Ростислав Ямненко

Побудова таблиць тривалості життя в Microsoft Excel із застосуванням Visual Basic for Applications

Київський національний університет імені Тараса Шевченка, 2013

Зміст

Лекція 1. Вступ до Microsoft Visual Basic for Applications

Вступ

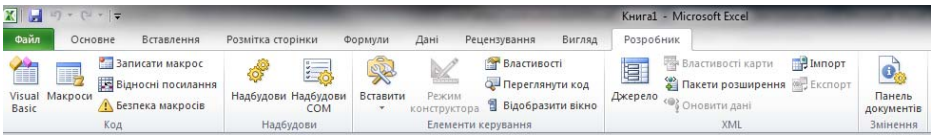
Microsoft Excel – це програма для роботи з електронними таблицями, яку використовують для створення списків, проведення обчислень й аналізування чисел. Вона може бути використана в бізнесі, економіці, бухгалтерському обліку тощо.

Хоча базових характеристик Microsoft Excel вистачає в більшості випадків, іноді потрібна більш складна функціональність для виконання складних операцій. Щоб зробити це можливим, Microsoft Excel супроводжується Microsoft Visual Basic, середовищем програмування, яке дозволяє використовувати мову Visual Basic з метою підвищення корисності і функціональності електронних таблиць.

Microsoft Visual Basic for Applications (VBA) – це мова програмування, заснована на Microsoft Visual Basic. Вона дозволяє писати код, який може автоматично виконувати дії над документом та/чи його змістом. При використанні цієї мови, Ви пишете фрагменти коду, використовуючи зовнішнє середовище.

Запуск VBA

Напевно, найкращий спосіб запускати Microsoft Visual Basic для програмістів полягає в наступному. Натисніть кнопку **Файл,** а потім **Параметри**. У діалоговому вікні **Настроїти стрічку** виберіть прапорець **Розробник** і натисніть кнопку **ОК**. Головна стрічка Excel стане оснащеною новою вкладкою:



Щоб запустити Microsoft Visual Basic із вкладки розробника, натисніть на кнопку **Visual Basic**.

Щоб допомогти з Вашою розробкою, VBA може відображати різні вікна.

Оглядове вікно проекту

У вікні **Project Explorer** наведено перелік сегментів коду, які доступні для робочого аркуша. Воно, як правило, розташоване у верхній лівій частині. У разі відсутності, натисніть кнопку **View -> Project Explorer** у головному меню.

Вікно властивостей

Вікно **Properties** показує характеристики вибраного об'єкта і, як правило, розташоване в нижній лівій частині екрана. Щоб відобразити його в разі відсутності, у головному меню виберіть **View -> Properties Window**.

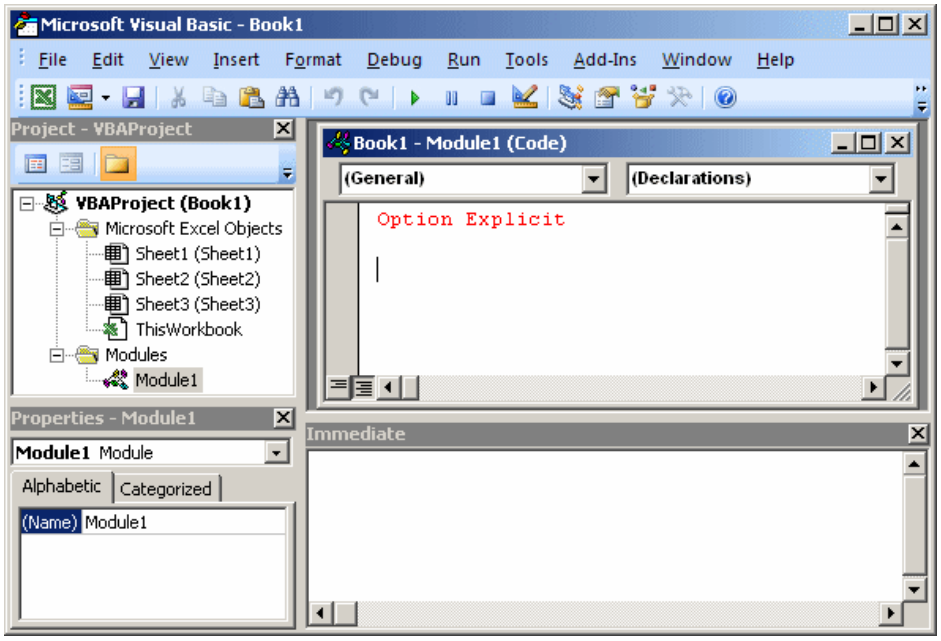
Вікно модулів

Модуль – це вікно, в якому ви пишете код. Коли Ви використовуєте Microsoft Excel для роботи над документом, йому автоматично призначається базовий модуль. Також можна створити модуль, який не залежить від жодного аркуша.

Для створення нового модуля, у головному меню натисніть **Insert -> Module**.

Миттєве вікно

Щоб допомогти тестувати код, VBA надає спеціальне вікно під назвою Immediate. Щоб відобразити його, у головному меню натисніть **View -> Immediate Window**.



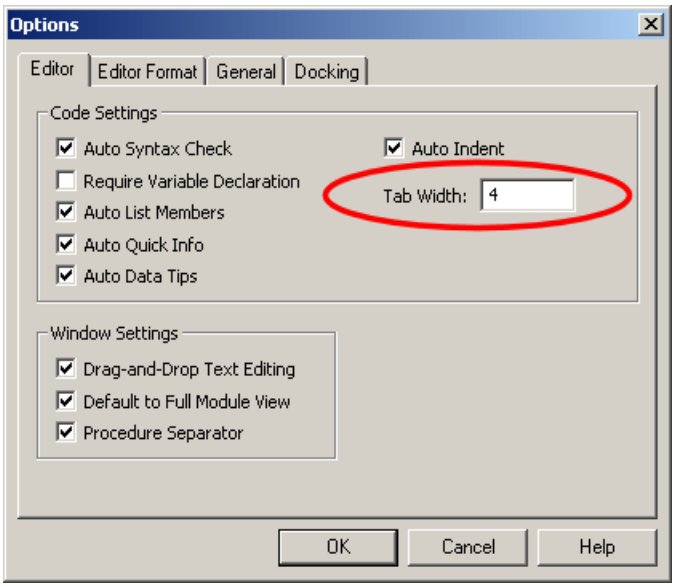
Написання коду

Відступи

Застосування відступів дозволяє писати код, який легко читати. Такий метод полягає у візуальній демонстрації початку й кінця ділянки коду. Кожен відступ – це переміщення коду праворуч.

Найпростіший і найбільш розповсюджений спосіб застосовування відступів полягає в натисканні клавіші Tab перед уведенням коду. За

замовчуванням, один відступ, зроблений при натисканні Tab, відповідає чотирьом символам. Цю величину можна автоматично встановити за допомогою параметру **Tab Width** на сторінці властивостей **Editor** в діалоговому вікні **Tools -> Options**.



Рекомендується використовувати стандартне значення відступу в 4 символи.

Коментарі

Коментар – це шматок тексту в коді, який не враховуватиметься під час інтерпретації коду. Таким чином, у коментарі може бути записано будь-що зручним для Вас способом.

У мові Visual Basic рядок, що містить коментарі, може починатися із ключового слова **Rem**, **rem**, **REM** чи символа **'**. Ось приклад:

```
' Ця лінія не є частиною коду
REM Я можу писати будь-що на цій лінії
```

Коментарі дуже корисні, настійно радимо Вам використовувати їх регулярно.

Змінні

Оголошення змінних

При написанні коду можна використовувати будь-яку змінну, просто вказавши її ім'я. Microsoft Visual Basic дозволяє безпосередньо використовувати будь-яке ім'я для змінної, яке Ви вважаєте за потрібне. Щоб оголосити змінну, треба почати рядок зі слова Dim, наприклад:

```
Dim
```

Кожна змінна повинна мати ім'я. Ім'я пишуть праворуч від слова Dim. Є правила, яких треба дотримуватися при іменуванні змінних:

- ім'я змінної має починатися з букви або символу підкреслення;
- після першої букви або символу підкреслення, ім'я може бути складене з букв, символів підкреслення і цифр у будь-якому порядку;
- ім'я змінної не може містити крапку;
- довжина імені не може перевищувати 255 символів;
- ім'я змінної має бути унікальним в області свого застосування.

Існують деякі слова, які не можна використовувати в якості змінних. Ці слова зарезервовані VBA для внутрішнього вжитку. Тому їх називають ключовими. Ось деякі з них:

And; As; Boolean; ByRef; Byte; ByVal; Call; Case; CBool; CByte; CDate; CDbl; CInt; CLng; Const; CSng; CStr; Date; Dim; Do; Double; Each; Else; ElseIf; End; EndIf; Error; False; For; Function; Get; GoTo; If; Integer; Let; Lib; Long; Loop; Me; Mid; Mod; New; Next; Not; Nothing; Option; Or; Private; Public; ReDim; REM; Resume; Select; Set; Single; Static; Step; String; Sub; Then; To; True; Until; vbCrLf; vbTab; With; While; Xor.

Як уже було сказано, щоб оголосити змінну, напишіть після Dim її ім’я. Наприклад,

```
Sub Exercise()

    Dim VarName

End Sub
```

Оголошення змінної просто передає Visual Basic ім'я цієї змінної. Ви все ще можете використовувати поєднання оголошених і неоголошених змінних. Якщо оголосити одну змінну, а потім почати використовувати іншу змінну з аналогічною, але трохи іншою назвою, Visual Basic буде, як і раніше, вважати, що Ви використовуєте дві змінні. Це може створити велику плутанину, тому що Ви, можливо, намагаєтеся використовувати ту саму змінну, описану двічі. Прикладом вирішення цієї можливої плутанини є вказівка Visual Basic, що змінна не може бути використана, якщо її спершу не було оголошена. Для цього вгорі кожного файлу, який Ви використовуєте в Code Editor, надрукуйте

```
    Option Explicit
```

Це також може бути зроблено автоматично для кожного файлу, за умови наявності відмітки **Require Variable Declaration** у діалоговому вікні **Options**.

Оголошення кількох змінних

Так само, Ви можете оголосити стільки змінних, скільки хочете. Замість того, щоб оголошувати кожну змінну в окремому рядку, можна оголосити кілька змінних в одному рядку. Щоб зробити це, використайте один раз ключове слово Dim і відокремте імена змінних комами. Наприклад,

```
Sub Exercise()

    Dim VarName1, VarName2
    Dim  VarName3, VarName4, VarName5
```

```
End Sub
```

Щоб зберегти значення в пам'яті, зарезервованої для змінної, Ви можете присвоїти змінній значення. Для цього введіть ім'я змінної, за яким слідує оператор присвоювання = та значення, яке ви хочете зберегти. Ось приклад:

```
Sub Exercise()

    Dim Value
    Value = 9374

End Sub
```

Є різні типи значень, які будуть використовуватися в документі. Крім того, значення, повинне відповідати типу пам'яті, що комп'ютер зарезервував для змінної.

Типи даних

Тип даних показує, які змінні Ви збираєтеся використовувати. Перш ніж використовувати змінну, Ви повинні знати, скільки місця вона буде займати в пам'яті. Різні змінні займають різний обсяг у пам'яті. Інформація, яка визначає, скільки місця потребує змінна, називається типом даних. Тип даних вимірюється в байтах.

