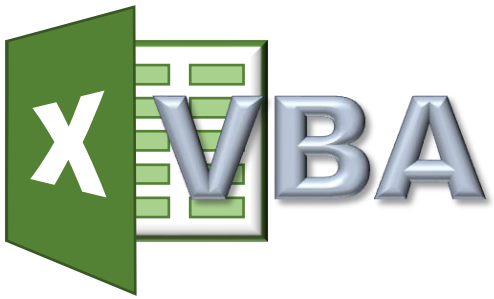
Techniki programowania



Zmienne, obsługa błędów, typy makr

operacje arytmetyczne, zmienne i stałe

obsługa błędów

procedury i funkcje użytkownika

Podsumowanie Wykładu01

- Makro jest procedurą zapisaną w VBA.
- Aplikacja jest zbiorem obiektów, każdy zawiera pewne właściwości i metody.
- Najważniejsze obiekty Excel-a: **Application**, **Workbook**, **Worksheet**, **Range.**
- Każda komórka, zaznaczenie, kolumna, wiersz, itp. są obiektem typu **Range**.
- Odwołanie do właściwości (metoda analogicznie):

```
nazwa_obiektu.nazwa_właściwości
```

- Aplikacja może zawierać kolekcję Workbook-ów (obiekt **Workbooks**), jeden Workbook zazwyczaj zawiera kolekcję arkuszy (obiekt **Worksheets**).
- Odwołanie do kolekcji:

```
nazwa_kolekcji(indeks)
```

- Do zmiany wartości dowolnego elementu (np. właściwości) służy „=”

```
element = wartość
```

- Obiekty: **Application**, aktywny **Workbook** i aktywny **Worksheet** są domyślne, na ogół mogą być pominięte w odwołaniach.

Notacja

- **`Słowa kluczowe języka`** (np. nazwy instrukcji) – czcionka prosta, pogrubienie,
- `Elementy predefiniowane` (np. nazwy standardowych funkcji) – czcionka prosta, bez pogrubienia,
- `[elementy opcjonalne]` – nawiasy kwadratowe,
- Elementy alternatywne (należy wybrać jeden z nich) są rozdzielane pionową kreską (znak |),
- *`<nazwa>`* – tekst w nawiasach ostrych, kursywa,
- *`opis`* – kursywa.

Przykład

```
[Private|Public] Sub <nazwa>
```

- Pierwsze słowo to **`Private`** lub **`Public`**, jest opcjonalne (nawiasy kwadratowe)
- Słowo **`Sub`** musi wystąpić zawsze
- Po słowie **`Sub`** występuje nazwa wprowadzana przez użytkownika
- **`Private`**, **`Public`** i **`Sub`** to słowa kluczowe (pogrubienie)

Instrukcja przypisania, operatory arytmetyczne

Instrukcja przypisania

Instrukcja przypisania pozwala na modyfikację każdego elementu programu, który nie został zdefiniowany z atrybutem read-only (tylko do odczytu)

```
<element> = wyrażenie
```

Operatory arytmetyczne

+	dodawanie	****	dzielenie całkowite
-	odejmowanie	**mod**	dzielenie modulo
*****	mnożenie	**^**	potęgowanie
/	dzielenie	**( )**	grupowanie działań

Operacje wykonywane są od lewej do prawej z zachowaniem priorytetów (po pierwsze potęgowanie, następnie mnożenie i dzielenie, później dodawane i odejmowanie). Nie można pomijać symboli działań, nawiasy okrągłe zmieniają kolejność operacji.

