

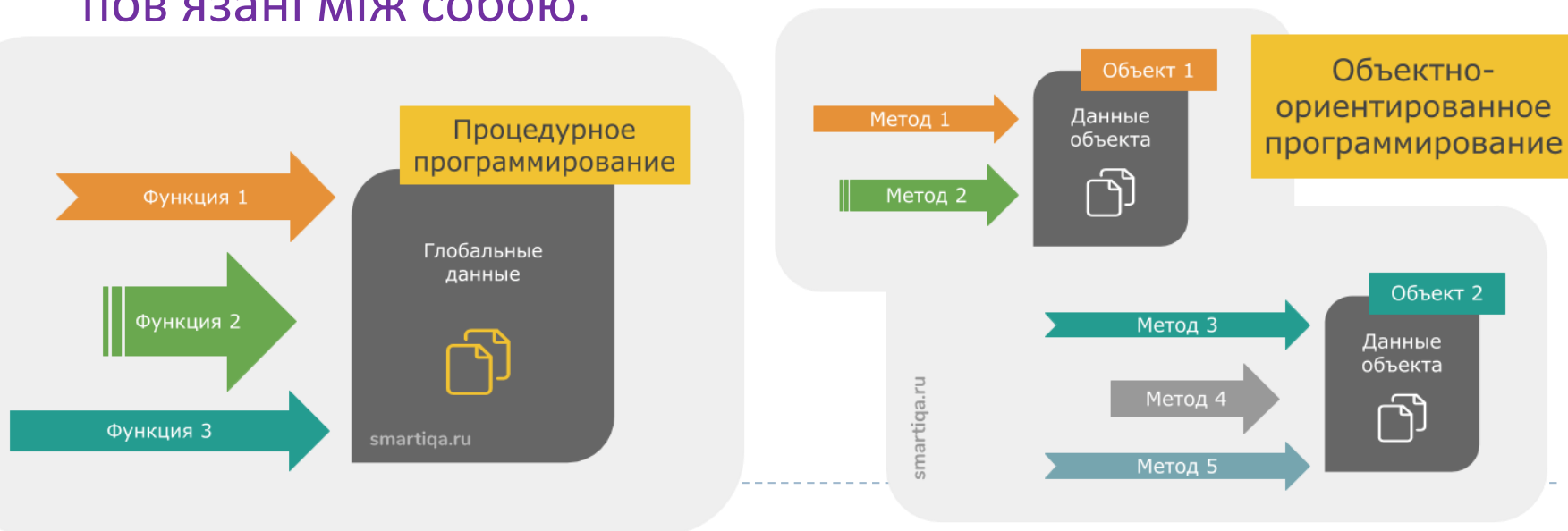
Офісне програмування

Засоби VBA для роботи з об'єктами

Слайди до лекцій
(2 частина)

Класи і об'єкти VBA

- ▶ У звичайному модулі дані та підпрограми (функції і процедури) функціонально не пов'язані між собою. Це означає, наприклад, що оголошуючи деяку змінну на рівні модуля, не можна задати їй різні значення для двох процедур.
- ▶ У модулі класу (при об'єктно-орієнтованому програмуванні (ООП)) дані та підпрограми логічно пов'язані між собою.



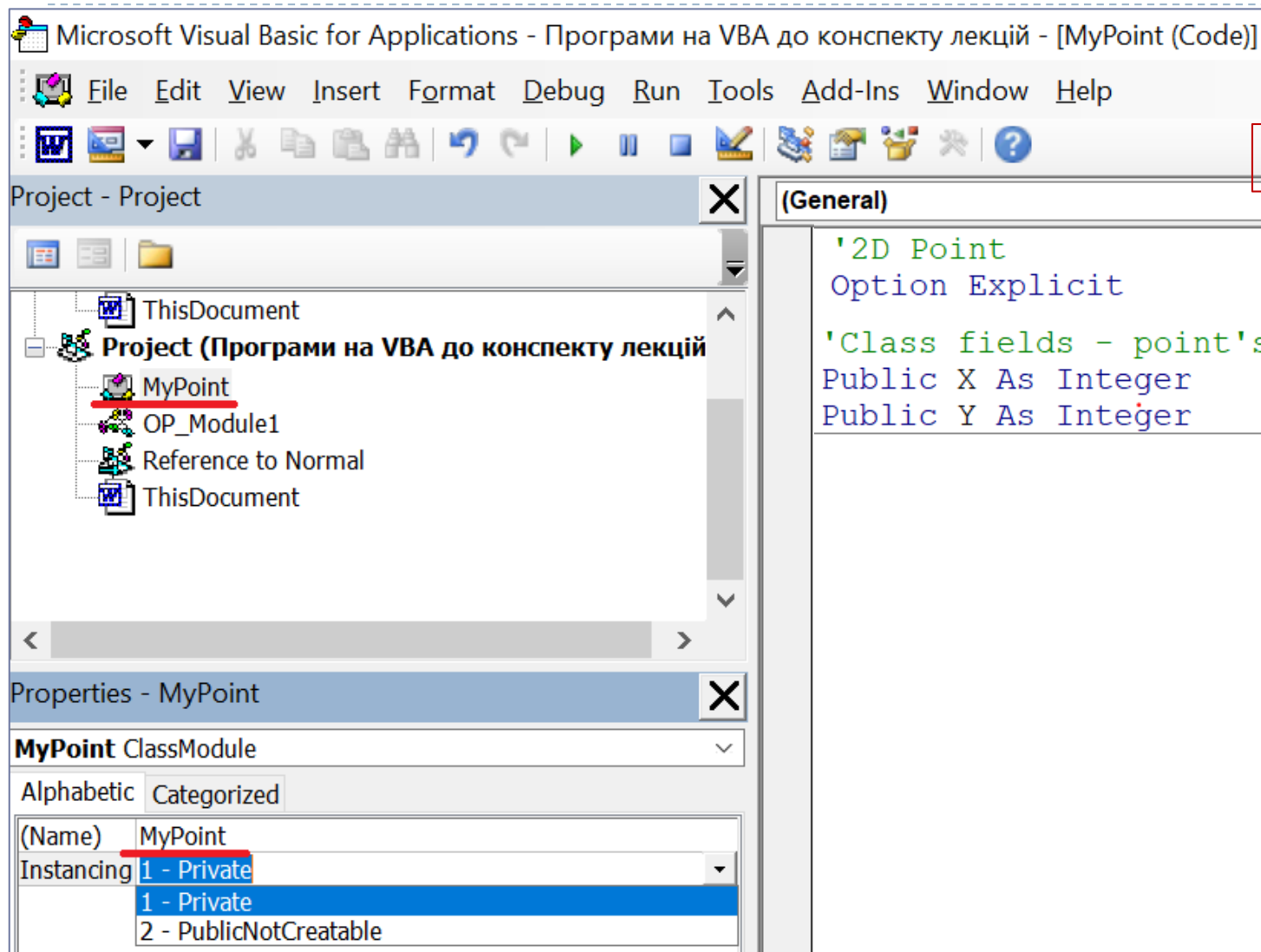
Класи і об'єкти VBA

- ▶ **Клас** - складений тип даних, в якому описується структура об'єкта. **Об'єкт** - основний будівельний матеріал для написання ООП програм. Об'єкт, створений на основі класу, можна називати **екземпляром класу**.

Клас має наступну структуру:

- ▶ **Поле** - елемент класу для зберігання даних,
- ▶ **Властивість** - елемент класу для зберігання даних з можливістю їх отримання-запису,
- ▶ **Метод** - аналог процедури або функції,
- ▶ **Подія** - сигнал при зміні стану об'єкта, наприклад, виконання методу або зміни даних.

Створення класу MyPoint



Область декларацій



Створення та видалення об'єктів

```
Option Explicit
```

```
Option Private Module
```

```
Private Sub TestPoint()
```

```
    Dim p1 As MyPoint
```

```
    Set p1 = New MyPoint
```

```
    Debug.Print "x=" & str(p1.X), "y=" & str(p1.Y)
```

```
    Set p1 = Nothing
```

```
End Sub
```

У вікні Immediate після запуску процедури



x= 0 y= 0

Set змінна-об'єкт = [**New**] вираз-об'єкта | **Nothing**



Конструктор та деструктор класу MyPoint

'2D Point

Option Explicit

'Class fields - point's coordinates

Public X As Integer

Public Y As Integer

'Constructor

Private Sub **Class_Initialize()**

 X = 1

 Y = 1

End Sub

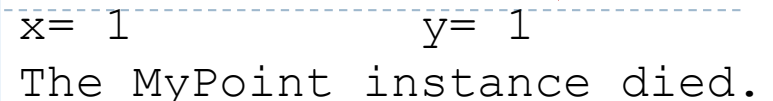
'Destructor

Private Sub **Class_Terminate()**

 Debug.Print "The Point instance died."

End Sub

У вікні Immediate після запуску процедури
TestPoint()



```
x= 1                y= 1
The MyPoint instance died.
```

Створення та видалення об'єктів

(General)

TestPoint

```
Option Explicit  
Option Private Module
```

```
Private Sub TestPoint()  
    Dim p1 As MyPoint  
  
    Set p1 = New MyPoint  
    Debug.Print "x=" & str(p1.X), "y=" & str(p1.Y)  
    p1.X = 3  
    p1.Y = 4  
    Print "x=" & str(p1.X), "y=" & str(p1.Y)  
  
    Set p1 = Nothing  
  
End Sub
```

Доступ до публічного поля об'єкту класу

Створення класу AdvPoint - інкапсуляція

Microsoft Visual Basic for Applications - Програми на VBA до конспекту лекцій - [AdvPoint (Code)]

File Edit View Insert Format Debug Run Tools Add-Ins Window Help

Project - Project

- ThisDocument
- Project (Програми на VBA до конспекту лекцій)**
 - AdvPoint
 - MyPoint
 - OP_Module1
 - OP_Module2
 - Reference to Normal
 - ThisDocument

Properties - AdvPoint

AdvPoint ClassModule

Alphabetic Categorized

(Name)	AdvPoint
Instancing	1 - Private

(General)

```
'2D Point (advanced)

'Class fields -
Private iX As Integer
Private iY As Integer

'Constructor
Public Sub Class_Initialize
    iX = 1
    iY = 1
End Sub

'Destructor
Private Sub Class_Terminate
    Debug.Print "AdvPoint destroyed"
End Sub
```

Add Procedure

Name: X

Type:

- ☐ Sub
- ☐ Function
- ☒ Property

Scope:

- ☒ Public
- ☐ Private

☐ All Local variables as Statics

OK Cancel

Створення класу AdvPoint - інкапсуляція

'Properties accessors

Public Property Get X() As Integer

 X = iX

End Property

Public Property Let X(ByVal iXValue As Integer)

If (iXValue >= 0) Then

 iX = iXValue

Else

 iX = 0

 Debug.Print "Attempt to assign negative value, X=0"

End If

End Property

Public Property Get Y() As Integer

 Y = iY

End Property

...



Створення класу AdvPoint - інкапсуляція

...

```
Public Property Let Y(ByVal iYValue As Integer)
    If (iYValue >= 0) Then
        iY = iYValue
    Else
        iY = 0
        Debug.Print "Attempt to assign negative value, Y=0"
    End If
End Property
```



Робота з об'єктом AdvPoint

```
Private Sub TestAdvPoint()  
    Dim ap1 As New AdvPoint
```

```
    Debug.Print "X=" & str(ap1.X), "Y=" & str(ap1.Y)  
    ap1.X = -1  
    ap1.X = 4  
    ap1.Y = 5  
    Debug.Print "X=" & str(ap1.X), "Y=" & str(ap1.Y)
```

```
    Debug.Print "Object instances are deleted automatically  
when Sub ends:"  
End Sub
```

У вікні Immediate після запуску процедури
TestAdvPoint()

X= 1 Y= 1

Attempt to assign negative value, X=0

X= 4 Y= 5

Object instances are deleted automatically when Sub ends:

The AdvPoint instance with X= 4 , Y= 5 died.

Методи об'єктів

'Returns distance between this point and another point
'with coordinates-parameters

Public Function DistanceCoord(aX As Integer, aY As Integer)

As Double

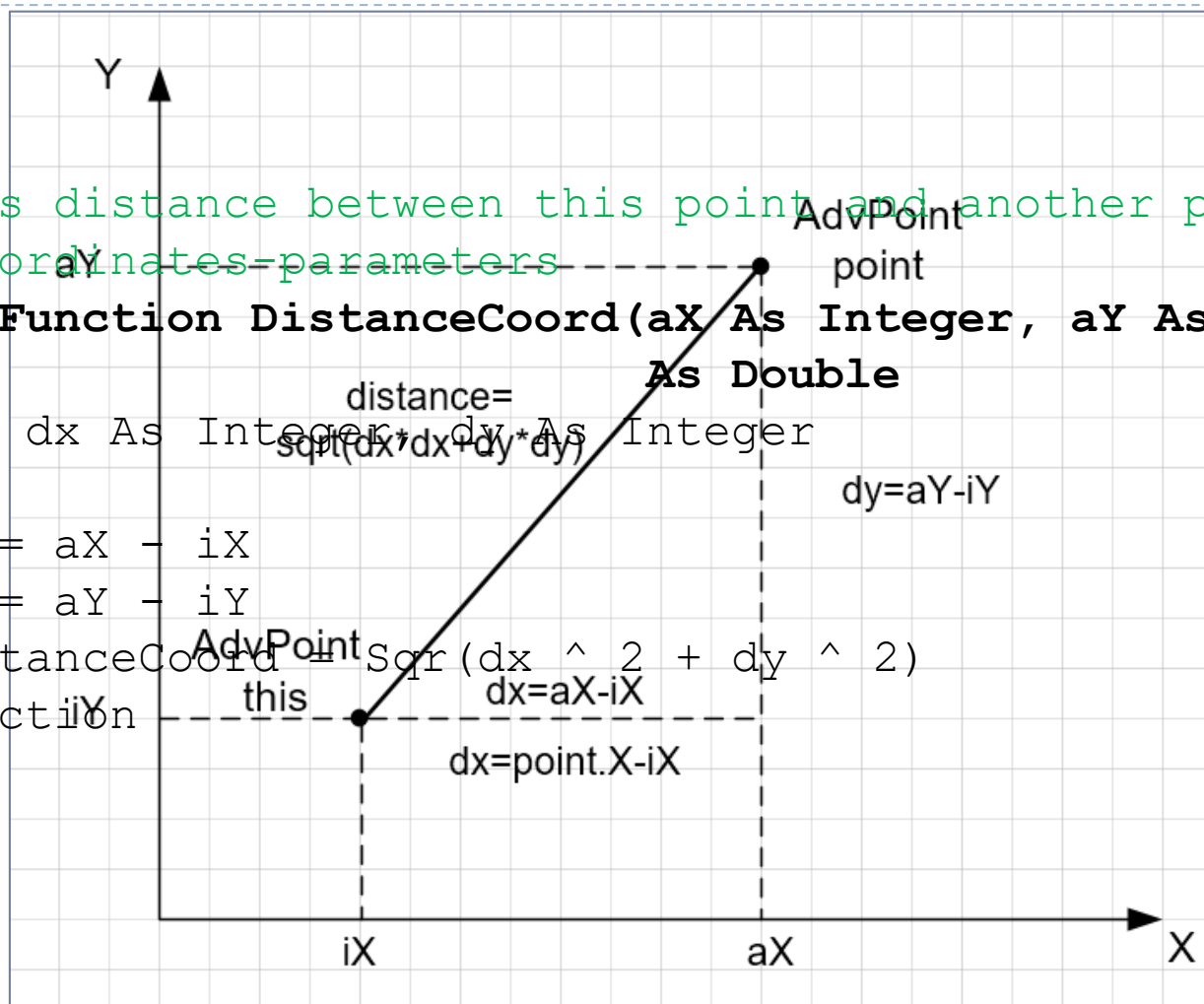
Dim dx As Integer, dy As Integer
distance = Sqr(dx * dx + dy * dy)

dx = aX - iX

dy = aY - iY

DistanceCoord = Sqr(dx ^ 2 + dy ^ 2)

End Function



Методи об'єктів

```
Private Sub TestAdvPoint()  
    Dim ap1 As New AdvPoint  
  
    Debug.Print "X=" & str(ap1.X), "Y=" & str(ap1.Y)  
    ap1.X = -1  
    ap1.X = 4  
    ap1.Y = 5  
    Debug.Print "X=" & str(ap1.X), "Y=" & str(ap1.Y)  
  
    Debug.Print "Distance = "; ap1.DistanceCoord(1, 1)  
  
    Debug.Print "Object instances are deleted automatically  
when Sub ends:"  
End Sub
```

У вікні Immediate після запуску процедури
TestAdvPoint()



```
Distance = 5
```

Методи об'єктів

'Returns distance between this point and another
'point-parameters

```
Public Function DistancePoint(point As AdvPoint) As Double  
    Dim dx, dy As Integer  
  
    dx = point.X - iX  
    dy = point.Y - iY  
    DistancePoint = Sqr(dx * dx + dy * dy)  
End Function
```



Методи об'єктів

```
Private Sub TestAdvPoint()  
    Dim ap1 As New AdvPoint  
  
    Debug.Print "X=" & str(ap1.X), "Y=" & str(ap1.Y)  
    ap1.X = -1  
    ap1.X = 4  
    ap1.Y = 5  
    Debug.Print "X=" & str(ap1.X), "Y=" & str(ap1.Y)  
  
    Debug.Print "Distance = "; ap1.DistanceCoord(1, 1)  
  
    Debug.Print "Distance = "; ap1.DistancePoint(ap2)  
  
    Debug.Print "Object instances are deleted automatically  
when Sub ends:"  
End Sub
```

**У вікні Immediate після запуску процедури
TestAdvPoint()**



Distance = 5

Події об'єктів

- ▶ **Подія** - це дія, яка розпізнається об'єктом, і у відповідь на яку можна запрограмувати певні дії.

Кожен клас вже має дві вбудованих події:

- ▶ **Class_Initialize** - відбувається при створенні екземпляра класу. У цій події зручно задавати значення полів об'єкту за замовчуванням.
 - ▶ **Class_Terminate** - відбувається при видаленні екземпляра класу.
 - ▶ Можливе додавання **власних подій** в клас, які будуть відбуватися за певних умов, при цьому екземпляр класу з подіями повинен бути оголошений в об'єктному модулі (модулі класу, форми, листа, книги, документу і т.і.) на рівні модуля.
-



Події об'єктів – додання події у клас AdvPoint

Microsoft Visual Basic for Applications - Програми на VBA до конспекту лекцій - [AdvPoint (Code)]

File Edit View Insert Format Debug Run Tools Add-Ins Window Help

Project - Project Class Initialize

'Returns distance between this point and another point-parameters

Public Function DistancePoint(point As AdvPoint) As Double

Dim dx, dy As Integer

dx = point.X - iX

dy = point.Y - iY

DistancePoint = Sqr(dx * dx + dy * dy)

End Function

Public Event DistanceCalculated()

'Class fields - point's coordinates

Private iX As Integer

Private iY As Integer

Constructor

Public Sub Class_Initialize()

iX = 1

iY = 1

End Sub

Destructor

Private Sub Class_Terminate()

Debug.Print "The AdvPoint instance with X="; iX; ", Y="; iY; "died."

End Sub

Fire event

RaiseEvent DistanceCalculated

AdvPoint Class Module

Alphabetic Categorized

(Name) AdvPoint

Instancing 1 - Private

Події об'єктів – використання події у класі MyPoint

The screenshot displays the Microsoft Visual Basic for Applications environment. The title bar reads "Microsoft Visual Basic for Applications - Програми на VBA до конспекту лекцій - [MyPoint (Code)]". The menu bar includes File, Edit, View, Insert, Format, Debug, Run, Tools, Add-Ins, Window, and Help. The Project Explorer on the left shows a project named "Project (Програми на VBA до конспекту лекцій)" containing AdvPoint, MyPoint (highlighted with a red circle), OP_Module1, OP_Module2, Reference to Normal, and ThisDocument. The Properties window for MyPoint shows its name and instancing set to "1 - Private". The Code window displays the VBA code for the MyPoint class module, with three specific lines highlighted and numbered in red:

```

X = 1
Y = 1
End Sub

'Destructor
Private Sub Class_Terminate()
    Debug.Print "The Point instance died."
End Sub

Public Sub CalcDistance() 2
    Dim p2 As New AdvPoint
    Dim d As Double
    Set p1 = New AdvPoint
    p1.DistancePoint p2
End Sub

'Event handler
Private Sub p1 DistanceCalculated() 3
    MsgBox "Event raized."
End Sub

'Destructor
Private Sub Class_Terminate()
    Debug.Print "The Point instance died."
End Sub

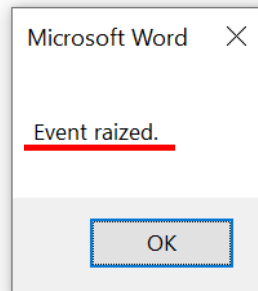
```

The event handler is named "DistanceCalculated", which is also reflected in the dropdown menu on the right side of the Code window.

Події об'єктів – тестування об'єкту з подією у звичайному модулі

```
Private Sub TestEvent()  
    Dim point As New MyPoint  
  
    point.CalcDistance  
End Sub
```

Програми на VBA до конспекту лекцій "Офісне програмування"



Демонстрація MyPoint, AdvPoint

Перегляд об'єктів - *Object Browser* (F2)

Microsoft Visual Basic for Applications - Програми на VBA до конспекту лекцій - [Object Browser]

File Edit View Insert Format Debug Run Tools Add-Ins Window Help

Project - Project

Normal
Project (ОП-лекції-модуль-2)
Project (Програми на VBA до конспекту лекцій)
 Class1
 OP_Module1
 OP_Module2
 Reference to Normal
 ThisDocument

Properties - ThisDocument

ThisDocument Document

Alphabetic Categorized

(Name)	ThisDocument
AutoFormatOverride	False
AutoHyphenation	False
ChartDataPointTrack	True
ConsecutiveHyphensLimit	0
DefaultTabStop	35,4
DefaultTargetFrame	
DisableFeatures	False

Project

- <All Libraries>
- IWshRuntimeLibrary
- Normal
- Office
- Project
- stdole
- VBA
- Word

Бібліотеки

Classes

- <globals>
- Class1 Клас
- Man Користувачський тип
- MyType1
- MyType2
- OP_Module1 модуль
- OP_Module2
- ThisDocument Клас

Members of 'ThisDocument'

- AcceptAllRevisions
- AcceptAllRevisionsShown
- Activate метод
- ActiveTheme властивість
- ActiveThemeDisplayName
- ActiveWindow
- ActiveWritingStyle
- AddToFavorites
- Application
- ApplyDocumentTheme
- ApplyQuickStyleSet2
- ApplyTheme

Project Project

C:\Users\kgp\Dropbox\EDU\ОП\Програми на VBA до конспекту лекцій.docm

Найпростіші форми та вставка елементів управління в документ

Microsoft Word

Введіть число елементів вектора

InputBox

OK

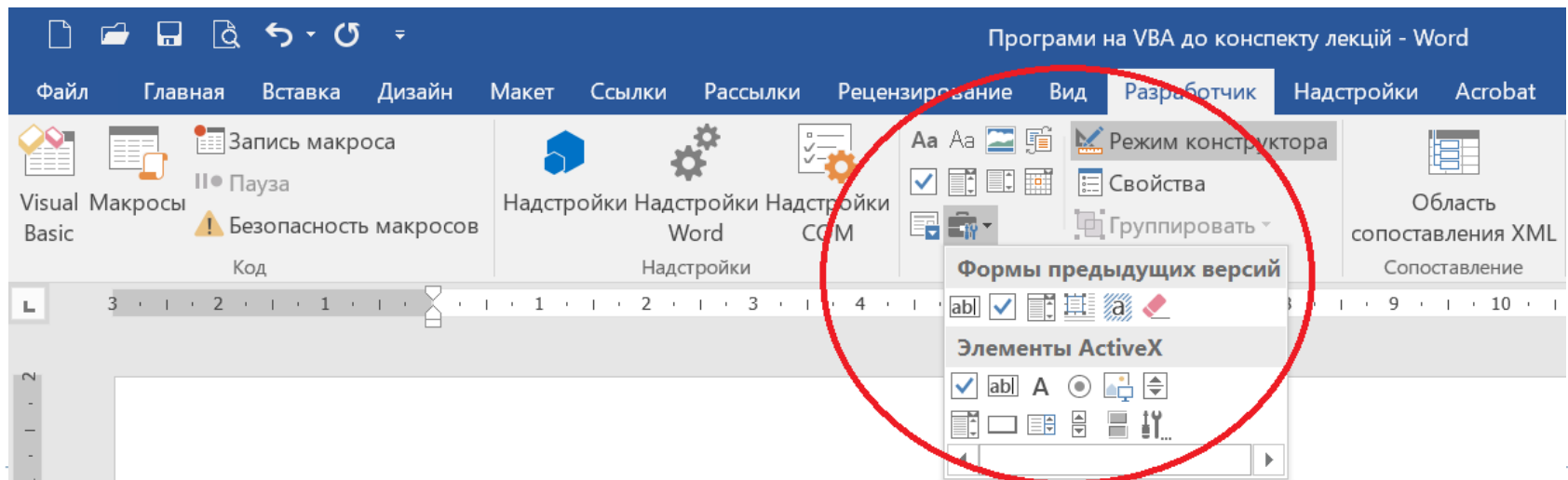
Cancel

Microsoft Word

MsgBox

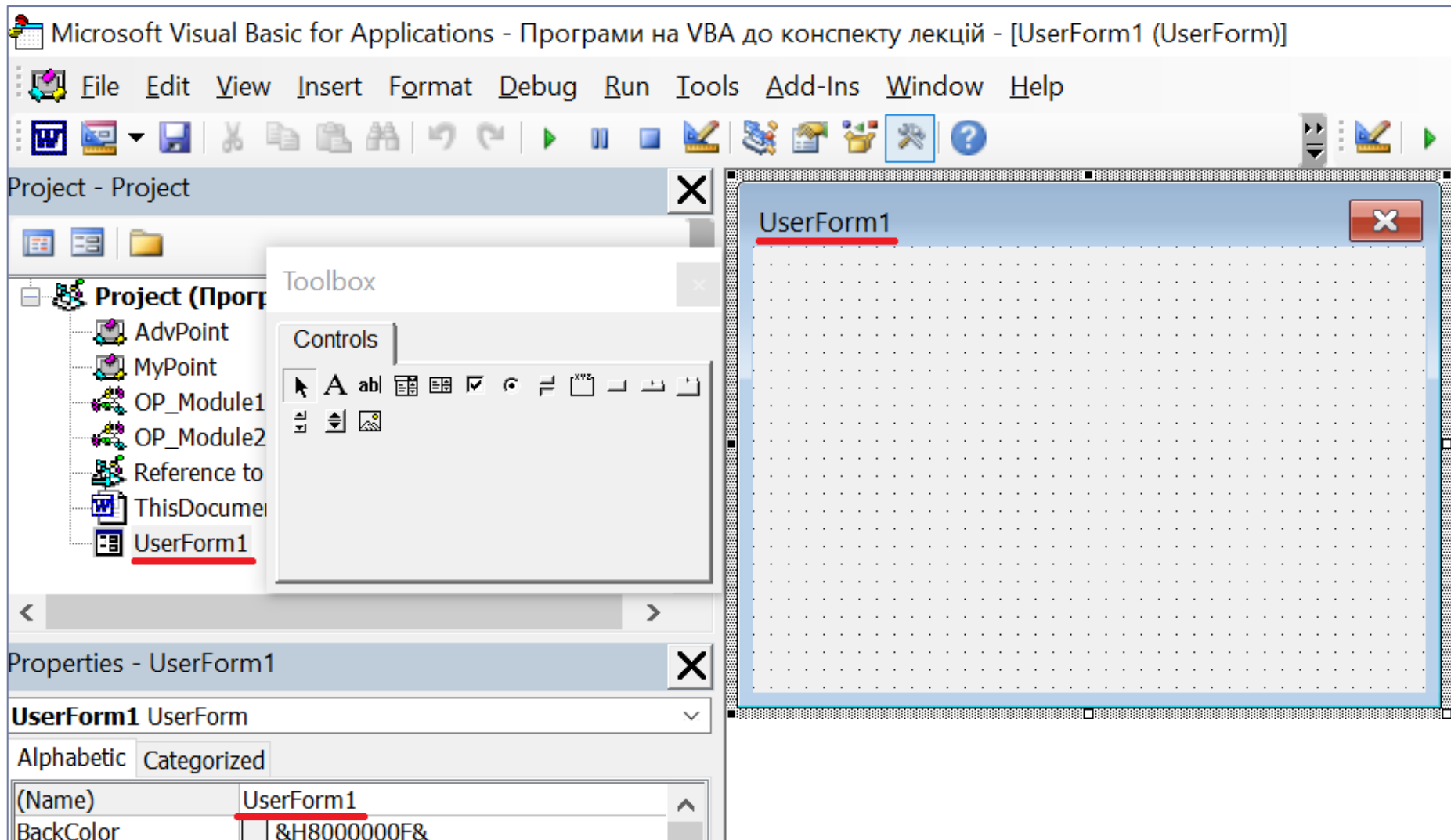
Ви ввели: 5

OK



Демонстрація вставки кнопки безпосередньо у документ

Створення форм



► **View -> Code** (клавіша <F7>), **View -> Object** (клавіша <Shift> + <F7>)

Основні властивості та методи форм

Властивості:

- ▶ `Name` - визначає ім'я форми, що використовується програмістом в програмному коді для цієї форми;
- ▶ `Caption` - визначає заголовок форми;
- ▶ `Enabled` - якщо встановлена в `False`, користувач працювати з формою не зможе;
- ▶ `ShowModal` - якщо встановлено в `True` (за замовчуванням), користувач не може перейти до інших форм або повернутися в документ, поки не зачне цю форму.

Більша частина основних властивостей відноситься до зовнішнього вигляду, розмірів і місцезнаходження вікна форми.

Методи:

- ▶ `UserForm1.Show` - запуск форми з якоїсь процедури (з макросів `AutoExec`, `AutoNew`, `AutoOpen`, `AutoClose`, `AutoExit`, кнопкою у документі);
- ▶ `UserForm1.Hide` – видалення форми з екрану, але вона залишиться в пам'яті;
- ▶ `Unload UserForm1` - видалення форми з пам'яті.

Решта методів відносяться або до обміну даними через буфер обміну (`Copy()`, `Cut()`, `Paste()`), або до службових можливостей форми

- ▶ (`PrintForm()`, `Repaint()`, `Scroll()`). Додати у документ кнопки, що запускають та ховають форму

Основні події форм

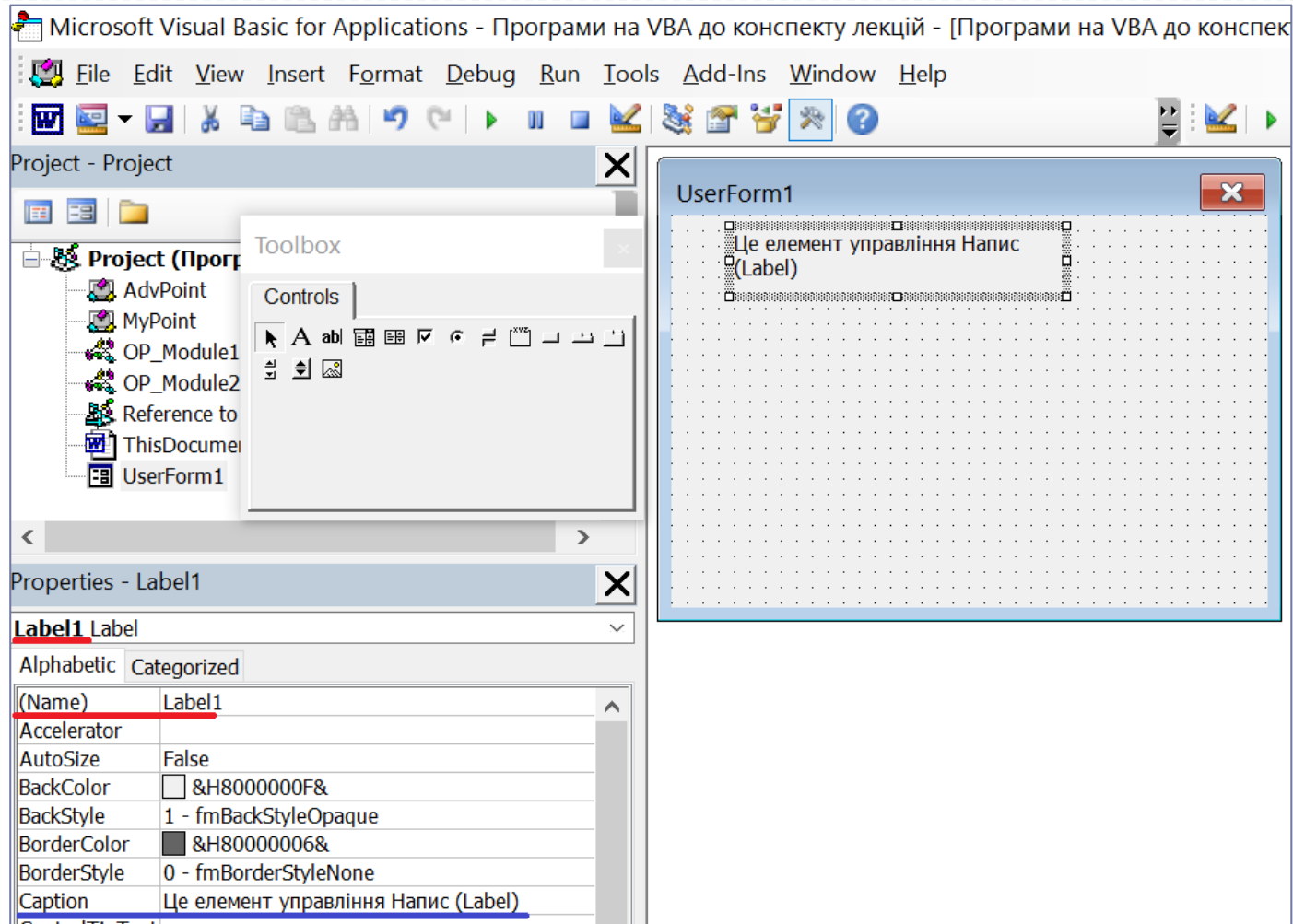
Події:

- ▶ `Initialize` - відбувається при підготовці форми до відкриття, зазвичай в обробнику цієї події розміщується код, пов'язаний з налаштуванням елементів управління на формі, присвоєння їм значень за замовчуванням і т.і.
- ▶ `Click` (ця подія вибирається за замовчуванням) і `DbClick` - реакція на одиночне та подвійне клацання мишею на формі, відповідно. Для форми ця подія використовується не так часто. Зазвичай обробник клацань використовується для кнопок (елементів управління `CommandButton`).
- ▶ `Error` - це подія, що виникає при помилці у формі, вона використовується для надання користувачеві можливості виправити зроблену ним помилку (або для інформування про помилку).
- ▶ `Terminate` - подія, що використовується при нормальному завершенні роботи форми і вивантаженні її з пам'яті, зазвичай використовується для розриву відкритих з'єднань з базами даних, звільнення ресурсів, протоколювання і т.і.

Інші події форми пов'язані або зі зміною розміру її вікна, або з натисканнями клавіш, або з активізацією (отриманням фокуса)/деактивізацією (втратою фокусу).