Щоб вказати тип даних, який буде використовуватися для змінної, після введення Dim та ім'я змінної напишіть ключове слово As, а за ним один з типів даних:

```
Sub Exercise()

    Dim FullName As DataType1, DateHired As DataType2
    Dim EmploymentStatus As DataType3

End Sub
```

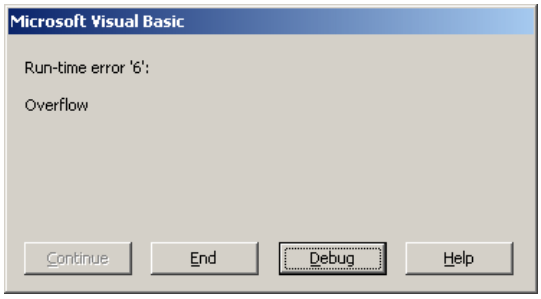
Цілочисельні змінні

Byte

Щоб оголосити змінну, яка буде містити натуральні числа в діапазоні від 0 до 255, використовуйте тип даних Byte.

```
Sub Exercise()

    Dim Value As Byte
    Value = 246

End Sub
```

Якщо ви задасте від'ємне або значення вище 255, то при спробі доступу до нього отримаєте повідомлення про помилку:



Integer

Щоб оголосити змінну, яка зберігатиме число в діапазоні від -32768 до 32767, використовуйте тип даних Integer.

```
Sub Exercise()

    Dim aVariable1 As Integer

End Sub
```

Замість запису As Integer, Ви можете використовувати символ %. Таким чином, декларація вище може бути зроблена таким чином:

```
Sub Exercise()
```

```
    Dim aVariable1%

End Sub
```

Long

Змінна типу Long може зберігати значення між – 2 147 483 648 та 2 147 483 647. Символ типу Long – це @.

```
Sub Exercise()

    Dim LongVariable@

End Sub
```

Для того, щоб конвертувати значення у тип Byte, Integer чи Long, використовуйте функції CByte(), CInt() та CLng() відповідно.

Дробові змінні

Якщо ви плануєте використовувати змінну, яка буде дробового типу, але точність не є Вашою головною турботою, оголосіть її за допомогою типу даних Single. Змінна цього типу може зберігати значення між $1.401298e^{-45}$ та $3.402823e^{38}$. Символом типу Single є !.

Якщо ви хочете використовувати десятковий дріб із великою точністю, оголосіть змінну за допомогою типу даних Double. Змінна цього типу може зберігати значення між $–1.79769313486231e^{308}$ та $–4.94065645841247e^{-324}$ для від'ємних значень і між $4.94065645841247e^{-324}$ та $1.79769313486231e^{308}$ для додатних. Замість запису As Double, Ви можете використовувати символ #.

```
Sub Exercise()

Dim aSingleVar1 As Single
Dim aSingleVar2!

Dim aDoubleVar1 As Double
Dim aDoubleVar2#
```

```
End Sub
```

Для конвертації значень у типи Single та Double використовують функції CSng() та CDbl().

Рядковий тип String

Рядок – це символ або комбінація символів, які є текстом будь-якого виду і практично будь-якої довжини. Щоб оголосити рядкову змінну, використовуйте тип даних String. Символом, що позначає цей тип, є $. Значення рядкової змінної повинне бути всередині подвійних лапок.

```
Sub Exercise()

Dim interest_name As String
Dim discount_name$

   interest_name = "Відсоткова ставка"
   discount_name = "Дисконт"

End Sub
```

Якщо Ви хочете перетворити значення, яке початково не було текстом, у рядок, використовуйте функцію CStr ().

Грошовий тип даних Currency

Тип даних Currency використовують, коли мають справу з грошовими значеннями.

Лекція 2. Листи Excel

Ідентифікація (визначення) листа

Лист – це документ в Microsoft Excel. Лист – це об'єкт, що створюється в робочій книзі (workbook). Таким чином робоча книга – це множина листів, що сприймається як окрема група.

Лист – це об'єкт типу **Worksheet**. Різні листи,які ви будете використовувати зберігаються в наборі **Worksheets**. Інша назва набору, що містить листи книги - **Sheets**. В більшості випадків, ви можете використовувати для посилання обидві назви.

Посилання на листи

У попередньому занятті ми бачили, що, якщо у вас є тільки одна відкрита книга, то щоб звернутися до неї вам потрібно вказати аргумент 1 до властивості **Item** елемента набору **Workbooks**. Нижче приклад:

```
Workbooks.Item(1).Activate
```

Ви можете опустити слово Item і отримаєте той самий результат:

```
Workbooks(1).Activate
```

Так як листи є частиною документа – відкритої книги, щоб посилатися на них клас **Workbook** має властивість, що називається **Worksheets** чи **Sheets**. Тому, після визначення робочої книги, потрібно використати властивість Worksheets (Sheets). Приклад нижче:

```
Workbooks.Item(1).Sheets
```

Як вже сказано раніше, листи зберігаються в наборі **Worksheets** , який насправді є класом. Розміщення кожного листа задається властивістю, що називається **Item**. Властивість **Item** по замовчанню задається натуральними числами починаючи з 1. Перший лист зліва має назву (**Item**) 1. Наступний має назву 2, і так далі. Щоб отримати доступ до листа, потрібно використати один з операторів **Worksheets** чи **Sheets**, за ним

визначити **Item()** і в круглих дужках вказати назву (номер) бажаного листа. Наприклад, код нижче показує посилання на другий зліва лист:

```
Workbooks.Item(1).Sheets.Item(2)
```

Вище було показано можливість опустити слово **Item** в об'єкті **Workbooks**, те ж саме можна зробити і в об'єкті **Worksheets** (**Sheets)**. Приклад, який вказує на активацію другого листа, наведено нижче:

```
Sub Exercise()
   Workbooks.Item(1).Sheets(2). Activate
End Sub
```

Або так:

```
Sub Exercise()
   Workbooks(1).Worksheets(2). Activate
End Sub
```

Кожен лист має також своє ім'я. По замовчанню, крайній лист зліва має назву Аркуш1. The Другий зліва – Аркуш2. Щоб зробити посилання на лист за допомогою його назви, потрібно в об'єкті **Worksheets** (**Sheets)** вказати назву листа як рядок. Приклад нижче показує посилання на лист з назвою Аркуш3:

```
Workbooks.Item(1).Sheets.Item("Аркуш3")
```

Вище ми завжди робили посилання на листи поточної (відкритої) книги. Як вже було сказано, за замовчанням, коли запускається Microsoft Excel, створюється книга по замовчанню і посилання на**Workbooks.Item(1)**. Це означає, що вам не потрібно додатково нічого писати для роботи в поточній книзі – ви вже знаходитесь в ній. Звідси слідує можливість опустити в вашому коді **Workbooks.Item(1)** чи **Workbooks(1)**. Наприклад:

```
Sheets.Item("Аркуш3")
```

Створення посилань на листи

В прикладах коду вище, ми робили припущення, що ви хочете зробити якісь дії на поточному листі і на цьому зупинитись. Іноді вам потрібно буде зробити посилання на лист. Щоб це зробити, треба визначити змінну типу Worksheet. Щоб ідентифікувати потрібний лист робочої книги потрібно використати властивість **Item** та оператор **Set**. Нижче приклад, який показує посилання на другий лист відкритої в даний момент книги і зберігає це посилання в змінній:

```
Sub Exercise()
    Dim Second, Third As Worksheet
    Set Second = Workbooks.Item(1).Sheets.Item(2)
    Second.Activate

  Set Third = Workbooks.Item(1).Sheets.Item("Аркуш3")
  Third.Activate
End Sub
```

Виділення листа

Для виділення листа потрібно прописати в **Sheets** ім’я листа у вигляді рядка та викликати **Select**. Приклад нижче показує виділення листа з назвою **Sheet1**:

```
    Sheets("Аркуш1").Select
```

Виділений лист або лист, з яким ви зараз працюєте називається активним листом. Він визначається як об’єкт **ActiveSheet** (насправді це властивість поточного документа).

Імена листів

Щоб перейменувати лист, потрібно вказати як рядок його поточне ім'я в **Sheets** (**Worksheets**), а потім використати властивість **Name** щоб присвоїти нове ім'я. Наприклад:

```
Private Sub Exercise()

   Sheets("Аркуш1").Name = "Комутаційні функції"

End Sub
```

Код вище змінить назву листа «Аркуш1» на «Комутаційні функції».

Як ми бачили раніше, ви можете виділити лист по його назві. Після того, як ви змініте назву листа, ви можете використовувати її для виділення листа. В прикладі виділяється лист з назвою «Ануїтети»:

```
Private Sub Exercise()

   Sheets("Ануїтети").Select

End Sub
```

Робота з групою листів

Фіксація комірки або кількох стовпців

Ви можете використати стовбець для фіксації його комірок. Щоб зафіксувати або навпаки комірку, використовується об'єкт **ActiveWindow**, а точніше його властивість **FreezePanes** , що є булевою. Якщо вона є істинною, то вікно ділиться на чотири частини спираючись на виділену комірку, або комірку, що є активною зараз. Приклад нижче:

```
Sub Freezing()

   ActiveWindow.FreezePanes = True

End Sub
```

Розділення листа

Щоб розділити лист, використовується об’єкт **ActiveWindow** та його булева властивість **Split**. Для розділення їй треба присвоїти значення істина:

```
Sub Splitting()

   ActiveWindow.Split = True

End Sub
```

І для оберненої операції – значення False.

Послідовність листів

Щоб змістити лист, використовуйте метод **Move()** в **Worksheets** (**Sheets)**. Синтаксис цього методу описується прикладом:

```
Worksheets(Index).Move(Before, After)
```

Обидва аргументи є необов'язковими. Якщо ви не визначите жоден з аргументів, Microsoft Visual Basic створить нову книгу з копією листа. Припустимо, що ви працюєте з книгою, в якій є листи Sheet1, Sheet2, and Sheet3. Якщо ж ви застосуєте даний метод до одного з листів, Microsoft Excel створить копію цього листа, створить нову книгу з одним листом, що є копією даного листа. Наприклад, слідуючий код створює нову книгу, що містить лист Sheet2 відкритої книги:

```
Private Sub CommandButton1_Click()

   Sheets.Item("Аркуш2").Move

End Sub
```

В цьому випадку, обов’язково потрібно вказати назву листа, який ви переміщаєте. Інакше ви отримаєте повідомлення про помилку. Замість використання назви листа можна використати номер листа, який ви

хочете скопіювати. Наприклад, наступний код створить нову книгу , що буде містити 1 лист з ім'ям Sheet3:

```
Private Sub CommandButton1_Click()
   Sheets.Item(3).Move
End Sub
```

Викликаючи властивість **Item**, переконайтесь в існуванні вказаного номеру, щоб не отримати повідомлення про помилку.

Для того щоб перемістити лист в межах однієї книги, ви маєте вказати лист, відносно якого переміщений лист буде лежати зліва чи справа. Щоб розмістити переміщуваний лист зліва деякого листа ви повинні вказати його в аргументі *Before*. Для розміщення зліва від листа, ви маєте вказати його в аргументі *After*. Розглянемо слідуючий шматок коду:

```
Private Sub cmdMove_Click()
   Worksheets("Аркуш3").Move After:=Worksheets("Аркуш1")
End Sub
```

Цей код задає переміщення листа Аркуш3 в місце справа від листа Аркуш1.

Додавання нових листів

При створенні нового листа, ви можете вказати його положення (перед чи після) стосовно вже існуючого листа. Для створення нового листа використовуйте метод **Add()** набору **Worksheets** (**Sheets)**. Його синтаксис нижче:

```
Workbook.Sheets.Add(Before, After, Count, Type)
```

Всі ці аргументи необов'язкові. Тобто можна описати його таким чином:

```
Private Sub cmdNewWorksheet_Click()
```

```
    Sheets.Add
End Sub
```

При описанні методу так, новий лист буде створено (додано) зліва від активного листа:

Якщо ви хочете створити новий лист зліва від деякого листа, ви можете спочатку виділити цей лист, а потім використати метод **Add()**. Наприклад, припустимо ви маєте три листи Аркуш1, Аркуш2 та Аркуш3 зліва направо і ви хочете вставити новий лист між Аркуш2 та Аркуш3, тоді це можна зробити за допомогою коду нижче:

```
Private Sub cmdNewWorksheet_Click()
    Sheets("Аркуш2").Select
    Sheets.Add
End Sub
```

Щоб бути більш точними, ви можете вказати місцезнаходження нового листа – зліва чи справа від листа, який ви виділили.

Видалення листів

Щоб видалити лист, використовується метод **Delete()**. При використанні методу треба вказати лист, який ви хочете видалити з робочої книги.

