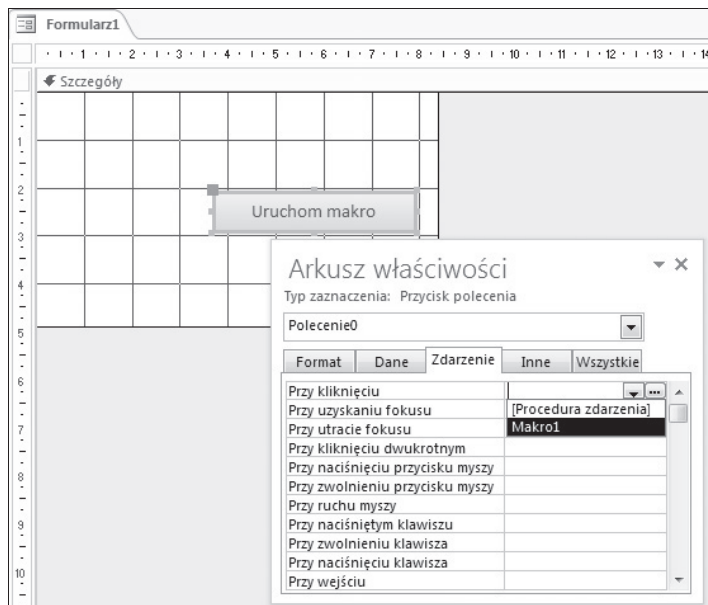
## Uruchamianie makra w odpowiedzi na zdarzenie związane z obiektem bazy danych

Aby makro było uruchamiane w odpowiedzi na zdarzenia zachodzące w obiekcie, należy je przypisać do określonego typu zdarzenia obiektu. Na przykład na rysunku 5.17 pokazano sposób przypisania makropolecenia do procedury obsługi zdarzenia *Przy kliknięciu* obiektu typu przycisk polecenia. Przypisanie takie należy wykonać w następujący sposób:

1. W widoku *Projekt* wybrać obiekt, do którego ma zostać przypisane makropolecenie.

**Rysunek 5.17.**

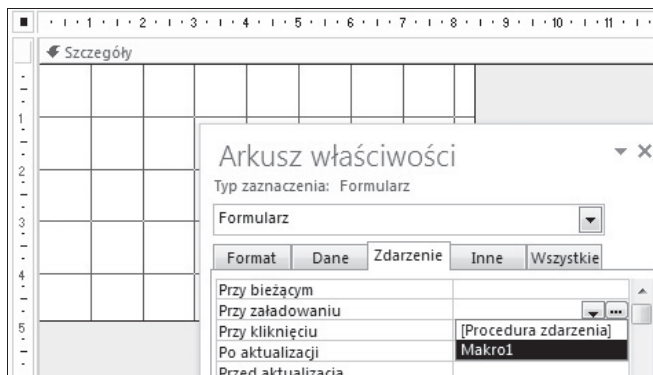
*Przypisanie makropolecenia do zdarzenia Przy kliknięciu obiektu przycisk polecenia*



2. Otworzyć okno *Arkusz właściwości* obiektu, na przykład okno właściwości obiektu typu przycisk polecenia (rysunek 5.17) lub okno właściwości formularza (rysunek 5.18).

**Rysunek 5.18.**

*Przypisanie makropolecenia do zdarzenia Przy załadowaniu obiektu Formularz*

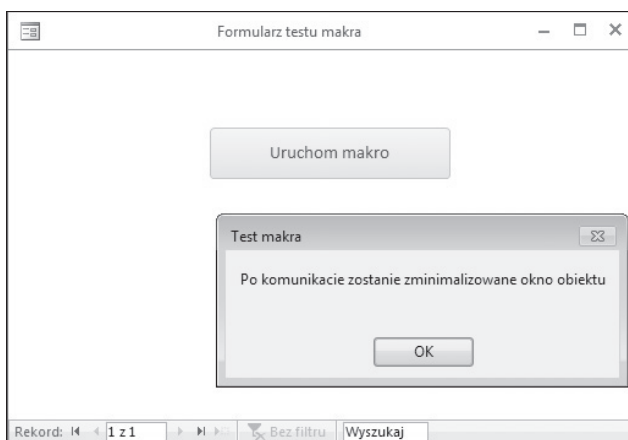


3. Na karcie *Zdarzenie* kliknąć w polu zdarzenia, które ma być obsłużone przez akcję makropolecenia.
4. Rozwinąć za pomocą przycisku ze strzałką listę dostępnych makropoleceń i wybrać żądane makro.
5. Zamknąć okno właściwości obiektu i zapisać dokonane zmiany.
6. Uruchomić „oprogramowany” w taki sposób formularz.

Gdy nasze przykładowe makro, składające się z trzech akcji: *OknoKomunikatu*, *Minimalizuj* oraz *SygnalizujDźwiękiem*, przypiszemy do obsługi zdarzenia *Przy kliknięciu* obiektu typu przycisk polecenia (znajdującego się na formularzu) — „praca” makra rozpocznie się dopiero wtedy, gdy przycisk polecenia zostanie przez nas kliknięty myszą. Działanie to spowoduje aktywację okna z komunikatem (rysunek 5.19).

### Rysunek 5.19.

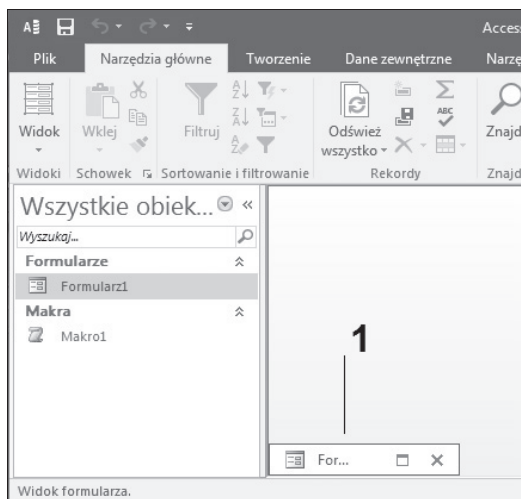
*Wygląd uruchomionego okna formularza oraz okna z komunikatem (po kliknięciu przycisku polecenia Uruchom makro)*



Efektom kliknięcia przycisku *OK* (znajdującego się w oknie z komunikatem) będzie przybranie przez okno formularza postaci oznaczonej na rysunku 5.20 symbolem 1.

### Rysunek 5.20.

*Wygląd formularza po zakończeniu pracy makropolecenia*





Gdy omawiane makropolecenie przypiszemy do obsługi zdarzenia formularza *Przy załadowaniu*, proces przebiegać będzie następująco:

- ♦ bezpośrednio po uruchomieniu formularza (np. przez dwukrotne kliknięcie nazwy formularza w okienku nawigacji) zostanie wyświetlone okno z komunikatem,
- ♦ gdy w oknie z komunikatem klikniemy przycisk *OK*, na chwilę zostanie wyświetlone okno formularza, po czym samoczynnie zmniejszy się ono do postaci paska (oznaczenie 1 na rysunku 5.20).



Uwaga

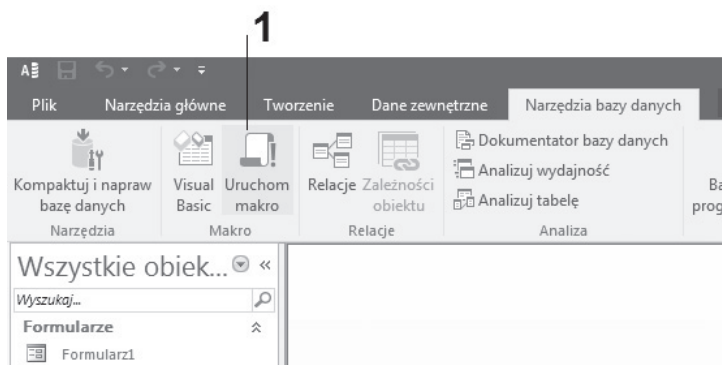
Uruchamianie makropolecenia z procedury języka VBA zostanie opisane w dalszej części niniejszego rozdziału.

