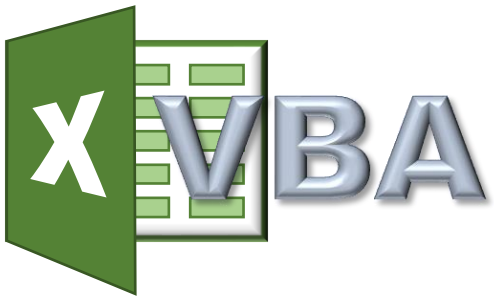
Techniki programowania



Instrukcje iteracyjne

Instrukcje Do-Loop

Instrukcje iteracyjne – podsumowanie

Instrukcje iteracyjne Do-Loop

```
Do While warunek
  instrukcja
Loop
```

Instrukcja jest wykonywana dopóki *warunek* jest spełniony (przerywa gdy ma wartość `False` , sprawdzenie na początku)

```
Do Until warunek
  instrukcja
Loop
```

Instrukcja jest wykonywana chyba że *warunek* jest spełniony (przerywa gdy ma wartość `True`, sprawdzenie na początku)

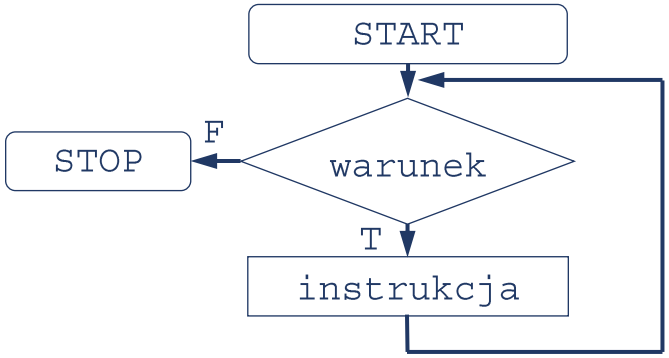
```
Do Loop
  instrukcja
While warunek
```

Instrukcja jest wykonywana dopóki *warunek* jest spełniony (przerywa gdy ma wartość `False`, sprawdzenie na końcu)

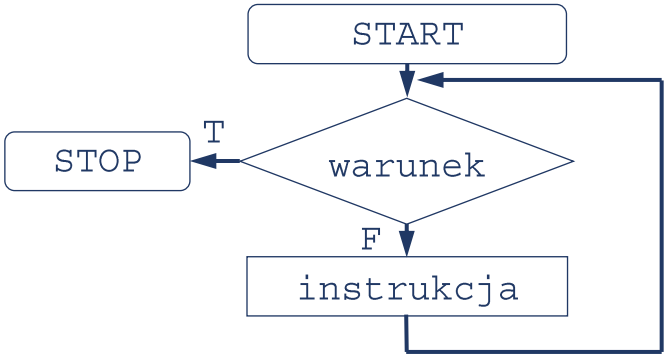
```
Do Loop
  instrukcja
Until warunek
```

Instrukcja jest wykonywana chyba że *warunek* jest spełniony (przerywa gdy ma wartość `True`, sprawdzenie na końcu)

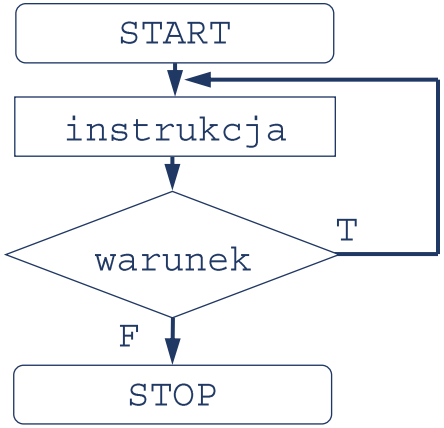
Instrukcje iteracyjne Do-Loop



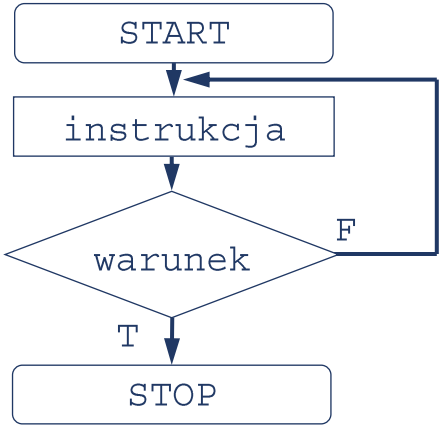
Do-While-Loop



Do-Until-Loop



Do-Loop-While



Do-Loop-Until

Funkcja SumaLiczb – wersja Do While

Funkcja oblicza sumę wartości liczbowych (z wyłączeniem dat) w podanym zakresie. Działanie identyczne z funkcją z wykładu 04 s.10, wykorzystana pętla **Do-While-Loop**.

```
Public Function SumaLiczb3(zakres As Range) As Double
 Dim k As Range
 Dim s As Double
 Dim i As Integer
 s = 0:    i = 0
 Do While i < zakres.Count
   i = i + 1
   If Not IsEmpty(zakres(i).Value) And IsNumeric(zakres(i).Value) Then
     s = s + zakres(i).Value
   End If
Loop
 SumaLiczb3 = s
End Function
```

=SumaLiczb3(G5:G11)

	G	H	I
2			
3	SumaLiczb3		
4	(wersja z DO..WHILE)		
5	1		
6	2		
7	a		
8	1 mar 22		
9	b		
10	3		
11	4		
12	10		
13			

Analiza, SumaLiczb3("G5:G11")

0. s=0, i=0
1. i=1, s=s+zakres(1)=0+1=1
2. i=2, s=s+zakres(2)=1+2=3
3. i=3
4. i=4
5. i=5
6. i=6, s=s+zakres(6)=3+3=6
7. i=7, s=s+zakres(7)=6+4=10

Kod dostępny na stronie przedmiotu

For Each i Do While – porównanie

For Each

```
Dim k As Range
Dim s As Double

s = 0
For Each k In zakres

  If … Then
    s = s + k.Value
  End If
Next
SumaLiczb1 = s
```

Do While

```
Dim k As Range
Dim s As Double
Dim i As Integer
s = 0:    i = 0
Do While (i < zakres.Count)
  i = i + 1
  If … Then
    s = s + zakres(i).Value
  End If
Loop
SumaLiczb3 = s
```

Licznik elementów kolekcji

Warunek końca pętli oparty na liczbie elementów

Inkrementacja licznika

Odwołanie do elementów kolekcji przez indeksowanie

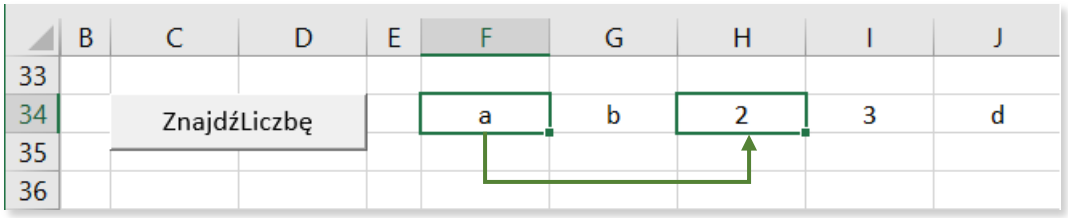
Wnioski:

- dowolna pętla For może być zastąpiona pętlą Do…Loop (np. Do-While-Loop),
- For zawsze przetwarza wszystkie elementy w kolekcji,
- warunek w Do…Loop jest dowolny i pozwala na przerwanie w dowolnym momencie,
- do operowania na kolekcji Do…Loop wymaga użycia licznika elementów oraz indeksowania, nie jest uzasadniona w przypadku przetwarzania całej kolekcji, może być wykorzystana do przetwarzania części kolekcji.

Poszukiwanie wartości, wersja I

Procedura poszukuje pierwszego wystąpienia liczby w bieżącym wierszu przeszukując komórki położone na prawo od komórki aktualnej, pętla **Do-Loop-Until**.

```
Public Sub ZnajdźLiczbę1()
  Dim k As Range
  Set k = ActiveCell
  On Error GoTo koniec
  Do
    Set k = k.Offset(0, 1)
  Loop Until Not IsEmpty(k.Value) And IsNumeric(k.Value)
  k.Select
koniec:
End Sub
```



Analiza programu

1. k = "F34"
2. Wykonanie pętli:
   - 2.1. k = k.Offset(0,1) = "G34"
   - 2.2. Not IsEmpty = True, IsNumeric = False
   - 2.3. k = k.Offset(0,1) = "H34"
   - 2.4. Not IsEmpty = True, IsNumeric = True
3. k.Select

Kod dostępny na stronie przedmiotu

Poszukiwanie wartości, wersja II

Procedura poszukuje pierwszego wystąpienia liczby w bieżącym wierszu przeszukując komórki położone na prawo od komórki aktualnej, pętla **`Do-Loop-While`**.

```
Public Sub ZnajdźLiczbę2()
  Dim k As Range
  Set k = ActiveCell
  On Error GoTo koniec
  Do
    Set k = k.Offset(0, 1)
  Loop While IsEmpty(k.Value) Or Not IsNumeric(k.Value)
  k.Select
koniec:
End Sub
```

negacja warunku z wersji I

Uwaga 1: instrukcja iteracyjna `Do-Loop-Until` kończy działanie gdy warunek jest prawdziwy, `Do-Loop-While` gdy warunek jest fałszywy, dowolną operację zapisaną przy pomocy jednej pętli można zamienić na drugą przez zanegowanie warunku.

Uwaga 2: Obsługa błędów gwarantuje prawidłowe działanie procedury, gdy przeszukiwanie osiąga ostatni elementu w wierszu (nie można wykonać operacji `offset`) – dotyczy obydwu wersji programu.

Kod dostępny na stronie przedmiotu

Sprawdzenie czy wiersz jest pusty

Funkcja prywatna, sprawdza czy wiersz przekazany jako argument jest pusty. Zwraca wartość **`True`** jeżeli wiersz jest pusty lub **`False`** w przeciwnym wypadku. Funkcja pomocnicza używana w kolejnych przykładach.

```
Private Function WierszJestPusty(wiersz As Range) As Boolean
  Dim n As Integer
  n = 0
  On Error Resume Next
  n = wiersz.Cells.SpecialCells(xlCellTypeConstants).Count
  n = n + wiersz.Cells.SpecialCells(xlCellTypeFormulas).Count
  If n = 0 Then WierszJestPusty = True _
  Else WierszJestPusty = False
End Function
```

Uwaga 1: do sprawdzenia wiersza została użyta metoda **`SpecialCells`**, która pozwala określić liczbę komórek zawierających wartości stałe oraz formuły (wiersz jest pusty jeżeli liczba wynosi zero).

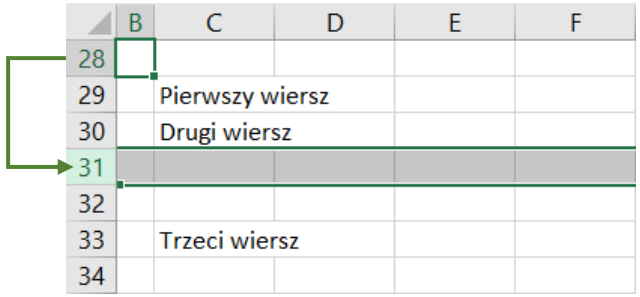
Uwaga 2: obsługa błędów gwarantuje prawidłowe działanie funkcji gdy wiersz nie zawiera komórek z określoną zawartością (w takim przypadku **`SpecialCells`** zgłasza wyjątek).

