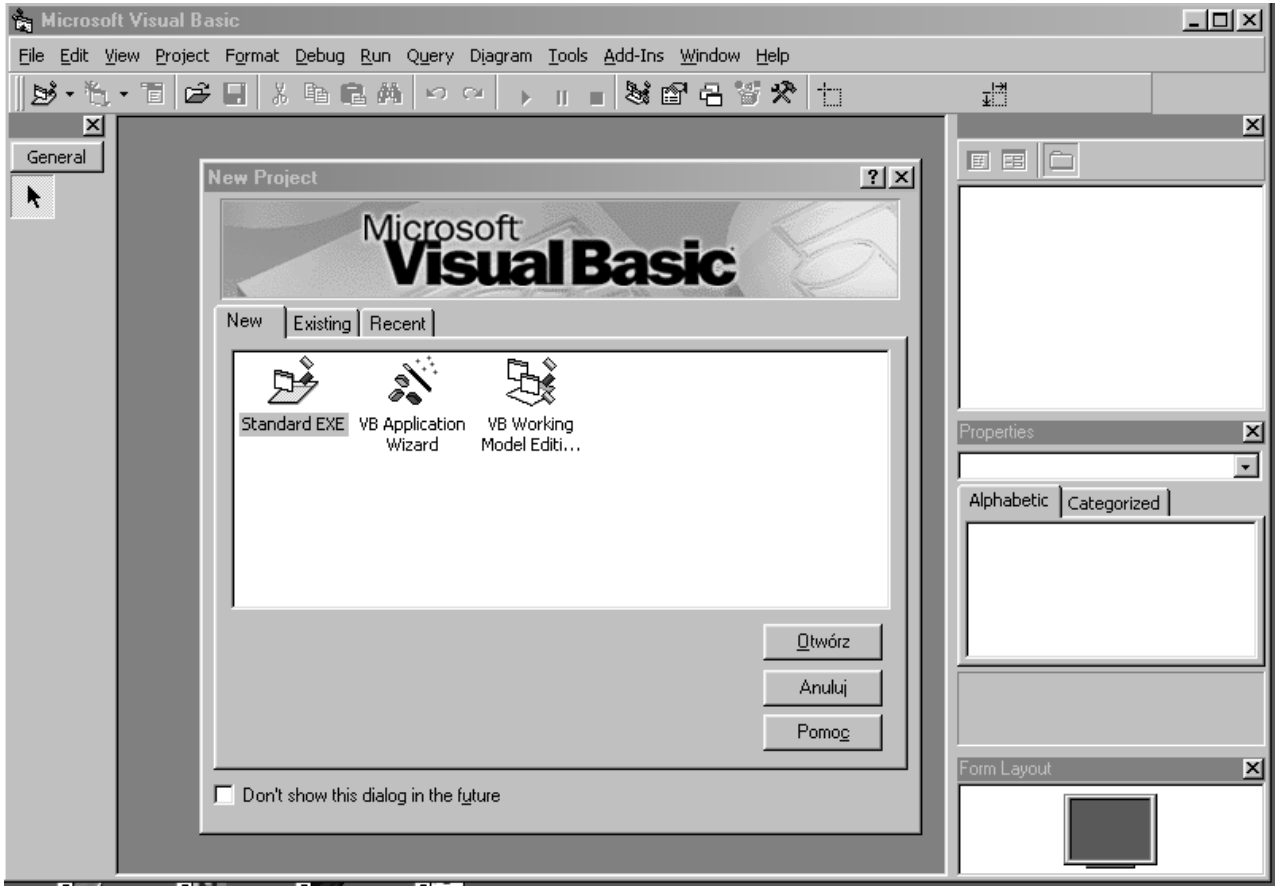
Technologia informacyjna

programowanie

Janusz Uriasz

2. Programowanie

2.1. Struktura programu, edycja, kompilacja, uruchomienie programu. Śledzenie programu



VB- cd
Składowe programu

- **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 przynajmniej jednej instrukcji umieszczonej między dwiema specjalnymi instrukcjami: z których pierwsza z nich deklaruje procedurę a ostatnia ją zamyka.
- **Moduł** zawiera z jedną lub więcej procedur oraz sekcje deklaracji w której umieszczamy instrukcje wspólne dla w wszystkich procedur tego modułu. Możemy wyróżnić dwa rodzaje modułów: moduł standardowy i moduł klasy.
- **Projekt** obejmuje wszystkie moduły, formularze, obiekty aplikacji macierzystej dokumentu oraz sam dokument

VB- cd
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 procedurę. 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.

