МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНА МЕТАЛУРГІЙНА АКАДЕМІЯ УКРАЇНИ

Г. Г. Швачич, О. В. Овсяніков, В. В. Кузьменко,
Н. І. Нєчаєва, Л. М. Петричук

ІНФОРМАТИКА ТА КОМП'ЮТЕРНА ТЕХНІКА
ЕЛЕМЕНТИ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ

Розділ «Реалізація концепції о'бєктно-орієнтованого програмування в мові Visual Basic for Application»

Затверджено на засіданні Вченої ради академії
як навчальний посібник

Дніпропетровськ НМетАУ 2007

УДК 004 (075.8)
Швачич Г.Г., Овсянніков О. В, Кузьменко В.В, Нечаєва Н.І., Петричук Л.М.. Інформатика та комп’ютерна техніка. Навчальний посібник. – Дніпропетровськ: НМетАУ, 2007. – 52 с.

Викладена реалізація концепції об’єктно-орієнтованого програмування в мові Visual Basic for Application.

Призначений для студентів спеціальності 6.020100 – документознавство та інформаційна діяльність.

Іл. 3. Бібліогр.: 10 найм.

Відповідальний за випуск Г.Г. Швачич, канд. техн. наук, проф.

Рецензенти: Б. І. Мороз, д-р техн. наук, проф. (Академія таможенної Служби України)

Т. І. Пашова, канд. техн. наук, доц. (Дніпропетровський державний аграрний університет)

ТЕМА 1 ІНТЕРФЕЙС VISUAL BASIC FOR APPLICATION. СКЛАД І СТРУКТУРА ПРОЕКТУ СЕРЕДОВИЩА VBA. РОЗРОБКА ПРОЕКТУ, РЕЖИМ ДИЗАЙНУ Й ЧАСУ ВИКОНАННЯ

Visual Basic for Application (**VBA**) являє собою консольний додаток, котрий реалізовано комплексом динамічних й об'єктних бібліотек. До них належать: MSO97.DLL, Excel8.olb, Msword8.olb, Msppt8.olb й ін. Visual Basic for Application функціонує тільки в складі **Office** додатків, таких як: **Excel**, **Word**, **Access і т.д.**

Запуск **VBA** із середовища **Excel**, як і інших **Office** додатків, здійснюється за допомогою виконання команди меню **Сервіс \ Макрос \ Редактор Visual Basic** [**Alt + F11**] або за допомогою елемента управління, установленого в дочірню форму головного додатка, наприклад в **лист Excel**.

Примітка: Дочірня форма **MDI** (**Multiple Document Interface**) додатка – це його внутрішнє вікно, котре є представником класу **FORM**. Клас **FORM** – це особливий вид об'єкта. Він являється **власником** інших об'єктів. Щоб отримати доступ до елементів управління (**візуальним об'єктам VB**A), необхідно виконати команду меню **Сервіс \ Настроювання й вибрати в списку опцію – «Елементи управління»**.

1.1 Інтерфейс VBA

Інтерфейс редактора **VBA** складається з головної форми й дочірніх форм. Головна форма редактора **VBA** містить системне меню й панель управління. Панель управління, в свою чергу, містить кнопки команд, що дублюють основні команди системного меню. Важливими дочірніми вікнами інтерфейсу являються вікно проекту, вікно редактора властивостей об'єктів і вікна редактора тексту програми (редактора коду). Доступ до вікна проекту й вікна редактора властивостей об'єктів здійснюється за допомогою команд меню **Вид шляхом Вид \ Вікно проекту, або відповідно Вид \ Вікно властивостей**. Доступ до вікна редактора коду здійснюється шляхом **Вид \ Програма**, при умові вибору форми або модуля у вікні проекту, або подвійним щигликом миші над обра-

ним об'єктом у вікні проекту. Іншими, менш значимими вікнами інтерфейсу являються вікно налагодження й вікно локальних змінних. Доступ до них також здійснюється з меню **Вид**.

Основними командами запуску й налагодження програмного коду являються команди меню **Запуск,** такі як **«Запуск програми», «Перервати», «Скидання» й «Конструктор»**. Дані команди дубльовані відповідними кнопками, які розміщені в панелі управління **VBA**. Призначення перших трьох команд інтуїтивно зрозуміло. Команда «**Конструктор»** переводить **VBA** у режим дизайну.

Примітка: Головний додаток, наприклад **Excel**, разом з **VBA** можуть знаходитися у двох станах, а саме: у стані часу виконання (**Run Time Mode**) і в стані дизайну (**Design Time Mode**). Стан часу виконання – це звичайний режим роботи **Excel**, у якому виконуються обчислення й інші активні дії із середовища **Excel**. Стан дизайну (часу розробки) – це такий стан **Excel,** у якому із середовища **VBA** можливо здійснювати установку елементів управління до форм (**листі Excel**), візуально редагувати властивості елементів управління й виконувати розробку програми (писати програмний код у редакторі коду).

1.2 Проект VBA for Excel

За замовчуванням проект **VBA** містить модуль **Книги Excel** (**ThisWorkbook**) і модулі трьох відкритих **листів** книги **Excel**. Форма **ThisWorkbook** являється головною формою **Excel**. Модуль **ThisWorkbook** є головним модулем проекту. Даний модуль призначений для опису оточення проекту, а саме: оголошення додаткових бібліотек об'єктів (.olb), оголошення ActiveX управлінь (Active DLL) і динамічних бібліотек (DLL). Ними безпосередньо не користується **Excel** й **VBA,** але вони необхідні для функціонування розроблювальної програми. Також при використанні відповідних процедур обробки подій форми **ThisWorkbook** можна змінювати поводження **Excel** й ініціювати глобальні змінні при запуску додатка.

Кожна дочірня форма – **Excel** (лист) має власний модуль. Всі додаткові елементи управління, які установлені в **листі Excel,** належать даному листу. Програмний код обробки подій елементів управління реалізується в даному модулі.

Проект **VBA** може містити додаткові користувальницькі форми (**UserForm**), модулі й модулі класу. Додаткові форми в середовищі **VBA** основним чином застосовуються для створення власних діалогових вікон. Включати в проект додаткові модулі доцільно у випадках використання методів, які вироблені безпосередньо користувачем. Модулі класу призначені для опису нових типів даних (**Класів**) або оголошення властивостей цих класів, а також подій і методів.

Примітка: Додавання до складу проекту зазначених форм і модулів виконується командами меню **Вставка**.

1.3 Елементи управління

Для виконання дизайну додатка в середовищі **VBA** є наступні керуючі елементи, які розташовані в палітрі **«Елементи управління»**:

- Опція «Прапорець» (**CheckBox**) – призначена для визначення умов.
- Поле (**TextBox**) – призначено для вводу \ виводу даних.
- Кнопка (**CommandButton**) – призначена для виконання деяких дій.
- Опція «Перемикач» (**OptionButton**) – призначена для побудови залежного перемикача. Вона ніколи не використовується як один елемент.
- Список (**ListBox**) – поле для вибору даних зі списку, котрий в даний час прокручується.
- Поле зі списком (**ComboBox**) – поле для вибору даних зі спадаючого списку.
- Вимикач (**ToggleButton**) – кнопка, що має властивість «злипання».
- Лічильник (**SpinButton**) – стрілки напрямку рахунку.
- Смуга прокручування (**ScrollBar**) – використовується, як елемент завдання значень по положенню.
- Напис (**Label**) – використовується як статичний напис і для виводу повідомлень.

- Малюнок (**Image**) – об'єкт, який утримує малюнок. Також використовується як декоративна панель.
- Рамка (**Frame**) – декоративний елемент.

У палітру елементів управління можна додати додаткові зареєстровані в **Windows** керуючі елементи. Додавання елементів управління в палітру здійснюється кнопкою **«Додаткові елементи»** або командою вкладеного меню **«Додаткові елементи»**.

Установка елемента управління у форму здійснюється при виборі відповідного об'єкта в палітрі й переміщенні його образа у форму. Для одержання доступу до властивостей елемента управління останній повинен бути обраний у формі. У вікні додатка **Excel** доступ до властивостей об'єкта здійснюється за допомогою кнопки **«Властивості»**, яка розташована в панелі **«Елементи управління»**.

Редагувати властивості елементів управління можливо в режимі «**дизайн**» тільки у період виконання додатка. У режимі **«дизайн»** властивості елементів управління можна змінювати в редакторі властивостей, вибираючи значення відповідних полів. У період виконання додатка зміна значень властивостей виконується із програми.

Існують такі властивості об'єктів, які можна редагувати тільки в період дизайну програми або тільки в період виконання додатка. Наприклад: Значення властивості **Name** (Ім'я) можна змінити тільки при виконанні дизайну додатка, тому що ця властивість являється ідентифікатором об'єкта в період виконання програми.

ТЕМА 2 ВЛАСТИВОСТІ, ПОДІЇ Й МЕТОДИ ЕЛЕМЕНТІВ УПРАВЛІННЯ

2.1 Основні властивості елементів управління

Основною властивістю будь-якого об'єкта являється властивість **Name**. Властивість **Name** – це ідентифікатор об'єкта. Не може існувати двох і більше об'єктів, котрі мають однакове ім'я. Властивість **Name** можна редагувати (змінювати) тільки в період дизайну додатка. Як правило, об'є-

кту присвоюють унікальне ім'я. Воно відображає фізичну сутність або призначення об'єкта.

Елементи управління – це видимі об'єкти. Об'єкт, котрий відбиває який-небудь образ, обов'язково має наступні властивості, що відображені в таких кодах:

- **Height** – висота об'єкта.
- **Width** – ширина об'єкта.
- **Left** – розташування об'єкта ліворуч.
- **Top** – розташування об'єкта праворуч.
- **Caption** – заголовок об'єкта.
- **BackColor** – кольори об'єкта.
- **BackStyle** – зовнішній вигляд заголовка об'єкта (стиль накреслення).
- **ForeColor** – кольори напису заголовка об'єкта.

Елементи управління можуть описуватися й іншими, характерними тільки їх призначенню властивостями, що відображені в таких кодах:

- **Picture** – завантаження картинки.
- **MouseIcon** – завантаження образа вказівника миші.
- **MousePointer** – відображення виду вказівника миші, обираного зі стандартних образів.
- **WordWrap** – відображення напису в кілька рядків.

Іноді виникає необхідність, щоб елемент управління був доступний тільки при виконанні деяких умов. За доступ до об'єкта відповідає властивість **Visible**. За доступ до управління об’єктами відповідає властивість **Enabled**.

