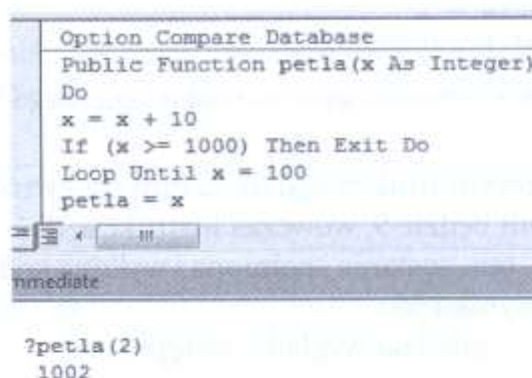


Rys. 29.23. Przykład funkcji, w której doszło do zapętlenia

Stosując instrukcję **Exit Do**, możemy opuścić pętlę i unikać sytuacji, w której mogłoby dochodzić do zapętleń. Dzięki tej technice możemy opuścić blok instrukcji **Do**, zawsze gdy znajdzie taka potrzeba.



Rys. 29.24. Przykład kodu VBA z użyciem EXIT DO

Omawiany kod można wykonać również za pomocą pętli **While Wend** i wówczas będzie mieć następującą postać:

```
Public Function petla(x As Integer)
While x <= 100
x = x + 10
Wend
petla = x
End Function
```

Rys. 29.25. Przykład pętli While Wend

Blok pętli rozpoczyna się od **While**, po którym występuje warunek. Kolejny wykonywany jest blok instrukcji pętli zakończony słowem kluczowym **Wend**.

```
While [warunek]
Instrukcja pierwsza
Instrukcja druga
...
Wend
```

Kolejnym typem pętli, bardziej elastycznym od poprzednich, jest popularna – znana z innych języków programowania – pętla **For**. Po słowie kluczowym **for** występuje licznik, którego zadaniem jest określenie, ile razy zostanie wykonany blok instrukcji **for**.

```
Public Function petla(x As Integer)
    Dim y As Integer

    For y = 10 To x
        x = x + 2
    Next

    petla = x
End Function
```

Blok instrukcji For zakończony jest słowem kluczowym Next

Rys. 29.26. Przykład kodu instrukcji FOR

Podobnie jak w przypadku pętli **Do While Until**, w wypadku **for** również możliwe jest wyjście z pętli za pomocą słowa kluczowego **Exit**, po którym umieszczamy słowo kluczowe **For – Exit For**. Analogicznie do **Exit Do** przed **Exit For** również powinien być umieszczony warunek, np. **if x=10 Then Exit For**.

```
Option Compare Database
Public Function petla(x As Integer)
    Dim y As Integer
    For y = 1 To x
        If x = 10 Then Exit For
        x = x + 10
    Next
    petla = x
End Function
```

Rys. 29.27. Przykład użycia Exit For

### PRZYKŁAD 29.1

Rozważmy powyższy fragment kodu, wprowadzając wartości 9, 10, 11 (np. **?petla(9)**). Pytanie do czytelnika: jakie wartości zostaną zwrócone przez program i dlaczego?

Podczas pracy w VBA zdarza się, że mamy kilka pętli **for**, np. pętlę w pętli.

Aby zaznaczyć, do której pętli **for** odnosi się stosowane aktualnie słowo kluczowe **Next**, bezpośrednio po **Next** umieszcza się zmienną stanowiącą **licznik** danego **For**.

```
Public Function petla(x As Integer)
    Dim y As Integer
    Dim z As String
    For y = 1 To x
        x = x + 1
        For x = 1 To y
            x = x + 1
        Next x
    Next y
    petla = x
End Function
```

Rys. 29.28. Przykład użycia Next



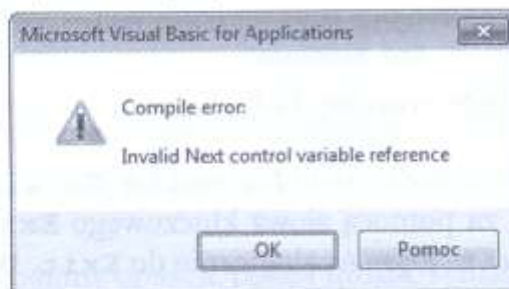
## PRZYKŁAD 29.2

Pytanie do czytelnika: jakie wartości zwróci funkcja dla parametrów od 1 do 20?

Warto pamiętać, by zawsze po **Next** używać sprawdzonych i właściwych identyfikatorów. Odwołanie do nieistniejącego identyfikatora może powodować problemy.

Gdy w 9. linii kodu zamiast **y** użyjemy **h**, wówczas zobaczymy komunikat błędu:

```
Public Function petla(x As Integer)
Dim y As Integer
Dim z As String
For y = 1 To x
x = x + 1
For x = 1 To y
x = x + 1
Next x
Next
petla = x
End Function
```



Rys. 29.29. Przykład komunikacji błędu w wypadku użycia niewłaściwego identyfikatora

Warto dodać, że pętle nie mogą się zazębiać, co oznacza, iż pętle wewnętrzne muszą być zakończone przed zewnętrznymi.

## Składnia If

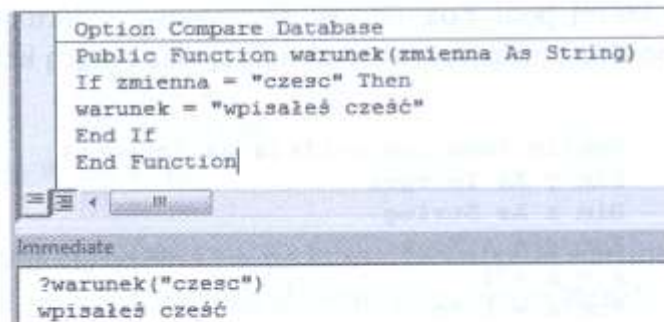
Podczas programowania w VBA, gdy zachodzi potrzeba zbadania warunku i – w zależności od jego prawdziwości – późniejszego wykonania instrukcji, stosuje się słowo kluczowe **If**.

Konstrukcja warunkowa rozpoczyna się od słowa kluczowego **If**, po którym występuje warunek, a następnie słowo kluczowe **Then**. **Then** wpisujemy pomiędzy warunkiem logicznym znajdującym się bezpośrednio po **If** a instrukcją (instrukcjami) znajdującą się przed **End If**. Jeśli warunek jest spełniony, jest to równoznaczne z przekazaniem do **If** wartości **true** i zostaje wówczas wykonana instrukcja lub grupa instrukcji znajdujących się pomiędzy **If** a **End If**.

```
IF [warunek] Then
Instrukcja...
```

```
...
```

```
End If
```



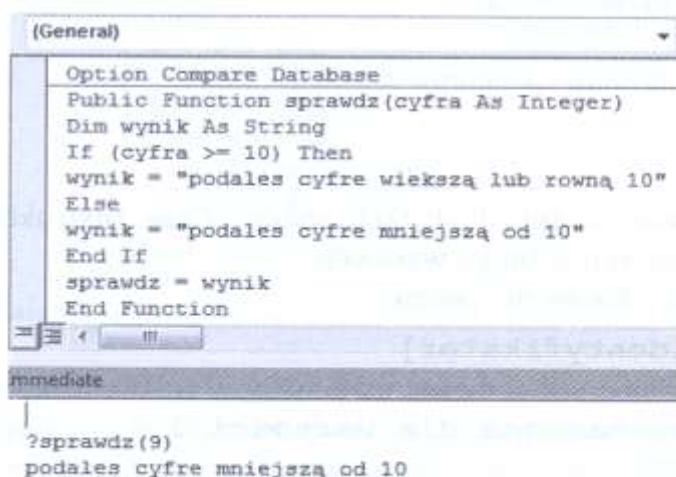
Rys. 29.30. Przykład kodu z zastosowaniem instrukcji warunkowej If Then

Powyższy program operuje na zmiennych typu string (zmienna **As String**). Wprowadzona wartość ma zatem charakter łańcucha znaków, dlatego wprowadzana jest w "" (znakach cudzysłowu).

If pozwala porównywać nie tylko liczby, lecz także ciągi znaków, co może okazać się szczególnie użyteczne podczas tworzenia programów dla bazy Access.

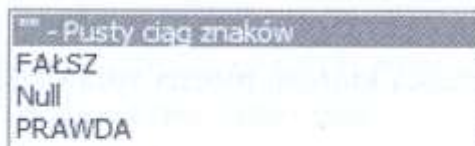
Osoby znające inne popularne języki programowania wysokiego poziomu z pewnością ucieszy to, że w VBA występuje również słowo **Else**.

Używane jest ono przy składni **If**, jeśli chcemy zaznaczyć, że pewna instrukcja ma zostać wykonana wtedy, gdy warunek nie będzie spełniony. Działanie takiego kodu przedstawiono na poniższym przykładzie:



Rys. 29.31. Zastosowanie słowa kluczowego Else

Warunki logiczne, których możemy używać, tworząc programy w VBA, oparte są na poniższych operatorach:



Rys. 29.32. Operatory logiczne w VBA

Tabela 29.1. Operatory logiczne

Operator	Działanie
=	Służy do porównywania, sprawdzania, np. czy a=b
>	Czy wartość po prawej stronie operatora jest większa
<	Czy wartość po prawej stronie operatora jest mniejsza
>=	Operator większy lub równy
<=	Mniejszy lub równy
<>	Różny
AND	Logiczne i
OR	Logiczne lub
NOT	Zaprzeczenie
XOR	Ex or Alternatywa wykluczająca



Użycie kilku operatorów w instrukcji **If** może mieć następującą postać:

```
Option Compare Database
Public Function sprawdz(cyfra As Integer)
Dim wynik As String
If (cyfra < 100 And cyfra > 90) Then
wynik = "Podales cyfre z przedzialu od 90 do 100"
Else
wynik = "Podales cyfre poza przedzialem"
End If
sprawdz = wynik
End Function
```

Rys. 29.33. Przykład użycia kilku operatorów w instrukcji If

## Użycie CASE

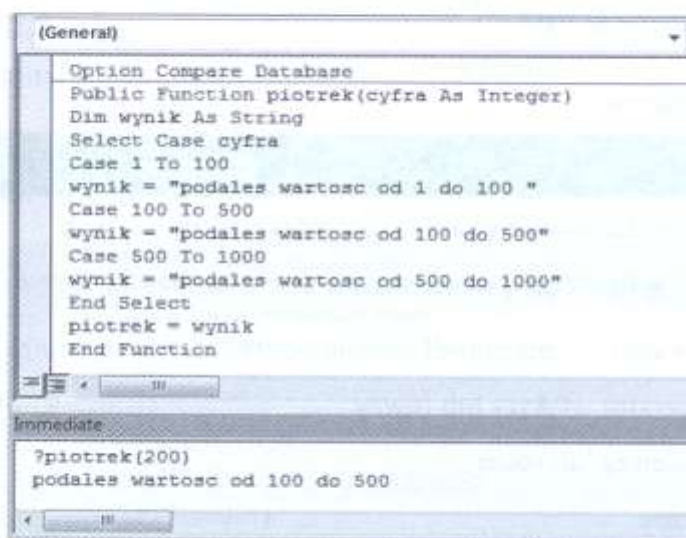
W VBA można stosować również instrukcje wyboru **Case**. Instrukcja porównuje zmienną pod kątem występujących w bloku wartości.

Konstrukcja instrukcji **Case** ma postać:

```
Select Case [identyfikator]
Case "wartosc1"
Instrukcja przeznaczona dla wartości 1
Case "wartosc2"
Instrukcja przeznaczona dla wartości 2
Case "wartosc3"
Instrukcja przeznaczona dla wartości 3
End Select
```

### PRZYKŁAD 29.3

Przykładowy kod, za pomocą którego można sprawdzić działanie instrukcji **Case** w VBA, ma postać:



Rys. 29.34. Zastosowanie instrukcji Case w VBA

Powyższy przykład pozwala prześledzić działanie funkcji **Case** dla wartości mieszczącej się w określonym przedziale, np.: fragment kodu **Case 100 to 500** wskazuje na przedział dla zmiennej **cyfra** (**Select Case cyfra**). Jeśli zmienna **cyfra** będzie

przechowywać wartość z wymienionego przedziału, wówczas zostanie wykonana instrukcja znajdująca się bezpośrednio po **Case 100 to 500**, tzn. do zmiennej typu string o identyfikatorze **wynik**, zostanie przypisany łańcuch znaków „podałeś wartość od 100 do 500”. Zamiast przedziału **Case 100 to 500**, możemy użyć porównania do pojedynczej wartości, np.

```
Option Compare Database
Public Function selekcja(x)
Select Case x
Case 1
x = x + 100
Case 2
x = x + 200
Case Else
x = "Wpisz 1 albo 2"
End Select
selekcja = x
End Function
```

Rys. 29.35. Przykład działania instrukcji Case

Ponieważ nie zadeklarowaliśmy wcześniej zmiennej *x*, zostanie zadeklarowana w locie, tzn. w momencie przypisania jej określonej wartości. Jak ilustruje powyższy przykład, funkcja może zwrócić nie tylko wartość liczbową, lecz także wartość typu string – ciąg znaków.

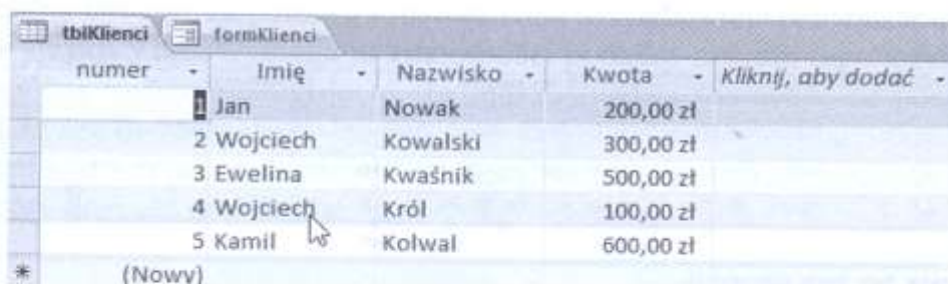
## VBA w praktyce

W programie Access 2010 wszystkie tworzone przez użytkownika elementy – typu tabela, formularz, kwerenda, raport – noszą nazwę obiektów i kolejny dostęp do nich uzyskiwany jest przez stosowanie znaku **wykrzyknika**. Gdy mamy do czynienia z obiektami predefiniowanymi przez system (obiektami, właściwościami, metodami), wówczas stosujemy znak **kropki**.

**Formularze!moj\_formularz!mojaMetoda.wartość**

Łatwo zauważyć, że oprócz wykrzyknika w wyrażeniu występuje również znak kropki. Jeżeli przyjmiemy, że tabele, kwerendy i formularze są obiektami tworzonymi przez użytkownika, to po kropce występują wartości, metody, którymi dysponuje system – w naszym przypadku Access.

Innymi słowy, projektant bazy danych Access stworzył bazę danych – nazwał ją **sprzedaż**. W tej bazie danych utworzył nowy obiekt – tabelę i nazwał ją **tblKlienci**.



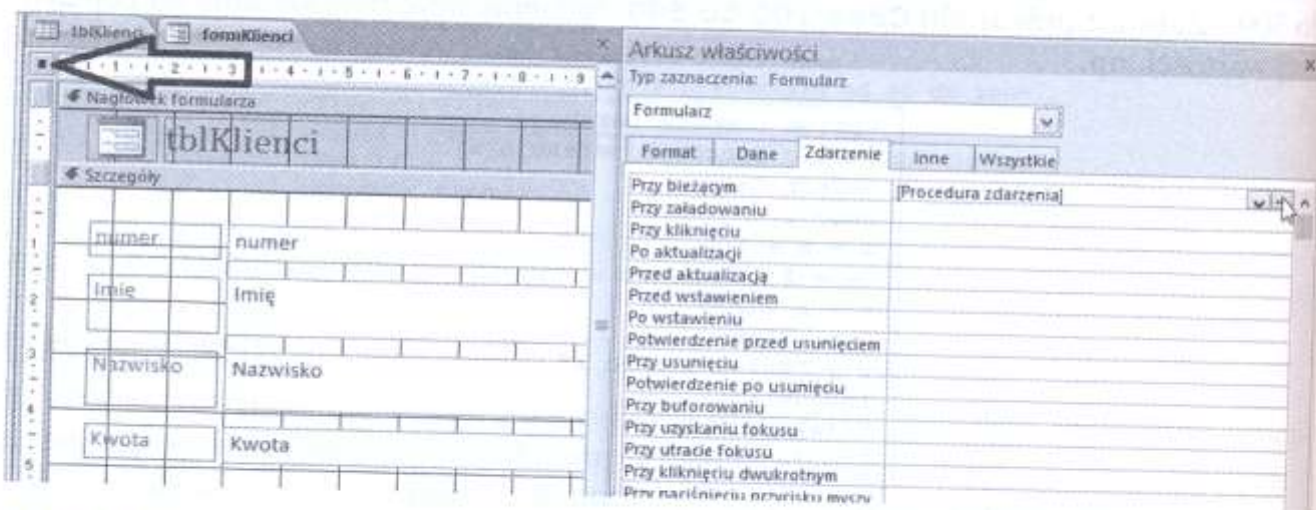
numer	Imię	Nazwisko	Kwota	Kliknij, aby dodać
1	Jan	Nowak	200,00 zł	
2	Wojciech	Kowalski	300,00 zł	
3	Ewelina	Kwaśnik	500,00 zł	
4	Wojciech	Król	100,00 zł	
5	Kamil	Kolwał	600,00 zł	
*	(Nowy)			

Rys. 29.36. Przykład tabeli

Dla wygody pracowników w firmie, którzy wprowadzali dane, utworzył formularz i nazwał go **formKlienci**. Ponieważ chciał się dowiedzieć, ilu klientów to kobiety, utworzył do tabeli kwerendę – nazwał ją **kweKobiety**. Aby podzielić się wynikami swojej pracy z zespołem, tworzył raport drukujący spis kobiet spośród klientów firmy

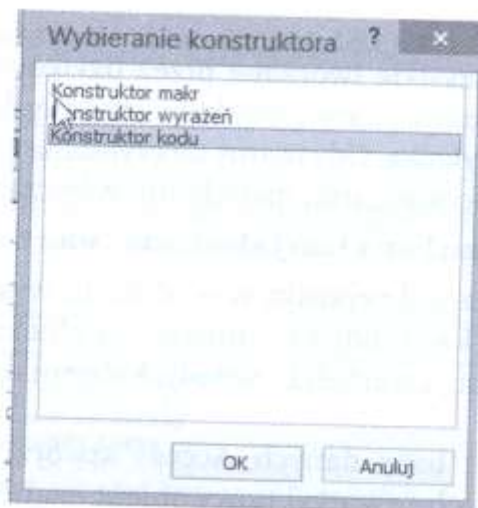


(raport utworzył na podstawie kwerendy) i nazwał go **rptKweKobiety**. Jeśli nasz projektant zechce stworzyć kod w VBA, wówczas, tworząc formularz, poszczególne elementy kodu powiąże z akcjami formularza.

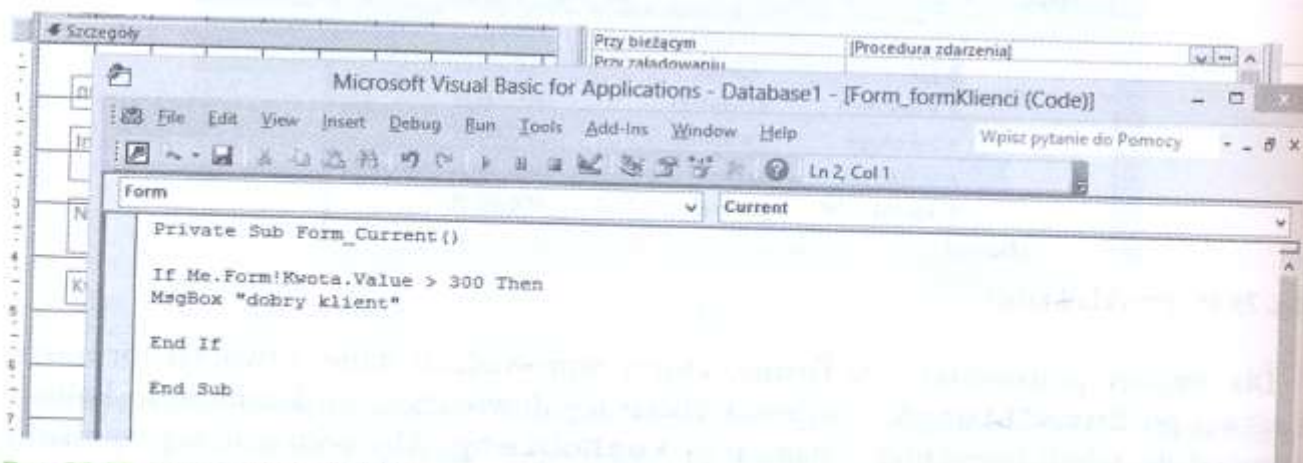


Rys. 29.37. Przykład tworzenia kodu VBA dla formularza

W **Widoku projektu** formularza klikamy oznaczony strzałką lewy górny róg, a następnie w polu **Przy bieżącym** zakładki **Zdarzenie** wybieramy oznaczony kursorem myszy blok z widocznymi trzema kropkami.



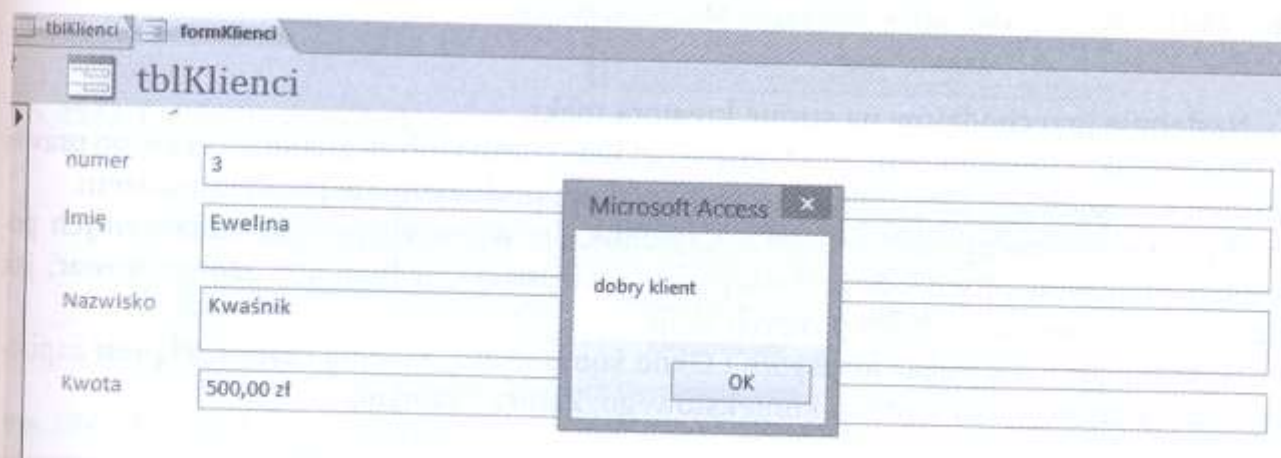
Rys. 29.38. Wybór konstruktora kodu



Rys. 29.39. Fragment kodu VBA

Po wybraniu opcji **Konstruktor kodu** (rys. 29.38) i zatwierdzeniu czynności za pomocą przycisku **OK** umieszczamy w oknie konstruktora następujący fragment kodu VBA (rys. 29.39).

Przykład kodu VBA zaprezentowany na poprzedniej stronie sprawi, że podczas przeglądania kolejnych rekordów, gdy natrafimy na klienta, dla którego wartość kwoty przekroczy 300, program poinformuje nas o tym w postaci okienka (MsgBox) z napisem **dobry klient**.



Rys. 29.40. Przykład działania kodu VBA podczas pracy z formularzem

## Makra

Podczas pracy z programem Access dość często pojawia się temat makr, który poruszany jest również podczas pracy z arkuszem kalkulacyjnym i innymi programami, gdzie zachodzi potrzeba zautomatyzowania pracy. Makro jest programem uruchamianym w środowisku innego programu i realizuje pewien algorytm działania, który polega na wykonaniu ciągu instrukcji mających na celu zautomatyzowanie i usprawnienie pracy z programem. Makra pisane są w językach skryptowych – w naszym wypadku językiem tym jest VBA. Podsumowując powyższe stwierdzenia, możemy uznać, że makro to rodzaj skryptu pisanego i działającego w ramach określonego programu, np. Access, Excel, AutoCAD. Jeżeli zachodziłaby potrzeba np. policzenia wyrazów zawierających „ó” w pracy magisterskiej znajdującej się w programie Word, wówczas piszemy makro zliczające wyrazy – ponieważ program może nie mieć takiej funkcji.

Makra można tworzyć za pomocą kreatora i jest to najłatwiejsza metoda. Można je również pisać w języku VBA. Wymaga to znajomości podstaw składni tego języka.

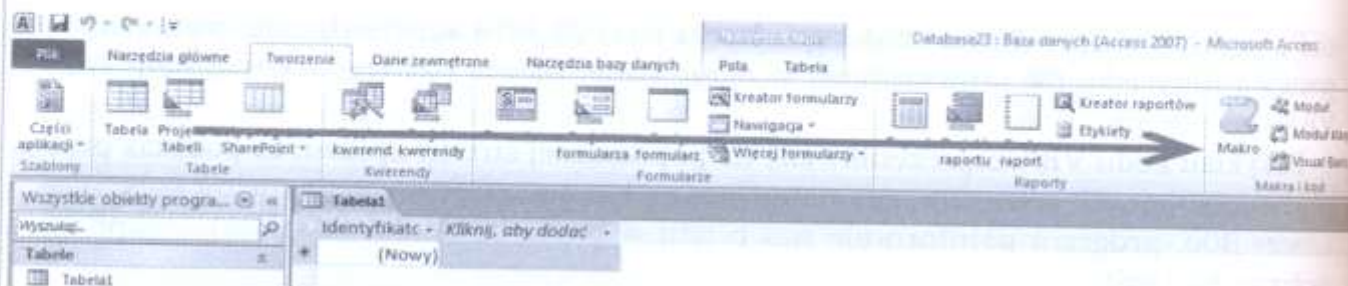
Na szczególną uwagę zasługuje to, że makro można utworzyć za pomocą kreatora i przekonwertować do kodu VBA – co przyspiesza pracę i zwiększa elastyczność tej funkcji.

Dla początkujących programistów VBA będzie to spora pomoc, gdyż mogą analizować kod VBA, który posiada makro utworzone za pomocą kreatora.

## Tworzenie makra

Aby utworzyć makro w programie Microsoft Access 2010, używamy zakładki **Tworzenie**, z której wybieramy **Makro**:





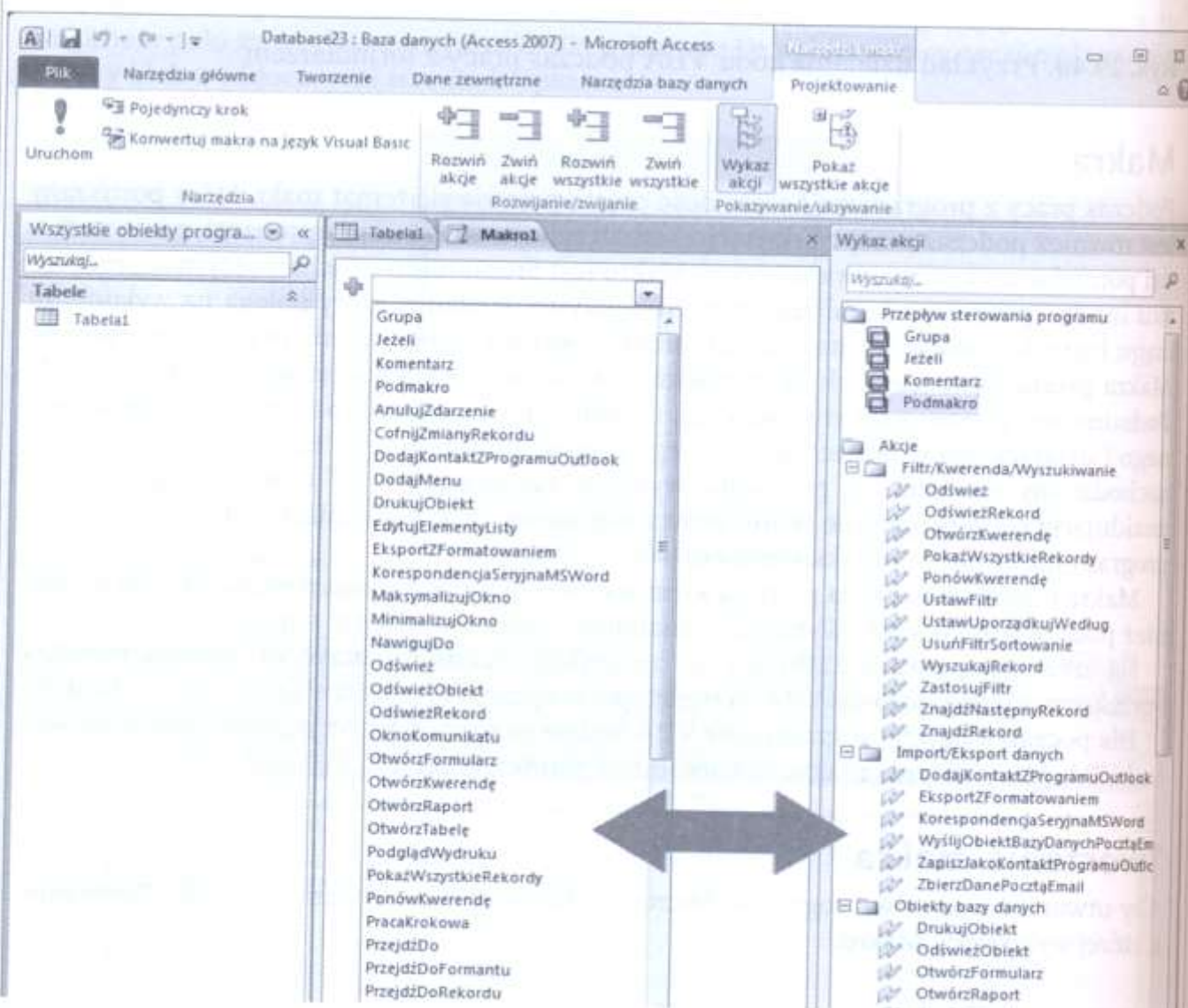
Rys. 29.41. Zakładka Tworzenie programu Microsoft Access

Następnie przechodzimy na stronę kreatora makr.

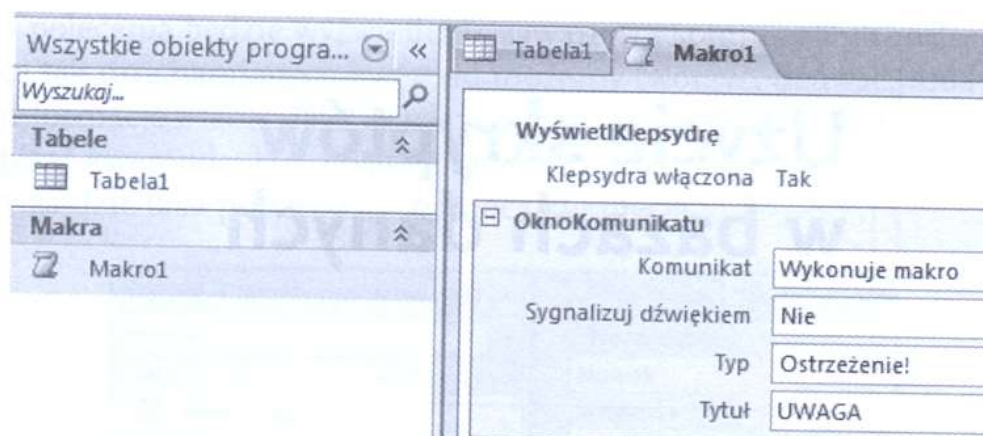
Akcję makra możemy wybrać z wysuwanej listy combo lub ze znajdującej się po prawej stronie wyszukiwarki akcji. Dla naszego przykładu posłużymy się prostym makrem.

Wyberzemy następujące wartości. Czytelnik dla własnych potrzeb edukacyjnych powinien przetestować większość dostępnych akcji makra, jednak aby zaobserwować, jak działa makro, wystarczy kilka prostych akcji.

Wybrane akcje to: **Pokaż klepsydę** i **Okno komunikatu**. Kolejną czynnością jest zapisanie makra przez wybór z menu kontekstowego: **Zapisz i zamknij**.



Rys. 29.42. Akcje oferowane przez program Access

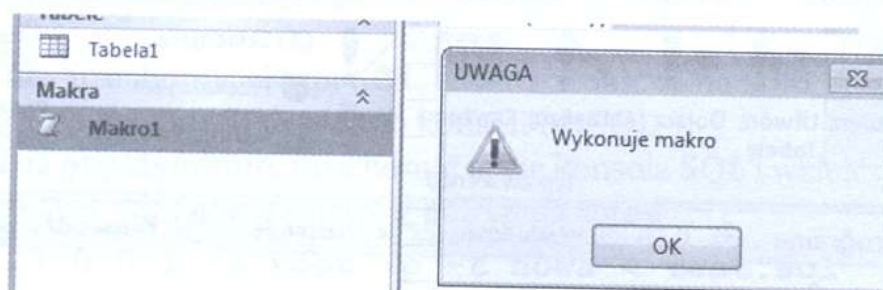


Rys. 29.43. Przykład zapisu makra



Rys. 29.44. Przykład zapisu makra

Po kliknięciu makra powinien zostać wyświetlony komunikat:



Rys. 29.45. Wykonanie Makra

W zależności od typu kursora myszy podczas wyświetlania komunikatu może być wyświetlana klepsydra. Makr takich możemy używać do operacji na danych w tabelach.

## SPRAWDŹ SWOJĄ WIEDZĘ

1. Wyjaśnij pojęcie Visual Basic.
2. Jakie zadania wykonują polecenia **sub** i **function**?
3. Wymień rodzaje modułów.
4. Jakie cechy posiadają następujące słowa kluczowe: **public**, **as**, **me**, **for**, **next**, **if**?
5. Jakie zastosowanie w Visual Basic ma podkreślnik?
6. Jaką funkcję pełni instrukcja **Exit Do**?
7. Czym są pętle **While Wend**, **Exit For**, **Exit Do**?
8. Co to jest makro i jak się je tworzy?