Доступ до листа

Щоб отримати доступ до листа, клас **Worksheet** має метод, що називається **Activate**. Його синтаксис такий:

```
Worksheet.Activate()
```

Аргументів не вимагається. Щоб використати його, зробіть посилання на потрібний вам лист та викличте метод **Activate()**. Ви можете зменшити кількість коду, якщо додасте індекс одразу після **Worksheets**. Нижче приклад:

```
Private Sub cmdSelectWorkbook_Click()

    Worksheets(2).Activate

End Sub
```

Стовпці листа

Базові поняття про стовпці

Аркуші поділені на стовпці. В мові VBA для Microsoft Excel для посилання на стовбець, ви будете використовувати різні класи. Один з класів, який ви будете використовувати для доступу до стовпця, - Range. Як ми побачимо в багатьох прикладах, ви зможете безпосередньо її використовувати.

Якщо ви хочете отримати посилання на стовпець або групу стовпців, треба визначити змінну типу **Range**:

```
Sub Exercise()

    Dim Series As Range

End Sub
```

До ініціалізації змінної ви повинні ідентифікувати (визначити) книги і листи, які використовуєте.

Клас Columns

Коли запускається Microsoft Excel, автоматично відображаються стопці, що вже були створені. Для того, щоб це відбулося, клас **Worksheet** має підклас **Columns**. Існують різні способи, якими ви можете визначити стовпці: використовуючи його індекс (номер) або ім’я.

Визначення стовпця

Стовпець по номеру

Стовпці на аркуші розташовані по порядку. Тобто позиція стовпця визначається його номером. Перший стовбець зліва має номер 1, другий

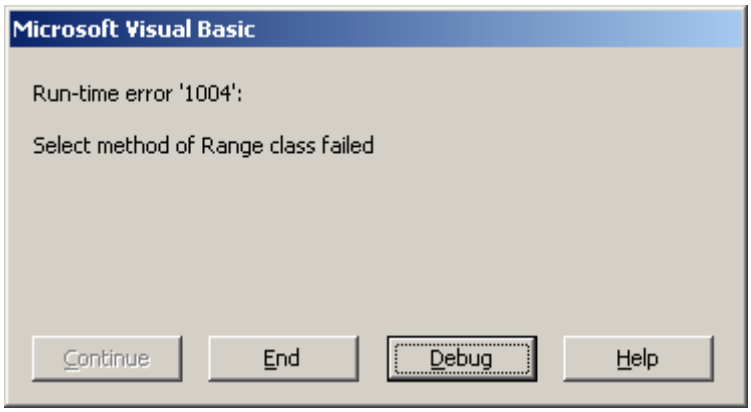
стовбець зліва має номер 2, і так далі. Знаючи це, посилаючись на стовбець, пишіть номер стовпця в круглих дужках після Columns. Нижче наведено приклади:

```
Workbooks(1).Worksheets(2).Columns(1)

Workbooks(1).Worksheets(2).Columns(12)
```

В попередньому уроці, ми бачили, що можна опускати визначення **Workbooks(1)** щоб працювати у книзі, що визначена за замовчуванням. Тобто, код вище може бути записаний таким чином:

```
Worksheets(2).Columns(4)
```

Цей код означає, що тепер ви маєте на увазі четвертий стовбець на другому аркуші. Якщо виконати його, Microsoft Excel має відображати другий лист. Якщо ви запустите цей код коли Microsoft Excel відображає будь-який інший лист, то побачите повідомлення про помилку:



Це означає, що якщо ви будете посилатися на стовбець з іншого листа, ніж визначений в вашому коді (в тому числі по замовчанню) , код не спрацює. Якщо ви хочете отримати доступ до певного стовпця в листі з книги Microsoft Excel, що в даний момент відображається, можна опустити визначення аркуша за допомогою Worksheets. Нижче приклад:

```
Columns(4)
```

На цей раз, код означає, що ви маєте на увазі четвертий стовпець аркуша, що відображається в даний момент.

Іменне посилання на стовпець

Для того щоб зробити посилання на стовбець за його ім'ям, треба ввести його назву (літеру чи кілька літер) у круглих дужках після Columns. Нижче наведені приклади:

```
Columns("A")
```

```
Columns("Рік")
```

Сусідні стовпці (діапазон стовпців)

Щоб зробити посилання на сусідні стовпці, також можна використовувати **Columns**. В круглих дужках треба вказати ім'я стовпця, що знаходиться з одного кінця діапазону за яким слідує двокрапка, а потім ім'я стовпця з іншого кінця діапазону. Нижче наведено приклад посилання на діапазон рядків від D до G:

```
Columns("D:G")
```

Також ви можете визначити діапазон за допомогою класу Range. Щоб зробити це, треба аналогічно вказати ім'я стовпця, що знаходиться з одного кінця діапазону за яким слідує двокрапка, а потім ім'я стовпця з іншого кінця діапазону. Приклад використання нижче:

```
Range("D:H")
```

Несуміжні стовпці

Будемо називати несуміжними стовпці, що не слідують безпосередньо один за одним. Наприклад, стопці B, D, та G є несуміжними. Для посилання на несуміжні стовпці також треба використовувати **Range**. В круглих дужках треба написати назву кожного стовпця, за якою слідує двокрапка і знову та ж назва, відділяючи кожну таку комбінацію від наступної комами. Нижче приклад:

```
Range("H:H, D:D, B:B")
```

Щоб визначити всі стовпці листа, використовуйте **Columns** без аргументів, як у прикладі нижче:

```
Columns
```

Вибір стовпців

Вибір стовпця

Для вибору стовпця, клас Column оснащений методом з ім'ям SELECT. Цей метод не має аргументів. Керуючись цим, щоб вибрати четвертий стовбець по його номеру, треба написати наступний код:

```
Sub Exercise()
    Rem Вибір четвертого стовпця
    Columns(4).Select
End Sub
```

Аналогічно, щоб вибрати стовбець по імені:

```
Sub Exercise()
    Rem Вибір стовпця з іменем «Відсоткова ставка»
    Columns("Відсоткова ставка").Select
End Sub
```

Після того як стовпець було обрано, він зберігається в об'єкті, що називається Selection. Ви можете використовувати цей об'єкт для виконання будь-яких дій над стовпцем.

Вибір діапазону сусідніх стовпців

Щоб вибрати діапазон стовпців, треба вказати в круглих дужках після **Columns** ім'я першого стовпця, потім ":", а потім ім'я останнього стовпця діапазону. Приклад нижче:

```
Sub Exercise()

   Rem Вибір діапазону від стовпця Column D до стовпця Column G

   Columns("D:G").Select

End Sub
```

Тими ж командами можна виділити один стовпець. Для цього в круглих дужках після **Range** треба ввести ім’я стовпця, за ним двокрапку, і знову ім’я стовпця. Нижче приклад:

```
Sub Exercise()

   Range("G:G").Select

End Sub
```

Вибір несуміжних стовпців

Щоб вибрати несуміжні стовпці потрібно використовувати аналогічний спосіб, визначений вище, а потім додавати **Select**. Приклад нижче:

```
Sub Exercise()

   Range("H:H, D:D, B:B").Select

End Sub
```

Коли виділено багато стовпців (неважливо суміжних чи несуміжних), всі вони будуть зберігатися в об’єкті **Selection**. Маючи доступ до цього об’єкту ви можете виконати однакові дії з всіма виділеними стовпцями.

Створення стовпців

Додавання нового стовпця

Для створення стовпців клас **Column** має метод, що називається **Insert**. Цей метод, як і попередній, також не потребує додаткових аргументів. Використовуючи його, необхідно вказати стовпець, який буде замінено новостворениим. В прикладі нижче на третій позиції створюється новий стовпець,а стовпці з 3 по 16384 зміщуються вправо:

```
Sub CreateColumn()

   Columns(3).Insert

End Sub
```

Додавання нових стовпців

Щоб додати нові стовпці, треба використати **Range**. Після нього використати метод **Insert** класу **Column**. Приклад нижче показує створення нових стовпців на місцях стовпців B, D, та H, що зміщуються вправо для звільнення місця для нових:

```
Sub CreateColumns()

   Range("H:H, D:D, B:B").Insert

End Sub
```

Видалення стовпців

Видалення стовпця

```
Sub DeleteColumn()

   Columns("D:D").Delete

End Sub
```

Видалення множини стовпців

Для видалення суміжних стовпців треба використати **Columns** як було показано раніше і потім додати метод Delete. Приклад нижче:

```
Sub DeleteColumns()

   Columns("D:F").Delete

End Sub
```

Для видалення множини несуміжних стовпців, також використовується **Range,** потім викликається Delete. В прикладі нижче показано видалення стовпців C, E, та P:

```
Sub DeleteColumns()

   Range("C:C, E:E, P:P").Delete

End Sub
```

Ширина стовпців

Для визначення розмірів стовпців, **Column** має властивість, що називається **ColumnWidth**. Тому, щоб в коді визначити ширину стовпця, треба визначити стовбець, потім його властивість **ColumnWidth** і присвоїти їй потрібне значення. В прикладі нижче стовпцю C присвоюється ширина 4.50:

```
Sub Exercise()

   Columns("C").ColumnWidth = 4.5

End Sub
```

Автоматична зміна розміру

Щоб задати автоматичну підгонку ширини, спочатку треба визначити стовпці командами **Selection** та **Columns**, потім викликати метод **AutoFit**. Це можна зробити як у прикладі нижче:

```
   Selection.Columns.AutoFit
```

Самостійне встановлення ширини стовпців

Для визначення ширини множини стовпців, треба виділити їх за допомогою **Range**, потім у властивості **ColumnWidth** присвоїти ширині

необхідне значення. У прикладі далі кожному зі стовпців C, E, та H присвоюється ширина 5:

```
Sub Exercise()
    Range("C:C, E:E, H:H").ColumnWidth = 5#
End Sub
```

Приховування, фіксація, розділення стовпців

Приховування та виявлення стовпців

Щоб зробити стовпець прихованим, спочатку треба його виділити, потім встановити значення **True** для властивості **Hidden** об'єкта **EntireColumn** об'єкта **Selection**. Приклад використання нижче:

```
Private Sub Exercise()
    Columns("F:F").Select
    Selection.EntireColumn.Hidden = True
End Sub
```

Щоб прихований стовбець знову відображався, треба присвоїти значення **False** властивості **Hidden**:

```
Private Sub Exercise()
    Columns("F:F").Select
    Selection.EntireColumn.Hidden = False
End Sub
```

Розділення стовпців

Щоб розділити стовпці, треба визначити об'єкт **ActiveWindow**, визначити властивість **SplitColumn**, в якій треба вказати номер стовпця. Нижче приклад:

```
Sub Exercise()

    ActiveWindow.SplitColumn = 4

End Sub
```

Щоб скасувати розділення, треба визначити об'єкт **ActiveWindow** і в властивісті **SplitColumn** прописати значення 0. Приклад нижче:

```
Sub Exercise()

    ActiveWindow.SplitColumn = 0

End Sub
```

Рядки листа

Ми вже знаємо, що інформація на аркуші організовується по стовпчиках. Ми показали, що кожен стовпець має особливе значення, але значення можуть відрізнятися і по горизонталі. Група значень, які відповідають тим же горизонтальним розташуванням, називається рядком.

Щоб визначати рядки на листі, клас **Worksheet** має підклас **Rows**. Тому, щоб визначити рядок, ви маєте по аналогії з визначенням стовпців використати **Worksheets** та підклас **Rows**. Інший спосіб – використати об'єкт **Range**.

Для ідентифікації рядка, треба визначити лист, а потім вказати номер рядка в круглих дужках **Rows**. В прикладі нижче вибирається 5-й рядок 2-го листа в поточній книзі:

```
    Workbooks.Item(1).Worksheets.Item(2).Rows(5)
```

Як вже показувалося для стовпців, цей код буде працювати тільки якщо 2-ий лист поточної книги зараз активний. Ви побачите помилку, якщо активний будь-який інший лист. Щоб отримати будь-який рядок, пропускайте опис **Workbooks** та **Worksheets**.

