Programowanie obiektowe

Błędy wykonania, instrukcje sterujące

obsługa błędów w VBA
projektowanie algorytmów
instrukcja warunkowa
przetwarzanie kolekcji, instrukcja pętli

Podsumowanie Wykładu01

- Aplikacja jest zbiorem obiektów, każdy zawiera pewne właściwości i metody, najważniejsze obiekty Excel-a: **Application**, **Workbook**, **Worksheet**, **Range.**
- Aplikacja może zawierać kolekcję Workbook-ów (obiekt **Workbooks**), jeden Workbook zazwyczaj zawiera kolekcję arkuszy (obiekt **Worksheets**).
- Każda komórka, zaznaczenie, kolumna, wiersz, itp. są obiektem typu **Range**.
- Odwołanie do właściwości/metody: **`obiekt.właściwość, obiekt.metoda`**
- Odwołanie do kolekcji: **`nazwa_kolekcji(indeks)`**
- Obiekty: **Application**, aktywny **Workbook** i aktywny **Worksheet** są domyślne, na ogół mogą być pominięte w odwołaniach.
- Program przechowuje dane w zmiennych, schemat deklaracji

  `Dim` *<nazwa>* **`As`** *typ*
- Do zmiany wartości dowolnego elementu (np. właściwości, zmiennej) służy „**=**”

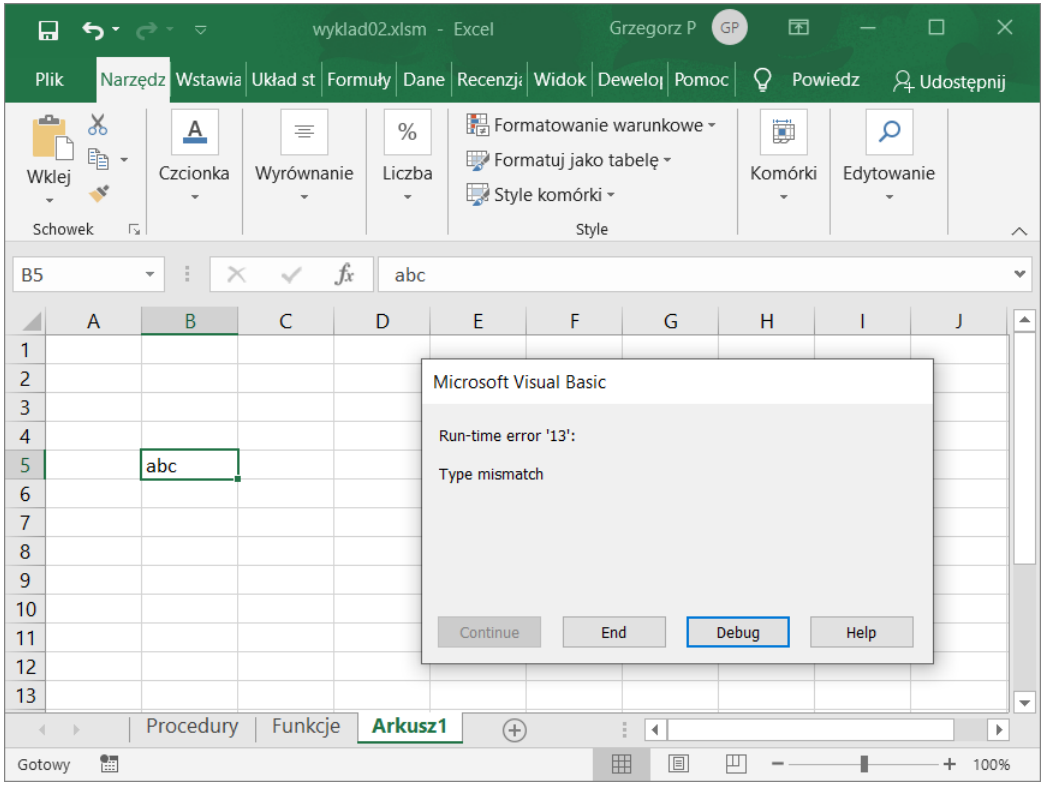
  `[Set] element = wartość`
- Operacje arytmetyczne: **`+`**, **`-`**, **`*`**, **`/`**, **`\`**, **`^`**, **`mod`**, **`( )`**
- Programy w VBA zapisywane są jako tzw. makra, dostępne są dwa typy makr: proceduralne (**`Sub`**) i funkcyjne (**`Function`**).

Błędy wykonania

Operacje arytmetyczne są zdefiniowana tylko dla wartości liczbowych (liczba całkowita, rzeczywista, waluta lub data), więc instrukcja:

```
ActiveCell.Offset(0, 1).Value = ActiveCell.Value + 2
```

zgłasza błąd, jeżeli aktywna komórka zawiera tekst (wartość String).



Ustawienia obsługi błędów

`On Error GoTo` *`<etykieta>`*

W przypadku błędu następuje skok do wskazanego miejsca w programie

`On Error Resume Next`

W przypadku błędu program kontynuuje wykonie kodu od następnego wiersza

`On Error GoTo` `0`

Anuluje aktualne ustawienia **`On Error`**, przywraca standardową obsługę błędów VBA.

`<etykieta>` to ciąg znaków zakończony ”**:**” (dwukropek), który wskazuje wiersz od którego rozpocznie się wykonanie kodu po wystąpieniu błędu.

