УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

Факультет інформаційно-керуючих систем та технологій

Кафедра обчислювальної техніки та систем управління

ОСНОВИ ПРОГРАМУВАННЯ ІНЖЕНЕРНО-ТЕХНІЧНИХ РОЗРАХУНКІВ АЛГОРИТМІЧНОЮ МОВОЮ ВИСОКОГО РІВНЯ: ВВЕДЕННЯ ДО VBSCRIPT

Конспект лекцій

з дисциплін

«ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА ПРОГРАМУВАННЯ», «ІНФОРМАТИКА»

Харків – 2020

Основи програмування інженерно-технічних розрахунків алгоритмічною мовою високого рівня: введення до VBScript: Конспект лекцій / С. Є. Бантюков, В. Г. Пчолін, І. Г. Бізюк, Р. О. Яровий, О. В. Казанко. – Харків : УкрДУЗТ, 2020. – 87 с.

Конспект лекцій розроблено відповідно до робочих програм дисциплін «Обчислювальна техніка та програмування» та «Інформатика».

Метою лекцій є опанування студентами навичок вирішення інженерно-технічних завдань алгоритмічною мовою високого рівня: Visual Basic Script.

Рекомендується для студентів спеціальностей «Галузеве машинобудування», «Будівництво та цивільна інженерія», «Залізничний транспорт» усіх форм навчання.

Іл. 10, табл. 9, бібліогр.: 14 назв.

Конспект лекцій розглянуто і рекомендовано до друку на засіданні кафедри обчислювальної техніки та систем управління 07 березня 2019 р., протокол № 9.

Рецензент

проф. В. І. Мойсеєнко

ЗМІСТ

ВСТУП

Одне з головних завдань курсу навчальної дисципліни «Обчислювальна техніка та програмування» — навчити студентів, майбутніх фахівців інженерно-технічного профілю, змушувати комп'ютер виконувати потрібні для них обчислення інженерного розрахунку. Цей конспект являє собою виклад заключної частини курсу. З першої частини курсу відомо, які етапи має пройти процес розв'язання обчислювальної задачі на ЕОМ, щоб комп'ютер вивів на друк або дисплей остаточні результати розрахунку для їх подальшого аналізу людиною. Формалізація задачі, підбір математичних методів для її розв'язання, розроблення алгоритму з графічним записом останнього у вигляді схеми та складання програмного коду — основа зазначеного процесу.

Як складається алгоритм обчислювальної задачі і будується відповідна йому схема, було детально розглянуто в першій частині курсу. У цьому конспекті розглядається етап програмування. Під програмуванням процесу розв'язання обчислювальної інженерної задачі розуміється зазначення послідовності деяких інструкцій, при яких, якщо їх належним чином ввести до комп'ютера, останній сприйме їх як систему пов'язаних між собою наказів до виконання належних обчислень, що відбуваються точно у відповідності зі схемою алгоритму. Під інструкціями тут розуміються елементи деякої алгоритмічної мови.

З першої частини курсу відомо, що алгоритмічних мов є досить багато. Їх можна з відомою часткою умовності поділити на дві групи: алгоритмічні мови низького і високого рівня. Алгоритмічні мови низького рівня — це машинно-орієнтовані мови. Вони полегшують людині складання програмного коду для конкретної системи команд, апаратно реалізованої в процесорі комп'ютера. Алгоритмічні мови високого рівня — це мови, близькі до природних мов людини. Питання про те, з якої групи вибрати мову для ілюстрування процесу переходу від схеми алгоритму до тексту програми, не ставиться, оскільки очевидно, що розглядати мови низького рівня, які жорстко пов'язані з особливостями обчислювача, тут не розумно, бо специфіка програмування конкретного пристрою може «заслонити» загальні принципи процесу, як кажуть, «за деревами лісу не побачиш». Але

є питання, яку з мов високого рівня доцільно вибрати для початкового вивчення, щоб на її основі наочно показати відповідність між елементами схем алгоритмів та інструкціями алгоритмічної мови.

На це питання навряд чи можна відповісти однозначно, оскільки в більшості алгоритмічні проблемно-орієнтовані мови схожі одна з одною у визначеності своїх конструкцій, структурі та іншому. Однак серед них можна назвати таку, яка у своїх засадах була орієнтована її авторами як інструментарій для програмістів-початківців. Це мова програмування Basic. Вона була створена в 1964 році професорами Дартмутського коледжу Джоном Кемені і Томасом Курцем. Назву мові було дано у відповідності з класичними традиціями інформатики. Це абревіатура від словосполучення «Beginner's All-purpose Symbolic Instruction Code» (багатоцільовий код символьних інструкцій для початківців). Мова призначалася як інструмент студентам-непрограмістам, щоб ті могли самостійно створювати комп'ютерні програми для розв'язання своїх задач.

На підтримку вибору мови для розгляду в рамках цього курсу слід, напевно, згадати і факт, що саме простий і зрозумілий інтерпретатор мови Basic став першим програмним продуктом, з якого в 1975 році компанія Microsoft (тоді це були лише двоє — Білл Гейтс і Пол Аллен) почала свій стрімкий розвиток.

Алгоритмічна мова Basic тепер є однією з найбільш використовуваних мов на платформі Windows. Це вже не одна мова, а ціле сімейство мов, що відрізняються одна від одної парадигмами програмування і синтаксисом. Ряд сучасних компіляторів на чолі з Visual Basic і VB.NET реалізують об'єктно-орієнтоване і подієве програмування. Але для цілей цього курсу все ж таки найбільш зручним буде розгляд ще однієї сучасної гілки Basic — Visual Basic Script.

Слід назвати основні причини такого вибору:

1 Мова VBScript, незважаючи на свої широкі можливості і специфіку (вона є об'єктно-орієнтованою, подієво-орієнтованою мовою, використовується для автоматизації адміністрування систем Windows, у серверних і клієнтських web-додатках), зберегла успадковану від своїх «предків» (перших діалектів Basic) простоту і зрозумілість.

2 VBScript має великий потенціал для розширення свого використання і своєї доступності, оскільки на кожному комп'ютері, яким управляє ОС Windows, є мова VBScript, і нею можна користуватися для здійснення будь-яких розрахунків.

3 Впроваджуючись у web-технології, обслуговуючи апарат створення HTML-документа, VBScript дозволяє користуватися і управляти елементами його потужного інтерфейсу, що наближує VBScript до можливостей Visual Basic 6 не тільки у виробництві обчислень, але і в організації графічних форм для діалогу з користувачем.

4 Оскільки корпорація Microsoft має намір використовувати VBScript як мову управління для web-розробок і надалі, вона має і буде корпорацією розвиватися і вдосконалюватися. Однак, за «неофіційним» правилом розробників web-мережі, нові рішення в управлінні HTML-документами не повинні скасовувати старі. І поки воно ними не порушувалось. Це дає визначену впевненість у тому, що той, хто навчився користуватися VBScript за ранньою версією, зможе користуватися тими самими знаннями і в пізніших її версіях.

1 ТИПИ ДАНИХ У МОВІ VBSCRIPT

З першої частини курсу ви знаєте, що об’єктами дії в алгоритмах є величини змінні, постійні (константи) і обчислювальні вирази (арифметичні, логічні та символьні). За своєю сутністю ці об’єкти зберігають і переносять дані від попереднього до поточного кроку алгоритму, від поточного кроку до наступного. Однак дані, оброблювані в алгоритмах, бувають різними за типом. Вони можуть бути числовими, символьними або логічними. Укладач алгоритму стежить, щоб операції в алгоритмі завжди узгоджувалися за типом даних з операндами. При роботі програм, складених мовою програмування, має відбуватися те саме, але додається ще один нюанс: дані різного типу і зберігаються в пам’яті комп’ютера по-різному.

Основними об’єктами дії в обчислювальному алгоритмі є змінні величини, кожна з яких постачена ім’ям (ідентифікатором), а ім’я змінної у програмній реалізації — це покажчик на адресу пам’яті комп’ютера, де містяться безпосередньо дані, пов’язані з цією змінною. Тому для спрощення роботи транслятора вигідно заздалегідь вказувати в програмі тип даних, які будуть відповідати тій чи іншій змінній. Однак, з іншого боку, це ускладнює роботу програміста. У зв’язку з цим існує така класифікація алгоритмічних мов високого рівня, яка заснована на способі «прив’язки» змінних до типу даних (типізації). Розрізняють мови статичної та динамічної типізацій.

Статично типізовані мови перевіряють типи і шукають помилки типізації перед виконанням обчислень. Динамічно типізовані мови перевіряють типи і шукають помилки типізації безпосередньо при виконанні обчислень.

Мова VBScript належить до динамічно типізованих мов. Вважається, що всі змінні, використовувані в програмі, складеною мовою VBScript, мають один загальний тип даних — Variant. Це спеціальний тип, який може містити в собі різні види даних. Всі обчислення і функції мови також повертають дані типу Variant. Різні види даних, які можуть міститися в типі Variant, називаються підтипами. У таблиці 1.1 наведено підтипи даних з їх описами, визначені типом Variant.

