

PROGRAMOWANIE MAKR W PROGRAMIE EXCEL W JĘZYKU VISUAL BASIC

W programie Microsoft Excel można często powtarzane czynności zautomatyzować przy użyciu tzw. makr. Makro jest serią poleceń i funkcji, które są zapisywane za pomocą języka programowania i może być uruchomione zawsze, gdy zachodzi potrzeba ich wykonania. Firma Microsoft dostarcza z pakietem Office język programowania Visual Basic. Makra można rejestrować, jak rejestruje się muzykę przy użyciu magnetofonu lub można napisać za pomocą języka Visual Basic, i dalej można uruchomić, aby wykonać zarejestrowane bądź zapisane polecenia.

Przykład 1. Rejestrowanie makra.

W nowym arkuszu wykonaj następujące czynności:

1. Z menu narzędzia wybierz polecenia makro $\Rightarrow$ zarejestruj nowe makro.
2. Zmień nazwę makra na „Jan” i wybierz klawisz skrótu na **Ctrl-j** i potwierdź OK – rozpoczął się tryb nagrywania makra.
3. W wybranej komórce wpisz napis „Jaś”, potwierdź wpisany napis myszką na pasku formuły.
4. Zakończ nagrywanie makra ikoną na pasku zadań.
5. Wybierz nową komórkę i naciśnij **Ctrl-j**.

Przykład 2. Edycja Makra.

1. Z menu narzędzia wybierz polecenia makro $\Rightarrow$ makra...
2. Wybierz makro „Jan” i kliknij Edycja.
3. W oknie edytora Visual Basic powinieneś zobaczyć tekst podobny do wypisanego poniżej:

```
Sub Jan()  
,  
    ' Jan Makro  
    ' Makro zarejestrowane 00-02-25, autor Tadeusz Ziębakowski  
,  
    ' Klawisz skrótu: Ctrl+j  
,  
  
    ActiveCell.FormulaR1C1 = "Jaś"  
End Sub
```

4. Dopisz następujący tekst:

```
Sub Małgorzata()  
    ActiveCell.FormulaR1C1 = "Małgosia"  
End Sub
```

5. Przejdź do arkusza kalkulacyjnego i wybierz polecenia makro $\Rightarrow$ makra...

6. Wybierz makro „Małgorzata” i kliknij Opcje i wpisz klawisz skrótu **Ctrl-m**.
7. Wybierz nową komórkę i naciśnij **Ctrl-m**.

Uruchamianie makra niekoniecznie musi odbywać się poprzez klawisz skrótu. Można to robić z menu Narzędzia bądź poprzez umieszczenie w arkuszu specjalnego przycisku.

Przejdziemy teraz do omówienia podstawowych elementów składni języka Visual Basic. Na potrzeby niniejszego zbioru zadań ograniczymy się do jego skrótowego omówienia bez wchodzenia w istotę bardziej złożonych zapisów.

1. Zmienne

Każde makro jest zapisem czynności, które wykonywane są na pewnym określonym zbiorze informacji, który będziemy nazywać danymi. Dane mogą być zapisane bądź w komórkach arkusza kalkulacyjnego bądź w pewnym zarezerwowanym obszarze pamięci i są reprezentowane wewnątrz makra poprzez zmienne. Każda zmienna określona jest poprzez podanie nazwy, którą ustala piszący makro tworząc ją z liter i cyfr (pierwszym znakiem w nazwie powinna być litera). Są jednak pewne „specjalne” zmienne, które są zdefiniowane w Excelu i służą do identyfikowania arkuszy, komórek czy zakresów komórek. Wypiszemy niektóre takie zmienne.

`ActiveCell.FormulaR1C1`

przechowuje wartość wybranej komórki

`Selection.FormulaR1C1`

pozwała wpisać wartości do wybranego zakresu komórek

`Range("A1:C5").FormulaR1C1`

pozwała wpisać wartości do zakresu „A1:C5”

`Selection.Cells(1,2).FormulaR1C1`

pozwała wpisać wartości do komórki w pierwszym wierszu i drugiej kolumnie w wybranym obszarze.

