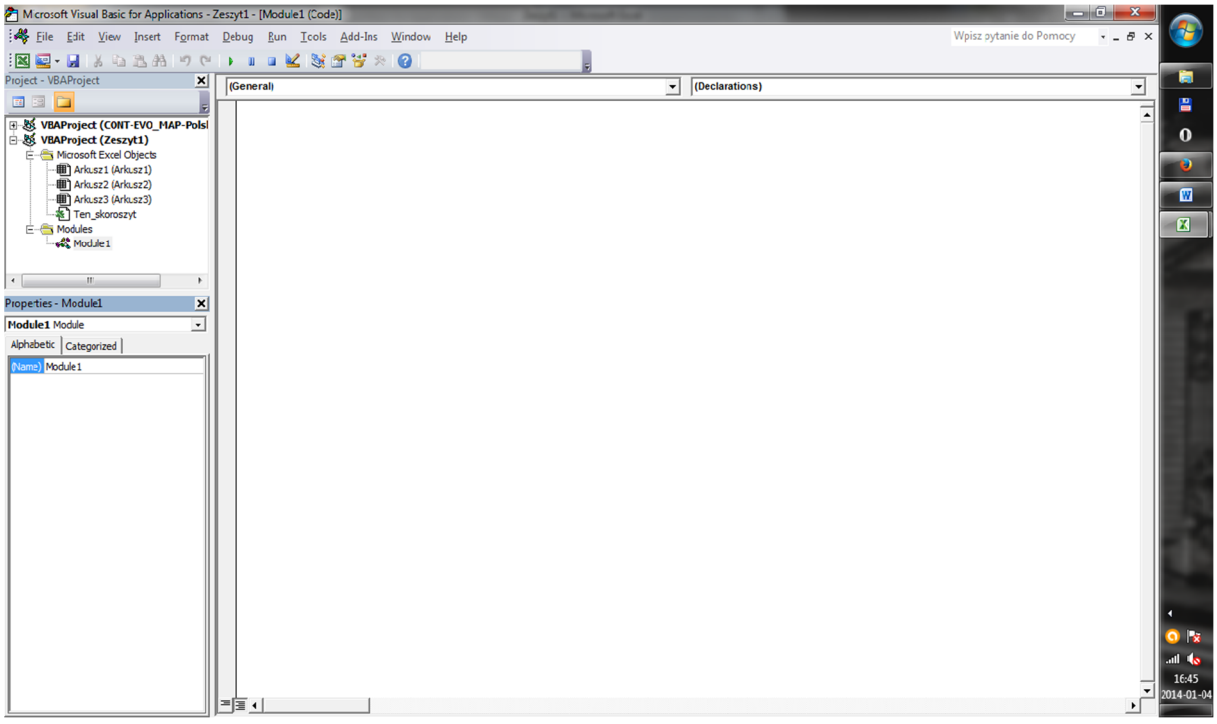
VBA – 1

- VBA
- TYPY PROCEDUR (PODPROGRAM, FUNKCJA)
- ZMIENNE, DEKLARACJA ZMIENNYCH
- FUNKCJA MsgBox

ZFPBIG – LABORATORIUM

makro - napisany lub zarejestrowany program, przechowujący szereg poleceń Microsoft Excel, którego można później użyć jako pojedynczego polecenia. Makra są przeznaczone do automatyzacji złożonych zadań i zmniejszania liczby kroków wymaganych do wykonania często powtarzających się zadań. Makra są rejestrowane w języku programowania Visual Basic for Applications. Makra można także pisać bezpośrednio korzystając z edytora Visual Basic.

Visual Basic for Applications (VBA) – język programowania oparty na Visual Basicu (VB) zaimplementowany w aplikacjach pakietu Microsoft Office oraz kilku innych, jak na przykład AutoCAD i WordPerfect. Ta uproszczona wersja Visual Basica służy przede wszystkim do automatyzacji pracy z dokumentami, na przykład poprzez makropolecenia.
Podstawową różnicą między VBA a VB jest to, że VBA nie pozwala na tworzenie samodzielnych skompilowanych aplikacji typu EXE. Kod programu napisanego w VBA zawsze zawarty jest w dokumencie utworzonym przy pomocy programu obsługującego VBA - na przykład w pliku *.DOC edytora MS Word lub pliku *.XLSx arkusza MS Excel. Program taki wymaga zatem środowiska uruchomieniowego, którym jest zainstalowana na komputerze aplikacja obsługująca dany dokument.
Wyjątkiem symulującym samodzielnie działające aplikacje są pliki utworzone w programie Microsoft Access, które - przy zakupie rozszerzenia Microsoft Office Developer lub innego, pozwalają na uruchamianie plików Accessa na dowolnej ilości komputerów w tzw. Microsoft Access Runtime, bez konieczności wyposażania każdego pojedynczego komputera w pełny pakiet Microsoft Office.



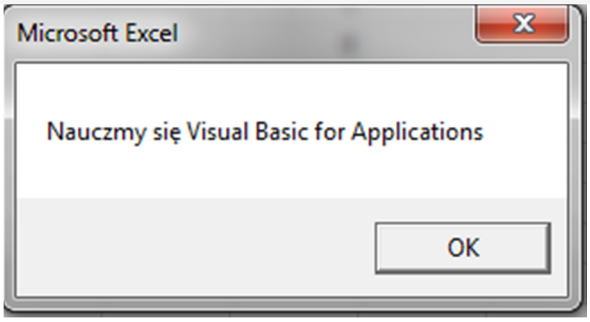
Programy VBA mogą zawierać wiele instrukcji i procedur rozmieszczonych w rożnych modułach a nawet projektach:

PROCEDURA

Procedura jest bardzo ważną częścią programu, ponieważ aby kod mógł być wykonany należy umieścić go w procedurze. Jest to najmniejsza część kodu którą można uruchomić niezależnie od innych części kodu. Procedura składa się z instrukcji deklarującej procedurę, linii kodu wykonywanych wewnątrz procedury oraz instrukcji zamykającej.