## Uruchamianie makra z karty Narzędzia bazy danych

Aby uruchomić makro z karty *Narzędzia bazy danych*, należy:

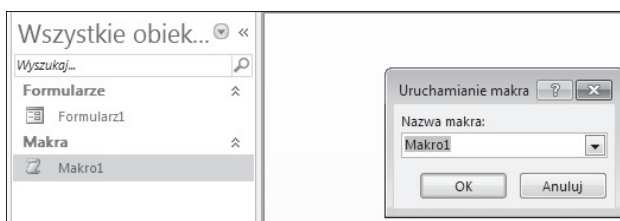
- ♦ otworzyć żądaną bazę danych,
- ♦ z karty *Narzędzia* wybrać polecenie *Uruchom makro* (rysunek 5.21, oznaczenie 1),

**Rysunek 5.21.**  
Karta *Narzędzia bazy danych* — polecenie *Uruchom makro*



- ♦ po aktywacji okna dialogowego *Uruchamianie makra* wybrać z listy rozwijanej *Nazwa makra* (rysunek 5.22) żądane makro, po czym kliknąć przycisk *OK*.

**Rysunek 5.22.**  
Okno dialogowe *Uruchamianie makra*



Uwaga

Jeżeli makro (przykładowe o nazwie *Makro1*) uruchomimy z menu, zmniejszanie okna do postaci paska dotyczyć będzie okna dialogowego aktywnego obiektu: okienka nawigacji, tabeli lub formularza. Na rysunku 5.22 aktywnym obiektem jest okienko nawigacji.

# Edytor języka Visual Basic

Edytor języka Visual Basic for Applications (języka programowania wysokiego poziomu, będącego wizualną wersją języka Basic) wchodzący w skład pakietu MS Office, na przykład w skład aplikacji Word, Excel, Access, PowerPoint, Outlook, jest w przypadku każdej z nich prawie identyczny (może poza Accessem), to znaczy ma taki sam interfejs użytkownika, identyczną składnię oraz takie same słowa kluczowe i instrukcje. Różnica polega tylko na różnych obiektach i ich metodach oraz właściwościach obiektów związanych z aplikacją macierzystą. Ponieważ wcześniej była już mowa o tym, co to jest obiekt oraz czym są jego właściwości, w tym miejscu należy jeszcze wyjaśnić pojęcie metody. Metoda jest procedurą podobną do instrukcji lub funkcji, pozwalającą na tworzenie nowych obiektów lub operującą na istniejących już obiektach (np. manipulowanie ich właściwościami).

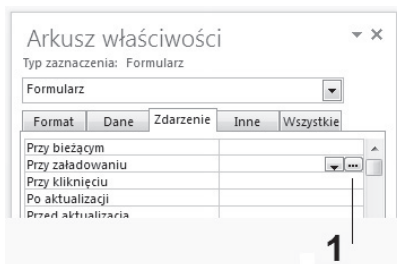
## Obsługa edytora VBA

Edytor Visual Basic może zostać uruchomiony przez wykonanie jednej z czynności:

- ◆ naciśnięcie klawiszy *Alt+F11*,
- ◆ kliknięcie polecenia *Visual Basic* na karcie *Narzędzia bazy danych*,
- ◆ kliknięcie w oknie właściwości formularza (formantu) przycisku z wielokropkiem (rysunek 5.23, oznaczenie 1), a następnie wybranie z wyświetlonego okna dialogowego *Wybieranie konstruktora* opcji *Konstruktor kodu* (rysunek 5.24).

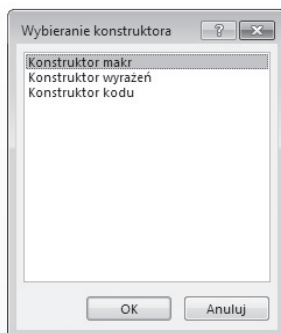
### Rysunek 5.23.

*Przycisk otwierający  
okno dialogowe  
Wybieranie  
konstruktora*



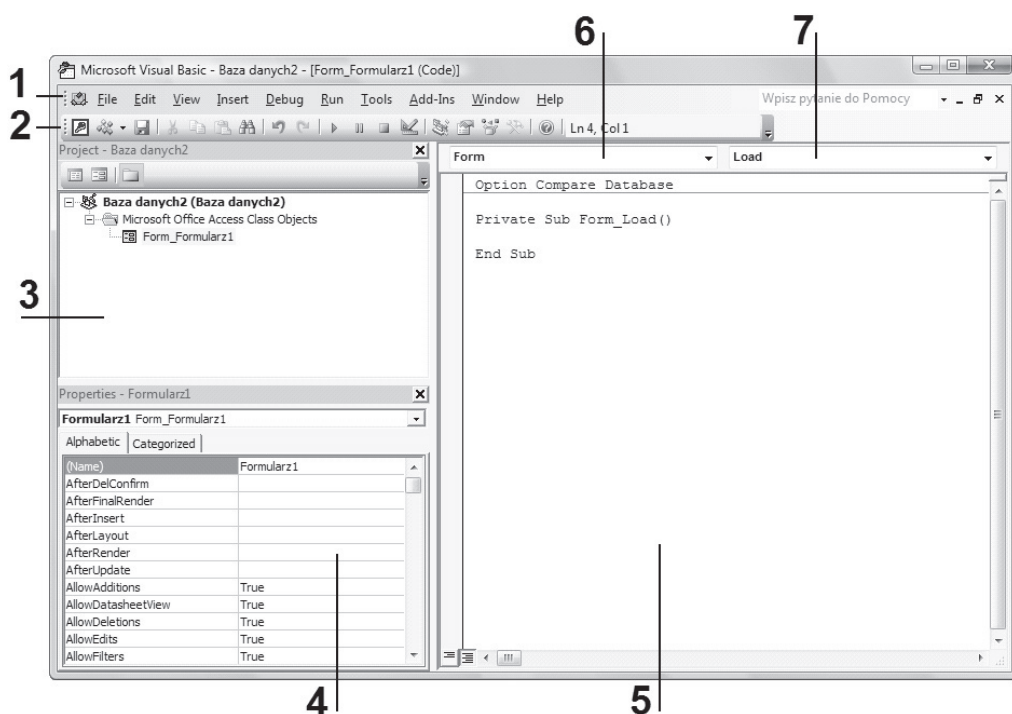
### Rysunek 5.24.

*Okno dialogowe  
Wybieranie  
konstruktora — opcja  
Konstruktor kodu*



Wykonanie jednego z wymienionych poleceń spowoduje aktywację okna edytora VBA o wyglądzie przedstawionym na rysunku 5.25. W oknie tym możemy wyróżnić następujące elementy:

- 1 — menu,
- 2 — pasek (paski) narzędzi,
- 3 — okno projektu bazy danych,
- 4 — okno właściwości,
- 5 — okno kodu,
- 6 — lista obiektów,
- 7 — lista zdarzeń obiektu.