Kod dostępny na stronie przedmiotu

Poszukiwanie wiersza, wersja I

Procedura poszukuje pierwszego pustego wiersza położonego poniżej wiersza bieżącego (tego, w którym znajduje się kursor), pętla **`Do-Loop-Until`**.

```
Public Sub ZnajdźWiersz1()
  Dim w As Range
  On Error GoTo koniec
  Set w = ActiveCell
  Do
    Set w = w.Offset(1, 0).EntireRow
  Loop Until WierszJestPusty(w)
  w.Select
koniec:
End Sub
```



Analiza programu

```
1. w = ActiveCell = "B28"
2. Pętla:
   2.1.  w = w.Offset(1,0).EntireRow = "$29:$29"
   2.2.  WierszJestPusty = False
   2.3.  w = w.Offset(1,0).EntireRow = "$30:$30"
   2.4.  WierszJestPusty = False
   2.5.  w = w.Offset(1,0).EntireRow = "$31:$31"
   2.6.  WierszJestPusty = True
3. k.Select
```

Kod dostępny na stronie przedmiotu

Poszukiwanie wiersza, wersja II

Procedura poszukuje pierwszego pustego wiersza położonego poniżej wiersza bieżącego (tego, w którym znajduje się kursor), pętla **Do-While-Loop**.

```
Public Sub ZnajdźWiersz2()
  Dim w As Range
  On Error GoTo koniec
  Set w = ActiveCell.Offset(1, 0).EntireRow
  Do While Not WierszJestPusty(w)
    Set w = w.Offset(1, 0).EntireRow
  Loop
  w.Select
koniec:
End Sub
```

Kod dostępny na stronie przedmiotu

Uwaga 1: instrukcja iteracyjna `Do-Loop-Until` sprawdza warunek na końcu, `Do-While-Loop` na początku, pierwszy sprawdzany wiersz musi być przypisany przed pętlą.

Uwaga 2: instrukcja iteracyjna `Do-Loop-Until` kończy działanie gdy warunek jest prawdziwy, `Do-While-Loop` gdy warunek jest fałszywy, warunek końca jest zanegowany.

Uwaga 3: Obsługa błędów gwarantuje prawidłowe działanie procedury, gdy przeszukiwanie osiąga ostatni wierszu (nie można wykonać operacji `offset`) – dotyczy obydwu wersji programu.

Odczyt wartości spełniającej zadane kryteria

Procedura prosi użytkownika (do skutku) o podanie liczby z przedziału 1-10. Jeżeli podana wartość nie spełnia określonych kryteriów pytanie jest ponawiane.

```
Public Sub OdczytWartości()
  Dim x As Double
  Do
    x = Application.InputBox("Podaj liczbę z przedz. 1-10", Type:=1)
  Loop Until (x >= 1 And x <= 10) Or (x = 0)
  If x <> 0 Then MsgBox "Poałeś liczbę: " & x
End Sub
```

Uwaga 1: w tego typu zastosowaniu wskazane jest użycie instrukcji iteracyjnej sprawdzającej warunek na końcu (w każdym przypadku niezbędne jest co najmniej jednokrotne pytanie o wartość).

Uwaga 2: dodatkowy warunek końca (`x=0`) umożliwia przerwanie pętli przez naciśnięcie przycisku "Anuluj" (InputBox z parametrem type=1 zwraca w takim przypadku 0).