2.2 Події й оброблювачі подій

OS Windows являється системою, котра реагує на виникаючі події. Існує два види подій – це події, які обумовлені діями користувача, (користувальницькі події) і події, що виникають у процесі виконання програми (програмно-керовані події).

До користувальницьких подій відносяться події стосовно миші й клавіатури. Всі інші події являється програмно-керованими подіями.

Кожна подія повинна бути оброблена. В **VBA** визначені спеціальні процедури обробки подій. Заголовки таких процедур називаються оброблювачами подій.

Найпростішим і найбільш часто використовуваним оброблювачем подій є процедура обробки події **«щиглик миші по об'єкту»**, наприклад: **CommandButton1_Click()**.

Приклад:

```
Private Sub CommandButton1_Click()
'рядок програми, написаний користувачем'
End Sub
```

Як видно з конструкції процедури, подія не має ніяких параметрів. Вона передає управління програмному коду, котрий написав користувач.

Більш складні оброблювачі подій передають у програмний код користувача один або кілька параметрів, які являються необхідними для реалізації управління.

Приклад:

```
Private Sub Image1_MouseMove(ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Single, ByVal Y As Single)
'рядок програми, написаний користувачем'
End Sub
```

Наведена вище процедура обробки події, переміщення вказівника миші, передає програмному коду процедури код натиснутої клавіші клавіатури (якщо клавіша натиснута). Крім того розглянута процедура передає програмному коду координати вказівника миші в об'єкті. Вони обумовлені відносно верхнього лівого кута об'єкта.

Розглянемо призначення основних оброблювачів подій, які необхідні для створення користувальницької форми (**UserForm**).

- **UserForm_Activate** – використовується для ініціалізації полів, загальнодоступних змінних або установки значень властивостей об'єктів у момент, коли форма стає активною.
- **UserForm_BeforeDragOver** – використовується для перевірки можливості переміщення об'єкта перед виконанням операції Drag & Drop.
- **UserForm_BeforeDropOrPaste** – використовується для перевірки можливості залишення об'єкта наприкінці операції Drag & Drop.
- **UserForm_Click** – використовується для будь-яких дій при виконанні щиглика мишею у формі.
- **UserForm_DblClick** – використовується для будь-яких дій при виконанні подвійного щиглика мишею у формі.
- **UserForm_Deactivate** – використовується для звільнення пам'яті й інших дій у період, коли форма стає неактивною.
- **UserForm_Error** – використовується для обробки виняткової ситуації (коли відбулася помилка).
- **UserForm_Initialize** – використовується для ініціалізації полів, загальнодоступних змінних або установки значень властивостей об'єктів у момент, коли образ форми створюється в оперативній пам'яті.
- **UserForm_KeyDown** – використовується для обробки коду клавіші при її натисканні.
- **UserForm_KeyPress** – використовується для обробки коду клавіші при її втриманні.
- **UserForm_KeyUp** – використовується для обробки коду клавіші при її відпусканні.
- **UserForm_MouseDown** – використовується для виконання будь-яких дій, при натисканні на клавіші миші.
- **UserForm_MouseMove** – використовується для виконання будь-яких дій, при переміщенні курсору миші.
- **UserForm_MouseUp** – використовується для виконання будь-яких дій, при відпусканні клавіші миші.

- **UserForm_Resize** – використається для пропорційної зміни розмірів об'єктів, які розміщені у формі, при зміні розмірів форми.

2.3 Доступні методи

Крім прихованих методів, котрі реалізують управління властивостями й подіями об'єктів, в **VBA** є доступні користувачу методи, які стосуються елементів управління. Розглянемо деякі з них.

За допомогою методу **Show** користувач може вивести форму на екран.

Приклад:

UserForm.Show

За допомогою методу **Hide** користувач може сховати форму.

Приклад:

UserForm. Hide

За допомогою методу **PrintForm** користувач може вивести на друк образ форми із установленими в ній елементами управління.

Приклад:

UserForm1.PrintForm

Для кожного конкретного об'єкта в **VBA** визначені доступні методи, які реалізують ефективне управління.

ТЕМА 3 ВВІД І ВИВІД ДАНИХ В КЛІТИНКИ ЛИСТА EXCEL. ОФОРМЛЕННЯ ВИДУ КЛІТИНОК ЛИСТА EXCEL

3.1 Ввід і вивід даних в клітинки листа Excel

Ввід і вивід даних для клітинок листа Excel здійснюється за допомогою властивостей **Cells** (для однієї клітинки) і **Range** (для діапазону клітинок). Властивості **Cells** й **Range** являються об'єктами й містять вкладені властивості такі, як **Volue** (значення), **Interior** (інтер'єр, оформлення), **formula** (формула), **Select** (вибір) та інші.

Об'єкт **Cells** і вкладену в нього властивість **Select** можна розглядати одночасно як базовий об'єкт і як основну властивість листа Excel. Доступ до клітинки за допомогою властивості **Cells** здійснюється через індекси **R**

(Row – рядок) і **C** (Column – стовпець). Для того, щоб за допомогою програмного коду вибрати деяку клітинку досить у властивості **Cells** указати **RC** адресу необхідної клітинки і добавити властивість **Select**.

Приклад:

```
Cells(2,3).Select
```

Виконання зазначеного рядка програми виділить клітинку, яка знаходиться на перетинанні другого рядка й третього стовпця активного листа **Excel,** тобто передасть в їй фокус вказівника миші.

Властивість **Range** дозволяє отримати доступ до клітинки за допомогою запису її символьної адреси.

Приклад:

```
Range("C2").Select
```

Виконання зазначеного рядка програми передасть фокус вказівника миші в клітинку, яка знаходиться на перетинанні другого рядка й третього стовпця активного листа Excel, тобто буде обрана клітинка [C2].

Зверніть увагу на той факт, що символьна адреса клітинки є текстом. Вона береться в лапки.

Для того, щоб вибрати діапазон клітинок за допомогою властивості **Range** необхідно вказати початкову й кінцеву символьні адреси діапазону клітинок.

Приклад:

```
Range("C2:F5").Select
```

У властивості **Range** за допомогою властивості **Cells**, можна відобразити діапазон клітинок. Для цього необхідно у властивості **Cells** вказати **RC** адреси клітинок.

Приклад:

```
Range(Cells(2, 3), Cells(5, 6)).Select
```

Запис в клітинку деяких даних можна здійснити як з використанням вкладених властивостей **Value**, **Value2** й **Formula** так і без їх використання. Застосування вкладених властивостей трохи підвищує швидкодію опе-

рацій. Це обумовлюється точністю обчислень. Наприклад, для запису значення числа (**e**) в пам’ять клітинки з високою точністю (12 байт) варто застосувати властивість **Value2.**

Приклад:

```
Range("A1").Value2 = Exp(1)                [2,718281828]
```

Відображення значення числа (e), яке обчислене зі звичайною точністю буде мати ту ж кількість знаків, що й відображення значення числа (e), котре обчислене з підвищеною точністю (8 байт). В оперативній пам'яті клітинки, значення, котре обчислене з високою точністю буде містити більше розрядів після коми на 4 байти. Однакове подання числа в клітинці визначається форматом виводу значення.

Приклад:

```
Cells(2,1).Volue = Exp(1)                  [2,718281828]
```

Якщо виникає необхідність виконати обчислення безпосередньо в клітинці або скористатися функціями, представленими в майстру функцій **Excel** необхідно застосувати властивість **Formula**.

Приклад:

```
Range("A1").Formula = "=Sin(0.5)"
```

Наведений вище програмний код (властивість **Formula**), передає управління на пов'язане з клітинкою поле вводу формул (**Parser** – обчислювач). Запис формули в програмі виглядає еквівалентно вводу формул в **Excel**. Запис виразу необхідно взяти в лапки тому, що будь-яке поле вводу має тип «рядок».

У зв'язку з тим, що об'єкт **Cells** містить велику кількість методів, ряд з яких реалізує інтерпретацію виразів, для запису даних в клітинки й зчитування даних з клітинок, вкладені властивості **Volue** й **Formula** можна не застосовувати. У практичній роботі можна використовувати наступні програмні коди.

Приклад:

```
Cells(2,1) = Exp(1)
Range("A1") = "=Sin(0.5)"
Range("A3") = Range("A1") + Cells(2,1)
Cells(4,1) = Range("A3")*Cells(2,1)^2
```

Застосування властивостей **Range** й **Cells** у практичній роботі має ряд особливостей. Властивість **Range** доцільно використовувати у випадках, коли визначено, у яку клітинку необхідно записати дані або з якої клітинки слід вивести необхідні дані. Властивість **Cells** зручно використовувати для вводу і виводу масивів даних або коли адреса клітинки задана у вигляді посилання.

3.2 Оформлення виду клітинок листа Excel

Крім вводу й виводу даних для клітинок листа Excel можна також програмно оформити їх зовнішній вигляд. Це стосується як однієї клітинки, так й діапазону клітинок. Оформлення зовнішнього вигляду клітинок здійснюється за допомогою властивості **Interior** (інтер'єр) і вкладених в ній властивостей **Color** (кольори), **ColorIndex** (індекс кольорів), **Pattern** (візерунок) та інші. Зазначені властивості мають конкретні значення. Наприклад, властивості **Color** присвоює значення кольорів за допомогою символічного імені, яке відповідає двійковому коду в шістнадцятирічному поданні:

vbBlack	&H000000	Чорний
vbRed&	&H0000FF	Червоний
vbGreen	&H00FF00	Зелений
vbYellow	&H00FFFF	Жовтий
vbBlue	&HFF0000	Синій
vbMagenta	&HFF00FF	Рожевий
vbCyan	&HFFFF00	Блакитний
vbWhite	&HFFFFFF	Білий

Приклад:

```
Range("A1:A10").Interior.Color = vbGreen
```

Можна безпосередньо задати значення кольорів у шістнадцятирічному коді.

Приклад:

```
Range("B1:B10").Interior.Color = &HFF0000
```

Для ранніх версій **Excel**, наприклад **Office97, є о**собливість, при застосування властивості **Interior** у властивості **Cells.** Необхідно обов'язково здійснити попередній вибір клітинки.

Приклад:

```
Cells(1, 1).Select
Cells(1, 1).Interior.Color = &HFF0000
Cells(1, 2).Select
Cells(1, 2).Interior.Color = vbGreen
```

Кольори клітинки (діапазону клітинок) можна визначити через індекс палітри квітів. Для цього досить скористатися властивістю **ColorIndex**. Палітра кольорів клітинок містить 40 значень, які представлені індексами в діапазоні [0-39].

