Офісне програмування

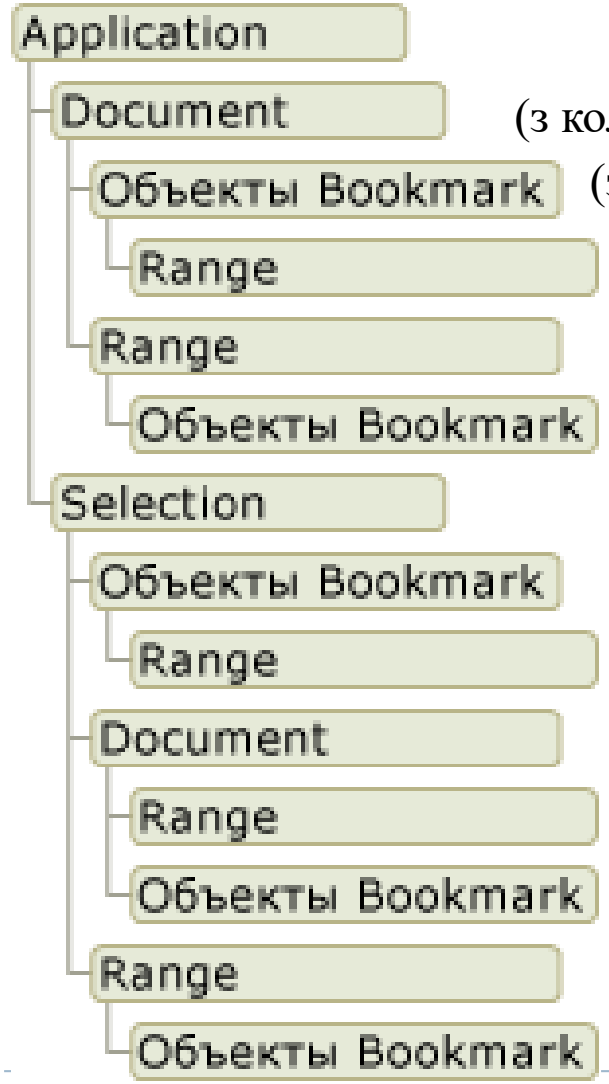
Об’єктна модель та програмування в Microsoft Word

Слайди до лекцій
(3 частина)

Приклади програм в Word

- Автоматизація повторюваних дій користувача при створенні однакових за формою документів Word.
- Автоматизація повторюваних дій користувача при вставці об'єктів у документ Word та його форматуванні.
- Створення в Word шаблонів звітів та завантаження у них даних з СУБД.
- Обмін даними між документом Word та файлів інших застосунків Microsoft Office.
- …

Об'єктна модель MS Word



(з колекцією `Documents`)

(з колекцією `Bookmarks`)

Об’єктна модель Word містить більше 300 класів

https://docs.microsoft.com/en-us/previous-versions/office/developer/office-xp/aa165321(v=office.10)

Об'єкт Application

- Об'єкт `Application` - це сам додаток Microsoft Word. Створити цей об'єкт - означає запустити Word на комп'ютері.

```
Dim oWord As New Word.Application
```

- за замовчуванням Word запускається в прихованому вікні;
- якщо в ньому не відкритих документів, він тут же і закривається (після того, як завершується процедура, у якій виконаний наведений вище оператор)

```
Dim oWord As New Word.Application
oWord.Visible = True
oWord.Documents.Add
```

Sub StartWord()

Об'єкт Application

Якщо Word вже відкритий, то можна отримати на нього посилання:

```
Set oWord = GetObject(, "Word.Application")
```

- функція `GetObject([pathname], [class])` повертає посилання на об’єкт, вказаний аргументом `class` у форматі `appname.objecttype`, де `appname` - назва застосунку, який надає об’єкт, `objecttype` - тип або клас об’єкту, на який функція повертає посилання.
- Аргумент `pathname` містить повний шлях та ім’я файлу, який містить об’єкт, якщо цей аргумент не вказується, аргумент `class` є обов’язковим.

Sub GetOpenDocuments()

Об'єкт Application

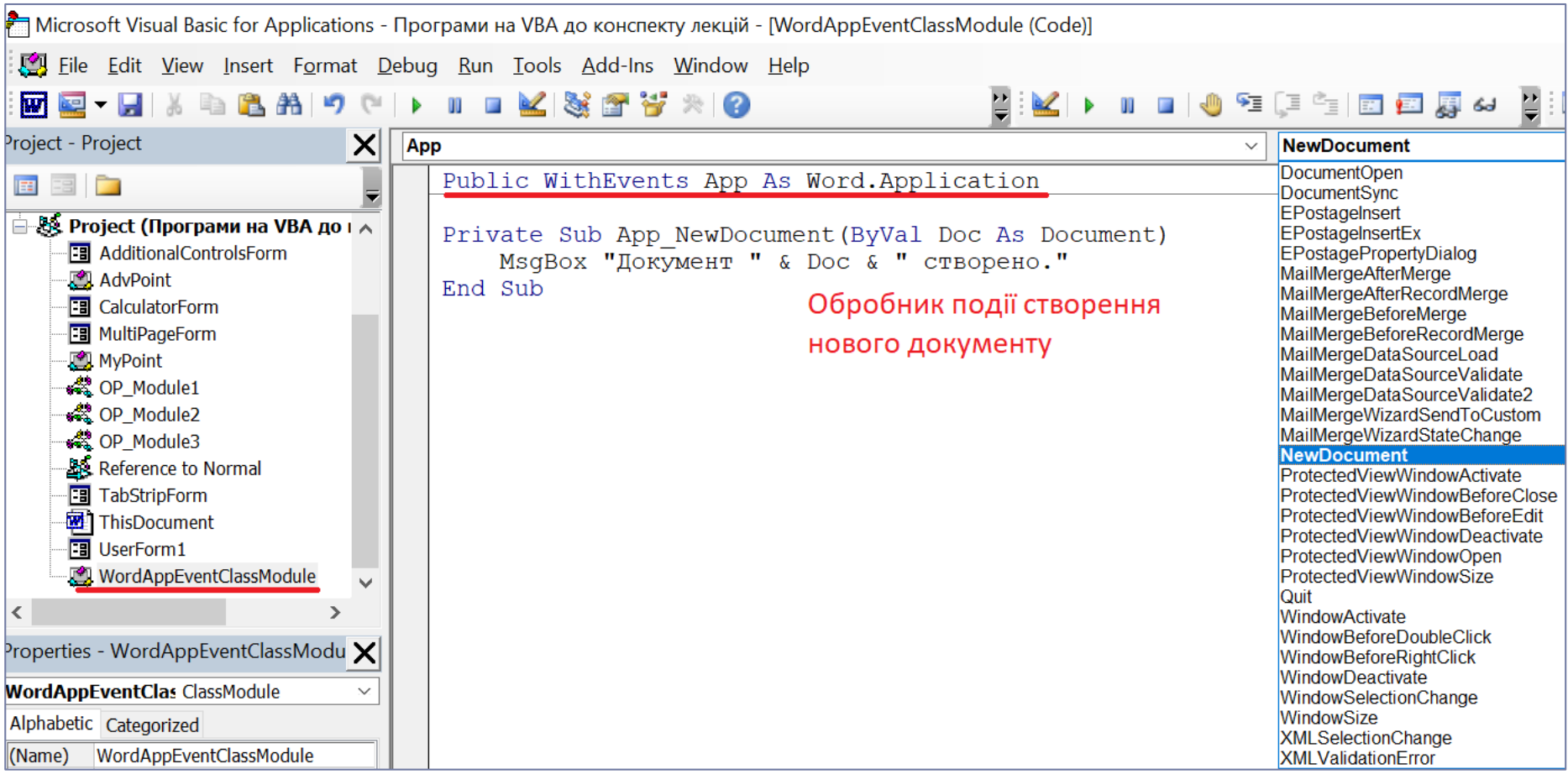
- Якщо код VBA виконується в Word (тобто Word вже запущений), то об'єкт `Application` створювати вже не потрібно. Він буде автоматично доступний.
- Якщо не вказано, до якого об'єкту відноситься та чи інша властивість або метод, компілятор VBA в Word автоматично вважає, що це властивість або метод належить об'єкту `Application`. Тому такий код функціонально однаковий:

```
Application.Selection.TypeText "Мій текст"
```

та

```
Selection.TypeText "Мій текст"
```

Об'єкт Application



Sub CreateDocWithEvent()

Властивості об'єкта Application

- `ActiveDocument` - повертає посилання на об'єкт активного документа (вікно якого є активним на цей момент) в даному екземплярі Word.
- `ActivePrinter` - дозволяє отримати або налаштувати активний принтер в ході роботи програми.
- `AutomationSecurity` - визначає рівень безпеки при програмному відкритті файлів. За умовчанням встановлено значення `msoAutomationSecurityLow`, що означає - відкривати зі включеними макросами. Можна також використовувати значення `msoAutomationSecurityForceDisable` - відключити макроси і `msoAutomationSecurityByUI` - використовувати налаштування безпеки, встановлені за допомогою графічного інтерфейсу.
- `BackgroundPrintingStatus` - містить кількість завдань Word, які знаходяться у черзі на друк.
- `Build` - містить версію і номер зборки Word.
- CapsLock - за допомогою цієї властивості можна перевірити, чи включений режим CapsLock на клавіатурі. Аналогічно працює властивість `NumLock`.

Властивості об'єкта Application

- `Caption` - дозволяє замінити слова "Microsoft Word" в заголовку вікна на інший текст, наприклад, "Мій додаток".
- `CheckLanguage` - вказує, чи визначає Word в автоматичному режимі мову, на якій виконується введення тексту. Якщо в системі встановлено декілька мов введення, то за замовчуванням перевіряє.
- `COMAdIns` - дозволяє отримати посилання на колекцію завантажених COM Add-ins - вбудованих в Word додатків, побудованих за технологією COM. Дуже зручно для перевірок перед зверненням до даного вбудованого додатку.
- `CustomizationContext` - дозволяє вказати шаблон або документ, на який будуть поширюватися внесені Вами зміни в меню, панелі інструментів і клавіатурні комбінації. Наприклад, код

  `CustomizationContext = NormalTemplate`

  говорить про те, що всі зміни, які ви будете вносити починаючи з цього моменту, будуть зберігатись в шаблоні normal.dot (і, таким чином, будуть застосовуватися у всіх документах).
- `Dialogs` - містить посилання на колекцію `Dialogs`, що представляє із себе всі можливі діалогові вікна Word.

