Wprowadzenie do programowania w VBA

Spis treści

Struktura programu

Sub nazwa_programu([argumenty])

Deklaracje zmiennych

Wprowadzanie wartości zmiennych

Instrukcje obliczeniowe i logiczne

Wyprowadzanie wartości zmiennych lub wyrażeń

End sub

Nazwa programu (takie same zasady dla zmiennych i stałych)Typy zmiennych

- Musi zaczynać się od litery (nie może zaczynać się od cyfry).
- Nie może zawierać znaków (symboli) ze zbioru: { kropka, przecinek, spacja, !, $, @, #, [,], {,},} oraz operatorów.
- Nie dłuższa niż 255 znaków.
- Nie może być taka sama jak nazwa: innych procedur i funkcji oraz słów kluczowych języka VBA.
- Do oddzielenia tekstu wieloczłonowego używamy znaku podkreślenia „_” (nie spacji).

Typy danych

Typ danych określa sposób, w jaki są interpretowane bity przechowywane w pamięci. W węższym rozumieniu typ danych jest zbiorem wartości, które może przyjmować zmienna.

W języku VB wyróżniamy 11 podstawowych (standardowych, predefiniowanych) typów danych. Oprócz tego typy mogą być definiowane przez użytkownika, np. typ "konto" z operacjami "wpłata", "wypłata" i "stan konta".

Typ	Zawartość	Wielkość pamięci	Zakres
Byte	całkowita	1 bajt	0 do 255
Boolean	całkowita	2 bajty	True lub False
Integer	całkowita	2 bajty	-32,768 do 32,767
Long	całkowita	4 bajty	-2,147,483,648 do 2,147,483,647
Single	liczba zmiennopozycyjna	4 bajty	Ujemne:-3.402823E38 do -1.401298E-45 Dodatnie: 1.401298E-45 do 3.402823E38
Double	liczba zmiennopozycyjna	8 bajtów	Ujemne:-0.79769313486232E308 do -4.94065645841247E-324 Dodatnie: 4.94065645841247E-324 do 1.79769313486232E308
Currency	liczba z ustaloną kropką dziesiętną	8 bajtów	-922,337,203,685,477.5808 do +922,337,203,685,477.5807
Date	data (miesiąc/dzień/rok)	8 bajtów	1/1/100 do 1/31/9999
Object	adres obiektu	4 bajty	dowolna referencja do obiektu
String	łańcuch znaków	10 bajtów + długość stringu (dla zmiennej długości stringu); długość stringu (dla stringów o ustalonej długości)	0 do 2 miliardów znaków (0 do 65,400 dla Windows 3.1 i wersji wcześniejszych)
Variant	dowolne dane (poza stringiem o ustalonej długości i typem zdefiniowanym przez użytkownika)	16 bajtów (dla danych liczbowych); 22 bajty + długość stringu (dla stringu)	Wartość liczbowa (dla danych liczbowych); jak w przypadku stringów zmiennej długości

Deklaracja zmiennych i stałych

Zmienna jest dostępna w programie za pomocą nazwy użytej w deklaracji. Podczas wykonywania programu zmiennej można przypisywać różne wartości, zawsze jednak zgodne z zadeklarowanym typem. Do przypisania wartości zmiennej służy instrukcja przypisania.

Deklaracja zmiennej, niezależnie od typu przechowywanych danych, ma następującą składnię:

Dim | Private | Public *NazwaZmiennej* [**As** TypDanych]

Deklarację stałej przeprowadza się wg poniższego schematu:

[Private | Public] Const *NazwaStałej* [**As** TypDanych] = wartość

Przykłady:

Dim *Numer* **As** *Byte* ' deklaracja zmiennej typu bajtowego

Public *Numer1* **As** *Integer* ' deklaracja zmiennej typu całkowitoliczbowego, ważna we wszystkich modułach, o ile podana na początku modułu

Private *Numer2* **As** *Double* 'deklaracja zmiennej typu rzeczywistego, ważna w module, na początku którego została podana

Dim *Cena* **As** *Single* ' deklaracja zmiennej o nazwie Cena, typ - liczby rzeczywiste

Const *KONIECROKU* **As** *Date* = #12/31/2015#

deklaracja stałej o nazwie KONIECROKU (typ data) z wartością 31 grudnia 2015

Const *NAZWISKO* ***As*** String = "Kowalski" ' deklaracja stałej o nazwie NAZWISKO (typ – ciąg znaków)

Deklaracja tablic

Tablica jest zbiorem zmiennych tego samego typu danych. Jest ona stosowana do zgrupowania razem zmiennych zależnych.