Таблиця 1.1 – Підтипи даних у мові VBS

Підтип	Опис
Byte	Цілі числа в діапазоні від 0 до 255
Boolean	Значення логічних величин, тобто значення True (істина) або False (хибність)
Integer	Цілі числа в діапазоні від -32768 до 32767
Long	Довгі цілі числа в діапазоні від -2147483648 до 2147483647
Single	Числа звичайної точності з рухомою крапкою в діапазоні від -3.402823E38 до -1.401298E-45 для від'ємних значень і від 1.401298E-45 до 3.402823E38 для додатних значень
Double	Числа подвійної точності з рухомою крапкою в діапазоні від -1.79769313486232E308 до -4.94065645841247E-324 для від'ємних значень і від 4.94065645841247E-324 до 1.79769313486232E308 для додатних значень
Currency	Грошові значення в діапазоні від 922337203685477.5808 до 922337203685477.5807
Data / (Time)	Числові значення, які задають дату в діапазоні від 1-го січня 100 року до 31 грудня 9999 року та час у діапазоні від 0:0:0 до 23:59:59
String	Рядки символів змінної довжини з максимальною довжиною до 2 мільйонів символів
Object	Дані про об'єкт
Empty	Ініціалізація типу Variant без визначення підтипу. Містить 0 для числових значень і порожній рядок ("") для символьних значень
Null	Неінтерпретовані дані
Error	Номер помилки

Який підтип даних має та чи інша величина в поточний момент виконання програми у VBScript, визначає спеціальний програмний апарат контролю даних, який спрацьовує при обчисленні будь-якої арифметичної, логічної або символьної операції. За знаком самої операції визначається, якого підтипу мають бути дані операндів. Далі перевіряється відповідність операндів операції. Якщо реальні дані в операндах не відповідають

операції, названий апарат намагається перетворити їх у потрібний підтип. Якщо таке перетворення для одного з операндів зробити неможливо, то підтип його даних оголошується Null, а підтип результату операції — Error.

Іноді за знаком операції неможливо точно визначити операцію. Наприклад, у VBScript знак плюс «+» може позначати і операцію арифметичного додавання, і операцію символьної конкатенації. У таких випадках апарат контролю даних використовує додаткові алгоритми розпізнавання операції за результатом порівняння підтипів операндів один з одним. Однак імовірність помилки в таких ситуаціях зростає, і результат часом може бути хибним. Програмісту слід всіляко уникати виникнення неоднозначностей при автоматичному перетворенні типів. Так, у наведеному прикладі надійніше для операції конкатенації застосовувати знак амперсанд «&».

Зауважимо, що апарат автоматичного контролю даних, крім зазначених випадків, проводить аналіз і перетворення даних для аргументів стандартних функцій і при візуалізації вихідних даних.

2 СКЛАДАННЯ ПРОГРАМ МОВОЮ VBSCRIPT

Програми, що складаються мовою VBScript (зазвичай їх називають сценаріями), являють собою послідовності інструкцій (англ. Statements). Кожна інструкція є записом спеціального виду, який або визначає деякі дії інтерпретатора над конкретними змінними, константами та виразами, що задані в сценарії, або описує (оголошує) змінні і константи необхідним для інтерпретатора способом.

Записуються інструкції рядками символів, для яких в електронній формі при формуванні файлів використовуються кодування ASCII або Юнікод. Інструкція зазвичай закінчується кінцем рядка (символ «повернення каретки» плюс символ «кінець рядка»). Однак у мові VBScript є інструкції, що відповідно до свого формату при запису займають кілька рядків.

У скриптах VBS часто зустрічаються і короткі інструкції. Тоді кілька інструкцій мови VBS можна об’єднати в один рядок з використанням розподільника інструкцій — символу двокрапки

«:». Іноді інструкція настільки довга, що записати її одним рядком, як вимагає її формат, неможливо або небажано. У цьому випадку інструкції можна записати кількома рядками, розриваючи в місцях пробілів і використовуючи в якості знака «продовження на наступному рядку» символ підкреслення «_».

Запис кожної інструкції мовою VBScript складається з двох типів компонент: ключових слів (ключів) і параметрів.

Ключові слова — це набори символів, що допомагають інтерпретатору розпізнавати інструкцію в тексті програми і визначати місцезнаходження записів її параметрів. Зазвичай ключі збігаються з певними словами з англійської мови, щоб людині, яка візуально переглядає текст програми, легше було розуміти її суть. Серед набору ключів для однієї інструкції завжди виділяється один ключ, який називають початковим, на відміну від інших, що називають вторинними. За початковим ключем інтерпретатор визначає, яку інструкцію треба виконати, за вторинними ключами — де шукати записи параметрів для виконання цієї інструкції. Ключі в записі інструкції обов'язково відокремлюються один від одного та від інших компонентів інструкції одним або декількома пробілами (пробіл розглядається інтерпретатором як нейтральний розподільник між будь-якими елементами тексту програми). Ключові слова інструкцій не можна застосовувати як імена користувацьких структур, наприклад для імен змінних або констант. Слід зауважити, що ключові слова можна заносити до початкового файлу як малими, так і великими латинськими літерами — інтерпретатор VBS при будь-яких варіантах набору сприймає їх правильно.

Параметрами інструкції можуть бути константи, імена змінних, вирази і навіть послідовності з інших інструкцій. Що і де знаходиться в інструкції, визначається її форматом, тобто правилом, за яким треба складати запис інструкції.

У рамках цього курсу ми розглянемо формати багатьох інструкцій мови VBScript. Описи форматів будуть наводитись у вигляді текстових шаблонів, у яких вказуються всі ключі і місця для необхідних параметрів. Крім того, щоб застосування шаблонів було максимально зрозумілим, у них будуть використовуватися такі загальні позначення: у квадратні дужки «[...]» укладаються ті частини шаблона, що є необов'язковими, і їх не завжди треба

застосовувати, у кутових дужках «<...>» дається загальний опис параметра, що в програмі має замінюватися конкретним символьним набором, наприклад числовим значенням, ім'ям тощо. До всіх елементів шаблона будуть обов'язково даватися коментарі.

Для формування файлу з вихідним кодом скрипту можна використовувати будь-який текстовий редактор, наприклад стандартну програму Windows Блокнот. Існують звичайно і більш потужні засоби для запису файлів із кодом VBScript, що дозволяють не тільки створити сам файл, а й провести необхідне налагодження коду безпосередньо у своєму середовищі. Наприклад, такий програмний засіб є в інтегрованому середовищі розробки Microsoft Visual Basic 6. Однак для його використання потрібно купити ліцензію.

Слід зауважити, що в літературі з програмування російською та українською мовами інструкції мови програмування зазвичай називають операторами, хоча це не зовсім точно: операторами в англомовних першоджерелах називають символи для позначення математичних, логічних і рядкових операцій (=, +, -, /, and, or, & та ін.). Далі буде використовуватися традиційна російсько-українська термінологія з використанням слова «оператор» для позначення інструкції мови VBS.

3 ЗАСТОСУВАННЯ ЗМІННИХ І КОНСТАНТ У МОВІ VBSCRIPT

Змінні і константи — це основні «дійові особи» будь-якого обчислювального алгоритму. Оскільки програма деякою алгоритмічною мовою є записом того самого алгоритму, то в алгоритмічній мові обов'язково мають бути засоби для визначення та використання змінних і констант.

Нагадаємо, що змінною величиною в алгоритмах називають деякий абстрактний об'єкт, який зберігає в собі якийсь елемент даних, що використовується при виконанні дій алгоритму. Змінна завжди має два атрибути (властивості): ім'я (ідентифікатор) і значення. Ім'я змінної використовується як замінник її значення в різних описах дій, які треба робити в алгоритмі з даними, а

значення змінної – це власне сам елемент даних, який зберігається у змінній на конкретному етапі виконання алгоритму. Значення змінної в процесі виконання алгоритму може змінюватися.

Постійна величина (константа) в алгоритмах є теж деяким абстрактним об'єктом, що зберігає в собі елемент даних подібно до змінної. Однак, на відміну від останньої, значення константи в процесі виконання алгоритму ніколи не змінюється. Тому константі, як правило, не потрібно задавати ім'я, а в описах дій алгоритму вона позначається відображенням свого значення.

Змінна в програмі (скрипті) задається, як вже зазначалось раніше, своїм ім'ям, яке по суті є покажчиком на адресу ділянки оперативної пам'яті комп'ютера, де записані дані, що є її значенням. А константа в тексті програми задається відповідним записом свого значення, що при виконанні дії програми транслюється в необхідній формі в ділянку оперативної пам'яті, яку відводять під операнди цієї дії. Якщо константа в тексті програми зустрічається багаторазово, то кожен раз під її значення відводиться нове місце в пам'яті комп'ютера. Тому, щоб скоротити обсяги оперативної пам'яті, що витрачаються на реалізацію програмного коду, і час програміста зі складання й налагодження коду, іноді має сенс дати константі ім'я. Тоді її значення буде записано в оперативній пам'яті тільки один раз. Мова VBS має усі названі можливості.

Формально ідентифікатор — це деяка послідовність символів. Ідентифікатори в VBScript, незалежно від їх використання, будуються за однаковими правилами:

1) ідентифікатор може складатися з латинських малих і великих літер, цифр і символу підкреслення;

2) ідентифікатор не повинен починатися з цифри;

3) довжина ідентифікатора — не більше 255 символів;

4) букви у верхньому і нижньому регістрах інтерпретатором VBScript не розрізняються;

5) ідентифікатор має бути унікальним в області визначення.

Ідентифікатори можна давати змінним, константам, функціям, підпрограмам, масивам даних, а також деяким іншим конструкціям мови VBScript, що не розглядаються в рамках курсу.

Зазвичай у сценарії на VBScript всі змінні, які в ньому беруть участь, оголошуються. Для цього можна використовувати

оператор Dim (назви операторам даються, як правило, за їхніми початковими ключами). Оператор Dim відносять до групи операторів-декларацій (оголошень). Подібні оператори є в багатьох алгоритмічних мовах. Оператори цієї групи служать для повідомлення трансляторам загальних характеристик і відомостей про особливості інформаційних структур, які використовуються в програмі. Оператор Dim — це не єдиний оператор-декларація в мові VBScript, що дозволяє оголошувати змінні, але він дуже зручний і простий у використанні, тому інші оператори VBS, що служать з цією ж метою, ми розглядати не будемо. Початковий ключ Dim виник від скорочення англійського слова dimension (розмірність), оскільки в ранніх діалектах Basic такий оператор оголошував тільки масиви даних.

