

## IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

## KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

## TWÓJ KOSZYK

DODAJ DO KOSZYKA

## CENNIK I INFORMACJE

ZAMÓW INFORMACJE  
O NOWOŚCIACH

ZAMÓW CENNIK

## CZYTELNIA

FRAGMENTY KSIĄŻEK ONLINE

# VBA dla Excela 2003/2007. Leksykon kieszonkowy

Autor: Mirosław Lewandowski

ISBN: 978-83-246-1221-5

Format: B6, stron: około 150



### Podręczna ściągą dla tych, którzy chcą usprawnić działanie Excela

- Opisy elementów VBA
- Zasady tworzenia makr
- Projektowanie formularzy

Możliwości najpopularniejszego na rynku arkusza kalkulacyjnego – Excela – są ogromne. Jednak czasem, szczególnie podczas wykonywania mniej typowych zadań, okazują się niewystarczające. Niekiedy też sekwencja czynności prowadzących do zrealizowania konkretnego zadania jest złożona i skomplikowana. Na szczęście, istnieje sposób na przyspieszenie i zautomatyzowanie pracy z Excelem – są nim makra, czyli programy tworzone za pomocą języka Visual Basic for Applications (VBA), bazujące na funkcjach Excela.

Książka „Tworzenie makr w VBA dla Excela 2003/2007. Leksykon kieszonkowy” to podręczny zbiór wiadomości dotyczących VBA. Znajdziesz w niej opis elementów języka VBA i obiektów MS Office oraz omówienie zasad budowania makr. Nauczysz się implementować mechanizmy przechwytywania zdarzeń dla obiektów, a także dowiesz się, jak tworzyć formularze i procedury ich obsługi.

- Stałe i zmienne
- Deklarowanie tablic
- Konwersja typów danych
- Obiekty i metody
- Przeglądarka obiektów
- Operacje na łańcuchach tekstowych
- Obsługa zdarzeń arkuszy i skoroszytów
- Formularze

**Wykorzystaj pełnię możliwości Excela**



---

# Spis treści

<b>Wstęp .....</b>	<b>5</b>
<b>1. Stałe, zmienne i tablice .....</b>	<b>6</b>
Deklarowanie zmiennych i stałych	6
Deklarowanie procedur i tablic	8
Typy zmiennych	10
Opcje modułu	13
Konwersja typów danych	15
<b>2. Obiekty i metody .....</b>	<b>21</b>
Metody	22
Przeglądarka obiektów	55
Obiekty	58
<b>3. Elementy języka Visual Basic .....</b>	<b>79</b>
Funkcje i operatory matematyczne	79
Data i czas	84
Interakcja z użytkownikiem	93
Operacje na łańcuchach	97
Pętle i skoki	106
Instrukcje warunkowe i wyboru	109
Przerwanie programu	114
Funkcje informacyjne	115
Błędy	119

<b>4. Procedury zdarzeniowe .....</b>	<b>122</b>
Procedury zdarzeniowe dla obiektu Worksheet	122
Procedury zdarzeniowe dla obiektu ThisWorkbook	124
Zdarzenia dla innych obiektów	130
<b>5. Formularze .....</b>	<b>131</b>
Procedury zdarzeniowe formantów	131
Właściwości formantów formularza	143
<b>Skorowidz .....</b>	<b>179</b>

# Rozdział 3. Elementy języka Visual Basic

## Funkcje i operatory matematyczne

### Funkcje trygonometryczne

Do wyboru mamy funkcje:

**Atn**

arcus tangens;

**Cos**

cosinus;

**Sin**

sinus;

**Tan**

tangens.

Składnia wszystkich jest taka sama:

`Funkcja(Wartość)`

Aby otrzymać wartość funkcji cotangens, należy zastosować funkcję  $1/\text{Tan}$ .

Wartość pi możesz obliczyć na dwa sposoby.

Bezpośrednio w VBA jako arcus tangens:

```
Pi = 4 * Atn(1)
```

lub korzystając z funkcji arkuszowej Pi:

```
pi = WorksheetFunction.Pi
```

### Exp i Log

Log zwraca wartość logarytmu naturalnego danej liczby. Podstawą logarytmów naturalnych jest stała  $e = 2,71828182845904$ .