Rozwiązanie postawionego zadania za pomocą programu w języku VB, VBA:

- zaprojektowanie i utworzenie interfejsu użytkownika, za pomocą którego program będzie komunikować się ze światem zewnętrznym;
- opracowanie algorytmu rozwiązania postawionego problemu oraz napisania na tej podstawie kodu źródłowego programu;
- uruchomienie napisanego programu i zaprojektowanego interfejsu użytkownika oraz poprawienia ewentualnych błędów,
- (**VB**) skompilowanie projektu do postaci programu wykonawczego (jego działanie wymaga obecności VB w pamięci).

Programowanie sprowadza się do:

- projektowania interfejsu, poprzez tworzenie, modyfikowanie formularza (formularzy) i rozmieszczanie formantów,
- modyfikowania właściwości utworzonych obiektów (ustawienie na potrzeby aplikacji),
- przypisania kodu każdemu elementowi (obiektowi).

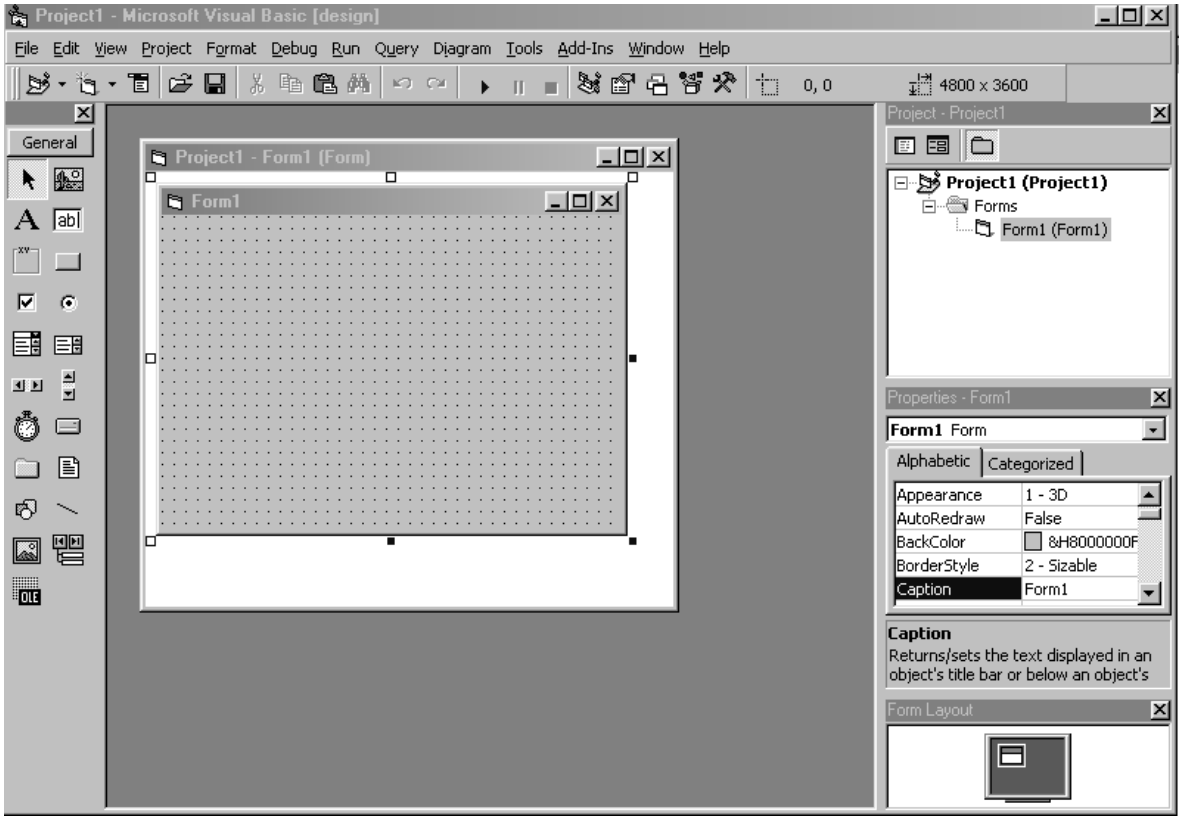
Uruchomienie programu

Śledzenie wykonywania programu

Program napisany w języku Visual Basic komunikuje się z użytkownikiem poprzez interfejs.

Często **interfejs użytkownika** stanowi odpowiednio zaprojektowany **formularz**. W formularzu umieszczone mogą być obiekty – kontrolki- takie jak **przyciski, pola tekstowe, menu**,

Obiekty pozwalają użytkownikowi na manipulowanie nimi w celu wykonania odpowiednich poleceń czy spowodowania odpowiednich zachowań.



2.2. Typy danych. Struktury danych. Zmienne. Instrukcje wejścia/wyjścia. Instrukcja podstawienia

Typy danych

Data type	Storage size	Range
Byte	1 byte	0 to 255
Boolean	2 bytes	**True** or **False**
Integer	2 bytes	-32,768 to 32,767
Long (long integer)	4 bytes	-2,147,483,648 to 2,147,483,647
Single (single-precision floating-point)	4 bytes	-3.402823E38 to -1.401298E-45 for negative values; 1.401298E-45 to 3.402823E38 for positive values
Double (double-precision floating-point)	8 bytes	-1.79769313486231E308 to -4.94065645841247E-324 for negative values; 4.94065645841247E-324 to 1.79769313486232E308 for positive values
Currency (scaled integer)	8 bytes	-922,337,203,685,477.5808 to 922,337,203,685,477.5807