Uwaga: $x + 5/3*y = x + \frac{5}{3}y,$ $(x + 5)/(3*y) = \frac{x+5}{3y}$

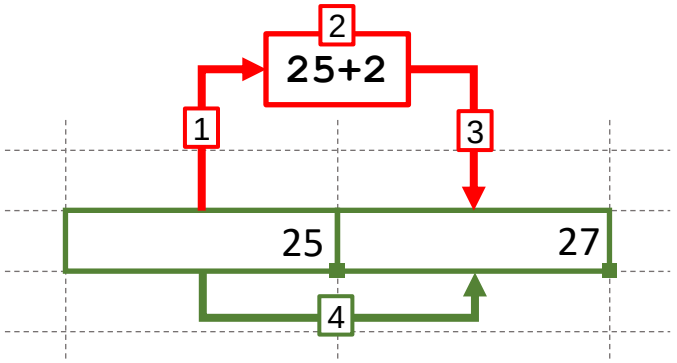
Przykład

Procedura pobiera wartość aktualnej komórki, powiększa ją o dwa, wynik zapisuje w komórce znajdującej się na prawo (ten sam wiersz, następna kolumna) i zaznacza ją.

Odwołania

- Aktualnie wybrana komórka: `ActiveCell`
- Odwołanie do wartości aktualnej komórki: `ActiveCell.Value`
- Odwołanie do wartości komórki sąsiedniej: `ActiveCell.Offset(0,1).Value`
- Zaznaczenie komórki sąsiedniej: `ActiveCell.Offset(0,1).Activate`

```
Public Sub InkrementujWPrawo()
  With ActiveCell
3 .Offset(0, 1).Value = .Value + 2
4 .Offset(0, 1).Activate
  End With
End Sub
```



Kod dostępny na stronie przedmiotu

Zmienna – element języka programowania używany do przechowywania danych. Podczas pracy programu wartości zmiennej mogą ulegać modyfikacji.

Stała – symbol reprezentujący pewną niezmienną wartość, która nie może być zmodyfikowana podczas pracy programu.

Schemat deklaracji zmiennej

```
Dim <nazwa> As typ
```

Schemat deklaracji stałej

```
Const <nazwa> [As typ] = wartość
```

<nazwa> musi być unikalną (nie mogą istnieć dwa elementy o takiej samej nazwie), powinna spełniać warunki opisane w punkcie 2 na wykładzie 1. s.1-21.

Przykłady

```
Dim x As Integer
Dim komórka As Range
Const VAT As Single = 0.23
```

Uwaga: Deklaracja zmiennych nieobiektowych nie jest wymagana. Niezadeklarowana zmienna otrzymuje typ **Variant**. Deklaracja zmiennych może być wymuszona przez umieszczenie dyrektywy **Option Explicit** na początku modułu.

Ustawianie wartości zmiennych

Ustawienie wartości zmiennej nieobiektowej

```
<nazwa> = wyrażenie
```

Ustawienie wartości zmiennej obiektowej

```
Set <nazwa> = wyrażenie
```

<nazwa> jest nazwą zmiennej

wyrażenie jest dowolnym wyrażeniem VBA (w tym również wartością stałą, nazwą zmiennej, itp.) o typie zgodnym z typem zmiennej.

Przykłady

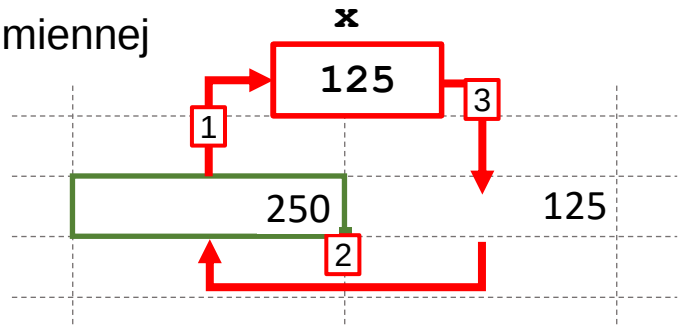
```
Dim x As Integer
Dim y As Integer
Dim k As Range

x = 5
y = 29 + x ^ 2
Set k = Worksheets("Arkusz5").Range("C3:E5")
```

Przykład

Procedura zamienia miejscami wartość komórki aktywnej i komórki znajdującej się na prawo od niej (ten sam wiersz kolejna kolumna).