Sub UseDialogs()

Властивості об'єкта Application

- `DefaultSaveFormat` - визначає формат збереження файлів Word за замовчуванням (той, який буде пропонуватися користувач в діалогових *вікнах Save As*: Word Document - `""` (пустий рядок), Document Template - `"Dot"`, Text Only - `"Text"`, `Text Only with Line Breaks – "CRText"`, MS-DOS Text - `"8Text"`, MS-DOS Text with Line Breaks - `"8CRText"`, Rich Text Format - `"Rtf"`, Unicode Text - `"Unicode"`, можна налаштувати на збереження в необхідному форматі.
- `DisplayAlerts` - дуже важлива властивість, вона дозволяє відключити виведення помилок у діалогових вікнах при роботі макросів і додатків VBA.
- `DisplayAutoCompleteTips` - дозволяє включити/відключити підказки для автозавершення тексту, найчастіше необхідно відключити.
- `Documents` - мабуть, найважливіша властивість, містить посилання на колекцію документів.
- `EmailOptions` - повертає дуже складний і насичений властивостями об'єкт `EmailOptions`, який використовується для налаштування Word як редактора поштових повідомлень в Outlook.

Властивості об'єкта Application

- `EnableCancelKey` - визначає, чи зможе користувач перервати виконання макросу при натисканні комбінації клавіш <Ctrl> + <Break>. Якщо встановити значення `WdCancelDisabled`, то макрос, який увійшов у нескінченний цикл, можна буде закрити лише разом з Word - через Task Manager.
- `FeatureInstall` - дозволяє заборонити Office встановлювати ще не встановлені компоненти. Для цього потрібно встановити значення цієї властивість у `msoFeatureInstallNone`.
- `FileDialog(fileDialogType)` - містить посилання на об'єкт `FileDialog`, тобто вікно вибору файлу, каталогу, відкриття файлу або збереження. Для відкриття цього вікна необхідно скористатися методом `Show()` цього об'єкта. Значення параметру `fileDialogType`:

  `msoFileDialogFilePicker` - дозволяє користувачу обрати файл.

  `msoFileDialogFolderPicker` - дозволяє користувачу обрати каталог.

  `msoFileDialogOpen` - дозволяє користувачу відкрити файл.

  `msoFileDialogSaveAs` - дозволяє користувачу зберегти файл.

Sub UseFileDialogOpen()

Властивості об'єкта Application

- `International` - містить інформацію про поточні регіональні налаштування.
- `Language` - визначає мову інтерфейсу MS Word. Для російської мови буде повертатися значення 1049, для англійської - 1033. Додаткову інформацію (про мову допомоги, мову програми установки і т.і.) можна отримати за допомогою властивості `LanguageSettings`.
- `MacroContainer` - дозволяє в ході виконання програми визначити, звідки був запущений поточний програмний код (зазвичай перевіряються два варіанти - normal.dot або поточний документ).
- `NewDocument` - одна з можливостей створити новий документ Word: `Application.NewDocument.Add()`
- `NormalTemplate` - дозволяє отримати посилання на об'єкт `Template`, який представляє normal.dot.
- `Options` - містить посилання на об'єкт `Option` з величезною кількістю властивостей. Через цей об'єкт програмним способом можна налаштувати значення з усіх вкладок, доступних на графічному екрані через меню *Сервіс -> Параметри*.

Властивості об'єкта Application

- `Path` - містить посилання на об'єкт зі шляхом до виконуваного файлу додатку MS Word.
- `PrintPreview` - дозволяє перейти в режим попереднього перегляду поточного документа або перевірити, чи перебуваємо ми в цьому режимі.
- `ScreenUpdating` - дозволяє заборонити оновлення екрану (якщо встановити її значення в `False`). Зазвичай використовується для прискорення роботи процедур, які виконують виведення на екран.
- `Selection` - містить посилання на об'єкт `Selection` - виділену частину документу або місце, у якому знаходиться курсор.
- `SpecialMode` - перевіряє, чи не перебуває Word в спеціальному режимі копіювання і вставки (для переходу в цей режим потрібно виділити текст і натиснути <F2> або <Shift> + <F2>), а потім перемістити курсор і натиснути <Enter>).
- `StartUpPath` - можливість переглянути/визначити шлях до каталогу автозапуску. Шаблони і вбудовані додатки, які знаходяться в цьому каталозі, Word при запуску відкриває автоматично. За замовчуванням каталог автозапуску знаходиться в профілі користувача за шляхом AppData\Roaming\Microsoft\Word\STARTUP\.

Властивості об'єкта Application

- `StatusBar` - дозволяє вивести текст у рядку в нижній частині вікна програми, в якій виводиться інформація про сторінки, мову і т.і.
- `System` - містить посилання на об'єкт `System`, призначений для отримання інформації з операційної системи (регіональні налаштування, тип курсору миші, тип процесора і т.і.). Також використовується для підключення мережевих дисків і запуску додатку `Microsoft System Information`.
- `Tasks` - містить посилання на колекцію `Tasks` з об'єктами `Task`, що представляють всі працюючі в системі процеси. За допомогою цих об'єктів можна програмним способом знайти додаток, який виконується у системі і що-небудь з ним зробити (активізувати, закрити, передати в його вікно повідомлення Windows - як при роботі з Windows API і т.і.). Запускати зовнішні додатки найкраще за допомогою спеціального об'єкта `Shell`.
- `UserControl` - дозволяє визначити, як саме був запущений Word - користувачем вручну або програмним чином.
- `UserInitials` і `UserName` - дозволяє отримати або визначити інформацію про ініціали та ім’я користувача з властивостей документу.

Властивості об'єкта Application

- `VBE` - містить посилання на недокументований, але дуже цікавий об'єкт `VBE`, що представляє собою редактор Visual Basic.
- `Version` - містить посилання на версію Word (менш детальну, ніж `Build`). Для Word 2016 значення цієї властивості - 16.0).
- `Visible` - дозволяє сховати вікно Microsoft Word.
- `Windows` - містить посилання на колекцію Windows, що представляє об'єкти вікон документів Word.
- `WindowsState` - дозволяє згорнути/розгорнути/відновити вікно Word.

Sub ApplicationProperties()

Найважливіші методи об'єкта Application

- `Activate()` - активізує вікно Word з поточним документом, зазвичай потрібно активізувати певний документ, тому використовується цей метод.
- `BuildKeyCode()` - повертає унікальний номер для клавіатурної комбінації.
- `ChangeFileOpenDirectory()` - дозволяє змінити каталог, який за замовчуванням використовується для роботи з документами (за замовчуванням, "Мої документи").
- `CheckGrammar()` і `CheckSpelling()` - дозволяють перевірити граматику і орфографію для аргументів-рядків. Найчастіше використовуються для об'єктів `Document` і `Range`.
- `CleanString()` - дозволяє "чистити" переданий аргумент-рядок (отриманий, наприклад, від об'єктів `Selection` або `Range`) від спеціальних символів Word і перетворювати їх в звичайний текст.
- `DefaultWebOptions()` - повертає однойменний об'єкт, за допомогою якого можна визначити властивості, що використовуються при збереженні документа Word в форматі HTML (кодування, робота з зображеннями, CSS, з якими браузерами забезпечувати сумісність і т.і.)

Найважливіші методи об'єкта Application

- `GoBack()` - забезпечує перехід на останнє місце редагування в документі (Word зберігає з документом три останні точки редагування).
- `GoForward()` - забезпечує перехід вперед по точкам збереження місця редагування в документі.
- `KeyBoard()` - дозволяє програмним способом перемикати розкладку клавіатури в Word, запобігаючи у такий спосіб помилкам при введенні тексту. Приймає код мови (1033 - англійська, 1058 - українська, 1049 російська, або https://docs.microsoft.com/en-us/deployoffice/office2016/language-identifiers-and-optionstate-id-values-in-office-2016)

  Якщо до цього методу нічого не передавати, він поверне поточну розкладку клавіатури.
- `KeyString()` - метод, зворотний `BuildKeyCode()`, - повертає клавіатурну комбінацію для даного унікального ідентифікатора;
- `ListCommands()` - створює новий документ і виводить в ньому у вигляді таблиці довідник по командам і клавіатурним комбінаціям Word - як стандартним, так і призначеним користувачем. При введенні аргументу `True` виводиться повний список можливих команд, а при введенні `False` - тільки команди, для яких є клавіатурні комбінації.

Найважливіші методи об'єкта Application

- `OnTime()` - дозволяє виконати макрос Word або у вказаний час, або по закінченні якогось часу. В Word одночасно може працювати тільки один таймер. За допомогою цього методу можна виконувати ресурсномісткі операції в автоматичному режимі, наприклад:

```
Application.OnTime When:="15:55:00", Name:="Macro1"
```

- `OrganizerCopy()` - дозволяє копіювати макрос, панель інструментів, запис автотекста або стиль між документами. Для об'єкта `Application` передбачені також методи `OrganizerDelete()` і `OrganizerRename()`.
- `PrintOut()` - дозволяє вивести на друк весь документ або його частину. Наступний приклад коду організує друк усіх документів каталогу:

```
adoc = Dir("*.DOC?")
Do While adoc <> ""
    Application.PrintOut FileName:=adoc
    adoc = Dir()
Loop
```

Найважливіші методи об'єкта Application

- `Quit()` - дозволяє закрити Word зі збереженням або без збереження документів. Його аргумент SaveChanges може мати одне з трьох значень для відповідної опції зберігання файлу: `WdSaveOptions.wdDoNotSaveChanges`, `WdSaveOptions.wdPromptToSaveChanges` або `WdSaveOptions.wdSaveChanges`.
- `Repeat()` - повторює останню виконану команду вказану аргументом кількість разів.
- `ResetIgnoreAll()` - в ході перевірки орфографії знімає мітку зі всіх ділянок тексту, позначених як "без перевірки".
- `Run()` - дозволяє запустити процедуру/макрос з відкритого шаблону/документа і передати їй параметри.
- `ScreenRefresh()` - оновлює вікно програми, зазвичай використовується після того, як автоматичне оновлення було відключено за допомогою властивості `ScreenUpdating`.
- `ShowClipboard()` - показує панель буфера обміну Word.

Решта методів відносяться до роботи протоколу DDE, перетворенню різних одиниць вимірювань і т.і.