Poniżej kod programu, który składa się z jednej procedury o nazwie `Powitanie`, w której umieszczona jest jedna instrukcja. Instrukcja (`MsgBox`) ta wyświetla okienko komunikatu z napisem: `Nauczmy się Visual Basic for Applications`.

```
Sub Powitanie()
 MsgBox " Nauczmy się Visual Basic for Applications "
End Sub
```



TYPY PROCEDUR:

- **Podprogram** - jest to podstawowy typ procedur języka VBA. Procedurę deklarujemy za pomocą słowa kluczowego Sub, instrukcja End Sub zamyka procedurę. Instrukcja deklarująca procedurę kończy się parą nawiasów - można w niej umieszczać parametry podprogramu. Jest to typ procedury, który można uruchomić niezależnie od innych procedur. Procedury tego typu wykonują akcje, lecz nie zwracają wartości. Podprogram może wywołać inną procedurę.

- **Funkcja** - procedura deklarowana za pomocą słowa kluczowego Function, instrukcja End Function kończy funkcję. Funkcja może pobierać argumenty, które są do niej przekazywane np. przez procedurę wywołującą. Procedura Function jest podobna do procedury Sub, jednak w przeciwieństwie do podprogramu zwraca wartość np. do procedury, która ją wywołała.

W edytorze zapisujemy treści procedur i funkcji. Funkcja jest fragmentem programu, który może wykonać obliczenia i zwrócić wartość. Procedura wykonuje najczęściej jakieś operacje, lecz nie zwraca wartości. Dlatego w programach funkcje wykorzystuje się do wykonania obliczeń, a procedury do wykonania określonych działań. W języku VBA funkcję tworzymy wg schematu:

```
Function nazwa_funkcji(lista parametrów) As typ
…
End Function
```

Lista parametrów to dane, które funkcja otrzymuje od wywołującego ją programu. Ma ona następującą postać:

```
nazwa_parametru As typ
```

Typ określa rodzaj informacji, którą funkcja otrzymuje w parametrze lub którą funkcja zwraca jako swój wynik. Oto niektóre typy danych języka VBA:

Boolean	- typ logiczny, przyjmuje tylko dwie wartości: True dla prawdy i False dla fałszu
Integer	- typ całkowity ze znakiem, 16 bitowy, zakres od -32768 do 32767
Long	- typ całkowity ze znakiem, 32 bitowy, zakres od -2147483648 do 2147483647
Double	- typ zmiennoprzecinkowy, dokładność 15 cyfr
String	- typ łańcuchowy, pozwala przetwarzać teksty

REGUŁY OBOWIĄZUJĄCE PRZY NAZYWANIU PROCEDUR I FUNKCJI

- Nazwa procedury nie może zawierać spacji ani żadnych znaków specjalnych oprócz podkreślenia _
- Nazwa musi rozpoczynać się literą.
- Nazwa nie powinna zawierać polskich znaków diaktrycznych (ani jakichkolwiek innych), jedynie litery alfabetu łacińskiego.
- Nazwa nie może być taka sama jak któreś ze słów kluczowych VBA
- Nazwa powinna dokładnie opisywać zadanie wykonywane przez procedurę, tak aby wracając do kodu napisanego na przykład rok wcześniej, na pierwszy rzut oka było wiadomo za co ta procedura jest odpowiedzialna.
- Wielkość liter w nazwach nie ma znaczenia, dlatego przykładowo nazwa `nazwaProcedury` jest dla kompilatora identyczna z nazwą `NAZWAPROCEDURY`.
- Przyjęło się, by nazwy procedur zapisywać małymi literami, a wielkimi literami rozpoczynać tylko poszczególne wyrazy w tej nazwie, np. `NazwaProcedury` (ewentualnie `nazwaProcedury`) zamiast `nazwaprocedury` lub `NAZWAPROCEDURY`.
- Nazwa procedury może liczyć maksymalnie 255 znaków, ale nikt nie nadaje tak długich nazw, gdyż uciążliwe byłoby późniejsze korzystanie z nich.

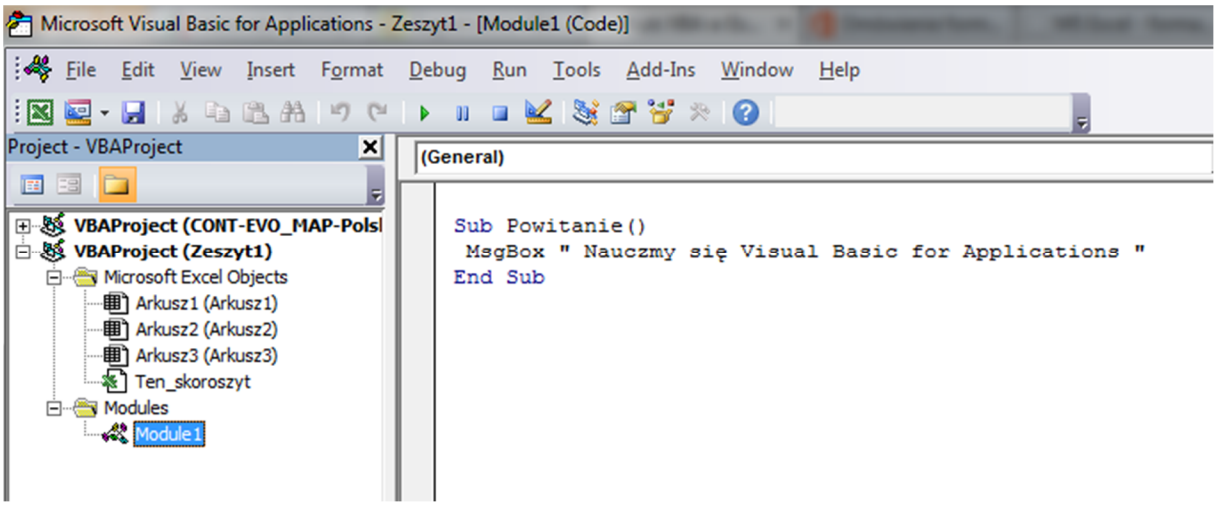
Uwaga!
Funkcja lub procedura nie może być zawarta w innej funkcji lub procedurze!

```
Sub Procedure1
    Sub Procedure2
        '(...)
    End Sub
End Sub
```