Демонстрація властивостей та подій `UserForm1`

Елементи управління форми – Label (напис)



```
Private Sub UserForm_Initialize()
```

```
Label1.Caption = "Це елемент управління Напис (Label)"
```

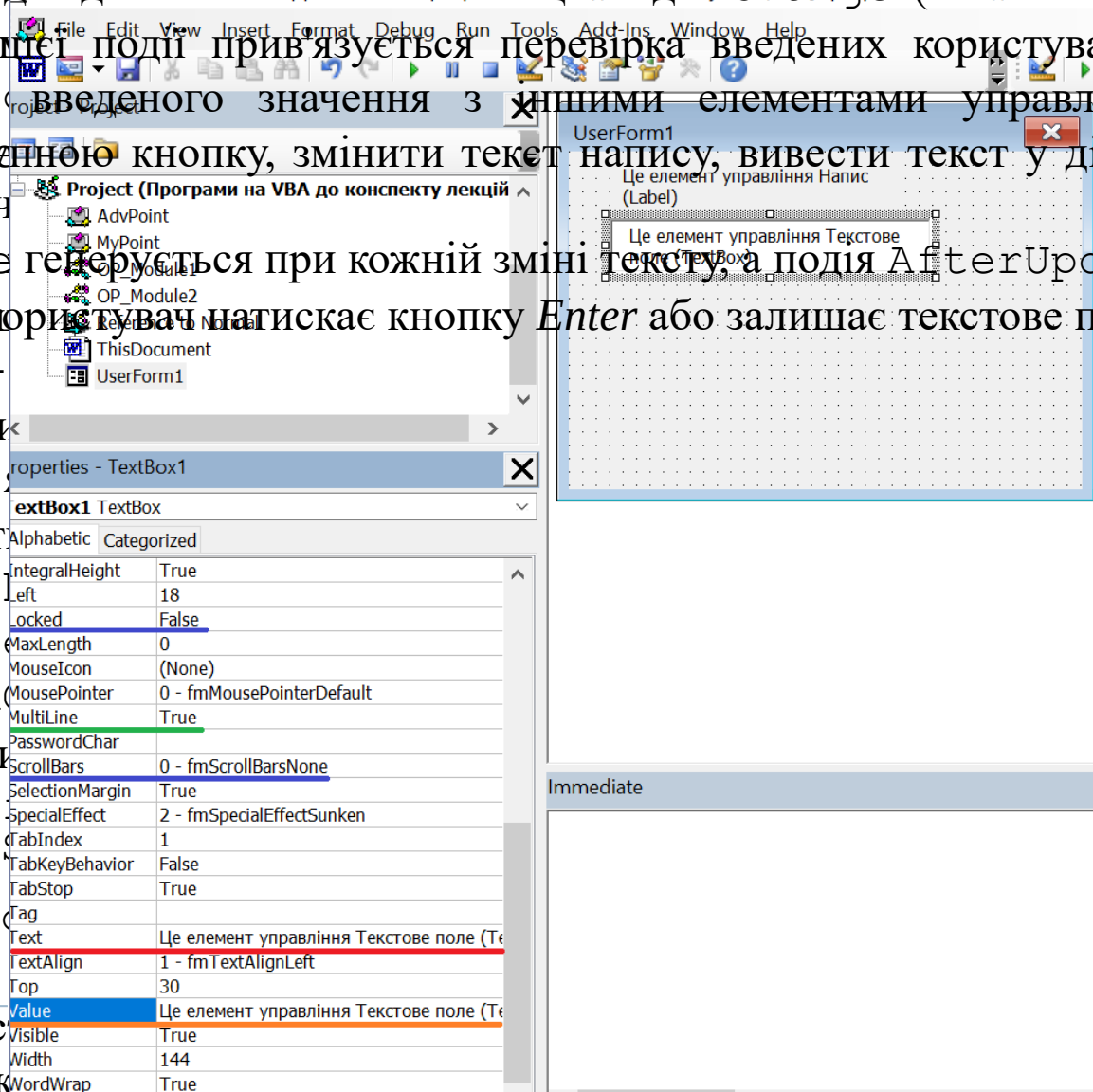
```
End Sub
```

Елементи управління форми – TextBox (текстове поле)

Міжовна (таблиця для текстового поля) — це подія Change (тобто змінний) сту текстове
Знаючи, яке має події прив'язується перевірка введених користувачем значень або
синхронізація введеного значення з іншими елементами управління (коміракБад,
туди Redo, сунною кнопку, змінити текст напису, вивести текст у діалоговому вікні
зміниться знач
Подія Change генерується при кожній зміні тексту, а подія AfterUpdate створює величину
зміна текст користувач натискає кнопку Enter або залишає текстове поле.

Enabled -
нічого зробити
Locked -
зможє виділяти
MaxLength
MultiLine
Password
вводяться кор
ScrollBar
Private
прокрутки.
WordWrap
поля
End Sub
Решта влас
змісту, а також

м і з вмістом поля
вичай, користувач
і кілька рядків.
тися" символи, що
ертикальна смуги
і. Максимального
гового поля і його



Property	Value
IntegralHeight	True
Left	18
Locked	False
MaxLength	0
MouseIcon	(None)
MousePointer	0 - fmMousePointerDefault
MultiLine	True
PasswordChar	PasswordChar
ScrollBars	0 - fmScrollBarsNone
SelectionMargin	True
SpecialEffect	2 - fmSpecialEffectSunken
TabIndex	1
TabKeyBehavior	False
TabStop	True
Tag	Tag
Text	Це елемент управління Текстове поле (Т
TextAlign	1 - fmTextAlignLeft
Top	30
Value	Це елемент управління Текстове поле (Т
Visible	True
Width	144
WordWrap	True

Елементи управління форми – ComboBox (комбінований список)

Використовується

- коли користувач вибирає значення зі списку
- коли список пов'язаний з джерелом даних

зі списку
ічно з даних

RowSource - джерело рядків для списку (наприклад, адреса діапазону на робочому листі Excel).

MatchEntry - визначає, чи будуть при введенні користувачем перших символів значення вибиратися відповідні позиції зі списку.

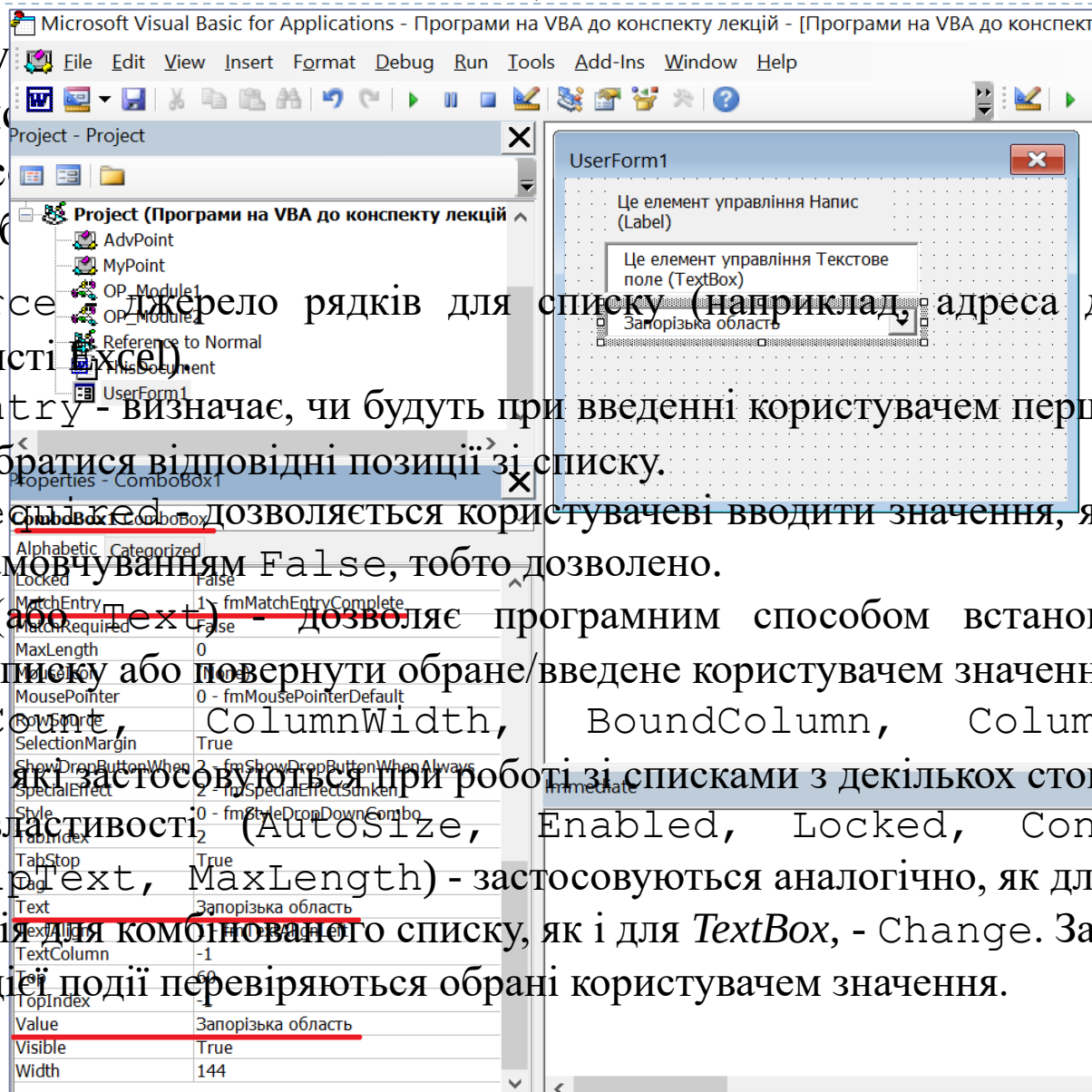
MatchRequired - дозволяється користувачеві вводити значення, якого немає в списку, за замовчуванням False, тобто дозволено.

Value (або Text) - дозволяє програмним способом встановити обране значення в списку або повернути обране/введене користувачем значення.

ColumnSource, ColumnWidth, BoundColumn, ColumnHeads - властивості, які застосовуються при роботі зі списками з декількох стовпців.

Решта властивості (**AutoSize, Enabled, Locked, ControlText, ControlTipText, MaxLength**) - застосовуються аналогічно, як для *TextBox*.

Головна подія для комбінованого списку, як і для *TextBox*, - **Change**. Зазвичай в обробнику цієї події перевіряються обрані користувачем значення.



Елементи управління форми – ComboBox (комбінований список)

Заповнення елементів списку у обробнику події ініціалізації форми:

```
Private Sub UserForm_Initialize()  
    ComboBox1.AddItem "Запорізька область"           'ListIndex = 0  
    ComboBox1.AddItem "Дніпропетровська область"    'ListIndex = 1  
    ComboBox1.AddItem "Київська область"             'ListIndex = 2  
    ComboBox1.AddItem "Львівська область"           'ListIndex = 3  
End Sub
```

Обробник події Change:

```
Private Sub ComboBox1_Change()  
    MsgBox "У комбінованому списку обрано: " _  
        & (ComboBox1.Value)  
End Sub
```



Елементи управління форми – ListBox (список)

Відміни від комбінованого списку:

- в ньому не можна відкрити список значень по спадаючій кнопці. Всі значення їх видно відразу в поле, аналогічному текстовому, і тому велика кількість позицій в ньому вмістити важко;
- користувач не може вводити свої значення - тільки вибирати з готових.
- користувач може вибирати не одне значення, як в *ComboBox*, а декілька

Зазвичай ListBox використовується:

- як проміжний засіб відображення введених/обраних користувачем через *ComboBox* значень (або будь-яких файлів);
- як засіб редагування бази даних (для видалення або отриманих з бази даних елементів, кнопки Видалити або Змінити).

Заповнення елементів

```
Private Sub User
```

```
    ListBox1.Add
```

```
    ListBox1.Add
```

```
    ListBox1.Add
```

```
    ListBox1.Add
```

```
End Sub
```

UserForm1

Це елемент управління Напис (Label)

Це елемент управління Текстове поле (TextBox)

Запорізька область

Запорізька область
Дніпропетровська область
Київська область
Львівська область

ListBox

```
'ListIndex = 0
```

```
'ListIndex = 1
```

```
'ListIndex = 2
```

```
'ListIndex = 3
```

Елементи управління форми – ListBox (список)

Основні властивості, методи і події у *ListBox* - ті ж, що і у *ComboBox*. Головна відмінність - те, що є властивість *MultiSelect*, яка дозволяє користувачеві вибирати кілька значень. За замовчуванням ця властивість відключена.

Заповнення елементів списку у обробнику події ініціалізації форми:

```
Private Sub ListBox1_Change()  
    Dim myMsg As String  
    Dim i As Long  
  
    For i = 0 To ListBox1.ListCount - 1  
        If ListBox1.Selected(i) Then  
            myMsg = myMsg & ListBox1.List(i) & " "  
        End If  
    Next i  
    MsgBox myMsg  
End Sub
```

Елементи управління CheckBox (прапорець) і ToggleButton (кнопка з фіксацією)

Прапорці (*CheckBox*) і кнопки з фіксацією (*ToggleButton*) використовуються для вибору не взаємовиключних варіантів (якщо цих варіантів небагато).

У *CheckBox* три головні властивості:

- *Caption* - напис поруч з прапорцем;
- *TriState* - як приймати тільки три значення: *True*, *False* та *Indeterminate*. Якщо *TriState* встановлена "сіра" кнопка, це означає, що вибір компонентів неможливий;
- *Value* - саме значення, яке встановлений, (*False* встановлено в *True*);
- Головна подія - *CheckedChanged*.

The screenshot shows a Windows form titled "UserForm1". Inside the form, there is a label "Це елемент управління Напис (Label)". Below the label is a text box containing "Це елемент управління Текстове поле (TextBox)". Below the text box is a dropdown menu with the text "Запорізька область". Below the dropdown menu is a list box containing the text "Запорізька область", "Дніпропетровська область", "Київська область", and "Львівська область". Below the list box are three checkboxes labeled "Red", "Green", and "Blue". The "Red" and "Blue" checkboxes are checked, while the "Green" checkbox is unchecked. Below the checkboxes are two buttons labeled "True" and "False". Red arrows point from the text "CheckBox" to the checkboxes and from "ToggleButton" to the "True" and "False" buttons.