Sub ApplicationMethods()

Найважливіші події об'єкта Application

- Об'єкт `Application` має багато подій - відкриття/закриття/ збереження/друк документа, клацання мишею, активізація, вихід з програми і т.і. Призначення подій зрозуміло з їх назв, тому вони тут розглядатися не будуть. Єдине, що слід ще раз відзначити - події об'єкта `Application` за замовчуванням не показуються в редакторі Visual Basic. Щоб вони з'явилися, в розділ `Declarations` об’єктного модуля (модуля класу або форми) потрібно помістити такий рядок:

```
Public WithEvents App As Word.Application
```

В цьому випадку в списку об'єктів у вікні редактора коду для форм з'явиться об'єкт `App` з усіма необхідними подіями.

Sub ApplicationMethods()

Колекція Documents та об’єкти Document

- Найчастіше в програмах нам потрібно:
  - запустити Word;
  - створити або відкрити документ;
  - щось з цим документом зробити (наприклад, ввести у необхідні місця цього документа значення, отримані з бази даних або від користувача).
- Запуск Word проводиться створенням об'єкту `Word.Application`, приклади створення наведені в попередньому розділі.
- Створення документа, відкриття документа, перевірка, чи вже відкритий документ, збереження документа і т.і. реалізується за допомогою колекції `Documents` обєкта `Application` і об'єкта `Document`.
- Найпростіший варіант створення документа виглядає так:

```
Dim oDoc As Word.Document
Set oDoc = Application.Documents.Add("E:\LR-Otchet-Template-ESEI-ZNU_uk.dotx")
```

- Аргументом методу Add вказаний шаблон, на базі якого створюється новий документ, якщо його не вказувати буде використаний шаблон normal.dotx.

Колекція Documents та об’єкти Document

- Після створення документу за допомогою об'єктів `Bookmark` і `Range` (про них - в наступних розділах) можна вводити текст в новий документ.
- Відкрити документ можливо за допомогою методу `Open()` колекції `Documents`. Найпростіший варіант застосування цього методу виглядає так:

```
Set oDoc = Documents.Open("e:\mydoc.docx")
```

  Якщо цей документ вже відкрито, то створюється посилання на вже відкритий документ.
- Зберігати документи краще за допомогою методів об'єкта Document `Save()`, який зберігає усі документи колекції `Documents`, і `SaveAs2()`, який зберігає документ з новим ім’ям.
- В Word можна відкривати не тільки документи Word різних версій, а й документи десятків інших різних форматів - TXT, HTML, XML і т.і. Зберігати також можна в одному з десятків вбудованих форматів або використовувати свій власний користувальницький формат (за допомогою об'єкта `FileConvertor`).

Sub DocumentCreationSavingOpen()

Властивості і методи колекції Documents

- Колекція `Documents` містить усі документи Word, відкриті в даний момент. Починається нумерація документів в колекції з `1`. З властивостей цієї колекції практичний інтерес може представляти властивість `Count` - кількість відкритих документів. Набагато частіше застосовуються методи цієї колекції:
- `Add()` - цей метод дозволяє створити і відразу ж відкрити новий документ (і повернути посилання на його об'єкт). Це - найбільш поширений спосіб створення нових документів в Word. Повний синтаксис цього методу:

  `Add (Template, NewTemplate, DocumentType, Visible)`

  `Template` - це шаблон для створення нового документа, `NewTemplate` (`True/False`) - вказує, чи буде новий документ шаблоном, `DocumentType` - варіанти: `wdNewBlankDocument`, `wdNewEmailMessage`, `wdNewFrameset` або `wdNewWebPage` (за замовчуванням - новий чистий документ `wdNewBlankDocument`), `Visible` - чи буде новий документ видимим (`True`) (за замовчуванням) або невидимим (`False`).

Методи колекції Documents

- `Open()` - дозволяє відкрити документ з диска і додати його в колекцію. Цей метод приймає безліч параметрів, з яких обов'язковим є тільки один - ім'я документа (разом з шляхом до нього).
- Метод `Item()` дозволяє знайти потрібний документ в колекції за його індексом. Але зазвичай для отримання посилання на потрібний документ використовується конструкція `For...Next` з перевіркою значення якої-небудь властивості документа через `If`. Найчастіше це властивість - `Name`.

```
Dim oDoc As Document

For i = 1 To Documents.Count
    Set oDoc = Documents.Item(i)
    If oDoc.Name = "doc1.doc" Then
        Exit For
    End If
    Set oDoc = Nothing
Next
```

Методи колекції Documents

- Через властивість `Item` можна отримати доступ до об'єкта документа безпосередньо. Наприклад, в цьому прикладі ми отримуємо ім'я першого документа в колекції `Documents`:

```
MsgBox Documents.Item(1).Name
```

- Методи `Save()` і `Close()` - дозволяють зберегти і закрити всі документи в колекції, відповідно.
- Методи `CanCheckOut()` (чи можна "забрати" документ в монопольний доступ) і `CheckOut()` (забрати документ в монопольний доступ) можна застосовувати, якщо документ знаходиться в бібліотеці документів в базі даних SharePoint Portal Server.

Робота з об'єктом Document

- Після запуску Word можна виконувати з ним різні дії. Розглянемо найбільш важливі і часто використовувані властивості, методи і події.
- Зверніть увагу на можливість звернення до об'єкту `Document` не створюючи спеціальної об'єктної змінної:
  - можливо працювати з документом як з елементом колекції `Documents`, при цьому звернення може виглядати, наприклад, так: `Documents.Item(1);`
  - можливо використовувати ключове слово `ThisDocument`, за допомогою якого можна отримати посилання на об'єкт документа, якому належить виконуваний програмний модуль, наприклад:

    ```
    MsgBox ThisDocument.Name
    ```
  - можливо використовувати властивість `ActiveDocument` об'єкта `Application`. Ця властивість повертає посилання на об'єкт активного документа:

    ```
    MsgBox Application.ActiveDocument.Name
    ```

    або просто

    ```
    MsgBox ActiveDocument.Name
    ```

Найважливіші властивості об'єкта Document

- `AttachedTemplate` - можливість підключити шаблон (з усіма макросами, стилями, записами автотекста і т.і.) або перевірити, який шаблон підключений (вручну це можна зробити через меню *Сервіс -> Шаблони і надбудови*).
- `Background` - повертає об'єкт `Shape`, що представляє фоновий малюнок;
- `BuiltInDocumentProperties` - можливість отримати посилання на колекцію `DocumentProperties` з однойменними об'єктами, що представляють вбудовані властивості документа (назва, автор, категорія, коментарі і т.і.).
- `Characters` - повертає колекцію об'єктів `Range`, кожен з яких представляє один символ. Це властивість є не тільки у об'єктів `Document`, а й у об'єктів `Selection` і `Range`. Може використовуватися, наприклад, для виконання операція пошуку та заміни або статистичних підрахунків (наприклад, кількості символів документу).
- `Content` - повертає об'єкт `Range`, який представляє *головний ланцюжок документа* (*main document story*) - текст документа, без колонтитулів, виносок, коментарів і т.і.

Найважливіші властивості об'єкта Document

- `CustomDocumentsProperties` - повертає колекцію об'єктів `DocumentProperties`, що представляють користувацькі властивості документа. Можна використовувати для збереження разом з документом властивостей, які зберігають інформацію, наприклад, про кількість відкриттів документів, прапорці друкувався/не друкувався документ, скільки разів викликалася та чи інша функція, на яких комп'ютерах і хто з користувачів відкривав документ і т.і.
- `DefaultTabStop` - визначити відступ за замовчуванням при використанні символу табуляції, за замовчуванням - 35 пунктів, що приблизно дорівнює 1,25 см;
- `DisableFeatures` - відключити можливості, які мають тільки останні версії Word (для сумісності з користувачами, у яких на комп'ютерах стоять старі версії). Зазвичай саме властивість `DisableFeatures` просто включає цей режим, а конкретний рівень сумісності задається за допомогою властивості `DisableFeaturesIntroducedAfter`.

Найважливіші властивості об'єкта Document

- `DoNotEmbedSystemFonts` - не вставляти в документ системні шрифти (за замовчуванням вставляються для російського, японського тощо). Дозволяє скоротити розмір документа - але тоді користувачі у системі, де не встановлено російську мову, не зможуть прочитати цей документ.
- `EmbedTrueTypeFonts` - дуже корисна властивість, якщо Ви працюєте з документом в місці, де використовуються екзотичні шрифти (наприклад, у видавництві). Вставка TrueType шрифтів гарантує, що одержувачі документа бачитимуть його точно таким же, як і творець.
- `Envelope` - дозволяє отримати посилання на спеціальний об'єкт `Envelope`, який використовується для створення поштових конвертів.
- `Fields` - можливість отримати посилання на колекцію `Fields` однойменних об'єктів. Дуже корисна при роботі з полями.
- `Footnotes` - можливість отримати колекцію виносок.
- властивості `Formatting...` - зазначають, що показувати в списку стилів в панелі інструментів Форматування.

Найважливіші властивості об'єкта Document

- `FormFields` - аналогічна `Fields`, але в цьому випадку ми отримуємо посилання на поля в формах.
- `FullName` - повне ім'я об'єкта (разом з шляхом до нього в файлової системі або Web), доступна тільки для читання.
- `GrammarChecked` - дозволяє позначити весь документ, як перевірений з точки зору граматики (фактично відключити перевірку граматики для даного документа). Таку ж властивість має і об'єкт `Range`. Колекцію помилок, виловлених при перевірці граматики, можна отримати за допомогою властивості `GrammaticalErrors`, а виділити помилки зеленим хвилястим підкресленням (якщо вони ще не виділені) - за допомогою властивості `ShowGrammaticalErrors`. Для орфографічних помилок існує аналогічні властивості `SpellingChecked`, `SpellingErrors` і `ShowSpellingErrors`.
- `HasPassword` - містить інформацію, чи призначено пароль для документа, Password - дозволяє призначити пароль. Унаслідок крайньої слабкості парольного захисту, паролі в Word, Excel і Access використовувати не рекомендується.