`Range("A1:C5").Cells(5).FormulaR1C1`

pozwała wpisać wartości do 5 komórki w zakresie „A1:C5” tj. komórki B2.

`Range("A1:C5").Cells.Count`

podaje liczbę komórek w zakresie, w tym przypadku 15.

`Range("A1:C5").Columns.Count`

podaje liczbę kolumn w zakresie, w tym przypadku 3.

`Range("A1:C5").Rows.Count`

podaje liczbę wierszy w zakresie, w tym przypadku 5.

Czynnik `.FormulaR1C1` można pominąć, gdy wartościami komórek nie są formuły. Podobnie jak komórki w programie Excel zmienne mogą przechowywać dane różnych typów, przy czym w Visual Basic różnorodność typów jest znacznie większa (co więcej można tworzyć własne typy danych, czego nie będziemy omawiać). Typy danych mają swoje nazwy. Wymienimy tylko kilka podstawowych typów:

typ	opis	operacje	symbol
Byte	liczby całkowite: Od 0 do 255	dodawanie odejmowanie	+ -
Integer	liczby całkowite: Od -32 768 do 32 767	mnożenie dzielenie całkowite reszta z dzielenia	* Div Mod
Single	liczby rzeczywiste: Od ok. $-3,4 \cdot 10^{38}$ do ok. $3,4 \cdot 10^{38}$	dodawanie odejmowanie	+ -
Double	liczby rzeczywiste: Od ok. $-1,8 \cdot 10^{308}$ do ok. $1,8 \cdot 10^{308}$	mnożenie dzielenie potęgowanie	* / ^
Boolean	wartości logiczne: True (prawda), False (fałsz)	koniunkcja alternatywa alternatywa wykluczna negacja równoważność implikacja	And Or Xor Not Eqv Imp
String	łańcuchy znaków	łączenie łańcuchów	& , +
Variant	Połączenie Double i String	jak dla Double i String	
		operatory relacji (wynik jest typu Boolean):	=, <, <=, >, > =, <>

W języku Visual Basic nadanie wartość zmiennej określa typ zmiennej, jednak w bardziej złożonych makrach w celu uniknięcia błędów powinno się określić typ zmiennej za pomocą specjalnej instrukcji deklaracji Dim.

2. Wyrażenia

Wyrażenia tworzymy podobnie jak formuły w arkuszu, z tym, że zamiast adresów komórek wstawiamy zmienne. W wyrażeniach możemy wykorzystywać funkcje wewnętrzne Visual Basic podobne do funkcji Excels¹, a także istnieje możliwość definiowania własnych funkcji. Przy tworzeniu wyrażeń należy zwrócić szczególną uwagę na zgodność typów zmiennych, funkcji i używanych operatorów.

3. Instrukcje

Czynności, które ma realizować makro zapisujemy w postaci tzw. instrukcji. Instrukcje zapisujemy w kolejnych liniach. Jeśli chcemy kilka instrukcji zapisać w jednej linii oddzielamy je dwukropkami. Uwaga: napisy zaczynające się od apostrofu nie są instrukcjami lecz komentarzami. Omówimy kilka najważniejszych instrukcji:

Instrukcja przypisania

Instrukcja ta ma postać:

¹ W Visual Basic-u należy bardziej niż w Excel-u uważać na zgodność typów argumentów funkcji z używanymi zmiennymi. Wykaz funkcji i ich opis można znaleźć w pomocy Visual Basic.

zmienna = wyrażenie

W wyniku wykonania instrukcji zmienna otrzymuje wartość wyrażenia.

Przykłady:

```
d = b^2 - 4*a*c
i = i + 1
ActiveCell.FormulaR1C1 = "Jaś"
```

Pierwsza instrukcja nadaje zmiennej *d* wartość wyrażenia $b^2 - 4*a*c$, gdzie *a*, *b*, *c* są zmiennymi. Druga powiększa wartość zmiennej *i* o jeden. Trzecia wstawia do aktywnej komórki napis "Jaś".