Як вже згадувалось вище, для опису рядків можна також використовувати **Range**. Для цього треба описати рядок в об'єкті **Range**. В круглих дужках пишеться номер рядка, потім двокрапка і знову номер рядка. В прикладі нижче описується посилання на 4-ий рядок:

```
Range("4:4")
```

Якщо ви хочете посилатися на рядок більше одного разу, можна визначити змінну типу Range та присвоїти їй значення, використовуючи оператор. Нижче приклад:

```
Sub Exercise()

    Dim SeriesOfRows As Range

    Set SeriesOfRows =
Workbooks.Item(1).Worksheets.Item("Аркуш1").Range("4:4")

End Sub
```

Визначення групи рядків

Група рядків називається діапазоном, якщо вони знаходяться поруч один з одним. Щоб визначити діапазон рядків, потрібно в круглих дужках **Rows** вказати номер верхнього рядка, за яким написати двокрапку та номер останнього нижнього рядка. В прикладі нижче береться діапазон рядків від 2 до 6:

```
Rows("2:6")
```

Група рядків визначається як група несуміжних рядків, якщо рядки в ній не знаходяться поруч один з одним. Щоб визначити групу несуміжних рядків, знову використовуємо **Range**. В круглих дужках вказуємо кожний номер рядка з наступною двокрапкою і знову тим самим номером рядка. Ці комбінації розділяються комами. В прикладі нижче виділяються рядки 3, 5, та 8:

```
Range("3:3, 5:5, 8:8")
```

Щоб визначити всі рядки листа, використовується Rows без аргументів. Нижче приклад:

```
Rows
```

Вибір рядків

Вибір рядка

Для можливості вибору рядка, клас **Row** має метод **Select**. Тому, щоб за допомогою коду вибрати рядок, треба прописати всі описані вище характеристики для **Rows**. А потім використати метод **Select**. Далі приклад виділення 6-го рядка:

```
Sub Exercise()

    Rows(6).Select

End Sub
```

Також бачимо, що вибір можна зробити за допомогою об’єкта Range. Тобто після визначення рядка цим об’єктом, викликати метод Select. У такому прикладі виділяється рядок 4:

```
Sub Exercise()

    Range("4:4").Select

End Sub
```

Після того, як рядок було обрано, він зберігається в об'єкті, що називається Selection. Ви можете використовувати цей об'єкт для виконання будь-яких дій над рядком.

Виділення групи рядків

Щоб виділити діапазон рядків запрограмуйте звернення до цього діапазону, як описувалось раніше, а потім використовуйте метод **Select**. Приклад нижче показує виділення діапазону рядків від 2 до 6:

```
Sub Exercise()

    Rows("2:6").Select

End Sub
```

Щоб виділити множину несуміжних рядків, зверніться до них способом, що вже описувався вище і також використайте метод **Select**. Приклад нижче позує виділення рядків 3, 5, та 8:

```
Sub Exercise()

    Range("3:3, 5:5, 8:8").Select

End Sub
```

Щоб виділити в за допомогою коду всі рядки листа, використайте метод **Select** без аргументів. Наприклад:

```
Sub Exercise()

    Rows.Select

End Sub
```

При виділенні багатьох рядків (неважливо суміжних чи несуміжних), всі вони будуть зберігатися в об’єкті **Selection**. Маючи доступ до цього об’єкту ви можете виконати однакові дії з всіма виділеними рядками.

Управління рядками

Висота рядка

Для вибору висоти рядка об’єкт Row має властивість, що називається **RowHeight**. Тому, для того щоб в коді визначити бажану висоту рядка, треба зробити посилання на рядок одним із описаних вище способів, а потім присвоїти властивості **RowHeight** потрібне значення. Приклад нижче показує присвоєння 6-му рядку висоти 2.50

```
Sub Exercise()
```

```
    Rows(6).RowHeight = 2.5
End Sub
```

Додавання нового рядка

Для того щоб була можливість додавати новий рядок клас **Row** має метод **Insert**. Тому для того, щоб програмно додати новий рядок, треба вказати номер рядка, що буде лежати нижче новоствореного і використати метод **Insert**. Приклад додавання рядка:

```
Sub Exercise()
    Rows(3).Insert
End Sub
```

Adding New Rows

Для додавання множини нових рядків,треба зробити посилання на рядки, що будуть знаходитися нижче нових рядків та використати метод Insert. У прикладі нижче додаються рядки на позиції 3, 6 та 10:

```
Sub Exercise()
    Range("3:3, 6:6, 10:10").Insert
End Sub
```

Видалення рядків

Видалення рядка

Для можливості видалення рядка, клас **Row** має метод **Delete**, з яким не треба використовувати аргументи. Таким чином, щоб видалити рядок, треба зробити посилання на нього і викликати метод **Delete**. Приклад видалення:

```
Sub Exercise()
    Rows(3).Delete
```

```
End Sub
```

Звісно, для посилання ви можете використовувати як об’єкт Rows так Range.

Видалення множини рядків

Щоб видалити групу рядків, потрібно визначити їх за допомогою **Range**. Потім викликати метод **Delete**. Приклад нижче:

```
Sub Exercise()

    Range("3:3, 6:6, 10:10").Delete

End Sub
```

Операції з рядками

Переміщення рядків

Для того щоб перемістити групу рядків, використайте **Range** щоб їх ідентифікувати. Потім використовуйте метод **Cut**. Пропишіть в аргументі **Destination** нове положення рядків (місце, куди ви їх переміщуєте). Нижче приклад:

```
Sub Exercise()

    Rows("11:12").Cut Destination:=Rows("16:17")

End Sub
```

Копіювання і вставка рядків

Для того щоб скопіювати рядок (чи групу рядків) визначте їх за допомогою **Rows**. Потім використайте метод Copy. В аргументі Destination пропишіть номери рядків, в які ви хочете їх скопіювати. Приклад нижче:

```
Sub Exercise()

    Rows("10:15").Copy Destination:=Rows("22:27")
```

```
End Sub
```

Приховування і фіксація рядків

Щоб зробити рядок прихованим, Для початку його треба виділити. Потім, використати властивість **Hidden** об'єкту **EntireRow** об'єкту **Selection**. Приклад нижче:

```
Private Sub Exercise()
   Rows("6:6").Select
   Selection.EntireRow.Hidden = True
End Sub
```

Пиклад коду вище зробить прихованим рядок 6. В той же спосіб можна приховати і множину рядків, спочатку зробивши на них посилання, як було описано вище, потім прописати **Selection.EntireRow.Hidden = True**.

Розділення рядків

Щоб розділити рядки, треба викликати об'єкт **ActiveWindow**, а потім в його властивості **SplitRow** вказати номер рядка. Приклад нижче:

```
Sub Exercise()
   ActiveWindow.SplitRow = 4
End Sub
```

Щоб відмінити розділення треба використати ту ж властивість **ActiveWindow**, але присвоїти їй значення 0. Приклад нижче:

```
Sub Exercise()
   ActiveWindow.SplitRow = 0
End Sub
```

Комірка робочого листа

Таблиця – це послідовність стовпчиків та рядків. Ці стовпчики та рядки перетинаються і утворюють комірки.

Коли Microsoft Excel створює новий робочий лист, на ньому утворюється 16 384 стовпчиків та 1 048 576 рядків. Як результат таблиця в Microsoft Excel має 16 384*1 048 576 = 17 179 869 184 комірок.

Активація комірки

Щоб отримати доступ до комірки, Ви маєте натиснути на неї. Комірка, на яку натиснули, стає активною коміркою. У VBA активна комірка представляється як об’єкт під назвою **ActiveCell.**

Прив'язка комірки

Щоб ідентифікувати комірку, можна використовувати об’єкт **Range.** В круглих дужках об’єкту **Range** вказують ім’я комірки. Наведемо приклад як посилатись на комірку D6:

```
Workbooks.Item(1).Worksheets.Item("Sheet1").Range("D6")
```

Щоб отримати посилання на комірку, необхідно оголосити змінну типу Range. Для ініціалізації змінної, необхідно визначити комірку і привласнити їй змінну за допомогою оператора Set. Наведемо приклад:

```
Sub Exercise()

   Dim Cell As Range

   Set Cell = Workbooks.Item(1).Worksheets.Item("Sheet1").Range("D6")

End Sub
```

Комірки називаються сусідніми, якщо вони торкаються одна одну. Для посилання на групу сусідніх комірок, в дужках об'єкту Range, необхідно визначити комірку, яка буде з одного боку, потім поставити двокрапку, а після неї визначити комірку, яка повинна бути з іншого боку. Наведемо приклад:

```
Range("B2:H6")
```

Цю техніку можна використовувати, щоб визначити одну комірку. Щоб це зробити, використовують ту саму адресу комірки на обох сторонах (зліва і справа) відносно двокрапки. Наприклад:

```
Range("D4:D4")
```

Замість того, щоб посилатись на одну групу сусідніх комірок, можна посилатись на більш ніж одну групу несусідніх комірок. Для того, щоб це зробити, використовують об'єкт **Range.** В дужках записують кожен потрібний діапазон, але розділяють їх за допомогою коми. Наприклад:

```
Range("D2:B5, F8:I14")
```

Вибір комірок

Вибір комірки

Перед тим, як виконати якусь дію з групою комірок, спершу Ви маєте відібрати її. Для підтримки вибору комірки об'єкт **Range** оснащений методом під назвою **Select**. Проте, для того,щоб відібрати за допомогою програми комірку, після посилання на неї, необхідно використати **Select** метод. Наприклад:

```
Sub Exercise()
    Range("D6").Select
End Sub
```

Коли комірка відібрана, вона зберігається в об'єкт під назвою **Selection**. Ви можете використовувати цей об'єкт, щоб виконувати дії з коміркою, яка вибрана в даний момент.

Вибір комірок

Щоб відібрати за допомогою програми групу сусідніх комірок, спочатку необхідно посилатись на дану групу, використовуючи техніки, які розглядались раніше, а потім використати **Select** метод.

Щоб вибрати за допомогою програми всі комірки з даного стовпчика, використовують відбір **Columns,** і ім’я стовпчика вказують типу string, а потім викликають **Select** метод. Наведемо приклад, який ми розглядали на дев’ятому уроці:

```
Sub Exercise()

   Rem This selects all cells from the fourth column

   Columns(4).Select

End Sub
```

Щоб виконати цю операцію, використовуючи ім'я колонки, необхідно передати це ім’я як аргумент. Наведемо приклад, в якому відбираються всі комірки з стовпчика ADH:

```
Sub Exercise()

   Rem This selects all cells from the column labeled ADH

   Columns("ADH").Select

End Sub
```

Також цю операцію можна виконати з використанням **Range** об'єкта. Щоб це зробити, використовують **Range** збірку. В круглих дужках збірки вводимо ім’я стовпчика, після якого слідує двокрапка, а потім вводимо те саме ім’я стопвчика. Наприклад:

```
Sub Exercise()
```

```
    Rem This selects all cells from Column G

    Range("G:G").Select

End Sub
```

Щоб за допомогою програми відібрати всі комірки, що належать до групи сусідніх стовпчиків, у круглих дужках збірки **Columns**, вводимо ім'я першого стовпчика, потім ставимо двокрапку, а потім вводимо ім'я стовпчика, який буде останнім. Наприклад:

```
Sub Exercise()

    Rem This selects all cells in the range of columns from Column D to Column
G

    Columns("D:G").Select

End Sub
```

Щоб відібрати комірки, що належать до групи несусідніх стовпчиків, використовують техніку, щоб посилатись на несусідні стовпчики, яку ми розглядали раніше, а потім викликають **Select** метод. Наприклад:

```
Sub Exercise()

    Rem This selects the cells from columns B, D, and H

    Range("H:H, D:D, B:B").Select

End Sub
```

Щоб вибрати за допомогою програми комірки, що належать до одного рядку, вибирають рядок зі збірки **Rows**, потім викликають **Select** метод. Наведемо приклад, в якому відбираються всі комірки з шостого рядку:

```
Sub Exercise()

    Rows(6).Select
```

```
End Sub
```

Також можна використовувати **Range** об’єкт. Після вибору рядку, викликаємо метод **Select**. Наведемо приклад, в якому відбираються всі комірки з четвертого рядку:

```
Sub Exercise()

   Range("4:4").Select

End Sub
```

Щоб відібрати всі комірки, що належать до діапазону рядків, вказують діапазон, а потім викликають метод **Select**. Наведемо приклад, в якому відбираються всі комірки з другого по шостий рядки:

```
Sub Exercise()

   Rows("2:6").Select

End Sub
```

Щоб відібрати всі комірки, що належать до несусідніх рядків, вказують рядки, а потім викликають метод **Select**. Наведемо приклад, в якому відбираються всі комірки з третього, п’ятого та восьмого рядків:

```
Sub Exercise()

   Range("3:3, 5:5, 8:8").Select

End Sub
```

Щоб відібрати за допомогою програми комірки з даної області, вказують їх ранг за допомогою типу string до об’єкту Range, потім викликають метод **Select**. Наприклад:

```
Sub Exercise()

   Range("B2:H6").Select
```

```
End Sub
```

Нагадаємо, що можна використовувати таку ж техніку лише для однієї комірки, тобто для того, щоб відібрати одну лише комірку. Наприклад:

```
Sub Exercise()

    Range("D2:B5, F8:I14").Select

End Sub
```

Щоб відібрати всі комірки з таблиці, модна викликати **Select** метод для **Rows**. Наприклад:

```
Sub Exercise()

    Rows.Select

End Sub
```

Замість збірки **Rows** можна використати збірку **Columns**, результат буде той самий.