Найважливіші властивості об'єкта Document

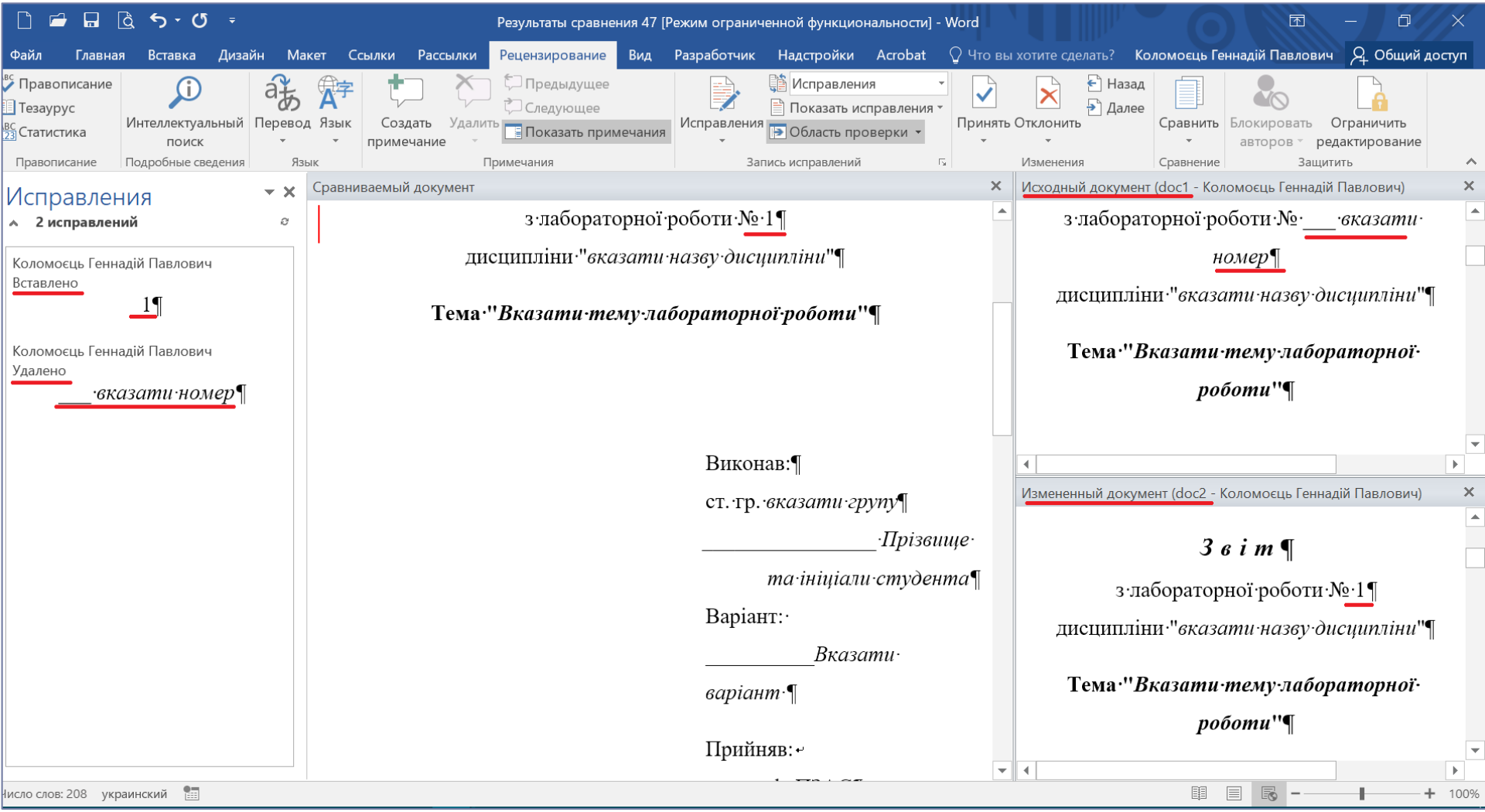
- `Indexes` - повертає колекцію індексів (тобто предметних покажчиків) для документа.
- `Name` - повертає ім'я документа (без шляху до нього).
- `PageSetup` - дозволяє отримати посилання на однойменний об'єкт, що використовується для встановлення налаштувань перед друком документу.
- `Paragraphs` - повертає посилання на колекцію абзаців в даному документі.
- `Path` - повертає шлях до документа у файлової системі (без імені), може стати в нагоді, щоб створити ще один файл тим самим шляхом.
- `Permission` - надає доступ до об'єкта `Permission`, який дозволяє управляти системою внутрішніх дозволів документа Word (який не був вирішений файловою системою).
- `PrintRevisions` - вказує, друкувати чи ні позначки редактора (виправлення) разом з документом (за замовчуванням - друкувати).
- `ProtectionType` - перевірити захист даного документа (дозволено все, або тільки коментарі, читання, зміни в полях форм і т.і.). Сам захист встановлюється за допомогою методу Protect().

Найважливіші властивості об'єкта Document

- `ReadOnly` - дозволяє встановити доступ до документа тільки для читання (відповідний атрибут встановлюється у файловій системі).
- `RemoveDateAndTime` і `RemovePersonalInformation` - дозволяє видалити з документа інформацію про дату і час проведених змін і всю інформацію про користувача (включаючи властивості документа).
- `Saved` - дозволяє визначити, чи змінювався документ з часу останньої зміни.
- `SaveFormat` - дозволяє отримати інформацію про формат документа (DOC, RTF, TXT, HTML і т.і.), доступна тільки для читання.
- `Sections` - повертає колекцію розділів документа. `Sentences` - те ж саме для речень. Аналогічно працюють властивості `Shapes`, `Styles`, `Subdocuments`, `Windows` і `Words`.
- `Tables` - повертає колекцію таблиць документа, властивість тільки для читання, зручна для доступу до таблиць у документі.
- `Type` - повертає тип документа (звичайний, шаблон або Web-сторінка з фреймами).
- `Variables` - можливо використовувати для збереження службових даних разом з документом.

Sub DocumentProperties()

Найважливіші методи об'єкта Document



Sub CompareDocuments()

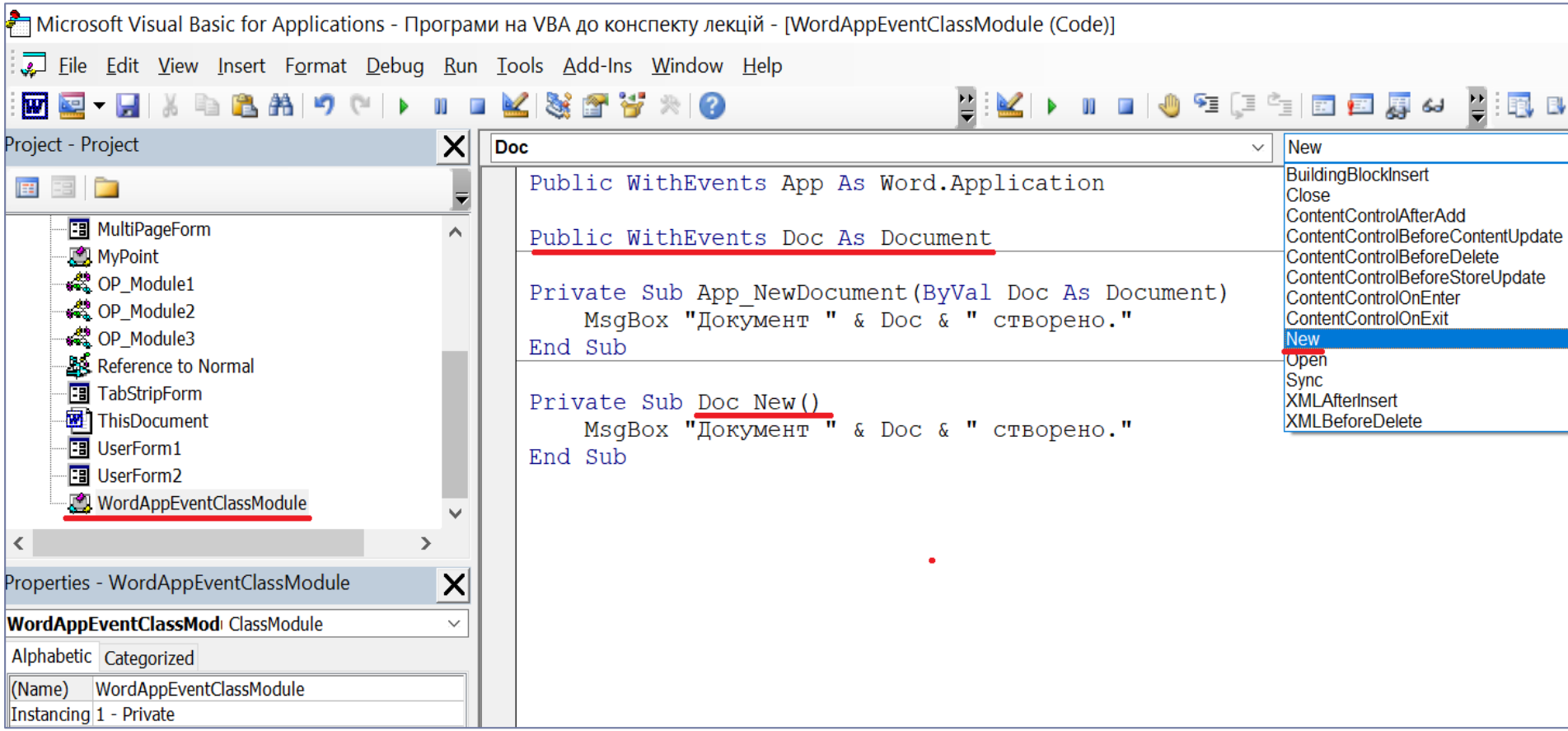
Найважливіші методи об'єкта Document

- `DetectLanguage()` - дозволяє визначити мову тексту, перевірка проводиться за реченнями, на основі звірки слів в них з вбудованими словниками. Така перевірка проводиться автоматично під час введення тексту або відкриття нового документа. Щоб заново провести перевірку мов, властивість `LanguageDetected` потрібно перевести в `False`.
- `FitToPages()` - дозволяє автоматично змінювати розмір шрифту таким чином, щоб текст став займати на одну сторінку менше. Можна використовувати для усунення "висячих сторінок" і інших проблем введення тексту.
- `FollowHyperlink()` - дозволяє відкрити вказаний документ у відповідному додатку (якщо HTML, то в Internet Explorer).
- `GoTo()` - існує для об'єктів `Document`, `Range` і `Selection`. У перших двох випадках він повертає об'єкт `Range`, в третьому - переміщує покажчик введення тексту на вказане як аргумент місце. Вміє переходити на вказані сторінку, рядок, закладку, коментар, таблицю, секцію, поле, посилання, формулу і т.і. (буде розглянутий двлі).