Instrukcja wywołania procedury

Instrukcja ta ma postać:

<nazwa procedury> parametr1, parametr2,...

Przykład:

```
MsgBox "Czas na naukę pisania makr!"
```

Instrukcja ta wyświetla komunikat "Czas na naukę pisania makr!" wykorzystując procedurę Visual Basic `MsgBox`.

Instrukcja warunkowa

Instrukcja ta ma postać:

If *warunek* **Then** *instrukcje* **Else** *instrukcje_else*

lub w wersji krótszej:

If *warunek* **Then** *instrukcje*

Inną możliwością jest zastosowanie składni blokowej:

```
If warunek Then
    instrukcje
Else
    instrukcje_else
End If
```

lub w wersji krótszej:

```
If warunek Then
    instrukcje
End If
```

warunek powinien być wyrażeniem logicznym. W przypadku gdy jest on prawdziwy wykonywane są *instrukcje* w przeciwnym przypadku *instrukcje_else*. W wersji krótszej gdy warunek nie jest spełniony następuje przejście do instrukcji następnej po **If**. W składni jednowierszowej instrukcje oddziela się dwukropkami.

Instrukcja pętli For

```
For licznik = początek To koniec  
  instrukcje  
Next licznik
```

Licznik jest zmienną numeryczną, *początek* i *koniec* są wartościami początkowymi i końcowymi licznika. Wykonywane są *instrukcje*, po każdym wykonaniu licznik zwiększany jest o 1. Wykonywanie zostanie przerwane gdy *licznik* przekroczy wartość *koniec*.

Instrukcja pętli Do

Instrukcja ta może mieć następujące warianty składni:

Do While <i>warunek</i> <i>instrukcje</i> Loop	albo	Do Until <i>warunek</i> <i>instrukcje</i> Loop
--	------	--

albo

Do <i>instrukcje</i> Loop While <i>warunek</i>	albo	Do <i>instrukcje</i> Loop Until <i>warunek</i>
--	------	--

Jedna lub kilka instrukcji oznaczonych przez *instrukcje* powtarzanych jest tak długo, jak długo *warunek* jest spełniony – wariant ze słowem **While** - albo dopóki nie stanie się prawdziwy - wariant ze słowem **Until**, przy czym *warunek* można sprawdzać na początku lub na końcu.

Instrukcja deklaracji

Instrukcja ta przydziela pamięć zmiennej i określa jej typ i w uproszczeniu jest następującej postaci:

```
Dim zmienna1 As typ1, zmienna2 As typ2, ...
```

Przykłady:

Poniżej zadeklarowano 3 zmienne: pierwszą typu **Variant** (typ domyślny), drugą typu **Double**, trzecią typu **String**.

```
Dim liczba, numer As Double, napis As String
```

W poniższym wierszu zadeklarowano tablicę o stałym rozmiarze jako tablicę o elementach typu **Integer** zawierającą 11 wierszy i 11 kolumn:

```
Dim MojaTab(10, 10) As Integer
```

Pierwszy argument reprezentuje wiersze; drugi argument reprezentuje kolumny, oba standardowo numerowane od 0 - inaczej niż ma to miejsce w przypadku wyrażenia `Cells(i, j)`.

4. Procedury i Funkcje

Każde makro w Excelu jest zapisywane jako procedura Visual Basic. Podczas pisania bardziej rozbudowanych makr często zdarza się, że taki sam lub podobny fragment makra występuje w kilku miejscach. Można wówczas taki powtarzający się fragment zapisać w postaci dodatkowej procedury lub funkcji.

Procedura lub funkcja - wyodrębniona sekwencja instrukcji, stanowiąca pewną całość, posiadająca jednoznaczną nazwę i ustalony sposób wymiany informacji z pozostałymi częściami programu lub makra.

Stosowanie procedur i funkcji na ogół skraca zapis, a także ułatwia pisanie dużych rozbudowanych makr dzięki podzieleniu go na odrębne logicznie spójne części. Różnica pomiędzy procedurą a funkcją polega na przekazywaniu wartości końcowych.