Dim | Private | Public *NazwaTablicy*(indeksy) [**As** TypDanych]

Dim Sprzedaż(5)

Dim NrRachunku(700 **To** 799) **As** Integer

Dim Współrz(4, 1 **To** 3) **As** Byte

Dim Współrz(0 **To** 4, 1 **To** 3) **As** Byte

Do elementu tablicy odwołujemy się podając nazwę tablicy i indeks umieszczony w nawiasach:

Sprzedaż(0), Sprzedaż(1).....Sprzedaż(5)

NrRachunku(700).....NrRachunku(799)

Numer początkowy indeksu tablic możemy zmienić z 0 na 1 stosując instrukcję **Option Base 1**. Instrukcje tę należy wpisać na początku modułu w sekcji deklaracji, czyli przed pierwszą występującą w nim procedurą.

Instrukcja przypisania

Instrukcja przypisania wykorzystywana jest do nadania zmiennej lub obiektowi wartości wyrażenia. Jako operator przypisania wykorzystuje się symbol ”=” lub słowo kluczowe LET.

zmienna = wyrażenie

Przykłady instrukcji przypisania:

Cena = 25 zmiennej o nazwie *Cena* przypisana zostaje wartości 25

Waga = 50.50 zmiennej o nazwie *Waga* przypisana zostaje wartości 50,50

Imie = ”Gosia” zmiennej o nazwie *Imie* przypisana zostaje wartość podana w cudzysłowie

Zm01 = Zm02 zmiennej o nazwie *Zm01* przypisana jest wartości przechowywana przez drugą zmienną o nazwie *Zm02*. *UWAGA !* Zmienne *Zm01* i *Zm02* muszą być zgodne co do typu, lub zmienna *Zm01* musi być typu nadrzędnego względem zmiennej *Zm02*.

Wprowadzanie danych

1. Instrukcja przypisania (bezpośrednio); niewygodne, nieelastyczne (jak w przykładach powyżej).
2. Instrukcja przypisania (poprzez odczyt wartości wprowadzanych z klawiatury).

   Zmienna = InputBox(komunikat**)**

   Zadaniem funkcji InputBox() jest:

   - wyświetlenie komunikatu,
   - oczekiwanie na wprowadzenie wartości (łańcucha znaków) przez użytkownika,
   - przekazanie wprowadzonej wartości do zmiennej.

   Przykład:

   z = InputBox(„Podaj wartość zmiennej z”**)**

3. Instrukcja przypisania (odczyt wartości z komórki arkusza)

   Przypisanie zmiennej wartości przechowywanej w komórce arkusza przeprowadza się wg szablonu:

   NazwaZmiennej = **Cells**(*nr_wiersza*, *nr_kolumny*).**Value**

Przykład - odczyt z komórki D2 (wiersz 2, kolumna 4):

Wiek = **Cells**(2, 4).**Value**

Wyprowadzanie wyników

1. Instrukcja *MsgBox*.

   Instrukcja MsgBox daje możliwość wyświetlenia tekstu komunikatu, bez zwracania wartości wybranego przycisku.

   MsgBox "Tekst wyprowadzanego komunikatu"

   Przykłady:

   MsgBox "Wartość zmiennej x =" **& Str(**x**)**

   MsgBox "Wykład z przedmiotu " **& Chr(13) &** Przedmiot

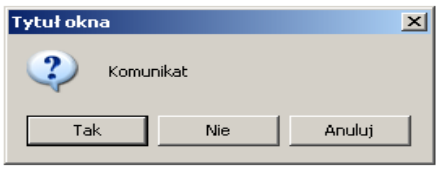
2. Funkcja **MsgBox**().

   Zadania funkcji **MsgBox()** obejmują:

   1. wyświetlenie okna dialogowego z komunikatem,
   2. oczekiwanie na wybór jednego z kolekcji przycisków,
   3. zwrot wartości typu **Integer** informującej, który z przycisków został wybrany przez użytkownika.

   Zwracana wartość jest zwykle przypisywana do zdefiniowanej przez użytkownika zmiennej.

   MsgBox(TekstKomunikatu [,Przyciski+Ikony+WartościDomyślne] [,TytułOkna])

   Sub Przyklad()

   Dim *KOD* **As Integer**

   KOD = **MsgBox**("Czy kwadrat liczby 3 to liczba 9 ?", **vbYesNo** _

   + vbQuestion, "Pytanie")

   If *KOD* = **vbYes Then** ***' zamiennie można użyć warunku If KOD=6 Then***

   MsgBox "Poprawna odpowiedź"

   Else

   MsgBox "Błędna odpowiedź"

   End If

   End Sub

3. Zapis wartości w komórce arkusza.

   Wypisanie wartości zmiennej w określonej komórce arkusza, przeprowadza się wg szablonu:

   Cells(*nr_wiersza*, *nr_kolumny*).**Value** = *NazwaZmiennej*

   Przykład - wpisanie do komórki A2 (wiersz 2, kolumna 1):

   Cells(2, 1).**Value** = Imie

Instrukcja warunkowa IF

W języku VBA instrukcja **IF** występuje w wielu wariantach składniowo-semantycznych.

Instrukcję sterującą **If** (prostą) wykorzystuje się do warunkowego wykonania fragmentu kodu programu.

Język VBA przewiduje dwojaki sposób zapisu tej instrukcji – blokowy oraz wierszowy.

1) postać wierszowa

If warunek **Then** polecenie [:polecenie_1]… [:polecenie_n]

2) postać blokowa

```
If warunek Then
        polecenie
        [polecenie_1]
         …
        [polecenie_n]
End If
```

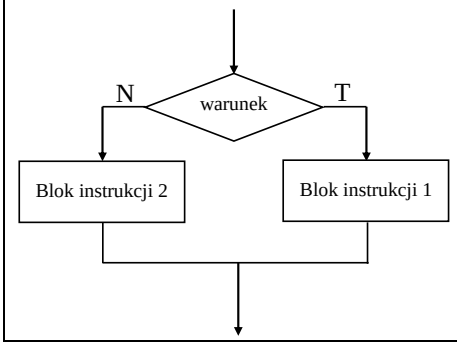
Przykład:

```
Sub Przyklad_6_2()
   Dim Liczba As Integer
   Liczba=InputBox("Podaj liczbę")
   If Liczba < 0 Then
      Liczba = -Liczba
   End If
   MsgBox "Moduł wynosi  " & Liczba
End Sub
```

Instrukcja warunkowa IF-THEN-ELSE

Instrukcja **If**…**Then**…**Else** umożliwia wykonanie tylko jednego z dwóch bloków kodu, z których jeden występuje po słowie kluczowym **Then**, drugi - po **Else**.

If…**Then**…**Else** może występować w postaci wierszowej i blokowej, aczkolwiek postać wierszowa jest rzadko stosowana ze względu na małą przejrzystość zapisu.




```
If warunek Then
   [blok_instrukcji_1]
Else
   [blok_instrukcji_2]
End If
```

Przykład:

```
Sub Przyklad_6_4()
   Dim Liczba As Byte
   Liczba = InputBox("Podaj dodatnią liczbę całkowitą mniejszą od 100")
   If Liczba < 10 Then
      MsgBox "Podano jednocyfrową liczbę"
   Else
      If Liczba < 100 Then
         MsgBox "Podano dwucyfrową liczbę"
      Else
         MsgBox "Podano nieprawidłową liczbę"
      End If
   End If
End Sub
```

Instrukcja skoku

Instrukcja **GoTo** powoduje przekazanie sterowania do linii programu wewnątrz procedury.
GoTo *Linia*
Argument *Linia* może być etykietą linii albo numerem linii.
Instrukcja **GoTo** może wykonywać skoki wewnątrz procedury, w której została użyta.

Przykład pętli z użyciem GoTo:

```
Sub skok_petla()
Dim i As Byte, n As Byte
i = 1
n = 7
etykieta: i = i + 1
MsgBox "Licznik wynosi " & i
If i < n Then GoTo etykieta
End Sub
```

Pętla liczona

Pętla liczona jest konstrukcją powtarzającą wykonanie pojedynczego bloku kodu określoną ilość razy. Pętle liczone są wykorzystywane wówczas, gdy z góry wiadomo, ile razy ma się wykonać jakiś fragment programu.

For *zmienna_ster* **=** *początek* **To** *koniec* [**Step** *krok*]
   Blok instrukcji
   [**Exit For**]
   Blok instrukcji
Next *zmienna_ster*

W *bloku instrukcji* nie należy zmieniać wartości zmiennej sterującej.

Przykład:

x = 3

For i =1 **To** 7 **Step** 2

  x = x + i

  If x > 7 **Then Exit For**

Next i

Pętla z warunkiem na początku/końcu

Pętle z logicznym warunkiem zakończenie to pętle, których zakończenie następuje dopiero w momencie spełnienia pewnego warunku. Podczas gdy pętle liczone wykonują jakiś blok kodu konkretną liczbę razy, pętle z logicznym warunkiem zakończenia mogą się wykonywać nawet nieskończoną liczbę razy albo nie wykonywać się w ogóle w zależności od stanu warunku logicznego kończącego taką pętlę.

Podstawową pętlą z logicznym warunkiem zakończenia jest pętla **Do…Loop**. Istnieją cztery warianty tej pętli:

1. z warunkiem **True** na początku,
2. z warunkiem **True** na końcu,
3. z warunkiem **False** na początku,
4. z warunkiem **False** na końcu.