Найважливіші методи об'єкта Document

- `Merge()` - виконує злиття двох документів.
- `PresentIt()` - відкриває документ Word в PowerPoint.
- `PrintOut()` - виводить на друк весь документ або його частину, може використовуватися для об'єктів Application, Document і Window.
- `PrintPreview()` - відкриває документ в режимі попереднього перегляду.
- `Protect()` - обмежує внесення змін до документа за допомогою пароля або IRM.
- `Range()` - повертає об'єкт `Range`, приймає в якості параметрів номер початкового символу діапазону і номер кінцевого символу (роботу з `Range` буде описано далі).
- `Redo()` - повторює останню дію, як параметр приймає кількість останніх дій, повертає True, якщо повтор був виконаний успішно.
- `Repaginate()` - виконує повторний розподіл документу за сторінками, зазвичай використовується, якщо автоматична розбивка була раніше відключена.
- `Save()` - зберігає документ, якщо документ ще не зберігався, відкривається діалогове вікно *Save As*.

Найважливіші методи та події об'єкта Document



Private Sub Doc_New() - модуль класу WordAppEventClassModule

Робота з об'єктом Selection

- Після запуску Word, створення/відкриття/активізації документу, наступна дія, які виконується найчастіше - введення або редагування тексту в потрібному місці. Для цього використовуються об'єкти `Selection`, `Range` і `Bookmark`.
- Зазвичай перед тим, як що-небудь зробити у вікні документа Word, користувач або виділяє потрібний фрагмент тексту, або переставляє покажчик вставки тексту в потрібне місце. Об'єкт `Selection` представляє саме такий виділений фрагмент тексту (а якщо нічого не виділено, то місце, де знаходиться покажчик вставки), саме цей об'єкт зазвичай використовує макрорекордер.
- Створювати об'єкт `Selection` і отримувати на нього посилання не обов'язково (а зазвичай і просто неможливо), оскільки об'єкт `Selection` в документі може бути тільки один. Він створюється автоматично при запуску Word і завжди доступний. Звертатися до нього можна так:

```
Application.Selection.Text = "текст, що
                                вставляється"
```

або просто

```
Selection.Text = "текст, що вставляється"
```

Робота з об'єктом Selection

Потрібно визначити місце, на яке вказує об'єкт Selection, щоб виділити потрібний нам фрагмент тексту або точку для введення. Для цього існують такі способи:

- покластися на виділення потрібного тексту користувачем, зазвичай такий спосіб застосовується для складного редагування/форматування ділянок тексту і для введення інформації у вказане користувачем місце документа, коли у автоматичному режимі потрібне місце знайти важко;
- скористатися методом `Select()` різних об'єктів (`Document`, `Range`, `Bookmark`, `Table` та їх підоб'єктів типу стовпців і рядків, `PageNumber`, `Field` і т.і.), цей метод просто виділяє увесь документ, діапазон, закладку, таблицю і т.і., відповідно;
- скористатися методами об'єкта `Selection`, щоб перетворити вже існуюче виділення на необхідне;
- скористатися об'єктом `Find` для пошуку потрібної ділянки тексту;
- якщо необхідно вводити інформацію в самий початок документа, можна взагалі нічого не робити - за замовчуванням покажчик вставки встановлюється на початок активного документа.

Робота з об'єктом Selection

- Якщо виділення потрібного місця виконує користувач, то він може виділити одночасно несуміжні ділянки тексту (за допомогою клавіші <Ctrl>), або виділити не текст, а частину таблиці, малюнок або інший нестандартний об'єкт в документі. У цьому випадку поведінка програми, яка працює з об'єктом `Selection`, може бути непередбачуваною, тому рекомендується завжди використовувати додаткові перевірки за допомогою властивостей `Type` і `Information` об'єкта `Selection`.
- Незважаючи на те, що застосування об'єкта `Selection` - найпростіший і наочний метод редагування тексту, на практиці програмісти використовують його рідко, оскільки при використанні цього об'єкта ми залежні від дій користувача. Якщо під час виконання коду користувач почне клацати по документу мишею, результат може бути абсолютно непередбачуваним. Захиститися від втручання користувача можна двома способами:
  - працювати з прихованим документом або з прихованим додатком Word. Для включення/відключення цього режиму використовують властивість `Visible` об'єктів `Document` і `Application`;
  - більш зручний спосіб - замість об'єкта `Selection` використовувати об'єкти `Range` і `Bookmark`.

Властивості об'єкта Selection

- `Bookmarks` - повертає колекцію `Bookmarks`, тобто все закладки, які є у виділеній ділянці тексту. *Закладки* - це один з найбільш використовуваних об'єктів у додатках VBA з використанням Word. детальніше про них буде викладено далі.
- `Start` і `End` - визначають номер першого і останнього символу у виділенні (по відношенню до *document story* - тобто *тексту документа*, або іншого story - наприклад, виносок). Перша позиція в *document story* - завжди 0.
- `ExtendMode` - дозволяє переключитися в режим виділення тексту, коли натискання клавіш зі стрілками <Home> і <End> призводить не до переміщення покажчика введення, а до зміни виділення.
- `Find` - повертає об'єкт `Find`, детальніше про цей об'єкт і про його вкладений об'єкт `Replace` буде викладено далі.
- `Font` - повертає об'єкт `Font`, за допомогою якого можна встановлювати властивості шрифту у виділенні.

```
Selection.Font.Name = "Arial"
Selection.Font.Size = 10
```

Властивості об'єкта Selection

- `Information` - повертає інформацію про виділення (в якій частині документа, всередині таблиці чи ні, чи включені клавіші <CapsLock> і <NumLock>, чи включений режим "Заміна" при введенні тексту, на якій сторінці знаходиться виділення і скільки всього сторінок і т. і.).
- `IPAtEndOfLine` - повертає `True`, якщо курсор введення тексту (insertion point - IP) знаходиться у кінці рядка.
- `LanguageId` - надає можливість помітити виділення, як написане певною мовою. Правильне визначення мови дозволяє уникнути проблем при перевірці орфографії.
- `NoProofing` - надає можливість скасувати для виділення перевірку орфографії і граматики. Дуже рекомендується позначати таким чином текст з програмним кодом, списками прізвищ, назв фірм, специфічними термінами і т.і.
- `Range` - надає можливість створити з виділення об'єкт Range.
- `Type` - використовується для перевірок, визначає тип виділеної частини тексту відповідно WdSelectionType Enumeration.
- `Text` - дозволяє ввести текст у місце виділення (або в місце, де стоїть покажчик).

Методи об'єкта Selection

- `Calculate()` - надає можливість порахувати математичний вираз прямо у процесі введення тексту і повернути його результат (використовуючи лише тип даних `Single`).
- `ClearFormatting()` - очищає форматування
- `Collapse()` - перетворює виділення у точку вставки. Можна використовувати два варіанти: точка вставки на початку виділення і у кінці виділення. Дуже зручно, якщо потрібно тільки вставити новий текст без видалення існуючого.
- `Copy(), CopyAsPicture(), Cut(), Paste() і Delete()` - виконують операції редагування з виділенням.
- `EndKey(Unit, Extend)` - дозволяє (в залежності від значення параметру `Unit`) перейти на кінець документу (`wdStory`), параграфа (`wdParagraph`), рядка (`wdLine`), слова (`wdWord`), стовпця (`wdColumn`) або рядка (`wdRow`) в таблиці (за замовчуванням - на кінець рядка (`wdLine`)) та або створити виділення до цього місця (параметр `Extend:=wdExtend`), або встановити у це місце точку вставки (параметр `Extend:= wdMove`). Щоб перевести курсор вставки на кінець тексту документа, можна скористатися командою:

```
Selection.EndKey Unit:=wdStory, Extend:=wdMove
```

Методи об'єкта Selection

- `HomeKey(Unit, Extend)` - аналогічний `EndKey(Unit, Extend)` метод, що дозволяє перейти на початок елемента.
- `EndOf(Unit, Extend)` - аналогічний `EndKey(Unit, Extend)` метод, на відміну від нього, він існує і у об’єкта `Range`.
- `StartOf(Unit, Extend)` - аналогічний `HomeKey(Unit, Extend)` метод, на відміну від нього, він існує і у об’єкта `Range`.
- `Expand(Unit)` - дозволяє розширити виділення на слово, речення, абзац тощо - в залежності від переданого параметра `Unit` з тими ж значеннями, як і для метода `EndKey(Unit, Extend)`.
- `Shrink(Unit)` - дозволяє звузити виділення на слово, речення, абзац тощо - в залежності від переданого параметра.
- `Extend(Unit)` - дозволяє розширити виділення на наступний юніт (замість слова - речення, замість речення - абзац і т.і.).

Методи об'єкта Selection

- `GoTo(What, Which, Count, Name)` - переміщує точку вставки в позицію символу, яка передує зазначеному елементу, і повертає об'єкт `Range` (всі параметри опціональні). У якості елементів (`What`) можуть виступати сторінка (`wdGoToPage`), розділ (`wdGoToSection`), рядок (`wdGoToLine`), заголовок (`wdGoToHeading`), таблиця (`wdGoToTable`), рисунок (`wdGoToGraphic`), закладка (`wdGoToBookmark`), формула (`wdGoToEquation`) та інші. У якості позиції, на яку переміщується точка вставки (`Which`), можуть виступати абсолютна позиція елемента (`wdGoToAbsolute`), перший вказаний елемент (`wdGoToFirst`), останній вказаний елемент (`wdGoToLast`), наступний вказаний елемент (`wdGoToNext`), попередній вказаний елемент (`wdGoToPrevious`), позиція вказаного елементу відносно поточної позиції точки вставки (`wdGoToRelative`). Параметр `Count` вказує кількість зазначених елементів, а `Name` - назву закладки (або коментаря, поля об'єкта).
- `GoToNext(What)` - дозволяє перейти на наступний рядок, сторінку, закладку і т.і. Аналогічно працює метод `GotoPrevious(What)` (перехід на попередній елемент). Параметр `What` має ті ж значення, що і для методу `GoTo(What, Which, Count, Name)`.

Методи об'єкта Selection

- Призначення методів `Insert(Text)` очевидне, найчастіше використовуються методи `InsertBefore(Text)` (вставити перед виділенням) і `InsertAfter(Text)` (вставити після виділення).
- Метод `Move(Unit, Count)` - дозволяє змішувати точку вставки на певну кількість символів, слів, речень, абзаців, розділів, стовпців і рядків таблиці і т.і (параметр `Unit` має ті ж значення, що і для методу `EndKey(Unit, Extend))`, параметр `Count` вказує кількість відповідних елементів. Використання цього методу дозволяє обійтися мінімальною кількістю змін в коді, якщо змінився вихідний шаблон для введення даних.
- Методи `Move`… також зустрічаються чи не в будь-якій програмі, пов'язаній з введенням тексту в Word. Найважливіші з цих методів: `MoveLeft(Unit, Count, Extend), MoveRight(Unit, Count, Extend), MoveUp(Unit, Count, Extend), MoveDown(Unit, Count, Extend), MoveEnd(Unit, Count), MoveStart(Unit, Count)`, призначення яких очевидне. Параметр `Extend` має значення як для методу `EndKey(Unit, Extend)`. Кожен з методів приймає додаткові параметри, за допомогою яких можна визначити, на скільки позицій буде переміщатися покажчик, чи буде рухатися виділення, поширюючись на нову область і т.і.