Формат оператора можна задати так:

Dim <ім’я змінної> [**,** <ім’я змінної> [**,** …]] ,

де <ім’я змінної> — ім’я змінної, складене за правилами складання імен у VBScript.

Довжина списку імен після ключа Dim формально не обмежується, задається відповідно до необхідності й зручності візуалізації. У сценарії послідовно може бути застосовано кілька операторів Dim.

Приклади рядків оператора Dim:

```
Dim Name, Address, City, State
Dim myName
Dim YouName, You_Nam
```

Опис кожної змінної за допомогою оператора Dim має передувати в програмі першому оператору, що використовує її для обчислення.

Якщо сценарій заданий єдиним модулем (єдиною процедурою), тобто являє собою послідовність операторів, яка має бути одноразово виконана інтерпретатором, і далі ніяких

додаткових обчислень не передбачено, то для оголошення змінних досить керуватися тільки вищезгаданим правилом. Однак такі сценарії в практиці зустрічаються рідко.

Зазвичай сценарій, навіть той, що реалізує інженерне обчислення, має більш складну структуру. Він — багатомодульний, має головну процедуру і підлеглі. Наявність підлеглих процедур добре тим, що їх можна неодноразово виконувати, наприклад за викликом із головної процедури до того моменту, коли функція скрипту буде повністю виконана. Така структура, щонайменше, може сильно скорочувати обсяги вихідного коду сценарію. А в умовах, за яких ми будемо застосовувати інтерпретатор VBS, вона просто необхідна. Використання багатомодульного скрипту призводить до виникнення в будь-якої змінної, що в ньому використовуються, додаткових якостей: області видимості і часу життя.

Область видимості змінної — це та частина сценарію, у якій оголошену зміну можна використовувати, тобто здійснювати розрахунки з її значенням і змінювати це значення. Наприклад, змінну, оголошену в головній процедурі, можна використовувати в будь-якому місці сценарію. Така змінна є глобальною і називається змінною рівня сценарію. Якщо ж змінна оголошена усередині допоміжної процедури, то така змінна може бути використовувана тільки операторами цієї процедури. Ця змінна є локальною і називається змінної рівня процедури.

Час життя змінної — це час, протягом якого існує змінна. Час життя змінної рівня сценарію починається з моменту її оголошення і триває до закінчення виконання сценарію. Час життя змінної рівня процедури починається з моменту виклику процедури і триває до закінчення виконання процедури. При виході з процедури локальна змінна знищується. Використання локальних змінних дозволяє економити пам’ять комп’ютера. Локальні змінні можуть мати такі самі імена, як і глобальні. У цьому випадку в процедурі використовується локальна змінна, а в інших місцях сценарію використовується глобальна змінна, «тезка» локальної.

Таким чином, чи є змінна глобальною або локальною, залежить від того, у коді якої процедури сценарію вона оголошена. Оголошення змінної за допомогою оператора Dim називають

явним, але можна оголосити змінну і неявно. Для цього її просто треба почати використовувати. За відсутності явних оголошень всі змінні, що використовуються в головній процедурі, є глобальними, а відсутні в головній процедурі, але які використовуються в допоміжних процедурах, — локальними.

Однак явне оголошення змінних сприяє зменшенню кількості помилок при програмуванні, зокрема запобігає конфліктам між змінними, що використовуються в основній процедурі, і допоміжними при однакових іменах. Тому у VBS передбачено використання оператора-декларації, що забороняє використовувати в сценарії змінні, оголошені неявним способом. Формат цього оператора задається ключовим рядком

Option Explicit

(явний режим), який мусить бути першим рядком програми.

Використання оператора Option Explicit дозволяє уникнути найпоширеніших помилок при записі імен змінних. Випадково можна оголосити неявно змінну, наприклад через помилку набору. Припустимо, є явно оголошена змінна Right, а десь у тексті сценарію випадково було набрано замість імені Right символьний набір Riht. Якщо не використовувався оператор Option Explicit, то це не буде вважатися помилкою, оскільки Riht буде розцінюватися як неявно оголошена змінна. Однак результат виконання такого сценарію буде непередбачуваним, тому що змінна Riht не планувалася для використання в сценарії.

Константи в операторах VBScript, як уже зазначалося, зазвичай задаються своїми значеннями. Але тим константам, які часто використовуються в скрипті, можна задати імена (ідентифікатори). Це робить тексти програм такими, що візуально легко читаються, і економить пам’ять комп’ютера при обчисленнях. Формат оператора, який це робить, такий:

Const <ім’я константи> = <значення> ,

де <ім’я константи> — ім’я, що призначається константі, складене за правилами складання імен у VBScript;

<значення> — значення константи, що задане за правилами задавання значень констант у VBScript.

Оператори Const можна вводити в будь-якій частині програми.

Розглянемо правила задавання значень констант. Цілі числові константи записуються так, як прийнято при рукописному записі. Числові константи, де є ціла і дробова частини, записуються з застосуванням розподільного знака — крапки «.», можна використовувати для них і експоненціальний спосіб запису, наприклад 1.15e-15 = 1,15 × 10^{-15}. Значення рядкових констант задаються набором символів, обмеженим з двох боків прямими подвійними лапками "...". Значення дати й часу обмежуються по обидва боки знаками решітки «#». При цьому розподільником складових дати є або дефіс «-», або прямий слеш «/», а розподільником складових часу — двокрапка «:», дата від часу відділяється пробілом. Однак формат відображення (а не зберігання) дати визначається настройками інтерпретатора.

Наведемо кілька прикладів використання оператора іменування констант.

```
Const N = 1.15e-15
Const FIO = "Иванов Иван Иванович"
Const Data_rec = #05-21-1996 06:30:00#
Const Time_rec = #06:30:00#
Const CITY = "Харьков"
```

4 ЗАПИС ОБЧИСЛЕНЬ У VBS ТА ОПЕРАТОР ПРИСВОЄННЯ

У мові VBScript можна задавати обчислення з арифметичними, логічними, символьними операціями і з операціями відношення (порівняння). Перелічені операції переважно є двомісними, тому запис кожної в складі рядків операторів VBS може бути заданий згідно з форматом

<лівий операнд> <ключ операції> <правий операнд> ,

де <ключ операції> — набір символів, прийнятий у мові VBScript для позначення конкретної операції;

<лівий операнд>, <правий операнд> — задавання операндів (вхідних даних) в операції. Вони можуть бути представлені значеннями констант, іменами змінних або констант, а також виразами.

Якщо ключ операції задається набором латинських літер, то його написання повністю підпорядковується правилам запису ключових слів в операторах VBS. Якщо ключем операції є один або два спеціальних символи, то пробіли з обох боків такого ключа, що мають виділяти ключ згідно з загальними правилами, можна опускати. Звичайно, таке «спрощення» призведе до ускладнення візуального сприйняття коду для людини.

Нижче в таблицях 4.1-4.3 наведені описи і символи для арифметичних операцій, операцій відношення (порівняння) і логічних операцій відповідно, використовуваних у VBScript.

У наведених таблицях є дві одномісні операції: унарний мінус (у таблиці 4.1) і заперечення (у таблиці 4.3). Їхній формат запису майже такий самий, як у двомісних операцій: відсутній тільки лівий операнд. Всі інші правила запису зберігаються.

Таблиця 4.1 — Арифметичні операції у VBS

Опис	Символ операції
Піднесення до степеня	^
Унарний мінус	-
Множення	*
Ділення	/
Цілочисельне ділення	\
Модуль (ділення за модулем)	Mod
Додавання	+
Віднімання	-

У VBScript є ще символьна операція — конкатенація. Вона одна у своїй групі, тому для символьних операцій таблиця не наведена. Це двомісна операція. Конкатенація з’єднує дві вхідні послідовності символів (лівий і правий операнди) в одну вихідну

символьну послідовність. У вихідний послідовності, якщо її читати зліва направо, спочатку йдуть символи лівого операнда від першого лівого до останнього правого, потім слідують символи правого операнда від першого лівого до останнього правого.

Таблиця 4.2 — Операції відношення (порівняння) у VBS

Опис	Символ операції
Дорівнює	=
Не дорівнює	<>
Менше ніж	<
Більше ніж	>
Менше або дорівнює	<=
Більше або дорівнює	>=

Таблиця 4.3 — Логічні операції у VBS

Опис	Символ операції
Заперечення (логічне НЕ, інверсія)	Not
Кон'юнкція (логічне множення, логічне І)	And
Диз'юнкція (логічне додавання, логічне АБО)	Or
Виключне АБО	Xor
Еквівалентність	Eqv
Імплікація	Imp

Ключем операції конкатенація у VBS служать знак амперсанд «&» або знак плюс «+», останній, як зазначалося раніше, з цією метою застосовувати не рекомендується.

Серед логічних операцій, що безпосередньо реалізуються у VBScript, крім трьох базових логічних операцій, є ще три: виключне «або», еквівалентність, імплікація. З початкової частини цього курсу і курсу «Вища математика» вам відомо, що, використовуючи базові логічні операції, можна задати логічні вирази, які реалізують будь-яке інше логічне двомісне обчислення. Нижче наведені рівносильні логічні вирази на основі базових операцій, що реалізують виключне «або», еквівалентність та імплікацію (*a* і *b* — початкові логічні величини):

$$a \oplus b(\text{виключне АБО}) = (a \wedge \neg b) \vee (\neg a \wedge b),$$
$$a \sim b(\text{еквівалентність}) = (\neg a \vee b) \wedge (\neg b \vee a),$$
$$a \to b(\text{імплікація}) = \neg a \vee b .$$

Наведені вище відомості про записи арифметичних операцій в операторах VBScript достатні, щоб їх почати використовувати на практиці, оскільки вони схожі на записи відповідних операцій у формулах, якщо останні писати вручну на папері. Уточнимо правила використання тільки для двох з них: піднесення до степеня і ділення за модулем.