Wznawianie programu po obsłudze błędu

`Resume`

Procedura wznawia wykonanie od wiersza, w którym wystąpił błąd

`Resume Next`

Procedura wznawia wykonanie od następnego wiersza po tym, w którym wystąpił błąd

Przykład I

Uzupełnienie procedury `InkrementujWPrawo` (s.1-37) o obsługę błędów – wersja 1. W przypadku błędu procedura wyświetla okno komunikatu i kończy działanie.

```
Public Sub InkrementujWPrawo1()
  On Error GoTo NieprawidłowaWartość
  With ActiveCell
    .Offset(0, 1).Value = .Value + 2
    .Offset(0, 1).Select
  End With
  Exit Sub
NieprawidłowaWartość:
  MsgBox "Wybierz komórkę zawierającą liczbę", vbCritical, "Błąd"
End Sub
```

Uwaga: Procedura wykonuje kolejne instrukcje do wystąpienia **`End Sub`**. Instrukcja **`Exit Sub`** umieszczona przed etykietą zapobiega wyświetlaniu komunikatu gdy błąd nie wystąpił.

Kod dostępny na stronie przedmiotu

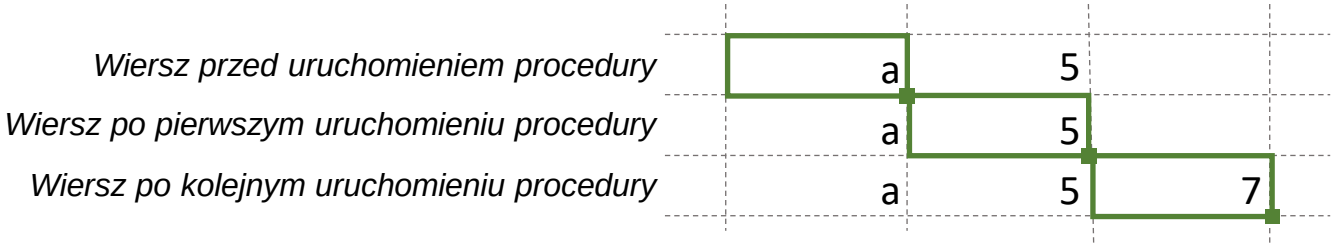
Przykład II

Uzupełnienie procedury `InkrementujWPrawo` (s.1-37) o obsługę błędów – wersja 2. W przypadku błędu procedura kontynuuje działanie od wiersza następnego.

```
Public Sub InkrementujWPrawo2()
  On Error Resume Next
  With ActiveCell
    .Offset(0, 1).Value = .Value + 2
    .Offset(0, 1).Select
  End With
End Sub
```

Kod dostępny na stronie przedmiotu

W przypadku komórek zawierających tekst operacja dodawania zgłasza błąd, który zostanie zignorowany (nie będzie wyświetlone okno komunikatu) i procedura wykona kolejną instrukcję (zaznaczenie komórki sąsiedniej).



Przykład III

Procedura kopiuje zawartość aktualnie wybranego zakresu do arkusza o nazwie "Kopia". Jeżeli arkusz "Kopia" nie istnieje zostanie utworzony.

```
Public Sub KopiujZakres()
  Dim a1 As Worksheet
  Dim a2 As Worksheet
  On Error GoTo DodajArkusz
  Worksheets("Kopia").Range(Selection.Address).Value =_
  Selection.Value
  Exit Sub
DodajArkusz:
  Set a1 = ActiveSheet
  Set a2 = Worksheets.Add()
  a2.Name = "Kopia"
  a1.Activate
  Resume
End Sub
```

Jeżeli arkusz "Kopia" nie istnieje wystąpi błąd

Obsługa błędu

Uwaga: po dodaniu nowego arkusza do kolekcji (metoda **Add**) Excel ustawia go jako aktywny, stąd konieczność "zapamiętania" aktualnie wybranego arkusza (zmienna **a1**) i powrót do niego przed zakończeniem bloku obsługi błędu.

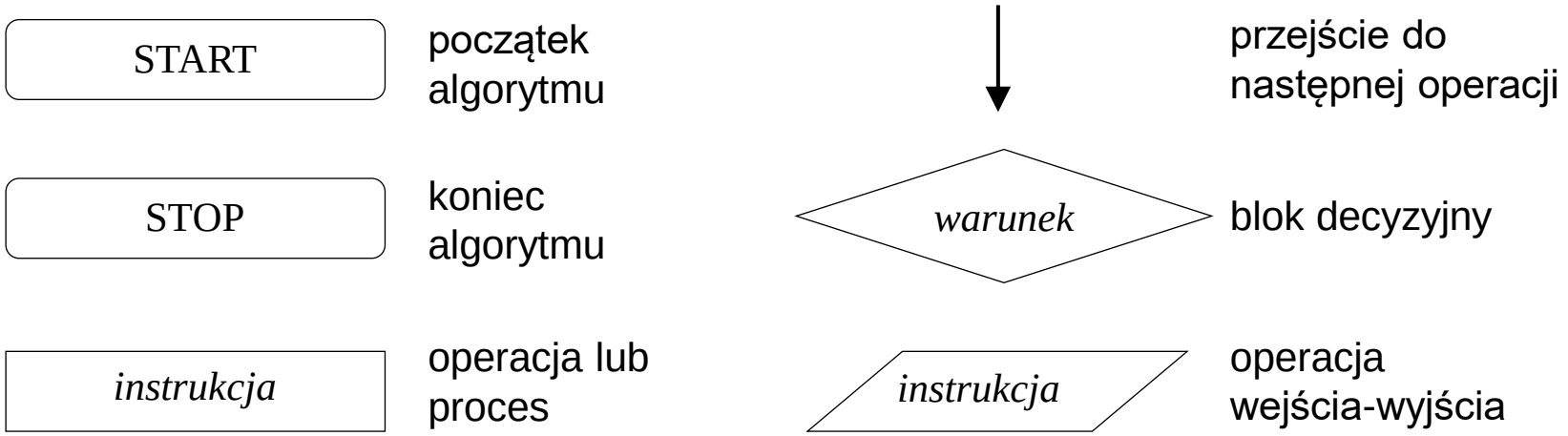
Kod dostępny na stronie przedmiotu

Projektowanie algorytmów

Algorytm – przepis postępowania prowadzący do rozwiązania określonego zadania; zbiór poleceń określających sposób przetwarzania zbioru danych ze wskazaniem kolejności w jakiej mogą być wykonane.

Realizacja każdego programu powinna być poprzedzona procesem projektowania, którego istotnym elementem jest zaplanowanie sposobu przetwarzania danych. Algorytm stanowi uniwersalny język zapisu, niezależny od docelowego narzędzia programowania.

Podstawowe symbole



Przykład

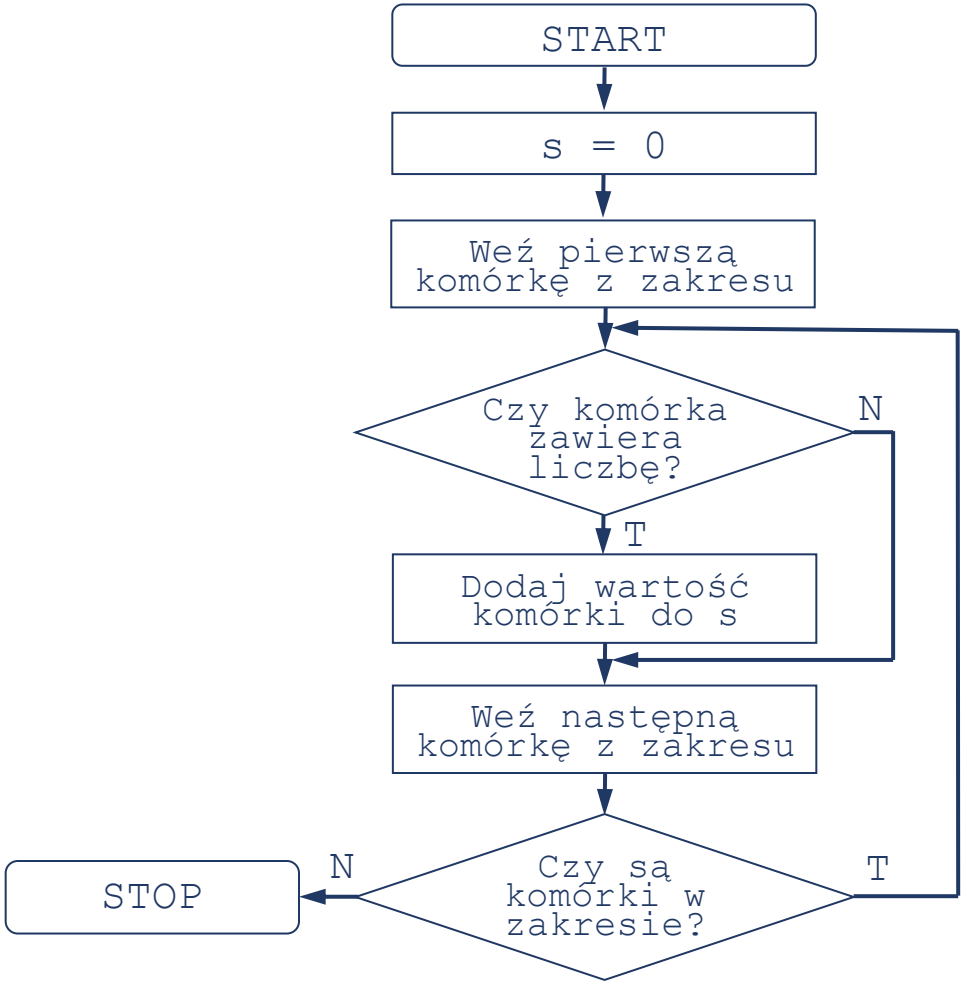
Zadanie: zaprojektować algorytm obliczający sumę wartości liczbowych w podanym zakresie arkusza.

	A	B	C
1	1		1
2	2		2
3	a		a
4	01.03.2022		01.03.2022
5	b		b
6	3		3
7	4		4
8	44631		10

=SUMA(A1:A7)

=SumaLiczb(C1:C7)

Uwaga: standardowa funkcja SUMA traktuje daty jak wartości liczbowe.



Instrukcje sterujące

Instrukcja sterująca – element języka programowania, który służy do określenia kolejności wykonania instrukcji zawartych w kodzie programu.

Instrukcje sterujące w VBA

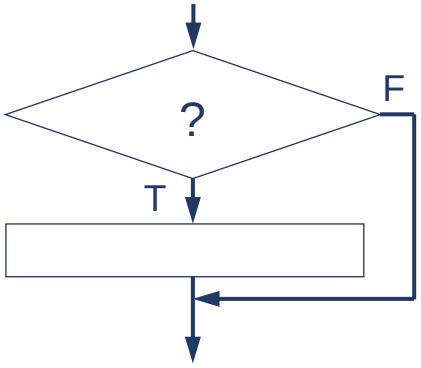
- **Instrukcja warunkowa** (**`If-Then`**, **`If-Then-Else`**) – wprowadza rozgałęzienie w kodzie programu, tworząc alternatywne sekwencje instrukcji.
- **Instrukcja wyboru** (**`Select Case`**) – wprowadza rozgałęzienie w kodzie programu tworząc dowolną liczbę alternatywnych sekwencji instrukcji.
- **Instrukcje iteracyjne (pętli)** (**`For-Each`**, **`For-Next`**, **`Do-Loop`**) – umożliwiają cykliczne powtarzanie sekwencji instrukcji.

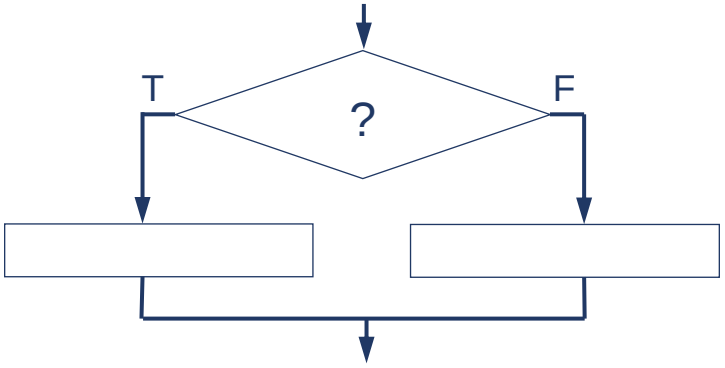
Dodatkowo

- **Funkcje wyboru wartości z listy** (**`Choose`**, **`Switch`**) – umożliwiają wybór wartości z predefiniowanej listy, zastępują szereg instrukcji warunkowych lub instrukcję wyboru.

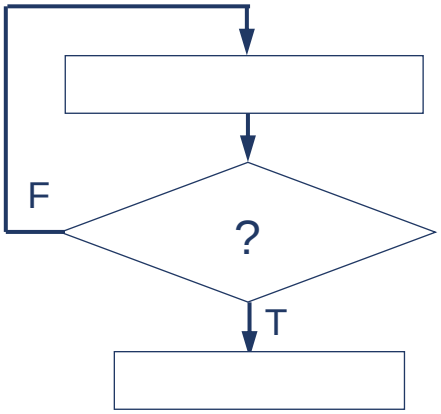
Instrukcje sterujące w algorytmach

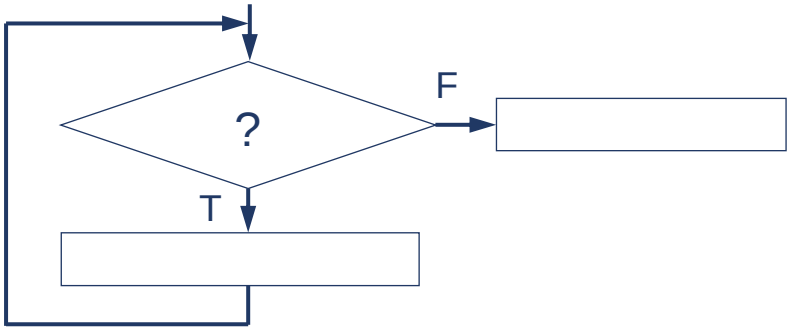
Instrukcje warunkowe





Instrukcje iteracyjne (pętli)

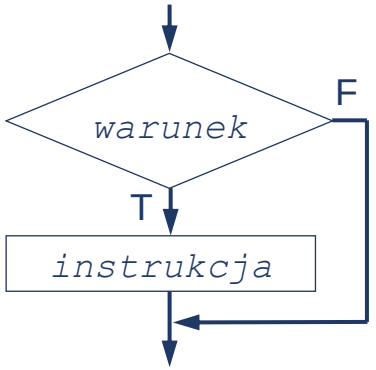




Instrukcja warunkowa

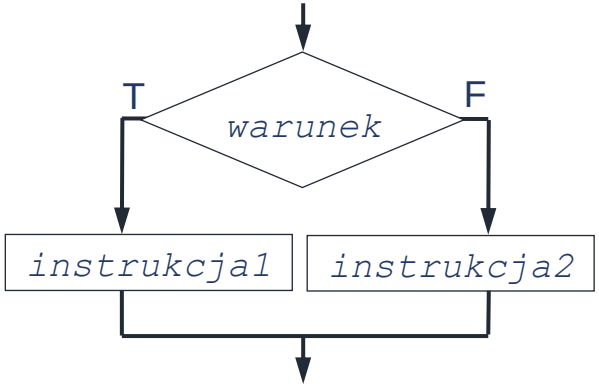
Jeżeli *warunek* jest prawdziwy wykonaj *instrukcję* (grupę instrukcji).

```
If warunek Then instrukcja

If warunek Then
  instrukcja1
  …
  instrukcjaN
End If
```



Jeżeli *warunek* jest prawdziwy wykonaj *instrukcję1* (pierwszą grupę instrukcji) w przeciwnym wypadku wykonaj *instrukcję2* (drugą grupę instrukcji).

```
If warunek Then instrukcja1 Else instrukcja2

If warunek Then
  instrukcja1-1
  …
  instrukcja1-N
Else
  instrukcja2-1
  …
  instrukcja2-N
End If
```



Operatory relacyjne i logiczne

Operatory relacyjne

`>`	większy
`<`	mniejszy
`>=`	większy lub równy
`<=`	mniejszy lub równy
`<>`	różny
`=`	równy

Operatory logiczne

`And`	koniunkcja
`Or`	alternatywa
`Not`	negacja

x	y	x And y	x Or y	Not x
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

Przykład (x jest zmienną liczbową)

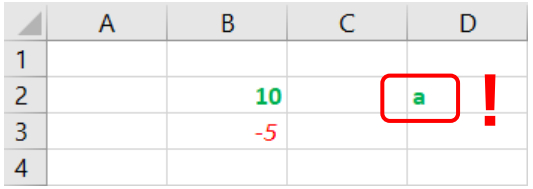
- Sprawdzenie czy x jest liczbą dodatnią: `x > 0`
- Sprawdzenie czy x jest różna od zera: `x <> 0`
- Sprawdzenie czy $x \in [-3,\ 5]$: `x >= -3 And x <= 5`
- Sprawdzenie czy $x \in (-\infty, -2] \cup (10,\ +\infty)$: `x <= -2 Or x > 10`

Procedura ustawia kolor zielony w komórkach zawierających wartości dodatnie i zero, czerwony w komórkach zawierających wartości ujemne.

```
Public Sub Koloruj1()
  If ActiveCell.Value >= 0 Then ActiveCell.Font.Color = vbGreen _
  Else ActiveCell.Font.Color = vbRed
End Sub
```

Procedura ustawia kolor zielony oraz pogrubienie w komórkach zawierających wartości dodatnie i zero, czerwony oraz kursywę w komórkach zawierających wartości ujemne.

```
Public Sub Koloruj2()
  If ActiveCell.Value >= 0 Then
    ActiveCell.Font.Color = vbGreen
    ActiveCell.Font.Bold = True
  Else
    ActiveCell.Font.Color = vbRed
    ActiveCell.Font.Italic = True
  End If
End Sub
```



Uwaga: tekst jest interpretowany jako liczba dodatnia.

Kod dostępny na stronie przedmiotu

Sprawdzenie typu zmiennej

Wybrane funkcje informacyjne VBA

- `IsDate(w)` – określa czy `w` może być konwertowane na typ `Date`,
- `IsEmpty(w)` – określa czy `w` jest puste/zainicjowane (tylko typ Variant),
- `IsNumeric(w)` – określa czy `w` jest liczbą (pusta komórka jest liczbą),
- `IsObject(w)` – określa czy `w` reprezentuje obiekt.

Uwaga: funkcje **`Is`**... zwracają wartość logiczną **`True`**/**`False`**.

Określenie nazwy/numeru typu danych

- `TypeName(zmienna)` – zwraca nazwę typu zmiennej jako tekst,
- `VarType(zmienna)` – zwraca kod typu zmiennej.

Kod	Nazwa
0	Empty
1	Null
2	Integer
3	Long
4	Single
5	Double
6	Currency
7	Date

Kod	Nazwa
8	String
9	Object
10	Error
11	Boolean
12	Variant
13	Object
14	Decimal
17	Byte

Przykład

Modyfikacja procedury `Koloruj2`. Dodatkowy warunek zapobiega modyfikacji ustawień czcionki, gdy wartość w komórce nie jest liczbą (warunek `.Vallue>=0` będzie sprawdzany tylko gdy komórka zawiera liczbę).

```
Public Sub Koloruj3()
  With ActiveCell
    If Not IsEmpty(.Value) And IsNumeric(.Value) Then
      If .Value >= 0 Then
        .Font.Color = vbGreen
        .Font.Bold = True
      Else
        .Font.Color = vbRed
        .Font.Italic = True
      End If
    End If
  End With
End Sub
```

	A	B	C	D
1				
2		10		a
3		-5		
4				

Kod dostępny na stronie przedmiotu

Zagnieżdżanie instrukcji sterujących

Zagnieżdżenie instrukcji – umieszczenie pewnej instrukcji sterującej w zasięgu działania innej instrukcji sterującej.

Przykład

```
If warunek1 Then                    'pierwsza instr. warunkowa
  if warunek2 Then instrukcja1      'druga instr. warunkowa
Else
  If warunek3 Then                  'trzecia instr. warunkowa
    instrukcja2
  Else
    instrukcja3
  End If
End If
```

Druga instrukcja warunkowa zostanie wykonana jeżeli `warunek1` jest prawdziwy (wykonanie `instrukcji1` zależy od `warunku1` i `warunku2`)

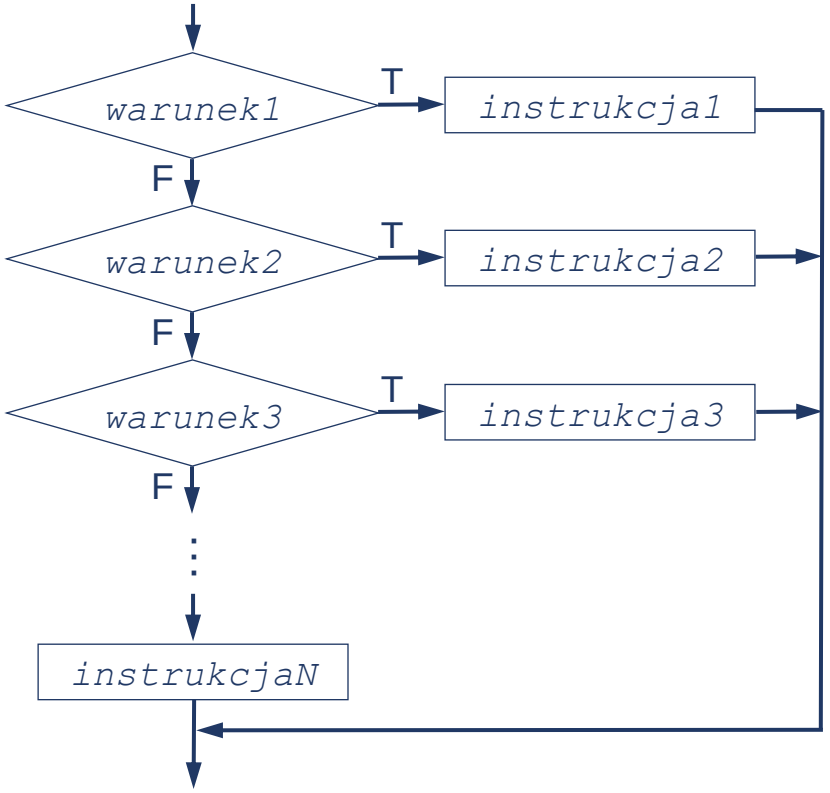
Trzecia instrukcja warunkowa zostanie wykonana jeżeli `warunek1` jest fałszywy (wykonanie `instrukcji2` i `instrukcji3` zależy od `warunku1` i `warunku3`)

Wcięcie – dodatkowy odstęp od lewego marginesu wprowadzany dla zagnieżdżonej instrukcji. Nie wpływa na wykonanie programu, zwiększa czytelność kodu.

Konstrukcja `ElseIf`

`ElseIf` jest opcjonalnym skladnikiem instrukcji **`If-End If`**. Może wystąpić wielokrotnie, każdy składnik wprowadza dodatkowy warunek.

```
If warunek1 Then
  instrukcja1-1
   …
  instrukcja1-N
ElseIf warunek2 Then
  instrukcja2-1
   …
  instrukcja2-N
ElseIf warunek3 Then
  instrukcja3-1
   …
  instrukcja3-N
ElseIf …
Else
  instrukcjaN-1
   …
  instrukcjaN-N
End If
```



Uwaga: warunek *i+1* będzie sprawdzony jeżeli fałszywy był warunek *i-ty*. W każdym przebiegu może zostać wykonana co najwyżej jedna grupa instrukcji. Instrukcje w sekcji **`Else`** (opcjonalne) zostaną wykonane jeżeli wszystkie warunki były fałszywe.

Przykład – funkcja Wiek

Funkcja określa wiek na podstawie daty urodzenia (niepełnoletni, pełnoletni, emeryt).

```
Public Function Wiek(dataUr As Date) As String
  If dataUr > Date Then
    Wiek = ""
  ElseIf DateAdd("yyyy", 18, dataUr) > Date Then
    Wiek = "niepełnoletni"
  ElseIf DateAdd("yyyy", 65, dataUr) > Date Then
    Wiek = "pełnoletni"
  Else
    Wiek = "emeryt"
  End If
End Function
```

	B	C	D	E
17				
18		**Data urodzenia**		**Wiek**
19		10.01.2020		niepełnoletni
20		25.08.2002		pełnoletni
21		15.06.1960		pełnoletni
22		01.12.2070		
23				

`=Wiek(C19)`
`=Wiek(C20)`
`=Wiek(C21)`
`=Wiek(C22)`

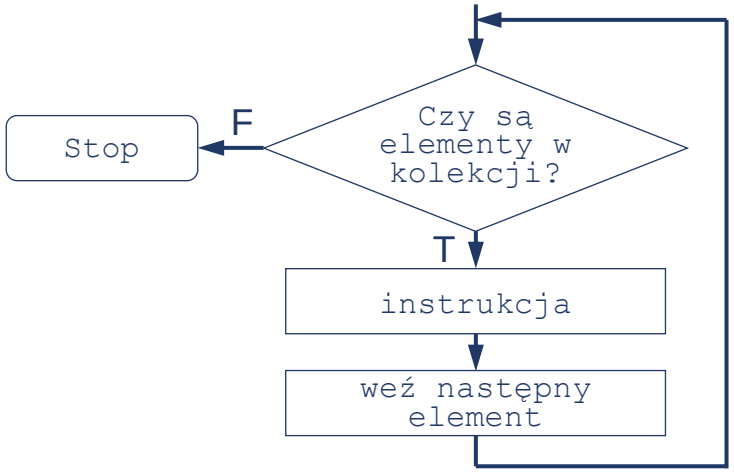
`Date` – funkcja zwraca aktualną datę

`DateAdd(`*`interwał`*`,`*`liczba`*`,`*`data`*`)` – do podanej *`daty`* dodaje *`interwał`* czasowy określony *`liczbą`* (możliwe interwały: `yyyy` – rok, `q` – kwartał, `m` – miesiąc, `d` – dzień, `ww`- tydzień, `h` – godzina, `n` – minuta, `s` – sekunda)

Przetwarzanie elementów kolekcji – `For Each`

Wykonaj instrukcję (grupę instrukcji) dla wszystkich elementów kolekcji.

```
For Each zmienna In kolekcja
   instrukcja1
   instrukcja2
   …
   instrukcjaN
Next
```



Pętla powtarza ciąg instrukcji tyle razy ile jest elementów w *`kolekcji`*. W kolejnych iteracjach (powtórzeniach) *`zmienna`* reprezentuje kolejny element *`kolekcji`*.

Kolekcje VBA

- `Cells` – kolekcja komórek arkusza,
- `Selection` – kolekcja komórek w aktualnie wybranym zakresie,
- `Rows`, `Columns` – kolekcja wierszy/kolumn,
- `Workbooks` – kolekcja aktualnie otwartych dokumentów,
- `Worksheets` – kolekcja arkuszy wybranego dokumentu.

Przykład – kolorowanie komórek

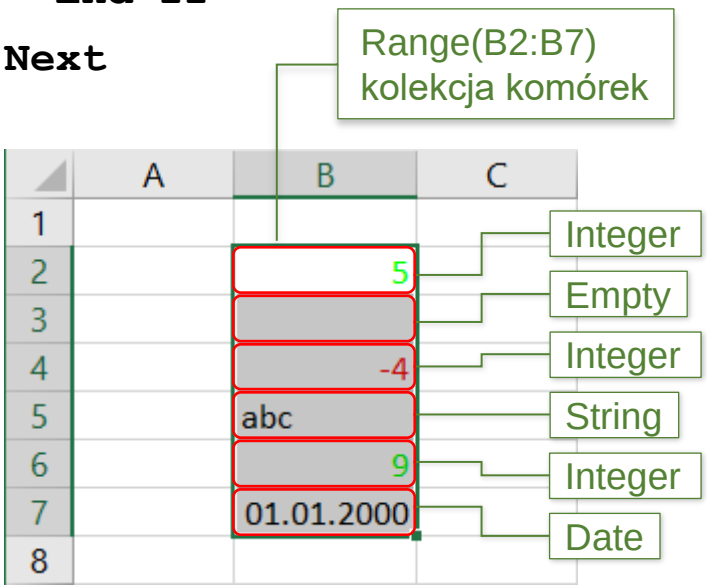
Procedura modyfikuje kolor wszystkich komórek zawierających wartości liczbowe we wskazanym zakresie (powtarza operacje zapisane w procedurze `Koloruj3`, s.10 dla wszystkich komórek w zakresie).

```
Public Sub KolorujKomórki1()
  Dim z As Range
  Dim k As Range
  On Error GoTo Wycofanie                                  wybór zakresu
  Set z = Application.InputBox("Wskaż zakres", Type:=8)
  For Each k In z                                          pętla
    If Not IsEmpty(k.Value) And IsNumeric(k.Value) Then
      If k.Value >= 0 Then k.Font.Color = vbGreen _
      Else k.Font.Color = vbRed
    End If                          ustawienie koloru gdy komórka zawiera liczbę
  Next
  Exit Sub
Wycofanie:
End Sub
```