Коли ви вибрали групу комірок, дана група зберігаються в об'єкті з ім'ям Selection. Ви можете використовувати цей об'єкт, щоб виконати однакові дії на всіх комірках, які в даний час відібрані.

Ім’я комірки

Ми вже побачили, що для того, щоб посилатись на комірку, використовуючи її ім’я, можна вказати це ім’я за допомогою типу string в **Range** об’єкті.

Після створення назви для групи комірок, щоб посилатись на дані комірки, використовуючи дане ім'я, викликають **Range** об'єкт і вказують ім'я за допомогою типу string.

Сітки і заголовки робочого аркуша

Показ сітки комірок

Щоб показати або сховати сітки, викликають **ActiveWindow** об'єкт та визначають його функцію **DisplayGridlines** типу Boolean. Якщо присвоїти даній функції величину True, то сітка з'явиться. Якщо присвоїти даній функції величину False, то сітка зникне. Наприклад:

```
Sub Exercise()

   ActiveWindow.DisplayGridlines = False

End Sub
```

Показ заголовків робочого аркуша

Щоб показати або сховати сітки, викликають **ActiveWindow** об'єкт та визначають його функцію **DisplayHeadings** типу Boolean. Якщо присвоїти даній функції величину True, то заголовки з'являться. Якщо присвоїти даній функції величину False, то заголовки зникнуть. Наприклад:

```
Sub ShowHeadings()

   ActiveWindow.DisplayHeadings = False

End Sub
```

Дії над комірками

Додавання комірок

Ми знаємо, що для того, щоб вставити стовпчики (вертикальних) комірок, можна використовувати Columns збірку, визначивши в круглих дужках індекс та викликавши **Insert** метод. Наприклад:

```
Sub CreateColumn()

   Columns(3).Insert

End Sub
```

Нам також відомо, як створювати набір рядків з комірок по горизонталі.

Введення даних до комірок

Введення даних до комірок полягає в додаванні однієї або більше величин до однієї або більше комірок. Це може бути зроблено вручну, автоматично або за допомогою програми. Ми вже знаємо, що комірка з робочого аркуша, яка вибрана на даний момент, називається **ActiveCell**. До того ж, щоб за допомогою програми додати величину до активної комірки, необхідно привласнити дане значення до даного об'єкту.

Естетичне форматування комірок

Форматування комірки за допомогою шрифту

Шрифт – це опис характеристик символів, що представляє значимі (графічні) характеристики. Шрифт – це об'єкт, що утворений за допомогою багатьох характеристик, включаючи ім'я, розмір та стиль.

Щоб визначити шрифт, бібліотека VBA забезпечена класом під назвою Font. Даний клас оснащений необхідними характеристиками.

Ім'я шрифту

Щоб за допомогою програми визначити ім'я шрифту, спочатку вказують комірку або групу комірок, для яких потрібно застосувати даний шрифт, потім вибираємо **Font** об'єкт, після якого іде функція **Name.** Потім вказуємо ім'я шрифту для комірки чи групи комірок.

```
Sub Exersice()

   Range("B2").Font.Name = "Rockwell Condensed"

   Range("B5").Font.Name = "Cambria"

End Sub
```

Розмір шрифту

Окрім ім'я, для шрифту ще вказують його розмір. Щоб за допомогою програми визначити розмір шрифту для комірки чи групи комірок,

викликають **Font** об’єкт, після якого слідує властивість **Size** та визначається бажана величина для розміру шрифту.

```
Sub Exersice()

   Range("B2").Font.Size = 24

End Sub
```

Стиль шрифту

Стиль шрифту – це техніка малювання характеристик тексту. Для підтримки стилю шифту, об’єкт **Font** оснащений багатьма характеристиками типу Boolean, такими як **Bold, Italic, Underline**, і **Strikethrough**. Проте, для того, щоб граматично визначити стиль шрифту для комірки чи групи комірок, спочатку визначають дану комірку чи групу комірок, потім викликають **Font** об’єкт, після якого слідує бажаний стиль та визначена бажана величина типу Boolean.

```
Sub Exersice()

   Range("B2").Font.Bold = True
End Sub
```

Колір тексту

Символ або текст може мати різний колір для кращого візуального представлення. VBA підтримує функцію колір тексту на різних рівнях. Щоб підтримати колір, об’єкт **Font** оснащений функцією під назвою **Color**. Щоб визначити колір, необхідно вказати бажаний колір для даної функції. VBA забезпечує (обмежений) список кольорів, які можуть бути визначені використовуючи ім’я-константу. Ось даний список:

Color Name	Constant	Value	Color
Black	vbBlack	&h00	
Red	vbRed	&hFF	

Green	vbGreen	&hFF00	
Yellow	vbYellow	&hFFFF	
Blue	vbBlue	&hFF0000	
Magenta	vbMagenta	&hFF00FF	
Cyan	vbCyan	&hFFFF00	
White	vbWhite	&hFFFFFF	

Таким чином, доступними кольорами є **vbBlack, vbRed, vbGreen, vbYellow, vbBlue, vbMagenta, vbCyan,** та **vbWhite.** Дані кольори є стандартними. Фактично, колір в Microsoft Windows визначається величиною між 0 та 16,581,375 (в наступному уроці ми дізнаємось звідки з'явились дані номери). Це означає, що Ви можете визначити додатнє число для функції **Font.Color** та використовувати відповідний колір.

Кольори для Font Color представлені за допомогою властивості під назвою **ThemeColor**. Кожен колір з секції Theme Colors має еквівалентне ім'я в VBA. Якщо Ви знаєте ім'я кольору, визначайте його за допомогою функції **ThemeColor**.

Альтернативним варіантом щоб визначити колір є використання функції під назвою RGB, це буде розглянуто в наступному уроці.

```
Sub Exersice()

   Range("B2").Font.Color = vbBlue

   Range("B5").Font.ThemeColor = 5

End Sub
```

Об’єднання комірок

Щоб об’єднати комірки за допомогою програми, спершу необхідно їх вибрати та визначити властивість **MergeCells** типу Boolean. Потім вказується **True** або **False** залежно від бажаного результату.

```
Sub Exersice()

    Rem Merge the cells H15, I15, H16, and I16

    Range("H15:I16").MergeCells = True

    Rem Hide the gridlines

    ActiveWindow.DisplayGridlines = False

End Sub
```

Вирівнювання комірок

Щоб вирівняти текст в комірці або групі комірок, спочатку вказують дану клітинку або групу клітинок, потім використовують **HorizontalAlignment** або **VerticalAlignment** властивості і вказують бажане значення.

```
Sub CreateWorkbook()

    Rem Об’єднання клітинок H15, I15, H16 та I16

    Range("H15:I16").MergeCells = True

    Rem Вирівнювання тексту ліворуч

    Range("H15:H16").VerticalAlignment = xlCenter

End Sub
```

Відступ в комірках

Щоб за допомогою програми зробити відступ в комірці або групі комірок, необхідно скористатись функцією **IndentLevel.** Потім вказується бажана величина. Наприклад:

```
Range("A1").IndentLevel = 5
```

Межі комірок

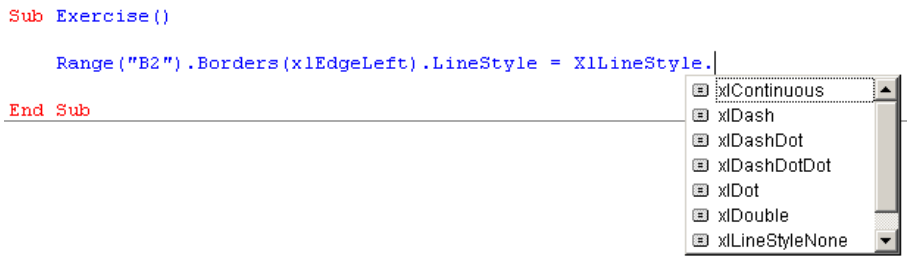
Стиль ліній для межі

Комірка показується як прямокутник з межами та фоном. Щоб за допомогою програми контролювати межі для комірки чи групи комірок, необхідно спочатку вказати дану комірку або групу комірок а потім використати їх Borders об’єкт. Цей об’єкт доступний як індексована функція. Наприклад:

```
Range("B2").Borders()
```

В круглих дужках для функції Borders вказують межу, яку необхідно змінити. Прості доступні змінні це **xlEdgeBottom, xlEdgeTop, xlEdgeLeft,** та **xlEdgeRight**. Інколи можна вибрати групу комірок і виконати дії з лінією (лініями) між ними. Щоб виконати це, функція **Borders** може прийняти індекс під назвою **xlInsideVertical** для вертикальних меж між двома комірками або індекс під назвою **xlInsideHorizontal** для горизонтальних меж між комірками.

Після того, як визначені межі, над якими будуть виконуватись дії, Ви маєте визначити тип характеристик, які Ви хочете змінити. Наприклад, Ви можете визначити тип ліній для межі. Для виконання даної дії, **Borders** об’єкт оснащений властивістю під назвою **LineStyle**. Щоб визначити тип ліній, який має мати межа, Ви можете визначити значення для функції **LineStyle**. Доступними змінними є **xlContinuous, xlDash, xlDashDot, xlDashDotDot, xlDot, xlDouble, xlSlantDashDot,** та **xlLineStyleNone.** Отже, Ви можете визначити будь-яку з цих змінних для функції. Щоб допомогти собі у цьому, Ви можете вказати **LineStyle** трохи зачекати, а потім вибрати бажане значення зі списку, який з’явиться:



Товщина ліній для межі

Після того, як визначений тип ліній для меж, можна визначити товщину даних ліній. Для виконання цього, **Borders** об'єкт оснащений властивістю під назвою **Weight**. Доступними є значення **xlHairline, xlThin, xlMedium,** та **xlThick**.

```
Sub Exersice()

   Range("B5:J5").Borders(xlEdgeBottom).Weight = xlMedium

   Range("D6:F6").Borders(xlEdgeBottom).Weight = xlHairline

   Range("B8:J8").Borders(xlEdgeBottom).Weight = xlThin

   Range("B12:J12").Borders(xlEdgeBottom).Weight = xlMedium

End Sub
```

Колір ліній для межі

Щоб за допомогою програми визначити колір ліній для межі, використайте функцію **Borders** для комірки чи групи комірок і визначіть межу, колір якої Ви хочете змінити так само, як ми це робили раніше. Щоб мати змогу змінювати колір меж, **Borders** об'єкт оснащений властивістю під назвою **Color**. Щоб визначити колір, вкажіть бажаний колір для даної функції. VBA забезпечує (обмежений) список кольорів, таких як **vbBlack, vbWhite, vbRed, vbGreen,** та **vbBlue**. Насправді, колір в Microsoft Windows представляється як колір між 0 і 16 581 375.

```
Sub Exersice()
```

```
    Range("B5:J5").Borders(xlEdgeBottom).ThemeColor = 5

End Sub
```

Фон комірки

Комірка має фон, який за замовчуванням є білим. Якщо Ви хочете змінити фон, визначте комірку або групу комірок, використовуючи Range клас. **Range** клас оснащений властивістю під назвою **Interior.** З даної властивості, Ви можете використати **ThemeColor** і вказати бажаний колір.

```
Sub Exersice()

    Range("B3:J3").Interior.ThemeColor = xlThemeColorLight2

End Sub
```

Лекція 3. Процедури і функції

Процедури

Процедура – це частина коду, створена для виконання завдання, відокремленого від електронних таблиць, але дія якого може бути використана для доповнення таблиці. Однією з переваг процедури є те, що як тільки її створено, ви можете отримати доступ до неї в разі потреби і стільки разів, скільки ви хочете.