Rysunek 5.25. Edytor kodu VBA

## Podstawowe informacje z zakresu tworzenia kodu programu w języku VBA

Ogólnie kod języka VBA dzieli się na jednostki nazywane procedurami. Procedura składa się z szeregu instrukcji (poleceń), które wykonują operacje na bazie danych lub obliczają żądane wartości. Procedura wykonywana automatycznie w odpowiedzi na zdarzenie zainicjowane przez użytkownika (np. kliknięcie przycisku) lub kod programu nazywana jest *procedurą zdarzenia*.

Procedury są przechowywane w modułach. W programie Microsoft Access istnieją trzy typy modułów:

- ◆ moduły formularzy,
- ◆ moduły standardowe,
- ◆ moduły klasy.

**Moduły formularzy** są podstawą do obsługi większości baz danych. Mogą one zawierać procedury obsługujące zdarzenia, procedury ogólne oraz deklaracje stałych i zmiennych użytych do przeprowadzenia procesów przetwarzania danych (wykonywania obliczeń). Kod zawarty w module formularza odnosi się do konkretnej bazy danych, do której należy formularz. Formularz wraz ze swoimi obiektami (formantami) może również odwoływać się (pobierać i przekazywać dane) do innych obiektów tworzących określoną bazę danych. Gdy po raz pierwszy tworzymy procedurę zdarzenia dla formularza lub raportu, program Microsoft Access automatycznie tworzy związany z nią moduł formularza lub raportu.

**Moduły standardowe** są „pojemnikami” dla procedur i deklaracji często wykorzystywanych przez inne moduły bazy danych (np. przez moduły formularzy). Na przykład może się okazać, że kod napisany w jednym z formularzy jest przydatny w innych formularzach — ze względu na to, że wykonuje ten sam proces przetwarzania danych. Ponieważ nie jest wskazane powielanie tego samego kodu w różnych formularzach, tworzy się odrębny moduł zawierający procedurę z tym właśnie zestawem instrukcji. Moduły standardowe mogą zawierać deklaracje stałych i zmiennych dostępnych dla całej bazy danych (*stałe i zmienne globalne*) i (lub) deklaracje stałych i zmiennych dostępnych tylko na poziomie modułu.



Uwaga

Kod modułu standardowego nie musi być związany z konkretną bazą danych. Jeżeli nie zostaną w nim umieszczone odwołania do nazw konkretnych formularzy lub formantów, to będzie go można zastosować także w wielu innych bazach danych.

**Moduły klasy** są podstawą programowania obiektowego, gdyż zawierają definicję nowego obiektu, to znaczy stanowią szablon niezbędny do utworzenia obiektu. Na przykład dwa obiekty typu przycisk polecenia pod względem opisujących ich cech i funkcjonalności są takie same, ponieważ utworzone są w oparciu o tę samą klasę. Obiekt stanowi więc konkretną reprezentację danej klasy. Ponieważ każdy z obiektów jest wzajemnie niezależny oraz każdy tworzony jest w oparciu o klasę, o obiekcie często mówi się jako o instancji danej klasy. Przykładowo każdy z formularzy jest instancją klasy Form, a każdy z formantów (za pomocą których konstruujemy formularz) instancją określonej klasy, w oparciu o którą został utworzony.



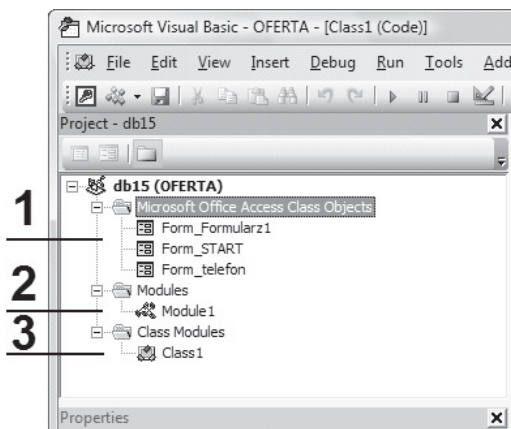
Uwaga

Moduły formularzy są w rzeczywistości modułami klas.

Dostęp do poszczególnych (opisanych wcześniej) modułów uzyskujemy w *oknie projektu* edytora VBA (rysunek 5.26) przez dwukrotne kliknięcie nazwyżądanego modułu.

**Rysunek 5.26.**

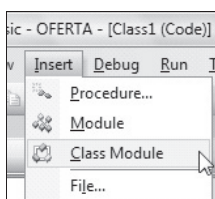
Przykładowe okno projektu edytora VBA zawierające:  
 1 — moduły formularzy,  
 2 — moduł standardowy,  
 3 — moduł klasy



Aby utworzyć (wstawić) nowy moduł standardowy lub nowy moduł klasy, należy z menu *Insert* wybrać, odpowiednio, polecenie *Module* lub *Class Module* (rysunek 5.27).

**Rysunek 5.27.**

Edytor VBA  
 — menu *Insert*



Każdy moduł formularzy, moduł standardowy i moduł klasy mogą posiadać:

- ♦ deklaracje stałych, zmiennych oraz typów,
- ♦ procedury typu Sub lub Function, zawierające fragment kodu wykonywany jako jedna całość.

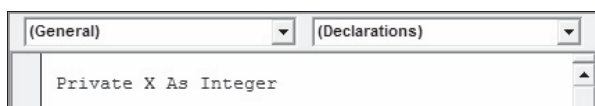
**Deklaracja zmiennych w kodzie języka VBA**

Podczas wykonywania obliczeń często zachodzi potrzeba tymczasowego przechowywania wartości. W języku VBA (jak w większości innych języków programowania) do tego celu służą *zmiennne*. Zmienna posiada *nazwę* (ciąg znaków), wykorzystywaną w celu odwołania się do wartości zmiennej, oraz *typ danych*, czyli atrybut określający rodzaj danych przechowywanych w zmiennej. Ze zmiennymi związane jest pojęcie *deklarowania*, czyli wcześniejszego poinformowania programu o ich istnieniu. Deklaracja zmiennych może być przeprowadzona w sposób jawny (*Implicit*) i niejawny (*Explicit*). Do deklarowania zmiennych używanych w obrębie określonej procedury (tzw. zmiennych lokalnych) służą słowa kluczowe Dim oraz Static, na przykład zapis Dim słowo As String oraz zapis Static rd As Integer są deklaracjami zmiennej słowo oraz zmiennej rd wraz z określeniem typu danych, odpowiednio: *łańcucha znaków* (String) oraz *liczby całkowitej* (Integer). Wartości zmiennych lokalnych, zadeklarowanych przy użyciu słowa kluczowego Static, istnieją przez cały czas działania bazy danych, natomiast zmienne zadeklarowane przy użyciu słowa Dim istnieją tylko w czasie wykonywania danej procedury.

Zmienne mające działać w obrębie modułu można utworzyć, deklarując je przy użyciu słowa kluczowego `Private` w sekcji `Declarations` na początku modułu (rysunek 5.28). Na poziomie modułu nie ma różnicy między deklaracjami `Private` i `Dim`.

### Rysunek 5.28.

Wygląd deklaracji zmiennej z poziomu modułu



Aby zmienna użyta w określonym module stała się dostępna w innych modułach, należy, deklarując ją, użyć słowa kluczowego `Public`. Wartości zmiennych publicznych są dostępne dla wszystkich procedur bazy danych. Podobnie jak zmienne poziomu modułu, zmienne publiczne deklaruje się w sekcji `Declarations` na początku modułu.



