Visual Basic dla aplikacji w Excelu 2000 (Jacek Zabawa, 2005)

Opracowano na podstawie:
Łukasz Tatarkiewicz, *Excel 5.0 Visual Basic i funkcje* EXIT Warszawa 1994
Julitta Korol, *Excel 5 następne kroki* MIKOM Warszawa 1995

Ćwiczenie I.

Definiowanie własnych funkcji i procedur i odwoływanie się do nich.
Utworzymy bardzo nieskomplikowaną funkcję obliczającą odwrotność liczby i jednocześnie wyświetlającą komunikat w przypadku dzielenia przez 0.

a) Proszę wybrać < Narzędzia| Makro | Edytor Visual Basic > lub nacisnąć ALT-F11
b) Proszę wybrać < Insert | Module>
c) Proszę wpisać w [modules] Module1 następujący tekst:

```
Function Odwrotność(x)
   If x = 0 Then
      Beep
      MsgBox "nie istnieje odwrotność liczby zero"
     Odwrotność = 1 / x
   Else
      Odwrotność = 1 / x
   End If
End Function
```

i już można wykorzystywać ją w arkuszu w taki sam sposób jak np. IRR.
Zatwierdzenie i powrót do arkusza po naciśnięciu ALT-Q (File | Close and return to MS Excel)

d) proszę przetestować tę funkcję

Ćwiczenie II.

Zmiana formatu komórki z formuł obliczeniowych (np. =323/13) na format liczbowy (=24,84 ....)

a) Wpisz np. do komórki A5 wyrażenie =10/2*13. W komórce znajdzie się liczba 65 (wynik wpisanego wzoru). Utrzymać stan aktywności tej komórki.
b) Wybierz <Narzędzia | Makro | Zarejestruj nowe makro> i wpisz nazwę nowego makra – np. Zamiana_na_liczbę;
c) Nagraj następujące czynności składające się na makro
d) <Edycja | Kopiuj>; <Edycja | Wklej specjalnie> i zaznacz opcję “wartości” i nacisnij <OK>
e) Zakończ nagrywanie makra – klawisz < ■ > określony jako „zatrzymaj rejestrowanie”

Wypróbuj polecenie (makro) zamiana_na_liczbę na kilku formułach (wpisz je do arkusza)
Zajrzyj do modułu [module2] z definicją nagranego makra i przeanalizuj jego zawartość. Zwróć uwagę na tzw. metody – np. “Copy”, “Paste Special” i ich argumenty (np. Paste:=xlValues). Porównaj definiowanie funkcji i procedury

Ćwiczenie III.

Bezpośrednie definiowanie zmiennych. Zmienna – nazwa do której przypisano dowolną wartość (liczbową, tekst, wartość logiczną). Uwaga: nazwa zmiennej musi rozpoczynać się od litery, nie może zawierać spacji i nie może przypominać adresów komórek oraz komend i funkcji języka VBA; nie mogą pokrywać się z nazwami procedur, funkcji.
Wprowadźmy tekst do „Module 1".

```
Function Zysk_Ciułacza(Naliczone_Odsetki)
Dim StopaPodatku, Podatek As Double
'  komentarz - zakładamy stopę podatku 19%, zmiennoprzecinkowa 64 bitowa

   StopaPodatku = 0.19
   Podatek = Naliczone_Odsetki * StopaPodatku
   Zysk_Ciułacza = Naliczone_Odsetki - Podatek
End Function
```

Podana poniżej funkcja oblicza zysk z operacji kupna i sprzedaży podanej liczby akcji bez uwzględnienia prowizji maklerskiej, po potrąceniu podatku.

```
Function Zysk_Gracza(LiczbaAkcji, CenaSprzedaży, CenaKupna)
'  bez deklaracji typu zmiennych
'  komentarz - zakładamy stopę podatku 19%
   StopaPodatku = 0.19
   Kupno = LiczbaAkcji * CenaKupna
   Sprzedaż = LiczbaAkcji * CenaSprzedaży
   ZyskBrutto = Sprzedaż - Kupno

   If CenaSprzedaży > CenaKupna Then
     Podatek = StopaPodatku * ZyskBrutto
     Zysk_Gracza = ZyskBrutto - Podatek
   Else
     Zysk_Gracza = ZyskBrutto
   End If
End Function
```

Inne informacje:

a) Deklarowanie zmiennych – instrukcja Dim, np. **Dim** a, b, zmienna
b) **Option Explicit** – umieszczane na początku (przed definicją pierwszej procedury) modułu wymusza konieczność deklarowania wszystkich zmiennych (dobry nawyk – pozwala na wynajdywanie ewentualnych błędów) – jest to tzw. wyłączenie trybu automatycznego rozpoznawania zmiennych – wszystkie nieznane nazwy traktowane są jako zmienne
e) Definiowanie stałej **Const** stała = wyrażenie np. const LICZBA_E = 2.7182818284
f) Operatory (niektóre działają także na tekstach):
   +, -, ^, *, /, ** (część całkowita), **Mod (reszta z dzielenia)

g) Operatory porównania i logiczne
=, >, <, >=, <=, <>, **And, Or, Imp** (implikacja), **Eqv** (równoważność), **Xor** (nierównoważność), **Not** (negacja), **Like** (tekst1 Like tekst2 daje TRUE gdy teksty są identyczne; “xyz” LIKE “???” daje TRUE

h) Komendy warunkowe
If ... Then ... ; If ... Then ... End If; If ... Then ... Else ... End If ; if ... Then ... ElseIf...
proszę przemyśleć zamieszczoną poniżej procedurę (pojawi się ona na liście makr):

```
Sub ABC()
Liczba = InputBox("Proszę wpisać liczbę różną od zera")
   If Liczba = 0 Then
      MsgBox "zostało wpisane zero!"
   Else
      MsgBox "udało się!"
   End If
End Sub
```

Zróżnicowanie formy komendy zależy od ilości poleceń zawartych między słowami kluczowymi **if, then, else, end if**. Można wpisywać wiele instrukcji w jednej linii oddzielając je znakiem dwukropka.
Przykład zastosowania: funkcja obliczająca prowizję od dokonanej wypłaty w bankomacie. Załóżmy następujące prowizje od transakcji:

Transakcje gotówkowe własne bankomaty i kasy - bez prowizji
obce bankomaty w kraju - 5 PLN, bankomaty lub kasy banków za granicą – 2% minimum 10 PLN

Komenda Select Case przykład: stawka podatku od posiadania psa zależy od ilości posiadanych psów:

```
Function PSOWE(Ilość_psów)
        Select Case Ilość_psów Then
        Case 1
                Podatek = 10
        Case 2
                Podatek = 15
        Case 3
                Podatek = 18
        Case 4
                Podatek = 195
        Case Is >=5
                Podatek = 200
        Case Else
                Podatek = 0
        End Select
End Function
```

i) Pętle

Do While warunek
    polecenie_1
    polecenie_2
    ...
Loop

Inna wersja:

Do
    Polecenie_1
    Polecenie_2
    ...
Loop While warunek

Inna wersja:

Do
    Polecenie_1
    Polecenie_2
    ...
Loop Until warunek

For x=1 **To** 4
    Polecenie_1
    Polecenie_2
    ...
Next x

Inna wersja:

For x=0 **To** 4 **Step** 2
    Polecenie_1
    Polecenie_2
    ...
Next x

Skoki:
12: zysk = zyskbrutto-podatek
To_jest_etykieta: InputBox(“podaj kwotę”)

Goto To_jest_etykieta
Goto 12

Zarządzenie obiektami:

Pojęcie **obiektu** określa pewną całość, która może być przedmiotem modyfikacji np. pakiet arkuszy, pojedynczy arkusz, blok komórek, okno, linia, prostokąt, belka narzędziowa.
Obiekty mogą zawierać inne obiekty (np. arkusz zawiera wiele bloków komórek)
Każdy obiekt ma swoje właściwości (*properties*) np. Czcionka (Font) ma właściwość Kolor, Rozmiar itd. Właściwości mogą przybierać pewne wartości: liczby, teksty, wartości logiczne, stałe VBA lub Excela.
Obiekty mają także **metody** (znaczeniowo zbliżone do procedur, jednak mogą one powiązane są z obiektami z pewnej **klasy**). Służą one do wykonywania operacji na obiekcie – np. metoda Copy, Clear. Są także metody pozwalające na zmianę wartości właściwości obiektu. Przykład: właściwość WorkSheet.ProtectContents i metoda WorkSheet.Protect.

Najważniejsze obiekty:
Obiekt Range – komórka lub blok komórek.
Zastosowanie właściwości: ActiveCell.Value = 2 (musi być zaznaczona jakaś komórka)
Metoda Cells – zwraca komórkę arkusza. Zastosowanie: Cells(1,”A”): Cells(4,”B”): Cells(2,2).Value = 6
Metoda Range
Range(”A1”, ”D4”) zwraca blok komórek A1:D4
Range(Cells(1,1), Cells(2,2)) zwraca blok komórek A1:B2
Range(”Tabelka”) zwraca blok komórek o nazwie “Tabelka”
Range(”A1”, ”A1”).Value= 4 – przypisanie wszystkim komórkom z bloku wartości 4

Obiekt Workbook i Workbooks
Obiekt Workbook jest konkretnym pakietem arkuszy i zwracany jest przez:
- Właściwość ActiveWorkbook
- metodę Workbooks

Przykład:
Worksheets(“Arkusz1”).Name:=”Nowa nazwa arkusza”

Ciągi obiektów
Workbooks("book1").Worksheets("Sheet1").Name="Nowa nazwa arkusza"
Worksheets("Sheet1").Range("A1").Value = 2

Metody – sposoby wykorzystania
Metodę wywołujemy w następujący sposób:

obiekt.metoda argumenty_metody
składnia przykładowej metody:
obiekt.Add x1, y1, x2, y2 – metoda tworząca nową linię
x1,y1 – współrzędne punktu będącego początkiem linii
x2, y2 – współrzędne końca linii

Przykład:

```
Worksheets("Sheet1").Lines.Add 50,50,100,100
Worksheets("Sheet1").Lines.Add 50,,100,100 – argument fakultatywny
Worksheets("Sheet1").Lines.Add x1:=50, y1:=50, x2:=100, y2:=100
```

Uruchomienie procedury Abc po uaktywnienieniu dowolnego arkusza:
ActiveWorkbook.OnSheetActivate = "Abc"
Uruchomienie procedury Abc po uaktywnienieniu arkusza "Sheet1":
Worksheets("Sheet1").OnSheetActivate = "Abc"

Przypisanie obiektu do zmiennej
Set zmienna = obiekt

Zastosowanie może być następujące:

```
Set Obiekt1 = Workbooks("Sheet1").Worksheets("Sheet1")
Obiekt1.Range("A1").value = 20
Obiekt1.Range("A2").value = 30
Obiekt1.Range("A3").value = 35
```

Lub z wykorzystaniem komendy With

```
With Obiekt1
    .Range("A1").value = 20
    .Range("A2").value = 30
    .Range("A3").value = 35
End With
```

Typy zmiennych:
Boolean; Integer; Long; Single; Double; Currency; String; Data; Object – dowolny obiekt; Array – macierz; Variant – zmienna uniwersalna
Deklarowanie:
Dim a As Integer
Dim d – domyślnie jest typ Variant
Dim a(9) – tablica o indeksach od 0 do 9
Na początku modułu można użyć komendy “Option Base 1” – numerowanie indeksów tablic od 1
Dim b(3 To 4) – tablica dwuelementowa z indeksami od 3 do 4

Można także deklarować zmienne komendą static:
Static e As Integer – zmienna wartość tak zadeklarowanej zmiennej lokalnej (patrz niżej) są pamiętane nawet po zakończeniu wykonywania procedury.

Zmienne mogą być
- lokalne (tylko w obrębie procedury)
- modułu (tylko w module) – zalecane użycie także komendy Private Dim

- publiczne – konieczne użycie komendy Public Dim

Zależy to od miejsca zadeklarowania danej zmiennej. Zmienne modułu i publiczne deklaruje się prze pierwszą procedurą w module.

- Proszę przećwiczyć dołączanie makra do przycisku ekranowego (z palety Formularze) oraz tworzenie procedur (sub ... end sub) i funkcji (public function ... end function)

W następnych punktach proszę intensywnie wykorzystywać help (pomoc) MS Excel z zakresu Visual Basic dla aplikacji. Przećwiczyć też pracę krokową i zakładanie "czujek" (podgląd wartości zmiennych) i punktów przerwań (w pracy krokowej)

a) zapoznać się z deklaracjami zmiennych np.

```
dim i, j as integer
dim znak as string
dim była_kropka as boolean
```

itd.

1. zapoznać się z funkcjami operującymi na ciągach znaków np.

len(napis) – długość ciągu znaków typu string
mid(napis, i, 3) – wyciąga z 'napisu' ciąg o długości 3 od pozycji 'i'
trim(napis) – usuwanie zewnętrznych spacji
oraz łączenie ciągów znaków za pomocą operatora '+'

2. sprawdzanie warunków

```
if ... then ... else ... end if
```

3. instrukcje pętli:

```
for i=1 to j ... next
while i<=len(źródło) ... wend
do while not eof(1) ... loop (czy koniec pliku o numerze 'uchwycie'(handle) 1)
```

4. operacje na plikach tekstowych:

```
open nazwa_pliku for input as #1
line input #1, wiersz
close #1
```

znaczenie tych kolejnych instrukcji: otwarcie pliku o nazwie takiej jak ciąg znaków 'nazwa_pliku' do czytania (uchwyt 1).
wczytanie całego wiersza z pliku (do znaku końca wiersza) do zmiennej wiersz
oraz zamknięcie pliku

5. "wyciągnięcie" zawartości komórki A1 z arkusza "Sheet1": (w kontekście punktu 6)

```
nazwa_pliku=Worksheets("Sheet1").cells(1,1)
```

6. Wpisanie do komórki wartości liczbowej:

```
Worksheets("Arkusz1").Cells(12,i).Value=DaneWejściowe
```

7. wywołanie procedury z innego modułu:

```
Moduł1!.wczytanie_pliku_zapasów(nazwa_pliku)
```

```
Sub Procedura_Podpięta_Pod_Przycisk()
   Dim Kom, Tytuł, Domyślne, nazwa_arkusza, wiersz, kolumna, wartość As String
   Tytuł = "Okno wprowadzania danych do komórki arkusza"
   Domyślne = "1"  ' Ustaw wartość domyślną.
   Kom = "Podaj numer wiersza"
   wiersz = InputBox(Kom, Tytuł, Domyślne)
   Kom = "Podaj LITERĘ kolumny"
   kolumna = InputBox(Kom, Tytuł, Domyślne)
   Kom = "Podaj wartość komórki"
   wartość = InputBox(Kom, Tytuł, Domyślne)
   nazwa_arkusza = ActiveSheet.Name 'zwraca nazwę bieżącego arkusza

   On Error Resume Next    ' Obsługa błędu - skok do następnej instrukcji.
   Err.Clear                        ' Wyczyszczenie cech obiektu Err
   Worksheets(nazwa_arkusza).Range(kolumna + wiersz).Value = wartość
   If Err.Number <> 0 Then
      MsgBox Err, , "Zarejestrowano błąd: ", Err.HelpFile, Err.HelpContext
      Err.Clear    ' Wyczyszczenie cech obiektu Err
   Else
     Odp = MsgBox("komórka (" + kolumna + wiersz + ")=" + wartość, , "Wartość
komórki arkusza " + nazwa_arkusza)
   End If
End Sub
```