При записі операції піднесення до степеня в рядку оператора VBS лівий операнд має задавати значення основи операції піднесення до степеня, а правий операнд — значення порядку степеня.

Ділення за модулем — арифметична операція, результатом якої є залишок від ділення цілого числа на інше ціле число. В операторах VBScript під час запису цієї операції лівий операнд має задавати число, яке ділиться (ділене), а правий операнд — число, на яке ділять (дільник). Крім того, операція Mod у VBS може виконуватися з дійсними числами, що при цьому попередньо округляються до цілих.

При проведенні складних інженерних розрахунків за допомогою VBScript їх виконавцю обов’язково знадобиться звертатися до обчислення вбудованих функцій, які є в розпорядженні інтерпретатора VBS.

Вбудована (або стандартна) функція — це готова процедура, код якої зберігається в спеціальній, закритій від втручання, програмній базі (бібліотеці) і може бути виконаний інтерпретатором, якщо в тексті програми користувача є до неї звернення (виклик), за допомогою якого організовується в ній передача вхідних параметрів і приймання від неї обчислених результатів.

Вбудована функція може бути викликана до виконання двома способами. Вони вибираються залежно від розташування виклику функції в тексті програми користувача. Якщо запис виклику функції є частиною символьного рядка оператора, то формат виклику вбудованої функції такий:

<ім’я функції>**(** [<вхідні параметри>] **)** ,

де <ім’я функції> — конкретне ім’я (ідентифікатор) функції, за яким інтерпретатор VBS визначає місце зберігання її коду і виконує його;

<вхідні параметри> — список вхідних параметрів, які мають бути використані інтерпретатором для отримання вихідного значення функції.

Кожен параметр у виклику функції може бути представлений конкретним значенням, ім’ям змінної або константи, виразом. Параметри в списку відокремлюються один від одного комами. Порядок проходження параметрів у списку введення свій для кожної конкретної функції, порушувати його не можна, оскільки відповідно до нього інтерпретатор визначає роль кожного вхідного параметра у виконанні функції. Тому якщо який-небудь параметр у вхідному списку опускається, то коми в будь-якому випадку, мають залишатися. Однак кома не повинна завершувати список. Якщо у списку опускається останній (правий) параметр, то в цьому випадку кому в кінці списку не ставлять. Сам запис виклику функції при такому способі виклику може вважатися ім’ям значення результату виконання функції, тобто в цій якості його можна включати до складу запису інших обчислень.

Якщо запис виклику функції розглядається як самостійний оператор у програмі, то формат такого виклику інший:

<ім’я функції> [<вхідні параметри>] ,

де <ім’я функції> та <вхідні параметри> — позначають те саме, що і в попередньому описі формату.

Легко побачити, що цей опис формату відрізняється від попереднього тільки відсутністю круглих дужок, які обмежують запис списку вхідних параметрів. Всі інші правила до запису списку параметрів зберігаються. І звичайно цей виклик функції завжди задається окремим рядком. Слід також зауважити, що при такому способі виклику функції не можна далі в програмі, що її

викликає, скористатися вихідним значенням цієї функції, оскільки воно буде втрачено.

Вхідні параметри у функціях, що використовуються у VBS, іноді називають аргументами цих функцій, щоб підкреслити «пряму спорідненість» останніх з функціями математики.

Однак у розпорядженні VBScript є набір вбудованих функцій, які безпосередньо реалізують елементарні математичні функції. Всі вони залежать тільки від одного аргументу. Описи деяких з них наведені в таблиці 4.4.

Таблиця 4.4 Виклики деяких стандартних математичних функцій у VBS

Ім'я функції	Опис значення аргументу	Опис функції
Sin	Число в радіанах	Обчислює синус від значення аргументу
Cos	Число в радіанах	Обчислює косинус від значення аргументу
Tan	Число в радіанах	Обчислює тангенс від значення аргументу
Atn	Число, що лежить між $-\pi/2$ та $\pi/2$	Обчислює арктангенс від значення аргументу
Sqr	Число, більше нуля	Обчислює квадратний корінь від значення аргументу
Exp	Число	Обчислює експоненту від значення аргументу
Log	Число, більше нуля	Обчислює натуральний логарифм від значення аргументу
Abs	Число	Обчислює абсолютну величину (модуль) значення аргументу
Round	Має два аргументи: перший — саме число, другий (опційний параметр) — кількість знаків після коми	Повертає число, округлене до зазначеної кількості знаків після коми за правилами бухгалтерського округлення

З першої частини курсу відомо, що в обчислювальному алгоритмі розрахунки виконуються переважно за допомогою реалізації виразів.

Виразом (обчислювальним виразом) називають запис обчислення, до якого входять значення констант, імена змінних і констант, імена (виклики) обчислювальних функцій, що пов’язані між собою символами обчислювальних операцій.

Розрізняють вирази арифметичні, логічні і символьні.

Арифметичні вирази — це записи обчислення, результатом якого є числове значення (число). Арифметичні вирази відповідають відомим з математики алгебраїчним виразам. Вони складаються зі значень числових констант, імен числових змінних величин і числових констант, імен математичних функцій, з’єднаних один з одним знаками арифметичних операцій.

Порядок виконання операцій в арифметичних виразах — загальноприйнятий у математиці (за пріоритетом математичних операцій):

1) математичні функції;
2) унарний мінус;
3) піднесення до степеня;
4) множення і ділення;
5) цілочисельне ділення;
6) отримання залишку від цілочисельного ділення;
7) додавання і віднімання.

Послідовність операцій одного пріоритету виконується зліва направо. Круглі дужки «(...)» можуть використовуватися для зміни вищеназваного порядку виконання операцій. Спочатку виконуються операції всередині дужок, потім поза ними. Якщо використовується вкладення дужок, то спочатку виконуються дії у внутрішніх дужках, потім — у зовнішніх.

Наведемо кілька прикладів арифметичних виразів, наданих термінами математичного запису та рядком символів мови VBScript:

Математичний запис обчислення	Запис обчислення мовою VBS
$\frac{ab}{c}$	a * b / c
$\frac{x+2}{c+d}$	(x + 2) / (c + d)
$\frac{a-\sqrt{d}}{2x}$	(a – Sqr(d)) / (2 * x)
$\sin^2 x - \cos x^2$	Sin(x)^2 – Cos(x ^2)
$3a^2 + \frac{b}{4} + \frac{7}{1-a}$	3 * a^2 + b / 4 + 7 / (1 – a)

Логічні вирази — це записи обчислення, результатом якого є логічне значення (True або False). Логічні вирази складаються зі значень логічних констант та імен логічних змінних, що з'єднують символами логічних операцій. Порядок виконання логічних операцій задається згідно з пріоритетом: 1) заперечення (Not); 2) кон'юнкція (And); 3) диз'юнкція (Or). Послідовність однакових операцій виконується зліва направо. Інші логічні операції (небазові) у складних логічних виразах, як правило, не використовуються. Для зміни послідовності виконання операцій у логічних виразах теж використовуються круглі дужки і допускаються вкладення дужок у дужки.

Слід зауважити, що, оскільки результат будь-якої операції відношення є логічним значенням, записи цих операцій (звичайно з їхніми числовими операндами) часто вставляються в логічні вирази поряд з логічними значеннями та іменами логічних змінних. При обчисленні таких комбінованих виразів операції відношення мають пріоритет вище, ніж логічні операції.

Символьні вирази — це записи обчислення, результатом якого є символьний рядок. До такого запису можуть входити значення символьних констант, імена символьних змінних і констант, а також імена (виклики) символьних функцій, що пов'язані між собою символами операції конкатенації (зчеплення).

У таблиці 4.5 наведені виклики деяких стандартних функцій для обробки символьних рядків.

Таблиця 4.5 – Виклики деяких стандартних функцій у VBS для обробки символьних рядків

Формат виклику функції	Опис функції
Left(<рядок>**,** _ <довжина підрядка>**)**	Повертає задану кількість символів (підрядок) з початку рядка
Right(<рядок>**,** _ <довжина підрядка>**)**	Повертає задану кількість символів (підрядок) з кінця рядка
Mid(<рядок>**,** _ <поч. позиція підрядка>**[,** _ <довжина підрядка>**])**	Повертає задану кількість знаків (підрядок) від заданої початкової позиції в рядку
Len(<рядок>**)**	Повертає число, що відповідає кількості символів у рядку
Trim(<рядок>**)**	Повертає рядок, де видалені лідируючі та замикаючі пробіли
Ltrim(<рядок>**)**	Повертає рядок з віддаленими лідируючими пробілами
Rtrim(<рядок>**)**	Повертає рядок з віддаленими замикаючими пробілами
Lcase(<рядок>**)**	Повертає той самий рядок, але з усіма алфавітними символами в нижньому регістрі
Ucase(<рядок>**)**	Повертає той самий рядок, але з усіма алфавітними символами у верхньому регістрі
StrReverse(<рядок>**)**	Повертає той рядок, але з переставленими символами задом-наперед
String(<кількість>**,** _ < символ>**)**	Повертає рядок із заданої кількості заданих символів

Параметри <рядок>, <символ> в описах форматів функцій з таблиці 4.5 в коді сценаріїв можуть бути символьними константами, іменами змінних, значення яких є відповідними рядками, і навіть виразами, обчислення яких дають потрібний символьний рядок. Параметри <кількість>, <довжина підрядка>,

<поч. позиція підрядка> є цілими числами; якщо задаються константами, можуть також бути змінними з відповідними числовими значеннями та виразами, що обчислюють потрібні числа.