Żeby rozpocząć nową funkcję lub procedurę należy zamknąć poprzednią.

ĆWICZENIE 1 - (MOJA PIERWSZA PROCEDURA):

- Uruchamiamy Microsoft Excel.
- Z paska narzędzi **Developer** wybieramy przycisk **Visual Basic**.
- W nowo otwartym oknie Microsoft Visual Basic-Zeszyt1 z menu **View** (Widok) wybieramy opcję **ProjectExplorer** (Eksploator projektu). Powinno się otworzyć okno **Project-VBAProject**
- Następnie z menu **Insert** (Wstaw) wybieramy opcję **Module** (Moduł). Wstawiony obiekt **Module1** (Moduł1) powinien się pojawić oknie Project-VBAProject.
- Powinno się też pojawić okno **Zeszyt1-Module1(Code)** Jeżeli okno się nie pojawi aby je uaktywnić w oknie Project-VBAProject klikamy dwa razy lewym przyciskiem myszy na obiekt **Module1** (Moduł1).
- W nowo otwartym oknie **Zeszyt1-Module1(Code)** piszemy (wstawiamy) kod z przykładu 1. Całość powinna wyglądać podobnie jak na rysunku:



- Zamykamy Edytor VisualBasic (**Alt+Q**) i powracamy do Microsoft Excel.
- Wstawiamy obiekt do którego przypisujemy opracowaną w procedurę o nazwie `Powitanie`, klikamy prawym przyciskiem myszy na nasz wstawiony obiekt i wybieramy opcję Przypisz makro.

ĆWICZENIE 2 – (MOJA PIERWSZA FUNKCJA)

```
Function nazwa_funkcji(lista parametrów) As typ
…
End Function
```

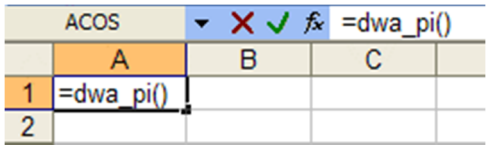
to dane, które funkcja otrzymuje od wywołującego ją programu.
Ma ona następującą postać: `nazwa_parametru As typ,`

Na początek utwórzmy funkcję bez parametrów, która będzie zwracać wartość 2π.

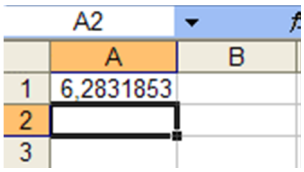
- W edytorze wpisz poniższy kod:

```
Function dwa_pi() As Double
    dwa_pi = 6.28318530717958
End Function
```

- Zwróć uwagę na sposób określania wartości funkcji - nadajemy wartość jej nazwie.
  Typ określa rodzaj informacji, którą funkcja otrzymuje w parametrze lub którą funkcja zwraca jako swój wynik. Oto niektóre typy danych języka VBA:
  - Boolean - typ logiczny, przyjmuje tylko dwie wartości: True dla prawdy i False dla fałszu
  - Integer - typ całkowity ze znakiem, 16 bitowy, zakres od -32768 do 32767
  - Long - typ całkowity ze znakiem, 32 bitowy, zakres od -2147483648 do 2147483647
  - Double - typ zmiennoprzecinkowy, dokładność 15 cyfr
  - String - typ łańcuchowy, pozwala przetwarzać teksty
- Przejdź do arkusza kalkulacyjnego (Alt+F11) i w komórce A1 wpisz formułę:



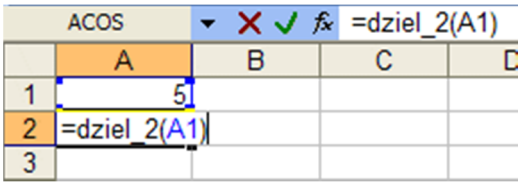
- W wyniku otrzymasz wartość:



- Rozszerzyliśmy zestaw dostępnych funkcji MS-Excel o nową funkcję **dwa_pi()** i możemy z niej korzystać w tym arkuszu jak z każdej innej funkcji wbudowanej.
- Bardziej użyteczna będzie funkcja, która otrzymuje parametr i na jego podstawie wylicza jakiś wynik. Dopisz w edytorze VBA następujący kod (pomiędzy arkuszem a edytorem VBA możesz szybko się przełączać za pomocą klawiszy Alt+F11):

```
Function dziel_2(liczba As Double) As Double
 dziel_2 = liczba / 2
End Function
```

- Przejdź do arkusza. W komórce A1 umieść liczbę 5, a do A2 wpisz formułę:



- Jako wynik w A2 otrzymasz 2,5. Zwróć uwagę, iż do funkcji przekazaliśmy zawartość komórki A1 zgodnie z konwencją arkusza kalkulacyjnego.

Funkcja z instrukcją warunkową

Kolejna funkcja będzie wykorzystywała instrukcję warunkową, która w języku VBA posiada następujące formy:

```
If warunek Then instrukcja
-----------------------------------
If warunek Then instrukcja, gdy warunek prawdziwy Else
instrukcja, gdy warunek fałszywy
-----------------------------------
If warunek Then
  instrukcje, gdy warunek prawdziwy
  ...
Else
  instrukcje, gdy warunek fałszywy
  ...
End If
-----------------------------------
If warunek1 Then
  instrukcje, gdy warunek1 prawdziwy
  ...
Elseif warunek2 Then
  instrukcje, gdy warunek2 prawdziwy
  ...
dalsze bloki Elseif
  ...
Else
  instrukcje, gdy żaden z powyższych warunków nie był
prawdziwy
End If
```

Część zaczynająca się od Else jest opcjonalna jeżeli interesuje nas tylko jeden warunek nie musimy z niej korzystać.

Warunek jest wyrażeniem logicznym, które daje w wyniku wartość **True** lub **False**. W warunkach można stosować operatory porównań:

Operator	Opis
=	równe, np. a = 5
<>	różne, np. a <> b
>	większe, np. a > 10
>=	większe lub równe, np. a >= b - 1
<	mniejsze, np. a < 10
<=	mniejsze lub równe, np. a + b <= 10

W warunkach mogą również być stosowane dwie funkcje logiczne:

- warunek1 **AND** warunek2 - wynik jest prawdziwy, gdy oba warunki są prawdziwe, inaczej wynik jest fałszem
- warunek1 **OR** warunek2 - wynik jest fałszywy, gdy oba warunki są fałszywe, inaczej wynik jest prawdą

ĆWICZENIE 3 - (FUNKCJA Z INSTRUKCJĄ WARUNKOWA)

- Napiszmy prostą funkcję, która będzie przetwarzała ocenę liczbową na ocenę słowną. Obowiązują następujące przedziały:

Przedział	Ocena
5,5...6	cel
4,4...5,5	bdb
3,5...4,5	db
2,5...3,5	dst
1,5...2,5	dop
< 1,5	ndst

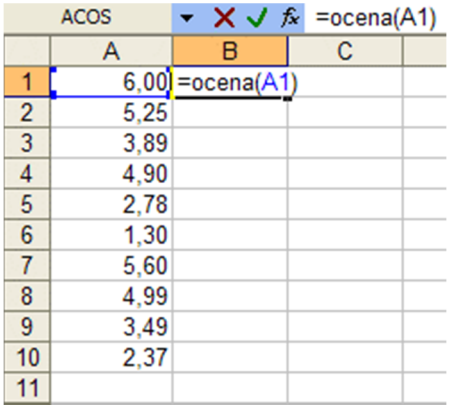
- W oknie edytora VBA dopisz poniższy kod:

```
Function ocena(w As Double) As String
        If w > 5.5 Then
          ocena = "cel"
        Elseif w > 4.5 Then
          ocena = "bdb"
        Elseif w > 3.5 Then
          ocena = "db"
        Elseif w > 2.5 Then
          ocena = "dst"
        Elseif w > 1.5 Then
          ocena = "dop"
        Else
          ocena = "ndst"
        End If
End Function
```

- Przejdź do arkusza, w komórkach od A1 do A10 umieść oceny liczbowe:

	A
1	6,00
2	5,25
3	3,89
4	4,90
5	2,78
6	1,30
7	5,60
8	4,99
9	3,49
10	2,37

- Przejdź do komórki B1 i wpisz w niej formułę:



- Skopiuj komórkę formułę z B1 na pozostałe komórki od B2 do B10. Arkusz wypisze oceny słowne:

	A	B
1	6,00	cel
2	5,25	bdb
3	3,89	db
4	4,90	bdb
5	2,78	dst
6	1,30	ndst
7	5,60	cel
8	4,99	bdb
9	3,49	dst
10	2,37	dop
11		

- Wg podobnych zasad możesz utworzyć również podobne funkcje, które muszą zamieniać liczby na słowa.

ZMIENNE

Zmienna to taki element programu, do którego można przypisać jakąś wartość.

Zmienna - opatrzone nazwą miejsce w pamięci do przechowywania danych, które mogą ulegać modyfikacjom w trakcie wykonywania programu. Każda zmienna zaopatrzona jest w unikatową nazwę, która identyfikuje ją w obrębie danego zakresu. Typ danych może być określony lub nie. Nazwy zmiennych muszą zaczynać się literą, muszą być unikatowe w obrębie swego zakresu, nie mogą być dłuższe niż 255 znaków i nie mogą zawierać kropki ani znaku deklarującego typ.

Deklarowanie zmiennej jest to operacja polegająca na nadaniu jej nazwy oraz określeniu typu i dostępności. Jeżeli zadeklarujemy zmienną to jednocześnie przydzielamy jej pamięć. Zmienną możemy zadeklarować wewnątrz konkretnej procedury lub w sekcji deklaracji modułu kodu. Miejsce deklaracji ma wpływ na dostępność danej zmiennej.

Do deklarowania zmiennej zazwyczaj stosowane jest słowo kluczowe **Dim**. Instrukcja deklaracji w której użyliśmy słowa kluczowego **Dim** może być umieszczona wewnątrz procedury, wówczas zostanie utworzona zmienna na poziomie procedury. Jeżeli natomiast deklaracja zostanie umieszczona na początku modułu w sekcji deklaracji, utworzona będzie zmienna na poziomie modułu. Poniżej przedstawiam przykład deklaracji w którym deklarujemy zmienną o nazwie `MojaLiczba`.

```
Dim MojaLiczba
```

Oprócz deklarowania zmiennych za pomocą słowa kluczowego **Dim**, w deklarowaniu zmiennych możemy użyć słów kluczowych Private, Public, oraz Static. Słowa te służą nie tylko do deklarowania zmiennych ale i do określania ich zakresu. Zakres zależy od miejsca, w którym zmienna jest zadeklarowana, w sekcji deklaracji modułu czy wewnątrz konkretnej procedury oraz za pomocą jakiego słowa kluczowego została zadeklarowana.

- Dim MojaLiczba
  Instrukcja ta może być umieszczona wewnątrz procedury, wówczas zostanie utworzona zmienna na poziomie procedury. Jeżeli natomiast deklaracja zostanie umieszczona na początku modułu, w sekcji deklaracji, utworzona będzie zmienna na poziomie modułu.
- Private MojaZmienna
  Stosowana na poziomie modułu do deklaracji zmiennych prywatnych oraz do przydziału pamięci. Zmienne te są dostępne tylko w tym module, w którym zostały zadeklarowane. Słowa kluczowego Private nie można użyć wewnątrz procedury.
- Public WynikRazem
  Stosowana do deklarowania zmiennych publicznych na poziomie modułu. Zmienne zadeklarowane za pomocą instrukcji Public są dostępne dla wszystkich procedur we wszystkich modułach wszystkich projektów. Słowo kluczowe Public należy stosować wyłącznie w sekcji deklaracji modułu.
- Static Licznik
  Wykorzystywana na poziomie procedury do deklaracji zmiennych i przydziału pamięci. Zadeklarowana w ten sposób zmienna zachowuje swoją wartość między wywołaniami procedury. Zmienne statyczne można deklarować tylko wewnątrz procedur.

Przypisanie wartości do zmiennej (przykłady):

```
MojaWartosc = 3
Przywitanie = "Pozdrawiam wszystkich"
Nazwisko = InputBox("Jak się nazywasz?")
MojaLiczba = Int((6 * Rnd) + 1)
```

PO CO DEKLAROWAĆ ZMIENNE?

1. Zadeklarowane zmienne zajmują mniej miejsca w pamięci komputera i skomplikowane makra będą działać szybciej, jeśli zadeklarujemy zmienne.
2. Zadeklarowanie wszystkich zmiennych na początku makra pozwoli nam utrzymać porządek i lepiej je kontrolować, szczególnie jeśli ich nazwy są opisowe i dodamy do zmiennych komentarze, łatwiej też będzie zrozumieć makro innej osobie.
3. Deklarowanie zmiennych w połączeniu z poleceniem Option Explicit, które wymusza zadeklarowanie wszystkich używanych zmiennych, chroni nas przed literówkami w nazwach zmiennych, bez tego polecenia, zmienna z literówką zostanie potraktowana jako nowa zmienna.
4. Deklarowanie i opisywanie zmiennych ułatwi innym pracę z naszymi makrami, jeżeli zamierzamy się nimi dzielić
5. Zawansowani programiści źle patrzą na makra (nawet te krótkie i proste) nie posiadające zadeklarowanych zmiennych :-)

Typ danych	Liczba używanych bajtów	Zakres wartości
Boolean	2	wartość logiczna prawda lub fałsz
		wartości: Ture, False
Integer	2	liczba całkowita od -32 768 do 32 767
		wartość domyślna 0
Long	4	liczba całkowita
		od -2 147 483 648 do 2 147 483 647
		wartość domyślna 0
Single	4	liczba rzeczywista
		od -3,402823E38 do -1,401298E-45 i od 1,401298E-45 do 3,402823E38 wartość domyślna 0
Double	8	liczba rzeczywista dwukrotnie większa od Single
		od -1,79769313486232e+308 do -4,9406564581247e-324
		i od 4,9406564581247e-324 do 1,79769313486232e+308
		wartość domyślna 0
String	1 + 1 na każdy znak	tekst maksymalnie do 32 767 znaków
		wartość domyślna: "" - pusty
Date	8	data i czas w przedziale od 1 stycznia 100 r. do 31 grudnia 9999 roku wartość domyślna 0
Currency	8	typ walutowy, liczby rzeczywiste od -922 337 203 658 477,5808 do 922 337 203 685 477,5807
		wartość domyślna 0
Variant	1 do 8	dowolny typ identyfikowany przez Visual Basic w zależności od pierwszego użycia tej zmiennej
		wartość domyślna: EMPTY (puste) domyślny typ dla nie zadeklarowanych zmiennych
Zdefiniowany przez użytkownika	zależy od definicji	zależny od definicji

ĆWICZENIE 4 (DEKLARACJA ZMIENNYCH)

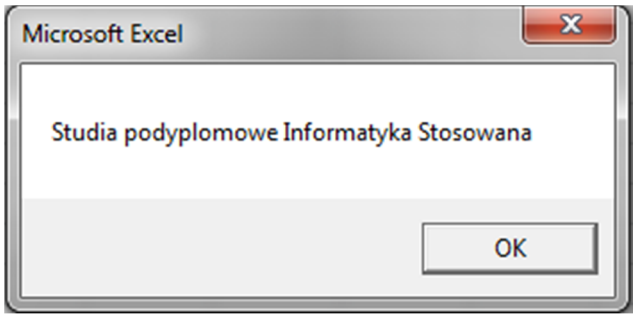
- Podstawową formą deklarowania jest użycie słowa kluczowego “Dim” przed nazwą zmiennej deklarowanej bezpośrednio w procedurze lub funkcji
- Napiszmy procedurę, w której lokalnie zadeklarujemy zmienną1 jako ciąg znaków.

```
Sub sp()

   Dim zmienna1 As String
   zmienna1 = " Studia podyplomowe Informatyka Stosowana"
   MsgBox zmienna1

End Sub
```

- Przypisz makro nazwie sp do obiektu i uruchom jego działanie, wynik na rysunku poniżej.