Kod dostępny na stronie przedmiotu

Przykład – analiza

```
For Each k In z
  If Not IsEmpty(k.Value) And IsNumeric(k.Value) Then
    If k.Value >= 0 Then
      k.Font.Color = vbGreen
    Else
      k.Font.Color = vbRed
    End If
  End If
Next
```



Analiza (z=Range('B2:B7'))

1. k = Range('B2')
   wartość niepusta, liczba > 0
   Color = vbGreen
2. k = Range('B3')
   wartość pusta
   kolor nie jest modyfikowany
3. k = Range('B4')
   wartość niepusta, liczba < 0
   Color = vbRed
4. k = Range('B5')
   wartość niepusta, string
   kolor nie jest modyfikowany
5. k = Range('B6')
   wartość niepusta, liczba > 0
   Color = vbGreen
6. k = Range('B7')
   wartość niepusta, data
   kolor nie jest modyfikowany

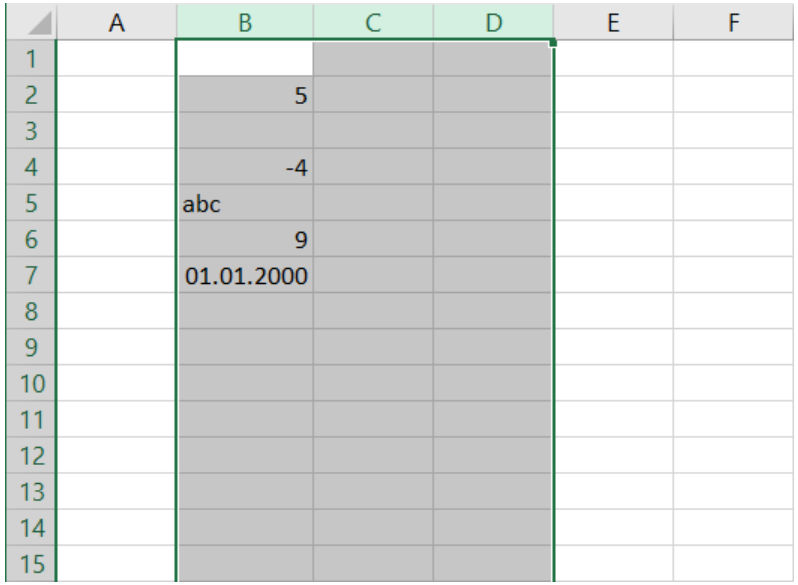
Przetwarzanie dużych zakresów komórek

Wybór dużego zakresu komórek może prowadzić do istotnego wydłużenia czasu wykonania makra (twórca makra nie ma wpływu na wielkość zakresu wybranego przez użytkownika). W wielu przypadkach zaznaczony zakres zawiera komórki puste lub zawierające wartości, które nie są przetwarzane, jednak ich obecność wymusza dodatkowe (puste) iteracje pętli.

Przykład

Zaznaczenie obejmuje kolumny od B do D, każda kolumna zawiera 1 048 576 komórek (Excel 2019). Wykonanie procedury `KolorujKomórki1` wymaga:

- 3 x 1 048 576 = 3 145 728 iteracji,
- czas wykonania (i7 3.4GHz) ok. 25s
- komórki zawierające wartości: 5
- w tym wartości liczbowe: 3
- 3 145 725 iteracji pustych



Działania na zakresach komórek

Wybór komórek arkusza z wartościami określonego typu (metoda obiektu Range)

```
SpecialCells(Type As XlCellType, _
             Value As XlSpecialCellsValue) As Range
```

`Type` określa typ wartości:

- `xlCellTypeBlanks` – puste komórki
- `xlCellTypeConstants` – komórki zawierające wartości stałe
- `xlCellTypeFormulas` – komórki zawierające formuły
- `xlCellTypeVisible` – komórki widoczne

`Value` (opcjonalny) określa szczegółowy typ wartości:

- `xlErrors` – komórki z błędami
- `xlLogical` – komórki z wartościami logicznymi
- `xlNumbers` – komórki z wartościami liczbowymi
- `xlTextValues` – komórki z wartościami tekstowymi

Operacje na zakresach (metody obiektu Application)

```
Union(z1 As Range, z2 As Range, z3, z4, … z30)
Intersect(z1 As Range, z2 As Range, z3, z4, … z30)
```

Metody wyznaczają sumę oraz część wspólną zakresów `z1...z30` przekazanych jako argumenty (wymagane co najmniej dwa niepuste zakresy).

Przykład – kolorowanie komórek, wersja 2

Modyfikacja procedury `KolorujKomórki1` (s.15). Wskazany przez użytkownika zakres jest ograniczony tylko do komórek zawierających wartości numeryczne.

```
Public Sub KolorujKomórki2()
  Dim z As Range
  Dim k As Range
  On Error GoTo Wycofanie
  Set z = Application.InputBox("Wskaż zakres", Type:=8)
  Set z = z.SpecialCells(xlCellTypeConstants, xlNumbers)
  For Each k In z
    If IsNumeric(k.Value) Then
      If k.Value >= 0 Then k.Font.Color = vbGreen _
      Else k.Font.Color = vbRed
    End If
  Next
  Exit Sub
Wycofanie:
End Sub
```

wskazanie zakresu i wybór komórek, które zawierają wartości numeryczne

Uwaga: sprawdzenie czy komórka jest niepusta w tym przypadku nie jest potrzebne, zakres zawiera wyłącznie wartości numeryczne (w tym daty).