Приклад:

```
Cells(1, 1).Select
Cells(1, 1).Interior.ColorIndex = 3      'Червоний'
Range("A2").Interior.ColorIndex = 5  'Синій'
```

Визначити кольори клітинки (діапазону клітинок) можна також шляхом присвоєння властивості **Color** десяткового значення.

Приклад:

```
Cells(1, 1).Select
Cells(1, 1).Interior.Color = 255                'Червоний'
Range("a2").Interior.Color = 16711680       'Синій'
```

Діапазон клітинок можна залити візерунком. В **VBA** визначені 20 видів візерунків, а саме: xlPatternAutomatic, xlPatternChecker, xlPatternCrissCross, xlPatternDown, xlPatternGray16, xlPatternGray25, xlPatternGray50, xlPatternGray75, xlPatternGray8, xlPatternGrid, xlPatternHorizontal, xlPatternLightDown, xlPatternLightHorizontal, xlPatternLightUp, xlPatternLightVertical, xlPatternNone, xlPatternSemiGray75, xlPatternSolid, xlPatternUp, xlPatternVertical.

Приклад:

```
Range("A5").Interior.Pattern = xlPatternCrissCross
```

В клітинки можна виводити будь-які значення, у тому числі текст, котрий після його вводу слід оформити. Вид виведеної інформації здійснюється за допомогою властивості **font**.

Приклад:

```
Range("A6") = "Текст"
Range("A6").Font.Name = "Arial"
Range("A6").Font.Color = vbBlue
Range("A6").Font.Size = 12
Range("A6").Font.Bold = True
Range("A6").Font.Italic = True
```

В процесі написання програми часто доводиться в різних випадках користуватися однаковими властивості. Для поліпшення читаності програми, а також для скорочення її рядків в **VBA** є конструкція (**With…EndWith**) перерахування властивостей. Наведений вище запис фрагмента програми, з використанням зазначеної конструкції буде мати такий вигляд.

Приклад:

```
Range("A6") = "Текст"
With Range("A6").Font
Name = "Arial":    .Color = vbBlue:    .Size = 12
Bold = True:       .Italic = True
End With
```

В процесі роботи часто виникає необхідність програмно оформити зовнішній вигляд таблиці. Для промальовування контуру однієї клітинки, або діапазону клітинок існує властивості **Borders** й **Weight**.

Приклад:

```
Range("A1").Borders.Weight = xlHairline     тонкий контур
Range("A2").Borders.Weight = xlThin         середній контур
Cells(3, 1).Borders.Weight = xlMedium       напівжирний контур
Cells(4, 1).Borders.Weight = xlThick        жирний контур
```

Злиття клітинок можна виконати скориставшись властивістю **MergeCells**.

Приклад:

```
Range("A1:F1").MergeCells = True
```

Злиті клітинки також можна оконтурити.

Приклад:

```
Range("A1:F1").MergeCells = True
Range("A1:F1").Borders.Weight = xlMedium
```

Для доступу до інтелектуального підказувача вкладених властивостей й отримання відомостей про властивість **Cells** існує наступне правило. На початку необхідно написати слово Cells(). без вводу індексів і поставити точку. Потім зі списку властивостей і методів, що відкрився, вибрати необхідну властивість і поставити точку. Далі зі списку властивостей слід вибрати необхідну властивість, наприклад: Cells().Interior.Color = vbRed. Тільки після вибору всіх властивостей ввести в дужки властивості Cells індекс -> Cells(2,2). Interior.Color = vbRed.

Рядки програми в **VBA** уводяться тільки всередині процедури або функції.

Приклад:

```
Sub Test()
Cells(2,2) = Range("A1")
End Sub
```

Тема 4 Оголошення локальних змінних у середовищі VBA. Змінні типу Variant

Змінна в середовищі **VBA** може мати будь-яке ім'я за винятком службових символів і зарезервованих слів. Це відноситься до назв інструкцій, імен вбудованих функцій і процедур, імен констант мови програмування й ключових слів мови програмування. Ім'я змінної може містити будь-які символи. Наприклад, цифри, літерні позначення, а також знак підкреслення. З метою поліпшення читаності програми, змінній доцільно привласнювати ім'я конкретного значення, наприклад: **Axis** (координата X), **Axis** (координата Y), **Count** (кількість рахунку). Ім'я змінної не повинне містити пробілів.

Для оголошення змінної в **VBA** використовується інструкція **Dim**. Змінна може бути простою. У цьому випадку вона призначена для збері-

гання чисел, тексту, масивів, які містять значення рядків, а також таблиць даних, у яких описуються об'єкт і його властивості. За замовчуванням у середовищі **VBA** проста змінна має тип **Variant**. Це такий строковий тип даних, котрий може приймати будь-які значення або послідовність значень. У процесі присвоєння конкретних значень змінна типу **Variant** приймає той тип даних або послідовність типів даних, які їй визначені.

Змінна повинна бути оголошена до її застосування. Якщо змінній не привласнене ніяке значення до її застосування, то вона приймає значення нуль. При оголошенні змінної типу **Variant** її тип можна не вказувати. Локальна змінна буде доступна тільки в тій процедурі або функції, у якій вона оголошена.

Приклад:

```
Sub Demo()
Dim Axis, Axis, Count
Axis = 100.5: Axis = 200.5: Count = 0
Range("B1") = Axis: Range("B2") = Axis: Range("B3") = Count
End Sub
```

Вище наведена процедура Demo (програма Demo). У першому рядку програми оголошені прості змінні **Axis**, **Axis** й **Count**. У другому рядку тексту програми зазначеним змінним привласнені конкретні значення. Третій рядок програми записує значення змінних у зазначені клітинки листа Excel.

Змінна типу **Variant** може бути масивом, що оголошений у явному, або неявному виді. Нижче наведений приклад оголошення змінної у вигляді одномірного масиву й присвоєння значень різним типам елементів цього масиву.

Приклад:

```
Sub Test()
Dim A(3)
A(0) = "Кількість елементів"
```

```
A(1) = " = "
A(2) = 4
A(3) = "у масиві A(3)"
Range("A1") = A(0) & A(1) & A(2) & A(3)
End Sub
```

У першому рядку процедури **Test** оголошений одномірний масив A(3), типу **Variant**. Він містить 4 елементи (початковий індекс масиву = 0). У другому рядку процедури першому елементу масиву привласнене значення тексту. Елементи масиву містять строковий тип даних. У третьому рядку процедури другий елемент масиву отримує значення знака (=), як строкового типу даних. У четвертому рядку процедури третій елемент масиву одержує значення цілого числа, яке дорівнює 4. У п'ятому рядку процедури четвертому елементу масиву привласнюється текст "у масиві A(3)". Це строковий тип даних.

Останній рядок процедури виведе в клітинку [A1] злиті дані різних типів, як текстове повідомлення: {Кількість елементів = 4 у масиві A(3)}.

Зверніть увагу на амперсант [&], що застосовується для злиття строкового типу даних з іншими типами даних. У результаті злиття утвориться строковий тип даних.

Оголошення двовимірного масиву типу **Variant** і присвоєння значень даному масиву виглядає таким чином.

Приклад:

```
Sub Test()
Dim B(2,2)
B(0,0) = "1стор. 1ст.": B(0,1) = "1стор. 2ст.": B(0,2) = "1стор. 3ст."
B(1,0) = "2стор. 1ст.": B(1,1) = "2стор. 2ст.": B(1,2) = "2стор. 3ст."
B(2,0) = "3стор. 1ст.": B(2,1) = "3стор. 2ст.": B(2,2) = "3стор. 3ст."
End Sub
```

Просту змінну типу **Variant** можна оголосити в одномірному масиві заново і присвоювати їй різні значення елементів даного масиву. Для цього використовуючи функцію **Array**.

Приклад:

```
Sub Test()
Dim Dcolor
DColor = Array(vbRed, vbGreen, vbBlue, vbMagenta, vbYellow, vbCyan)
End Sub
```

Масив типу Variant можна оголосити в явному виді як:

```
Dim MyXYZ(1 To 10, -5 To 15, 10 To 20)
```

Оголошення масиву в неявному виді здійснюється при розробці власних методів (процедур і функцій), коли спочатку невідомий розмір масиву. Якщо масив оголошується в неявному виді в тілі процедури (програми), то надалі він оголошується заново в явному виді при використанні інструкції **ReDim**.

Приклад:

```
Sub Test()
Dim N()             'Оголошення масиву в неявному виді
Dim Count           'Оголошення змінної
Count = 10          'Визначення значення змінної
ReDim N(Count)      'Визначення розмірів масиву
End Sub
```

ТЕМА 5 ТИПИ ДАНИХ І ФУНКЦІЇ. ПЕРЕТВОРЕННЯ ТИПІВ ДАНИХ

Розглянемо типи даних та їх основні характеристики, які визначені в Visual Basic for Application.

Тип Variant

Змінна **Variant** являється особливим типом даних. Змінні цього типу можуть містити будь-які дані, за винятком тих типів даних, котрі обумовлені користувачем. Змінна типу **Variant** може також містити спеціальні значення **Empty** (порожньо), **Error** (помилка), **Nothing** (нічого) і Null (не дійсний). Указати характер підтипів типу **Variant** дозволяють функції **VarType** або **TypeName.**

Припустимими числовими даними є будь-які цілі або дійсні числа в діапазоні від -1,797693134862315E308 до -4,94066E-324 для негативних значень, і від 4,94066E-324 до 1,797693134862315E308 для позитивних значень. У загальному випадку, числові дані типу **Variant** зберігають свій вихідний тип усередині типу **Variant**. Наприклад, якщо привласнити змінній типу **Variant** значення типу **Integer**, то в наступних операціях **Variant** трактується як **Integer.** Якщо арифметична операція, виконана над змінною типу Variant, котра містить значення типу **Byte, Integer, Long** або **Single** і це приводить до того, що результат виходить за межі діапазону припустимих значень вихідного типу, то результат перетворюється до наступного більш широкого типу усередині **Variant**. Значення типу **Byte** перетвориться до типу **Integer**, тип **Integer** перетвориться до типу **Long**, а значення типу **Long** й **Single** перетворяться до типу **Double**. Помилка виникає, якщо за межі припустимого діапазону значень виходять змінні типу **Variant**, котрі містять значення типу **Currency**, **Decimal** або **Double**.

Користувач має можливість використати тип **Variant** замість будь-якого типу даних, щоб забезпечити більшу гнучкість при обробці даних. Якщо змістом змінної типу **Variant** являються цифри, то вони в різному контексті можуть розглядатися або як строкове подання числа, або як число.

Тип Boolean

Змінні типу Boolean – це логічні значення, які зберігаються як 16-розрядні (двох байтові) числа. Але вони можуть мати тільки значення True (істина) або False (неправда)

Тип Byte

Змінні типу Byte зберігаються як 8-розрядні (1 байт) числа без знака в діапазоні від 0 до 255. Тип даних Byte використовується для запису двійкових значень.

Тип Integer

Змінні типу Integer (цілі) зберігаються як 16-розрядні (двох байтові) числа в діапазоні від -32 768 до 32 767.

Тип Long

Змінні типу Long (довге ціле) зберігаються як 32-розрядні (чотирьох байтові) числа зі знаком у діапазоні від -2 147 483 648 до 2 147 483 647

Тип Single

Змінні типу Single – це числа із плаваючою точкою звичайної точності. Вони зберігаються як 32-розрядні (чотирьох байтові) числа із плаваючою точкою стандарту IEEE у діапазоні від -3,402823E38 до -1,401298E-45 для негативних значень, і від 1,401298E-45 до 3,402823E38 для позитивних значень.

Тип Double

Змінні типу Double – це числа із плаваючою точкою подвійної точності. Вони зберігаються як 64-розрядні (восьми байтові) числа із плаваючою точкою стандарту IEEE у діапазоні від -1,79769313486232E308 до -4,94065645841247E-324 для негативних значень, і від 4,94065645841247E-324 до 1,79769313486232E308 для позитивних значень.

Тип Decimal

Змінні типу Decimal зберігаються як 96-розрядні (12-байт) цілі без знака, масштабовані ступенями 10. Ступінь масштабування визначає число знаків дробової частини, що може змінюватися від 0 до 28. Для ступеня масштабування 0 (числа без дробової частини) максимальними по абсолютній величині значеннями є +/-79 228 162 514 264 337 593 543 950 335. При 28 знаках дробової частини максимальними по абсолютній величині значеннями являється +/-7,9228162514264337593543950335, а мінімальними +/-0,0000000000000000000000000001.

Тип String

Існує два типи строкових значень:

- Рядка змінної довжини, які можуть містити приблизно до 2 мільярдів (2^31) символів.
- Рядка постійної довжини, які можуть містити від 1 до 64K (2^16) символів.

Кодами для символів, що утворять значення типу String, служать цілі числа в діапазоні від 0 до 255. Перші 128 символів (0-127) набору символів відповідають буквам і символам стандартної американської клавіатури. Ці перші 128 символів збігаються з набором символів ASCII. Наступних 128 символів (128-255) представляють букви національних алфавітів, букви з надрядковими символами, символи грошової одиниці й дробу.

Тип Date

Змінні типу Date (значення дати й часу) зберігаються як 64-розрядні (восьмибайтові) числа із плаваючою точкою стандарту IEEE, що представляють дати в діапазоні від 1 січня 100 р. до 31 грудня 9999 р. і значення часу від 0:00:00 до 23:59:59. Змінним типу Date можуть бути привласнені будь-які значення, котрі задають розпізнавання дат в явному поданні.

Тип Currency

Змінні типу Currency (грошові значення) зберігаються як 64-розрядні (восьмибайтові) цілі числа, які після розподілу на 10000 дають число з фіксованою десятковою точкою та з п’ятнадцятью розрядами в цілій частині й чотирма розрядами в дробовій частині. Таке подання дозволяє відобразити числа в діапазоні від -922 337 203 685 477,5808 до 922 337 203 685 477,5807.

Тип Object

Змінні типу Object зберігаються як 32-розрядні (чотирьох байтові) адреси, у яких утримуються посилання на об'єкти. Змінній, описаній типом Object, можна за допомогою інструкції Set привласнити посилання на будь-який об'єкт, створений у додатку.

Для оголошення змінних як явних типів даних на рівні модуля використовується інструкція **Option Explicit**. Дана інструкція повинна бути зазначена в розділі [GENERAL ЗАГАЛЬНА ОБЛАСТЬ] [DECLARATION ОПИС] модуля.

Для оголошення змінних, як явного типу даних, використовується спеціальна інструкція **AS**.

Приклад:

```
Option Explicit
--------------------------------------------------------------------------------
Sub Test()
Dim Color as Byte
Color = 127
End Sub
```

У наведеному вище прикладі змінна Color оголошена як тип Byte, тобто може приймати тільки цілі значення в діапазоні 0 – 255.

Аналогічно указуються типи даних для простих змінних і масивів:

Dim a **as** Integer: **Dim** MA() **as** Double

Dim b **as** Long: **Dim** MB(10) **as** Single

Dim c **as** String: **Dim** MC(100,100) **as** Byte

Для роботи зі змінною, котра утримує конкретний тип даних, існують спеціальні функції перетворення типів даних. До них належать: CBool(), CByte(), CCur(), CDate(), CDbl(), CDec(), CInt(), CLng(), CSng(), CVar(), CStr(). У дужках функцій указується значення змінної, або значення виразу.

Наведений нижче приклад демонструє застосування функції CBool для перевірки значень змінних A й B.

Приклад:

```
Sub Test()
Dim A As Byte
Dim B As Byte
Dim Check As Boolean
A = 5: B = 5
Check = CBool(A = B)
Range("A1") = Check          'Значення істина
A = 0
Check = CBool(A)
Range("A2") = Check          'Значення неправда
End Sub
```

Наведений нижче приклад демонструє застосування функції CByte для перетворення типу Double у тип Byte.

Приклад:

```
Sub Test
Dim MyDouble As Double: Dim MyByte As Byte
MyDouble = 125.5678          'MyDouble має тип Double.
MyByte = CByte(MyDouble)     'MyByte містить значення 126.
End Sub
```

Наведений нижче приклад демонструє застосування функції CCur для перетворення типу Double у тип Currency.

Приклад:

```
Sub Test
Dim MyDouble As Double: Dim MyCurr As Currency
MyDouble = 543.214588          'MyDouble має тип Double.
MyCurr = CCur(MyDouble * 2) 'Перетворить результат виразу
'MyDouble * 2 (1086.429176) в 'значення типу Currency
(1086.4292)
End Sub
```

Наведений нижче приклад демонструє застосування функції CDate для перетворення типу String у тип Date.

Приклад:

```
Sub Test
Dim MyDate As String: Dim MyShortDate As Date
Dim MyTime As String: Dim MyShortTime As Date
MyDate = "February 12, 1969"          'Визначає дату.
MyShortDate = CDate(MyDate)           'Перетворить до типу
                                      Date.
MyTime = "4:35:47 PM"                 'Визначає час.
MyShortTime = CDate(MyTime)           'Перетворить до типу
                                      Date.
End Sub
```

Наведений нижче приклад демонструє застосування функції CInt для перетворення типу Double у тип Integer.

Приклад:

```
Sub Test()
Dim MyDouble As Double: Dim MyInt As Integer
MyDouble = 2345.5678          'MyDouble має тип Double.
MyInt = CInt(MyDouble)        'повертає значення 2346.
End Sub
```

Наведений нижче приклад демонструє застосування функції CLng для перетворення типу Single у тип Long.

Приклад:

```
Sub Test()
Dim MyVal1 As Single: Dim MyVal2 As Single
Dim MyLong1 As Long: Dim MyLong2 As Long
    MyVal1 = 25427.45:
    MyVal2 = 25427.55 'MyVal1 й MyVal2 мають тип Single.
    MyLong1 = CLng(MyVal1)      'MyLong1 містить 25427.
    MyLong2 = CLng(MyVal2)      'MyLong2 містить 25428.
End Sub
```

Наведений нижче приклад демонструє застосування функції CSng для перетворення типу Double у тип Single.

Приклад:

```
Sub Test()
Dim MyDouble1 As Double
Dim MyDouble2 As Double
Dim MySingle1 As Single
Dim MySingle2 As Single 'MyDouble1 й MyDouble2 маю тип
                        Doubles.
    MyDouble1 = 75.3421115: MyDouble2 = 75.3421555
    MySingle1 = CSng(MyDouble1) 'MySingle1 місти 75.34211.
    MySingle2 = CSng(MyDouble2) 'MySingle2 містить
                                75.34216.
End Sub
```

Наведений нижче приклад демонструє застосування функції CVar для перетворення типів Integer й String у тип String зі злитими даними.

Приклад:

```
Sub Test()
Dim MyInt As Integer
Dim MyVar As String
    MyInt = 4534                    'MyInt має тип Integer.
    MyVar = CVar(MyInt & "000")     'MyVar містить рядок
                                    4534000.
End Sub
```

Наведений нижче приклад демонструє застосування функції CSng для перетворення типу Double у тип String.

Приклад:

```
Sub Test()
Dim MyDouble As Double
Dim MyString As String
MyDouble = 437.324              'MyDouble має тип Double.
MyString = CStr(MyDouble)       'MyString містить "437.324".
End Sub
```

Вказівка інструкції Option Explicit й оголошення змінних, як явних типів даних, значно знижує ресурси ЕОМ і час на реалізацію програми, а також дозволяє транслювати вихідний текст програми, розробленої в середовищі VBA в інші мови програмування, такі як C++ й Object Pascal.

Якщо виникає необхідність використовувати значення функції, яка обчислена в одній процедурі, в іншій процедурі одного і того модуля, то таку змінну необхідно оголосити як загальнодоступну змінну. Загальнодоступні змінні для конкретного модуля оголошуються в розділі [GENERAL ЗАГАЛЬНА ОБЛАСТЬ] [DECLARATION ОПИС] модуля.

Приклад:

```
Option Explicit
Dim MyVol As Variant   'Оголошення загальнодоступної змінної MyVol
Sub Init()
    MyVol = Sqr(Sin(0.5))    'Ініціалізує змінну MyVol
End Sub
Sub Test()
Dim a As Variant
a = Cos(0.5)                 'Ініціалізує змінну MyVol
Range("A1") = MyVol * a      'Виводить значення в клітинку A1
End Sub
```

У наведеному прикладі в процедурі Init обчислюється значення загальнодоступної змінної MyVol. Значення цієї змінної використовується в процедурі Test.