Decimal	14 bytes	+/-79,228,162,514,264,337,593,543,950,335 with no decimal point; +/-7.9228162514264337593543950335 with 28 places to the right of the decimal; smallest non-zero number is +/-0.0000000000000000000000000001

2.2. Typy danych. Struktury danych. Zmienne. Instrukcje wejścia/wyjścia. Instrukcja podstawienia

Typy danych

Date	8 bytes	January 1, 100 to December 31, 9999
Object	4 bytes	Any **Object** reference
String (variable-length)	10 bytes + string length	0 to approximately 2 billion
String (fixed-length)	Length of string	1 to approximately 65,400
Variant (with numbers)	16 bytes	Any numeric value up to the range of a **Double**
Variant (with characters)	22 bytes + string length	Same range as for variable-length **String**
User-defined (using **Type**)	Number required by elements	The range of each element is the same as the range of its data type.

Note Arrays of any data type require 20 bytes of memory plus 4 bytes for each array dimension plus the number of bytes occupied by the data itself. The memory occupied by the data can be calculated by multiplying the number of data elements by the size of each element. For example, the data in a single-dimension array consisting of 4 **Integer** data elements of 2 bytes each occupies 8 bytes. The 8 bytes required for the data plus the 24 bytes of overhead brings the total memory requirement for the array to 32 bytes.

A **Variant** containing an array requires 12 bytes more than the array alone.

Note Use the **StrConv** function to convert one type of string data to another.

Zmienne

Programy manipulują danymi, które są przechowywane w zmiennych.

Zmienne mają różne atrybuty i mogą przechowywać różne typy danych (liczby, tekst, data, ..)

Przez zdeklarowanie zmiennej jako określonego typu danych określa się dla Visual Basic typ danych przechowywanych przez zmienną.(sposób interpretacji danej przechowywanej w pamięci).

UWAGA!!!
Nazwy zmiennych mogą zawierać tylko litery, cyfry (nigdy jako pierwszy znak) oraz znak '_'

Deklarowanie zmiennych

Dim, Private, Public

Dim liczba As Integer
Public liczba As Integer
Private liczba_1, liczba_2 As Single

VB- cd

zmienne - deklaracja

- 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.