**Exp** jest odwrotnością funkcji **Log** — zwraca wartość liczby e podniesioną do wskazanej potęgi.

Składnia:

**Exp**(*Wykładnik*)

*Wykładnik*

wykładnik potęgi;

**Log**(*Liczba*)

*Liczba*

liczba rzeczywista dodatnia, której logarytm należy obliczyć.

## **Sqr**

Zwraca pierwiastek kwadratowy podanego argumentu.

Składnia:

**Sqr**(*Argument*)

*Argument*

liczba rzeczywista większa od 0.

## **Randomize, Rnd**

**Randomize** służy do zainicjowania generatora liczb losowych.

Składnia:

**Randomize**(*Baza*)

*Baza* (argument opcjonalny)

wartość początkowa do obliczenia zbioru liczb pseudolosowych. Jeżeli go pominiesz, zostanie on ustalony na podstawie wskazań zegara systemowego, co dodatkowo korzystnie wpłynie na losowane liczby.

**Rnd** generuje liczbę losową z zakresu od 0 do <1.

## Składnia:

`Rnd(Liczba)`

*Liczba*

argument opcjonalny;

- jeżeli *Liczba* = 0, funkcja zwróci ostatnio wygenerowaną liczbę;
- jeżeli *Liczba* < 0, funkcja za każdym razem zwróci tę samą, raz wygenerowaną wartość;
- jeżeli pominiemy argument lub *Liczba* > 0, funkcja zwróci kolejną liczbę ze zbioru liczb losowych.

Wartość argumentów *Baza* i *Liczba* nie ma znaczenia, jeżeli zależy Ci na losowym generowaniu liczb. Jednakże za ich pomocą możesz spowodować ponowne wygenerowanie tego samego zestawu. Jeśli więc chcesz, aby liczby losowe zaczęły powtarzać się w tej samej kolejności przed zainicjowaniem generatora, wywołaj funkcję `Rnd` z parametrem ujemnym, a następnie zainicjuj generator liczb losowych. Wyjaśni to poniższy przykład:

```
Sub pseudolosowa()  
'pierwsza inicjacja  
  Rnd (-1)  
  Randomize 2  
  For a = 1 To 20  
    Cells(a, 1) = Rnd  
  Next  
'druga inicjacja  
  Rnd (-1)  
  Randomize 2  
  For a = 1 To 20  
    Cells(a, 2) = Rnd  
  Next  
End Sub
```

W przykładzie otrzymamy dwie kolumny z wygenerowanymi losowo liczbami z zakresu 0 do 1. Losowo, lecz w tej samej kolejności.

## Abs

Oblicza wartość bezwzględną (moduł) podanej liczby, czyli odcina znak minus, jeżeli występuje.

Składnia:

**Abs**(*Liczba*)

*Liczba*

dowolna liczba rzeczywista.

## Sgn

Zwraca wartość w zależności od znaku podanego argumentu.

Składnia:

**Sgn**(*Argument*)

*Argument*

dowolna liczba rzeczywista.

Funkcja zwraca następujące wartości:

- 1   gdy argument jest mniejszy od zera
- 0     gdy argument jest równy zero
- 1     gdy argument jest większy od zera

## Fix, Int

Zwracają część całkowitą argumentu.

Składnia:

**Fix**(*Argument*)

**Int**(*Argument*)

*Argument*

dowolna liczba rzeczywista.

W zakresie liczb dodatnich funkcje odcinają część ułamkową argumentu. Różnice w działaniu są widoczne podczas działań na liczbach ujemnych. **Int** zaokrągla argument w dół, podczas gdy **Fix** — w górę.

Przykład:

**Int**(3.2)  
da wynik 3

**Fix**(3.2)  
da wynik 3

**Int**(-3.2)  
da wynik -4

**Fix**(-3.2)  
da wynik -3

## Operatory matematyczne

Znak	Opis	Składnia — przykład użycia
^	Znak potęgowania	wynik = liczba^wykładnik
- +	Znaki odejmowania i dodawania	wynik = składnik + składnik
* /	Znaki mnożenia i dzielenia	wynik = dzielna/dzielnik
\	Zwraca część całkowitą z wyniku dzielenia. Dodatkowo dzielna i dzielnik przed wykonaniem obliczeń zostaną pozbawione części ułamkowej.	wynik = dzielna\dzielnik
Mod	Zwraca resztę z dzielenia.	reszta = dzielna Mod dzielnik
&	Służy do łączenia dwóch ciągów znaków.	wynik = "łańcuch1"&"łańcuch2"



## Round

Zwraca liczbę zaokrągloną do podanego miejsca po przecinku.