Algorytm

1. Odczytaj wartość komórki aktywnej do zmiennej
2. Zapisz w komórce aktywnej wartość komórki po prawej
3. Zapisz w komórce po prawej wartość zapisaną w zmiennej




```
Public Sub ZamieńWPrawo()
  Dim x As Integer
  x = ActiveCell.Value
  ActiveCell.Value = ActiveCell.Offset(0, 1).Value
  ActiveCell.Offset(0, 1).Value = x
End Sub
```

Kod dostępny na stronie przedmiotu

Uwaga: Procedura działa prawidłowo tylko dla wartości typu **`Integer`**.

Do przemyślenia (1): Zamiana dowolnych wartości.

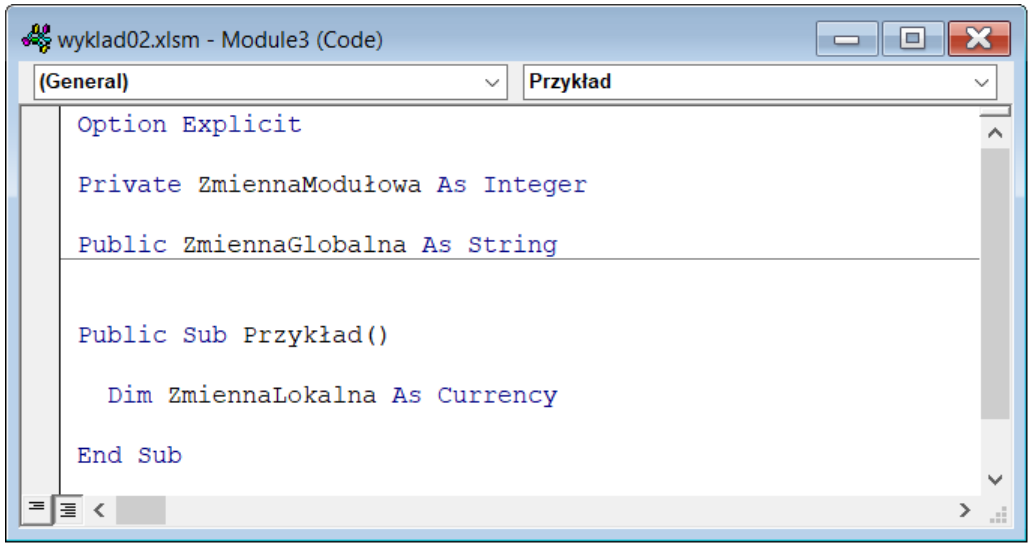
Do przemyślenia (2): Zamiana całego zakresu.

Czas życia i zasięg zmiennych

Zmienna lokalne – zmienna zadeklarowana wewnątrz procedury VBA przy pomocy **`Dim`**. Istnieje tylko podczas wykonania procedury, jest usuwana po jej zakończeniu.

Zmienna modułowa – zmienna zadeklarowana na poziomie modułu (przed pierwszą procedurą) przy pomocy **`Dim`** lub **`Private`** (równoważne, ale zalecane **`Private`**). Jest dostępna we wszystkich procedurach modułu, istnieje przez cały czas wykonania programu.

Zmienna globalna – zadeklarowana na poziomie modułu przy pomocy **`Public`**. Jest dostępna we wszystkich modułach, istnieje przez cały czas wykonania programu.

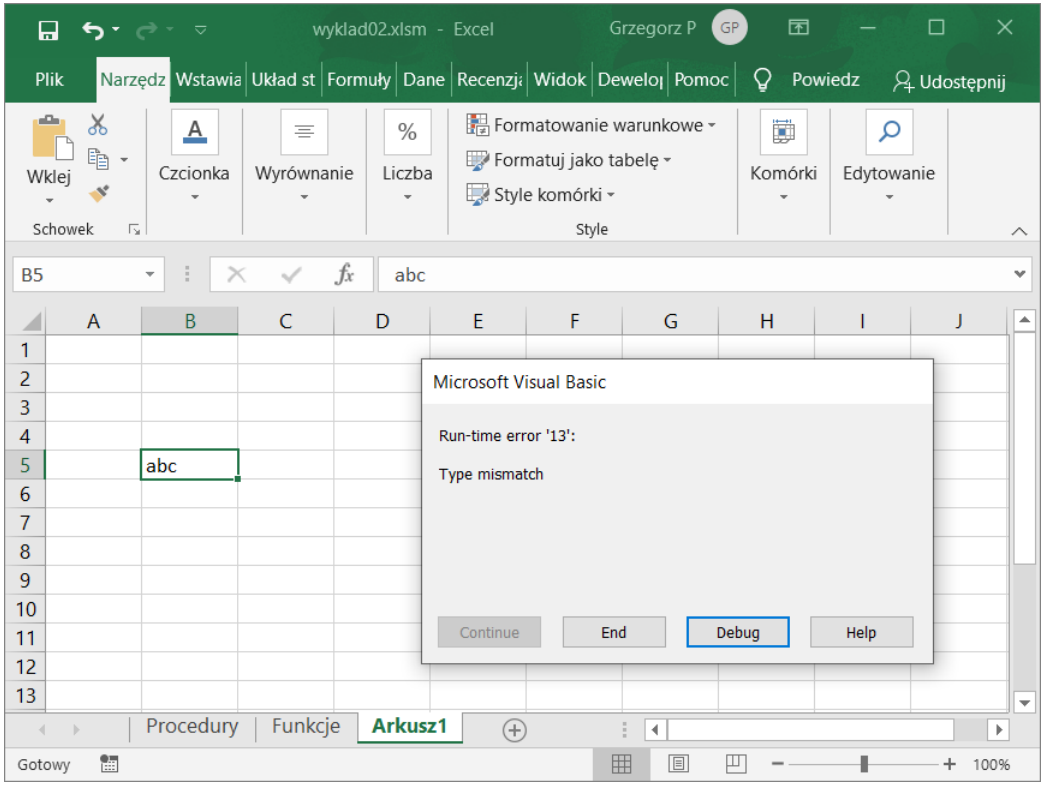


Błędy wykonania

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.5) 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 instrukcje do wystąpienia **`End Sub`**. Instrukcja **`Exit Sub`** (przerwanie wykonywania procedury) umieszczona przed etykietą zapobiega wyświetlaniu komunikatu gdy błąd nie wystąpił.

Kod dostępny na stronie przedmiotu

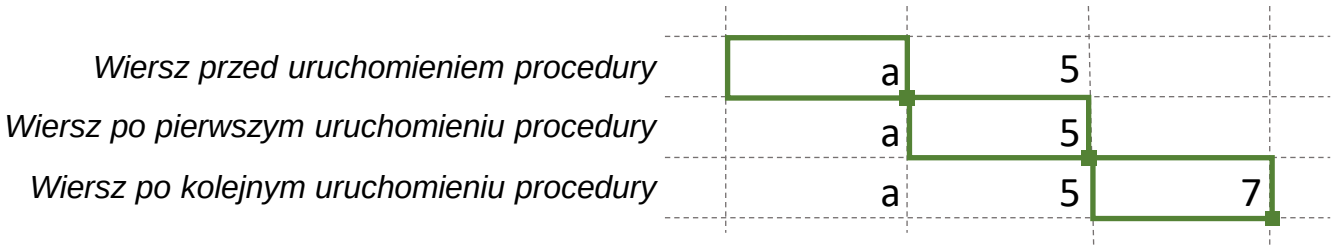
Przykład II

Uzupełnienie procedury `InkrementujWPrawo` (s.5) 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

```
[Private|Public] Sub <nazwa>([arg1, arg2, … , argN])
  miejsce na kod procedury
End Sub
```

- **Sub** jest słowem kluczowym języka określa nagłówek procedury (subroutine)
- **Private** oznacza procedurę prywatną, dostępną tylko w module
- **Public** oznacza procedurę publiczną, dostępną w dowolnym miejscu programu
- Słowa **Private** i **Public** są opcjonalne, domyślnie procedury są publiczne
- *<nazwa>* jest nazwą nadawaną przez użytkownika (patrz wykład 1., s.1-21, punkt 2.)
- *arg1, arg2, … argN* to opcjonalna lista argumentów zdefiniowanych jako:

  `[Optional] <nazwa_argumentu> As typ [= wartość_domyślna]`

- **Optional** oznacza argument opcjonalny, który może być pominięty podczas wywołania (w takim przypadku przyjmuje *wartość_domyślną*
- Procedura wykonuje kolejne instrukcje do wystąpienia **End Sub**
- Działanie procedury można przerwać umieszczając w kodzie instrukcję **Exit Sub**

Schemat wywołania procedury

```
<nazwa_procedury> [arg1, arg2, … , argN]
```

- W wywołaniu procedury nie występują nawiasy
- *<nazwa_procedury>* określa procedurę standardową lub zdefiniowaną przez użytkownika i dostępną w aktywnym arkuszu
- *arg1, … argN* odpowiadają kolejnym argumentom zgodnie z definicją zapisaną w nagłówku procedury
- Każdy argument opcjonalny może być pominięty (puste miejsce)
- Argumenty procedury można podać w dowolnej kolejności określając ich nazwy zgodnie ze schematem:

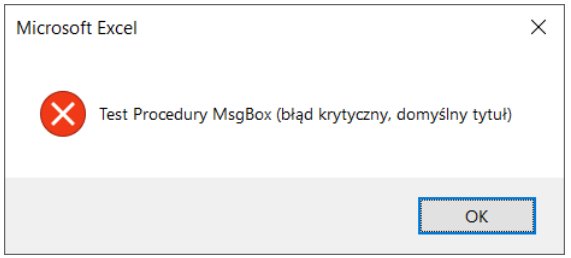
```
<nazwa_argumentu> := wartość
```

Przykład – MsgBox

Nagłówek

```
MsgBox(prompt As String, _
       Optional buttons As VbMsgBoxStyle = vbOKOnly, _
       Optional title As String = "Microsoft Excel")
```

- `prompt` – tekst wyświetlany w oknie komunikatu
- `buttons` – zestaw przycisków dostępnych w oknie (`vbOKOnly` – tylko przycisk OK) oraz typ okna (wyświetlana ikona). Dostępne wartości: `vbCritical`, `vbQuestion`, `vbExclamation`, `vbInformation` (błąd krytyczny, pytanie, ostrzeżenie, informacja)
- `title` – tytuł okna (domyślnie „Microsoft Excel”)



Przykład wywołania

```
MsgBox "Test Procedury MsgBox"
MsgBox "Test Procedury MsgBox, vbCritical
MsgBox "Test Procedury MsgBox", vbQuestion, "Mój komunikat"
MsgBox "Test Procedury MsgBox", , "Mój komunikat"
MsgBox buttons:=vbExclamation, title:="Mój komunikat", _
       prompt:="Test Procedury MsgBox"
```

Kod dostępny na stronie przedmiotu

```
[Private|Public] Function <nazwa>([arg1,… , argN]) As typ
  miejsce na kod funkcji
  <nazwa> = wartość
End Function
```

- **Function** jest słowem kluczowym języka określa nagłówek funkcji
- **Private|Public** są opcjonalne, określają dostępność funkcji, funkcja publiczna jest traktowana jak standardowa funkcja arkuszowa i może być użyta w formułach
- *<nazwa>* jest nazwą nadawaną przez użytkownika (patrz wykład 1., s.1-21, punkt 2.)
- *typ* określa typ wartości zwracanej jako wynik działania funkcji
- Wynik funkcji jest określany przez przypisanie *wartości* do *<nazwy>*
- *arg1,… argN* to opcjonalna lista argumentów, definicja jak w procedurze (s.2-6)
- Funkcja wykonuje kolejne instrukcje do wystąpienia **End Function**
- Działanie funkcji można przerwać umieszczając w kodzie instrukcję **Exit Function**
- Obsługa błędów może być realizowana tak jak w przypadku procedur