Kod dostępny na stronie przedmiotu

Przykład – funkcja Słownie cz.I

Funkcja prywatna zamienia na tekst liczby od 0 do 999. Używana przez właściwą funkcję `Słownie` do zamiany kolejnych trójek (tysięcy, milionów, itd.)

```
Private Function Słownie_pomoc(liczba As Integer) As String
  Dim txt As String
  Dim s As Integer, d As Integer, j As Integer

  s = liczba \ 100
  d = (liczba Mod 100) \ 10
  j = liczba Mod 10

  If s > 0 Then txt = txt & Choose(s, "sto", "dwieście",...)
  If d > 1 Then txt = txt & " " & Choose(d, "", "dwadzieścia",...)
  If d = 1 Then
    txt = txt & " " & Choose(j + 1, "dziesięć", "jedenaście",...)
  Else
    txt = txt & " " & Choose(j, "jeden", "dwa", "trzy",...)
  End If

  Słownie_pomoc = txt
End Function
```

Przykład – funkcja Słownie cz.II

```
1  Public Function Słownie(liczba As Long) As String
2    Dim txt As String
3    Dim l As Integer, i As Integer
4    Dim ujemna As Boolean
5    If liczba = 0 Then: Słownie = "zero":  Exit Function
6    If liczba < 0 Then: liczba = Abs(liczba):  ujemna = True
7    txt = ""
8    i = 1
9    Do While liczba > 0
10     l = liczba Mod 1000
11     liczba = liczba \ 1000
12     txt = Słownie_pomoc(l) & _
13           Choose(i, "", " tys.", " mln", " mld") & " " & txt
14     i = i + 1
15   Loop
16   If ujemna Then txt = "minus " & txt
17   Słownie = txt
18 End Function
```

Kod dostępny na stronie przedmiotu

Funkcja Słownie – analiza

Analiza działania funkcji Słownie dla wartości 17 359

5-8. `ujemna = False, txt = "", i = 1`

9-15. `liczba > 0` (warunek prawdziwy), start pierwszej iteracji pętli:

10. `l = 17359 Mod 1000 = 359` (ostatnia trójka cyfr),

11. `liczba = 17359/1000 = 17` (pozostała część liczby),

12-13. Wywołanie funkcji `Słownie_pomoc` z wartością 359 (zwraca tekstową reprezentację wartości), ponieważ `i=1` żaden przyrostek nie będzie dodany, więc `txt="trzysta pięćdziesiąt dziewięć"`,

14. `i=i+1=2`, powiększenie licznika iteracji.

9-15. `liczba > 0` (warunek prawdziwy), start drugiej iteracji pętli:

10. `l = 17 Mod 1000 = 17` (druga trójka cyfr – w tym przypadku pozostały dwie),

11. `liczba = 17/1000 = 0` (pozostała część liczby),

12-13. Wywołanie funkcji `Słownie_pomoc` z wartością 17 (zwraca tekstową reprezentację wartości), ponieważ `i=2` będzie dodany przyrostek `"tys."`, więc `txt="siedemnaście tys."+"trzysta pięćdziesiąt dziewięć"`,

14. `i=i+1=2`, powiększenie licznika iteracji.

9. `liczba = 0` (warunek fałszywy), zakończenie pętli Do-While-Loop

17. `Słownie = txt = "siedemnaście tys. trzysta pięćdziesiąt dziewięć"`

Instrukcje iteracyjne – podsumowanie

`For Each`

Cechy: specjalizowana, iteruje kolekcję, pobierając kolejne elementy.

Zastosowanie: przetwarzanie całych kolekcji.

`For…Next`

Cechy: ogólnego przeznaczenia, dostarcza automatycznie powiększany licznik numerujący iteracje, możliwe ustalenie kroku licznika.

Zastosowanie: powtarzanie ciągu instrukcji w przypadku z góry znanej liczby powtórzeń, przetwarzanie elementów kolekcji z ustalonym krokiem.

`Do…Loop`

Cechy: uniwersalne, umożliwiają rozwiązanie dowolnego zadania, warunek zakończenia dowolnie ustalany przez użytkownika.

Zastosowanie: powtarzanie ciągu instrukcji w przypadku nieznanej liczby powtórzeń.

`Do-Loop-While, Do-Loop-Until`: gdy operacje umieszczone w pętli muszą być wykonane co najmniej jeden raz.

`Do-While-Loop, Do-Until-Loop`: gdy operacje umieszczone w pętli mogą nie być wykonywane ani razu.