Składnia:

**Round** (*Liczba*, *IleMiejsc*)

*Liczba* (wymagany)

dowolna liczba rzeczywista poddana zaokrągleniu;

*IleMiejsc* (opcjonalny)

wskazuje, z jaką dokładnością (do ilu miejsc po przecinku) należy zaokrąglić liczbę. Jeżeli pominiemy ten parametr, funkcja zwróci liczbę całkowitą.

## Data i czas

### Hour, Minute, Second

Funkcje zwracają godzinę, minutę lub sekundę z podanego argumentu. Argumentem może być liczba w postaci dziesiętnej lub w formacie czasu.

Przykład:

**Hour**(0.593888889)

**Hour**(#2:15:12 PM#)

W obu powyższych przypadkach funkcja zwróci liczbę 14, obydwa argumenty przedstawiają bowiem tę samą godzinę. Analogicznie:

**Minute**(0.593888889)

da wynik 15. Z kolei:

**Second**(0.593888889)

da wynik 12.

## Day, Month, Year

**Day** zwraca liczbę o wartości od 1 do 31 reprezentującą dzień miesiąca.

**Month** zwraca liczbę w zakresie od 1 do 12 reprezentującą miesiąc z podanej daty.

**Year** zwraca rok z podanej daty.

Składnia:

```
Day(data)
Month(data)
Year(data)
```

gdzie *data* to wyrażenie reprezentujące datę.

## Weekday

Funkcja zwraca wartość liczbową (od 0 do 7) reprezentującą dzień tygodnia wskazanej daty.

Składnia:

```
Weekday(Data, 1dzieńTygodnia)
```

*Data*

wymagany;

*1dzieńTygodnia* (opcjonalny)

wskazuje pierwszy dzień tygodnia.

Przykład:

```
Weekday(data, 2)
```

zwróci wartość 1, jeżeli rozpatrywany dzień będzie poniedziałkiem.

```
Weekday(data, 4)
```

zwróci wartość 1 dla środy, 2 dla czwartku i tak dalej.

Domyślną wartością parametru **1dzieńTygodnia** jest 1 (czyli niedziela).

## **TimeSerial**

Zwraca w wyniku czas.

Składnia:

```
TimeSerial(Godzina, Minuta, Sekunda)
```

*Godzina, Minuta, Sekunda* (wymagane)  
dowolne dodatnie liczby całkowite.

Przykład:

```
TimeSerial(2, 34, 7)  
da w wyniku godzinę 2:34:07.
```

## **TimeValue**

Konwertuje ciąg znaków o ustalonej składni na zmienną zawierającą czas.

Przykład:

```
TimeValue("4:35:17 PM")  
da w wyniku zmienną typu Date wskazującą czas 16:35:17.
```

## **DateSerial**

Zwraca w wyniku datę.

Składnia:

```
DateSerial(Rok, Miesiąc, Dzień)
```

*Rok, Miesiąc, Dzień* (wymagane)  
dowolne liczby całkowite.

Przykład:

**DateSerial**(0, 4, 7)  
da w wyniku datę 07.04.2000

**DateSerial**(99, 4, 7)  
da w wyniku datę 07.04.1999

**DateSerial**(100, 4, 7)  
da w wyniku datę 07.04.100

Warto stosować pełny (czterocyfrowy) zapis roku, aby uniknąć pomyłek pokazanych powyżej.

## **DateValue**

Konwertuje ciąg znaków o ustalonej składni na zmienną typu `Date` zawierającą datę.

Przykłady:

```
DateValue("luty 3 3002")  
DateValue("3 luty 3002")
```

W powyższych poleceniach zostanie obliczona data 03.02.3002.

```
DateValue("3 2 3002")  
DateValue("3,2,3002")
```

Po wykonaniu powyższych poleceń program zwróci wartość 02.03.3002.

Poniższy zapis spowoduje błąd:

```
DateValue(3, 2, 3002)
```

VBA obsługuje daty z zakresu od 1.01.100 do 31.12.9999 i wyrażenia zawierające takie wartości mogą zostać podstawione jako argument funkcji `DateSerial`.

## **DateAdd**

Dodaje do podanej daty określony interwał czasowy.