Ćwiczenie 5 (deklaracja zmiennych)

```
Sub procedura_glowna()
      Dim zmienna as Double
      zmienna = 77.1234
      Msgbox zmienna          'wyświetli 77.1234
      Call procedura1         'wywołanie procedura1 'wyświetli 5
      Call procedura2          'wywołanie   procedura2   'wyświetli
      "tekst"
End Sub

Sub procedura1()
      Dim zmienna As Integer
      zmienna = 5
      MsgBox zmienna           'wyświetlenie komunikatu 5
End Sub

Sub procedura2()
      Dim zmienna As String
      zmienna = "tekst"
      MsgBox zmienna           'wyświetlenie komunikatu "tekst"
End Sub
```

ĆWICZENIE 6 (WYWOŁANIE PODPROGRAMU)

```
Sub ObliczPole()
      Dim wartość, pole
      wartość = InputBox("Podaj długość boku kwadratu do obliczenia pola
      powierzchni")
  If IsNumeric(wartość) = True Then
      If wartość > 0 Then
            pole = PoleKwadratu(wartość) ' wywołujemy funkcje PoleKwadratu.
            MsgBox "Pole kwadratu wynosi " & pole
      Else
            BłędnaWartość ' wywołujemy podprogram BłędnaWartość.
      End If
      Else
            BłędnaWartość ' wywołujemy podprogram BłędnaWartość.
      End If
End Sub


Function PoleKwadratu(bok)
       PoleKwadratu = bok * bok
End Function

Sub BłędnaWartość()
       MsgBox "Wprowadź wartość numeryczną większą od zera"
End Sub
```

Uwaga!
Jest wiele sposobów uruchomienia podprogramu:

- Podprogram można wywołać (uruchomić) z innego podprogramu. Aby wywołać podprogram z innego podprogramu należy w procedurze wywołującej wpisać instrukcję zawierającą jego nazwę.
- Jeżeli podprogram przez nas napisany nie posiada parametrów możemy wywołać go tak jak uruchamia się makro. Będąc w arkuszu Excela naciskamy kombinacje klawiszy **Alt + F8**, w nowo otwartym oknie wybieramy nazwę odpowiedniej procedury a następnie przycisk Uruchom.

ĆWICZENIE 7

- Uruchom Microsoft Excel.
- W **Przyborniku formantów** wyszukaj i kliknij na ikonę **Przycisk polecenia** a następnie miejsce w arkuszu gdzie chcesz go umieścić. Ikona Tryb projektowania w przyborniku powinna się uaktywnić.
- Kliknij dwa razy lewym przyciskiem myszy na wstawiony przycisk (ikona Tryb projektowania w przyborniku powinna być aktywna). Powinien uruchomić się Edytor Visual Basic z widocznym oknem Kod programu (Code), w oknie tym zawarta powinna być deklaracja procedury Click naszego Przycisku polecenia.
- W procedurze zdarzenia Click Przycisku polecenia wpisz kod:

```
If Range("A1").Value = 0 Then
   Range("A2").Value = "wartość wynosi zero"
 Else
   Range("A2").Value = "wartość jest różna od zera"
 End If
```

- Zamknij Edytor Visual Basic **Alt+Q** i powróć do arkusza Excela.
- Następnie wyłącz tryb projektowania (jeżeli jest aktywny) klikając na ikonę **Zakończ tryb projektowania** w Przyborniku formantów.
- Wpisz wartość **0 (zero)** do komórki A1 arkusza następnie kliknij na Przycisk polecenia.
- Celem ponownego przetestowania, wpisz inną **wartość numeryczną** do komórki A1 arkusza i ponownie kliknij na przycisk.

ĆWICZENIE 8

- Jak w ćwiczeniu 7 wprowadź kolejny przycisk i w procedurze zdarzenia Click Przycisku polecenia wpisz kod:

```
If Range("C1").Value = 0 Then
  Range("C2").Value = "wartość wynosi zero"
 Else
  If Range("C1").Value > 0 Then
   Range("C2").Value = "wartość dodatnia"
  Else
   Range("C2").Value = "wartość ujemna"
  End If
 End If
```

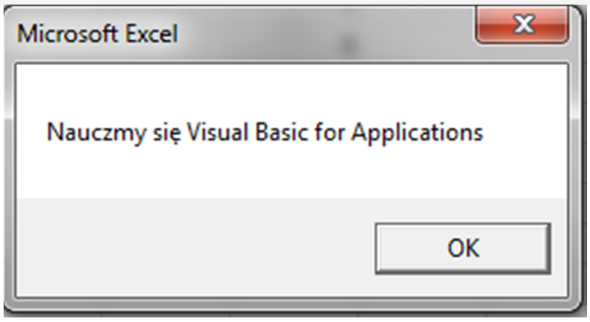
- Aby przetestować jego działanie pamiętaj, że analizowany argument wprowadzony jest do komórki o adresie C1 a wynik powinien pojawić się w komórce o adresie C2

FUNKCJA MSGBOX

Funkcja MsgBox umożliwia wyświetlanie komunikatów na ekranie, w zależności od wybranych parametrów może wyświetlać różne przyciski, to co użytkownik wybierze ma wpływ na to jakie czynności wykona makro. Funkcja MsgBox jest bardzo użyteczna, ponieważ umożliwia łatwą komunikację z użytkownikiem makra.

PATRZ ĆW. 1.

```
Sub Powitanie()
 MsgBox " Nauczmy się Visual Basic for Applications "
End Sub
```



Ale funkcja to oferuje znacznie więcej niż tylko wyświetlanie komunikatów, które można tylko zaakceptować.

Składnia:

MsgBox (prompt [, buttons] [, title] [, helpfile, context])

prompt – to tekst jaki ma się wyświetlać, to jedyna konieczna składowa tej funkcji
buttons – za pomocą tej stałej wybieramy jakie przyciski mają zostać wyświetlone
title – tytuł okienka z naszym tekstem, jeżeli nie podamy tytułu, zostanie wyświetlony napis: Microsoft Excel
helpfile – pomoc, opcja nie jest używana przy prostych makrach
context – kontekst, opcja ta nie jest używana przy prostych makrach

Buttons – wybór przycisków i ikon jakie mogą być wyświetlane w oknie dialogowym zawiera poniższa tabela

Stała	Opis
vbOKOnly	Wyświetlany jest tylko przycisk OK., tak samo jak wtedy gdy nic nie zostanie wybrane.
vbOKCancel	Wyświetlane są przyciski OK i Anuluj
vbAbortRetryIgnore	Wyświetlane są przyciski Przerwij, Ponów i Ignoruj
vbYesNoCancel	Wyświetlane są przyciski Tak, Nie i Anuluj
vbYesNo	Wyświetlane są przyciski Tak i Nie
vbRetryCancel	Wyświetlane są przyciski Ponów i Anuluj
vbCritical	Wyświetlana jest ikona Komunikat Krytyczny 
vbQuestion	Wyświetlana jest ikona Pytanie Ostrzegawcze 
vbExclamation	Wyświetlana jest ikona Komunikat Ostrzegawczy 

vbInformation	Wyświetlana jest ikona Komunikat Informacyjny 
vbDefaultButton1	Pierwszy przycisk jest wybrany jako domyślny
vbDefaultButton2	Drugi przycisk jest wybrany jako domyślny
vbDefaultButton3	Trzeci przycisk jest wybrany jako domyślny
vbDefaultButton4	Czwarty przycisk jest wybrany jako domyślny
vbApplicationModal	Użytkownik musi odpowiedzieć na pytanie przed kontynuowaniem pracy z Excelem
vbSystemModal	Okienko komunikatu będzie na wierzchu nawet po przejściu do innych aplikacji.
vbMsgBoxHelpButton	Dodaje przycisk Pomoc

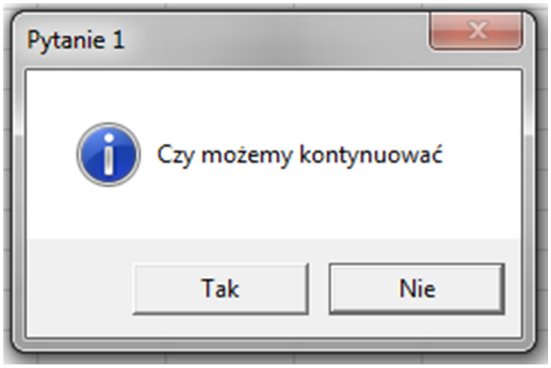
MsgBox (prompt [, buttons] [, title] [, helpfile, context])

MsgBox („_____”, _____+_____+_____ ,„_____”)

ĆWICZENIE 9

```
Sub MsgBox_przyklad_1
   MsgBox „Czy możemy zaczynać?” ,vbYesNo + vbQuestion +
   vbDefaultButton2, „Pytanie 1”
End Sub
```

Powyższe makro wyświetli okno takie jak poniżej.



vbYesNo – spowodowało wyświetlenie przycisków Tak i Nie.
vbQuestion – dodało ikonę ze znakiem zapytania
vbDefaultButton2 - Drugi przycisk jest wybrany jako domyślny (Nie)
Połączenie 3 stałych następuje przez wpisanie pomiędzy nimi znaku +
Po drugim przecinku znajduje się tytuł okna „Pytanie 1”.

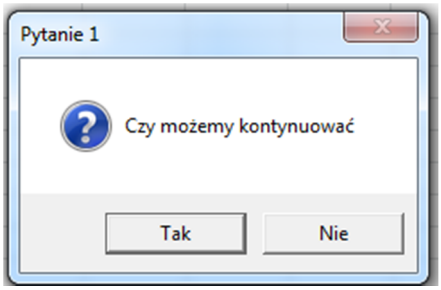
Aby to co wybierze użytkownik mogło zostać wykorzystane w kodzie makra najwygodniej będzie przypisać wartość funkcji MsgBox do zmiennej. Funkcja MsgBox przyjmuje wartości liczbowe od 1 do 7 przedstawione w poniższej tabeli.

Stała	Wartość	Opis
vbOK	1	Użytkownik wybrał OK
vbCancel	2	Użytkownik wybrał Anuluj
vbAbort	3	Użytkownik wybrał Przerwij
vbRetry	4	Użytkownik wybrał Ponów
vbIgnore	5	Użytkownik wybrał Ignoruj
vbYes	6	Użytkownik wybrał Tak
vbNo	7	Użytkownik wybrał Nie

W poniższym przykładzie wynik tego co wybierze użytkownik makra przypisywane jest do zmiennej ‘odpowiedź’.
W kolejnej linii kodu jeśli odpowiedź wynosi 6 (użytkownik wybrał Tak) wyświetlany jest komunikat „Wybrano: Tak” w przeciwnym razie wyświetlany jest komunikat „Wybrano: Nie”.