Методи об'єкта Selection

- Методи `MoveStartUntil(Cset, Count)`, `MoveStartWhile(Cset, Count)`, `MoveEndUntil(Cset, Count)`, `MoveStartWhile(Cset, Count)` відрізняються тим, що курсор вставки переміщується не на певну кількість символів, а поки не зустрінеться будь-який з зазначеної як параметр послідовності символів.
- `Next(Unit, Count)` - дозволяє перейти вперед на певну кількість символів, слів, речень, абзаців, розділів, стовпців і рядків таблиці і т.і. Перейти назад дозволяє метод `Previous(Unit, Count)`.
- `NextField()` - дозволяє перейти на наступне поле в формі (або перевірити, чи не скінчилися поля (в цьому випадку цей метод замість об'єкта `Field` поверне `Nothing`). Існує також і метод `PreviousField()`.
- `SelectColumn()`, `SelectRow()`, `SelectCell()` - дуже зручні методи для виконання різних операцій у таблиці Word.
- `SelectCurrentAlignment()`, `SelectCurrentFont()`, `SelectCurrentIndent()`, `SelectCurrentColor()` і т.і. - дозволяють виділити текст до місця зміни вирівнювання, зміни шрифту, зміни відступу, зміни кольору і т.і. Дуже зручні для цілей форматування або для виділення спеціальним чином доданого тексту.

Методи об'єкта Selection

- `SetRange()` - налаштовує виділення, методу передаються номер першого і останнього символу ділянки тексту, яку потрібно виділити. Нумерація починається з `0`, приховані службові символи рахуються.
- `Sort(), SortAscending(), SortDescending()` - виконують сортування за алфавітом, датами і т.і. абзаців або стовпців таблиці (які входять у виділення).
- `ToggleCharacterCode()` - дозволяє ввести код службового символу і тут же перетворити його в символ Unicode. Наприклад, щоб ввести символ Євро, можна скористатися командами:

```
Selection.TypeText Text: = "20ac"
Selection.ToggleCharacterCode
```

- `TypeText(Text)` - дозволяє вводити текст `Text` у виділення, приймає єдиний параметр - текст, який потрібно ввести. Чи буде перезаписаний поточний текст виділення, залежить від значення властивості `ReplaceSelection` об'єкта `Options`.
- `TypeParagraph()` - вставляє пустий параграф.
- `WholeStory()` - дозволяє виділити поточну частину документа (document story), зазвичай використовується, щоб виділити текст документа без виносок, редакторської правки, колонтитулів і т.і.

Sub SelectionUsing()

Робота з об'єктом Range, його властивості та методи

- `Range` - це програмний об'єкт, який представляє безперервний фрагмент тексту в документі. Цей об'єкт не залежить від об'єкта `Selection` - можливо працювати з об'єктом `Range`, не змінюючи поточного виділення. Об’єкт `Range` може не включати в себе жодного символу (представляти курсор введення тексту).
- Головна відмінність між об'єктами `Range` і `Selection` полягає в тому, що об'єкт `Selection` може визначити і користувач (виділивши текст мишею), а об'єкт `Range` можна визначити тільки програмно, і він не залежить від поточного положення покажчика або дій користувача.
- Також об'єктів `Range` в кожен момент часу може бути скільки завгодно, а об'єктів `Selection` - тільки один.
- Рекомендується, якщо це можливо, завжди використовувати об'єкт Range замість об'єкта Selection, тим самим захищаючи програму від можливих помилок, пов'язаних з діями користувача.

Робота з об'єктом Range, його властивості та методи

- Існують декілька способів створення об'єктів `Range`:
  - за допомогою методу `Range()` об'єкта `Document` з передачею як аргументів номерів початкового і кінцевого символу діапазону (якщо параметри опускаються, буде обраний увесь документ). Наприклад, створити діапазон, який буде включати в себе перші 10 символів документа, можна так:

```
Dim rngDoc As Range
Set rngDoc = ActiveDocument.Range(Start:=0,
                                  End:=10)
```

  - за допомогою властивості `Range`, яка передбачена для багатьох об'єктів (`Bookmark`, `Selection`, `Table-Row-Cell`, `Paragraph` і т.і.).;
  - за допомогою допоміжних властивостей (`Sections`, `Paragraphs`, `Sentences`, `Words`, `Characters`, `Tables` і т.і.), які ділять текст на діапазони - об'єкти `Range`. Ці властивості повертають колекції об'єктів `Range`;
  - перевизначенням існуючого об'єкту `Range`. Зазвичай для цієї мети використовується метод `Range.SetRange`;
  - використовуючи об'єкти `Range` закладок шаблону (найзручніший).

Робота з об'єктом Range, його властивості та методи

Деякі унікальні методи об'єкта Range:

- InStory() - [illegible] Range [illegible] Selection [illegible]
- IsEqual() - [illegible] повертає True.
- Relocate(Direction) - надає можливість переставити місцями абзаци в діапазоні, значення Direction - wdRelocateDown або wdRelocateUp вказують направлення переміщення поточного об'єкту Range.
- Select() - надає можливість створити об'єкт Selection на основі об'єкта Range (тобто просто виділити весь текст в цьому об'єкті), дуже зручно для налагоджувальних цілей.

Основні [illegible] Наприклад, [illegible] виділити [illegible] початкова позиція в документі [illegible] (або кінця виділення), можна за допомогою команд:

```
Sub RangeUsing()
Dim MyRange As Range
Set MyRange = ActiveDocument.Range(Start:=0, End:=0)
MyRange.SetRange Start:=MyRange.Start, End:=Selection.End
MyRange.Select 'для демонстрації діапазона
```

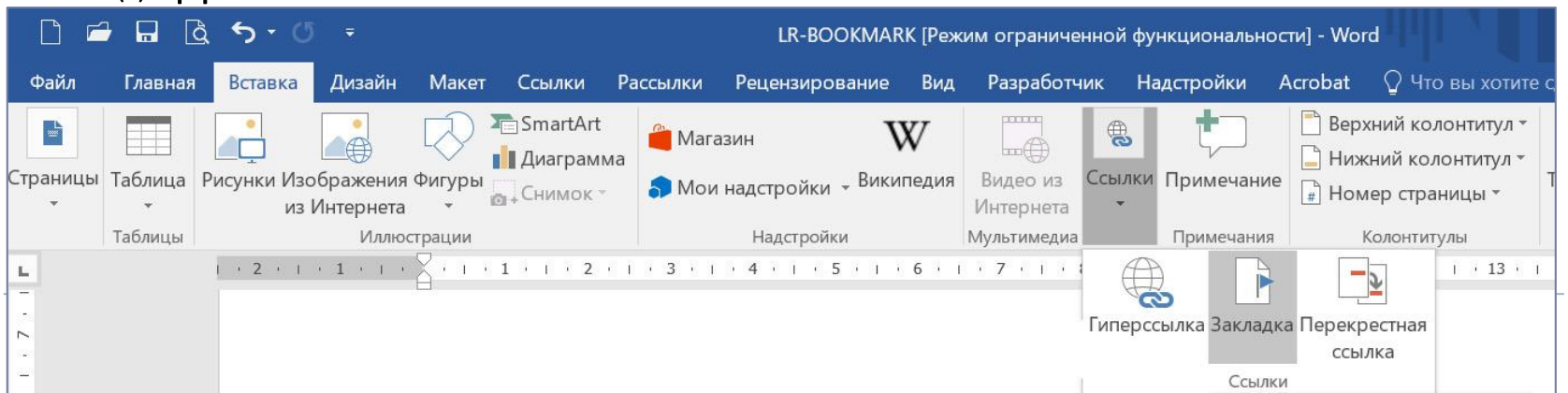
Форматування тексту за допомогою властивостей об’єктів Selection та Range

- Під час "ручного" форматування тексту ми, зазвичай, виділяємо певну частину тексту (слово, речення, параграф і т.і.) і далі застосовуємо до нього той або інший формат, при цьому команди форматування умовно поділяються на власне форматування (група команд *Главная-Шрифт*) та вирівнювання і встановлення відступів для частин тексту (група команд *Главная-Абзац*).
- Аналогічним чином форматування виконується і за допомогою програми: спочатку створюється об’єкт `Selection` (або об’єкт `Range`, або отримується об’єкт `Find` (буде викладено далі)), а потім використовуються його властивості `Font`, `ParagraphFormat` та інші.
- Також можливе встановлення кольору шрифту за допомогою властивості `Font.Color`.
- Для властивості `ParagraphFormat.Alignment` також можливі найбільш використовувані значення: `wdAlignParagraphLeft` `wdAlignParagraphRight`, `wdAlignParagraphCenter` та `wdAlignParagraphJustify` (останнє вирівнює текст по ширині сторінки).