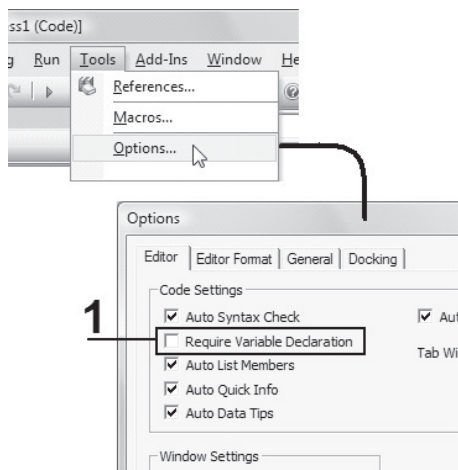
Uwaga Nie można deklarować zmiennych publicznych w procedurze, lecz tylko w sekcji `Declarations` modułu.

Język Microsoft VBA nie wymaga bezwzględnie jawnego deklarowania *zmiennej* przed użyciem jej w **procedurze**. Jeśli zostanie użyta zmienna, która nie była zadeklarowana w sposób jawny, w języku Visual Basic jest ona deklarowana niejawnie jako zmienna typu `Variant`. Mimo że takie niejawne deklarowanie jest wygodne, może prowadzić do powstawania błędów w kodzie. Można wymagać deklaracji zmiennych przed użyciem ich w procedurze. Aby Microsoft Access automatycznie dołączał instrukcję `Option Explicit` do sekcji deklaracji wszystkich nowych *modułów* w bazie danych (także do *modułów formularzy* i *raportów* skojarzonych z nowymi formularzami lub raportami), należy:

1. W menu *Tools* (rysunek 5.29) kliknąć polecenie *Options*.

### Rysunek 5.29.

Ustawianie opcji automatycznego dołączania instrukcji `Option Explicit`



2. Kliknąć kartę *Editor*.
3. W obszarze *Code Settings* zaznaczyć pole wyboru *Require Variable Declaration* (Żądaj deklaracji zmiennych) (rysunek 5.29, oznaczenie 1).

## Deklaracja stałych w kodzie języka VBA

Często można spotkać się z kodem, który zawiera powtarzające się określone wartości lub wręcz wartości te są używane do budowy kodu (np. do określenia właściwości obiektów). Mówimy wtedy, że mamy do czynienia ze *stałymi*. Stałe przypominają nieco *zmiennie*, gdyż pobranie z nich wartości odbywa się przez odwołanie do ich nazwy, jednak różnią się od zmiennych tym, że nie można ich modyfikować ani przypisać im nowej wartości. W VBA są dwa źródła stałych:

- ♦ stałe *wewnętrzne* lub *systemowe* — dostarczane przez aplikację i formanty. Stałe te są wymienione w bibliotekach obiektów systemu VBA w przeglądarce obiektów *Object Browser*. Stałe z bibliotek obiektów systemu VBA mają prefiksy „vb” — na przykład: vbNewLine,
- ♦ stałe *symboliczne* lub *zdefiniowane przez użytkownika* — są deklarowane przy użyciu instrukcji Const. Składnia deklarowania stałej jest następująca:

```
[Public|Private] Const nazwa_stałej [As type] = wyrażenie
```

gdzie:

*nazwa\_stałej* — nazwa utworzona przez użytkownika, jak w przypadku nazwy zmiennej,

*type* — typ danych określony przez użytkownika,

*wyrażenie* — liczby bądź ciągi znaków i operatorów.



Zapis w nawiasach kwadratowych [ ] oznacza występowanie danej deklaracji (instrukcji) w sposób opcjonalny.

## Procedury typu Sub

Procedury typu Sub można umieszczać w modułach formularzy, modułach standardowych oraz w modułach klas. Procedury typu Sub są domyślnie procedurami publicznymi we wszystkich modułach, to znaczy, że mogą być wywoływane w bazie danych z dowolnego miejsca. Składnia procedury typu Sub jest następująca:

```
[Public|Private][Static] Sub nazwa_procedury(argumenty)
...[deklaracje]...
...instrukcje...
End Sub
```

Argumenty procedury odpowiadają deklaracjom zmiennych, to znaczy deklaruje się w nich wartości przekazywane z procedury wywołującej dane. W VBA możemy rozróżnić dwa rodzaje procedur typu Sub:

- ♦ procedury ogólne,
- ♦ procedury zdarzenia.

*Procedura ogólna* służy w bazie danych do wykonania określonych przez użytkownika zadań i musi być jawnie wywołana przez podanie w kodzie jej nazwy. W odróżnieniu od niej *procedura zdarzenia* pozostaje nieaktywna aż do chwili pojawienia się zdarzenia

wywołanego działaniem użytkownika lub systemu. Stosowanie procedur ogólnych jest wskazane, gdy w kilku *procedurach zdarzeń* potrzebne są te same akcje. W ten sposób unikamy powielania kodu.

*Procedura zdarzenia* jest wywoływana automatycznie, jeśli dowolny obiekt rozpoznaje zajście zdarzenia (np. kliknięcie myszą w jego obszarze). Nazwa procedury zdarzenia składa się z:

- ◆ **w przypadku formantu:** aktualnej nazwy formantu (określonej przez właściwość *Nazwa* znajdującą się w kategorii *Inne*), znaku podkreślenia, nazwy zdarzenia oraz znaków (). Na przykład jeśli przycisk polecenia o nazwie *Polecenie10* ma wywołać po kliknięciu procedurę zdarzenia, należy wpisać żądane instrukcje w procedurze o nazwie *Polecenie10\_Click()*,
- ◆ **w przypadku formularza:** wyrazu *Form*, znaku podkreślenia, nazwy zdarzenia oraz znaków (). Na przykład jeśli w formularzu kliknięcie w jego obszarze wskaźnikiem myszy ma wywołać określone działanie, należy wpisać żądane instrukcje w procedurze o nazwie *Form\_Click()*.



Uwaga

Nazwy formularzy w procedurach zdarzeń nie są używane (pomimo że są unikatowe, podobnie jak nazwy formantów).

Wszystkie procedury zdarzeń mają tę samą składnię ogólną:

dla formantów:

```
Private Sub nazwaformantu_nazwa zdarzenia(argumenty)
...[deklaracje]...
...instrukcje...
End Sub
```

dla formularzy:

```
Private Sub Form_nazwa zdarzenia(argumenty)
...[deklaracje]...
...instrukcje...
End Sub
```

## Procedury typu Function

Oprócz dostępnych (wbudowanych) w VBA funkcji procedura typu *Function*, podobnie jak procedura typu *Sub*, może przyjmować argumenty i wykonywać serie instrukcji. W przeciwieństwie jednak do procedury *Sub* procedura *Function* może zwracać wartość do wywołującej ją procedury. Składnia procedury typu *Function* jest następująca:

```
[Public|Private][Static] Function nazwa_procedury(argumenty)[As type]
...[deklaracje]...
...instrukcje...
End Function
```

Procedury *Function* zwykle wywołuje się, umieszczając nazwę funkcji i jej argumenty z prawej strony instrukcji lub wyrażenia (zmiennej):

```
zwracana_wartość = nazwa_funkcji(argumenty)
```



Procedury typu Function, tak samo jak zmienne, mają pewien typ danych. Określa on typ wartości zwracanej przez funkcję. Jeżeli nie ma klauzuli As, to zwracana zmienna posiada domyślny typ Variant. Zwracanie wartości polega na przypisaniu jej do zmiennej o tej samej nazwie co nazwa funkcji. Na przykład jeśli w procesie przetwarzania danych często zachodzi potrzeba wyliczenia: *jednej trzeciej połowy pola kwadratu o boku opartym na przeciwprostokątnej trójkąta prostokątnego przy podanych długościach przyprostokątnych tego trójkąta*, to niezwykle poręcznym rozwiązaniem będzie napisanie własnej funkcji o poniższej postaci:

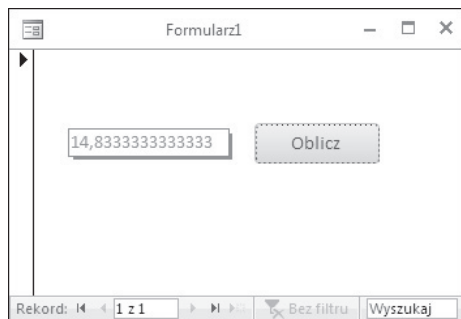
```
Function Przekwadratna (A as Integer, B as Integer) As String
    Przekwadratna = (0.5 * (SQR (A ^ 2 + B ^ 2)) ^ 2) / 3
End Function
```

gdzie:

- SQR() — jest wbudowaną funkcją VBA pierwiastka kwadratowego,
- ^ — jest operatorem potęgowania (^2 — zapis oznaczający „do kwadratu”).

Praktyczne użycie opisanej powyżej funkcji przedstawimy na przykładzie formularza (rysunek 5.30), w którym po kliknięciu przycisku polecenia (przycisk o nazwie *Oblicz*) w formancie typu etykieta zostanie wyświetlona wartość wyliczona za pomocą funkcji Przekwadratna(). Argumenty funkcji (długości przyprostokątnych) zostały podane w sposób jawny (odpowiednio: wartość 5 oraz wartość 8).

**Rysunek 5.30.**  
Wygląd formularza  
po kliknięciu  
przycisku *Oblicz*



Aby formularz mógł wykonać postawione przed nim zadanie, należy w jego module wpisać:

#### 1. Kod funkcji Przekwadratna():

```
Function Przekwadratna (A As Integer, B As Integer) As String
    Przekwadratna = (0.5 * (SQR(A ^ 2 + B ^ 2)) ^ 2) / 3
End Function
```

#### 2. Kod obsługi zdarzenia *Przy kliknięciu* przycisku polecenia:

```
Private Sub Polecenie0_Click()
    Etykieta5.Caption = Przekwadratna(5,8)
End Sub
```

Wygląd zapisu powyższych kodów w oknie edytora VBA pokazany został na rysunku 5.31.

**Rysunek 5.31.**

Wygląd kodu funkcji oraz kodu obsługi zdarzenia przycisku Oblicz w oknie edytora VBA — argumenty funkcji podane w sposób jawny

```

Polecenie0
Option Compare Database
Function Przeciwprostokątna(ByVal A As Integer, ByVal B As Integer) As String
    Przeciwprostokątna = (0.5 * (Sqr(A ^ 2 + B ^ 2)) ^ 2) / 3
End Function

Private Sub Polecenie0_Click()

    Etykieta5.Caption = Przeciwprostokątna(5, 8)

End Sub

```

W przypadku gdy chcemy przekazywać (móc wprowadzać) dane dotyczące długości przyprostokątnych w czasie pracy formularza, formularz powinien być zaopatrzony w dwa formanty typu pole tekstowe (niezwiązane), jak pokazano na rysunku 5.32.

**Rysunek 5.32.**

Wygląd formularza zaopatrzonego w formanty typu pole tekstowe



Uwaga

Dodatkowo formularz posiada trzy formanty typu etykieta wyświetlające teksty: „A=”, „B=” oraz „C=”, stanowiące opisy dla pól tekstowych oraz dla etykiety, w której jest wyświetlony wynik obliczeń funkcji.

Dla tak skonstruowanego formularza zapis funkcji `Przeciwprostokątna()` powinien mieć poniższą postać:

```

Function Przeciwprostokątna (ByVal A As Integer, ByVal B As Integer) As String
    Przeciwprostokątna = (0.5 * (SQR(A ^ 2 + B ^ 2)) ^ 2) / 3
End Function

```

Natomiast kod obsługi zdarzenia *Przy kliknięciu* przycisku polecenia *Oblicz* powinien być zapisany w następujący sposób:

```

Private Sub Polecenie0_Click()
    Tekst1.SetFocus
    przyprostokątna_A = Tekst1.Text
    Tekst3.SetFocus
    przyprostokątna_B = Tekst3.Text
    Etykieta5.Caption = Przeciwprostokątna(przyprostokątna_A, przyprostokątna_B)
End Sub

```

Wygląd zapisu powyższych kodów w oknie edytora VBA pokazany został na rysunku 5.33.

**Rysunek 5.33.**

Wygląd kodu funkcji oraz kodu obsługi zdarzenia przycisku Oblicz w oknie edytora VBA — argumenty funkcji podane za pomocą zmiennych

```

(General) | Przekształć
-----
Option Compare Database

Function Przekształć(ByVal A As Integer, ByVal B As Integer) As String
    Przekształć = (0.5 * (Sqr(A ^ 2 + B ^ 2)) ^ 2) / 3
End Function

Private Sub Polecenie0_Click()
    Tekst1.SetFocus
    przekształćna_A = Tekst1.Text
    Tekst3.SetFocus
    przekształćna_B = Tekst3.Text
    Etykieta5.Caption = Przekształćna(przekształćna_A, przekształćna_B)
End Sub

```

## Zagadnienia techniczne dotyczące tworzenia kodu programu

Do tworzenia kodu programu w języku VBA wykorzystamy ze „słów”, które stanowią:

- ♦ nazwy instrukcji programu,
- ♦ nazwy funkcji standardowych języka VBA,
- ♦ nazwy obiektów, na przykład nazwy formularzy, formantów lub raportów,
- ♦ nazwy właściwości obiektów, czyli parametrów obiektów,
- ♦ nazwy metod, czyli czynności, które można wykonać na obiekcie (każdy obiekt ma odpowiedni dla siebie zestaw metod).

Można ustawić opcje edytora Visual Basic tak, aby w oknie kodu programu były wyświetlane informacje o składni:

- ♦ gdy zostanie wpisana nazwa obiektu, a następnie znak *kropki*, wyświetlona zostanie lista dostępnych (dla danego obiektu) właściwości lub metod (rysunek 5.34) umożliwiająca — przez kliknięcie żądanej pozycji — automatyczne uzupełnienie kodu,

**Rysunek 5.34.**

Wygląd fragmentu listy właściwości i metod dostępnych dla obiektu typu pole tekstowe

```

Private Sub Polecenie0_Click()
    Tekst1.
        Section
        SelLength
        SelStart
        SelText
        SelFocus
        ShortcutMenuBar
        ShowDatePicker

```

- ♦ gdy zostanie wpisana nazwa metody lub istniejącej procedury, na przykład funkcji, a po niej *spacja* lub nawias otwierający, pojawi się porada z informacjami o składni metody lub procedury, takimi jak wymagane argumenty (rysunek 5.35).

**Rysunek 5.35.**

Wygląd porady dla standardowej funkcji *Sqr* — funkcji pierwiastka kwadratowego

```

Function Przekształćna(ByVal A As Integer, ByVal B As Integer) As String
    Przekształćna = (0.5 * (Sqr(|
        Sqr(Number As Double) As Double

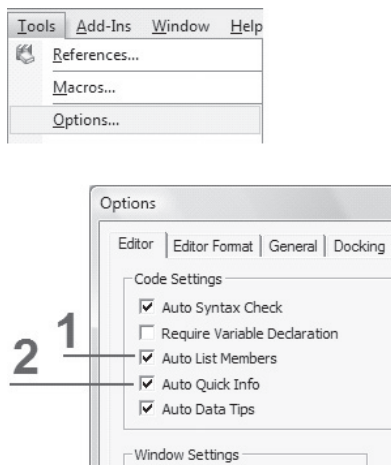
```

Aby program Microsoft Access pomagał w uzupełnianiu instrukcji i (lub) wyświetlał listę dostępnych właściwości i metod obiektu podczas pisania kodu języka Visual Basic, należy:

- ◆ W menu *Tools* kliknąć polecenie *Options*, powodując tym samym aktywację okna dialogowego *Options* (rysunek 5.36).

### Rysunek 5.36.

Ustawianie wyświetlania pomocy



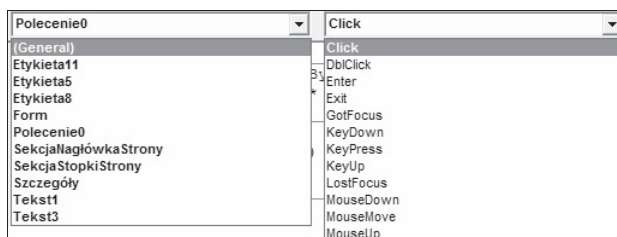
- ◆ Wybrać kartę *Editor*.
- ◆ Sprawdzić, czy w sekcji *Code Settings* zaznaczona jest opcja *Auto List Members* — powodująca wyświetlenie listy właściwości i metod obiektu (rysunek 5.36, oznaczenie 1).
- ◆ Sprawdzić, czy w sekcji *Code Settings* zaznaczona jest opcja *Auto Quick Info*, powodująca wyświetlenie składni metody lub procedury (rysunek 5.36, oznaczenie 2).

W czasie pisania kodu programu może się zdarzyć, że tworzone wyrażenie (instrukcja) jest bardzo długie, co powoduje, że początkowy ciąg znaków jest niewidoczny w oknie kodu (został przesunięty w lewą stronę). W takim wypadku należy podzielić długą instrukcję na kilka linijek w następujący sposób:

- ◆ po ostatnim elemencie składni instrukcji wpisać spację i znak `_`,
- ◆ nacisnąć klawisz *Enter*,
- ◆ kontynuować instrukcję w następnym wierszu.

Aby umieścić żądany kod w procedurze obsługi zdarzenia określonego obiektu — bez wychodzenia do formularza wyświetlonego w widoku projektu — wystarczy z listy rozwijanej obiektów wybrać nazwę obiektu, a z listy zdarzeń obiektu żądane zdarzenie (rysunek 5.37), co spowoduje, że edytor przeniesie nas automatycznie do wybranej w ten sposób procedury.

**Rysunek 5.37.**  
Przykładowy wygląd  
rozwiniętej listy  
obiektów oraz listy  
zdarzeń obiektu



## Uruchamianie makra z procedury języka Visual Basic

Do tworzenia kodu programu najczęściej stosowany jest obiekt DoCmd (*do command* — „wykonaj polecenie”). Metodami tego obiektu są takie czynności, jak zamykanie okien, otwieranie formularzy i ustawianie wartości formantów, będące odpowiednikami makr. Można na przykład użyć metody OpenForm obiektu DoCmd, aby otworzyć formularz, lub metody Hourglass, aby zmienić wskaźnik myszy na ikonę klepsydry. Zapis obiektu w kodzie VBA odbywa się według składni:

```
DoCmd.metoda [argument1, argument1, argument1...]
```

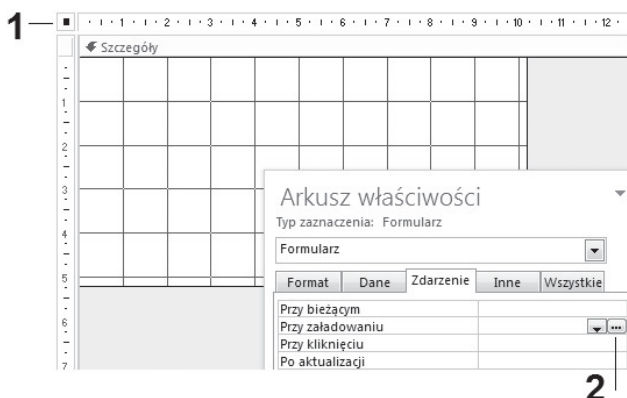
Większość z metod obiektu DoCmd ma argumenty — niektóre z nich są wymagane, inne zaś tylko opcjonalne. Argumenty są oddzielone przecinkami. Jeśli argumenty opcjonalne zostaną pominięte, przyjmą wartości domyślne dla danej metody. Na przykład metoda OpenForm używa siedmiu argumentów, ale tylko pierwszy z nich, *nazwa formularza*, jest niezbędny. Dla niektórych metod istnieją predefiniowane stałe, na przykład dla metody GoToRecord stałą taką jest acNewRec, która nakazuje przejść do nowego rekordu:

```
DoCmd.GoToRecord , , acNewRec
```

Aby z procedury języka Visual Basic uruchomić makropolecenie, należy dla obiektu DoCmd użyć metody RunMacro. Kod programu uruchamiający przykładowe makro, składające się z trzech akcji (rysunek 5.13), utworzymy według następujących kroków:

1. Otworzyć formularz w widoku *Projekt*.
2. Otworzyć arkusz właściwości formularza (np. przez dwukrotne kliknięcie selektora formularza — rysunek 5.38, oznaczenie 1).

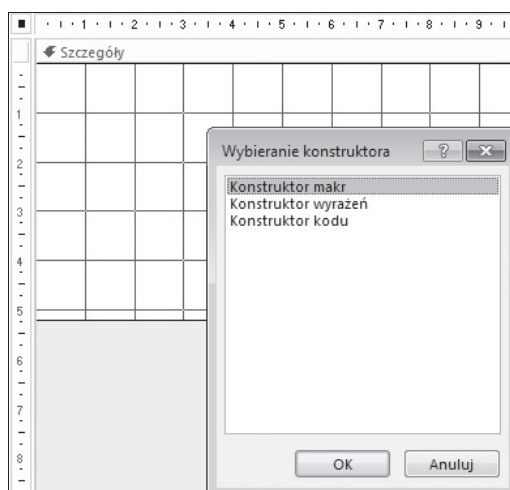
**Rysunek 5.38.**  
Okno właściwości  
formularza



- Na karcie *Zdarzenia* w polu właściwości *Przy załadowaniu* kliknąć przycisk z wielokropkiem (rysunek 5.38, oznaczenie 2), co spowoduje aktywację okna dialogowego *Wybieranie konstruktora* (rysunek 5.39).

### Rysunek 5.39.

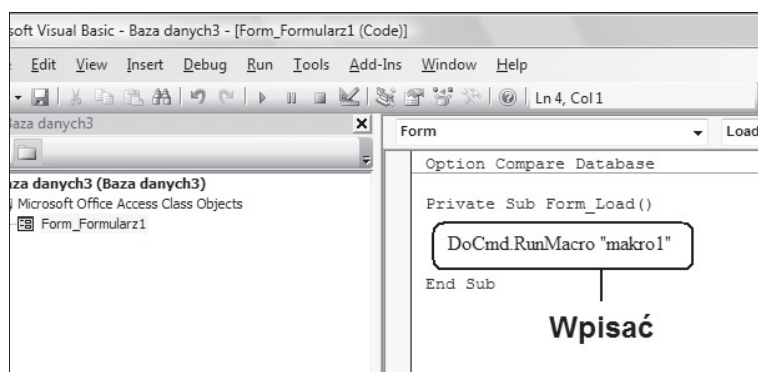
Okno dialogowe  
*Wybieranie konstruktora*



- Podświetlić opcję *Konstruktor kodu*, a następnie kliknąć przycisk *OK*.
- Działanie z punktu 4. spowoduje aktywację okna edytora kodu VBA, w którym poniżej zapisu `Private Sub Form_Load()` zobaczymy migający znak kursora.
- Wpisać kod programu: `DoCmd.RunMacro "makro1"` (rysunek 5.40).