Kod dostępny na stronie przedmiotu

Funkcja SumaLiczb

Funkcja oblicza sumę wartości liczbowych (z wyłączeniem dat) w podanym zakresie.

```
Public Function SumaLiczb1(zakres As Range) As Double
  Dim k As Range
  Dim s As Double
  s = 0
  For Each k In zakres
    If Not IsEmpty(k.Value) And IsNumeric(k.Value) Then
      s = s + k.Value
    End If
  Next
  SumaLiczb1 = s
End Function
```

Uwaga: funkcja może być uzupełniona o wstępny wybór komórek zawierających wartości numeryczne. Obydwie wersja są dostępne na stronie przedmiotu.

	J	K	L
26			
27		SumaLiczb1	
28		1	
29		2	
30		a	
31		1 mar 22	
32		b	
33		3	
34		4	
35		10	

=SumaLiczb1(K28:K34)

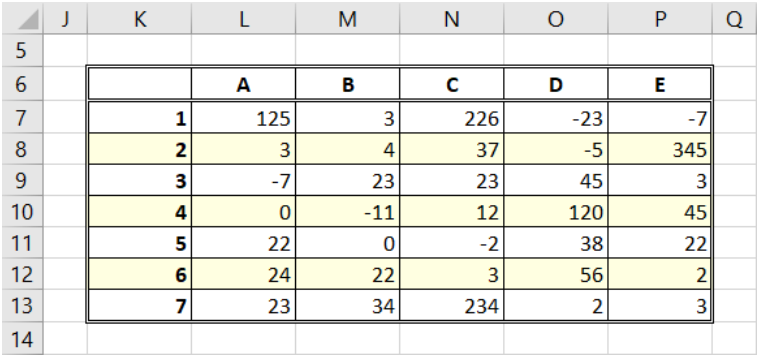
```
Analiza, SumaLiczb1("K28:K34")
0. s=0
1. k=Range("K28"), s=0+1=1
2. k=Range("K29"), s=1+2=3
3. k=Range("K30")
4. k=Range("K31")
5. k=Range("K32")
6. k=Range("K33"), s=3+3=6
7. k=Range("K34"), s=6+4=10
```

Kod dostępny na stronie przedmiotu

Przykład – kolorowanie wierszy

Procedura ustawia kolor tła w parzystych wierszach wskazanego zakresu.

```
Public Sub KolorujWiersze()
  Dim z As Range
  Dim w As Range
  Dim i As Integer
  On Error GoTo Wycofanie
  Set z = Application.InputBox(…)
  i = 1
  For Each w In z.Rows
    If i Mod 2 = 0 Then wiersz.Interior.Color = RGB(255,255,225) _
    Else  wiersz.Interior.ColorIndex = xlColorIndexNone
    i = i + 1
  Next
Wycofanie:
End Sub
```



	A	B	C	D	E
1	125	3	226	-23	-7
2	3	4	37	-5	345
3	-7	23	23	45	3
4	0	-11	12	120	45
5	22	0	-2	38	22
6	24	22	3	56	2
7	23	34	234	2	3

Uwaga 1: pętla wykonuje operacje dla każdego wiersza w wybranym zakresie (kolekcja **`z.Rows`**).

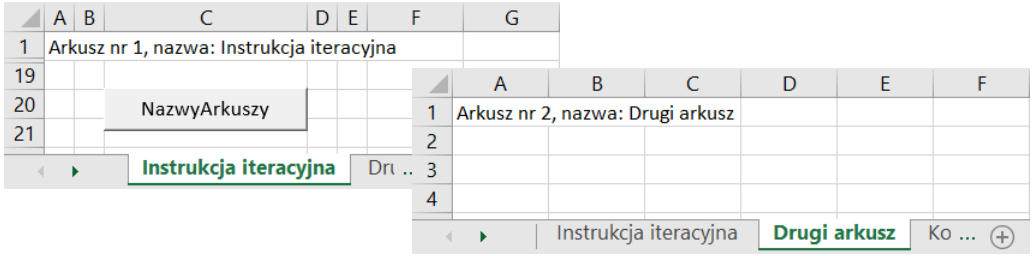
Uwaga 2: zmienna **`i`** jest powiększana w każdym przebiegu pętli (jest licznikiem wierszy).
`i Mod 2 = 0` gdy numer wiersza jest parzysty.

Kod dostępny na stronie przedmiotu

Przykład – przetwarzanie kolekcji arkuszy

Procedura wypełnia pierwszą komórkę każdego arkusza w bieżącym dokumencie wartością zawierającą numer arkusza i jego nazwę.

```
Public Sub NazwyArkuszy()
  Dim arkusz As Worksheet

  For Each arkusz In ActiveWorkbook.Worksheets
    arkusz.Cells(1, 1).Value = "Arkusz nr " & arkusz.Index & _
                               ", nazwa: " & arkusz.Name
  Next
End Sub
```



Uwaga 1: pętla wykonuje operacje dla <u>każdego arkusza</u> w bieżącym dokumencie (kolekcja **`ActiveWorkbook.Worksheets`**).

Uwaga 2: Operator „`&`” łączy wartości tekstowe tworząc pojedynczy łańcuch znakowy (konkatenacja łańcuchów), np. **`"abc" & "def" = "abcdef"`**.

Kod dostępny na stronie przedmiotu