Sub SelectionRangeFormat()

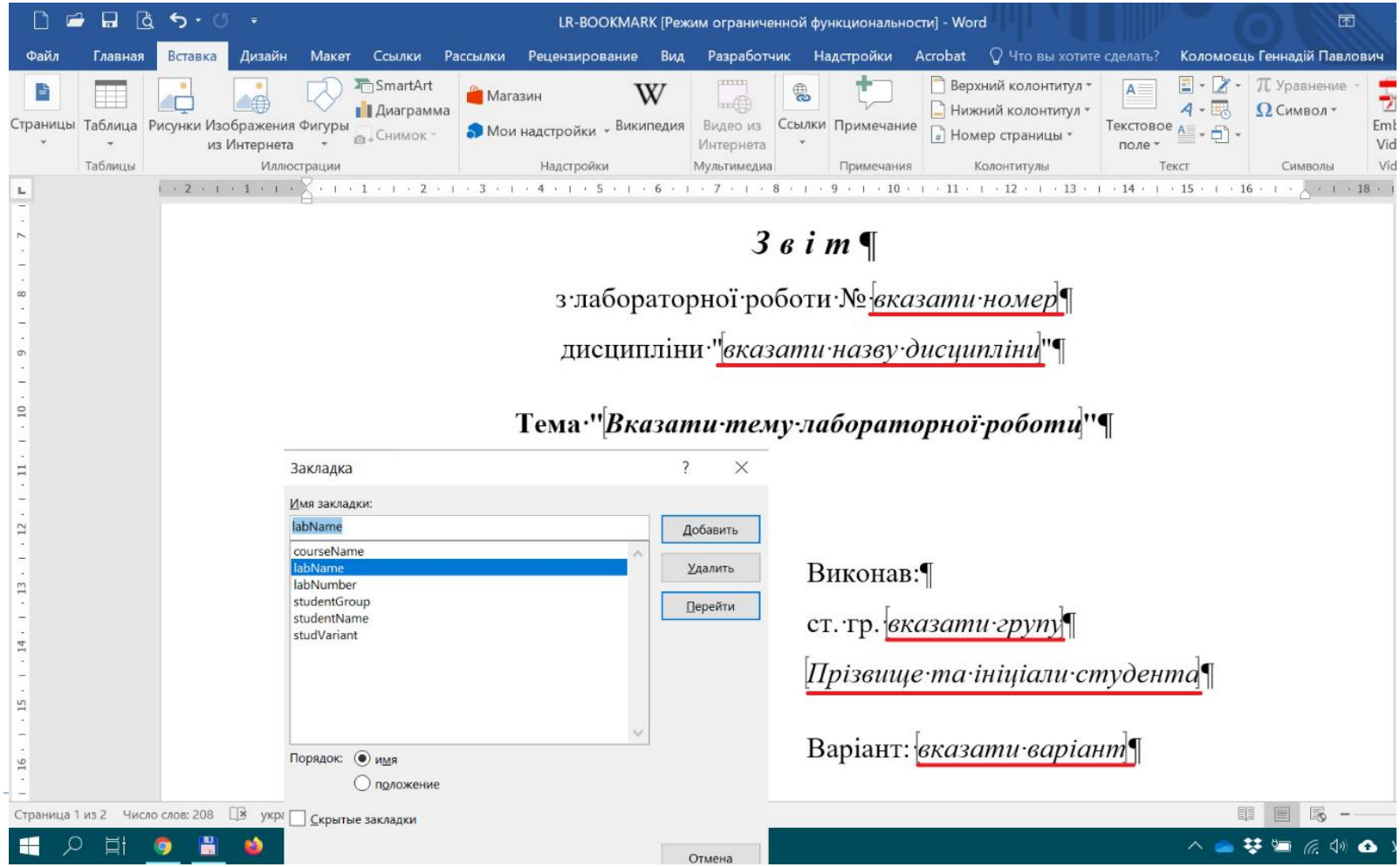
Робота з об'єктом Bookmark, його властивості та методи

- Об'єкт `Bookmark` - це закладка, використання якої є найзручнішим способом навігації по документах, створених за допомогою шаблонів (наприклад, звітів). Принципова відмінність об'єктів `Bookmark` від об'єктів `Selection` і `Range` полягає в тому, що всі закладки зберігаються разом з документом і після роботи з ним, у той час, як об’єкти `Selection` і `Range` перестають існувати після закриття документа. Якщо документ створено на основі шаблону, то всі закладки, які були визначені в шаблоні, будуть визначені і в створеному на основі цього шаблону документі.
- Створити закладку (меню *Вставка - Ссылки - Закладка*) набагато простіше, ніж рахувати кількість символів (розділів, абзаців і т.і.) для об'єкта `Range` від початку документа, або виконувати операції `Move()` для об'єкта `Selection`.



Робота з об'єктом Bookmark, його властивості та методи

- Досить зручним є відображення закладок у документі, яке можна включити командою *Файл-Параметры-Дополнительно-Показать содержимое документа-Показывать закладки*



Робота з об'єктом Bookmark, його властивості та методи

- Зазвичай немає потреби використовувати об'єкт `Bookmark` для роботи з текстом безпосередньо. З об'єкта `Bookmark` дуже просто отримати об'єкт `Selection` (за допомогою методу `Select()`) або об'єкт `Range` (за допомогою властивості `Range`) і далі - користуватися можливостями цих об'єктів, наприклад:

```
ThisDocument.Bookmarks("Bookmark1").Select
MsgBox Selection.Text
```

- Створювати об'єкти `Bookmark` програмним способом необов'язково, але якщо є така необхідність, то можна використовувати метод `Add()` колекції `Bookmarks`:

```
ThisDocument.Bookmarks.Add Name: = "temp",
                           Range: = Selection.Range
```

У цього методу - всього лише два параметри, обидва з яких використовуються у прикладі.

Робота з об'єктом Bookmark, його властивості та методи

Деякі важливі властивості об'єкта Bookmark:

- `Empty` - якщо це властивість повертає `True`, то це означає, що закладка вказує на точку вставки, а не та текст;
- `Name` - ім'я закладки, дуже зручно, що знайти потрібну закладку в колекції закладок можна не тільки за допомогою індексу (номера) закладки, але й за її ім’ям;
- `Range` - повертає об'єкт `Range` на місці цієї закладки;
- `Start, End, StoryType` - аналогічні таким же властивостям у об'єкта `Selection`.

Методів у об'єкта Bookmark всього три - Copy(), Delete() і Select().

- `Copy()` - створює закладку на основі існуючої,
- `Delete()` - видаляє її,
- `Select()` - виділяє те, на що посилається закладка.

Sub BookmarksUsing()

Об'єкти Find і Replacement

Об'єкти `Range` та `Selection` мають властивість `Find`, яка містить посилання на об’єкт `Find`, що належить цьому діапазонові або виділеній області. Для пошуку за допомогою об’єкту `Find` певного тексту або ознак форматування у тексті необхідно виконати наступні дії:

1) отримати посилання на об’єкт `Find` необхідного діапазону або виділеної області (для пошуку у всьому документі можна використовувати його властивість `Content`: `ActiveDocument.Content.Find);`
2) присвоїти шаблон пошуку та опції пошуку необхідним властивостям об’єкту `Find` (`Text`, `Style`, `Font`, `Format`, `Forward`, `MathCase`, `LanguageID` і т.і.);
3) викликати метод `Execute()` об’єкта `Find`, метод повертає `True` у разі, якщо шаблон пошуку був знайдений.
4) виконати перевірку властивості `Found` об’єкта `Find`, яка автоматично встановлюється у результат методу `Execute()`.

Використання методу `ClearFormatting()` об’єкту `Find` є загально прийнятою практикою і дозволяє виконувати пошук без врахування ознак форматування тексту.

Sub FindUsing()

Об'єкти Find і Replacement

- Об’єкт `Replacement` використовується для пошуку з заміною. Посилання на цей об’єкт можна отримати з властивості `Replacement` об’єкта `Find`.
- Властивості об’єкта `Replacement` - `Text, Style, Font` і т.і. встановлюються у значення, на які потрібно замінити знайдений текст та/або ознаку форматування.
- Параметр `Replace := wdReplaceAll` методу `Execute()` вказує на необхідність заміни усіх входжень шаблону, що шукається, інші доступні значення - `wdReplaceOne` - виконує заміну тільки першого входження, `wdReplaceNone` - не виконує заміну взагалі.

Sub ReplaceUsing()

- Для пошуку довільного тексту з необхідними налаштуваннями форматування необхідно задати шаблон пошуку для відповідних властивостей об’єкту `Find`, а його властивості `Text` присвоїти пустий рядок, а для заміни форматування довільного тексту необхідно присвоїти пустий рядок властивості `Text` об’єкту `Replacement`

Sub FindAndReplaceFormatUsing()

Об'єкти Table, Column, Row і Cell, робота з таблицями

- Створення таблиці у документі починається з того, що в колекцію `Tables` (вона передбачена для об'єктів `Document`, `Selection` і `Range`) додається новий об'єкт `Table`. Для цього використовується метод `Add()` з обов'язковими параметрами `Range` - діапазон, у який вставляється таблиця, `NumRows` - кількість рядків таблиці, `NumColumns` - кількість стовпців таблиці.
- Заповнення комірок таблиці даними може виконуватись за допомогою методу `Cell(Row, Column)` об'єкта `Table` з зазначенням координат комірки у таблиці або за допомогою метода `InsertAfter()` властивості `Range` об'єкта комірки `Cell`, який є елементом колекції `Cells`, доступної як властивість об'єкту `Range`, який, у свою чергу, є властивістю об'єкту `Table`.
- Зазвичай після створення таблиці використовують метод `AutoFormat()` об'єкта `Table` для виконання форматування та додання границь відповідно до одного з кілька десятків стилів-форматів, що визначаються елементом перерахування `WdTableFormat`.
- При зчитуванні змісту комірки виконується зміщення кінцевої межі діапазону на один символ вліво - це необхідно, щоб не зчитувати ознаку форматування кінця комірки.

Sub CreateTableAndReadTable()

Об'єкти Shape, колекція Shapes, робота з рисунками

- Об’єкт `Document` має властивість-колекцію `Shapes`, яка містить об’єкти `Shape`, що представляють фігури в документі. Кожен об'єкт `Shape` представляє об'єкт у *області рисунку* (*Canvas*), такий як автофігура (існує перерахування `MsoAutoShapeType`, яке визначає більше 180 автофігур - лінія, прямокутник, еліпс, зірки, стрілки і т.і), рисунок вільної форми який складається з автофігур, OLE-об'єкт або рисунок, що завантажується з файлу.
- Для цілей автоматизації найбільш цікава робота з зображеннями, які завантажуються з файлів. Для їх додання до документу використовується його властивість `Shapes`, до якої методом `AddCanvas(Left, Top, Width, Height, Anchor)` додають *область рисунку* (*Canvas*).

```
Set shpCanvas = ActiveDocument.Shapes.AddCanvas(
Left:=0, Top:=0, Width:=400, Height:=300,
Anchor:=rng)
```

- `Left, Top, Width, Height` визначають відстань у пунктах від місяця вставки та ширину та висоту області рисунку, а необов’язковий параметр `Anchor` представляє собою об’єкт `Range`, початок якого є місцем вставки рисунку.