### Rysunek 5.40.

Wygląd kodu dla obsługi zdarzenia formularza *Przy załadowaniu*

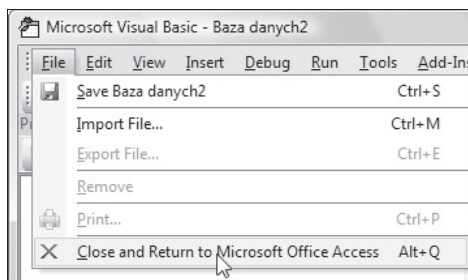


Uwaga

Wielkość liter użytych do zapisu argumentów, na przykład nazwy obiektu, nie ma znaczenia.

- Powrócić do formularza wyświetlonego w widoku *Projekt* przez wydanie z menu *File* polecenia *Close and Return to Microsoft Access* (rysunek 5.41).
- Zamknąć formularz i zapisać dokonane zmiany.
- Uruchomić formularz.

**Rysunek 5.41.**  
*Menu File — polecenie  
Close and Return  
to Microsoft Access*



Działanie tak oprogramowanego formularza będzie identyczne jak opisany wcześniej sposób działania formularza, w którym przypisano makro do właściwości *Przy załadowaniu*, i przebiegać będzie następująco:

- ♦ bezpośrednio po uruchomieniu formularza (np. przez dwukrotne kliknięcie nazwy formularza w oknie bazy) zostanie wyświetlone okno z komunikatem,
- ♦ po kliknięciu w oknie z komunikatem przycisku *OK* na chwilę zostanie wyświetlone okno formularza, po czym samoczynnie zmniejszy się ono do postaci paska (oznaczenie 1, na rysunku 5.20).





# Skorowidz

## A

Access wersja, 25, 26  
Access 2016, 7  
    pierwsze uruchamianie, 11  
    zamykanie, 13  
algorytm Gaussa, 318, 363  
autoformularz, *Patrz:* formularz związany

## B

baza danych, 9, *Patrz też:* tabela  
    aktualizacja, 260  
    format, 22  
        domyślny, 22, 24  
        zmiana, 24  
    nazwa, 22, 23  
    otwieranie, 13  
        z wcześniejszej wersji Accessa, 26  
        ze zmodyfikowanymi parametrami, 16, 17  
system zarządzający, *Patrz:* DBMS  
tworzenie, 20, 256, 257, 264  
zapisywanie, 22  
Boże Ciało, 318, 363

## D

dane  
    aktualizacja, 74, 99, 113  
        nota, 82, 84  
        obiekt OLE, 75  
alfanumeryczne, 74, 173  
arkusz, 178  
filtrowanie, *Patrz:* filtrowanie  
formatowanie, 143, 409, 410  
    warunkowe, 406  
prezentacja, 62

sortowanie, *Patrz:* sortowanie  
typ, *Patrz:* typ  
udostępnianie, 193  
wprowadzanie, 11  
data, 316  
database, *Patrz:* baza danych  
Database Management System, *Patrz:* DBMS  
DBMS, 7, 9  
dokument kartotekowy, 42

## E

ekran startowy, 12  
etykieta, 10

## F

field, *Patrz:* pole  
filtrowanie, 68, 364  
    kryteria, 72  
    według formularza, 71  
    według pustego pola, 72  
    z wyłączeniem wyboru, 70  
formant, 10, 11, 173, 209, 242  
    ActiveX, 215  
    etykieta, 211, 213, 272, 290, 332, 340, 413  
    grupa opcji, 213  
    karta, 214  
    pole, 214  
    linia, 214  
    lista, 373  
    narzędzia, 213  
obraz, 196, 211, 214, 251, 273, 280, 290  
podformularz, 214, 373, 388  
podział strony, 214  
pole  
    listy, 388  
    tekstowe, 213, 266, 332, 335, 373, 388, 413  
    wyboru, 213

pole  
 prostokąt, 214, 273, 280  
 przycisk  
 opcji, 213  
 polecenia, 214, 273, 290, 332, 373, 389, 413  
 przełącznika, 213  
 punkty konstrukcyjne, 211  
 ramka  
 niezwiązana, 196, 211, 214  
 związana, 196, 214, 266  
 usuwanie, 211  
 wstawianie, 210, 211  
 formularz, 10, 11, 173, 226, 242  
 arkusz danych, 178  
 dzielony, 181  
 forms, 10  
 główny, 202  
 hierarchiczny, 202, *Patrz też:* podformularz  
 jako okno dialogowe, 199  
 niezwiązany, 199  
 obraz  
 kopia, 199  
 wielkość, 197  
 wstawianie, 196  
 startowy, 254, 272, 274, 304, 317, 373  
 tabelaryczny, 176, 177  
 tworzenie, 183, 188, 265, 266, 274, 290, 291, 332  
 uruchamianie, 173  
 widok, 180  
 właściwości, 391, 413, 420, 421  
 zmiana, 194  
 z wieloma elementami, *Patrz:* formularz  
 tabelaryczny  
 związany, 173

**G**

Gausa algorytm, *Patrz:* algorytm Gaussa

**I**

indeks, 47

**J**

jednoreki bandyta, 251  
 język Visual Basic, *Patrz:* Visual Basic

**K**

kalendarz, 315  
 karta, 214

klucz podstawowy, 46, 47, 52, 93, 101  
 autonumerowanie, 52  
 jednopolowy, 54  
 wielopolowy, 55  
 kwerenda, 10, 116, 226  
 aktualizująca, 116, 153, 157, 327, 328, 329  
 z parametrem, 158, 159  
 dodawanie  
 pola, 118, 121  
 tabeli, 121  
 dołączająca, 116, 167, 172  
 formatowanie danych, 143  
 funkcjonalna, 116, 148  
 krzyżowa, 116, 139  
 błędy, 144  
 z parametrem, 145, 148  
 parametryczna, 116, 135, 138  
 tworząca tabelę, 116, 149, 153  
 tworzenie, 116, 117, 118, 120, 122, 323, 384  
 usuwająca, 116, 160, 163  
 z parametrem, 164, 166  
 usuwanie  
 pola, 120, 121, 123  
 tabeli, 120, 121  
 wybierająca, 116, 125

**L**

lista, 373, 388  
 rozwijana, 33

**M**

makro, 10, 11, 225, 226, 446  
 tworzenie, 226, 229, 298, 300, 301, 302, 347, 349  
 uruchamianie, 230, 231, 233, 235, 247  
 Microsoft Graph, 374  
 moduł, 10  
 formularzy, 238, 239  
 klasy, 10, 238, 239  
 standardowy, 10, 238, 239

**N**

Null, 29, 52, 54, 65, 93

**O**

obiekt, 10, 21, 209, 226  
 DoCmd, 247  
 okno, 42  
 OLE, 29, 75  
 prezentacja wizualna, 10  
 ramka, 78  
 typ, 10

obraz, 78, 196, 214, 251, 416  
 okno  
   dialogowe, 42, 199  
   nawigacji, 16, 17, 21  
   obiektu, *Patrz:* obiekt okno

**P**

Paint, 77  
 pasek przewijania, 58  
 plik, 9  
   .accdb, 10  
   .mdb, 10  
   .txt, 10  
   format, 10, 22, 24  
   nazwa, 9, 10  
 podformularz, 202, 214, 373, 388, 406  
   tworzenie, 202, 203, 205, 207, 209  
 pole, 10, 27, 31  
   blokowanie, 64  
   dodawanie, 89  
   kombi, 214  
   listy, 214, 388  
   memo, 82  
   Memo, *Patrz też:* typ nota  
   nazwa, 27, 31  
   odblokowanie, 64  
   odkrywanie, 63, 64  
   odnośników, 29, 33  
   rozmiar, 48  
     domyślny, 37  
   tekstowe, 10, 213, 266, 332, 335, 373, 388, 413  
   ukrywanie, 63  
   usuwanie, 31, 89, 90  
   wstawianie, 31  
   wyboru, 10, 213  
   zaznaczanie, 61, 62  
   zmiana położenia, 84, 85  
 polecenie  
   importuj, 30, 43  
   raport, 216  
 procedura, *Patrz też:* moduł  
   Function, 239, 242, 243  
   ogólna, 241  
   Sub, 239, 241  
   zdarzenia, 242  
 program Microsoft Graph, *Patrz:* Microsoft Graph  
 przełącznik, 213  
 przycisk, 10, 173, 373  
   Formanty ActiveX, 215  
   kreatora formantów, 215  
   opcji, 213  
   polecenia, 251, 332, 389, 413  
   przełącznika, 213  
   zaznaczania obiektów, 215

**Q**

query, *Patrz:* kwerenda

**R**

ramka, 213  
   niezwiązana, 214  
   wiązana, 214  
   związana, 266  
 raport, 10, 11, 209, 215, 226  
   kolor, 445  
   tworzenie, 215, 216, 219, 436, 438, 443, 445  
   układ, 438  
 record, *Patrz:* rekord  
 rekord, 10, 27, 30  
   dołączanie, 167  
   filtrowanie, *Patrz:* filtrowanie  
   selektor, 56, 57  
   sortowanie, *Patrz:* sortowanie  
   zaznaczanie, 61, 62  
 relacja, 91, 93  
   edytowanie, 98  
   jeden do jednego, 93  
   jeden do wielu, 94, 99, 380  
   tworzenie, 94, 380  
   usuwanie, 98  
   wiele do wielu, 93  
     tworzenie, 102, 113  
 report, *Patrz:* raport

**S**

siatka projektowa, 334  
 słowo kluczowe  
   Dim, 239  
   Private, 240  
   Public, 240  
   Static, 239  
 sortowanie, 47, 65  
   proste, 65  
   złożone, 67  
 stała, 239  
   deklarowanie, 241  
 strona dostępu, 209  
 system zarządzający bazą danych, *Patrz:* DBMS

**Ś**

święto  
   ruchome, 318, 363  
   stałe, 329, 363

**T**

tabela, 10, 27, 55, *Patrz też*: baza danych  
aktualizacja, *Patrz*: dane aktualizacja  
kolumna, 30, *Patrz też*: pole  
modyfikacja struktury, 84  
nadrzędna, 92  
nawigowanie, 56, 57, 58, 60  
podrzędna, 92, 100  
podstawowa, *Patrz*: tabela nadrzędna  
prezentacja danych, *Patrz*: dane prezentacja  
tworzenie, 28, 30, 116, 149, 153, 375, 376, 377  
importowanie, 43  
łączenie z tabelą zewnętrzną, 49  
przez wprowadzanie danych, 30, 33  
w widoku projektu, 35, 36

table, *Patrz*: tabela

typ, 28  
autonumerowanie, 29, 52  
data/godzina, 29  
hiperłącze, 29  
konwersja, 85, 86  
liczba, 29  
nota, 29, 82, 84  
obiekt OLE, 29, 75  
pole odnośników, 29, 33  
tak/nie, 29  
tekst, 29  
waluta, 29  
załączniki, 29

**V**

VBA, 236, 237  
kod programu, 245  
tworzenie, 247

moduł, *Patrz*: moduł  
procedura, *Patrz*: procedura  
stała, *Patrz*: stała  
zmienna, *Patrz*: zmienna  
Visual Basic, 10, 225  
Visual Basic for Applications, *Patrz*: VBA

**W**

wartość Null, *Patrz*: Null  
Wielkanoc, 318, 363  
wykres, 374, 420, 422  
kolor, 433  
modyfikacja, 428, 430  
typ, 423  
tytuł, 425  
wyszukiwanie, 47

**Z**

załącznik, 29  
zdarzenie  
Form\_Load, 305, 352  
formularzy obsługa, 302, 449  
obsługa, 173  
przy bieżącym, 353  
przy cyklu czasomierza, 306  
przy kliknięciu, 303, 309, 310, 357, 360  
przy załadowaniu, 304, 314  
przy zamknięciu, 353  
SpinButton, 355, 356, 357  
uruchamianie makra, 233  
zmienna, 239  
deklarowanie, 239, 240  
lokalna, 239

# PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW  
w działający bankomat!

**Dowiedz się więcej i dołącz już dzisiaj!**

<http://program-partnerski.helion.pl>

# Access 2016

**Informacja to władza.** Informacja to pieniądź. Właściwie wykorzystana przyczynia się do sukcesu projektów i przedsiębiorstw, pozwala oszczędzić czas i środki, zmniejszać nakład pracy, odkrywać złożone zależności w przyrodzie, a nawet ratować ludzkie życie. Właśnie dlatego bazy danych są tak ważne w dzisiejszym, dynamicznie rozwijającym się świecie, a zapotrzebowanie na specjalistów w tej dziedzinie stale rośnie. Jednym z najpopularniejszych narzędzi bazodanowych jest wchodzący w skład pakietu Microsoft Office program Access, który przez 18 lat swojej obecności na rynku zjednał sobie rzesze wiernych użytkowników.

**Jeśli Twoja praca wiąże się** z gromadzeniem i przetwarzaniem informacji lub po prostu interesujesz się informatyką i chcesz poznać najbardziej rozpowszechniony system relacyjnych baz danych na świecie, sięgnij po tę książkę. Krok po kroku wprowadzi Cię ona w tematykę baz danych, przedstawi podstawowe pojęcia, niezbędne do zrozumienia modelu relacyjnego, zaprezentuje interfejs programu Access, a także pokaże, jak za jego pomocą zaprojektować i utworzyć bazę, zarządzać nią oraz wypełnić ją danymi. Nauczy Cię w praktyce korzystać z tych danych za pomocą kwerend, formularzy i raportów, tworzyć wykresy i posługiwać się makrami.

- Informacje o bazach danych i programie Access
- Podstawowe operacje na bazach
- Projektowanie, tworzenie i modyfikowanie tabel
- Tworzenie, edytowanie i usuwanie różnych typów relacji
- Korzystanie z różnych rodzajów kwerend
- Tworzenie i używanie formularzy, formantów i raportów
- Posługiwanie się makrami i językiem VBA
- Tworzenie i modyfikowanie wykresów
- Praktyczne przykłady zastosowania Accessa

**Uporządkuj swoją pracę z Accessem!**

sięgnij po **WIĘCEJ**



**KOD KORZYŚCI**

**Helion** 

37508 numer katalogowy

księgarnia internetowa



<http://helion.pl>

zamówienia telefoniczne



0 801 339900



0 601 339900

Sprawdź najnowsze promocje:

- <http://helion.pl/promocje>
- Książki najchętniej czytane:
- <http://helion.pl/bestsellery>
- Zamów informacje o nowościach:
- <http://helion.pl/nowości>

Helion SA  
ul. Kościuszki 1c, 44-100 Gliwice  
tel.: 32 230 98 63  
e-mail: [helion@helion.pl](mailto:helion@helion.pl)  
<http://helion.pl>

ISBN 978-83-283-1737-6



9 788328 317376

Informatyka w najlepszym wydaniu

cena: 77,00 zł