визначається цим

прапорець може приймати три значення: *Null*, коли прапорець не встановлено, наприклад, при виборі компонентів, а лише

True (прапорець встановлено) та *False* (прапорець не встановлено). Якщо *TriState* встановлено, то прапорець може приймати три значення: *True*, *False* та *Indeterminate*.

Елементи управління CheckBox (прапорець) і ToggleButton (кнопка з фіксацією)

```
Private Sub CheckCheckBoxes()  
    Dim cBox As Control  
    Dim msg As String  
    For Each cBox In Me.Controls  
        If TypeName(cBox) = "CheckBox" Then  
            'CheckBox "Red" is in Tripple State  
            If IsNull(cBox) Then  
                Msg = "Null "  
            If cBox.Value = True Then  
                msg = msg & cBox.Caption & " "  
            End If  
        End If  
    Next  
    MsgBox msg  
End Sub  
  
Private Sub CheckBox1_Change()  
    Call CheckCheckBoxes  
End Sub
```

And 2 And 3



Елементи управління CheckBox (прапорець) і ToggleButton (кнопка з фіксацією)

ToggleButton виглядає як кнопка, яка при натисканні стає "натиснутою", а при повторному натисканні відключається. У неї можуть бути ті ж два (або три, відповідно до властивістю *TriState*) стану, що і у *CheckBox*. Властивості і методи - ті ж самі, як у *CheckBox*. Єдина відмінність - в сприйнятті їх користувачем. Зазвичай *ToggleButton* сприймається користувачем як перехід в якийсь режим або початок виконання тривалої дії.

```
Private Sub ToggleButton1_Click()  
    If ToggleButton1.Value = True Then  
        ToggleButton1.Caption = "State On"  
    Else  
        ToggleButton1.Caption = "State Off"  
    End If  
End Sub
```

CheckToggleButtons()



Елементи управління OptionButton (перемикач) і Frame (рамка)

Microsoft Visual Basic for Applications - Програми на VBA до конспекту лекцій - [Програми на VBA до конспекту лекцій]

File Edit View Insert Format Debug Run Tools Add-Ins Window Help

Project - Project

Project (Програми на VBA до конспекту лекцій)

- AdvPoint
- MyPoint
- OP_Module1
- OP_Module2
- Reference to Normal
- ThisDocument
- UserForm1

Properties - Frame1

Frame1 Frame

Alphabetic Categorized

(Name)	Frame1
BackColor	&H8000000F&
BorderColor	&H80000012&
BorderStyle	0 - fmBorderStyleNone
Caption	Рамка

UserForm1

Це елемент управління Напис (Label)

Це елемент управління Текстове поле (TextBox)

Запорізька область

☐ Red ☐ Green ☐ Blue

True False

Рамка

☐ Перемикач 1 ☐ Перемикач 2

Елементи управління OptionButton (перемикач) і Frame (рамка)

```
Private Sub OptionButtonsCheck()  
    Dim I As Integer  
    Dim Msg(0 To 1) As String  
  
    For I = 0 To Me.Frame1.Controls.Count - 1  
        Msg(I) = Me.Frame1.Controls(I).Name & " - " _  
                & Me.Frame1.Controls(I).Value  
    Next I  
    MsgBox Msg(0) & ", " & Msg(1)  
End Sub  
  
Private Sub OptionButton1_Click()  
    Call OptionButtonsCheck  
End Sub  
  
Private Sub OptionButton2_Click()  
    Call OptionButtonsCheck  
End Sub
```

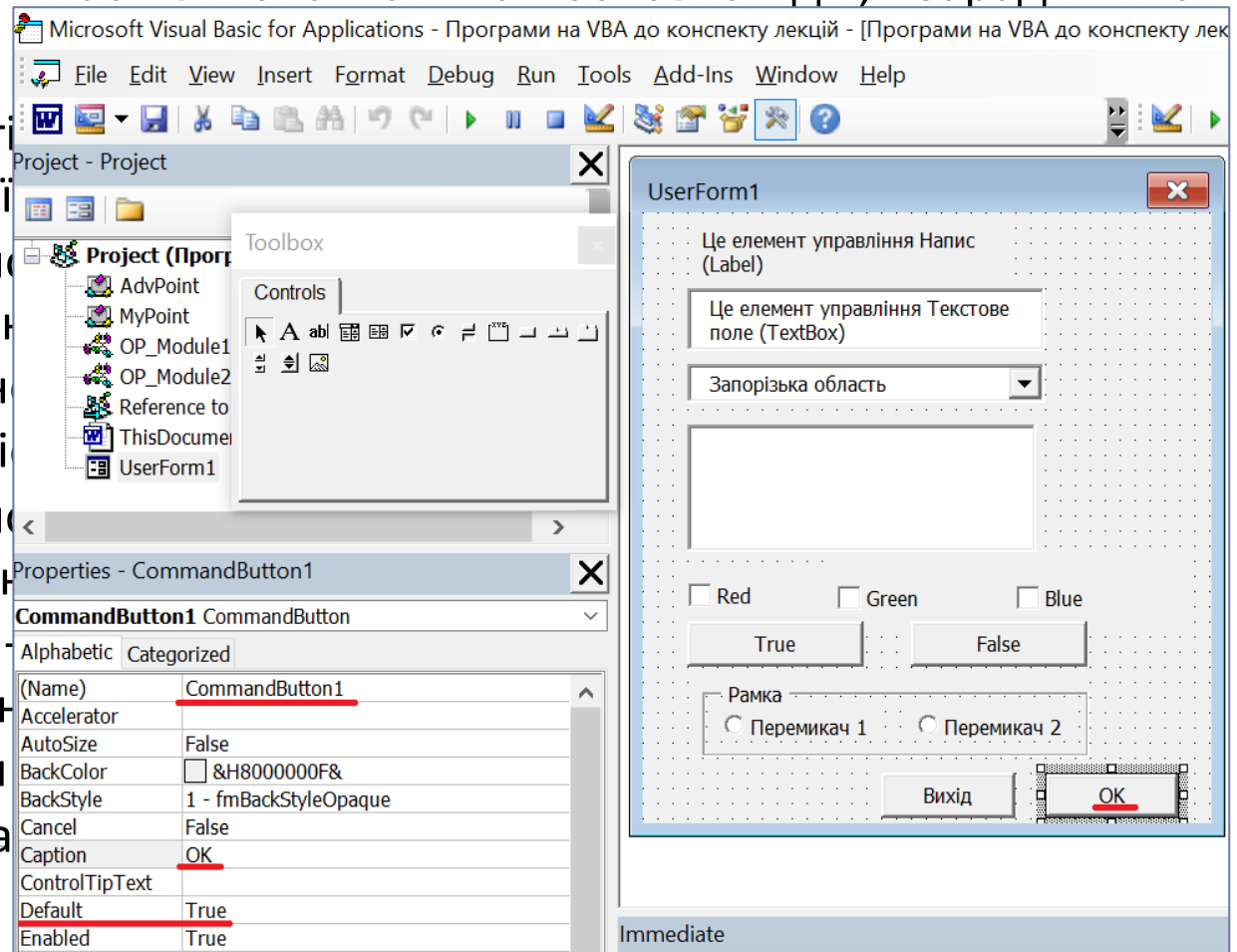


Елемент управління CommandButton (кнопка)

CommandButton (кнопка) - найпоширеніший елемент управління в формах. У більшості форм обов'язково буде принаймні дві кнопки: *Отмена* (*Cancel*) і *OK*. При натисканні кнопки *Отмена* форма повинна закритися, після натискання кнопки *OK* має виконатися та основна дія, заради якої створювалася ця форма.

Найважливіші властивості

- `Cancel` - якщо для цієї кнопки буде автоматично натиснуто, ще додати код в обробку
 - `Caption` - напис на кнопці
 - `Default` - якщо для цієї кнопки буде автоматично натиснуто, виконати основну подію
 - `Picture` - посилання на зображення
 - `TakeFocusOnClick` - взяти фокус на кнопку при натисканні
- Головна подія для кнопки - `OnClick`. У цій події можна виконати той програмний код, заради якого створювалася форма.



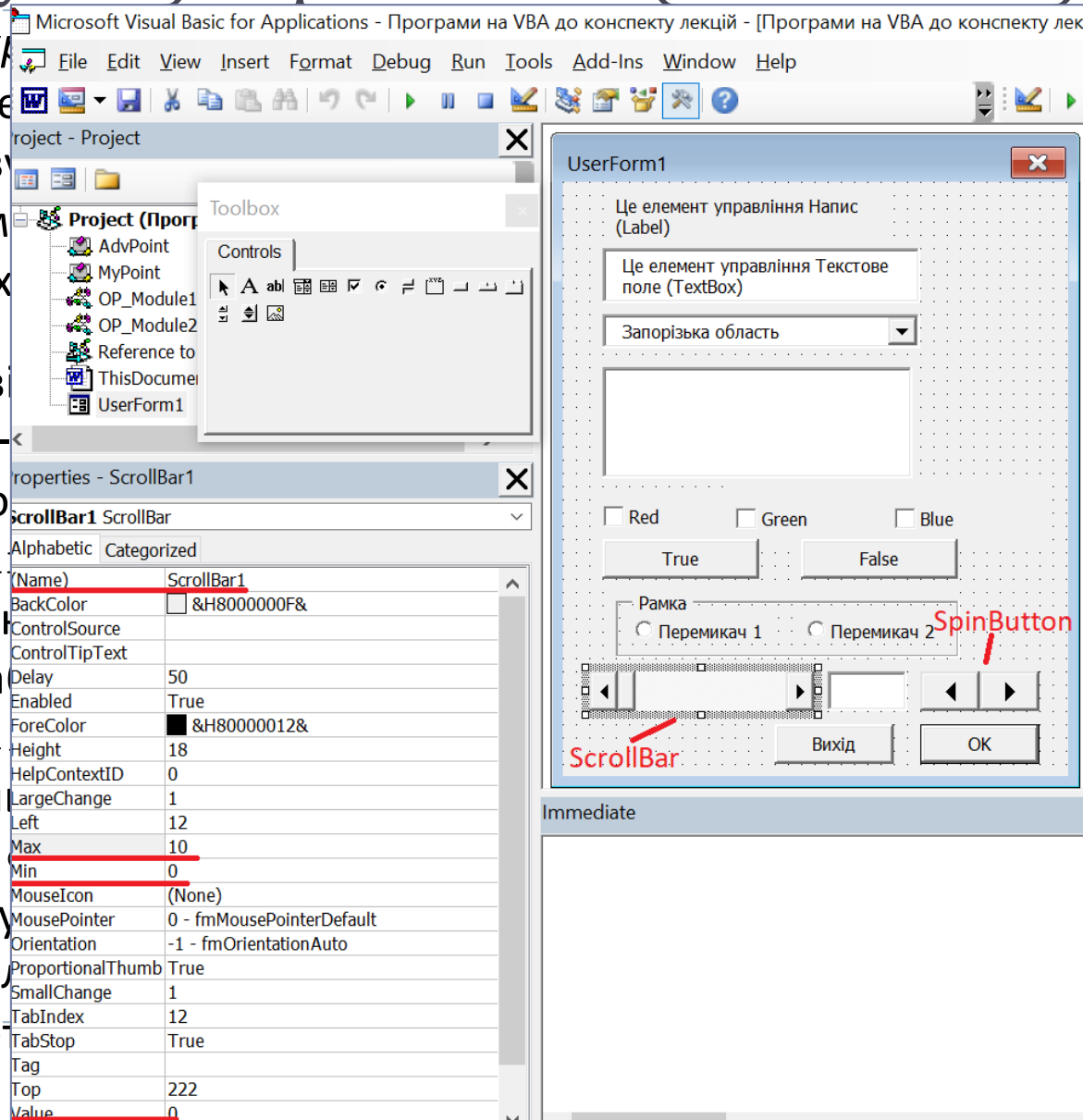
Елемент управління CommandButton (кнопка)

```
Private Sub OptionButtonsCheck()  
    Dim I As Integer  
    Dim Msg(0 To 1) As String  
  
    For I = 0 To Me.Frame1.Controls.Count - 1  
        Msg(I) = Me.Frame1.Controls(I).Name & " - " _  
                & Me.Frame1.Controls(I).Value  
    Next I  
    MsgBox Msg(0) & ", " & Msg(1)  
End Sub  
  
Private Sub OptionButton1_Click()  
    Call OptionButtonsCheck  
End Sub  
  
Private Sub OptionButton2_Click()  
    Call OptionButtonsCheck  
End Sub
```



Елементи управління ScrollBar (смуга прокрутки) і SpinButton (лічильник)

Смуги прокрутки введені та використовують "повзунок" змінюваних. Найважливіші Max і Min - допомогую LargeChange при переміщенні повзунка, а Orientation - горизонтальна. ProportionalThumb - розміру смуги Value - буде повертати значення, яке



вих полях, коли ливість о часто називають апазону плавно і. дати за ся повзунок при льній від). не чи н пропорційний розміру. те значення, яке

Елементи управління ScrollBar (смуга прокрутки) і SpinButton (лічильник)

Елемент управління *SpinButton* - це та ж смуга прокрутки, позбавлена самої смуги і повзунка. *SpinButton* використовується у тих ситуаціях, коли діапазон обраних значень зовсім невеликий (наприклад, треба вибрати кількість копій для друку). Всі властивості *SpinButton*, збігаються з властивостями *ScrollBar*.

```
Private Sub ScrollBar1_Change()  
    TextBox2.Value = ScrollBar1.Value  
    SpinButton1.Value = TextBox2.Value  
End Sub
```

```
Private Sub SpinButton1_Change()  
    TextBox2.Value = SpinButton1.Value  
    ScrollBar1.Value = TextBox2.Value  
End Sub
```



Елементи управління TabStrip (набір вкладок) і MultiPage (набір сторінок)

TabStripForm

Країна | Область | Адреса |

Індекс:

Країна:

- Україна
- Росія
- Молдова

- MultiRow - визначає, чи можна бути вкладок.

TabStripForm

Країна | Область | Адреса |

Вулиця, дім

Тип вулиці

- вулиця
- проспект
- провулок

уювати на формі кілька вкладок, які
ва відмінність між цими елементами
х *TabStrip* завжди розташовуються
цях *MultiPage* - різні.

ління практично ідентичні.

TabStripForm

Країна | Область | Адреса |

Місто

Область

- Запорізька область
- Дніпропетровська область
- Київська область
- Львівська область

емиканні між вкладками.

Елементи управління TabStrip (набір вкладок) і MultiPage (набір сторінок)

```
Private Sub UserForm_Initialize()  
    TabStrip1.Value = 0 'Встановлюється активною перша  
вкладка  
    ComboBox1.Clear  
    ComboBox1.AddItem "Україна"  
    ComboBox1.AddItem "Росія"  
    ComboBox1.AddItem "Молдова"  
End Sub
```

```
Private Sub TabStrip1_Change()  
    Select Case TabStrip1.SelectedItem.Caption  
        Case "Країна"  
            Label1.Caption = "Індекс"  
            Label2.Caption = "Країна"  
            ComboBox1.Clear  
            ComboBox1.AddItem "Україна"  
            ComboBox1.AddItem "Росія"  
            ComboBox1.AddItem "Молдова"
```



Елементи управління TabStrip (набір вкладок) і MultiPage (набір сторінок)

...

```
Case "Область"  
    Label1.Caption = "Місто"  
    Label2.Caption = "Область"  
    ComboBox1.Clear  
    ComboBox1.AddItem "Запорізька область"  
    ComboBox1.AddItem "Дніпропетровська область"  
    ComboBox1.AddItem "Київська область"  
    ComboBox1.AddItem "Львівська область"  
Case "Адреса"  
    Label1.Caption = "Вулиця, дім квартири"  
    Label2.Caption = "Тип вулиці"  
    ComboBox1.Clear  
    ComboBox1.AddItem "вулиця"  
    ComboBox1.AddItem "проспект"  
    ComboBox1.AddItem "провулок"  
End Select  
End Sub
```



Елементи управління TabStrip (набір вкладок) і MultiPage (набір сторінок)

У разі використання елементу управління MultiPage всі елементи управління на різних сторінках - різні

The screenshot shows a window titled "MultiPageForm" with a close button (X) in the top right corner. The window contains a TabStrip with two tabs: "Page1" (selected, indicated by a red underline) and "Page2". Below the tabs is a large container area. Inside this area, there is a "Frame1" containing three radio buttons labeled "First", "Second", and "Third". The "First" radio button is selected. Below "Frame1" is a "Label2" followed by a text input field. At the bottom right of the container area is a "CommandButton2".

The screenshot shows the same "MultiPageForm" window, but now "Page2" is selected (indicated by a red underline). The content of the container area has changed. It now contains an "Items:" label followed by a list of four Ukrainian regions: "Запорізка область", "Дніпропетровська область", "Київська область", and "Львівська область". At the bottom right of the container area is a "CommandButton1".



Елемент управління Image (рисунок)

Елемент управління *Image* дозволяє відобразити на формі рисунок в одному з поширених форматів, який буде реагувати на клацання мишею (а може просто використовуватися для прикраси форми) .

Деякі зауваження щодо використання *Image*:

- в якості альтернативи можна використовувати властивість *Picture* для форми (особливо якщо вам потрібен фоновий малюнок для всієї форми);
- ще дві альтернативи - застосування властивості *Picture* для елементів управління *Label* або *CommandButton*.

Основні властивості:

- *Picture* - дозволяє вибрати саме зображення для форми;
- *PictureAlignment* - дозволяє вибрати вирівнювання зображення у відведеної йому області. За замовчуванням - по центру;
- *PictureSizeMode* - дозволяє вибрати режим розтягування/зменшення елемента в разі, якщо він не точно відповідає розміру області;
- *PictureTiling* - дозволяє розмножувати маленький рисунок так, щоб він покрити все відведену йому область.

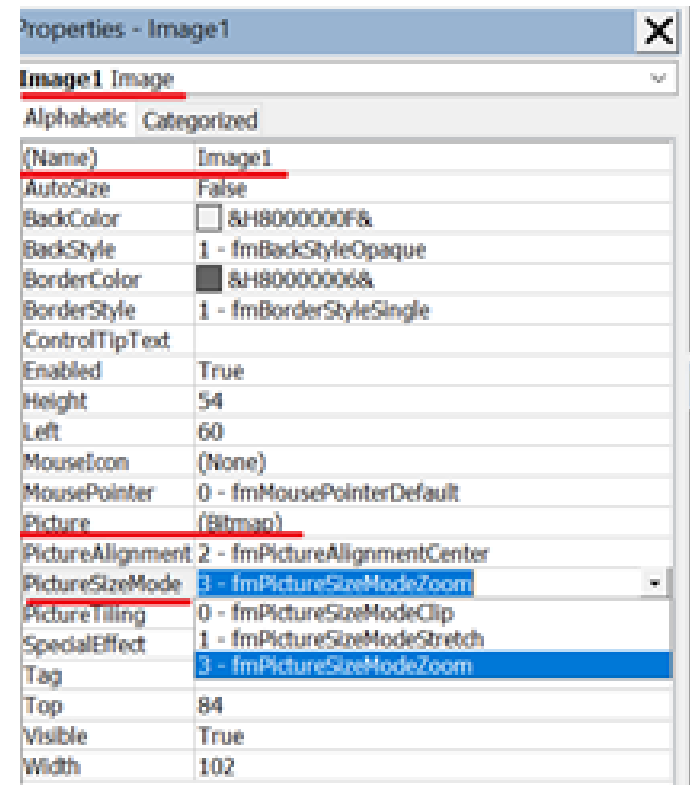
Головна подія елемента управління *Image* - подія *Click*.



Елемент управління Image (рисунок)

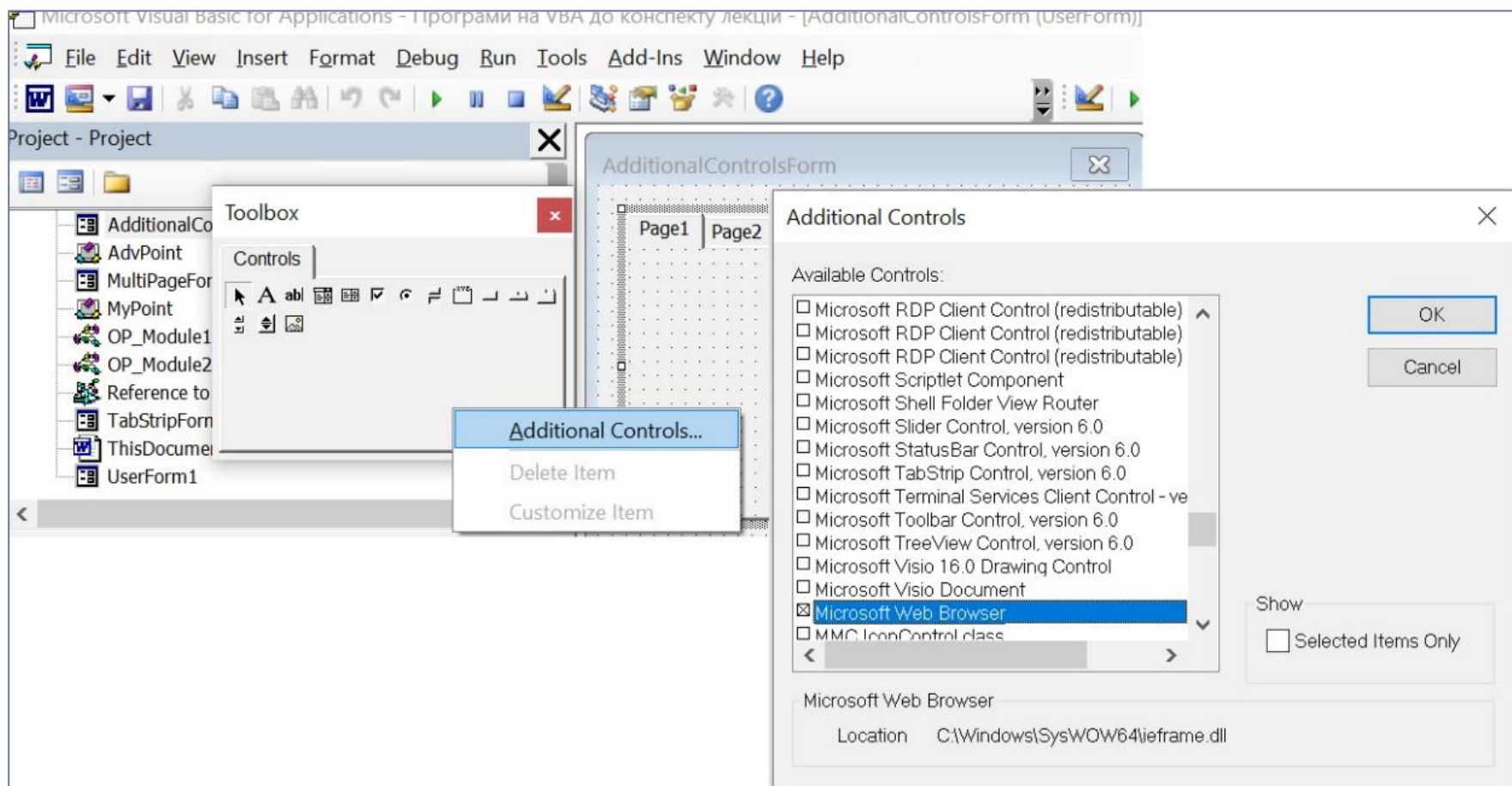
Для динамічного завантаження рисунків використовують метод `LoadPicture`, у якому вказують повний шлях до файлу рисунку, наприклад:

```
Image1.Picture =  
LoadPicture("D:\Documents\images\countries.jpg")
```

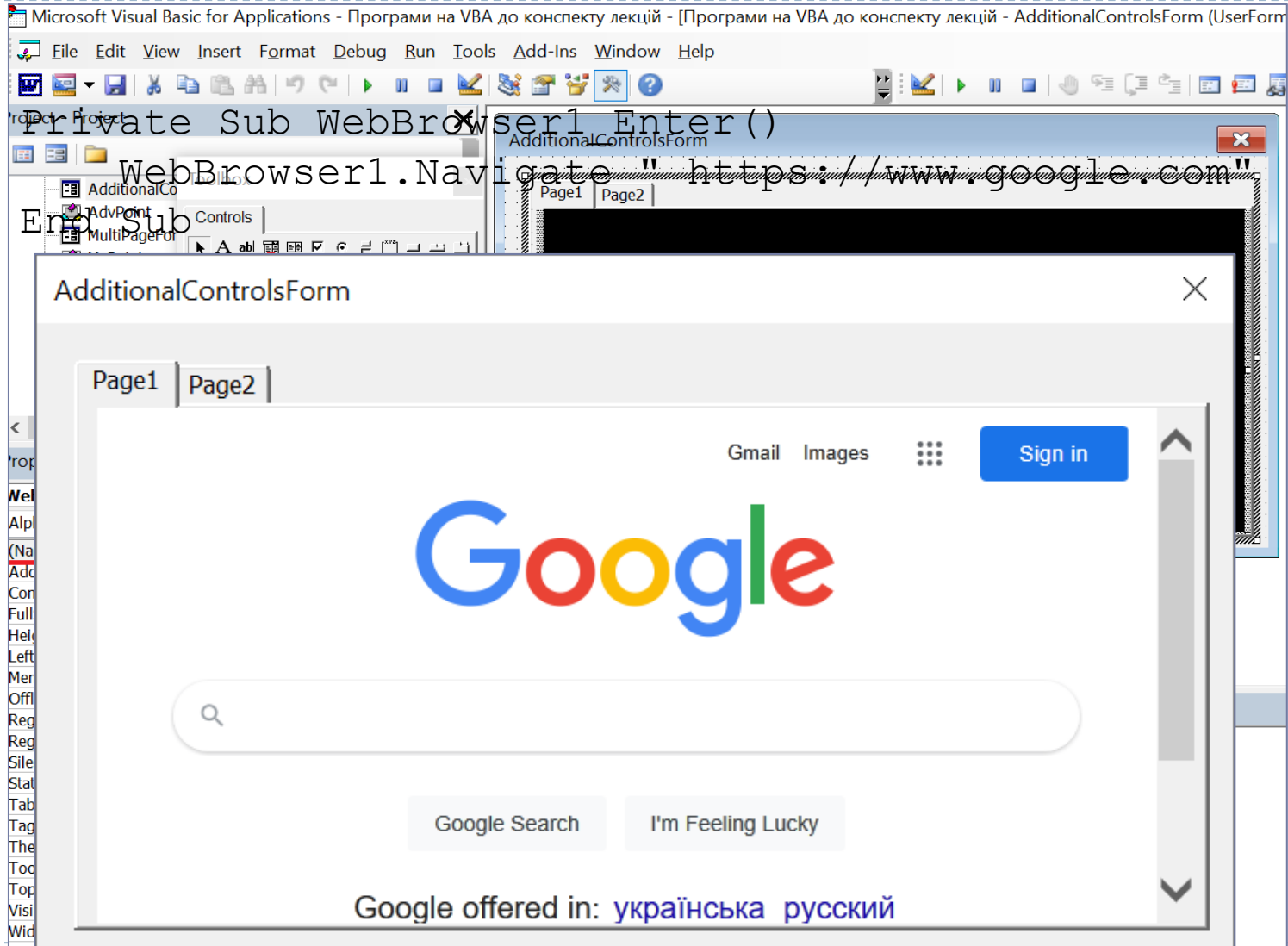


Додаткові елементи управління

Для розміщення на формі додаткових елементів управління, клацніть правою кнопкою миші по порожньому простору в *Toolbox* і виберіть пункт *Additional Controls* (або командою головного меню *Tools- Additional Controls*) - а далі оберіть потрібний елемент



Додатковий елемент управління - Microsoft Web Browser



Додаткові елементи управління - *Microsoft Date and Time Picker Control* та *Microsoft MonthView Control*

1. Скачати бібліотеку https://www.ocxme.com/files/mscomct2_ocx.
2. Розпакувати архів та скопіювати файл MSCOMCT2.OCX у каталог Windows\SysWOW64 (для 64-бітних версій Windows) або у каталог Windows\System32 (для 32-бітних версій Windows).
3. Виконати у консолі з правами адміністратора з каталогу з бібліотекою команду:
`regsvr32 MSCOMCT2.OCX`

Увікні додаткових елементів управління панелі *Toolbox* з'являться:

- *Microsoft Animation Control* - анімація виконання будь-якого процесу (копіювання/переміщення файлу з однієї папки в іншу, пошук та інше)
 - *Microsoft UpDown Control* - покращений лічильник з додатковими можливостями;
 - ***Microsoft MonthView Control*** - вдалий календарик "в чистому вигляді";
 - ***Microsoft Date and Time Picker Control*** - цей же календарик MonthView, але у вигляді комбінованого списку, крім дати можна працювати і з часом;
 - *Microsoft Flat ScrollBar Control* - смуга прокрутки.
-



Додаткові елементи управління - *Microsoft Date and Time Picker Control* та *Microsoft MonthView Control*

Головна властивість елементів управління *Microsoft Date and Time Picker Control* та *Microsoft MonthView Control* - властивість `Value`, тобто та дата,

яка об
відобр

AdditionalControlsForm

Page1 Page2

01.02.2021

01.02.2021

Февраль 2021

Пн	Вт	Ср	Чт	Пт	Сб	Вс
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
1	2	3	4	5	6	7

Today: 27.02.2021

DTPicker

Февраль 2021

Пн	Вт	Ср	Чт	Пт	Сб	Вс
25	26	27	28	29	30	31
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
1	2	3	4	5	6	7

Today: 27.02.2021

MonthView

S

End Sub

Типи помилок

- **синтаксичні** (неправильно написаний оператор, ім'я змінної і т.і.) - "відловлюються" редактором коду VBA ще в процесі написання коду або в ході компіляції і запуску програми. При цьому компілятор VBA видає інформацію, в якому рядку коду виявлена помилка, і в чому вона полягає.
- **логічні** - в ході виконання програма веде себе не так, як планувалося. Як правило, для виявлення та виправлення помилок такого типу і призначені прийоми **налагодження**.
- **помилки часу виконання (run-time error)** - це ситуація, коли в процесі виконання програма зіткнулася з проблемою, вирішити яку вона не в змозі (файл з таким ім'ям вже існує, виник конфлікт записів при вставці в базу даних, зроблено спробу записати інформацію на переповнений диск і т.і.). Необхідно передбачити можливість виникнення помилок часу виконання і забезпечити їх **перехоплення і обробку**.



Перехід в режим паузи

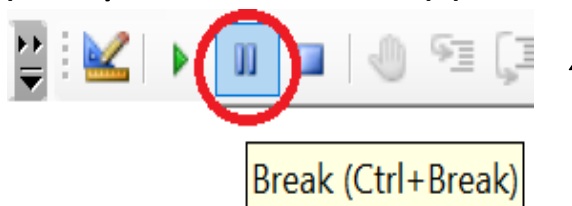
ListBox1		Change
	<pre>Private Sub TextBox1_AfterUpdate() MsgBox "У текстове поле введено: " & (TextBox1.Value) End Sub</pre>	
	<pre>Private Sub ComboBox1_Change() MsgBox "У комбінованому списку обрано: " & (ComboBox1.Value) End Sub</pre>	
		<pre>Private Sub ListBox1_Change() Dim myMsg As String Dim I As Long For I = 0 To ListBox1.ListCount - 1 If ListBox1.Selected(I) Then myMsg = myMsg & ListBox1.List(I) & " " End If Next I Debug.Print myMsg End Sub</pre>

Точка зупину



Перехід в режим паузи

- На жаль, точки зупину не зберігаються після закриття документа. Якщо потрібно запам'ятати місце зупинки між сеансами налагодження, то потрібно просто надрукувати в цей місце рядок з єдиною командою `Stop`. Програма в ході виконання автоматично зупиниться на цьому рядку.
- Якщо програма не хоче завершуватися (наприклад, у вас виконується нескінченний цикл), можна в ході її виконання натиснути на кнопку *Break*



або скористатися командою *Break* з меню *Run*, або просто натиснути на клавіші `<Ctrl> + <Break>`.

Дії в режимі паузи

В режимі паузи можливо:

- Продовжити виконання в покроковому режимі. Для цього можна скористатися кнопкою  панелі інструментів *Debug*, виконати команду *Debug -> Step Into* або натиснути на клавішу <F8>;
- Якщо викликається процедура, яка вже налагоджена, існує можливість виконати її без зупинок і перейти до наступного оператора. Для цієї мети використовується кнопка  панелі інструментів *Debug*, команда *Debug -> Step Over* або клавіатурна комбінація Shift + <F8>;
- Якщо після заходу у процедуру необхідно виконати її до кінця, використовуйте кнопку  панелі інструментів *Debug*, команду *Debug -> Step Out* (або клавіатурну комбінацію <Ctrl> + <Shift> + <F8>);
- Якщо необхідно виконати код не покроково, а ділянками до курсору, можна натиснути правою кнопкою миші по потрібній ділянці коду і в контекстному меню вибрати *Run to Cursor* (альтернатива - скористатися тією ж командою в меню *Debug* або натиснути <Ctrl> + <F8>).;



Дії в режимі паузи

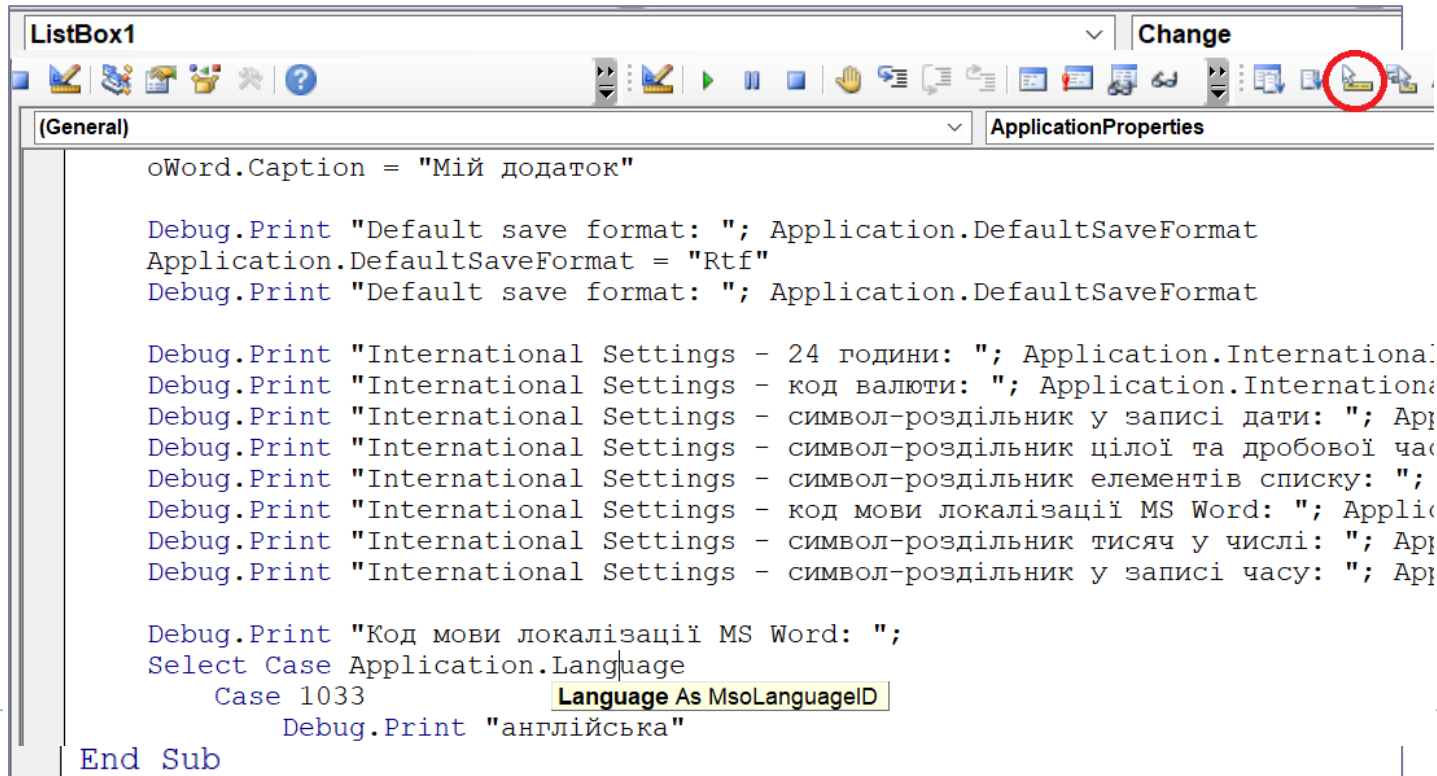
В режимі паузи можливо:

- Якщо необхідно "перестрибнути" через якусь ділянку коду, що викликає помилку, можна скористатися командою меню *Debug -> Set Next Statement* (або натиснути <Ctrl> + <F9>), а потім перетягнути жовту позначку по лівій межі вниз (або вгору). Альтернативний спосіб пропуску команд - виділити непотрібний блок і за допомогою панелі інструментів Edit його коментувати кнопкою  (але тоді потім доведеться знімати коментарі кнопкою );
- Якщо в процесі перегляду коду Ви пішли досить далеко і необхідно повернутися до місця зупинки без довгих розшуків, виконайте команду *Show Next Statement*;
- Щоб продовжити виконання після зупинки, можна натиснути на клавішу <F5> або кнопку  або скористатися командою *Continue* (вона з'явиться замість команди *Run*) в меню *Run*;
- Припинити виконання програми можна за допомогою команди *Reset* в тому ж меню або кнопкою  або клавішами <Alt> + <F4>.