Є дві категорії процедур, які Ви будете використовувати в таблицях: ті, які вже встановлені з Microsoft Excel, і ті, які Ви створите.

У мові Visual Basic, як і більшості інших мов, існує два види процедур: функції та суб-процедури.

Суб-процедури

Суб-процедура – це завдання, яке виконується, але не повертає результат. Щоб створити суб-процедуру, починають із ключового слова **Sub** та ім'я процедури. Після імені процедури завжди пишуть дужки.

Наприкінці процедури треба ввести **End Sub**. Таким чином, основною формою створеної суб-процедури є:

```
Sub ProcedureName()

End Sub
```

Ім'я процедури має задовольняти правила для назв змінних. Якщо процедура виконує дію, яку можна описати дієсловом, можете використовувати його для назви, наприклад: Show, Play, Dispose, Close.

Код між лініями **Sub** і лінії називають тілом процедури.

Викликання суб-процедури

Використання процедури, як написаної Вами, так і вбудованої в Visual Basic, називають викликанням.

Щоб викликати просту процедуру, досить ввести її ім'я. Ось приклад:

```
Sub CreateCustomer()
    Dim strFullName As String
    strFullName = "Paul Bertrand Yamaguchi"
End Sub
Sub Exercise()
    CreateCustomer
End Sub
```

Також для викликання процедури перед її іменем пишуть ключове слово **Call**. Ось приклад:

```
Sub Exercise()
    Call CreateCustomer
```

```
End Sub
```

Під час викликання процедури із або без ключового слова **Call**, можна додатково писати дужки праворуч від імені процедури.

```
Sub Exercise()

    CreateCustomer()

End Sub
```

Процедури і рівні доступу

Як і для змінної, можна керувати рівнем доступу до процедури. Процедура може бути приватною або публічною. Щоб вказати рівень доступу до процедури, перед нею вказують ключове слово **Private** або **Public**. Ось приклад:

```
Private Sub CreateCustomer()

    Dim strFullName As String

    strFullName = "Paul Bertrand Yamaguchi"

End Sub
```

Правила є такими самими, які були застосовані для глобальних змінних:

- **Private**: Якщо процедура зроблена приватною, її можна викликати іншими методами того самого модуля. Процедури ззовні модуля не можуть отримати доступ до такої процедури. Крім того, коли процедура є приватною, її ім'я не з'являється в діалоговому вікні Макросу.
- **Public**: процедура, створена публічною, може бути викликана процедурами і цього модуля, і процедурами інших модулів. Крім того, якщо процедуру було створено як публічну, коли ви отримуєте доступ до діалогового вікна Макросу, з'явиться її ім'я, і ви зможете запустити її звідти.

Функції

Як і суб-процедуру, функцію використовують для виконання завдання. Основна відмінність між суб-процедурою і функцією є те, що функція повертає результат. Ми також говоритимемо, що функція "повертає значення". Щоб розрізняти, для функції використовують інший синтаксис.

Створення функції

Щоб створити функцію, використайте ключове слово **Function**, після якого напишіть ім'я та дужки. Оскільки функція повертає значення, то, на відміну від суб-процедури, необхідно вказати його тип. Щоб дати цю інформацію, праворуч від дужок, вкажіть ключове слово **As** та відповідний тип даних. Щоб вказати кінець тіла функції, напишіть **End Function**. Виходячи з цього, мінімальний синтаксис для створення функції такий:

```
AccessModifier Function FunctionName() As DataType

End Function
```

Як було показано для суб-процедур, функція може мати модифікатор доступу.

Як і для змінних, ви також можете використовувати символьний тип в якості типу функції і опустити вираз **As DataType**. Тип символу друкується праворуч від імені функції і перед дужками. Наприклад, **GetFullName$()**. Як і для змінних, необхідно використовувати відповідний тип символу для функції:

Символ	Тип даних, який повертає функція
$	Рядковий
%	Цілочисельне значення від -32768 до 32767
!	Десяткове число одинарної точності
#	Десяткове число подвійної точності
@	Грошове значення

Приклад:

```
Function GetFullName$()

End Function
```

Повертання значення

Після виконання завдання у функції, щоб вказати значення, що повертається, десь перед рядком Function End, треба ввести ім'я функції, а потім знак =, за яким написати значення, яке повертається функцією. Ось приклад, в якому функція повертає ім'я:

```
Function GetFullName$()
   Dim FirstName As String, LastName As String
   FirstName = "Patricia"
   LastName = "Katts"
   GetFullName = LastName & ", " & FirstName
End Function
```

Викликання функції

Як і для суб-процедур, для того, щоб використовувати функцію у вашій програмі, ви повинні викликати її. Щоб викликати функцію, ви можете просто ввести його ім'я у відповідному розділі програми.

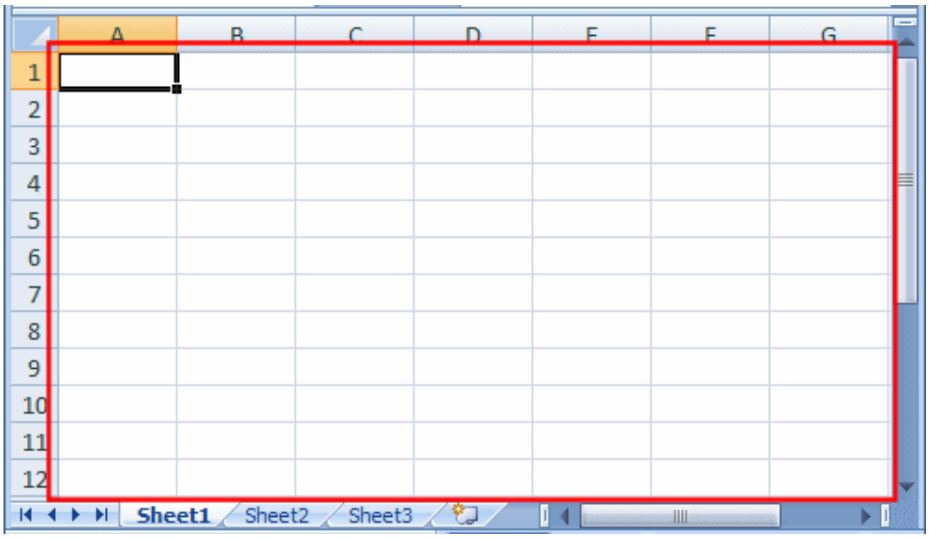
При викликанні функції можна, хоча і необов’язоко, писати дужки праворуч від її імені.

Основна мета функції – повернути значення. Щоб краще скористатися таким значення, ви можете присвоїти змінній ім’я функції. Ось приклад:

```
Sub Exercise()

   Dim FullName$

   FullName = GetFullName()

   ActiveCell.FormulaR1C1 = FullName

End Sub
```

Викликання функції в таблиці

Оскільки функція повертає значення, то аналогічно, Ви можете використовувати її безпосередньо в таблиці. Щоб зробити це, клацніть будь-який клітинці в робочій зоні:



Після натискання на клітинці, введіть =, за яким вкажіть ім'я функції. Під час набирання імені функції, Microsoft Excel представить список функцій, які відповідають першим введеним літерам. Якщо ви бачите ім'я функції, ви можете двічі клацнути по ньому, або можете просто продовжувати друкувати. Після введення імені функції, надрукуйте дужки і натисніть Enter або натисніть кнопку введення  на панелі **Формула**.

Аргументи і параметри

Аргументи

Досі для використання значення в процедурі, ми повинні були оголосити його. У деяких випадках процедурі може знадобитися зовнішнє значення для виконання свого завдання. Значення, яке подається процедурі, називають аргументом.

При створенні процедури, яка буде використовувати зовнішнє значення, оголосіть аргумент, який відображає це значення, у дужках процедури. Для суб-процедури синтаксис має вигляд:

```
Sub ProcedureName(Argument)

End Sub
```

Якщо Ви створюєте функцію, то синтаксис буде таким:

```
Function ProcedureName(Argument) As DataType

End Function
```

Аргумент повинен бути оголошений як звичайна змінна, опускаючи ключове слово **Dim**. Ось приклад, який створює функцію, яка приймає рядок в якості аргументу:

```
Function CalculatePayroll(strName As String) As Double
```

У той час як певна процедура може використовувати один аргумент, інша процедура може потребувати більше одного аргументу. У цьому разі в дужках процедури аргументи розділяють комами. Ось приклад суб-процедури, яка приймає два аргументи:

```
Sub EvaluateInvoice(EmplName As String, HourlySalary As Currency)
```

У тілі процедури, яка приймає один або декілька аргументів, використовуйте аргументи, як Ви вважаєте за потрібне, так, як якщо б вони були локально оголошені змінні. Наприклад, Ви можете залучити їх

зі значеннями всередині процедури. Ви також можете використовувати виключно значення аргументів для виконання завдання.

Викликання процедури з аргументом

Значення, задане в аргументі, також називають параметром. Для виклику процедури, яка приймає аргумент, введіть її ім'я. Тоді у вас є різні варіанти, щоб отримати доступ до її аргументів.

Раніше ми бачили, що, для виклику процедури, ви могли б просто написати її ім'я. Після імені процедури відкрийте дужку "(", після неї запишіть значення аргументу. Якщо процедура вимагає більше одного аргументу, розділіть значення комами. Приклад:

```
Private Function GetFullName$(First As String, Last As String)
    Dim FName As String
    FName = First & Last
    GetFullName = FName
End Function
Sub Exercise()
    Dim FirstName As String, LastName As String
    Dim FullName As String
    FirstName = "Patricia "
    LastName = "Katts"
    FullName = GetFullName(FirstName, LastName)
    ActiveCell.FormulaR1C1 = FullName
End Sub
```

Як вже згадувалося раніше, ви можете також використовувати ключове **Call** для виклику процедури.

Коли ви викликаєте процедуру, яка приймає більше одного аргументу, ви повинні надати значення аргументів в тому порядку, в якому вони перераховані всередині дужок. Але ця вимога не обов'язкова. Якщо ви знаєте імена аргументів, ви можете ввести їх в будь-якому порядку зі значенням для кожного. Щоб зробити це, в дужках процедури, яку викликаєте, введіть ім'я аргументу, значення якого ви хочете задати, а потім оператор: = та бажане значення для аргументу. Ось приклад:

```
Sub Exercise()
   Dim FullName$
   FullName$ = GetFullName(Last:="Roberts", First:="Alan ")
   ActiveCell.FormulaR1C1 = FullName
End Sub
```

Вищеописаний спосіб по використанню дужок дійсний як для суб-процедур, так і для функцій. Якщо процедура, яку Ви викликаєте, є суб-процедурою, дужки можна опустити. Для виклику суб-процедури після її імені залиште пробіл, за яким напишіть ім'я аргументу призначений бажаного значення. Ось приклад

```
Private Sub ShowResult(ByVal Result As Double)
   ActiveCell.FormulaR1C1 = Result + 1
End Sub
Public Sub Exercise()
   Dim Number As Double
   Number = 100
```

```
    ShowResult Number
End Sub
```

Передавання аргументів

Передавання аргументу за значенням

При викликанні процедури, яка приймає аргумент, ми надавали значення для цього аргументу. Коли це зроблено, процедура, яку викликають, копіює значення аргументу і робить цю копію доступною для викликаної процедури. Таким чином, до самого аргументу процедура доступу не отримує. Цей спосіб називають передачею аргументу за значенням. Щоб вказати на це, пишуть ключове слово **ByVal** ліворуч від аргументу. Ось кілька прикладів:

```
Private Function GetFullName$(ByVal First As String, ByVal Last As String)
    Dim FName As String
    FName = First & Last
    GetFullName$ = FName
End Function
```

Якщо ви створили процедуру, яка приймає аргумент за значенням, і використали ключове слово **ByVal** для аргументу, то при викликанні процедури, не потрібно знову використовувати ключове слово **ByVal**, досить написати ім'я аргументу, як це робилося в прикладах раніше. Ось приклад:

```
Private Function GetFullName$(ByVal First As String, ByVal Last As String)
    Dim FName As String
    FName = First & Last
    GetFullName$ = FName
```

```
End Function

Sub Exercise()

    Dim FirstName As String, LastName As String

    Dim FullName As String

    FirstName = "Raymond "

    LastName = "Kouma"

    FullName = GetFullName(FirstName, LastName)

    ActiveCell.FormulaR1C1 = FullName

End Sub
```

Передавання аргументу через посиланням

Альтернативою передачі аргументів за значенням є передача адреси аргументу процедурі, яку викликають. При цьому процедура отримує доступ до аргументу через адресу в пам'яті. За допомогою цієї техніки, будь-яка дія, що виконана над аргументом, буде збережена після закінчення процедури. Якщо значення аргументу змінюється, аргумент матиме тепер нове значення, втрачаючи при цьому початкове. Цей спосіб називають передаванням аргументу через посилання.