Procedury i funkcje definiuje się przy pomocy instrukcji **Sub** i **Function**:

```
Sub nazwa (lista_argumentów)  
    instrukcje  
End Sub  
  
Function nazwa (lista_argumentów) As typ  
    instrukcje  
    nazwa = wyrażenie  
End Function
```

lista_argumentów jest listą zmiennych oddzielonych przecinkami i jest nieobowiązkowa podobnie jak *typ*, który określa typ zwracanego wyniku przez funkcję. W definicji funkcji *typ* wyrażenia (*wyrażenie*) powinien być zgodny z typem funkcji. Ponadto w definicji procedury i funkcji mogą się pojawić instrukcje **Exit Sub** i odpowiednio **Exit Function** przerywające działanie procedury lub funkcji.

Przykład 3.

Makro **Main** wywołuje dwie procedury: **Sygnał**, która wysyła krótkie sygnały dźwiękowe (procedura **Beep**) w ilości określonej przez parametr i procedurę **Komunikat**, która wypisuje napis "Czas na kolejny przykład!".

```
Sub Main()  
    'źródło: plik pomocy Visual Basic  
    Sygnał 100  
    Komunikat  
End Sub  
  
Sub Sygnał(ile_dzw)  
    For licznik = 1 To ile_dzw  
        Beep  
    Next licznik  
End Sub  
  
Sub Komunikat()  
    MsgBox "Czas na kolejny przykład!"  
End Sub
```

Przykład 4

Makro rozwiązuje równanie kwadratowe $ax^2 + bx + c = 0$ w liczbach rzeczywistych. Współczynniki a, b, c należy wpisać do komórek A1, B1, C1. Pierwiastki, jeśli są, wypisane będą w komórkach A3 i A4. Makro zawiera jawną deklarację zmiennych instrukcją Dim, bez której makro będzie działać, lecz zmienne a, b, c i Δ będą typu Variant.

```
Sub równanie_kwadratowe()
Dim a As Single, b As Single, c As Single, delta As Single
a = Range("A1")
b = Range("B1")
c = Range("C1")
delta = b ^ 2 - 4 * a * c
If a = 0 Then
    MsgBox "To nie jest równanie kwadratowe"
    Exit Sub
End If
If delta < 0 Then MsgBox "Nie ma rozwiązań": Exit Sub
If delta = 0 Then
    Range("A3") = -b / (2 * a)
Else
    Range("A3") = (-b + Sqr(delta)) / (2 * a)
    Range("A4") = (-b - Sqr(delta)) / (2 * a)
End If
End Sub
```

Przykład 5

Makro w kolumnie A mnoży po kolei przez 2 każdą liczbę, aż do napotkania komórki pustej.

```
Sub Podwajaj()
i = 1
Do While Range("A1").Cells(i, 1) <> ""
    Range("A1").Cells(i, 1) = 2 * Cells(i, 1)
    i = i + 1
Loop
End Sub
```

Przykład 6

Funkcja wyznacza normę euklidesową wektora o współrzędnych x, y, z . Funkcja ta jest dostępna w arkuszu jako funkcja użytkownika. Należy pamiętać aby w formule Excela parametry funkcji oddzielać średnikami.

```
Function Norma(x As Single, y As Single, z As Single) _
    As Single
    Norma = Sqr(x ^ 2 + y ^ 2 + z ^ 2)
End Function
```

Uwaga: Podkreślenie poprzedzone spacją oznacza kontynuację instrukcji w następnym wierszu