Об'єкти Shape, колекція Shapes, робота з рисунками

- Для завантаження до *області рисунку* об'єкта Shape файлу з зображенням використовують метод `addPicture` властивості `CanvasItems` об’єкта `Shape`.

```
shpCanvas.CanvasItems.AddPicture
fileName:="image.jpg", LinkToFile:=msoFalse,
SaveWithDocument:=msoTrue, Left:=0, Top:=0,
Width:=400, Height:=300
```

`FileName` - повний шлях до файлу рисунку. Необов’язковий параметр `LinkToFile` дозволяє встановлювати зв’язок з файлом, з якого завантажується зображення (`msoTrue`) так, щоб при зміні файлу автоматично змінилося зображення у документі. За замовчуванням для цього параметру використовується `msoFalse`, що робить вставлений у документ рисунок незалежною копією. Значення `msoTrue` параметру `SaveWithDocument` дозволяє зберігати пов’язаний рисунок разом з документом (значення за замовчуванням - `msoFalse`). Параметри `Left`, `Top`, `Width`, `Height` позиціонують та визначають розмір рисунку відносно області рисунку (Canvas) (при встановленні для параметрів `Width` та `Height` значення `–1`, використовується оригінальний розмір рисунку).

Об'єкти Shape, колекція Shapes, робота з рисунками

- У разі, якщо використовуються тільки картинки-зображення з файлів, їх можна завантажувати до документу прямо у текстову область без використання *області рисунку* (*canvas*). При цьому вони додаються як об’єкти `InlineShape` у колекцію `InlineShapes`, посилання на яку міститься в однойменній властивості документа, діапазону (`Range`) або області виділення (`Selection`). Додання рисунків виконується методом `AddPicture(FileName, LinkToFile, SaveWithDocument, Range)` з параметрами аналогічними першим трьом параметрам методу `AddPicture()` об’єкта `CanvasShapes` (дивись вище). Параметр `Range` вказує на діапазон, у якому буде розміщений рисунок. Якщо діапазон не стиснутий (не використаний його метод `Collapse()`), при вставці картинка замінює діапазон, у противному разі - картинка вставляється у діапазон. Якщо параметр `Range` не вказаний картинка вставляється автоматично у поточний діапазон, який містить посилання на поточну колекцію `InlineShapes`.

Sub InsertPictures()

Колекція Windows і об'єкт Window

- Колекція `Windows` і об'єкт `Window` представляють вікна відкритих документів Word (кожен документ має мінімум одне вікно) і використовуються в основному для настройки зовнішнього вигляду цих вікон і навігації по ним. Колекцію `Windows` має об’єкт `Application`, а також об’єкт `Document`. Кожен об’єкт `Window` присутній в обох колекціях. При використанні посилання `ActiveWindow` об’єкта `ThisDcument` ми будемо звертатись до вікна, яке було активним при запуску процедури, а при використанні посилання `Application.ActiveWindow` (або просто `ActiveWindow`) - до документу, який був відкритий першим.

```
Dim Window1 As Window
Set Window1 = ThisDocument.ActiveWindow
Debug.print Window1.Caption
Set Window1 = ActiveWindow
Debug.print Window1.Caption
```

- Отримати доступ до вікна потрібного нам документа по його імені можна так:

```
Set Window1 = Windows("doc2.doc")
```

Колекція Windows і об'єкт Window

- Для документів, що створені у попередніх версіях додає до назви [Режим ограниченной функциональности] - це видно у заголовку вікна. У цьому разі ім’я документу-параметр повинне включати такий суфікс.
- Властивості, які визначають зовнішній вигляд вікна зазвичай приймають значення `True` або `False`, наприклад, для відображення вертикальної лінійки:

```
Window1.DisplayVerticalRuler = True
```

- Щоб змінити заголовок вікна, можна використовувати такий код:

```
Window1.Caption = "Мій додаток"
```

- За допомогою властивостей `Top, Left, Height` та `Width` можна змінювати положення та розміри вікна, якщо воно не розгорнуте на весь екран.
- Об’єкт `Window` має декілька методів, які дозволяють керувати вертикальною та горизонтальною прокруткою:

```
SmallScroll(Down, Up, ToRight, ToLeft)
LargeScroll(Down, Up, ToRight, ToLeft)
```

Аргументи Down, Up, ToRight, ToLeft зазначають кількість одиниць прокрутки у кожному напрямі.

Колекція Windows і об'єкт Window

- У методі `SmallScroll()` - це відстань, на яку зміщується екран при клікі на стрілці смуги прокрутки (приблизно один рядок для вертикальної смуги), а у методі `LargeScroll()` - вміст одного екрану. Якщо аргументи не вказуються, обидва методи прокручують вміст документа на одну одиницю вниз.
- Для посторінкової прокрутки використовують метод `PageScroll(Down, Up)` - його аргументи вказують кількість сторінок, на яку виконується прокрутка у відповідному напрямі, якщо аргументи не вказуються, виконується прокрутка на одну сторінку вниз.
- Для прокрутки до визначеної позиції використовують метод `ScrollIntoView(Obj, Start)`, аргумент `Obj` вказує на об’єкт `Range` або `Shape`, до якого потрібно прокрутити вміст документу. Необов'язковий аргумент `Start` визначає чи повинен лівий верхній кут об’єкта `Obj` співпадати з лівим верхнім кутом екрану (`Start=True` - значення за замовчуванням) або повинні співпадати праві нижні кути (`Start=False`).

Колекція Windows і об'єкт Window

- Кожен об’єкт `Window` має властивість `View` з посиланням на об’єкт класу `View`. Властивості цього об’єкту визначають особливості відображення вікна або його частини. Найбільш використовувані властивості цього об’єкту:
- `FullScreen` - перемикає відображення вікн у стандартний або повноекранний режим;
- `ShowAll` - вмикає відображення усіх символів, які не виводяться на друк, існує можливість вмикати відображення окремих об’єктів: `ShowBookmarks`, `ShowHighlight` (виділення);
- `TableGridlines` - визначає відображення меж таблиці;
- `Type` - визначає режим роботи з документом: `wdMasterView` та `wdOutlineView` - режим відображення структури документу , `wdNormalView` - нормальний режим, `wdPrintPreview` - режим попереднього перегляду при друку, `wdReadingView` - режим читання документу та інші.

Sub WindowsUsing()

Робота з діалоговими вікнами

- Об’єкт `Application` має властивість `Dialogs`, яка представляє собою колекцію діалогових вікон, які можна використовувати у програмі. Для використання діалогового вікна необхідно створити на нього посилання:

```
Dim MyDialog As Dialog
Set MyDialog = Dialogs(Dialog_Type)
```

- Існує `WdWordDialogEnumeration`, яка визначає близько 300 типів діалогових вікон із різним призначенням, наприклад, `wdDialogFileNew` - для створення нового файлу документу, `wdDialogFileOpen` - для відкриття існуючого файлу документу, `wdDialogConnect` - для підключення до мережевого диску, `wdDialogCopyFile` - для копіювання файлу і т.і.
- Після створення посилання на діалогове вікно його можна відобразити методом `Show(Timeout)` або `Display(Timeout)` цього об’єкту, `Timeout`- час у мілісекундах на протязі якого буде відображено діалогове вікно. Якщо параметр не вказаний, вікно буде відображене доки користувач не зачинить його. При використанні методу `Show()` зміни, виконані у діалоговому вікні виконуються відразу автоматично, а при використанні методу `Display()` програма після закриття вікна повинна отримати від нього ці зміни і виконати з ними необхідні дії.

Робота з діалоговими вікнами

- Обидва методи повертають значення `Long`, що визначає, яка кнопка була натиснута у діалоговому вікні:

Значення	Опис
-2	Кнока Закрыть (Close)
-1	Кнока ОК або її еквівалент (наприклад, Открыть у вікні відкриття файлу)
0	Кнопка Отмена (Cancel)
> 0	Інша командна кнопка (1 -перша кнопка, 2- друга кнопка іт.д.)

Наступний приклад демонструє відображення вікна відкриття файлу:

```
Dim dlgOpenFile As Dialog
Set dlgOpenFile = Dialogs(wdDialogFileOpen)
dlgOpenFile.Show
```

Якшо у діалоговому вікні користувач обере файл та виконає клік на кнопці *Открыть*, файл буде відкритий і для нього автоматично створиться новий об'єкт `Document`.

Sub DialogUsing()

Використання інших об’єктів об’єктної моделі Word

- Об'єкт `Options` має велику кількість властивостей, які дозволяють налаштовувати практично будь-які властивості самого додатка Word і поточного документа (доступні командою *Файл-Параметры*), методи у цього об’єкта відсутні. Наприклад, щоб змінити каталог за замовчуванням для збереження документів Word, можна використовувати команду:

```
Options.DefaultFilePath(wdDocumentsPath) =
"C:\Documents"
```

- За допомогою об'єкта `System` можна отримати велику кількість інформації про систему, в якій працює ваш додаток. Часто використовувані властивості об'єкта `System`:
  - `CountryRegion` - повертає поточні регіональні настройки операційної системи. Якщо встановлена англомовна - `1`, якщо російськомовна - `7`.
  - `FreeDiskSpace` - повертає обсяг доступного дискового простору для користувача на поточному диску (можна змінити поточний диск). Якщо документи дуже великі і іноді виникають проблеми з місцем на диску, можна реалізувати перевірку наявності вільного місця, наприклад, при виконанні операцій збереження.

Використання інших об’єктів об’єктної моделі Word

- `HorizontalRezolution` і `VerticalResolution` - можливість отримати інформацію про поточний розмір екрану у користувача, наприклад, для правильного відображення великих форм.
- `LanguageDesignation` - можливість визначити мову інтерфейсу операційної системи. Повертається у вигляді строкового значення, наприклад:

  ```
  Debug.Print System.LanguageDesignation
  ```

- `OperationSystem` - повертає інформацію про операційну систему - "Windows NT" - для сучасних Windows.

- У об'єкта `System` передбачено всього два методи - `Connect(Path, Drive, Password)` - підключити мережевий диск і `MsInfo()` - показати вікно "Відомості про систему" з відповідною інформацією.

Використання інших об’єктів об’єктної моделі Word

- За допомогою об'єкта-колекції `Tasks` можна працювати з задачами - об'єктами `Task`, що запущені у операційній системі. У колекції `Tasks` є корисні методи `Exists(Name)` - що перевіряє, чи запушений застосунок з ім'ям `Name`.
- Метод `ExitWindows()` - дозволяє завершити сеанс роботи в Windows. Незбережені документи Word при цьому закриються без збереження (і без питань до користувача), а документи інших додатків користувачеві буде запропоновано зберегти.

У об'єкта Task цікавих властивостей і методів більше:

- `Height, Width, Top, Left` - ці властивості дозволяють точно налаштувати розмір і розміщення вікна вибраної програми.
- `Visible` - надає можливість сховати додаток.
- `WindowState` - надає можливість розгорнути, згорнути або відновити вікно.
- Призначення методів `Activate(), Close(), Move(), Resize()` об'єкту `Task` очевидне.

Sub SystemUsing()