Warunek na początku:

Do While | Until *warunek*

  blok instrukcji

  [**Exit Do**]

  blok instrukcji

Loop

Warunek na końcu:

Do

  blok instrukcji

  [Exit Do]

  blok instrukcji

Loop While | Until *warunek*

Słowa kluczowe **While** i **Until** decydują o znaczeniu warunku wykonywania pętli. **While** wskazuje, że pętla będzie się wykonywać, dopóki warunek ma wartość **True,** a słowo kluczowe **Until** wskazuje, że pętla będzie się wykonywać, aż warunek przyjmie wartość **True**.

Instrukcja **Exit Do** powoduje wcześniejsze zakończenie pętli i jest zazwyczaj elementem struktury logicznej (instrukcji **If**) sprawdzającej alternatywny warunek zakończenia pętli, taki jak np. wystąpienie błędu.

Przykład (kontrola poprawności wprowadzanych danych):

```
Do
   r1 = InputBox("Podaj r1 > 0")
Loop Until r1 > 0

Do
   r2 = InputBox("Podaj r2 > 0")
Loop While r2 <= 0
```

Operatory arytmetyczne

Wykonywana operacja	Symbol operatora	Przykład	Wynik działania
Dodawanie	+	2+5	7
Odejmowanie	-	10-8	2
Mnożenie	*	2*3	6
Dzielenie	/	10/4	2.5
Dzielenie całkowite	\	10\4	2
Reszta z dzielenia	Mod	10 mod 3	1
Potęgowanie	^	3^2	9
Zmiana znaku liczby	-	5	-5

Funkcje wbudowane

Funkcja służy do obliczenia pojedynczej wartości. Zaletą funkcji jest możliwość bezpośredniego korzystania z nich w wyrażeniach (tak samo jak zmiennych).

Rnd liczba losowa <0, 1)

Abs(X) wartość bezwzględna

Sgn(X) znak liczby

Fix(X) część całkowita (po usunięciu części ułamkowej)

Int(X) część całkowita (po zaokrągleniu w dół)

Log(X) logarytm naturalny

Exp(X) *e* do potęgi

Sqr(X) pierwiastek kwadratowy

Sin(X) sinus

Cos(X) cosinus

Tan(X) tangens

Chr*(KodZnaku)* kod ASCII na znak

Str*(Wartość)* liczba na tekst

Val*(Łańcuch)* tekst na liczbę

Funkcje użytkownika

Funkcja zaczyna się od instrukcji deklaracji funkcji ***Function*** i kończy instrukcją ***End Function***.

[Private | Public] Function *nazwa_funkcji* ([*argumenty*]) [**as** typ]

[*instrukcje*]
[*nazwa_funkcji =Wartość*]
[**Exit Function**]
[*instrukcje*]
[*nazwa_funkcji =Wartość*]

End Function

Parametry (argumenty) umieszczone w nagłówku procedury (funkcji) nazywamy parametrami formalnymi, a parametry użyte w miejscu wywołania - parametrami aktualnymi.

Liczba parametrów aktualnych musi być równa liczbie parametrów formalnych. Z reguły typ parametrów aktualnych musi być zgodny z typem odpowiednich parametrów formalnych (za wyjątkiem, gdy parametry przekazywane są przez wartość).

Nazwy zmiennych w liście argumentów muszą być oddzielone od siebie przecinkami.

Przykład:

```
Function Przyklad(k As Integer, Optional b As Byte = 3)
 If k=0 Then
        Przyklad = b
        Exit Function  '' opuść procedurę
 Else
        k = k * 7
        Przykład = k  '' wykonaj obliczenie i zwróć wartość
 End If
End Function  '' koniec procedury
```

Przekazywanie argumentów

[**Optional**] [**ByVal** | **ByRef**] *nazwa_arg* [**As** *typ*] [=*domyślna_wartość*]

Optional (opjcjonalny); słowo kluczowe, określające, że argument nie jest wymagany. Jeżeli zostanie użyty, wszystkie argumenty listy muszą być również opcjonalne i zadeklarowane przy użyciu słowa **Optional**.

ByVal (opcjonalne); określa, że wartość zostanie przekazana przez wartość. **ByRef** (opcjonalne, domyślne); określa, że wartość zostanie przekazana przez referencję.

domyślna_wartość (opcjonalne); wartość domyślna argumentu. Może to być stała lub wyrażenie stałe, dotyczy tylko argumentów **Optional**.

Funkcje arkusza

W kodzie VBA możemy wykorzystywać wbudowane funkcje arkusza. Zapis:

Application.WorksheetFunction.*NazwaFunkcji*([argumenty])

Przykłady:

```
Pi = Application.WorksheetFunction.PI()  ‘ liczba π
Si = Application.WorksheetFunction.FactDouble(120)  ‘ silnia
x =  Application.WorksheetFunction.NormInv(Rnd, xSt, odch)  ‘ wylosowanie liczby z rozkładu normalnego
```