VB- cd

stałe- deklaracja

- **stała** - element o nadanej nazwie, który zachowuje stałą wartość przez cały czas działania programu. Stała może być ciągiem znaków lub literałem numerycznym, inną stałą lub dowolną kombinacją zawierającą operatory arytmetyczne i logiczne, z wyjątkiem operatora Is oraz operatora potęgowania. Każda aplikacja główna może definiować własny zestaw stałych. Dodatkowe stałe mogą być definiowane przez użytkownika za pomocą instrukcji Const. Stałych można użyć w dowolnym miejscu kodu programu zamiast ich rzeczywistych wartości

VB- cd

stałe- deklaracja

- Const LiczbaPi = 3.14159265359 'za pomocą słowa kluczowego Const deklarujemy stałą o nazwie LiczbaPi, która w naszym przypadku przechowuje właśnie wartość liczby Pi. Jeżeli instrukcje deklarującą stałą umieścimy wewnątrz procedury to stała ta dostępna jest tylko wewnątrz tej procedury. Jeżeli zaś instrukcję tą umieścimy poza procedurą w sekcji deklaracji modułu to stała ta będzie dostępna dla wszystkich procedur danego modułu.
- Public Const LiczbaPi = 3.14159265359 'stałe zadeklarowane na poziomie modułu są domyślnie prywatne czyli widoczne i dostępne tylko wewnątrz tego modułu. Aby zadeklarować stałą publiczną czyli dostępną we wszystkich procedurach wszystkich modułów, należy poprzedzić instrukcję Const słowem kluczowym Public. Stałe publiczne możemy deklarować tylko w sekcji deklaracji modułu standardowego. Nie można deklarować stałych publicznych w procedurach czy modułach klas.
- Private Const LiczbaPi = 22/7 'możliwe jest także jawne zadeklarowanie stałej prywatnej, przez poprzedzenie instrukcji Const słowem kluczowym Private. Słowo kluczowe Private wykorzystujemy do jawnego zadeklarowania stałej prywatnej, celem poprawienia czytelności kodu. Stosowanie go w procedurach jest niedozwolone.

VB- cd

tablice - deklaracja

- **tablica** - zbiór kolejno indeksowanych elementów mających ten sam wewnętrzny typ danych. Każdy element tablicy posiada unikatowy numer indeksu. Przeprowadzenie zmian dla jednego elementu tablicy nie wpływa na inne jej elementy.

Deklaracja – identycznie jak zmienne

- Dim DniTygodnia(6)
- Dim DniTygodnia(1 To 7)
- Dim DniTygodnia(1 To 7) As String
- Dim Oceny(9, 9) As Byte
- Dim Oceny(1 To 10, 1 To 10) As Byte

VB- cd

tablice – zapis, odczyt

- Dim DniTygodnia(1 To 7) As String
  DniTygodnia(1) = "Poniedziałek"
  DniTygodnia(2) = "Wtorek" 'idt.

- MsgBox DniTygodnia(2)
- Textbox2.text=DniTygodnia(1)

Zmienne

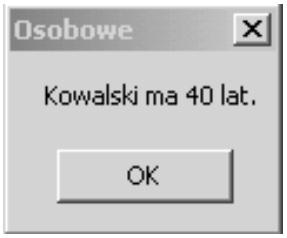
Inicjowanie zmiennych i nadawanie im wartości

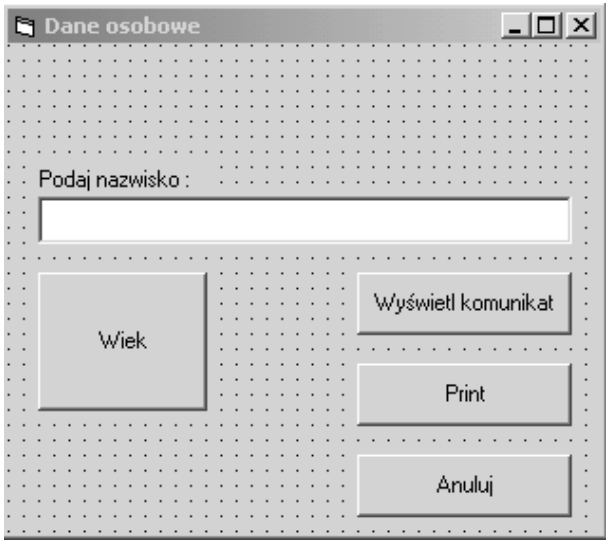
- Jeżeli zmienna nie zostanie zainicjowana wartością, to:
- Zmiennej numerycznej nadawana jest wartość 0,
- Zmiennej łańcuchowej o stałej długości przypisywany jest łańcuch o zerowej długości (""),
- Zmienna łańcuchowa o stałej długości zostanie wypełniona zerami,
- Zmienna typu Variant jest inicjowana jako Empty,

Przybornik – lista formantów

Instrukcje wejścia/wyjścia.

InputBox
MsgBox
TextBox





Operatory

Operatory porównania

Operator	True if	False if	Null if
< (Less than)	*expression1* < *expression2*	*expression1* >= *expression2*	*expression1* or *expression2* = **Null**
<= (Less than or equal to)	*expression1* <= *expression2*	*expression1* > *expression2*	*expression1* or *expression2* = **Null**
> (Greater than)	*expression1* > *expression2*	*expression1* <= *expression2*	*expression1* or *expression2* = **Null**
>= (Greater than or equal to)	*expression1* >= *expression2*	*expression1* < *expression2*	*expression1* or *expression2* = **Null**
= (Equal to)	*expression1* = *expression2*	*expression1* <> *expression2*	*expression1* or *expression2* = **Null**
<> (Not equal to)	*expression1* <> *expression2*	*expression1* = *expression2*	*expression1* or *expression2* = **Null**

Operatory

Operatory logiczne

NOT()	zaprzeczenie
() AND ()	koniunkcja
() OR ()	alternatywa

NOT(12 <= 9.87)	‘True
NOT(12 <> 9.87)	‘False
(12 <= 9.87) AND (12 <> 9.87)	‘False
(9.87 < 12) AND (12 <> 9.87)	‘True
(12 <= 9.87) OR (12 <> 9.87)	‘True
(12 <= 9.87) OR NOT(12 <> 9.87)	‘False

Operatory

Operatory arytmetyczne

+	dodawanie
–	odejmowanie
*	mnożenie
/	dzielenie
^	potęgowanie
\	dzielenie całkowite
MOD	reszta z dzielenia całkowitego

Priorytety wykonywania działań

- (wartość przeciwna)

^

/ \ * mod

+ -

Operatory

Operatory łączenia tekstów

+	łączy dwa teksty
&	wymusza łączenie tekstu z innym typem danych

Działanie	Wynik
“Akademia” + “ Morska”	-> “Akademia Morska”
“Jan “ + “Kowalski”	-> “Jan Kowalski”
“Wynik = “ + “2.5”	-> “Wynik = 2.5”
“Wynik = “ & 2.5	-> “Wynik = 2,5”
“(X > Y) : “ & (5 > 6)	-> “(X > Y) : False”

Wyrażenia

DIM K As INTEGER, X As SINGLE, S As STRING

X = 12.6
K = 8
S = “AM posiada jacht”

Wyrażenie:	Wartość:
(K ^ 2) MOD 5	‘ 4
((X * 2) – K – 1.2) / K * (K – 6)	‘ 4
LEFT(S, 11) & K & MID(S, 11, 4) + “ów”	“Akademia posiada 8 jachtów”

Instrukcja podstawienia

DIM K As INTEGER, X, Y, Z As SINGLE, S, W As STRING

X = 12.6
K = 8
S = “AM posiada jacht”

Wyrażenie:	Wartość:
Y=(K ^ 2) MOD 5	‘Y= 4
Z=((X * 2) – K – 1.2) / K * (K – 6)	‘ Z=4
W=LEFT(S, 11) & K & MID(S, 11, 4) + “ów”	‘ W= “Akademia posiada 8 jachtów”

VB- cd
Wywołanie

- Podprogram można wywołać (uruchomić) z innego podprogramu. Aby wywołać podprogram z innego podprogramu należy w procedurze wywołującej wpisać instrukcję zawierającą jego nazwę.

- Możemy odpowiednią procedurę (bez parametrów) przypisać do **Przycisku** z paska narzędzi **Formularze**

VB- cd
przykład wywołania

```
Sub commandbutton_click()
      Dim wartość, objetosc
       wartość = InputBox("Podaj długość boku szescianu")
      If IsNumeric(wartość) = True Then
       If wartość > 0 Then
        objetosc= Objetosc_szescian(wartość) ' wywołujemy funkcje .
        MsgBox „Objetosc szescianu" & objetosc
       Else
        BłędnaWartość ' wywołujemy podprogram BłędnaWartość.
       End If
      Else
       BłędnaWartość ' wywołujemy podprogram BłędnaWartość.
      End If
End Sub
```

VB- cd
Podstawowe instrukcje

- Instrukcja If... Then... Else - prawdopodobnie 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.
- For... Next - pętla For... Next powtarza blok instrukcji określoną liczbę razy, stosujemy ją jeżeli z góry wiadomo ile razy pętla ma być wykonana.
- For Each... Next - pętla służąca do wykonywania operacji na obiektach kolekcji

VB- cd

Instrukcja If... Then... Else

```
If warunek Then
    [blok kodu wykonywany w przypadku gdy warunek jest spełniony]
    Else
    [blok kodu wykonywany w przypadku gdy warunek nie jest spełniony]
End If
```

```
Private Sub CommandButton1_Click()
 If Range("A1").Value = 0 Then
      Range("A2").Value = "wartość wynosi zero,,
Else
 If Range("A1").Value > 0 Then
       Range("A2").Value = "wartość dodatnia,,
Else
       Range("A2").Value = "wartość ujemna,,
  End If
End If
End Sub
```

VB- cd

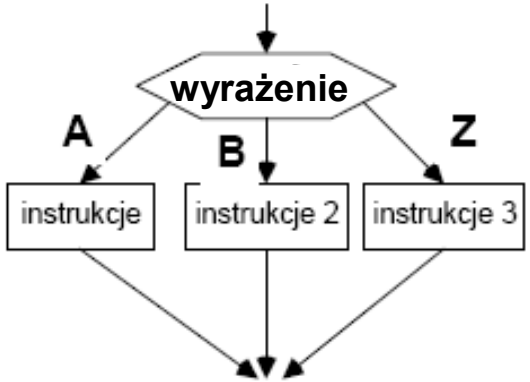
Instrukcja Select Case

```
Select Case Wyrażenie
    Case Wartość1
     [blok kodu wykonywany, jeżeli Wyrażenie równa się Wartość1]
    Case Wartość2
     [blok kodu wykonywany, jeżeli Wyrażenie równa się Wartość2]
    ...
    Case Else
     [blok kodu wykonywany, jeżeli Wyrażenie nie równa się żadnej z
   wartości określonej przez instrukcje Case]
End Select
```

VB- cd

Instrukcja Select Case - przykład

```
Private Sub CommandButton1_Click()
        Dim NumerDnia
         NumerDnia = Range("A1").Value
        If IsNumeric(NumerDnia) = True Then
         Select Case NumerDnia
          Case 1
           Range("A2").Value = "Niedziela"
          Case 2
           Range("A2").Value = "Poniedziałek"
          Case 3
           Range("A2").Value = "Wtorek"
          Case 4
           Range("A2").Value = "Środa"
          Case 5
           Range("A2").Value = "Czwartek"
          Case 6
           Range("A2").Value = "Piątek"
          Case 7
           Range("A2").Value = "Sobota"
          Case Else
           Range("A2").Value = "Poza zakresem wpisz wartość od 1 do 7"
         End Select
        Else
         Range("A2").Value = "Wpisz wartość liczbową"
        End If
       End Sub
```



VB- cd

Pętle warunkowe Do...Loop

Do...Loop Wielokrotnie wykonuje blok kodu tak długą aż instrukcja warunkowa umieszczona wewnątrz tej pętli wykona instrukcje **Exit Do**. W tym przypadku użycie instrukcji Exit Do jest praktycznie obowiązkowe gdybyśmy jej nie zastosowali pętla byłaby wykonywana w nieskończoność.

Do While...Loop Rozpoczyna i powtarza blok kodu umieszczony wewnątrz pętli jeżeli jest spełniony warunek umieszczony na początku tej pętli. Jest to prawdopodobnie najczęściej stosowana odmiana pętli warunkowej,

Do...Loop While Wykonuje blok kodu umieszczony wewnątrz pętli jeden raz i powtarza go tak długą jak długo jest spełniony warunek umieszczony na końcu pętli.

Do Until...Loop Rozpoczyna i powtarza blok kodu umieszczony wewnątrz pętli dopóki nie zostanie spełniony warunek umieszczony na początku tej pętli.

Do...Loop Until Wykonuje blok kodu umieszczony wewnątrz pętli jeden raz i powtarza go do czasu gdy zostanie spełniony warunek umieszczony na końcu pętli .

VB- cd

Instrukcja For... Next

```
For Licznik = Początek To Koniec Step Krok
    [blok instrukcji]
Next Licznik
```

Przykład ‘ tabliczka mnozenia:

```
Sub PrzykładPętli()
    Dim wiersz, kolumna As Integer
     For wiersz = 1 To 10
     For kolumna = 1 To 10
      Cells(wiersz, kolumna) = wiersz * kolumna
     Next kolumna
    Next wiersz
   End Sub
```

VB- cd

Instrukcja For Each... Next

```
For Each element In kolekcja
    [blok kodu wykonywany dla każdego elementu kolekcji]
Next element
```

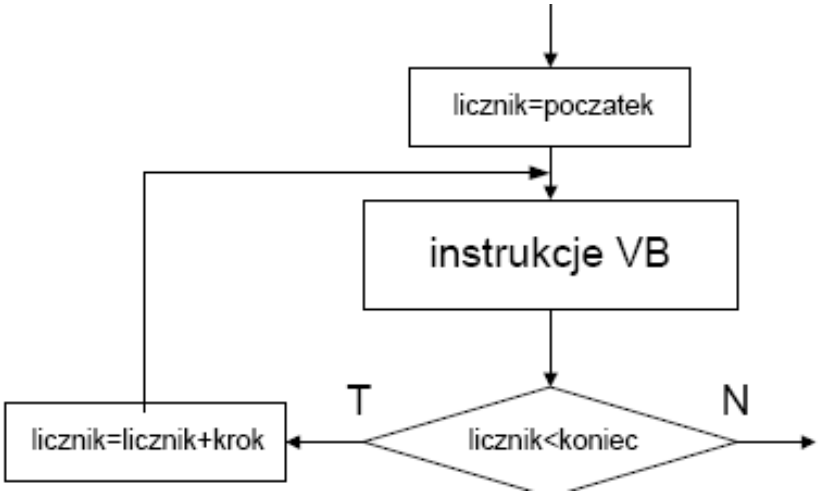
Przykład

```
Sub Wyszukaj()
    For Each element In Range("A1:M25")
     If IsNumeric(element.Value) = True Then
      If element.Value < 0 Then
       element.Interior.ColorIndex = 3
       Exit For ‘ mozliwosc zakonczenia petli
      End If
     End If
    Next
   End Sub
```

Instrukcje organizacji pętli

iteracja

For licznik = początek To koniec [STEP krok]
Instrukcje VB
Next licznik



Licznik: dowolna nazwa zmiennej, która przechowuje liczbę powtórzeń

Początek: określa początkową wartość licznika

Koniec: określa maksymalną liczbę powtórzeń

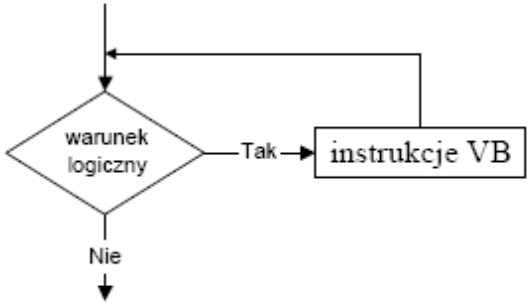
Słowo kluczowe ***Next*** odsyła VB do nazwy licznika.

Za każdym razem, gdy VB wykona instrukcje pętli, wartość licznika będzie zwiększana o jeden. Można to zmienić dopisując słowo kluczowe ***Step*** oraz liczbę oznaczającą o ile powiększamy licznik

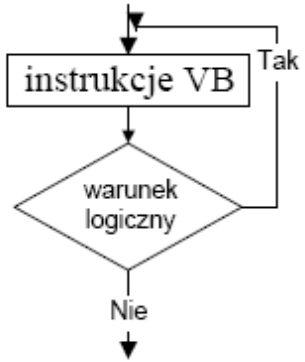
Instrukcje organizacji pętli

cykl

Do While warunek logiczny
Instrukcje VB
Loop



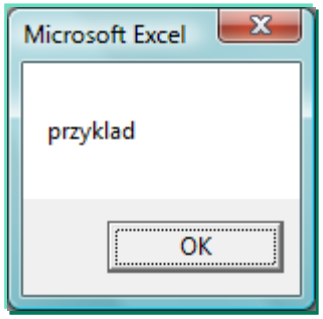
Do
Instrukcje VB
Loop While warunek logiczny



***Do Until* warunek logiczny**
Instrukcje VB
Loop

VB- cd

okno dialogowe MsgBox



- Funkcja **MsgBox** - wyświetla okno dialogowe z jednym lub więcej przyciskami i czeka na reakcję, po czym zwraca wartość typu Integer określającą który przycisk został naciśniety
- **MsgBox(*prompt*[, *buttons*] [, *title*] [,*helpfile*, *context*])**

  Msgbox „przyklad”
- ***prompt*** - argument obowiązkowy, komunikat w oknie dialogowym. Maksymalna długość prompt wynosi około 1024 znaki, zależnie od szerokości znaków w zastosowanej czcionce. Jeśli prompt składa się z kilku wierszy, należy je rozdzielić wstawiając znak powrotu **Chr(13)** lub znak nowego wiersza **Chr(10)** albo kombinację znaków powrót karetki i nowy wiersz **Chr(13) & Chr(10)**.
- ***buttons*** - argument nieobowiązkowy,
- ***title*** - argument nieobowiązkowy,
- ***helpfile*** - argument nieobowiązkowy,
- ***context*** - argument nieobowiązkowy,

VB- cd

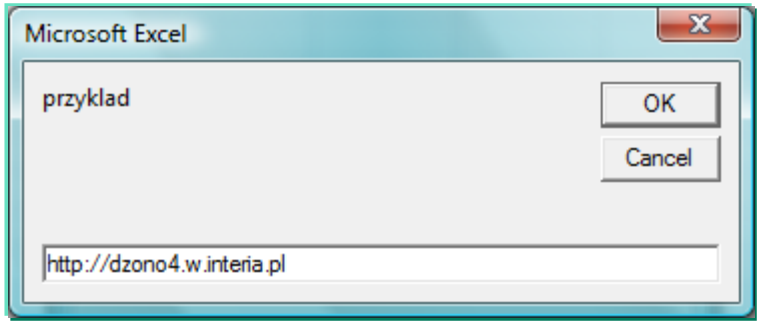
okno dialogowe InputBox

- Funkcja InputBox wyświetla okno dialogowe z polem tekstowym i dwoma przyciskami, po czym zwraca typ danych **String** będący zawartością pola tekstowego.

InputBox(*prompt*[, *title*] [, *default*] [, *xpos*] [, *ypos*] [,*helpfile, context*])

InputBox(„przykład, , http://.....)

- ***prompt*** - argument obowiązkowy, komunikat
- ***title*** - argument nieobowiązkowy,
- ***default*** - argument nieobowiązkowy,
- ***xpos*** - argument nieobowiązkowy,
- ***ypos*** - argument nieobowiązkowy,
- ***helpfile*** - argument nieobowiązkowy,
- ***context*** - argument nieobowiązkowy,



VB- cd

odwołanie do komórek skoroszytów

- Range(“Zakres")
  - Range("A1:M25") Range("A1") Range("A1;M25")
- Cells(wiersz,kolumna)
  - Cells(2,3)
- [adres]
  - [B7]