Zadania

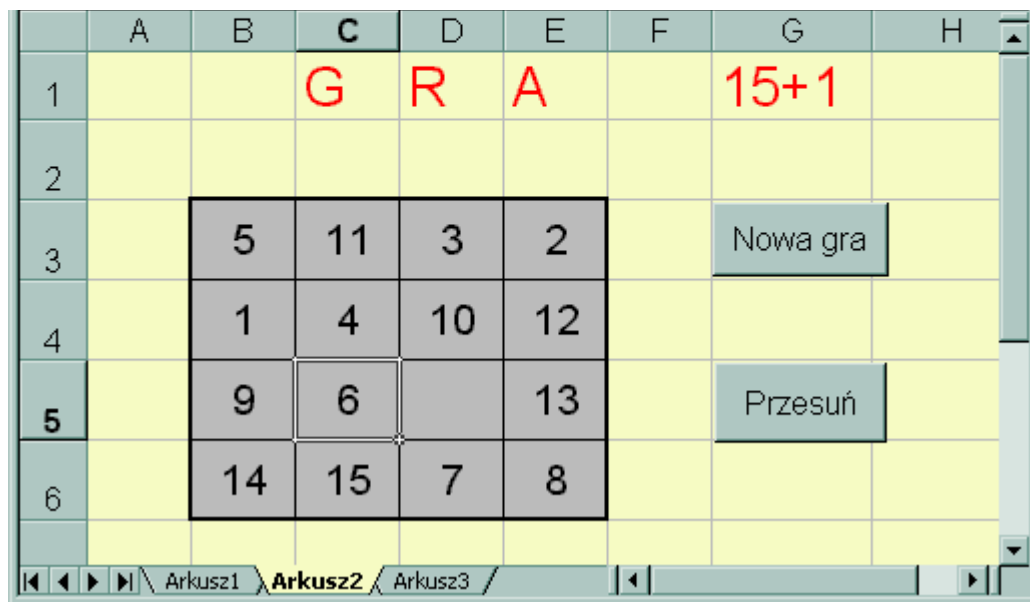
1. Napisz makro, które przepisuje wartości co drugiej komórki kolumny A do kolumny B po kolei, aż do napotkania komórki pustej.
2. Napisz makro, które przepisuje wartość komórki A1 do A2, A2 do A3,...,A19 do A20 i A20 do A1.
3. Napisz makro, które w zaznaczonym zakresie komórek odwraca kolejność danych tzn. zamienia wartości pierwszej i ostatniej komórki, drugiej i przedostatniej itd.
4. Zaprojektuj makro, które rozwiązuje układ dwóch równań liniowych z dwoma niewiadomymi:

$$\begin{cases} a_1x + b_1y = c_1 \\ a_2x + b_2y = c_2 \end{cases}$$

Wskazówki:

Przyjmijmy $D = a_1b_2 - b_1a_2$, $D_x = b_2c_1 - b_1c_2$, $D_y = -a_2c_1 + a_1c_2$. Z algebry wiadomo, że jeśli $D \neq 0$ to mamy jedno rozwiązanie $x = D_x / D$ i $y = D_y / D$. Jeśli $D = 0$, $D_x = 0$ i $D_y = 0$ to mamy nieskończenie wiele rozwiązań: dla dowolnego x , $y = (c_1 - a_1x) / b_1$ spełnia układ równań. Jeśli $D = 0$, $D_x \neq 0$ lub $D_y \neq 0$ to układ nie ma rozwiązań. Współczynniki a_i, b_i, c_i można wpisać do zakresu A1:C2. Makro powinno zbadać również „złotliwe” przypadki, gdy współczynniki a_i, b_i, c_i nie wyznaczają układu równań z dwoma niewiadomymi.

5. Należy zaprojektować arkusz i makra, tworzące grę 15+1. Plansza tej gry jest pokazana poniżej:



Zadaniem gracza jest ustawienie liczb od 1 do 15 po kolei przez przesuwanie liczb. Można przesuwać liczby pionowo lub poziomo na puste miejsce (w obrębie prostokąta). Po wskazaniu komórki, przycisk Przesuń powinien uruchomić makro, które wykona przesunięcie np. w powyższej sytuacji liczba 6 powinna się przesunąć w prawo. Zadaniem przycisku Nowa gra jest ponowne, losowe rozmieszczenie liczb wewnątrz prostokąta. Liczby losowe można otrzymać za pomocą funkcji Rnd, - jej omówienie można znaleźć w pomocy Visual Basic.