Wywołanie funkcji

Schemat wywołania funkcji

```
zmienna = <nazwa_funkcji>[(arg1, … , argN)]
```

- W wywołaniu funkcji nawiasy są wymagane jeżeli następuje przekazanie argumentów. W przypadku funkcji bezargumentowej są dozwolone, ale niewymagane.
- `<nazwa_funkcji>` określa procedurę standardową lub zdefiniowaną przez użytkownika i dostępną w aktywnym arkuszu.
- `zmienna` oznacza nazwę zmiennej, w której zostanie umieszczona wartość zwracana przez funkcję.
- `arg1, … argN` odpowiadają kolejnym argumentom zgodnie z definicją zapisaną w nagłówku procedury.
- Każdy argument opcjonalny może być pominięty (puste miejsce)
- Argumenty funkcji można podać w dowolnej kolejności określając ich nazwy zgodnie ze schematem:

```
<nazwa_argumentu> := wartość
```

Komunikacja z użytkownikiem – InputBox

Funkcja `InputBox`

```
InputBox(prompt As String,
         Optional title As String = "Microsoft Excel",
         Optional default As String = "") As String
```

Metoda `InputBox` (klasa Application)

```
InputBox(prompt As String,
         Optional title As Variant = "Microsoft Excel",
         Optional default As Variant = "", …
         Optional type As Variant) As Variant
```

- `prompt` – tekst wyświetlany w oknie
- `title` – tytuł okna (domyślnie „Microsoft Excel”)
- `default` – wartość domyślna
- `type` – typ odczytywanej wartości (tylko metoda `InputBox`): 0 – formuła, 1 – liczba, 2 – tekst, 4 – wartość logiczna, 8 – zakres (**Range**).

Uwaga: Zarówno funkcja jak i metoda `InputBox` ma cztery dodatkowe parametry (położenie okna i odwołanie do systemu pomocy), które nie zostały opisane.

Przykład – InputBox

Procedura wypełnia wskazany zakres określoną wartością. Do odczytu zakresu wykorzystano metodę `InputBox`, do odczytu wartości funkcję `InputBox`.

```
Public Sub Wypełnij()
  Dim Zakres As Range
  Dim wartość As String
  On Error Goto Wycofanie
  Set Zakres = Application.InputBox("Wskaż zakres", type:=8)
  wartość = InputBox("Podaj wartość")
  Zakres.Value = wartość
  Exit Sub
  Wycofanie:
End Sub
```

błąd wykonania

Uwaga: Obsługa błędu zapewnia prawidłową reakcję programu w przypadku naciśnięcia przycisku Anuluj w oknie InputBox (metoda obiektu Application). W takiej sytuacji zwracana jest wartość pusta, która nie może być przypisana do zmiennej obiektowej Zakres (wartość pusta nie jest obiektem, więc próba przypisania prowadzi do zgłoszenia błędu wykonania), co powoduje skok do etykiety Wycofanie i zakończenie działania procedury.

Kod dostępny na stronie przedmiotu

Funkcje użytkownika – przykład

```
Const DomyślnyVAT As Single = 0.23

Public Function CenaBrutto1(netto As Currency,
                            vat As Single) As Currency
  CenaBrutto1 = netto + netto * vat
End Function

Public Function CenaBrutto2(netto As Currency,
                            Optional vat As Single = DomyślnyVAT) As Currency
  CenaBrutto2 = netto + netto * vat
End Function
```

	A	B	C	D	E	F	G	H
1	**Funkcja CenaBrutto1**					**Funkcja CenaBrutto2**		
2	**Netto**	**VAT**	**Brutto**			**Netto**	**Brutto**	
3	127,00 zł	23%	156,21 zł			127,00 zł	156,21 zł	
4	250,00 zł	23%	307,50 zł			250,00 zł	307,50 zł	
5	35,00 zł	8%	37,80 zł			35,00 zł	37,80 zł	
6	875,00 zł	23%	1 076,25 zł			875,00 zł	1 076,25 zł	
7	1 230,00 zł	8%	1 328,40 zł			1 230,00 zł	1 328,40 zł	
8	27,00 zł	23%	33,21 zł			27,00 zł	33,21 zł	
9								

=CenaBrutto2(F3)

=CenaBrutto2(F5;0,08)

=CenaBrutto1(A3;B3)

Kod dostępny na stronie przedmiotu