ĆWICZENIE 10

```
Sub MsgBox_Przyklad_2()
      Odpowiedz = MsgBox("Czy możemy kontynuować", vbYesNo +
      vbQuestion, "Pytanie 1")
      If Odpowiedz = 6 Then MsgBox "Wybrano: Tak" Else MsgBox
      "Wybrano: Nie"
End Sub
```



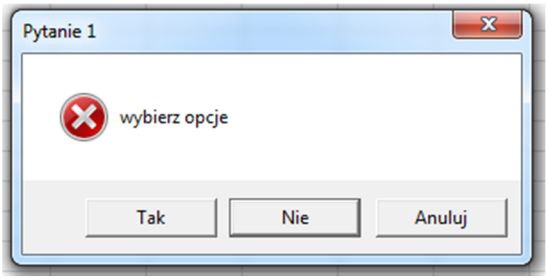
Wprowadzona procedura wyświetli zapytane „Czy możemy kontynuować”.
Wyświetlone okno na niebieskim pasku tytułu będzie mieć zamieszczony tekst: „Pytanie 1”. Jest to tekst zamieszczony w opisie funkcji MsgBox po drugim przecinku.
Wyświetlone zostaną przyciski: yes, no, obok pytania pojawi się ikona pytania ostrzegawczego.
Po udzieleniu odpowiedzi, w zależności od jej wartości pojawi się kolejny komunikat informujący o udzielonej odpowiedzi.

ĆWICZENIE 11

```
Sub MsgBox_Przyklad_3()
Pytanie1 = MsgBox("wybierz opcje", vbYesNoCancel +
vbDefaultButton2 + vbCritical, "Pytanie 1")

If Pytanie1 = 6 Then
    MsgBox "użytkownik wybrał TAK"
ElseIf Pytanie1 = 2 Then
   MsgBox "użytkownik wybrał CANCEL"
Else
MsgBox "użytkownik wybrał NIE"
End If

End Sub
```



vbYesNo – spowodowało wyświetlenie przycisków Tak, Nie, Anuluj
vbCritical – dodało ikonę ze znakiem komunikatu krytycznego
vbDefaultButton2 - Drugi przycisk jest wybrany jako domyślny (Nie)
Po drugim przecinku znajduje się tytuł okna „Pytanie 1".

INPUTBOX

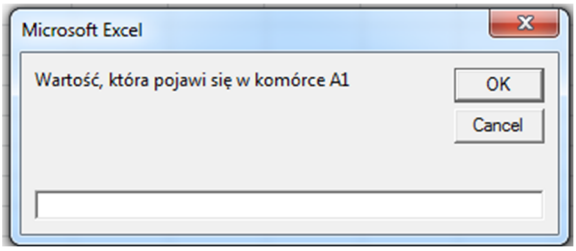
Obok omówionej funkcji `MsgBox`, wyświetlającej komunikaty i umożliwiającej użytkownikom podejmowanie prostych decyzji (*Tak/Nie*, *Zatwierdź/Anuluj*), język VBA udostępnia jeszcze drugą funkcję służącą do komunikacji z użytkownikami - `InputBox`.

Zadaniem funkcji `InputBox` jest pobieranie od użytkowników bardziej złożonych informacji, takich jak np. haseł, nazw, dat lub wartości liczbowych.

ĆWICZENIE 12

```
Sub pobieranieWartosci()
    Dim tekst As String

    tekst = InputBox("Wartość, która pojawi się w komórce A1")
    Cells(1,1) = tekst
End Sub
```



Postać ogólna funkcji `InputBox` przedstawia się następująco:

```
InputBox(Prompt As String, _
    Optional Title As String, Optional Default, _
    Optional XPos As Single, Optional YPos As Single, _
    Optional HelpFile, Optional Context) As String
```

Argumenty `Prompt` oraz `Title` zostały już szczegółowo omówione przy okazji funkcji MsgBox. W funkcji `InputBox` ich działanie jest identyczne: argument `Prompt` odpowiada więc za treść komunikatu dla użytkownika (w tym przypadku będzie to instrukcja dotycząca pobieranej wartości, np.: *Wpisz hasło*), natomiast wartość argumentu `Title` pojawi się na górnym pasku okna wyświetlanego na ekranie.

Argumenty `XPos` i `YPos` odpowiadają za początkową pozycję okna `InputBox` na ekranie (po jego wyświetleniu możesz je przesunąć w dowolne miejsce ekranu).

Argument `XPos` określa odległość okna `InputBox` od lewej krawędzi ekranu, natomiast argument `YPos` jego odległość od górnej krawędzi. Obie te wartości wyrażane są w jednostkach zwanych *twipsami* (1440 twipsów = 1 cal).
W sytuacji, gdy wartości `XPos` i `YPos` zostaną pominięte przy wywoływaniu tej funkcji, okno `InputBox` zostanie wyświetlone na środku ekranu w poziomie i mniej więcej w jednej trzeciej wysokości ekranu.

C.D.N

Język VBA posiada kilka instrukcji do tworzenia pętli warunkowych:

- **Instrukcja If... Then... Else** - najczęściej stosowana instrukcja warunkowa.
- **Instrukcja Select Case** - jest to inna droga realizacji procesu podjęcia decyzji w programie.
- **Pętle warunkowe Do...Loop** - bardzo wygodnym narzędziem są pętle, służą one do wielokrotnego wykonywania danego bloku kodu. Instrukcji Do...Loop użyjemy jeżeli nie wiemy ile razy pętla ma być wykonana. Jest to pętla warunkowa, w której kluczową cechą jest warunek.
- **Instrukcja For... Next - pętla For... Next** powtarza blok instrukcji określoną liczbę razy, stosujemy ją jeżeli z góry wiadomo ile razy pętla ma być wykonana.
- **Instrukcja For Each... Next** - pętla służąca do wykonywania operacji na obiektach kolekcji.