ТЕМА 6 АРИФМЕТИЧНІ Й ІНШІ НАЙБІЛЬШЕ ЧАСТО ЗАСТОСОВУВАНІ ФУНКЦІЇ В МОВІ ПРОГРАМУВАННЯ СЕРЕДОВИЩА VBA. ОСНОВНІ КОНСТРУКЦІЇ МОВИ ПРОГРАМУВАННЯ СЕРЕДОВИЩА VBA

6.1 Перелік функцій VBA

Середовище **VBA** надає користувачу наступні вбудовані математичні функції:

- Функція **Abs** – Повертає абсолютне значення указаного числа.
- Функція **Atn** – Повертає значення арктангенса числа.
- Функція **Cos** – Повертає значення косинуса кута.
- Функція **Exp** – Повертає значення зведення числа **E (exp)** в указаний ступінь.
- Функція **Fix** – Повертає значення аргументу, що утримує цілу частину числа, котре більше, або дорівнює указаному.
- Функція **Int** – Повертає значення аргументу, що утримує цілу частину числа, котре менше, або дорівнює указаному.
- Функція **Log** – Повертає значення логарифма числа.
- Функція **Rnd** – Повертає значення випадкового числа.
- Функція **Sgn** – Повертає знак указаного числа.
- Функція **Sin** – Повертає значення синуса кута.
- Функція **Sqr** – Повертає значення квадратного кореня указаного числа.
- Функція **Tan** – Повертає значення тангенса кута.

6.2 Приклади використання деяких функцій

Приклад:

```
Sub Test()
Dim MyNumber
    MyNumber = Abs(50.3)          'Повертає 50.3.
    MyNumber = Abs(-50.3)         'Повертає 50.3.
End Sub
```

У наведеному прикладі функція **Abs** використана для повернення абсолютного значення числа.

Приклад:

```
Sub Test()
Dim MyAngle, MyHSin
    MyAngle = 1.3        'Задає кут у радіанах.
    MyHSin = (Exp (MyAngle) - Exp(-1 * MyAngle)) / 2 'Обчислює гіперболічний синус.
End Sub
```

У наведеному вище прикладі функція **Exp** використана для зведення числа (**e**) в указаний ступінь.

Приклад:

```
Sub Test()
Dim MyNumber
    MyNumber = Int(99.8)      'Повертає 99.
    MyNumber = Fix(99.2)      'Повертає 99.
    MyNumber = Int(-99.8)     'Повертає -100.
    MyNumber = Fix(-99.8)     'Повертає -99.
    MyNumber = Int(-99.2)     'Повертає -100.
    MyNumber = Fix(-99.2)     'Повертає -99.
End Sub
```

У наведеному вище прикладі показано, як функції **Int** й **Fix** повертають цілі частини чисел. У випадку негативного аргументу функція **Int** повертає найближче негативне ціле число, котре менше, або дорівнює указаному. Функція **Fix** повертає найближче негативне ціле число, котре більше, або дорівнює указаному.

Приклад:

```
Sub Test()
Dim MyValue
    MyValue = Int((6 * Rnd) + 1)     'Повертає випадкове число
                                     від 1 до 6.
End Sub
```

У наведеному вище прикладі функція **Rnd** використана для одержання випадкового цілого числа в діапазоні від 1 до 6.

Функція **Rnd** повертає значення числа, яке менше одиниці і більше або дорівнює нулю. Аргумент функції **Rnd** визначає спосіб генерації випадкового числа. При використанні однакових опорних чисел виходять однакові послідовності випадкових чисел, оскільки при генерації кожного наступного члена послідовності використовується попередній член.

Перед викликом функції **Rnd** для ініціалізації генератора випадкових чисел використовується інструкція **Randomize** без аргументу.

Нижче наведена формула призначена для одержання випадкових цілих чисел у заданому діапазоні: **Int((верхня Границя – нижня Границя + + 1) * Rnd + нижня Границя). Верхня Границя** становить максимальне число в заданому діапазоні, **нижня Границя** – мінімальне число.

Примітка. Для повторення послідовності випадкових чисел, відразу після використання інструкції **Randomize** із числовим аргументом, варто викликати функцію **Rnd** з негативним аргументом. Повторне використання інструкції **Randomize** з тим же числовим аргументом не приведе до повторення попередньої послідовності випадкових чисел.

Приклад:

```
Sub Test()
Dim MyValue
Randomize          'Ініціалізує генератор випадкових чисел
    MyValue = Int((6 * Rnd) + 1)    'Повертає випадкове число
                                    від 1 до 6.
End Sub
```

У наведеному вище прикладі інструкція **Randomize** використана для ініціалізації генератора випадкових чисел.

Приклад:

```
Sub Test()
Dim MyVar1, MyVar2, MyVar3, MySign
    MyVar1 = 12: MyVar2 = -2.4: MyVar3 = 0
    MySign = Sgn(MyVar1)          'Повертає 1.
    MySign = Sgn(MyVar2)          'Повертає -1.
    MySign = Sgn(MyVar3)          'Повертає 0.
End Sub
```

У наведеному вище прикладі функція **Sign** використана для визначення знака числа.

У процесі роботи з об'єктом **«лист Excel»** часто виникає необхідність перевірки значень змінних в клітинках Excel. Для цього в **VBA** існує кілька вбудованих функцій:

- **IsNill** – Показує, чи є результатом виразу порожнє значення (**Null**).
- **IsEmpty** – Показує, чи була ініціалізована змінна.
- **IsNumeric** – Показує, чи має вираз числове значення.
- **IsError** – Показує, чи містить вираз помилку.

В **VBA** існують й інші можливості перевірки значень змінних в клітинках Excel. Розглянемо на прикладах застосування програмної конструкції **If … Then … Else … End If**...із використанням указаних функцій.

Приклад:

```
Sub Test()
If IsEmpty(Range("A1")) = True Then
   Range("B1") = "Порожньо"
Else
   Range("B1") = "Значення"
End If
End Sub
```

У наведеному вище прикладі перевіряється значення клітинки [A1]. Якщо клітинка [A1] не ініціалізована тобто є порожньою, то в клітинку [B1] записується повідомлення «Порожнє». У противному випадку, якщо клітинка [A1] містить деяке значення, в клітинку [B1] буде виведене повідомлення «Значення».

Приклад:

```
Sub Test()
If IsNumeric(Range("A1")) = True Then
   Range("B1") = "Число"
Else
   Range("B1") = "Не число"
End If
End Sub
```

У наведеному вище прикладі перевіряється значення клітинки [A1]. Якщо клітинка [A1] містить деяке число, то в клітинку [B1] записується повідомлення «Число». У противному випадку, якщо клітинка [A1] містить значення, котре не являється числом, у клітинку [B1] буде виведене повідомлення «Не число».

Розглянемо докладніше конструкцію **If … Then … Else … End If**... Як видно з наведених прикладів, дана конструкція перевіряє деяке значення. Якщо значення відповідає умові, котра записана між інструкцією **If** і ключовим словом **Then**, то виконуються рядки програми, що знаходяться між ключовими словами **Then** й **Else**. Якщо значення не відповідає заданій умові, то виконуються рядки програми, що знаходяться між ключовим словом **Else** і завершенням дії ключового слова **If** (**End If**). У загальному випадку синтаксис програмної конструкції **If … Then … Else … End If** має вигляд:

If умова **Then**
  [інструкції]
  [ElseIf умова-n **Then**
  [інструкції_elseif] ...
[**Else**
  [інструкції_else]]
End If

Наведена вище конструкція дозволяє ефективно управляти ходом виконання програми, а саме: на основі аналізу даних створювати обчислювальні процеси, що розгалужуються; управляти поведінкою об'єктів і передавати управління в інші процедури створюваної програми.

Якщо при не виконанні заданої умови виникає необхідно не виконувати які-небудь дії, то конструкція може мати скорочений запис.

Приклад:

```
Sub Test()
If IsError(Range("A1")) = True Then Range("B1") = "помилка"
End Sub
```

У наведеному вище прикладі перевіряється наявність помилки у веденій в клітинку [A1] формулі. Якщо формула містить помилку, то в клітинку [B1] буде виведене повідомлення «помилка». Якщо формула вірна, то ніяких повідомлень виводиться не буде.

Перевірка умов може виконуватися з використанням складних реляційних виразів.

Приклад:

```
Sub Test()
If IsError(Range("A1")) = True Or IsEmpty(Range("A1")) = True
Then _
   Range("B1") = "не вірна дія"
End Sub
```

У наведеному вище прикладі перевіряється умова, при котрій в клітинку [A1] не вірно введена формула, або в ній зовсім відсутній будь-який запис. У випадку виконання умови в клітинку [B1] виводиться повідомлення «не вірна дія».

При перевірці умов можуть використатися наступні реляційні оператори й вирази: **<, >, =, <>, And, Or, Xor, Not**, а також функції й вирази, що повертають логічні значення.

У процесі розробки програм часто виникає необхідність заповнення таблиць значеннями функцій й значеннями обробки масивів даних. Для рішення таких задач у VBA є програмна конструкція циклу типу лічильник **For …To … Steep … Next**... У загальному випадку синтаксис конструкції має такий вигляд:

```
For лічильник = початок To кінець [Step крок]
      [інструкції]
      [Exit For]
      [інструкції]
Next [лічильник]
```

Розглянемо на прикладах практичне застосування даної конструкції.

Задача. необхідно обчислити значення функції виду y = f(x) з кроком зміни s = 0,01 у діапазоні значень [–1;+1]. Результат обчислень необхідно вивести в перший стовпець листа Excel. Нехай задана функція **y = Sin(x).**

Приклад:

```
Sub Test()
Dim i, AC
AC = 1        'присвоєння початкового значення покажчику адреси
For i = -1 To 1 Step 0.01
AC = AC + 1        'поточне значення покажчика адреси
Cells(AC, 1) = Sin(i) 'висновок значень функції
Next i
End Sub
```

Пояснимо роботу вищенаведеної процедури. Змінна AC є вказівником рядка клітинок листа Excel. Їй присвоюється початкове значення = 1. При виконанні кожного кроку циклу значення змінної AC збільшується на 1. На першому кроці циклу змінній **(i)** присвоюється початкове значення рівне –1, котре збільшується в кожному наступному кроці циклу на 0,01. Значення змінної **(i)** являється аргументом функції **Sin(i)**. Цикл завершується, коли значення змінної **(i)** досягне 1.

Задача. Необхідно заповнити випадковими числами сто клітинок листа Excel, які знаходяться у другому стовпці.

Наступний приклад демонструє реалізацію поставленої задачі.

Приклад:

```
Sub Test()
Dim i
Randomize
    For i = 1 To 100
        Cells(i, 2) = Rnd()
    Next i
End Sub
```

Наведений приклад демонструє, що цикл типу лічильник ефективно застосовувати у випадку, коли заздалегідь відома кількість циклів, котрі необхідні для реалізації задачі.

Примітка: У конструкції циклу типу лічильник ключове слово **Step** можна не вказувати, якщо змінна циклу збільшується на 1.

Задача. Заповнити таблицю розміром 10 клітинок випадковими числами, котрі змінюються в діапазоні значень від -500 до 500.

Наступний приклад демонструє реалізацію поставленої задачі.

Приклад:

```
Sub Test()
Dim i, j
Randomize
    For j = 1 To 10
    For i = 1 To 10
            Cells(i, j) = 500 - Int(Rnd() * 1000)
    Next i
    Next j
End Sub
```

Для рішення задачі необхідно застосувати дві конструкції циклів типу лічильник. Цикл зі змінною **(i)** являється внутрішнім циклом і повністю виконується для кожного кроку зовнішнього циклу зі змінною **(j)**.

Лист Excel є таблицею, тобто двовимірним масивом. Тому, немає ніякої різниці при застосуванні наведених конструкцій для обробки даних будь-яких одномірних і двовимірних масивів.

У практичній роботі нерідко зустрічаються задачі, котрі пов'язані зі знаходженням заданих значень даних при невідомому числі кроків, за які шукане значення може бути досягнуто. Для реалізації подібних задач в VBA є керуючі структури, що реалізують цикли з перевіркою значень на вході й виході із циклу.

Наступний приклад демонструє роботу керуючої структури **While … Wend**...

Приклад:

```
    Sub Test()
  Dim    Count
         Count = 0          'Ініціалізує змінну.
  While  Count < 20         'Аналізує значення лічильника.
         Count = Count + 1  'Збільшує лічильник.
  Wend          'Завершує цикл While, якщо Count > 19.
    Debug. Print Count      'Виводить 20 у вікно налагодження.
    End Sub
```

У даному прикладі конструкція **While...Wend** використовується для збільшення лічильника. Інструкції в циклі будуть виконуватися доти, поки зазначена умова не буде виконана (стане дійсною – **True)**.

Конструкція **Do...Loop** використається для виконання наборів інструкцій невизначене число раз. Набір інструкцій повторюватиметься, доки умова не прийме значення **True**.

Є два способи перевірки умови в керуючій структурі **Do...Loop:**

- За допомогою ключового слова **While** – умова перевіряється до входу в цикл.
- Умова перевіряється після хоча б однократного виконання циклу.

У наступній процедурі **ChkTest** умова перевіряється до входу в цикл. Якщо **Num** задати рівним 9 замість 20, інструкції усередині циклу виконуватися не будуть.

Приклад:

```
Sub ChkTest()
    count = 0
    Num = 20
    Do While myNum > 10
        Num = myNum - 1
        count = count + 1
    Loop
    MsgBox "Виконано" & count & "ітерацій циклу."
End Sub
```

У процедурі **ChkLast** інструкції усередині циклу виконуються тільки один раз перше ніж умова прийме значення **False**.

Приклад:

```
Sub ChkLast()
    count = 0
    Num = 9
    Do
        Num = Num - 1
        count = count + 1
    Loop While Num > 10
    MsgBox "У циклі виконане " & counter & "ітерацій."
End Sub
```

Є два способи перевірки умови в конструкції **Do...Loop** за допомогою ключового слова **Until**:

- Умова перевіряється до входу в цикл (процедура **Sub** ChkFirstUntil()).
- Умова перевіряється після хоча б однократного виконання циклу (Процедура **Sub** ChkLastUntil()).

Приклад:

```
    Sub ChkFirstUntil()
    count = 0
    Num = 20
    Do Until Num = 10
        Num = myNum - 1
        count = count + 1
    Loop
    MsgBox "У циклі виконане" & counter & "ітерацій"
End Sub
```

Приклад:

```
Sub ChkLastUntil()
    count = 0
    Num = 1
```

```
Do
        Num = Num + 1
        count = count + 1
Loop Until Num = 10
MsgBox "У циклі виконане" & counter & "ітерацій"
End Sub
```

Інструкції **Do...Loop** можна завершити за допомогою інструкції **Exit Do**. Наприклад, для завершення нескінченного циклу використовується інструкція **Exit Do If...Then...Else**. Якщо умова має значення **False**, цикл буде виконуватися як звичайно.

У наступному прикладі змінній **Num** присвоюється значення, котре приводить до нескінченного циклу. У блоці **If...Then...Else** перевіряється умова **Num**, а потім завершується виконання інструкції **Do...Loop**.

Приклад:

```
Sub ExitExample()
count = 0
Num = 9
Do Until Num = 10
        Num = myNum - 1
        count = count + 1
        If Num < 10 Then Exit Do
Loop
MsgBox "У циклі виконане " & count & "ітерацій"
End Sub
```

Примітка. Для припинення нескінченного циклу використовуються клавіші ESC або CTRL+BREAK.

У наведеному нижче прикладі показано, як можна використати інструкції **Do...Loop**. Внутрішній цикл **Do...Loop** виконується 10 разів. Потім логічній змінній присвоюється значення **False**, після чого цикл передчасно завер-

шується за допомогою інструкції **Exit Do**. Зовнішній цикл завершується негайно після перевірки значення логічної змінної.

Приклад:

```
Sub Test()
Dim Check, Count
Check = True: Count = 0     'Ініціалізує змінні.
Do          'Зовнішній цикл.
   Do While Count < 20      'Внутрішній цикл.
   Count = Count + 1        'Збільшує лічильник.
   If Count = 10 Then       'Якщо умова істинно.
   Check = False            'Привласнює змінній значення False.
   Exit Do                  'Завершує внутрішній цикл.
   End If
   Loop
Loop Until Check = False    'Негайно завершує зовнішній цикл.
End Sub
```

ТЕМА 7 МЕТОДИ, СТВОРЕНІ КОРИСТУВАЧЕМ У СЕРЕДОВИЩІ VBA

Методи – це процедури й функції, які необхідні для виконання визначених дій. Процедури та функції дозволяють багаторазово використовувати один і той самий програмний код, тобто оптимізувати розроблювальну програму, або застосовувати раніше реалізовані алгоритми обробки інформації в різних програмах і додатках.

У середовищі **VBA** методи, залежно від їх застосування або призначення, описуються в модулі об'єкта «**Книга»**, модулі об'єкта «**Лист»**, модулі «**Форми»**, «**Зовнішньому модулі»** або декларуються в **модулі Класу**. Процедури, які розроблені користувачем, повинні бути декларовані й описані в розділі [**Загальна область** (**General**)], [**Опис** (**Declaration**)].

Процедури й функції необхідно писати вручну, а саме:

1. Написати ключове слово **Private** або **Public** залежно від області видимості методу.

2. Написати інструкцію **Sub** (процедура) або **Function** (функція) залежно від виду методу.
3. Написати **Ім'я** методу. **Ім'я** методу повинне бути унікальним і відбивати сутність виконуваних дій. **Ім'я** методу не повинне містити пробілів у назві.
4. За ім'ям методу випливають дужки (параметр 1, ..., Параметр N), у яких описується список параметрів (аргументів).
5. Після опису заголовка методу варто нажати на клавішу [**Enter**] клавіатури, що приведе до утворення повної конструкції методу.

Примітка 1: Якщо процедура або функція не має вхідних параметрів, то за ім'ям методу випливають порожні дужки ().

Примітка 2: Якщо функція повертає певний тип даних, то за дужками указується конкретний тип даних, наприклад: **Private** Test (X **as** Double, Y **as** Double) **as** Double.

Примітка 3: В **VBA** допускається не вказувати інструкцію **Public** для загальнодоступних процедур і функцій, якщо розроблювальний метод надалі не передбачається транслювати в іншу мову програмування.

7.1 Метод «функція»

Функція – це програмна структура, котра повертає результат через власне ім'я.

Припустимо, що необхідно розробити метод, котрий повертає суму значень елементів невідомого одномірного масиву й застосовувати даний метод тільки в процедурах одного модуля. З поставленої задачі видно, що ефективним методом буде функція, яка повертає конкретний результат.

Приклад:

```
Private Function MinArray(x())
Dim i, s
s = 0
For i = LBound(x()) To UBound(x())
s = s + x(i)
Next i
MinArray = s
End Function
```

Зазначена функція використовується в процедурі CommandButton1_Click().

Приклад:

```
Private Sub CommandButton1_Click()
Dim A(4)
A(0) = 1: A(1) = 2: A(2) = 3: A(3) = 4: A(4) = 5
Range("A1") = MinArray(A)
End Sub
```

Розглянемо на схемі (рис.1), яким способом передаються параметри із процедури у функцію і який способом функція повертає результат.

Схема реалізації функції MinArray та її використання в процедурі CommandButton1_Click()

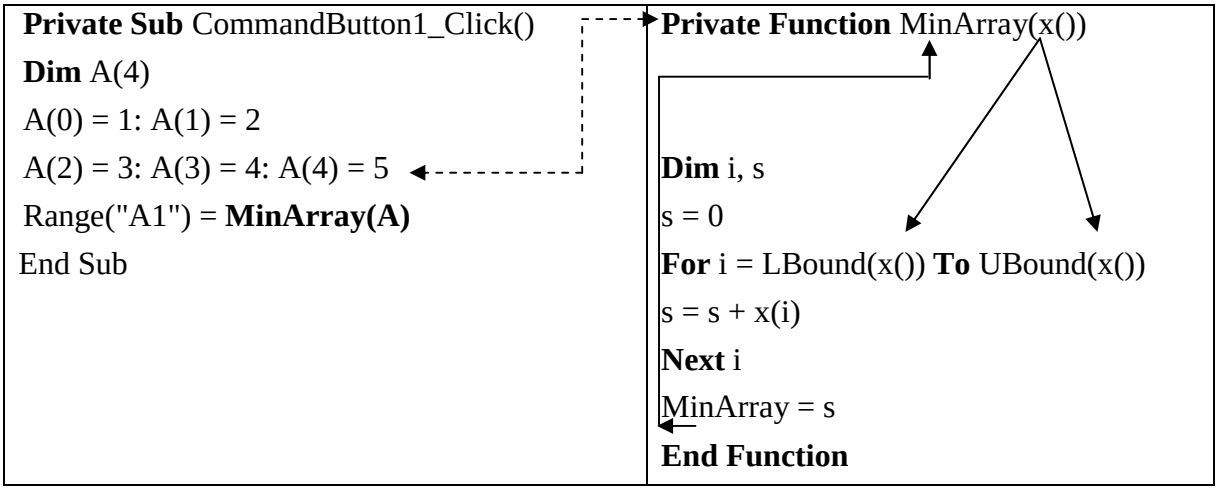

Рис.1

У процедурі **CommandButton1_Click()** ініціалізовано масив **A**, значення якого передаються як параметр **x()** у функцію **MinArray**. У функції **MinArray** за допомогою стандартних функцій **Lbound** й **Ubound** визначаються нижній і верхній індекси (границі) масиву **x()**. У діапазоні цих значень в акумуляторі **s** накопичується сума значень елементів масиву **A**, переданого як параметр **x()** у функцію **MinArray**. Після виконання всіх ітерацій змінної **MinArray**, котра є ім'ям розробленої функції, привласнюється значення акумулятора **s** (локальної змінної **s**). Таким чином, клітинка [A1] у процедурі **CommandButton1_Click()** одержує значення функції **MinArray(A):** Range("A1") = **MinArray(A)**.

Розглянемо приклад конструкції функції, котра не має параметрів.

Приклад:

```
Public Function e()
    e = Exp(1)
End Function
```

Наведена вище функція опублікована в зовнішньому модулі, вона автоматично з'явиться в списку майстра функцій Excel і може бути використана для обчислення формул, котрі написані у вигляді, близькому до математичного запису, наприклад: $y = e^x$ буде мати вигляд: [= e() ^ x] за умови, якщо деякому діапазону клітинок привласнене ім'я **x**.

7.2 Метод «процедура»

Процедура – це програмна структура, котра повертає результати обробки інформації через власні параметри. Процедура може містити один або кілька вхідних параметрів, один або кілька вихідних параметрів або не містити ніяких параметрів.

Розглянемо на прикладі конструкцію процедури, що містить тільки вхідні параметри. Припустимо, що необхідно заповнювати таблиці різних розмірів випадковими числами, значення яких лежать у деякому діапазоні [–N … N]... Тоді, процедура, що реалізує поставлену задачу, може мати такий вигляд:

Приклад:

```
Private Sub TableRND(Row, Row, Col, Col, Base)
Dim i, j
For i = Row To Row
For j = Col To Col
Cells(i, j) = Base / 2 - Rnd() * Base
Next j
Next i
End Sub
```

Дана процедура використовується в оброблювачі події щиглика по кнопці CommandButton1_Click().

Приклад:

```
Private Sub CommandButton1_Click()
TableRND 2, 4, 2, 6, 1000
End Sub
```

Розглянемо на схемі (рис.2), яким чином передаються параметри із процедури CommandButton1_Click() у процедуру **TableRND** й яким способом розроблена процедура виконує дії.

Схема реалізації процедури TableRND та її використання в процедурі CommandButton1_Click()

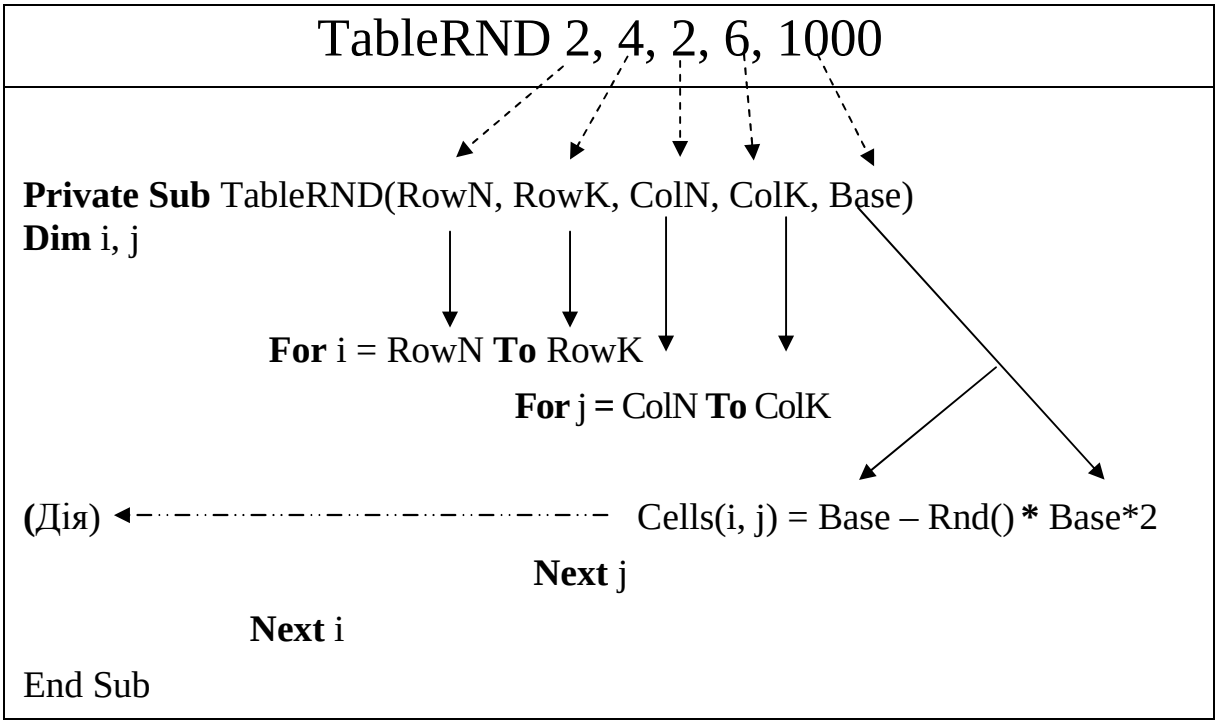

Рис.2

Програмний рядок **TableRND 2, 4, 2, 6, 1000** процедури обробки події **CommandButton1_Click()** передає в процедуру **TableRND** значення параметрів розміщення таблиці в листі Excel й опорне значення діапазону випадкових чисел:

- Початковий рядок – Row = 2.
- Кінцевий рядок – Row = 4.
- Початковий стовпець – Col = 2.
- Кінцевий стовпець – Col = 6.
- Опорне значення – Base = 1000.

Програмний рядок **Cells(i, j) = Base – Rnd() * Base*2** містить розроблені методи, котрі заносять в клітини таблиці значення випадкових чисел. В **VBA** передбачені два види запису для виклику процедури. Перший вид запису показаний у розглянутому прикладі. Другий вид запису виклику процедури містить інструкцію **Call**: **Call** TableRND(2, 4, 2, 6, 1000)

Розглянемо на прикладі конструкцію процедури, котра містить вхідні й вихідні параметри. Припустимо, що необхідно перетворити код кольорів об'єкта, представлений у вигляді довгого цілого в значення RGB каналів. Розробити процедуру рекомендується в зовнішньому модулі **ColorModel** з метою багаторазового її застосування. Така процедура містить один вхідний параметр xColor і три вихідних параметри R, G й B.

Приклад:

Модуль ColorModel

```
Option Explicit
Public Sub ColorToRGB(xColor As Long, R As Byte, G As
Byte, B As Byte)
R = xColor And &HFF&
G = (xColor And &HFF00&) \ 256
B = (xColor And &HFF0000) \ 65536
End Sub
```

Виклик процедури виконується з будь-якого модуля.

Приклад:

```
Private Sub TestColor()
Dim XC As Long
Dim m As Byte
Dim m As Byte
Dim m As Byte
XC = &H80C0FF 'значення кольорів можна задати в десятковому поданні,          'наприклад: XC = 45627
Call TestColor.ColorToRGB(XC, m, m, m)
Debug.Print XC, m, m, m
End Sub
```

Схема функціонування методу наведена на рис.3. Командний рядок процедури **TestColor**(): **Call** TestColor.ColorToRGB(XC, m, m, m) передає процедурі **ColorToRGB** єдиний параметр коду кольору через змінну **XC**. Процедура **ColorToRGB** повертає значення колірних каналів **R**, **G** й **B**, як значення змінних **m**, **m** й **m**. У процедурі **TestColor** код кольорів і значення колірних каналів виводяться у вікно налагодження програм **VBA**.

Схема реалізації процедури ColorToRGB й її використання в процедурі TestColor

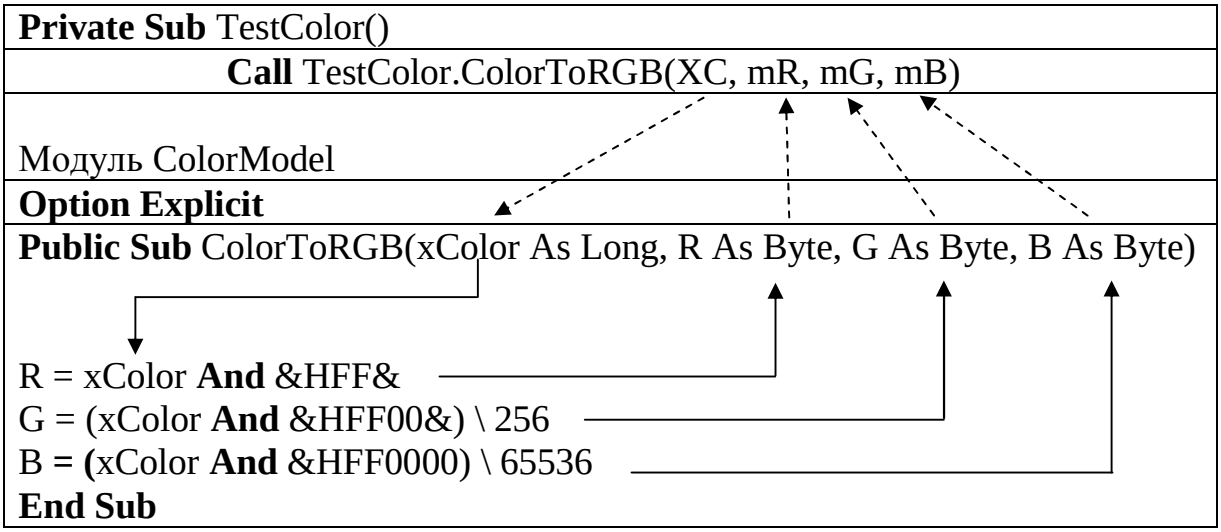

Рис.3

Типовими представниками процедур, котрі не містять ніяких параметрів, являються макроси або процедури обробки подій виду:

Private Sub CommandButton1_Click().

У процедурах такого виду обмін даними може здійснюватися через значення загальнодоступних змінних або глобальних змінних (полів). Глобальні змінні декларуються в розділі [**Загальна область** (**General**)], [**Опис** (**Declaration**)], а значення їм привласнюються в тілі методів.

Приклад:

```
Public XX
Sub Test1()
XX = 10
End Sub
Sub Test2()
Range("a1") = XX
End Sub
```

ТЕМА 8 ГРАФІЧНІ ПРИМІТИВИ, ОБ'ЄКТ SHAPES. АНІМАЦІЯ ПРОЦЕСІВ, ПРОГРАМНА КОНСТРУКЦІЯ TIMER

Графічні примітиви представлені в **Excel** палітрою малювання. Використовуючи графічні об'єкти можна якісно й інформативно представити матеріал в електронному документі. Особливий інтерес становить програмне управління графічними примітивами, котре дозволяє створювати анімаційні електронні документи.

Об'єкт Shapes

Розглянемо деякі методи й властивості об'єкта **Shapes** (фігури).

Метод **AddShape** дозволяє створити фігуру:

Приклад:

```
Shapes.AddShape(Type, Left, Top, Width, Height).Select
```

- **Type** – тип фігури.
- **Left**, **Top** – координати верхнього лівого кута фігури.
- **Width**, **Height** – координати правого нижнього кута фігури.

Тип фігури вибирається зі списку фігур, які в **VBA** представлені у великій кількості.

Приклад:

```
Shapes.AddShape(msoShapeRectangle, 50, 50, 200, 100).Select
```

Наведений програмний рядок побудує прямокутник шириною 200, висотою 100 з відступом ліворуч 50, зверху 30.

Побудованій та розміщеній в листі Excele фігурі автоматично присвоюється індекс, котрий дорівнює номеру фігури. Про кількість фігур, розміщених в листі Excel, можна довідатися викликавши метод Count.

Приклад:

```
sNum = Shapes.Count
```

Для звертання до об'єкта, при великій кількості фігур створених в листі Excel, рекомендується кожному об'єкту присвоювати конкретне ім’я.

Приклад:

```
Shapes.Item(sNum).Name = “Rectangle_1”
```

У наведеному прикладі властивість **Item** (пункт) забезпечує доступ до конкретного об'єкта через його індекс.

Для того, щоб вибрати потрібний об'єкт по імені необхідно написати наступний рядок:

Приклад:

```
Shapes.Item("Rectangle_1").Select
```

Обраний об'єкт можна залити колірним тоном.

Приклад:

```
Shapes.Item("Rectangle_1").Fill.ForeColor.RGB = RGB(255, 204, 153)
```

Для заливання можна використовувати одну з обраних у списку текстур або завантажити власну текстуру, як графічний файл.

Приклад:

```
Shapes.Item(("Rectangle_1").Fill.PresetTextured sTexture
```

VBA надає користувачеві наступні, найбільш розповсюджені методи побудови фігур:

- **AddLine** – додати лінію.
- **AddPolyline** – додати ламану лінію.
- **AddCurve** – додати криву лінію.
- **AddPicture** – додати картинку.
- AddShape **– додати фігури зі списку фігур.**
- BuildFreeform **– побудувати полігон (багатокутник).**

Найцікавішим являється метод **BuildFreeform**. Він дозволяє будувати довільні види об'єктів, на основі табличних або розрахункових даних. Розглянемо приклад зчитування даних з таблиці й побудови фігури на основі табличних даних. Зчитування даних з таблиці виконується процедурою **Sub DataTable** Побудова фігури виконується процедурою **Sub MyShape**.

Приклад:

```
Private Sub DataTable(X(), Y(), Rn, Cn)
Dim i As Integer
For i = LBound(X()) To UBound(X())
X(i) = Cells(i + Rn, Cn)
Y(i) = Cells(i + Rn, Cn + 1)
Next i
End Sub
```

```
Private Sub MyShape(myName, X(), Y(), c, c)
Dim myType As MsoRGBType
Dim myVid As MsoRGBType
Dim i As Integer
myType = msoEditingAuto
myVid = msoSegmentLine
With Shapes.BuildFreeform(myType, X(0) + c, Y(0) + c)
    For i = LBound(X()) + 1 To UBound(X())
    AddNodes myVid, myType, X(i) + c, Y(i) + c
    Next i
    ConvertToShape.Select
End With
myIndex = Shapes.Count                    'Визначення індексу
Shapes.Item(myIndex).Name = myName 'Присвоєння імені
End Sub
```

Анімація. Програмна конструкція Timer

Анімація являється дуже ефективним способом подання процесів в електронних документах. Вона припускає виконання якої-небудь дії або сукупності дій, котрі відбуваються в певний період часу. Анімація реалізується програмною конструкцією, котра використовує функцію **Timer**.

Розглянемо на прикладі структуру процедури **Zoom**, що реалізує ефект наближення об'єкта.

У даному прикладі конструкція **Timer** використовується для того, щоб задати інтервал часу відображення об'єкта **Star**. Крім того, у конструкції використається інструкція **DoEvents**, котра під час паузи організує передачу управління іншим процесам.

При виконанні зазначеної процедури об'єкт **Star** створюється на початку процедури й знищується наприкінці процедури.

Приклад:

```
Private Sub Zoom()
Dim PauseTime, Start, Finish, i
Shapes.AddShape msoShape16pointStar, 5, 5, 10, 10
                                                    ‘Створення
Shapes.Item(Shapes.Count).Name = "Star"      ‘Об'єкта Star
PauseTime = 0.1                                  'Задає тривалість.
For i = 1 To 100
Start = Timer                                    'Задає початок паузи.
Do While Timer < Start + PauseTime
    DoEvents        'Передає керування іншим процесам.
Loop
Finish = Timer                                   'Задає кінець паузи.
Shapes.Item("Star").Width = 10 + I       ‘Збільшує
Shapes.Item("Star").Height = 10 + I      ‘Розмір об'єкта Star
Next i
Shapes.Item("Star").Delete                   ‘Знищує об'єкт
End Sub
```

ЛАБОРАТОРНІ РОБОТИ ДО НАВЕДЕНИХ ТЕМ

Запуск **VBA** із середовища **Excel**, як й інших **Office** додатків, здійснюється за допомогою виконання команди меню **Сервіс \ Макрос \ Редактор Visual Basic** [**Alt + F11**].

ТЕМА 2

ВЛАСТИВОСТІ ОБ'ЄКТІВ І МЕТОДИ УПРАВЛІННЯ НИМИ

ЛАБОРАТОРНА РОБОТА №1

СИМВОЛЬНА АДРЕСАЦІЯ КЛІТИНОК, ЗМІННА ТИПУ VARIANT

Введення й вивід даних для клітинок листа Excel здійснюється за допомогою властивостей **Cells** (для однієї клітинки) і **Range** (для діапазону клітинок). Властивості **Cells** й **Range** являються об'єктами та містять вкла-

дені властивості, такі як **Volue** (значення), **Interior** (інтер'єр, оформлення), **formula** (формула), **Select** (вибір) та інші.

Зверніть увагу на той факт, що символьна адреса клітинок є текстом. Він береться в лапки.

Вправа 1

Присвоєння клітинці деякого значення

```
Private Sub CommandButton1_Click()
Range("A5") = 5: Range("B5") = 5 + 5
Range("C5") = 5 * 5: Range("D5") = 5/5
Range("E5") = Sin(0.5): Range("F5") = "ТЕКСТ"
End Sub
```

Вправа 2

Присвоєння клітинці значення іншої клітинки

```
Private Sub CommandButton2_Click()
Range("A10") = Range("A5")
Range("B10") = Range("A5") + Range("B5")
Range("C10") = Range("C5") * Range("B5")
Range("D10") = Range("D5") / Range("C5")
Range("E10") = Range("E5")  2 + Cos(0.5)  2
Range("F10") = Range("F5") & Range("A5")
End Sub
```

Вправа 3

Заливання клітинки указаними кольорами

```
Private Sub CommandButton3_Click()
Range("A16").Select: Range("A16").Interior.Color = vbRed
Range("B16").Select: Range("B16").Interior.Color = vbGreen
Range("C16").Select: Range("C16").Interior.Color = vbBlue
Range("D16").Select: Range("D16").Interior.Color = vbMagenta
Range("E16").Select: Range("E16").Interior.Color = vbYellow
Range("F16").Select: Range("F16").Interior.Color = vbCyan
End Sub
```

Література

1. Баженов В. А., Венгерский П. С., Горлач В. М., Левченко О. М., та ін. Інформатика. Комп'ютерна техніка. Комп'ютерні технології: Підручник. – К.: Каравела, 2003. – 464с.
2. Береза А. М. Основи створення інформаційних систем: Навч. посібник. – К.: КНЕУ, 1998. – 140с.
3. Винтер Р., Винтер П. Microsoft Office 97. в 2-х т. Т.1 – СПб.: – BHV, 1997. – 453с.
4. Глушаков С. В., Сурядный А. С. Microsoft Office 2000: Учебный курс. – Харьков: Фолио, 2002. – 500с.
5. Джонс Э., Саттон Д. Библия пользователя Office 97. – К.: Диалектика, 1997. – 642с.
6. Кассер Б. Использование PowerPoint 97. – К.: Диалектика, 1998. – 231с.
7. Краткий курс Microsoft Office 2000. – Ростов на Дону: Феникс, 2000. – 458с.
8. Персон Р. Excel 97 в 2-х т. Т.1 – СПб.: BHV, 1997. – 421с.
9. Рычков В. Microsoft Excel 2000: краткий курс – СПб.: Питер, 2001. – 318с.
10. Рычков В. Самоучитель Excel 2000. – СПб.: Питер, 2002. – 336с.

ЗМІСТ

Навчальне видання

Швачич Геннадій Григорович
Овсянніков Олександр Васильович
Кузьменко Вячеслав Віталійович
Нєчаєва Наталія Іванівна
Петричук Ліна Миколаївна

Інформатика та комп'ютерна техніка. Елементи об'єктно-орієнтованого програмування

Розділ «Реалізація концепції о'бєктно-орієнтованого програмування в мові Visual Basic for Application»

Навчальний посібник

Тем. план 2007, поз. 155

Підписано до друку 06.02.07. Формат 60x84 1/16. Папір друк. Друк плоский. Облік.-вид. арк. 3,05. Умов.друк. арк. 3,02 Тираж 100 пр. Замовлення №11

Національна металургійна академія України
49600, Дніпропетровськ – 5, пр. Гагаріна, 4

Редакційно – видавничий відділ НМетАУ