Вище було показано, що для задавання будь-якого обчислення в сценарії VBScript треба скласти відповідний вираз. Інтерпретатор VBS обчислює вираз і отримує результат у вигляді числового, логічного значень або символьного рядка. Однак результат цей треба десь зберегти, перш ніж інтерпретатор приступить до виконання нової дії відповідно до сценарію, інакше поточний результат буде втрачено. З початкової частини курсу відомо, що результати проміжних обчислень зберігаються в змінних величинах. Для задавання такої дії в сценарії VBS застосовується оператор присвоювання. Він має такий формат:

[**Let**] <ім’я змінної результату> **=** <вираз> ,

де <ім’я змінної результату> — конкретне ім’я (ідентифікатор) змінної, якій інтерпретатор VBS «повідомить» значення обчисленого виразу, що наведено праворуч від знака «=» в записі оператора;

<вираз> — запис згідно з синтаксисом VBS будь-якого допустимого виразу, що буде обчислено інтерпретатором при виконанні оператора.

Початковий ключ оператора присвоєння був його обов’язковим атрибутом у ранніх діалектах Basic. Тепер у сучасних діалектах цієї мови конструювання присвоєння може бути зроблено без нього: функцію початкового ключа виконує знак «=», який тут називають «знаком присвоювання». Однак, якщо ключ «Let» все ж таки буде застосований у сценарії, інтерпретатор VBScript його обробить правильно.

Приклади рядків коду сценарію VBS з присвоєннями:

```
a = 3 : b = 4 : a1 = True : b1 = False : d = "ABC"
c = Sqr(a ^ 2 + b ^ 2)                ' c = 5
f = (a > b or a1) and True or b1      ' f = True (Істина)
k = d & "DEF"                         ' k="ABCDEF"
```

У першому рядку вищенаведеного прикладу задано п’ять операторів присвоєння. Записи кожного з них короткі, і вони візуально легко читаються в одному рядку коду. Розподільниками між ними слугують знаки двокрапки. У другому рядку коду наводиться оператор присвоєння з арифметичним виразом, у третьому рядку — з логічним виразом, у четвертому — з символьним виразом. Апострофами від рядків коду відокремлені коментарі. Їх присутність у коді інтерпретатор при виконанні коду ігнорує. У коментарях вказані значення змінних після виконання всіх наведених у прикладі операторів присвоєння.

5 НАЙПРОСТІШИЙ СПОСІБ ВВЕДЕННЯ І ВИВЕДЕННЯ ДАНИХ У СЦЕНАРІЯХ VBSCRIPT

Розглянемо найпростіший (але не найзручніший) спосіб «спілкування» сценарію VBS з користувачем. У складі бібліотеки вбудованих функцій для VBS є дві функції, що дозволяють організувати такий процес. Мова VBScript не розроблялася для написання програм зі складним інтерфейсом користувача, але вона може забезпечувати виведення простих повідомлень і дає змогу користувачеві вводити до програмного середовища прості рядки за допомогою наявних у ній стандартних функцій MsgBox та InputBox.

У своїй найбільш загальній формі функція MsgBox виводить текст повідомлення користувачу. Однак функцію MsgBox можна також використовувати для задавання простих запитань користувачеві, на які передбачається відповідь «так» або «ні». Функція MsgBox відкриває на екрані дисплея спеціальне вікно, у яке виводяться всі названі дані.

Наприклад, якщо в сценарії реалізувати фрагмент коду

```
x = 33
MsgBox "Значення змінної X дорівнює " & x, , _
"Перевірка коду"
```

то на екрані виникне вікно, наведене на рисунку 5.1.

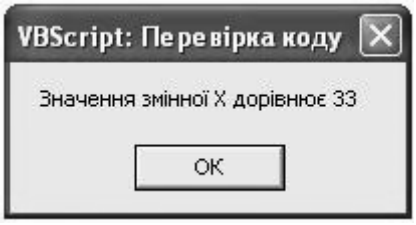

Рисунок 5.1 – Приклад вікна, що виводиться функцією MsgBox

Загальний формат виклику функції MsgBox у сценаріях такий:

MsgBox <повідомлення>**,** <тип набору кнопок>**,** <заголовок> ,

де <повідомлення> **—** символьна величина, яка, власне, є повідомленням, що виводиться;

<тип набору кнопок> — ім’я відповідної стандартної константи або її значення, або сума з двох таких констант. Цей параметр встановлює для вікна, що виводиться на екран, тексти написів на кнопках, кількість кнопок і визначає піктограму, яка виводиться в діалоговому вікні поруч з текстом повідомлення;

<заголовок> — символьний величина, яка відображується у верхній частині діалогового вікна, що виводиться в результаті виконання функції.

Будь-який з останніх двох параметрів (або останні два параметри) у виклику цієї функції можуть бути відсутні. Звичайно, якщо відсутній середній параметр, то коми, які його оточують, повинні зберігатися. Якщо ж відсутній третій (крайній правий) параметр, то зберігається кома тільки між першим і другим параметрами.

Коли вихідна величина функції MsgBox використовується в сценарії, то функцію викликають згідно з форматом, у якому використовуються дужки:

MsgBox (<повідомлення>**,** <тип набору кнопок>**,** <заголовок>**)** .

Сенс позначень <повідомлення>, <тип набору кнопок>, <заголовок> тут такий самий, як і в описі формату виклику функції, де відсутні дужки.

Для формування значення параметра <тип набору кнопок> підсумовуються значення стандартних констант (тобто констант, імена і значення яких «відомі» інтерпретатору VBS завжди) з двох наборів, поданих у таблицях 5.1, 5.2.

Таблиця 5.1 — Стандартні константи, що задають кількість кнопок і текст написів на них для вікна функції MsgBox

Ім’я константи	Значення	Відображувані кнопки
VbOKOnly	0	OK
VbOKCancel	1	OK та Скасування (Cancel)
VbAbortRetryIgnore	2	Перервати (Abort), Повтор (Retry), Пропустити (Ignore)
VbYesNoCancel	3	Так (Yes), Ні (No), Скасування (Cancel)
VbYesNo	4	Так (Yes), Ні (No)
VbRetryCancel	5	Повтор (Retry), Скасування (Cancel)

Таблиця 5.2 — Стандартні константи, що задають піктограми (зображення-символи), що виводяться у вікні функції MsgBox поруч з текстом повідомлення

Ім’я константи	Значення	Значок, що відображується
VbCnitical	16	Важливе повідомлення
VbQuestion	32	Питання
VbExclamation	48	Вигук
VbInformation	64	Інформація

Щоб, наприклад, отримати значення другого параметра для функції MsgBox, яка розкриє вікно з двома кнопками «Так» (Yes), «Ні» (No) і піктограмою «Питання», треба його значення задати виразом VbYesNo + VBQuestion або числом 36.

Виводить MsgBox ціле додатне число, що відповідає віртуальній кнопці, яка була «натиснута», згідно з даними таблиці 5.3.

Таблиця 5.3 – Значення, що повертаються функцією MsgBox, та імена стандартних констант, які їм відповідні

Ім’я константи	Значення
VbOK	1
VbCancel	2
VbAbort	3
VbRetry	4
VbIgnone	5
VbYes	6
VbNo	7

Функція MsgBox виводить діалогове вікно, що залишається на екрані, поки користувач не «натискає» на віртуальну кнопку. Це гальмує загальний процес обчислень в усій обчислювальній системі. Тому, якщо потрібно забезпечити виконання сценарію в автоматичному режимі сумісно з іншими обчислювальними діями комп’ютера, цю функцію не слід використовувати. Названа особливість функції MsgBox буде детально розглянута пізніше.

Серед вбудованих функцій VBScript є також функція, що дозволяє користувачеві ввести до пам’яті комп’ютера простий текстовий рядок. Ця функція має ім’я InputBox. При її виклику відкривається вікно з полем введення, куди користувач за допомогою клавіатури може ввести послідовність символів (максимальна довжина послідовності 256 символів). На відміну від функції MsgBox, при використанні функції InputBox завжди треба отримувати значення, яке, власне, і є тією послідовністю символів, що була введена користувачем з клавіатури, тому виклик цієї функції в сценарії завжди оформляється з застосуванням дужок. Крім того, при виклику вказується список вхідних параметрів. Загальний формат виклику функції InputBox такий:

InputBox (<повідомлення>**,** <заголовок>**,** <ряд. за замовч.>**,** _
<коорд. X>**,** <коорд. Y>**)**

де параметри <повідомлення>, <заголовок> такі самі, як в описах формату функції MsgBox;

<ряд. за замовч.> — рядкове значення в полі введення, яке буде показано за замовчуванням при відкритті вікна;

<коорд. X>, <коорд. Y> — координати лівого верхнього кута вікна в одиницях twips (1440 twips = 1 дюйм, 567 twips = 1 см) відносно лівого верхнього кута екрана. Якщо координати не вказані, то вікно виводиться приблизно в центрі екрана.

Всі елементи списку параметрів, крім першого, у виклику функції InputBox можуть бути опущені. При опусканні параметра коми, що його обмежують, у списку зберігаються. Однак список параметрів не повинен закінчуватися комою.

Наведемо приклад фрагмента коду, у якому використовуються виклики обох функцій:

```
UserName = InputBox("Введіть, будь ласка, своє ім'я.")
MsgBox "Привіт, " & UserName & "!"
```

При його виконанні інтерпретатор спочатку змусить з’явитися на екрані вікно, зображене на рисунку 5.2. Потім після введення користувачем у текстове поле вікна символьної послідовності (для демонстрації роботи прикладу було введено рядок «Віктор») і «натискання» на кнопку «OK» відкриється вікно, зображене на рисунку 5.3.

Рисунок 5.2 — Вікно, що виводиться функцією InputBox при реалізації прикладу фрагмента коду

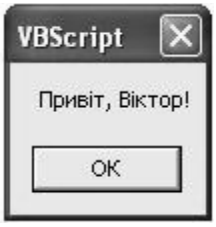

Рисунок 5.3 — Вікно, що виводиться функцією MsgBox при реалізації прикладу фрагмента коду

Зауважимо, що функція InputBox при відкритті свого вікна, як і функція MsgBox, вводить загальний процес обчислень комп'ютера в стан очікування.

6 ПРОГРАМУВАННЯ ЛІНІЙНОГО ОБЧИСЛЮВАЛЬНОГО ПРОЦЕСУ ЗАСОБАМИ VBSCRIPT

Відомостей, наведених вище, досить, щоб скласти сценарій лінійного обчислювального процесу з організацією введення початкових даних і виведенням розрахованого результату. Розглянемо послідовно його створення на прикладі. Раніше в курсі наводилася схема алгоритму лінійного обчислення формули

$$y = 3 + \frac{\ln\left(x^2 + 1\right)}{2x + a - 5}, \text{ якщо } x = \frac{a + b}{b^2}, \; a = 3{,}45.$$

У схемі весь процес розбивався на сім виконавчих блоків. Після блока початку алгоритму вводилася величина *b*, далі величина *a* отримувала значення 3,45, потім обчислювалося значення змінної *x*, далі обчислювалися чисельник дробу, знаменник дробу і робився завершальний розрахунок величини *y*. Останньою дією перед блоком кінця алгоритму було виведення значення величини *y*.

Згідно з зазначеною вище послідовністю дій запишемо відповідні інструкції мови VBScript. Для забезпечення введення і виведення даних застосуємо вбудовані функції MsgBox та InputBox. Отримаємо такий код:

```
b = InputBox("Введіть значення для величини b:", _
             "Вікно введення даних")
a = 3.45
x = (a + b) / b ^ 2
h = Log(x ^ 2 + 1)
z = 2 * x + a - 5
y = 3 + h / z
MsgBox "Значення вихідної величини y дорівнює " & y, _
       , "Вікно виведення даних"
```

Виконати сценарій VBS у середовищі операційної системи Windows можна декількома способами. Але оскільки VBScript використовується для адміністрування систем Windows, майже всі вони є потенційно небезпечними для функціонування цих систем. Далі буде детально розглянуто досить безпечний, простий і зручний один з них.

7 ВИКОНАННЯ СЦЕНАРІЮ VBS ЗА ДОПОМОГОЮ БРАУЗЕРА MS INTERNET EXPLORER

Незважаючи на те, що фірма Microsoft використовує мову VBScript досить широко у своїх розробках, скриптові мови (до яких належить і VBScript) з'явилися перш за все для забезпечення найбільшого і грандіозного проекту Internet — служби WWW. Тому програми мовою VBS найкраще виконувати в браузері Microsoft Internet Explorer (IE).

Основна функція будь-якого браузера (і IE теж) полягає в читанні спеціальних електронних документів, які він отримує по Internet від вузлових серверів, і візуалізації їх вмісту у своєму вікні на екрані клієнтського комп'ютера. Ці електронні документи є символьним поданням спеціальними мовами функціональної розмітки. Однією з таких мов (найпопулярнішою, але не єдиною) є мова HTML (HyperText Markup Language — мова розмітки гіпертексту). Електронні документи, складені за допомогою цієї мови, зазвичай називають HTML-документами.

Такий документ є описом його компонентів і їх взаємного розташування для однієї веб-сторінки (англ. web page), тобто того,

що буде показано браузером в одному відкритому його вікні. Компонентами веб-сторінки можуть бути фрагменти тексту, картинки, віртуальні панелі управління плеєрів для відео та звуку, інші графічні пасивні й активні елементи (поля для введення тексту, віртуальні клавіші, кнопки та ін.). Але найголовніше для наших цілей, компонентами веб-сторінки можуть бути програми, написані скриптовими мовами. Ці програми виконуються інтерпретатором у браузері, щоб реалізувати динаміку зміни відображення деяких активних компонентів сторінок.

Можливістю застосування названих компонентів для веб-сторінок скористаємося і ми, вставивши до HTML-документа в якості такого компонента текст обчислювальної програми мовою VBScript.

Щоб скласти символьний код потрібного для нас HTML-документа, необхідно ознайомиться з основами HTML.

Управляючі конструкції мови HTML називаються тегами (англ. tag — іменована мітка) і вставляються безпосередньо в текст документа. Всі теги укладаються в кутові дужки «<...>». Відразу після дужки відкривання поміщується ключове слово, яке визначає тег, наприклад <DIV>. Теги HTML бувають парними і непарними. Непарні теги впливають на весь документ або визначають разовий ефект у місці своєї появи. При використанні парних тегів у документ додаються теги відкривання і закривання, які впливають на частину документа, укладену між ними. Тег закривання відрізняється від тегу відкривання наявністю символу «/» (прямий слеш) перед ключовим словом (</ DIV>). Закриття парних тегів виконується так, щоб дотримуватися правил вкладення.

Ефект застосування тегу може видозмінюватися шляхом додавання атрибутів (властивостей). У парних тегах атрибути додаються тільки до тегу відкривання. Атрибути являють собою додаткові ключові слова, які відділяються від ключового слова, що визначає тег, і від інших атрибутів пробілами і розміщуються перед символом «>», що закінчує тег. Спосіб застосування деяких атрибутів вимагає надання значення атрибуту. Значення атрибута відділяється від ключового слова атрибута символом «=» (знак рівності) і укладається в подвійні або одинарні прямі лапки.

Приклад застосування парного тегу в тілі HTML-документа:

```
<H1 ALIGN=" Center"> Заголовок якогось тексту </H1>
```
.

Такий запис змусить IE вивести по центру відкритого вікна з певними відступами зверху і знизу від іншого вмісту фразу «Заголовок якогось тексту» дуже великим жирним шрифтом.

У наведеному прикладі парного тегу те, що виводиться у вікні браузера відносно горизонталі вікна, визначається вмістом, укладеним між кутовою дужкою закривання початкового тегу і кутовою дужкою відкривання кінцевого тегу. Ключ тегу «H1» повідомляє браузеру, що цей парний тег виводить заголовок тексту, який повинний бути представлений найбільшим жирним шрифтом з усіх тих, які використовуються браузером для виведення заголовків. Це випливає з того, що для задавання заголовків з більш дрібними жирними шрифтами в HTML-документі використовуються теги з ключами «H2», «H3» і так до «H6», теги з останнім із ключів визначають найдрібніший для заголовків шрифт. Мається на увазі, що сам текст повинен слідувати нижче у вікні браузера, і описуватися він, звичайно, повинен іншими тегами, які слідують у документі після цього парного тегу. Про те, що заголовок має виводитися по горизонталі в середині вікна, повідомляє браузеру атрибут ALIGN (вирівняти) зі значенням Center (центр) у початковому тегу.

Слід зауважити, що ті параметри про виведення HTML-документа, не повідомлені безпосередньо в його HTML-коді, браузер задає для візуалізації останнього у своєму вікні «за замовчуванням». Для різних браузерів і різних версій браузерів установлення значень «за замовчуванням» різні, тому часто має сенс, щоб вигляд сторінки, що відображується, був завжди коректним, у HTML-коді сторінки вказувати всі основні атрибути тегів.

Додамо, що в HTML-документі ключі тегів, їхні атрибути зі значеннями можна задавати як літерами верхнього регістра, так і нижнього регістра.

Всі HTML-документи мають одну і ту саму структуру, яка визначається фіксованим набором спеціальних тегів структури. HTML-документ завжди повинен починатися з тегу <HTML> і закінчуватися відповідним тегом закривання </HTML>. Усередині документа виділяються два основні розділи: розділ заголовків і

тіло документа, які йдуть саме в такому порядку. Розділ заголовків містить інформацію, що описує документ у цілому, і обмежується тегами <HEAD> та </HEAD>. Зокрема розділ заголовків повинен містити назву документа, обмежену парним тегом <TITLE>.

Основний зміст міститься в тілі документа, яке обмежується парним тегом <BODY>. Строго кажучи, положення структурних тегів у документі неважко визначити, навіть якщо вони опущені. Тому стандарт мови HTML вимагає тільки наявності тегу <TITLE> (і відповідно </TITLE>). Проте при створенні документа HTML опускати структурні теги не рекомендується.

Розглянемо питання про розміщення сценарію VBScript в HTML-коді веб-сторінки. Вихідний код сценарію мовою VBScript повинен розміщуватися всередині парного тегу <SCRIPT>, який у свою чергу може бути вставлений як всередину парного тегу <HEAD>, так і всередину парного тегу <BODY>. При цьому початковий тег <SCRIPT> має обов'язково оголошувати атрибут LANGUAGE зі значенням VBScript.

Простий приклад коду HTML-документа, що містить всі теги, які визначають структуру документа і місця розташування коду сценарію мовою VBScript, може виглядати так (у фігурних дужках туди додані пояснення, де слід змінювати або розширювати код документа для вирішення конкретних задач):

```
<HTML>
<HEAD>
<TITLE>{ Сюди вставляється заголовок документа } </TITLE>
<SCRIPT LANGUAGE="VBScript">
   <!--
      { Сюди можна вставляти код сценарію мовою VBScript }
   -->
</SCRIPT>
</HEAD>
 <BODY>
      { Сюди можна вставляти інші теги HTML-документа }
```

Продовження програми:

```
<SCRIPT LANGUAGE="VBScript">
  <!--
      { Сюди можна вставляти код сценарію мовою VBScript }
  -->
 </SCRIPT>
      { Сюди можна вставляти інші теги HTML-документа }
</BODY>
</HTML>
```

Зауважимо, що два теги «<!--» і «-->» мови HTML, які наведені в прикладі, повідомляють браузеру, що укладені між ними символи є коментарем і, отже, їх треба ігнорувати при візуалізації документа. Застосування цих тегів у зазначених місцях може іноді запобігти некоректній роботі браузера. Зазвичай сучасні браузери, які не підтримують оголошену мову сценарію, просто «не помічають» все, що розташовано в парному тезі <SCRIPT>. Однак для деяких більш давніх браузерів потрібно, щоб все, що знаходиться в такому тезі, було оформлено у вигляді HTML-коментарів. У протилежному випадку весь код сценарію буде виведений на екран як звичайний текст.

Наведений приклад коду HTML-документа без вставлених до нього пояснень рекомендуємо набрати окремим файлом у текстовому редакторі, дати файлу ім’я з розширенням *.htm і використовувати далі як шаблон для швидкого оформлення програм мовою VBScript при їх розрахунку в браузері IE.

8 ПРОЦЕДУРНИЙ СТИЛЬ ПРОГРАМУВАННЯ І ОБРОБНИКИ ПОДІЙ

Якщо ви вже «випробовували» роботу прикладу сценарію лінійного обчислення, наведеного вище, за допомогою вставлення його до файлу шаблону HTML-документа і «відкриття» останнього у вікні IE, то, без сумніву, помітили деякі особливості роботи браузера.

Безпосередньо після завантаження HTML-документа з названим сценарієм відкривається діалогове вікно «Вікно введення даних», у якому є текстове поле для введення даних із клавіатури, і якщо користувач не «натискає» за допомогою миші на яку-небудь з віртуальних кнопок цього вікна, то виконання робіт у браузері призупиняється. При цьому не можна відкрити в ньому новий HTML-документ, не можна згорнути або закрити вікно браузера за допомогою віртуальних кнопок, розташованих у правому верхньому куті смуги заголовка вікна програми. Якщо все ж таки ввести дані у «Вікно введення даних» і закрити вікно, то з’являється вікно повідомлення з результатом розрахунку. Спроба користувача не закривати його також призведе до блокування будь-яких дій у середовищі браузера.

Така ситуація при реалізації сценарію VBScript в IE дуже небажана, оскільки браузер являє собою клієнтську частину забезпечення однієї з найбільш затребуваних онлайнових служб, роботу якої переривати навіть для окремого клієнтського ПК не слід. Тому застосування вбудованих функцій InputBox і MsgBox у сценаріях VBScript намагаються уникати.

Але тоді потрібна альтернатива, яка дозволить забезпечити зручне введення даних у робоче середовище сценарію і дасть засоби виведення даних з неї за допомогою візуалізації. Такий шлях реалізації сценаріїв VBS існує, хоча він вимагає зміни стилю написання сценаріїв.

Вище вже зазначалося, що код сценарію рідко являє собою одну програмну одиницю, у якій оператори йдуть рядок за рядком, а виконання операторів починається з першого рядка і закінчується виконанням оператора останнього рядка коду. Сценарій звичайно складається з декількох програмних одиниць (процедур, модулів). І серед них є одна головна процедура

(з неї-то і починається виконання сценарію), а інші процедури є підлеглими головній процедурі (головна процедура в процесі своєї роботи викликає їх до виконання інтерпретатором, причому часто багаторазово) або ж спеціальному програмному апарату, який називають обробником подій.

Цей апарат підтримується спільно як операційною системою, так і великою частиною її додатків (зокрема IE). Однією з цілей функціонування обробників подій є забезпечення «незалежності» роботи програмних процесів, спільно ініційованих в обчислювальному середовищі операційної системи, але повністю не завершені один відносно одного.

Під подіями, що відбуваються під час виконання обчислювального процесу, розуміють зміни стану деяких його компонент або етапів, які можуть «відслідковуватися» і «відзначатися» іншими спільно триваючими, але зовнішніми відносно першого, обчислювальними процесами. Такими подіями, наприклад, є кінець завантаження і виконання будь-якого фрагмента програми, завершення введення ланцюжка кодів символів у буферну область оперативної пам’яті, що пов’язана з клавіатурою, акти натискання якихось кнопок на корпусі пристрою «миша», коли його покажчик знаходиться в заданому районі екрану, і т. д.

Обробники подій — це процедури, які виконуються в разі настання певних подій з ініціативи зовнішніх обчислювальних процесів, що «відстежують» ці події.

Запровадження обробників подій до складу загального коду сценарію VBScript дозволяє задати його виконання новим способом. Розглянемо цей спосіб організації процесу обчислень у сценарії докладніше.

Процесор комп’ютера виконує обчислювальні процеси, якщо їх було розпочато кілька, послідовно (застосування паралельних обчислень — підхід «некласичний», і тут не розглядається). Тобто спочатку виконується один обчислювальний процес до кінця або до якогось певного моменту, далі — інший процес до кінця або до якогось моменту, потім — третій процес і т. д. Таке «поступове» виконання обчислювальних процесів комп’ютером може повторюватися, поки вони всі не будуть виконані. Зрозуміло, що коли в якомусь фрагменті обчислювального процесу з’являється

повільна дія, то вона буде гальмувати ні тільки виконання свого обчислювального процесу, а й виконання інших. Таких дій в обчислювальних процесах треба уникати в застосуванні, але вони — потрібні. Однак зауважимо, що до цих дій, як правило, належать дії обміну даними між ОП і зовнішніми пристроями комп'ютера, тобто ті дії, які безпосередньо не мають потреби в «послугах» процесора. Тому добрим виходом з такої ситуації є виключення цих дій із загальної низки обчислень, що обслуговуються процесором, тим більше, що при їх виконанні обчислювальна компонента процесора (АЛП) «відпочиває».

Застосування обробників подій дозволяє розбити сценарій VBScript на фрагменти так, щоб у них не було повільних дій. На рисунку 8.1 наведена найпростіша структура організації сценарію, де умовно зображено зміст сценарію і шляхи обміну даними між усіма учасниками обчислювального процесу.

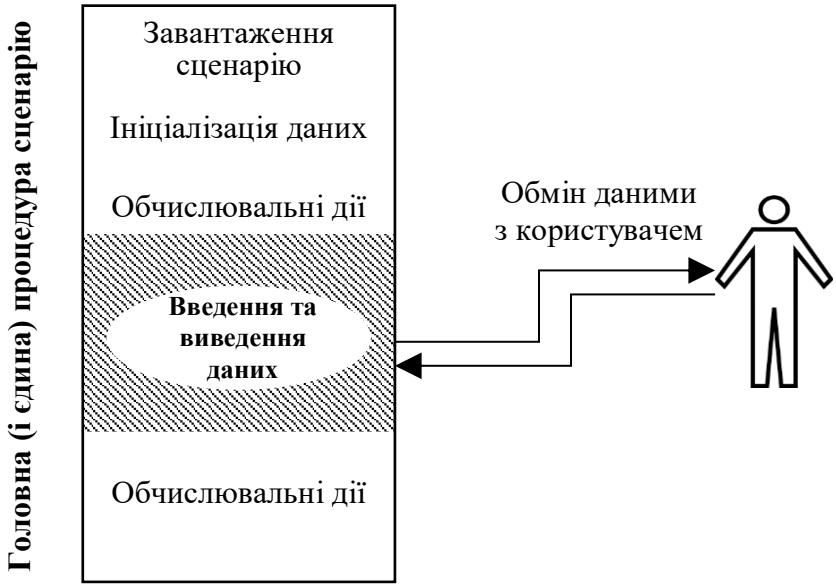

Рисунок 8.1 — Структура організації простого сценарію

Штриховкою на структурній схемі позначені найповільніші операції сценарію. Звичайно, користувача (людину, яка працює за комп'ютером), який вказаний на схемі, не можна віднести до зовнішніх пристроїв, з якими сценарій веде обмін даними. Однак користувач безпосередньо управляє роботою таких пристроїв, наприклад клавіатурою, пристроєм вказівки «миша» і їм подібних, від яких у робоче середовище сценарію надходять необхідні для

обчислень відомості. Тому введення до складу схеми фігурки користувача цілком обґрунтовано. І участь людини в діях сценарію, зрозуміло, може загальмувати закінчення його виконання на невизначений час. Зауважимо, що наведений вище сценарій лінійного обчислення своєю структурою багато в чому (у ньому відсутні тільки дії з ініціалізації даних) відповідає схемі рисунку 8.1. Він має всі зазначені недоліки.

На рисунку 8.2 наведена структурна схема сценарію, виконаного в процедурному (модульному) стилі програмування. Такий сценарій ще не позбавлений повільних дій (на схемі вони всередині процедур, як і раніше, позначені штриховкою). Більш того, як показано на рисунку 8.2, повільні операції, на швидкість виконання яких безпосередньо впливає користувач, можуть мати місце не тільки в головній процедурі, а й у допоміжних процедурах.

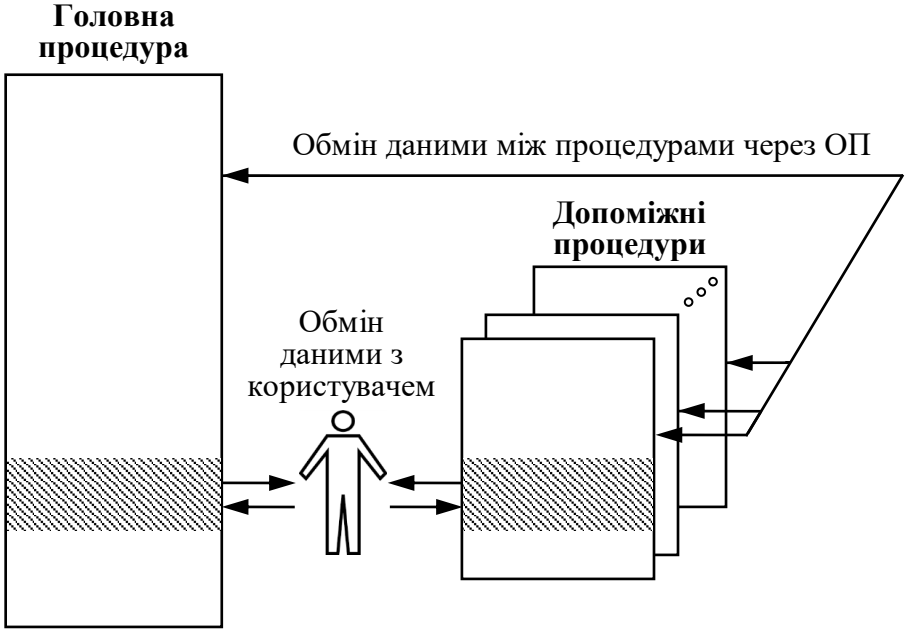

Рисунок 8.2 — Структура організації сценарію, виконаного процедурним (модульним) способом програмування

Однак сценарії з модульною організацією все ж таки мають багато переваг, якщо їх порівнювати зі сценаріями, заданими однією процедурою. По-перше, за обсягами коду вони коротші, оскільки одні й ті самі допоміжні процедури можуть виконуватися в сценарії багато раз. По-друге, сценарії з багатопроцедурною організацією, як правило, розробляються швидше, ніж довгі за кількістю рядків коду однопроцедурні сценарії, оскільки

програміст-розробник при створенні нового сценарію може використовувати готові процедури, взяті ним зі своїх «старих» розробок, у правильності роботи яких він впевнений. По-третє, програміст-розробник може використовувати як компоненти сценарію, що розробляється, не тільки свої «старі» процедури, але і процедури своїх колег, що не тільки прискорює процес розроблення сценарію, а й робить його колективним.

Наступним кроком вдосконалення структури сценаріїв VBScript є введення до їхнього складу особливих допоміжних процедур — обробників подій. На рисунку 8.3 наведена структурна схема такого сценарію.

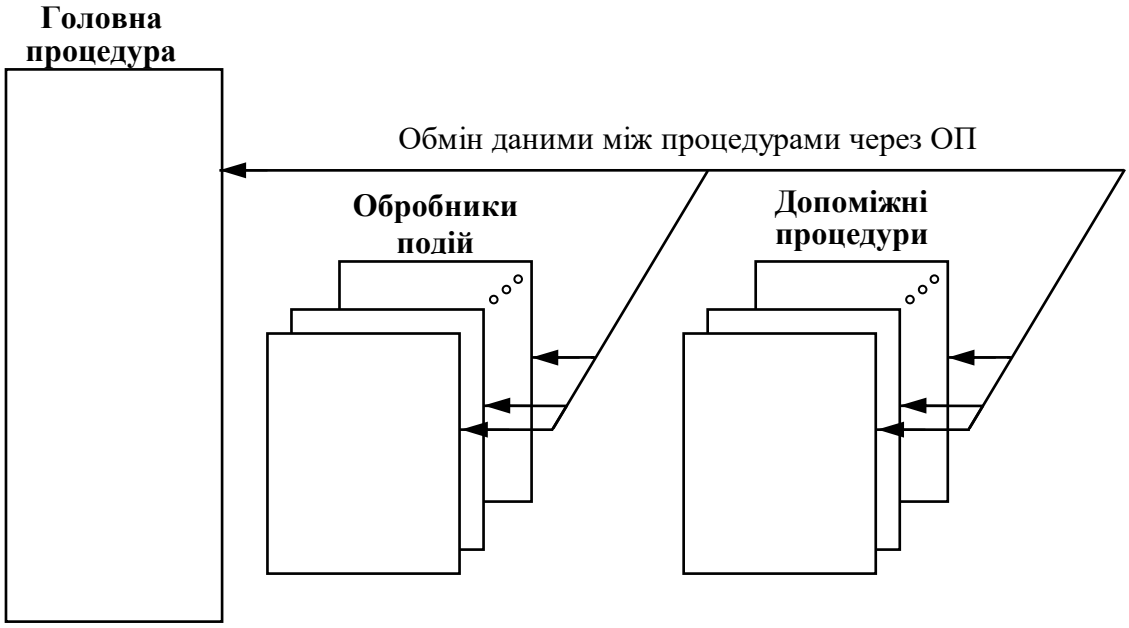

Рисунок 8.3 — Структура організації сценарію, виконаного процедурним способом із застосуванням обробників подій

Характерною відмітною її властивістю є відсутність як у головній процедурі, так і допоміжних процедурах повільних дій, швидкість виконання яких пов’язана з активністю користувача. Тому у схемі прямокутники, що позначають тексти процедур, не мають смуг штриховки, і зі схеми видалена фігурка людини-користувача.

Головна процедура сценарію виконується від початку і до кінця відразу ж після завантаження. При цьому викликаються нею до виконання і виконуються до кінця в потрібній послідовності і потрібну кількість раз допоміжні процедури, причому допоміжні

процедури, якщо це в них передбачено, можуть самі викликати до виконання інші допоміжні процедури і навіть самих себе. Загальний обчислювальний процес завершується без зупинок для обміну даними з користувачем. Однак якщо до його сценарію входять процедури обробки подій, всі основні дані, що визначають стан процесу, зберігаються в ОП, і вмикається зовнішній до процесу програмний апарат стеження за подіями, обробники яких входять до складу сценарію.

Інші обчислювальні процеси, ініційовані браузером за іншими сценаріями, будуть виконуватися незалежно від події, на яку «чекає» процес. Вони теж можуть входити в стан очікування «своїх» подій, і спільне виконання всіх процесів, відкритих браузером, буде відбуватися відповідно до настанням потрібних подій і за порядком організації одночасного їх обслуговування з боку ОС.

9 ЗАДАВАННЯ ПРОЦЕДУР У СЦЕНАРІЯХ VBSCRIPT

Головна процедура в сценарії VBScript ніяк не титулується. Власне наявність у HTML-документі тегу відкривання <SCRIPT LANGUAGE = "VBScript"> і тегу закривання </SCRIPT> означає, що між ними знаходяться оператори сценарію VBScript, причому оператори, які задають саме головну процедуру сценарію. Якщо між тегами <SCRIPT LANGUAGE = "VBScript"> і </SCRIPT> ніякого наповнення нема, то вважається, що головна процедура (а може, навіть єдина) сценарію все одно задана, але в ній не передбачено жодних дій. Допоміжні процедури та обробники подій зазвичай записуються після останнього оператора головної процедури перед тегом </SCRIPT>, що закриває сценарій. Ці процедури, за правилами VBScript, мають початкові і завершальні оператори.

Процедура в сценарії — це окрема секція коду, що виконує конкретну задачу. Процедури в VBScript бувають двох типів: функції і підпрограми. Функції завжди повертають в обчислювальний процес, який їх викликає, деяке своє вихідне значення, підпрограми цього не роблять.

Вище вже розглядалися деякі вбудовані функції VBScript. Це обумовлені процедури, код кожної з яких завантажується в процесі обчислення і виконується інтерпретатором, як тільки той виявляє в тексті сценарію їхні виклики. Однак програміст, який розробляє сценарій мовою VBScript, завжди може також створити і свою власну функцію. Формат задавання процедури-функції в тексті сценарію VBScript такий:

Function <ім'я функції>**(** [<параметри>] **)**
[<оператори>]
[**<**ім'я функції**> = <**вираз>]
[<оператори>]
[**Exit Function**]
[<оператори>]
[**<**ім'я функції**> = <**вираз>]
End Function

,

де <ім'я функції> — ім'я функції, складене за правилами складання ідентифікаторів у VBScript;

<вираз> — обчислення значення, яке повертає функція в точку свого виклику в сценарії;

<оператори> — група операторів, яку треба використовувати в процедурі;

<параметри> — список параметрів, що передаються процедурі-функції, коли вона викликається для виконання.

Компоненти списку параметрів відокремлюються один від одного комами. Кожен компонент списку задається відповідно до загального формату, який можна визначити так:

[**ByVal** | **ByRef**] <ім'я змінної-параметра> [**( )**] ,

де ByVal | ByRef — два режими отримання даних для змінної-параметра, ім'я якої зазначено в атрибуті <ім'я змінної-параметра>. Вибирається завжди з двох режимів тільки один або не вибирають ніякого. В останньому випадку вважається, що за замовчуванням був оголошений режим ByRef;

<ім’я змінної-параметра> — ім’я змінної, що є вхідним параметром у процедурі, складене за правилами написання ідентифікаторів у VBScript.

Ключові слова ByVal і ByRef утворені від словосполучень by value (за значенням) і by reference (за посиланням) відповідно. Існує два способи передачі даних з процедури, що викликає, у підпорядковану за значенням і посиланням. Коли дані передаються за значенням, підпорядкована процедура робить «копію» значення переданого їй вхідного даного, тому, якщо далі в процесі своєї роботи вона цю копію змінює, ніяких змін з використовуваними нею даними у процедурі, яка її викликала, не відбувається. