# Programowanie w Excelu 2007 PL. Niebieski podręcznik


Autor: Denise Etheridge
Tłumaczenie: Paweł Koronkiewicz
ISBN: 978-83-246-1663-3
Tytuł oryginału: Microsoft Office Excel 2007 Programming: Your visual blueprint for creating interactive spreadsheets (Visual Blueprint)
Format: 170x230, stron: 360

**Wykorzystaj niezwykłe możliwości Excela i spraw, aby pracował za Ciebie**

- Jak stworzyć własne okno dialogowe w edytorze VBA?
- Jak otwierać i modyfikować pliki XML w Excelu?
- Jak przypisywać makra do paska szybkiego dostępu?

Excel – najpopularniejszy elektroniczny arkusz kalkulacyjny – udostępnia narzędzie do zautomatyzowania czynności wykonywanych w tym programie. Jest to język programowania nazywany Visual Basic for Applications (VBA). Pozwala on na tworzenie makr, czyli zapisu pewnej sekwencji poleceń, które mogą zostać automatycznie wykonane jako całość. Zapisanie makr w rejestratorze i przygotowanie w ten sposób konkretnego dokumentu, na przykład raportu miesięcznego, sprawia, że każdy następny tego rodzaju dokument Excel wykonuje samodzielnie. Dzięki VBA można także modyfikować makra oraz tworzyć bloki poleceń, przygotowywać własne aplikacje i dodatki do programu głównego.

Książka „Programowanie w Excelu 2007 PL. Niebieski podręcznik" to przewodnik, który w prosty i przejrzysty sposób zapozna Cię z systemem makr Excela. Dokładnie opisano tu ponad 140 technik programowania, które dodatkowo zostały bogato zilustrowane za pomocą ułatwiających zrozumienie materiału zrzutów ekranowych. Dzięki temu dowiesz się, na czym polega deklarowanie tablic wielowymiarowych i cyfrowe podpisywanie makr. Nauczysz się, jak modyfikować wstążkę poleceń, tworzyć makra, zmienne obiektowe i własne dodatki. Będziesz umiał tak zautomatyzować czynności programu przy tworzeniu dokumentów, aby to Excel pracował za Ciebie.

- Visual Basic for Applications
- Makra i formanty
- Model obiektów Excela
- Deklarowanie tablic
- Instrukcje sterujące
- Funkcje arkuszy
- Debugowanie kodu
- Skoroszyty i pliki
- Arkusze i wykresy
- Praca z zakresami komórek
- Zdarzenia
- Pliki XML

**Patrz i ucz się, jak wykorzystać możliwości programu, aby pracować szybko i bez wysiłku!**

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

# Spis treści

# Operatory porównania

Wyrażenia porównujemy przy pomocy operatorów porównania. Wynikiem takiej operacji jest zawsze jedna z dwóch wartości: `True` (prawda) lub `False` (fałsz). Najprostsze wyrażenie tego rodzaju to `A = B`, prowadzące do porównania wartości zmiennych `A` i `B`. Wynikiem obliczenia jego wartości jest `True`, jeżeli wartość zapisana w zmiennej `A` jest taka sama jak wartość zapisana w zmiennej `B`.

Operator porównania zawsze umieszcza się między dwoma wyrażeniami. Poza znakiem równości do dyspozycji są znaki: nierówności (<>), większości (>), mniejszości (<), „większe lub równe" (>=) i „mniejsze lub równe" (<=).

## Wskazówka

W tabeli przedstawiamy zestawienie operatorów porównania.

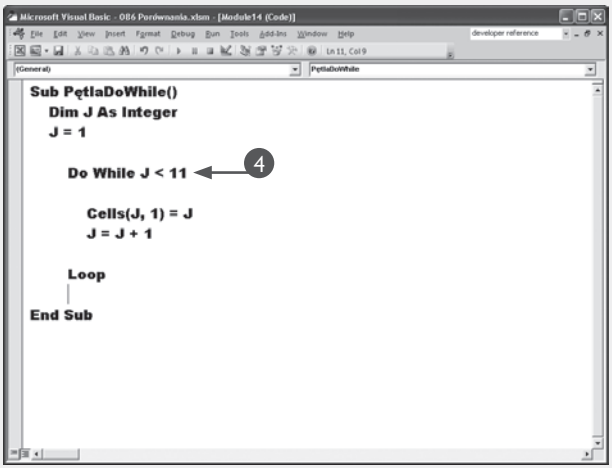
| OPERATOR | FUNKCJA |
|---|---|
| = | jest równe |
| <> | jest różne od |
| > | jest większe niż |
| < | jest mniejsze niż |
| <= | jest mniejsze lub równe |
| >= | jest większe lub równe |

## Operacje porównania

1. Operator porównania w pętli `Do While`.

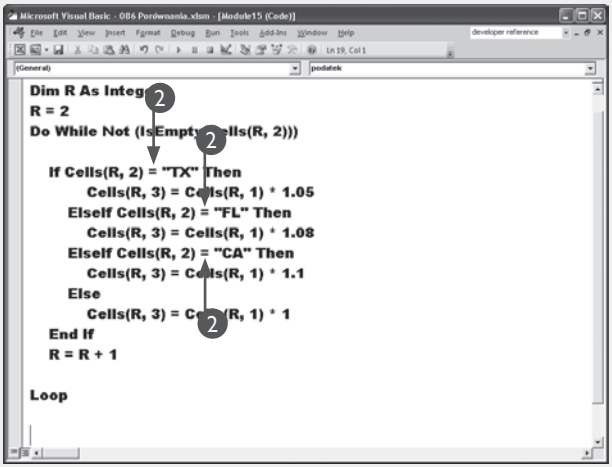
   W tym przykładzie, dopóki `J` jest mniejsze od 11, program wykonuje kod wewnątrz pętli.

**Uwaga:** O pętlach `Do While` piszemy w podrozdziale „Pętla Do While".




2. Operator porównania w instrukcji `If Then ElseIf`.

**Uwaga:** O instrukcjach `If` piszemy w podrozdziale „Instrukcja If Then Else".



# Operatory logiczne

Do łączenia wyrażeń porównania używa się operatorów logicznych: `Or`, `And`, `Xor`, `Eqv`, `Imp` i `Not`.

## Logiczna operacja Or

Logiczny operator `Or` zwraca wartość `True`, gdy co najmniej jedno z dwóch wyrażeń zwraca `True`.

| WYRAŻENIE A | WYRAŻENIE B | WYNIK |
|---|---|---|
| prawda | prawda | prawda |
| prawda | fałsz | prawda |
| fałsz | prawda | prawda |
| fałsz | fałsz | fałsz |

**Przykład:**

```
Sub LogiczneOr()
Dim Wynik As Boolean
Wynik = 10 < 20 Or 30 < 20 'Zwraca True
MsgBox (Wynik)
End Sub
```

## Logiczna operacja And

Logiczny operator `And` zwraca wartość `True`, jeśli oba wyrażenia zwracają `True`.

| WYRAŻENIE A | WYRAŻENIE B | WYNIK |
|---|---|---|
| prawda | prawda | prawda |
| prawda | fałsz | fałsz |
| fałsz | prawda | fałsz |
| fałsz | fałsz | fałsz |

## Logiczna operacja Xor

Logiczny operator `Xor` zwraca wartość `True`, gdy jedno i tylko jedno z dwóch wyrażeń jest prawdziwe.

| WYRAŻENIE A | WYRAŻENIE B | WYNIK |
|---|---|---|
| prawda | prawda | fałsz |
| prawda | fałsz | prawda |
| fałsz | prawda | prawda |
| fałsz | fałsz | fałsz |

## Logiczna operacja Eqv

Logiczny operator `Eqv` zwraca wartość `True` tylko wtedy, gdy oba wyrażenia są prawdziwe lub gdy oba są fałszywe.

| WYRAŻENIE A | WYRAŻENIE B | WYNIK |
|---|---|---|
| prawda | prawda | prawda |
| prawda | fałsz | fałsz |
| fałsz | prawda | fałsz |
| fałsz | fałsz | prawda |

## Logiczna operacja Imp

Logiczny operator `Imp` zwraca wartość `True`, o ile nie zachodzi sytuacja, że pierwsze wyrażenie jest prawdziwe, a drugie fałszywe.

| WYRAŻENIE A | WYRAŻENIE B | WYNIK |
|---|---|---|
| prawda | prawda | prawda |
| prawda | fałsz | fałsz |
| fałsz | prawda | prawda |
| fałsz | fałsz | prawda |

## Logiczna operacja Not

Logiczny operator `Not` neguje wartość wyrażenia, zmieniając wartość `True` na `False` i `False` na `True`.

**Przykład:**

```
LogiczneNot()
Dim Wynik As Boolean
Wynik = Not (10 = 10) 'Zwraca False
MsgBox (Wynik)
End Sub
```

# Pętla Do While

Pętla `Do While` („wykonuj, gdy”) pozwala powtarzać wykonywanie bloku instrukcji, dopóki dane wyrażenie jest prawdziwe. Jej składnia to:

```
Do [While warunek]
    [instrukcje]
Loop
```

Warunek to wyrażenie, które zwraca `True` lub `False`. Gdy program napotyka na pętlę `Do While`, najpierw sprawdza wartość warunku pętli. Jeżeli jest on spełniony (zwraca `True`), są wykonywane instrukcje wewnątrz pętli. Sprawdzanie warunku i wykonywanie instrukcji są powtarzane do momentu, gdy warunek zwróci wartość `False`. Spowoduje to przejście do wykonywania instrukcji następującej po zamykającym pętlę słowie `Loop`.

W pętli `Do While` można wyróżnić cztery główne elementy: instrukcję `Do` inicjującą pętlę, instrukcję `While`, która bada warunek pętli, blok instrukcji wykonywanych, gdy warunek jest spełniony, oraz końcową instrukcję `Loop`.

Można też stosować składnię pętli `Do Loop While`:

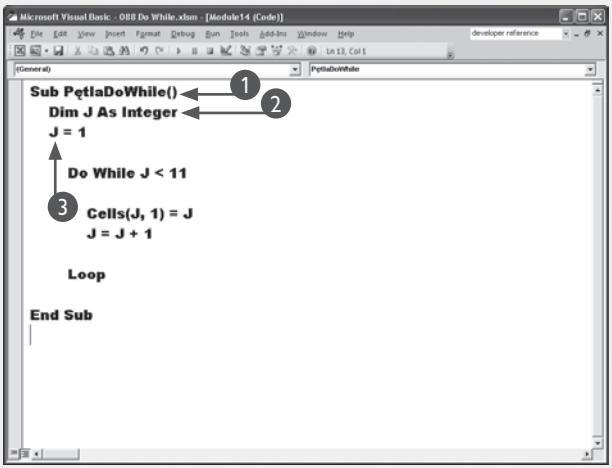
```
Do
    [instrukcje]
Loop [While warunek]
```

Jej działanie jest podobne, ale warunek sprawdzany jest zawsze po wykonaniu bloku instrukcji, co powoduje, że wykonanie następuje co najmniej raz.

## Pętla Do While

1. Utwórz nową procedurę.
2. Zadeklaruj zmienną.
3. Zainicjuj zmienną.

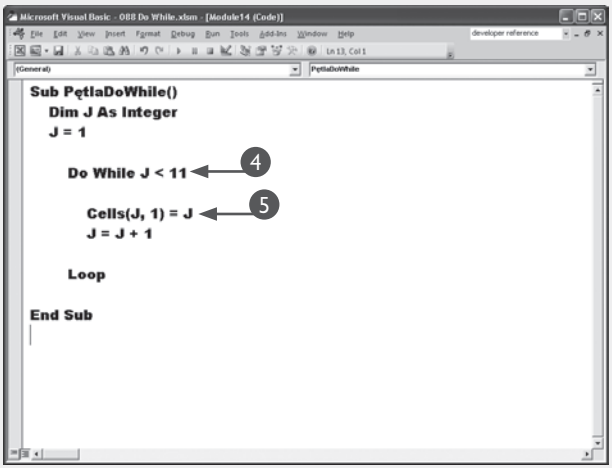
   W przedstawionym przykładzie zmiennej `J` przypisana jest wartość 1, aby dalej użyć jej jako licznika pętli.




4. W pierwszym wierszu pętli nastąpi sprawdzenie stanu licznika.

   W przedstawionym przykładzie sprawdzana jest wartość zmiennej `J` i powtarzany jest blok instrukcji, dopóki jest ona mniejsza niż 11.
5. Zapisz wartość `J` w komórce arkusza.

   W przedstawionym przykładzie numer wiersza jest równy wartości `J`, a numer kolumny to 1.



6. Zwiększ wartość `J`.

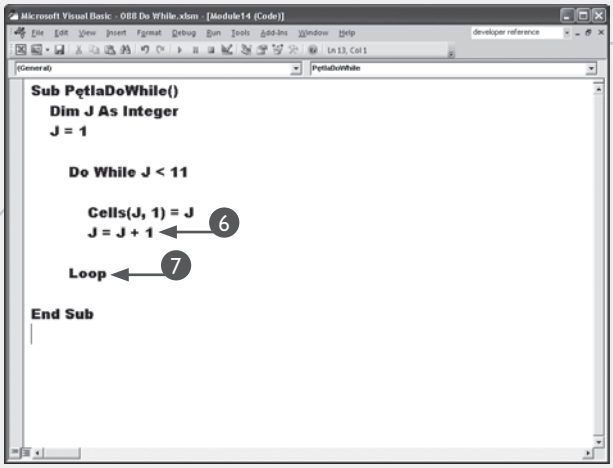
   Tutaj dodawane jest 1 do bieżącej wartości `J`.

7. Zakończ pętlę instrukcją `Loop`.

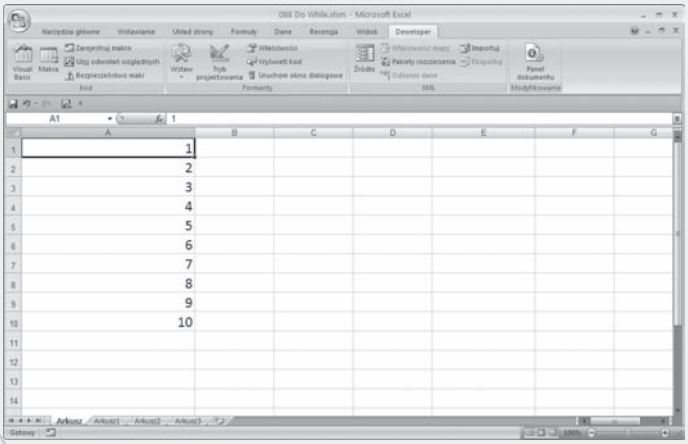
   Program powróci do instrukcji `Do While` i będzie kontynuować powtarzanie pętli do czasu, gdy warunek nie będzie spełniony.

8. Wciśnij kombinację klawiszy *Alt+F11*, aby powrócić do Excela i uruchomić kod.

**Uwaga:** O uruchamianiu procedur piszemy w rozdziale 1.




Procedura umieści w komórkach arkusza wartości od 1 do 10.




## Wskazówka

Pętla musi zawierać instrukcję, której wykonanie prowadzi do zmiany wyniku obliczania warunku pętli. Jeżeli nie zwróci on w pewnym momencie wartości `False`, utworzy się **pętla nieskończona** (ang. *infinite loop*).

Aby uniknąć takiej sytuacji, dobrze jest stosować jasno określoną zmienną licznika. W przedstawionym poniżej przykładzie jest to zmienna `J` o początkowej wartości 1. Każde kolejne wykonanie pętli zależy od tego, czy stan licznika jest mniejszy od 5. Główna instrukcja wykonywanego bloku przeprowadza operację zapisywania danych do arkusza. W ostatniej licznik jest zwiększany. Bezpośrednio po tym następuje ponowne sprawdzenie warunku. W efekcie pętla zostaje powtórzona cztery razy, za piątym razem wartość `J` wynosi 5 i warunek pętli nie zostaje spełniony.

**Przykład:**

```
Dim J As Integer
J = 1
Do While J < 5
    ActiveSheet.Rows(J).Cells(1).Value = J
    J = J + 1
Loop
```

# Pętla Do Until

Pętla `Do Until` („wykonuj do czasu, gdy”) jest wykonywana do czasu spełnienia wskazanego warunku. Można ją na przykład wykorzystać do modyfikowania kolejnych komórek aż do napotkania komórki pustej.

Działanie pętli nie różni się zasadniczo od pętli `Do ↪ While`. Różnica polega na tym, że o ile `Do While` jest wykonywana, gdy warunek wejścia jest spełniony, powtarzanie pętli `Do Until` trwa tak długo, jak długo wartość warunku wynosi `False`.

Wyróżniamy cztery elementy pętli: instrukcję inicjującą pętlę `Do`, warunek `Until`, na którego spełnienie oczekujemy, blok powtarzanych instrukcji i kończącą instrukcję `Loop`.

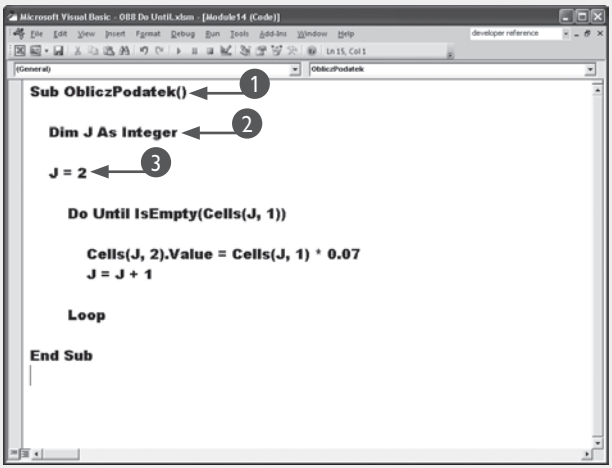
Jeżeli warunek `Until` zostanie umieszczony w wierszu `Do`, będzie sprawdzany przed każdym wykonaniem bloku. Oznacza to, że jeżeli jego obliczenie od razu doprowadzi do wartości `False`, pętla nie zostanie wykonana ani razu.

Można również umieścić warunek na końcu pętli; wówczas wykonanie bloku instrukcji będzie poprzedzone jego sprawdzeniem. Zapewni to co najmniej jednorazowe wykonanie pętli. Kolejne powtórzenia nastąpią tylko wtedy, gdy wyrażenie warunku zwróci `False`.

## Pętla Do Until

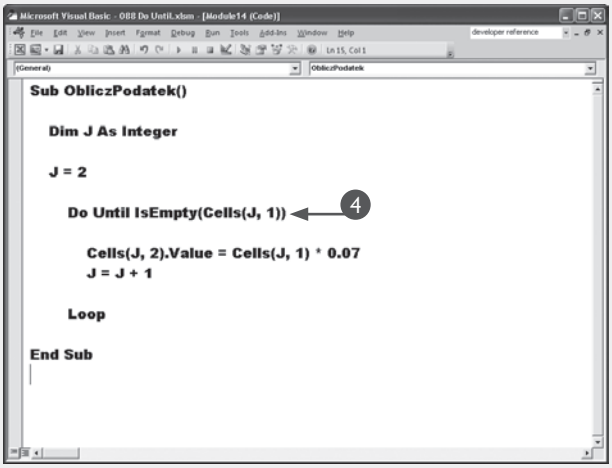
1. Utwórz nową procedurę.
2. Zadeklaruj zmienną.
3. Zainicjuj zmienną.

   W przedstawionym przykładzie zmiennej J przypisywana jest wartość 2, aby dalej przechowywać w niej numer wiersza.




4. W pierwszym wierszu pętli nastąpi sprawdzenie warunku.

   W tym przykładzie sprawdzana jest zawartość wskazywanych wartością zmiennej J komórek arkusza; blok instrukcji powtarzany jest do momentu natrafienia na komórkę pustą.



5. Wprowadź blok instrukcji.

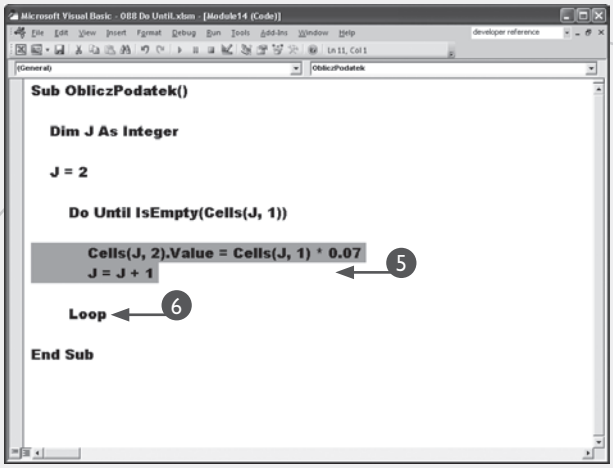
   W przedstawionym przykładzie zawartość komórki w kolumnie A mnożona jest przez 0,07; wynik umieszczany jest w kolumnie B.

6. Zamknij pętlę słowem kluczowym `Loop`.

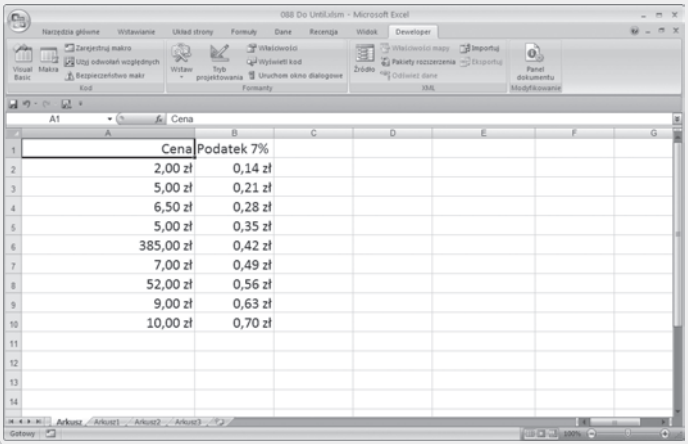
   Program powróci do początku pętli i będzie powtarzać blok instrukcji, dopóki warunek wejścia nie zostanie spełniony.

7. Wciśnij kombinację klawiszy *Alt+F11*, aby powrócić do Excela i uruchomić kod.

**Uwaga:** O uruchamianiu procedur piszemy w rozdziale 1.




Procedura umieści w kolumnie B wartości reprezentujące 7 procent wartości w kolumnie A.



| | A | B |
|---|---|---|
| 1 | Cena | Podatek 7% |
| 2 | 2,00 zł | 0,14 zł |
| 3 | 5,00 zł | 0,21 zł |
| 4 | 6,50 zł | 0,28 zł |
| 5 | 5,00 zł | 0,35 zł |
| 6 | 385,00 zł | 0,42 zł |
| 7 | 7,00 zł | 0,49 zł |
| 8 | 52,00 zł | 0,56 zł |
| 9 | 9,00 zł | 0,63 zł |
| 10 | 10,00 zł | 0,70 zł |

## Wskazówka

Podczas pracy z pętlami niejednokrotnie pojawiają się sytuacje, gdy wygodniejsze jest natychmiastowe przerwanie wykonywania pętli i przejście do dalszych instrukcji, niezależnie od stanu warunku `While` czy `Until`. Umożliwia to instrukcja `Exit Do`. Można ją umieścić w dowolnym miejscu bloku instrukcji w pętli. Można stosować wiele takich instrukcji. Efektem `Exit Do` jest zawsze przejście do wykonywania pierwszej instrukcji następującej po `Loop`.

`Exit Do` towarzyszy najczęściej instrukcja warunkowa, taka jak `If Then`. Sprawdza ona pewien specjalny warunek i w przypadku jego spełnienia inicjuje natychmiastowe wyjście z pętli.

**Przykład:**

```
Do While Wyrażenie1 = True
    If Wyrażenie2 = True
        Exit Do
    End If
Loop
```

# Pętla For Next

Pętla `For Next` („dla wartości …”) służy do powtarzania bloku instrukcji ściśle określoną ilość razy. Można jej użyć na przykład do wypełnienia pewnej liczby komórek arkusza.

Pętla `For Next` zawsze korzysta ze zmiennej licznika. Instrukcje umieszczone pomiędzy wierszami `For` i `Next` są powtarzane do czasu osiągnięcia przez zmienną licznika wartości maksymalnej. Gdy wartość licznika zwiększa się, wykonywana jest instrukcja następująca po `Next`.

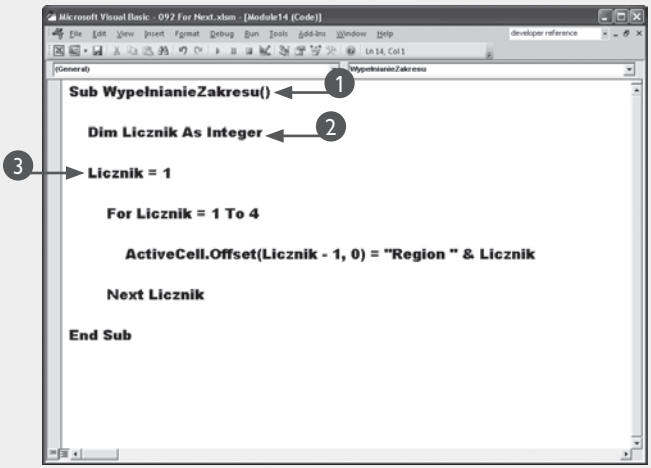
Pętla `For Next` składa się z trzech części. Pierwszą jest wiersz `For`, w którym określane są parametry pętli: zmienna licznika oraz jej wartości początkowa i maksymalna, na przykład `X = 1 To 5`. Potem następuje ciąg instrukcji i zakończenie pętli, czyli słowo kluczowe `Next`.

Wykonywanie pętli rozpoczyna się od sprawdzenia, czy wartość licznika nie przekracza zdefiniowanego maksimum. Licznik to wartość liczbowa standardowo zwiększana o 1. Pętla jest wykonywana tak długo, jak długo wartość licznika jest mniejsza od wartości maksymalnej lub jej równa. Pętla nie zostanie wykonana ani razu tylko wtedy, gdy wartość początkowa licznika jest większa od maksymalnej.

## Pętla For Next

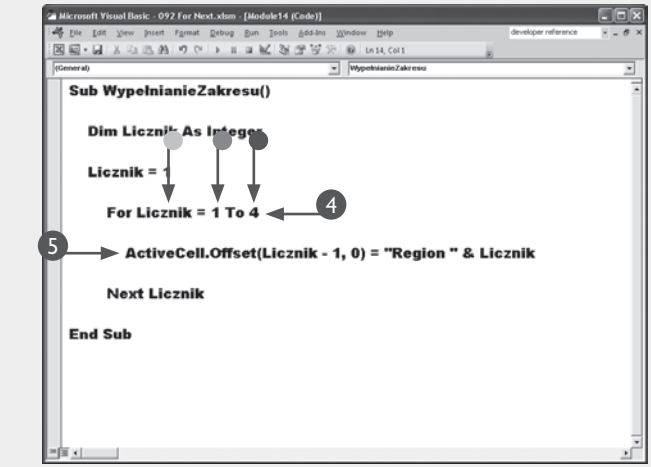
1. Utwórz nową procedurę.
2. Zadeklaruj zmienną.
3. Zainicjuj zmienną.

   W przedstawionym przykładzie licznikiem jest zmienna `Licznik`.




4. W pierwszym wierszu pętli pokazane są parametry decydujące o jej pracy.
   - Zmienna licznika.
   - Wartość początkowa.
   - Wartość maksymalna.
5. Wprowadź blok instrukcji.

   W przedstawionym przykładzie w czterech kolejnych komórkach zapisane są ciągi od „Region 1” do „Region 4”.

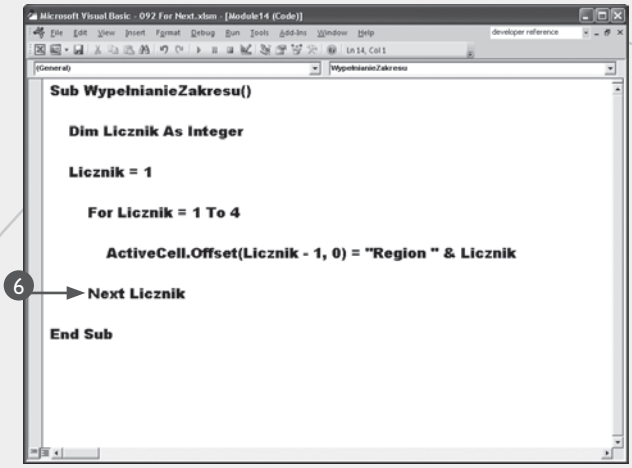


6. Zamknij pętlę słowem kluczowym `Next`.

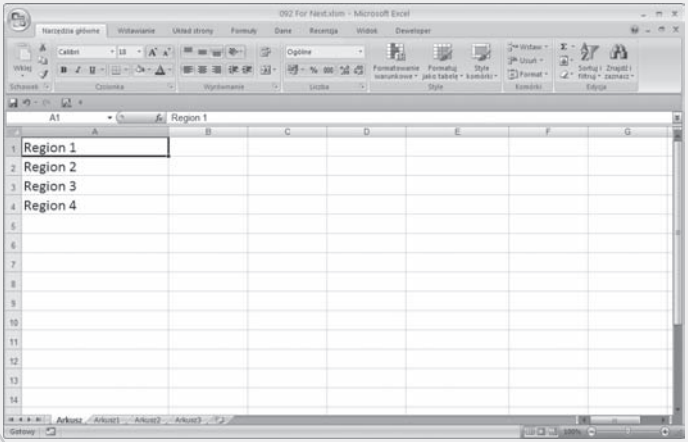
   Program powróci do początku pętli i będzie powtarzać blok instrukcji, dopóki warunek przekroczenia wartości maksymalnej nie zostanie spełniony.

7. Wciśnij kombinację klawiszy *Alt+F11*, aby powrócić do Excela i uruchomić kod.

**Uwaga:** O uruchamianiu procedur piszemy w rozdziale 1.




Procedura umieści w kolejnych komórkach teksty od „Region 1" do „Region 4".




## Wskazówka

Wartość zmiennej licznika można zmieniać o dowolną wartość. Standardowo wynosi ona 1. Aby podać inną, należy skorzystać ze słowa kluczowego `Step` dodawanego w pierwszym wierszu pętli. Wartość zwiększająca licznik może być ujemna — osiągnie się wtedy efekt zmniejszania licznika. W poniższym przykładzie licznik zainicjowany jest wartością 2 i zwiększany o 2 do 20. Kolejne wartości zmiennej licznika `J` są sumowane w zmiennej `SumaWartości`. Pętla zostanie wykonana dziesięć razy. Gdy wartość licznika osiągnie 20, blok instrukcji przestanie być powtarzany.

**Przykład:**

```
For J = 2 To 20 Step 2
    SumaWartości = SumaWartości + J
Next
```

# Pętla For Each In

Pętla `For Each In` („dla każdego”) pozwala wykonać blok instrukcji dla każdego elementu tablicy lub każdego obiektu kolekcji. Nie ma tu zmiennej licznika ani sprawdzanego warunku. O liczbie powtórzeń decyduje liczność zbioru elementów tablicy lub kolekcji. Zakończenie następuje po zakończeniu przetwarzania ostatniego elementu. Oto składnia pętli:

```
For Each element In grupa
[instrukcje]
Next [element]
```

Wyróżnia się trzy części pętli: pierwszy wiersz `For Each ↪ element In grupa` inicjuje pętlę; *`element`* to zmienna używana do przechowywania pojedynczego elementu w trakcie wykonywania bloku instrukcji; *`grupa`* to nazwa tablicy lub kolekcji. Dalej następuje ciąg powtarzanych instrukcji i kończąca pętlę instrukcja `Next`.

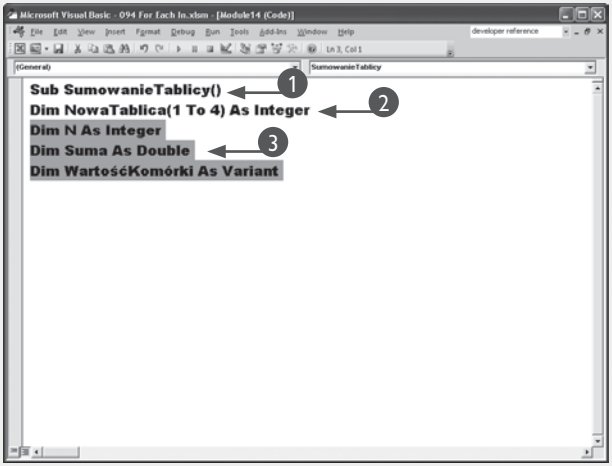
Jeśli pętla przetwarza kolejne elementy tablicy, zmienna używana do przechowywania pojedynczych elementów musi być typu `Variant`. W przypadku kolekcji do wyboru są typy: `Variant`, ogólny `Object` lub dowolny obiektowy.

## Pętla For Each In

1. Utwórz nową procedurę.
2. Zadeklaruj tablicę.

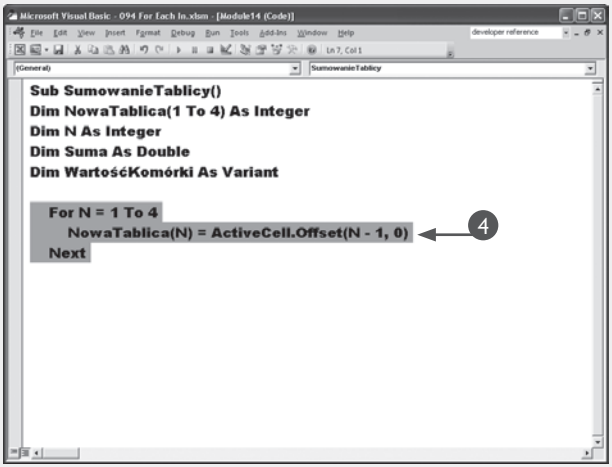
**Uwaga:** O tablicach pisaliśmy w rozdziale 5.

3. Zadeklaruj zmienne.




4. Przypisz wartości elementom tablicy.

   W tym przypadku będą to wartości komórki aktywnej i trzech kolejnych komórek.

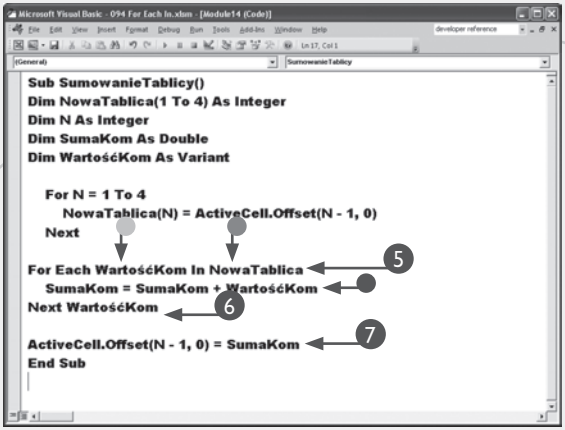


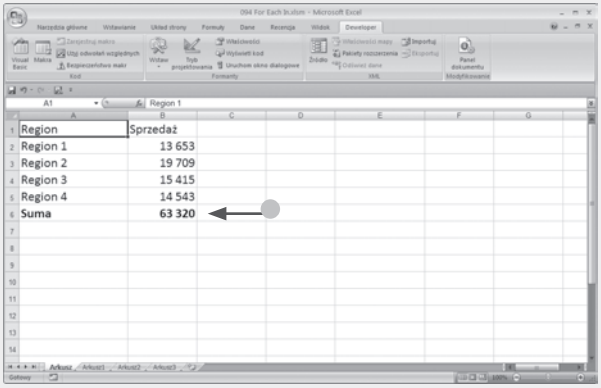
5. Wprowadź instrukcję `For Each In`.
   - Zmienna przechowująca pojedynczy element.
   - Nazwa tablicy lub kolekcji.
   - Powtarzane instrukcje.
6. Zamknij pętlę słowem kluczowym `Next`.

   W tym przypadku sumowane są elementy tablicy.
7. Wprowadź instrukcje do wykonania po wyjściu z pętli.
8. Wciśnij kombinację klawiszy *Alt+F11*, aby powrócić do Excela i uruchomić kod.

**Uwaga:** O uruchamianiu procedur piszemy w rozdziale 1.

- W przedstawionym przykładzie sumowane są elementy tablicy; wynik umieszczany jest w kolejnej komórce arkusza.




## Wskazówka

Pętle można zagnieżdżać. Pozwala to między innymi łatwo wypełniać liczbami tablice wielowymiarowe. O zagnieżdżaniu mówi się wtedy, gdy umieszcza się kolejną pętlę w bloku powtarzanych instrukcji innej pętli. Przy operacjach na tablicach wielowymiarowych należy utworzyć tyle kolejnych pętli, ile wymiarów ma tablica. W poniższym przykładzie wykorzystane są dwie pętle `For Next`. Zwróć uwagę, że w każdym przebiegu pętli sterowanej zmienną K (pętli zewnętrznej) wykonywany jest pełny cykl powtórzeń pętli sterowanej zmienną L (pętli wewnętrznej). W każdym przypadku zagnieżdżania pętli wyjście z pętli wewnętrznej przed przejściem do kolejnego powtórzenia pętli zewnętrznej jest bezwzględnie wymagane.

**KOD DO WPROWADZENIA:**

```
Sub WypełnijTablicę()
Dim NowaTablica(1 To 3, 1 To 3)
➥ As Integer
Dim K As Integer
Dim L As Integer
X = 1
For K = 1 To 3
    For L = 1 To 3
        NowaTablica(K, L) = X
        X = X + 1
    Next L
Next K
End Sub
```

**WYNIK**

Przedstawiony kod utworzy dwuwymiarową tablicę o następujących wartościach:

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

# Instrukcja If Then Else

Instrukcja `If Then Else` („jeżeli… to… w przeciwnym razie…") pozwala określić grupę instrukcji jako wykonywane warunkowo. W tym podrozdziale przedstawiony jest przykład obliczania premii dla sprzedawców, którzy osiągnęli wynik powyżej 50 000 zł. Gdy warunek przekroczenia tego progu nie jest spełniony, drukowany jest tekst „bez premii". Sprawdzanie warunku umożliwi instrukcja `If Then Else`. Ma ona składnię:

```
If warunek Then
    [instrukcje]
Else
    [instrukcje]
End If
```

Kluczowym elementem instrukcji jest warunek — dowolne wyrażenie, którego wartość to `True` lub `False`. W podanym przykładzie jest to wyrażenie badające wartość zmiennej `Sprzedaż`. Obliczenie `Sprzedaż > 50000` prowadzi do wyniku `True` lub `False` („prawda" lub „fałsz"). Jeżeli wyrażenie jest prawdziwe, wykonywany jest ciąg instrukcji między `Then` a `Else`. Jeżeli jest fałszywe, wykonywany jest blok między `Else` a `End If`. Nieokreślona wartość wyrażenia jest interpretowana jako `False`.

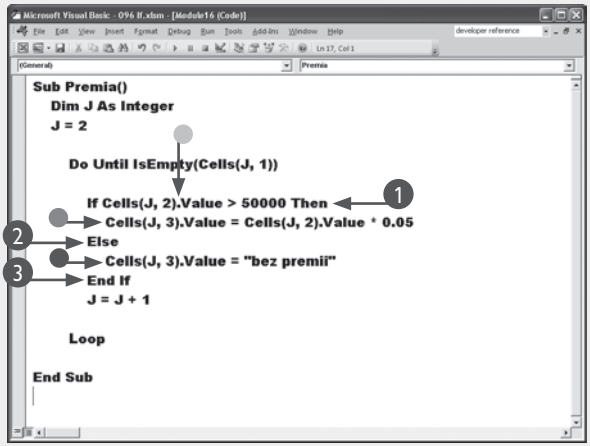
Jeżeli pojawia się potrzeba badania wielu warunków, można skorzystać z odmiany instrukcji warunkowej ze słowem `ElseIf`. Prostym przykładem może być obliczanie wartości podatku VAT, 7-procentowego dla produktu A, 22-procentowego dla produktu B i zerowego dla pozostałych. Wiersz `ElseIf warunek Then` pozwala wprowadzać kolejne bloki instrukcji przed ostatnim wierszem (i blokiem) `Else`. Wiersz i blok instrukcji `Else` są opcjonalne i można je w każdej wersji instrukcji pominąć.

## Instrukcja If Then Else

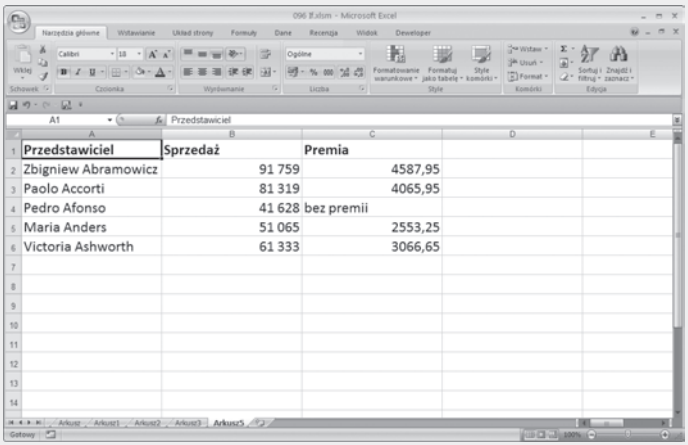
### Wersja If Then Else

1. Wprowadź instrukcję `If Then`.
   - Warunek.
   - Instrukcja do wykonania.
2. Wprowadź blok `Else`.
   - Instrukcja do wykonania.
3. Wprowadź wiersz `End If`.
4. Wciśnij kombinację klawiszy *Alt+F11*, aby powrócić do Excela i uruchomić kod.

**Uwaga:** O uruchamianiu procedur piszemy w rozdziale 1.




W przedstawionym przykładzie, jeżeli wartość w kolumnie sprzedaży jest większa od 50 000, program obliczy wartość 4-procentowej premii. W przeciwnym wypadku wstawi do arkusza tekst „bez premii".

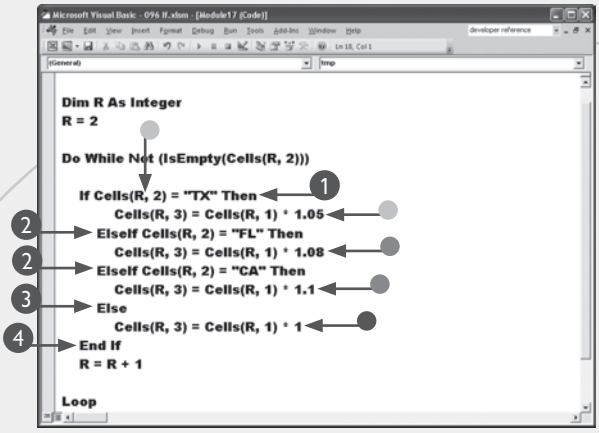


| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Przedstawiciel | Sprzedaż | Premia | | |
| 2 | Zbigniew Abramowicz | 91 759 | 4587,95 | | |
| 3 | Paolo Accorti | 81 319 | 4065,95 | | |
| 4 | Pedro Afonso | 41 628 | bez premii | | |
| 5 | Maria Anders | 51 065 | 2553,25 | | |
| 6 | Victoria Ashworth | 61 333 | 3066,65 | | |

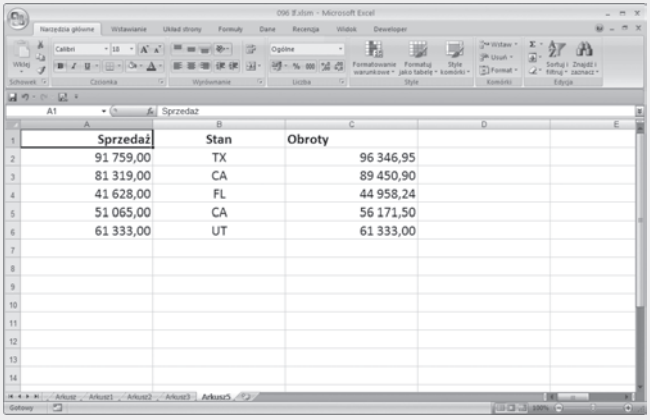
### Wersja ElseIf

1. Wprowadź instrukcję `If Then`.
   - Warunek.
   - Instrukcja do wykonania.
2. Wprowadź blok `ElseIf`.
   - Instrukcja do wykonania.
3. Wprowadź blok `Else`.
   - Instrukcja do wykonania.
4. Wprowadź wiersz `End If`.
5. Wciśnij kombinację klawiszy *Alt+F11*, aby powrócić do Excela i uruchomić kod.

**Uwaga:** O uruchamianiu procedur piszemy w rozdziale 1.




W przedstawionym przykładzie obliczana jest wartość sprzedaży brutto po doliczeniu podatku według jednej z trzech stawek.



| Sprzedaż | Stan | Obroty |
|---|---|---|
| 91 759,00 | TX | 96 346,95 |
| 81 319,00 | CA | 89 450,90 |
| 41 628,00 | FL | 44 958,24 |
| 51 065,00 | CA | 56 171,50 |
| 61 333,00 | UT | 61 333,00 |

## Wskazówka

Robienie wcięć w kodzie programu nie jest obowiązkowe, ale znacznie poprawia jego czytelność. Wcięcia uwydatniają przede wszystkim strukturę kodu, która staje się dzięki nim widoczna, jeszcze zanim zaczniesz czytać właściwą treść. Jest to szczególnie ważne, gdy korzystasz z instrukcji wykonania warunkowego i pętli, gdzie przetwarzanie nie przebiega sekwencyjnie. Poniżej przedstawiony został przykład wcięcia treści pętli `For Next`, ułatwiający odnalezienie jej zakończenia. Wcięta została również instrukcja `If Then`.

**Przykład:**

```
For I = 1 To 5
      If J < 10 Then
         J = J + 1
      End If
Next
```

Jeżeli blok instrukcji wykonywanych po Then zawiera tylko jedno polecenie, możesz wprowadzić całą konstrukcję `If` w jednym wierszu i nie wpisywać `End If`.

**Przykład:**

```
If Suma < 10 Then Suma = Suma + 1
'jest równoważne:
If Suma < 10 Then
    Suma = Suma + 1
End If
```

# Instrukcja Select Case

Instrukcja `Select Case` („wybierz przypadek”) to dalsze rozwinięcie instrukcji wykonania warunkowego, odpowiadające zastosowaniu wielu bloków `ElseIf`. Korzysta się z niej, gdy pojawia się potrzeba dokonania wyboru pomiędzy kilkoma instrukcjami na podstawie wartości komórki lub zmiennej. Ponownie wykorzystany zostanie przykład naliczania jednej z trzech stawek podatku VAT. Oto składnia instrukcji:

```
Select Case wyrażenie
[Case lista_wyrażeń –n
    [instrukcje-n]]
[Case Else
    [instrukcje]]
End Select
```

W wierszu `Select Case` podaje się wyrażenie, którego wartość będzie badana w kolejnych wierszach `Case`. Każdy wiersz `Case` zawiera wartość, która zostanie porównana ze wskazanym wyrażeniem. Po nim następuje blok instrukcji wykonywanych, gdy wynik porównania będzie pozytywny, na przykład:

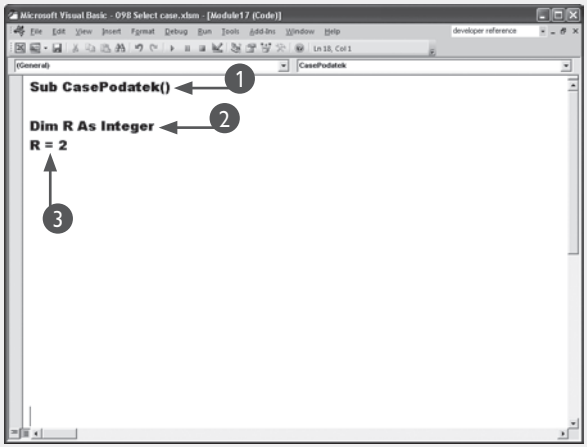
```
Select Case Wartość
    Case 4
        Instrukcje
...
End Select
```

Powyższy przykład sprawdza, czy `Wartość = 4`. Instrukcje zostaną wykonane tylko wtedy, gdy warunek ten jest spełniony (`Wartość = 4` zwraca `True`). Instrukcję kończy wiersz `End Select`.

Dodatkowo można użyć bloku `Case Else`, który zostanie wykonany tylko wtedy, gdy żadne z wcześniejszych porównań `Case` nie doprowadzi do uzyskania wyniku „prawda”.

## Instrukcja Select Case

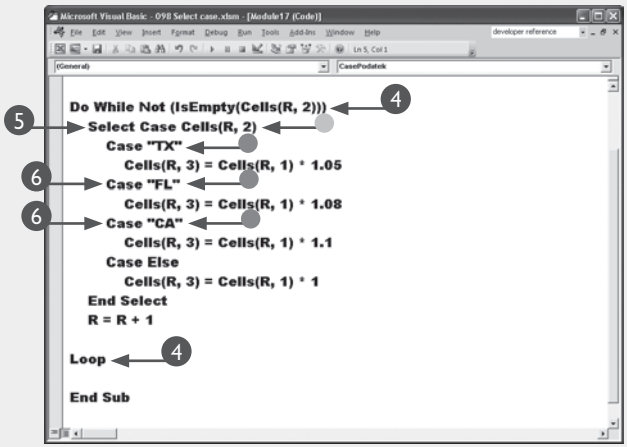
1. Utwórz nową procedurę.
2. Zadeklaruj zmienną.
3. Zainicjuj zmienną.




4. Utwórz pętlę `Do While`.

**Uwaga:** O pętlach `Do While` pisaliśmy w podrozdziale „Pętla Do While”.

5. Wprowadź instrukcję `Select Case`.
   - Wartość, która posłuży do dalszych porównań z wartościami w wierszach `Case`.
6. Wprowadź bloki `Case`.
   - Jeżeli wartość podana w wierszu `Select Case` jest równa wartości podanej w wierszu `Case`, zostaną wykonane instrukcje w danym bloku.



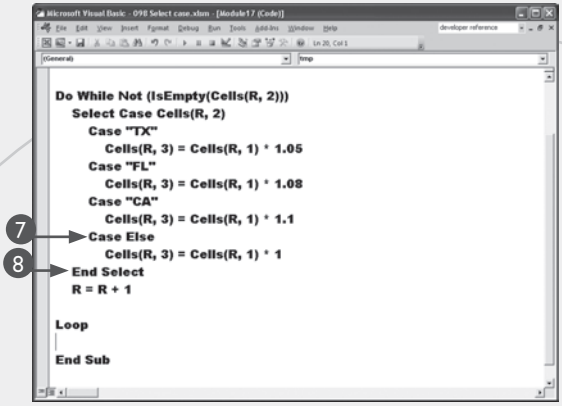
7. Wprowadź blok `Case Else`.

   Instrukcje w tym bloku zostaną wykonane, gdy żaden ze sprawdzanych wcześniej warunków nie zostanie spełniony.

8. Wprowadź instrukcję `End Select`.

9. Wciśnij kombinację klawiszy *Alt+F11*, aby powrócić do Excela i uruchomić kod.

**Uwaga:** O uruchamianiu procedur piszemy w rozdziale 1.

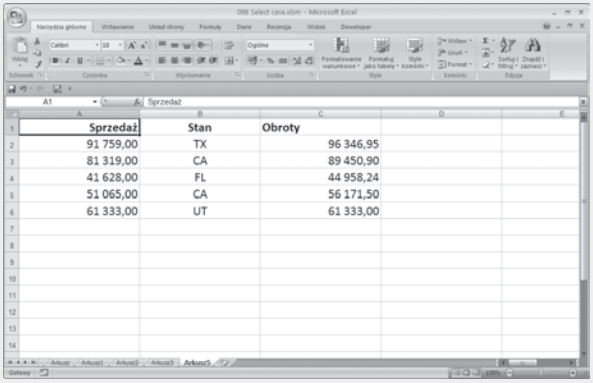


```
Do While Not (IsEmpty(Cells(R, 2)))
   Select Case Cells(R, 2)
      Case "TX"
         Cells(R, 3) = Cells(R, 1) * 1.05
      Case "FL"
         Cells(R, 3) = Cells(R, 1) * 1.08
      Case "CA"
         Cells(R, 3) = Cells(R, 1) * 1.1
      Case Else
         Cells(R, 3) = Cells(R, 1) * 1
   End Select
   R = R + 1

Loop

End Sub
```

W tym przykładzie obliczana jest wartość sprzedaży brutto dla towarów o różnej stawce podatku VAT.



| Sprzedaż | Stan | Obroty |
|---|---|---|
| 91 759,00 | TX | 96 346,95 |
| 81 319,00 | CA | 89 450,90 |
| 41 628,00 | FL | 44 958,24 |
| 51 065,00 | CA | 56 171,50 |
| 61 333,00 | UT | 61 333,00 |

## Zastosuj to

W instrukcjach `Select Case` można również korzystać ze specjalnych wyrażeń umożliwiających sprawdzenie, czy wartość należy do pewnej grupy.

**KOD DO WPROWADZENIA:**

```
Select Case LiczbaOperacji
    Case 1 To 5
        Prowizja = Suma * .05
    Case 6 To 15
        Prowizja = Suma * .1
End Select
```

**WYNIK:**

Instrukcja `Select Case` sprawdzi, czy wartość `LiczbaOperacji` należy do jednego z dwóch zakresów.

**KOD DO WPROWADZENIA:**

```
Select Case LiczbaStudentów
    Case Is < 10
        MsgBox("Zbyt mała liczba
➥ zapisanych studentów.")
End Select
```

**WYNIK:**

Instrukcja `Select Case` sprawdzi, czy wartość `LiczbaStudentów` jest mniejsza od 10. Jeżeli tak, wyświetli się ostrzegawcze okienko dialogowe.

**KOD DO WPROWADZENIA:**

```
Select Case Stan
    Case "TX", "CA"
        Suma = Suma * 1.085
End Select
```

**WYNIK:**

Instrukcja `Select Case` sprawdzi, czy wartość `Stan` to `TX` lub `CA`. Jeżeli tak, suma zostanie zwiększona o podatek 8,5%.

# Instrukcja skoku GoTo

Instrukcja `GoTo` („idź do”) pozwala natychmiast przejść do innego miejsca w procedurze. Przed jej użyciem dane miejsce docelowe musi zostać oznaczone etykietą. Etykieta to nazwa i znak dwukropka (`:`). Instrukcją wykonywaną po instrukcji `GoTo` jest zawsze pierwsza instrukcja po etykiecie. Oto składnia polecenia `GoTo`:

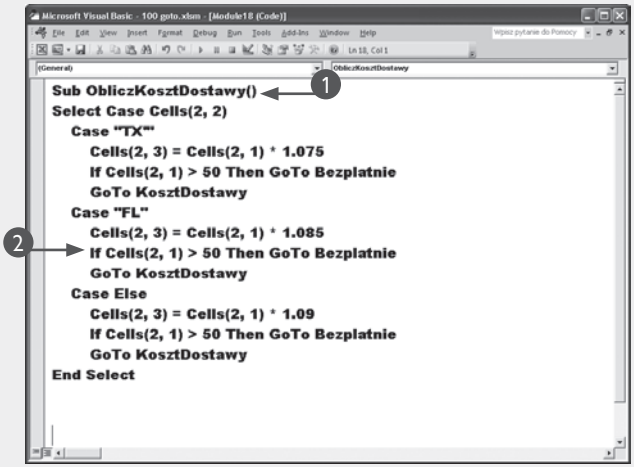
`GoTo` *`etykieta`*

Jak widać, nie jest skomplikowana i obejmuje słowo kluczowe `GoTo` oraz etykietę. Etykieta może zostać umieszczona w dowolnym miejscu w tej samej procedurze. Instrukcja `GoTo` nie pozwala przejść do innej procedury nawet w tym samym module. Można użyć dowolnej liczby instrukcji `GoTo` i dowolnej liczby etykiet, a różne instrukcje mogą korzystać z tych samych lub innych etykiet.

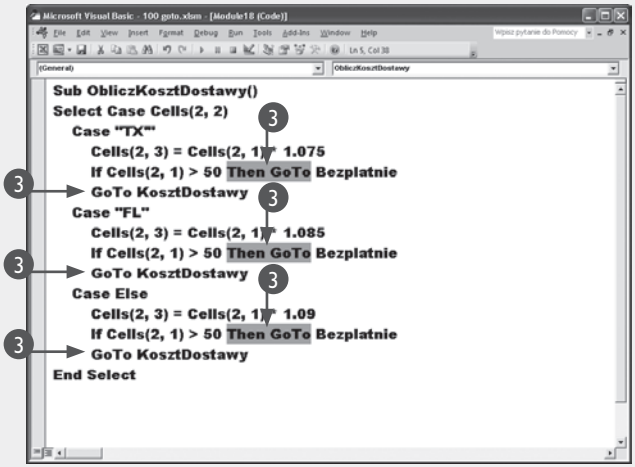
Jedną z podstawowych zasad dobrego stylu programowania jest korzystanie z instrukcji `GoTo` wyłącznie wtedy, gdy nie można osiągnąć pożądanego wyniku przy użyciu instrukcji warunkowych i pętli. Jest to w zasadzie relikt z czasów, kiedy każda instrukcja kodu miała swój numer, a zasób dostępnych instrukcji sterujących był ograniczony. Choć w języku VBA instrukcje `GoTo` wykorzystuje się powszechnie do obsługi błędów, warto wiedzieć, że dla większości programistów korzystanie z nich to dowód braku umiejętności pisania poprawnych programów. Do tematu wykorzystywania instrukcji `GoTo` w debugowaniu kodu powrócimy w rozdziale 8.

## Instrukcja skoku GoTo

1. Utwórz nową procedurę.
2. Wprowadź kod.

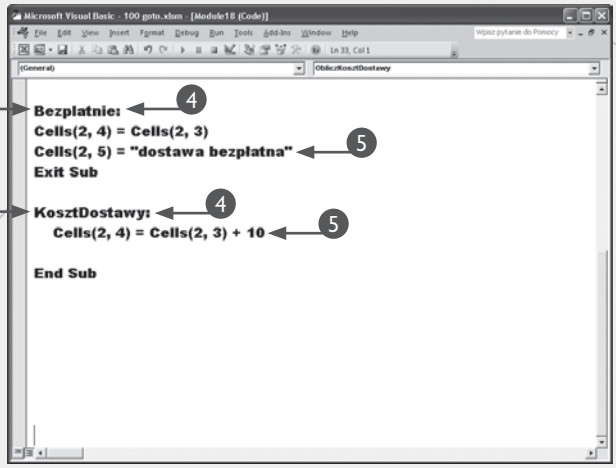



3. Wprowadź instrukcję `GoTo`.

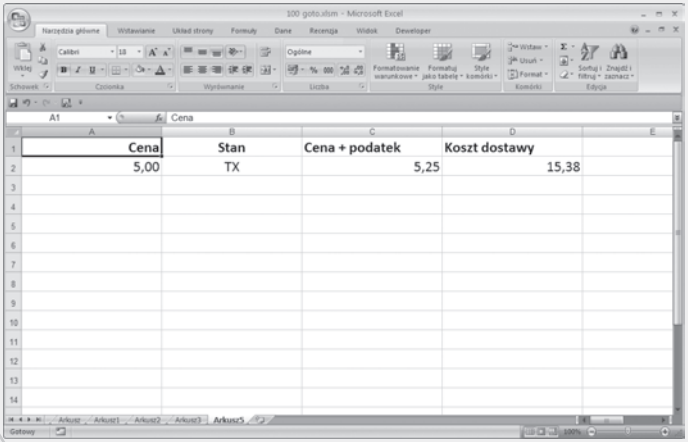


4. Wstaw etykietę.
   - Po nazwie wstaw dwukropek.
5. Wprowadź dalszy kod.
6. Wciśnij kombinację klawiszy *Alt+F11*, aby powrócić do Excela i uruchomić kod.

**Uwaga:** O uruchamianiu procedur piszemy w rozdziale 1.




W tym przykładzie obliczany jest koszt dostawy, o ile po doliczeniu podatku wartość zakupu jest mniejsza niż 50 zł.




## Wskazówka

Etykiety to oznaczenia wybranych miejsc w kodzie. Same nie mają żadnego wpływu na przebieg programu. Nie są w najmniejszym stopniu podobne do instrukcji warunkowych lub pętli, które operują pewnym zamkniętym blokiem kodu.

W przypadku stosowania wielu etykiet pojawia się problem pomijania kodu w kolejnych oznaczonych nimi blokach. Należy wtedy użyć jeszcze jednej instrukcji `GoTo` lub instrukcji `Exit Sub`, która przerywa wykonywanie procedury.

W poniższym przykładzie użyta została instrukcja `Exit Sub` bezpośrednio przed etykietą, aby uniknąć wykonania instrukcji `T = 50`. Ma ona zostać wykonana wyłącznie po skoku do etykiety `ZwiększWartość`. Sam skok wyróżnia nietypową sytuację, w której niepożądana jest operacja `T = T * 5`.

**Przykład:**

```
Sub Test Goto()
Dim T As Integer
T = Cells(1,1)
If T < 5 Then GoTo ZwiększWartość
    T = T * 5
Exit Sub
ZwiększWartość:
End Sub
```

# Wywoływanie procedur

Do wywoływania innych procedur służy instrukcja `Call`. Aby jej użyć, wprowadza się słowo `Call`, nazwę procedury i ujętą w nawiasy listę parametrów. Po jej wykonaniu przetwarzana jest pierwsza instrukcja kodu procedury wywołanej. Po zakończeniu jej wykonywania program przechodzi do pierwszej instrukcji po instrukcji `Call` w procedurze wywołującej.

Wywoływanie procedur często poprzedza się sprawdzeniem warunku, na przykład instrukcją `If Then`. Powoduje to efekt wykonania określonych w wywoływanej procedurze operacji tylko w przypadku, gdy określony w instrukcji `If Then` warunek jest spełniony. Kod w kolejnych wierszach jest wykonywany zawsze. Oczywiście można go pominąć, wstawiając polecenie `Exit Sub`.

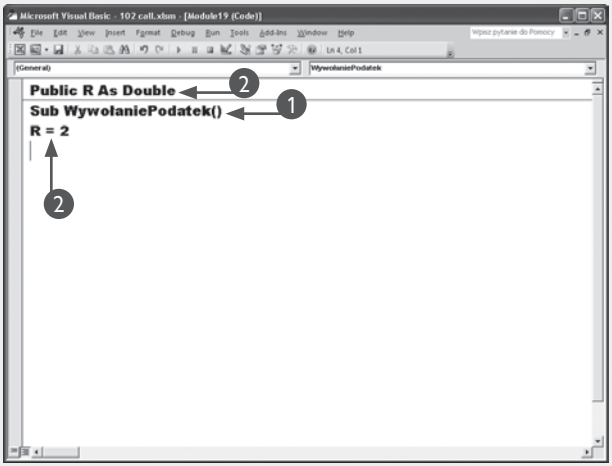
Słowo kluczowe `Call` nie jest wymagane. Można użyć samej tylko nazwy procedury. Wówczas lista argumentów nie może być umieszczana w nawiasach. Kod wywoływany w ten sposób może być zapisany w procedurach `Sub`, `Function` lub bibliotece DLL.

## Wywoływanie procedur

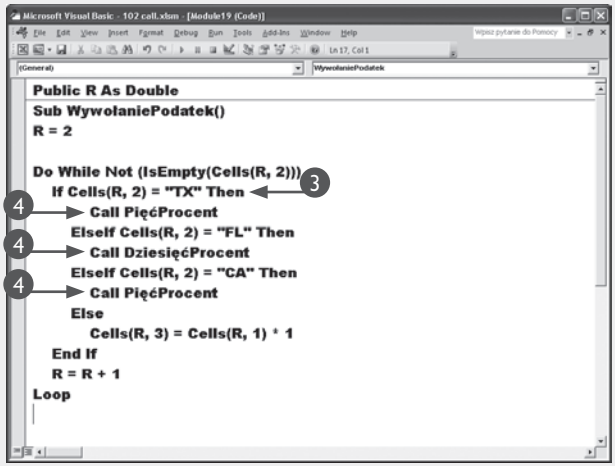
1. Utwórz nową procedurę.
2. Zadeklaruj i zainicjuj zmienne.

   Mogą być potrzebne zmienne publiczne.

**Uwaga:** O zmiennych publicznych pisaliśmy w rozdziale 3.

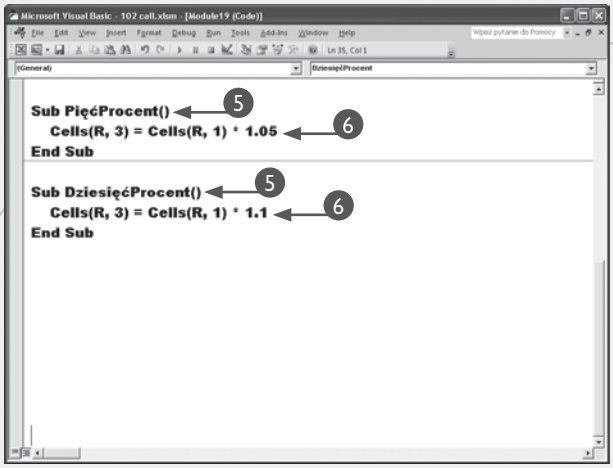



3. Utwórz warunek `If Then`.
4. Wywołaj procedurę.

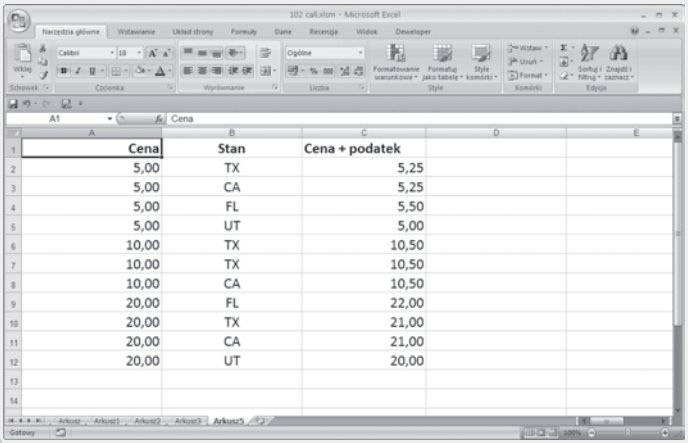


5. Utwórz procedury wywoływane.
6. Wprowadź kod, który będzie wykonywany w wywołaniach.
7. Wciśnij kombinację klawiszy *Alt+F11*, aby powrócić do Excela i uruchomić kod.

**Uwaga:** O uruchamianiu procedur piszemy w rozdziale 1.




Gdy warunek zostanie spełniony, instrukcja `If Then` wywoła procedurę.



| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Cena | Stan | Cena + podatek | | |
| 2 | 5,00 | TX | 5,25 | | |
| 3 | 5,00 | CA | 5,25 | | |
| 4 | 5,00 | FL | 5,50 | | |
| 5 | 5,00 | UT | 5,00 | | |
| 6 | 10,00 | TX | 10,50 | | |
| 7 | 10,00 | TX | 10,50 | | |
| 8 | 10,00 | CA | 10,50 | | |
| 9 | 20,00 | FL | 22,00 | | |
| 10 | 20,00 | TX | 21,00 | | |
| 11 | 20,00 | CA | 21,00 | | |
| 12 | 20,00 | UT | 20,00 | | |
| 13 | | | | | |
| 14 | | | | | |

## Zastosuj to

Korzystanie ze słowa kluczowego `Call` jest opcjonalne. Jego stosowanie pozwala wyróżnić wywołania procedur zdefiniowanych jako funkcje i procedury programu (w odróżnieniu od funkcji standardowych i dołączonych w bibliotekach). Gdy `Call` zostanie pominięte, trzeba również opuścić nawiasy otaczające listę argumentów:

| TA INSTRUKCJA: | JEST RÓWNOWAŻNA Z: |
|---|---|
| `Call NowaProcedura(zmienna1, zmienna2)` | `NowaProcedura zmienna1, zmienna2` |