Дії в режимі паузи

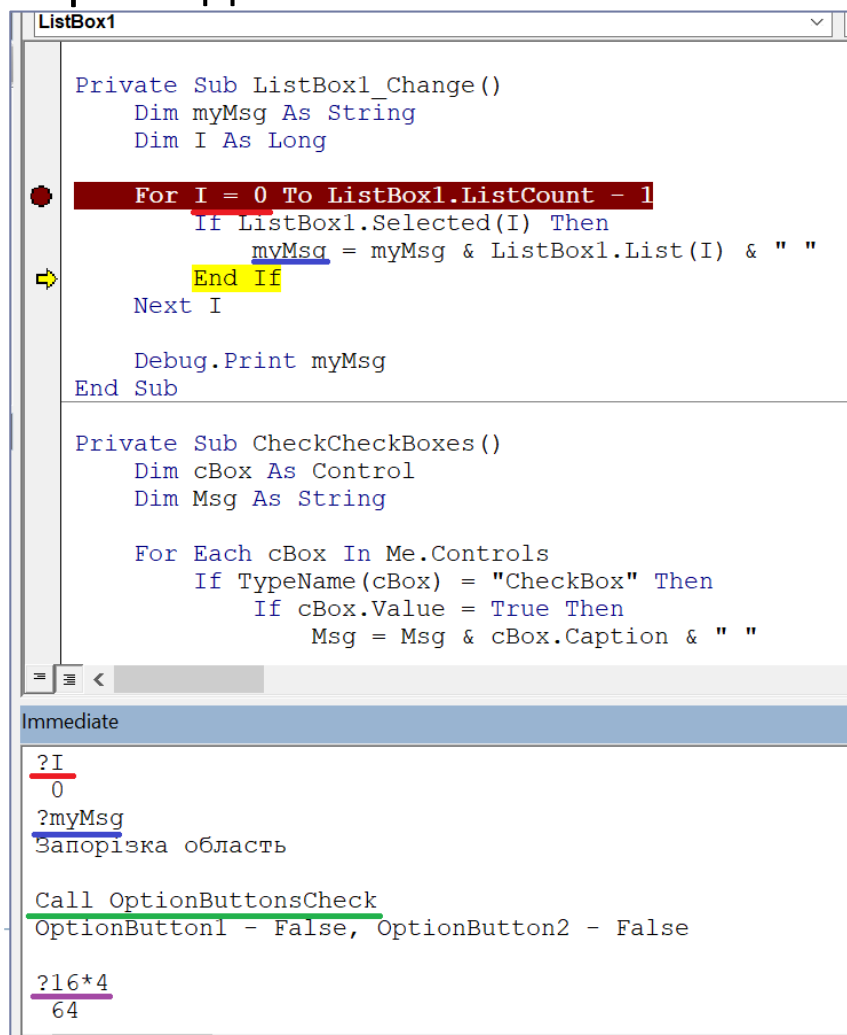
В режимі паузи можливо:

- Щоб отримати інформацію про об'єкт змінної, досить зайти в меню **Debug** (якщо великий курсор миші встановити на об'єкті змінної) та вибрати **Watch** (можливо з **Options** на панелі інструментів **Edit** або клавіатурну комбінацію **<Ctrl> + <I>**).



Вікно Immediate

У вікні Immediate (викликати його можна через меню View або <Ctrl> + <G>) можна переглядати або змінювати значення змінних і властивостей об'єкта.



The screenshot displays the Visual Studio IDE with a code file named 'ListBox1.vb' and the 'Immediate' window open at the bottom. The code file contains two private subroutines. The first, 'ListBox1_Change()', iterates through the 'ListBox1' control, building a string 'myMsg' of selected items. The second, 'CheckCheckBoxes()', iterates through all controls on the form, checking for checkboxes and building a string 'Msg' of their captions. The 'Immediate' window shows the current state of variables: 'I' is 0, 'myMsg' is an empty string, and the memory address '16*4' is 64. The 'Call OptionButtonsCheck' statement is also visible in the Immediate window.

```
Private Sub ListBox1_Change()  
    Dim myMsg As String  
    Dim I As Long  
  
    For I = 0 To ListBox1.ListCount - 1  
        If ListBox1.Selected(I) Then  
            myMsg = myMsg & ListBox1.List(I) & " "  
        End If  
    Next I  
  
    Debug.Print myMsg  
End Sub  
  
Private Sub CheckCheckBoxes()  
    Dim cBox As Control  
    Dim Msg As String  
  
    For Each cBox In Me.Controls  
        If TypeName(cBox) = "CheckBox" Then  
            If cBox.Value = True Then  
                Msg = Msg & cBox.Caption & " "  
            End If  
        End If  
    Next cBox  
  
    Call OptionButtonsCheck  
End Sub
```

Immediate

```
?I  
0  
?myMsg  
Запорізка область  
  
Call OptionButtonsCheck  
OptionButton1 - False, OptionButton2 - False  
  
?16*4  
64
```


Вікно Locals

The screenshot shows a Visual Basic IDE with the following components:

- Code Editor:** Contains the following code:

```
End If
End Sub

Private Sub OptionButtonsCheck()
    Dim I As Integer
    Dim Msg(0 To 1) As String

    For I = 0 To Me.Frame1.Controls.Count - 1
        Msg(I) = Me.Frame1.Controls(I).Name & " - " & Me.Frame1.Controls(I).Value
    Next I
    Debug.Print Msg(0); ", "; Msg(1)
End Sub

Private Sub OptionButton1_Click()
    Call OptionButtonsCheck
End Sub

Private Sub OptionButton2_Click()
    Call OptionButtonsCheck
End Sub

Private Sub CommandButton1_Click()
```
- Locals Window:** Displays the state of local variables for the current function call, `Project.UserForm1.OptionButtonsCheck`.

Expression	Value	Type
Me		UserForm1/UserForm1
I	1	Integer
Msg		String(0 to 1)
Msg(0)	"OptionButton1 - True"	String
Msg(1)	"OptionButton2 - False"	String
- Immediate Window:** Shows the output of the `Debug.Print` statement: `, OptionButton2 - False` and `OptionButton1 - True, OptionButton2 - False`.

Вікно Watches

- Вікно Watches дозволяє виконувати "спостереження" відповідно до заданих Вами умов. При роботі з ним спочатку потрібно визначити контрольоване значення, яке називається *Контрольованим виразом (Watch expression)*.

Створити контрольований вираз можна так:

- клацнути по змінній/властивості/виразу у вікні редактора коду правою кнопкою миші і в контекстному меню вибрати команду *Add Watch*;
 - скористатися командою *Add Watch* меню *Debug*;
 - скористатися командою *Quick Watch* меню *Debug*, в цьому випадку у вікно *Watch* буде поміщено вираз, на якому знаходиться курсор. Як умова для "спрацьовування" в нього буде поміщено поточне значення змінної/властивості;
 - перетягнути вираз з коду у вікно *Watches* (його можна, як і всі інші, відкрити командою *View* головного меню).
-



Вікно Watches

(General)

```
Private Sub CheckBox3_Change()  
    Call CheckCheckBoxes  
End Sub  
  
Private Sub ToggleButton1_Click()  
    If ToggleButton1.Value = True Then  
        ToggleButton1.Caption = "Sta  
    Else  
        ToggleButton1.Caption = "Sta  
    End If  
End Sub  
  
Private Sub OptionButtonsCheck()  
    Dim I As Integer  
    Dim Msg(0 To 1) As String  
  
    For I = 0 To Me.Frame1.Controls.Count - 1  
        Msg(I) = Me.Frame1.Controls(I).Name & " - " & Me.Frame1.Controls(I).Value  
    Next I  
    Debug.Print Msg(0); ", "; Msg(1)  
End Sub
```

Add Watch

Expression:
Me.Frame1.Controls(I).Name

Context
Procedure: OptionButtonsCheck
Module: UserForm1
Project: Project

Watch Type
☒ Watch Expression
☐ Break When Value Is True
☐ Break When Value Changes

OK
Cancel
Help

Immediate

Watches

Expression	Value	Type	Context
Me.Frame1.Controls(I).Name	<Out of context>	Variant/Empty	UserForm1.OptionButtonsCheck

Перехоплення і обробка помилок часу виконання

Перед небезпечним кодом (збереження/відкриття файлу, можливість поділу на нуль, можливість введення помилкових даних і т.і.)

поміщається команда

`On Error GoTo` позначка_обробника_помилки,
а далі в кодї програми розміщується позначка обробника помилки і програмний код обробки:

```
Private Sub UserForm_Click()
```

```
    Dim a As Integer, b As Integer, c As Integer
```

```
    On Error GoTo ErrorHandlerDivision
```

```
        c = a / b
```

```
Exit Sub
```

UserForm2

```
ErrorHandlerDivision:
```

```
    MsgBox "Ділення на нуль"
```

```
End Sub
```

Щоб код обробника помилки не виконувався якщо помилки не було, є сенс поставити перед міткою обробника команду `Exit Sub` (якщо це підпроцедура) або `Exit Function` (якщо це - функція).

Перехоплення і обробка помилок часу виконання

Після виконання коду обробника помилки необхідно або продовжити виконання тієї процедури, в якій виникла помилка, або припинити її виконання і передати управління процедурі, яка її викликала.

Існують три можливості:

1. Ще раз виконати оператор, який викликав помилку (якщо обробник помилки вирішує виниклі проблеми). Для цього в обробник помилок необхідно вставити команду `Resume`;
2. Пропустити оператор, який викликав помилку. Для цього в обробник помилок необхідно вставити команду `Resume Next`;
3. Продовжити виконання з певного місця в програмі. Для цього в обробник помилок необхідно вставити команду `Resume` позначка Синтаксис роботи з позначкою - такий же, як в `GoTo`.



Перехоплення і обробка помилок часу виконання

Зверніть увагу на кілька моментів, які пов'язані з обробкою помилок:

- Щоб повернутися в нормальний режим роботи після проходження небезпечної ділянки коду (скасувати обробку помилок) можна скористатися командою `On Error GoTo 0`
- команда `On Error Resume Next` наказує компілятору ігнорувати всі виникаючі помилки і переходити до виконання наступного оператора. На практиці дуже часто перед виконанням небезпечного оператора дається `On Error Resume Next`, а потім за допомогою конструкції `Select ... Case` перевіряється номер помилки (через властивості об'єкта `Err`) і в залежності від цього організовується подальше виконання програми.



Перехоплення і обробка помилок часу виконання

При виникненні помилки автоматично створюється об'єкт Err. У цього об'єкта - дві головних властивості і два методи:

`Number` - ця властивість містить номер помилки, зазвичай він перевіряється в обробнику помилок, щоб з'ясувати, яка саме помилка виникла. Якщо номер помилки дорівнює 0, то помилок не було.

`Description` - текстовий опис помилки, за замовчуванням відображається користувачеві у діалоговому вікні.

`Err.Clear` - цей метод очищає всі властивості об'єкта `Err`.

`Err.Raise` - цей метод дозволяє згенерувати програмно помилку у програмі.



Перехоплення і обробка помилок часу виконання

Синтаксис методу:

```
Err.Raise number [, source] [, description]  
           [, helpfile] [, helpcontext]
```

Єдиний обов'язковий параметр `number` передає номер помилки (для користувацьких помилок зарезервовані номери 513–65535). При встановленні параметра `number` необхідно додати користувацький номер коду помилки до константи `vbObjectError`. Наприклад, щоб згенерувати номер помилки 513, призначте параметру `number` значення `vbObjectError + 513`.

Параметр `source` - рядок-назва об'єкту або програми, що породили помилку.

Параметр `description` - рядок-опис помилки

Параметри `helpfile` та `helpcontext` описують шлях до файлу довідки та розділ в ньому, відповідно.



Перехоплення і обробка помилок часу виконання

Генерування помилки може виконуватись у довільній частині процедури/функції після оператора

On Error GoTo позначка_обробника_помилки
викликом оператора, наприклад:

```
On Error GoTo ErrHandler
    Err.Raise vbObjectError + 513, "Calculator", _
        "Programmatically raised Error!"
ErrHandler:
MsgBox " Programmatically raised Error with #" & _
    & Err.Number, vbCritical, "Error"
```

UserForm2