## Składnia:

**DateAdd**(*Interwał*, *Ilość*, *Data*)

*Interwał* (wymagany)

podaje, jaki przedział czasowy zostanie dodany do daty.

## Możliwe wartości:

yyyy

rok

q kwartał

m miesiąc

y dzień roku

d dzień

w dzień tygodnia

ww tydzień

h godzina

n minuta

s sekunda

Na potrzeby funkcji **DateAdd** parametry *y*, *d* i *w* oznaczają zawsze dodanie dnia do wskazanej daty. Jednak przy innych funkcjach daty i czasu parametry te mają już różne znaczenia.

*Ilość* (wymagany)

wskazuje, ile interwałów czasowych ma być dodanych;

*Data* (wymagany)

data bazowa.

## Przykład:

```
data = DateSerial(2, 4, 7) 'tworzymy datę 07.04.2002  
nowa_data1 = DateAdd("n", 3, data)
```

```
nowa_data2 = DateAdd("d", 3, data)
nowa_data3 = DateAdd("q", 3, data)
nowa_data4 = DateAdd("ww", 3, data)
nowa_data5 = DateAdd("yyyy", 3, data)
```

W wyniku działania powyższego kodu zmienne przyjmą następujące wartości:

```
nowa_data1
2002-04-07 00:03
```

```
nowa_data2
2002-04-10
```

```
nowa_data3
2003-01-07
```

```
nowa_data4
2002-04-28
```

```
nowa_data5
2005-04-07
```

## DateDiff

Zwraca różnicę między podanymi datami.

Składnia:

```
DateDiff(Interwał, Data1, Data2, 1dzieńTygodnia,  
1TydzieńRoku)
```

*Interwał* (wymagany)  
patrz funkcja DateAdd;

*Data1*, *Data2* (wymagane)  
daty, między którymi zostanie obliczona różnica;

*1dzieńTygodnia* (opcjonalny)  
stała wskazująca początek tygodnia. Możliwe są wartości od 0 (niedziela) do 7 (sobota) lub stałe z kolekcji vbDayOfWeek.

*1TydzieńRoku* (opcjonalny)

stała wskazująca, w jaki sposób ma zostać wskazany pierwszy tydzień roku.

Możliwe wartości:

*vbUseSystem* lub 0

Używa ustawień systemowych.

*vbFirstJan1* lub 1

Pierwszy tydzień roku zawiera datę 1 stycznia.

*vbFirstFourDays* lub 2

Pierwszy tydzień roku zawiera przynajmniej cztery dni nowego roku.

*vbFirstFullWeek* lub 3

Pierwszy pełny tydzień roku.

## DatePart

Oblicza, w jakiej części interwału czasowego mieści się podana data.

Składnia:

```
DatePart(Interwał, Data, 1dzieńTygodnia, 1TydzieńRoku)
```

Parametry zostały opisane przy funkcjach DateDiff i DateAdd.

Przykład:

```
data = DateSerial(2, 4, 7) 'tworzymy datę 07.04.2002  
nowa_data1 = DatePart("w", data)  
nowa_data2 = DatePart("y", data)  
nowa_data3 = DatePart("q", data)  
nowa_data4 = DatePart("ww", data)  
nowa_data5 = DatePart("yyyy", data)
```

W wyniku działania powyższego kodu zmienne *nowa\_data* przyjmą następujące wartości:

nowa\_data1

1 — wskazana data to niedziela.

nowa\_data2

97 — wskazana data to 97. dzień roku.

nowa\_data3

2 — kwiecień jest w drugim kwartale.

nowa\_data4

15 — wskazaną datę obejmuje 15. tydzień roku.

nowa\_data5

2002 — wskazaną datę obejmuje rok 2002.

### Date, Now, Time

- **Date** zwraca dzisiejszą datę;
- **Time** zwraca aktualny czas;
- **Now** zwraca wyrażenie w postaci dzisiejszej daty i aktualnego czasu.

Wartości są obliczane na podstawie zegara systemowego.

Składnia:

zmienna = **Date**

zmienna = **Time**

zmienna = **Now**

Funkcje bezparametrowe.

### Timer

Wskazuje, ile sekund (wraz z ułamkami) upłynęło od północy.

Funkcja bezparametrowa.

Składnia:

zmienna = **Timer**



## MonthName

Podaje (po polsku!) nazwę miesiąca.

Składnia:

**MonthName**(*Numer*, *Skrócona*)

*Numer* (wymagany)

podaje numer miesiąca;

*Skrócona* (opcjonalny)

jeżeli wprowadzisz wartość `True`, to nazwa miesiąca będzie podana w formie skróconej (na przykład *mar* zamiast *marzec*). Domyślna wartość to `False`.

## WeekdayName

Podaje (po polsku) nazwę dnia tygodnia.

Składnia:

**WeekdayName**(*Dzień*, *Skrócona*, *IdzieńTygodnia*)

*Dzień* (wymagany)

numer dnia;

*Skrócona* (opcjonalny)

patrz funkcja `MonthName`;

*IdzieńTygodnia* (opcjonalny)

wskazuje pierwszy dzień tygodnia. Patrz funkcje `WeekDay` i `DateDiff`.

## Calendar

Właściwość, która zwraca lub ustawia rodzaj używanego kalendarza w Twoim projekcie.

Składnia:

**Calendar** = jaki

Możliwe są dwie wartości parametru:

`vbCalGreg` lub `0`  
kalendarz gregoriański

`vbCalHijri` lub `1`  
Hidżra — kalendarz księżycowy używany w krajach islamskich

## Interakcja z użytkownikiem

### **MsgBox**

Wyświetla okno komunikatu. Może także służyć do pobierania danych od użytkownika.

Składnia:

**MsgBox**(*Tekst*, *Przyciski*, *Tytuł*, *PlikPomocy*, *Kontekst*)

*Tekst* (wymagany)

komunikat, który zostanie wyświetlony; może nim być ciąg do 1024 znaków lub zmienna;

*Przyciski* (opcjonalny)

zawiera informację o tym, jakie przyciski będą wyświetlone w oknie oraz jaki będzie typ komunikatu. Z typem komunikatu wiążą się wyświetlana w oknie ikona i efekty dźwiękowe (jeżeli użytkownik z nich korzysta).

Wartości przycisków okna:

`vbOKOnly` lub `0`  
wartość domyślna — tylko przycisk *OK*

`vbOKCancel` lub `1`  
przyciski *OK* i *Anuluj*

`vbAbortRetryIgnore` lub 2  
przyciski: *Przerwij, Ponów próbę, Ignoruj*

`vbYesNoCancel` lub 3  
*Tak, Nie, Anuluj*

`vbYesNo` lub 4  
*Tak, Nie*

`vbRetryCancel` lub 5  
*Ponów próbę, Anuluj*

`vbMsgBoxHelpButton` lub 16384  
dodatkowo przycisk *Pomoc*

Wartości typu komunikatu:

`vbCritical` lub 16  
zatrzymanie krytyczne

`vbQuestion` lub 32  
pytanie

`vbExclamation` lub 48  
ostrzeżenie

`vbInformation` lub 64  
informacja

`vbMsgBoxRight` lub 524288  
tekst jest wyrównany do prawej

`vbMsgBoxRtlReading` lub 1048576  
arabski układ okna (od prawej do lewej)

Odpowiednią wartość parametru *Przyciski* oblicza się przez dodanie do siebie wartości stałych (można podać składniki rozdzielone znakiem + lub ich sumę) albo podanie ich nazw rozdzielonych znakiem +.

*Tytuł* (opcjonalny)

komunikat, który będzie widoczny na pasku tytułu (jeżeli go pominiesz, zostanie tam wyświetlona nazwa „Microsoft Excel”);

*PlikPomocy, Kontekst*

plik pomocy i miejsce w nim, do którego prowadzić będzie łącze po kliknięciu przycisku *Pomoc*.

Funkcja **MsgBox** może zwrócić wartości w zależności od działania podjętego przez użytkownika:

vbOK lub 1

kliknięto przycisk *OK*

vbCancel lub 2

kliknięto przycisk *Anuluj*

vbAbort lub 3

kliknięto przycisk *Przerwij*

vbRetry lub 4

kliknięto przycisk *Ponów Próbę*

vbIgnore lub 5

kliknięto przycisk *Ignoruj*

vbYes lub 6

kliknięto przycisk *Tak*

vbNo lub 7

kliknięto przycisk *Nie*

## **InputBox**

Wynikiem wykonania tej funkcji jest wartość typu `String` wpisana przez użytkownika w oknie dialogowym.

## Składnia:

**InputBox**(*Komunikat*, *Tytuł*, *Domyślna*, *x*, *y*, *PlikPomocy*,  
*Kontekst*)

*Komunikat* (wymagany)

parę słów zachęty dla użytkownika; będą one wyświetlone w oknie komunikatu;

*Tytuł* (opcjonalny)

komunikat, który będzie widoczny na pasku tytułu. Jeżeli go pominiesz, zostanie tam wyświetlona nazwa „Microsoft Excel”.

*Domyślna* (opcjonalny)

zawiera wartość domyślną wprowadzanej zmiennej; będzie ona wyświetlana w miejscu wprowadzania danych. Jeżeli pominiesz ten parametr, Excel nie wyświetli żadnej wartości w oknie.

*x*, *y* (opcjonalny)

współrzędne (w pikselach) lewego górnego narożnika okna dialogowego względem lewego górnego narożnika ekranu;

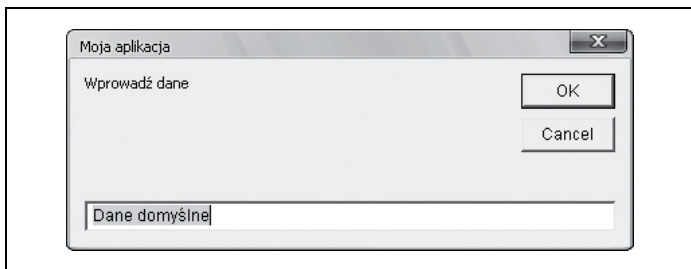
*PlikPomocy*, *Kontekst*

plik pomocy i miejsce w nim, do którego prowadzić będzie łącze po kliknięciu przycisku *Pomoc*.

## Przykład:

Efektem wykonania poniższego kodu będzie okno dialogowe pokazane na rysunku 3.1. Jeżeli użytkownik nie wprowadzi żadnej wartości i kliknie OK, zmiennej *a* zostanie przypisana wartość podana jako dane domyślne. Jeżeli wybierze przycisk *Cancel*, funkcja zwróci wartość ciągu zerowej długości.

```
a = InputBox("Wprowadź dane", "Moja aplikacja",  
            "Dane domyślne")
```



Rysunek 3.1. Okno dialogowe wyświetlone za pomocą funkcji `InputBox`

## Funkcje logiczne

VBA oferuje pełną gamę ogólnie znanych operatorów logicznych:

**Not**, **And**, **Or**, **Xor**, **Eqv**, **Imp**.

Wszystkich operatorów oprócz **Not** możemy używać w taki sam sposób:

wynik = wartość1 operator wartość2

gdzie wartość1 i wartość2 to wyrażenia, na których dokonuje się operacji.

Operator **Not** ma łatwiejszą składnię:

wynik = **Not** argument

czego wynikiem będzie oczywiście odwrotność podanego argumentu.

## Operacje na łańcuchach

### StrComp

Zwraca wynik porównania dwóch ciągów tekstowych.

## Składnia:

**StrComp**(*Ciag1*, *Ciag2*, *Porównanie*)

*Ciag1*, *Ciag2* (wymagany)  
porównywane ciągi;

*Porównanie* (opcjonalny)  
typ porównania. Może przybierać wartości:

*vbUseCompareOption* lub -1  
Wykonuje porównania według ustawień domyślnych lub określonych w wyrażeniu *Option Compare*.

*vbBinaryCompare* lub 0  
Dokonuje porównania binarnego.

*vbTextCompare* lub 1  
Dokonuje porównania tekstowego.

Patrz też: *Option compare*.

## StrConv

Konwertuje wskazany ciąg znaków według zadanych parametrów.

### Składnia:

**StrConv**(*Ciag*, *Konwersja*, *LCID*)

*Ciag* (wymagany)  
ciąg znaków poddany konwersji;

*Konwersja*  
sposób konwersji. Możliwe wartości dla polskiej wersji pakietu Office:

*vbUpperCase* lub 1  
Zamienia na duże litery.

*vbLowerCase* lub 2  
Zamienia na małe litery.

`vbProperCase` lub 3

Pierwsza litera każdego wyrazu duża, pozostałe małe.

`vbUnicode` lub 64

Zamienia znaki na Unicode (nie dostępne na Macu).

`vbFromUnicode` lub 128

Zamienia znaki z Unicode na format określony w stronie kodowej komputera (nie dostępne na Macu).

*LCID* (opcjonalny)

ID ustawień regionalnych. Domyślnie są to ustawienia systemowe.

## **Lcase, Ucase**

**Lcase** w podanym tekście zmienia wszystkie litery na małe. **Ucase** zamienia wszystkie litery na duże.

Składnia:

**Ucase**(*Ciqg*)

**Lcase**(*Ciqg*)

*Ciqg*

argument wymagany.

## **Space, String**

**Space** zwraca podaną liczbę spacji. **String** wstawia podaną liczbę dowolnych znaków.

Składnia:

**Space**(*Ile*)

**String**(*Ile*, *Znak*)

*Ile* (wymagany)

liczba wstawionych znaków;

*Znak* (wymagany)

znak, który zostanie wstawiony.



## Len

Zwraca długość wskazanego ciągu znaków.

Składnia:

**len**(*Ciąg*)

*Ciąg* (wymagany)

ciąg znaków ujęty w cudzysłów lub zmienna reprezentująca wyrażenie.

## Format

Zwraca podane wyrażenie w formacie określonym w funkcji.

Składnia:

**Format**(*Wyrażenie*, *Format*, *1dzieńTygodnia*, *1TydzieńRoku*)

*Wyrażenie* (wymagany)

obiekt poddany konwersji;

*1dzieńTygodnia* (opcjonalny)

wartość określająca pierwszy dzień tygodnia (patrz funkcja `DateDiff`);

*1TydzieńRoku* (opcjonalny)

wartość określająca pierwszy tydzień roku (patrz funkcja `DateDiff`);

*Format* (opcjonalny)

oczekiwany format wyrażenia po konwersji. Jeżeli pominiemy ten argument, funkcja zamieni argument na tekst.

Do wyboru mamy następujące argumenty:

General Date      General Number

Long Date          Currency

Medium Date        Fixed

Short Date          Standard

Long Time	Percent
Medium Time	Scientific
Short Time	Yes/No
	True/False
	On/Off

### Przykłady:

```
Czas = #10:24:07#
Data = #Luty 2, 2003#
```

```
Wynik = Format(Czas, "h:m:s")
Zwróci "17:4:23".
```

```
Wynik = Format(Now(), "short time")
Zwróci tylko aktualną godzinę i minuty.
```

```
Wynik = Format(Data, "long date")
Zwróci "2 luty 2003".
```

```
Wynik = Format(Data, "short date")
Zwróci "2003-02-02".
```

Pominięcie parametru `Format` spowoduje zmianę argumentu na tekst:

```
Wynik = Format(23) ' Zwróci "23".
```

### Przykłady formatów definiowanych przez użytkownika:

```
Wynik = Format(35.7, "###0.00")
Zwróci "35.70".
```

```
Wynik = Format(0.1, "0.00%")
Zwróci "10.00%".
```

```
Wynik = Format("MAŁE LITERY", "<")
Zwróci "małe litery".
```

```
Wynik = Format("Duże litery", ">")
Zwróci "DUŻE LITERY".
```

## LSet, Rset

Zamieniają tekst we wskazanej zmiennej, licząc od lewej (**LSet**) lub od prawej (**RSet**), pozostawiają przy tym oryginalną długość ciągu.

Przykład:

Nadanie zmiennej a pierwotnej wartości.

```
a = "Excel"
```

**LSet** a = ("abc")

Zwróci w wyniku "abc".

**RSet** a = ("abc")

Zwróci w wyniku " abc".

**LSet** a = ("123456789")

Zwróci w wyniku "12345".

**RSet** a = ("123456789")

Zwróci w wyniku "12345".

## InStr, InStrRev

Zwraca pozycję poszukiwanego ciągu znaków w innym, licząc zawsze od lewej. **InStr** wyszukuje od lewej strony ciągu, **InStrRev** — od prawej.

Składnia:

```
InStr(Start, Ciąg1, Ciąg2, Porównanie)
```

```
InStrRev(Ciąg1, Ciąg2, Start, Porównanie)
```

*Start* (opcjonalny)

od którego znaku szukać?

*Ciąg1* (wymagany)

w czym przeszukujesz?

Ciąg2 (wymagany)  
czego szukasz?

Porównanie (opcjonalny)  
patrz funkcja StrComp.

Przykład:

```
Sub wyszukaj()  
a = InStr(3, "Niewielki", "w")  
b = InStrRev("Niewielki", "w", 7)  
MsgBox "Wynik funkcji InStr= " & a _  
& Chr(13) & "Wynik funkcji InStrRev= " & b  
End Sub
```

Zmienne a i b przyjmą wartość 4. Dla zmiennej a wyszukiwanie litery „w” rozpocznie się od trzeciej pozycji w słowie „Niewielki”. Excel znajdzie literę „w” zaraz po literze, od której rozpoczyna wyszukiwanie, i jak mogłoby się wydawać, powinien zwrócić wartość 1. Jednak litera „w” jest na czwartym miejscu w słowie, licząc od jego początku, i taką wartość przyjmie zmienna a.

Dla instrukcji

```
a = InStr(6, "Niewielki", "w")
```

zmienna a przyjmie wartość zero, bo po szóstym znaku w słowie „Niewielki” nie występuje już litera „w”. Dla zmiennej b wyszukiwanie litery rozpocznie się od siódmej pozycji, licząc od prawej strony (czyli również od litery „e”). Wynikiem działania kodu będzie liczba 4, bo litera „w” jest na czwartym miejscu w słowie, licząc od jego początku (od lewej).

**Left, Right,**

Wycina określoną liczbę znaków we wskazanym ciągu od lewej (**Left**) lub od prawej strony (**Right**).

Składnia:

```
Right(Ciąg, Długość)  
Left(Ciąg, Długość)
```

*Ciąg* (wymagany)  
analizowany tekst;

*Długość* (wymagany)  
liczba znaków do wycięcia.

### **LTrim, RTrim, Trim**

Usuwa początkowe (**LTrim**) lub końcowe (**RTrim**) spacje w analizowanym tekście.

Funkcja **Trim** jest złożeniem funkcji **LTrim** i **RTrim**.

Składnia:

**Trim** (*Ciąg*)

*Ciąg*  
wymagany.

Przykład:

a = " przykładowy tekst "

wynik = **LTrim**(a)  
Zwróci wartość "przykładowy tekst".

wynik = **RTrim**(a)  
Zwróci wartość " przykładowy tekst".

wynik = **Trim**(a)  
Zwróci wartość "przykładowy tekst".

### **Replace**

Znajduje i zamienia wskazane ciągi znaków.

Składnia:

**Replace**(*Ciąg*, *Znajdź*, *Zamień*, *Start*, *Ile*, *Porównanie*)

*Ciąg* (wymagany)  
rozpatrywany ciąg znaków;

*Znajdź* (wymagany)

ciąg, którego szukasz;

*Zamień* (wymagany)

ciąg, który wstawisz w miejsce starego;

*Start* (opcjonalny)

od którego znaku szukać? Znaki znajdujące się przed wskazanym zostaną usunięte.

*Ile* (opcjonalny)

ile zamian wykonać? Domyślnie wartość wynosi  $-1$ , co oznacza, że zostaną zamienione wszystkie poszukiwane znaki.

*Porównanie* (opcjonalny)

patrz funkcja `StrComp`.

Przykład:

```
a = Replace("mama", "m", "t")
da wynik "tata".
```

```
a = Replace("mama", "m", "t", , 1)
da wynik "tama".
```

```
a = Replace("mama", "m", "t", 3, 1)
da wynik "ta", ponieważ dwie pierwsze litery zostaną
w wyniku pominięte.
```

## **StrReverse**

Zwraca łańcuch znaków w odwrotnej kolejności.

Przykład:

```
a = StrReverse("mama")
da wynik "amam".
```

```
a = StrReverse("kajak")
da wynik "kajak".:-)
```

## Mid

Wycina ze wskazanego tekstu określoną liczbę znaków.

Składnia:

```
Mid(Ciąg, Start, Długość)
```

*Ciąg* (wymagany)  
analizowany tekst;

*Start* (wymagany)  
od którego znaku zacząć?

*Długość* (opcjonalny)  
ile znaków wyciąć?

## Pętle i skoki

### GoSub...Return,

Określa skok do (**GoSub**) i powrót z podprogramu (**Return**).

Podprogram musi znajdować się w tej samej procedurze, z której ma nastąpić skok do niego.

Składnia:

```
    kod programu głównego  
GoSub etykieta  
    ...  
    koniec programu głównego  
etykieta:  
    ...kod podprogramu  
Return
```

### GoTo

Określa skok bezwarunkowy w obrębie procedury do miejsca określonego etykietą.