Щоб передати аргумент через посилання, ліворуч від нього введіть ключове слово **ByRef**. Це треба робити тільки при створенні процедури. При викликанні процедури ключове слово **ByRef** писати не потрібно. Тепер розглянемо ту саму програму, але з аргументом, що передається через посилання:

```
Private Sub ShowResult(ByRef Result As Double)

    ActiveCell.FormulaR1C1 = Result + 1

End Sub
```

```
Public Sub Exercise()
    Dim Number As Double
    Number = 100
    ShowResult Number
End Sub
```

Після виконання процедури значення змінної Number зміниться і дорівнюватиме 101.

Використовуючи цю техніку, ви можете передати будь-яку кількість аргументів через посилання і за значенням.

Передача аргументів через посиланням дозволяє процедурі повертати стільки значень, скільки Вам потрібно, в той час як звичайна функція може повернути тільки одне значення.

Константи, вирази і формули

Константи

Константа є значенням, яке не змінюється. Воно може бути числом, рядком або виразом. Щоб створити константу, використовуйте ключове слово **Const** і призначте потрібне значення. Ось приклад:

```
Const Number6 = 6
```

Вирази

Вираз – це один або більше символів, поєднані з іншими значеннями, створюючи нове значення. Наприклад, +16 – це вираз, який створює додатне значення 16. Більшість виразів, які ми знаємо, зроблені з арифметичних розрахунків. Наприклад, 422,82 * 15,55.

Щоб додати вираз у вибрану комірку, призначте його об'єкту **ActiveCell**.

```
Sub Exercise()
```

```
    ActiveCell = 422.82 * 15.5
End Sub
```

Формула

Формула – це інша назва виразу. Вона поєднує в собі одне або більше значень однієї або декількох змінних, із оператором, щоб отримати нове значення.

Для зручності присвоєння результату формули клітинці або групі клітинок, діапазон класу **Range** оснащений властивістю **Formula**. Ця властивість має тип **Variant**, що означає, що його значення може бути будь-яким, не обов'язково числом. Після отримання доступу до властивості **Formula**, ви можете призначити будь-яке значення, вираз чи формулу. Ось кілька прикладів:

```
Sub Exercise()
    Rem Using the Formula property to assign a string to the active cell
    ActiveCell.Formula = "Weekly Salary:"
    Rem Using the Formula property to assign an expression to cell B2
    Range("B2").Formula = 24.5 * 42.5
    Rem  Using the Formula property to assign
    Rem the same string to a group of cells
    Range("C2:F5, B8:D12").Formula = "Antoinette"
End Sub
```

Якщо ви створюєте аркуш, який буде використовуватися на комп'ютерах із різними мовами, натомість використовуйте властивість **FormulaLocal**. Властивість **FormulaLocal** здатна у разі потреби адаптуватися до іншої мовної версії Microsoft Excel.

Крім **Formula,** клас **Range** також оснащено властивістю **FormulaR1C1**. Його функціональні можливості в основному такі самі, як і в **Formula**. Ось кілька прикладів:

```
Sub Exercise()
   Rem Using the Formula property to assign a string to the active cell
   ActiveCell.FormulaR1C1 = "Weekly Salary:"
   Rem Using the Formula property to assign an expression to cell B2
   Range("B2").FormulaR1C1 = 24.5 * 42.5
   Rem  Using the Formula property to assign
   Rem the same string to a group of cells
   Range("C2:F5, B8:D12").FormulaR1C1 = "Antoinette"
End Sub
```

Лекція 4. Елементи керування Windows

Форми

Комп'ютерні програми, зокрема ті, що працюють на платформі Microsoft Windows, оснащені об'єктами, які називають елементами керування Windows. Це ті об’єкти, що дозволяють людині взаємодіяти з комп'ютером.

Первинний елемент керування, який використовують у більшості додатків, називають формою. Форму в основному використовується в якості платформи, на яку Ви додаєте інші елементи керування. З цієї причини форму називають контейнером. Сама по собі форма не є особливо корисною. Ви повинні додати інші об'єкти до неї.

Щоб додати форму, на стрічці натисніть **Розробник**, далі **Visual Basic**, у головному меню виберіть **Insert -> UserForm**.

При створенні або додаванні форми, також автоматично створюється відповідний модуль. Щоб отримати доступ до модуля, пов'язаного з формою, Ви можете натиснути правою кнопкою миші на форму та вибрати **Перегляд коду**.

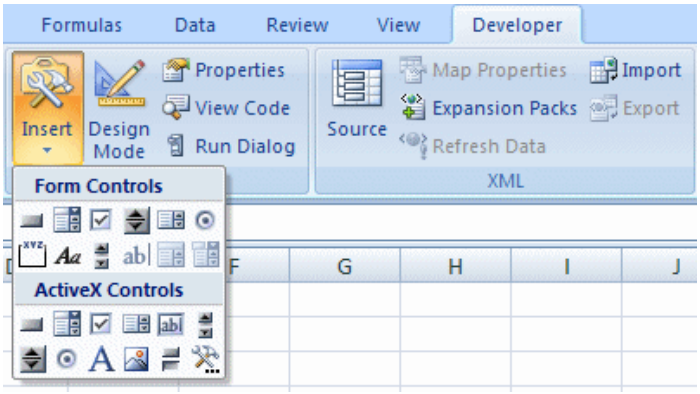
Аркуші

При роботі в Microsoft Excel, ви можете використовувати елементи керування Windows як на робочій області аркуша, так і в формі Microsoft Visual Basic. Таким чином, так само, як і форма, таблиця також є контейнером управління.

Елементи керування

Основні об'єкти, які використовують, щоб допомогти людині взаємодіяти з комп'ютером, – це вікна управління. Є два основних способи доступу до цих об'єктів при роботі з Microsoft Excel.

Якщо Ви працюєте в Microsoft Excel, ви можете додавати або позиціонувати деякі вікна управління в документі. Щоб зробити це, на стрічці натисніть **Розробник**. У розділі **Елементи керування** виберіть команду **Вставити**:



При цьому буде відображено список елементів керування, доступних в Microsoft Excel. Їх відображено у двох розділах: *Елементи керування форми* й *Елементи керування ActiveX*. Якщо Ви працюєте із таблицею в Microsoft Excel, Ви повинні використовувати тільки елементи управління із розділу *Елементи керування ActiveX*. Якщо Ви працюєте із формою в Microsoft Visual Basic, з'явиться панель інструментів, оснащена різними елементами керування.

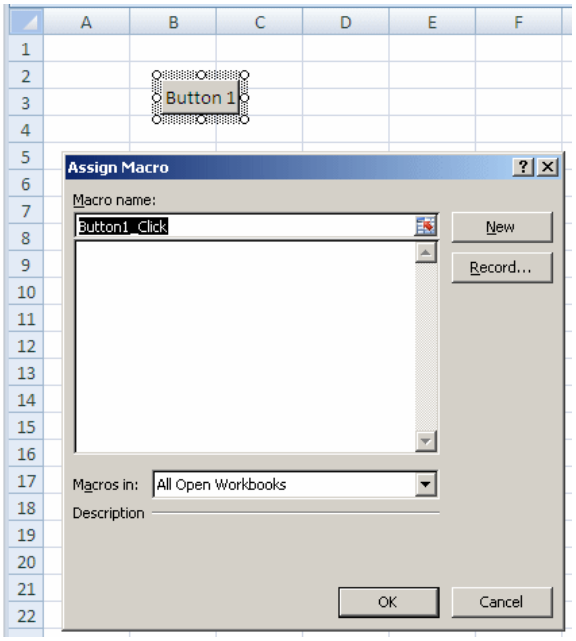
Кожен елемент управління у вкладці Розробника на стрічці або в арсеналі Microsoft Visual Basic має певне ім'я. Ви можете бути знайомі з деякими з цих елементів управління. Якщо ви не впевнені, можна розташувати курсор миші над контролем і винирне підказка. Імена, які ми будемо використовувати:

Елементи керування ActiveX	**Назва**	**Елементи керування форми**	**Назва**
	*Command Button*		*Label*
	*Combo Box*		*Toggle Button*
	*Check Box*		
	*List Box*		*TabStrip*
	*Text Box*		*MultiPage*
	*Scroll Bar*		*ScrollBar*
	*Spin Button*		*Text Box*
	*Option Button*		*Image*
	*Label*		*RefEdit*
	*Image*		*Frame*
	*Toggle Button*		

Додавання елемента керування

Щоб долучити один з елементів управління зі стрічки або панелі інструментів, Ви можете клацнути по ньому. Якщо потім Ви просто натиснете його розміщення, елемент буде додано там, де Ви натиснули із деякими розмірами за замовчуванням.

У Microsoft Excel, якщо Ви натиснете на кнопку в секції керування форми та натиснете робочу область таблиці, відразу після додавання елементу керування виникне діалогове вікно **Призначити макрос**:



Щоб програмно додати елемент керування до таблиці, використовуйте таку формулу:

```
Private Sub Exercise()
    Worksheets(1).OLEObjects.Add "Forms.ControlName.1"
End Sub
```

Єдине, що Вам потрібно знати і змінити в цій формулі – це *ControlName*. Використовуйте такі назви:

Використовуйте цю назву	Щоб отримати		Використовуйте цю назву	Щоб отримати
CheckBox	Поле з прапорцем		ComboBox	Поле зі списком
CommandButton	Командну кнопку		Label	Мітку
ListBox	Список		Image	Рисунок
OptionButton	Кнопку вибору		ScrollBar	Смугу прокрутки
SpinButton	Кнопку прокрутки		TextBox	Текстове поле
ToggleButton	Кнопку перемикання			

Повідомлення та події елементів керування Windows

Коли використовують елемент управління, він повинен взаємодіяти з операційною системою. Наприклад, при натисканні на об'єкті, той повинен повідомити операційній системі, що його було натиснуто. Це стосується кожного елемента управління, що використовуються в додатку. Оскільки типовий додаток може включати в себе багато елементів управління, було розроблено механізм для керування цим процесом.

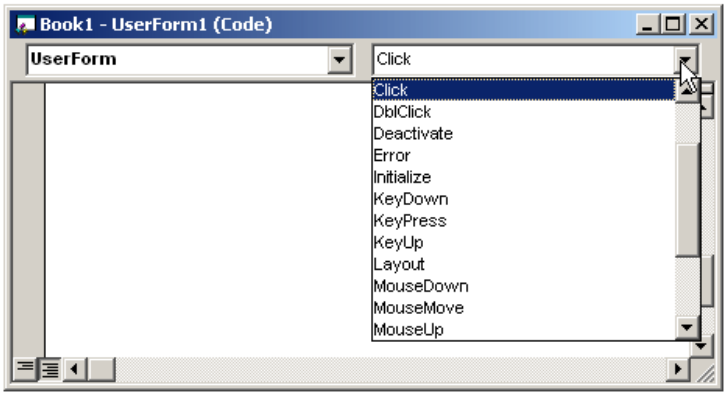
Щоб повідомити про свій намір операційній системі, елемент управління Windows має створити повідомлення і відправити його (операційній системі).

Для того, щоб зробити повідомлення зрозумілим, елемент керування, який хоче відправити його, має надати три важливих шматки інформації:

- *Хто послав повідомлення?* Коли елемент управління посилає повідомлення, він повинен ідентифікувати себе, тому що одночасно може бути багато повідомлень від різних елементів. Операційній системи треба знати, звідки прийшло повідомлення. Це одна з причин, чому кожен елемент повинен мати ім'я. Крім того, оскільки кожне повідомлення є характерним для елемента управління, який послав його, повідомлення вважають його приватною частиною.
  Відповідно до цього код повідомлення починається з **Private Sub**, за яким слідує ім'я елемента керування, який посилає повідомлення:

```
Private Sub ControlName
```

- *Яке повідомлення?* Коли елемент управління посилає повідомлення, він повинен вказати тип повідомлення. Елемент управління може відправляти різні типи повідомлень. Наприклад, коли елемент управління натиснуто, він посилає повідомлення Click. Якщо цей самий елемент отримує фокус, коли Ви натискаєте клавішу, він посилає повідомленні клавіатурного типу. Коли миша проходить через цей елемент керування, він посилає ще інший тип повідомлення.
  Кожне повідомлення, яке може послати елемент управління, має ім'я. Щоб побачити типи повідомлень, які доступні для конкретного елемента управління, відкрийте Microsoft Visual Basic. У полі зі списком об'єктів, виберіть ім'я елемента управління. Потім натисніть на стрілку поля зі списком дій:



За домовленістю, ім'я повідомлення пишуть після імені елемента управління, відокремлюючи його підкресленням:

Private Sub ControlName_Push

- *Аргументи.* Аргумент – це додаткова інформація, необхідна для обробки повідомлення. Коли елемент управління посилає повідомлення, то, можливо, йому буде потрібно супроводити його деякою інформацією. Наприклад, якщо Ви встановите курсор миші над елементом керування і натиснули, операційна система може хотіти знати, яку кнопку миші було використано. З іншого боку, якщо Ви оберете об'єкт і почнете перетягування, операційна система може треба буде знати, чи було при цьому натиснуто клавіші Shift чи Ctrl.

  Аргументи повідомлення вказують у дужках. Вони будуть виглядати так:

```
Private Sub ControlName_Push(Argument1, Argument2, Argument_n)
End Sub
```

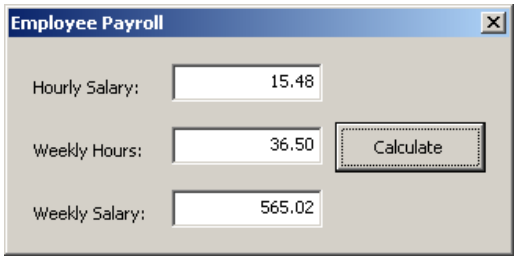
Як згадувалося раніше, повідомлення має бути створено і надіслано. Дію відправлення повідомлення називають подією. Виходячи з вищевикладеного опису, щоб створювати і відправляти повідомлення, у

полі зі списком об'єктів Ви можете вибрати ім'я елемента управління, який буде відправляти повідомлення, а потім вибрати потрібне повідомлення в полі зі списком Procedure. Коли Ви зробите це, Microsoft Visual Basic напише для Вас перший рядок, який визначає ім'я елемента управління, назву події, її аргументи, якщо такі є, і напише End Sub. Потім можна ввести потрібний код між цими двома лініями.

Подія Click

Більшість елементів управління Windows мають спеціальну подію, яку вважають обраною за замовчуванням. Наприклад, коли ви думаєте про кнопку, перша дія, яке приходить на думку, це клацнути. З цієї причини, **Click** є подією за замовчуванням для кнопки. Якщо ви не хочете використовувати цю подію або запустити іншу подію для цього елементу керування, можете просто вибрати подію, в полі зі списком Procedure.

Наступний приклад описує подію Click для кнопки у формі з трьома текстовими полями:




```
Private Sub cmdCalculate_Click()
    Dim HourlySalary As Currency
    Dim WeeklyHours As Double
    Dim WeeklySalary As Currency
    HourlySalary = CCur(txtHourlySalary.Text)
    WeeklyHours = CDbl(txtWeeklyHours.Text)
```

```
    WeeklySalary = HourlySalary * WeeklyHours

    txtWeeklySalary.Text = CStr(WeeklySalary)

End Sub
```

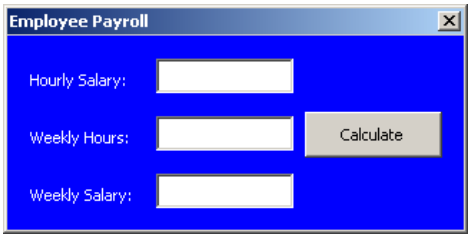
Подія Double-Click

Іншою поширеною дією, яку виконують над елементом управління, є подвійне клацання. Ця дія спричиняє запуск елементом управління події **DblClick**.

Повертаючись до попереднього прикладу, у списку Procedure виберемо **DblClick** та опишемо структуру події так:

```
Private Sub UserForm_DblClick(ByVal Cancel As MSForms.ReturnBoolean)

    lblHourlySalary.BackColor = vbBlue

    lblWeeklyHours.BackColor = vbBlue

    lblWeeklySalary.BackColor = vbBlue

    lblHourlySalary.ForeColor = vbWhite

    lblWeeklyHours.ForeColor = vbWhite

    lblWeeklySalary.ForeColor = vbWhite

    BackColor = vbBlue

End Sub
```

Двічі клацнувши на формі, отримаємо



Подія входу Enter

Так само, як програма може мати різні форми, і форма може бути оснащеною різними елементами управління. Зокрема і будь-яка форма введення даних. На формі, яка оснащена великою кількістю елементів управління, одночасно можна змінити тільки один елемент управління. Тоді кажуть, що такий елемент управління отримав фокус. Щоб дати фокус елементу управління, ви можете клацнути на ньому або натискати **Tab**, поки бажаний контроль не вкаже, що він має фокус. У формі з великою кількістю елементів управління, елемент, що має фокус, може показати курсор або пунктирну лінію навколо свого виділення або заголовку.

Коли форма або елемент керування отримує фокус, вони запускають подію **Enter**. Якщо елемент управління є текстовим, то курсор, який блимає на елементі, вказує, що на отримання фокуса.

Подія **Enter** не приймає аргументів:

```
Private Sub TextBox1_Enter()

End Sub
```

Подія виходу Exit

Після використання елементу управління, ви можете переключитися на інший елемент, клацнувши на інший або натиснувши **Tab**. Це зміщує фокус із поточного елемента управління на інший. При переході фокусу, елемент управління, який його мав до цього, запускає подію **Exit**.

Подія **Exit** приймає один аргумент:

```
Private Sub TextBox1_Exit(ByVal Cancel As MSForms.ReturnBoolean)

End Sub
```

Події клавіатури

Щоб відобразити символи під час роботи з текстом, потрібно натикати клавіші на клавіатурі. Якщо програма налаштована на отримання тексту,

ваше натискання буде відображено символами на екрані. Клавіатуру також використовують для виконання й інших дій, зокрема погодження з тим, що відображається в діалоговому вікні, або його відкидання.

При натисканні клавіш на клавіатурі, елемент управління, у якому вводять символи, посилає одне або кілька повідомлень операційній системі. Існують три основні події, які Microsoft Windows асоціює з клавіатурою.

KeyDown: при натисканні клавіші на клавіатурі, запускають подію **KeyDown**. Подія **KeyDown** приймає два аргументи:

```
Private Sub TextBox1_KeyDown(ByVal KeyCode As MSForms.ReturnInteger, _
        ByVal Shift As Integer)
End Sub
```

KeyUp: Під час відпускання натисненої клавіші запускає подія **KeyUp**.

Ці попередні дві події застосовні майже до будь-якої клавіші на клавіатурі, навіть якщо ви не друкуєте чи якщо в результаті натискання клавіші у документі не буде відображено символу.

```
Private Sub TextBox1_KeyUp(ByVal KeyCode As MSForms.ReturnInteger, _
        ByVal Shift As Integer)
End Sub
```

KeyPress: подія **KeyPress**, стартує після натиснення клавіші, яку розпізнано як символьну.

```
Private Sub TextBox1_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
End Sub
```

Деякі клавіші на клавіатурі не відобразять нічого в документі. Замість цього, вони виконують (тільки) дії. Прикладами таких клавіш є Enter, Tab,

Esc. Тому, якщо ви хочете з'ясувати, яку клавішу натиснуто, використовуйте подію **KeyDown** замість події **KeyPress**, навіть якщо Ви натиснули клавішу.

Натискання кнопки миші

Миша оснащена, як правило, двома кнопками, які натискають, щоб викликати дію. У порівнянні з клавіатурою, миша вимагає багато інших подій, які прямо або опосередковано пов'язані з натискання однієї з її кнопок.

При натисканні на одну з кнопок на миші, стартує подія **MouseDown**. Ця подія несе достатньо інформації через три аргументи.

```
Private Sub txtFirstName_MouseDown(Button As Integer, Shift As Integer,
    X As Single, Y As Single)
End Sub
```

- Операційна система повинна знати, яку кнопку було натиснуто. Ліва кнопка називається **vbLeftButton**. Права кнопка – **vbRightButton**. Якщо миша оснащена середньою кнопкою, то на неї посилаються як **vbMiddleButton**. У дійсності, ці кнопки мають (сталі) числові значення 0, 1 і 2 відповідно.
- По-друге, операційна система повинна знати, чи була натиснуто спеціальний клавішу: Shift, Ctrl або Alt. Ці кнопки називають **vbShiftMask**, **vbCtrlMask** і **vbAltMask** відповідно. Їхні значення 1, 2 і 4 відповідно.
- Нарешті, операційна система повинна знати екранні координати курсору миші. **X** являє собою відстань від верхнього лівого кута батьківського вікна до миші. **Y** – це вертикальна відстань до точки від лівого верхнього кута вниз.

Відпускання кнопки миші

Коли ви відпустите кнопку, яка була натиснута на миші, стартує нова подія. Цю подію називають **MouseUp**. Він надає ті ж типи інформації, як і подія **MouseDown**:

```
Private Sub txtFirstName_MouseUp(Button As Integer, Shift As Integer,
        X As Single, Y As Single)
End Sub
```

Рухання миші

Повідомлення **MouseMove** надсилають при переміщенні миші над елементом управління. Воно надає таку саму інформацію як і події **MouseDown** та **MouseUp**:

```
Private Sub txtFirstName_MouseMove(Button As Integer, Shift As Integer,
        X As Single, Y As Single)
End Sub
```

Зміна текстового поля

Одне з найбільш важливих повідомлень текстового поля надсилається при зміні його вмісту. Тобто, коли текст всередині видалено, додано або відредаговано. При натисканні в текстовому полі і початку друкування в ньому або зміні його тексту, елемент управління запускає подію **Change**.

У прикладі форми з трьома текстовими полями застосуємо подію **Change** так:

```
Private Sub txtFirstName_Change()
  Dim FirstName As String
  Dim LastName As String
  Dim FullName As String
```

```
    FirstName = txtFirstName.Text
    LastName = txtLastName.Text
    FullName = FirstName & " " & LastName
    txtFullName.Text = FullName
End Sub
```

Далі виберемо поле **txtLastName** та застосуємо його подію Change так:

```
Private Sub txtLastName_Change()
    Dim FirstName As String
    Dim LastName As String
    Dim FullName As String
    FirstName = txtFirstName.Text
    LastName = txtLastName.Text
    FullName = FirstName & " " & LastName
    txtFullName.Text = FullName
End Sub
```

В результаті під час друкування в полі **Last Name** одночасно змінюватиметься і поле **Full Name**: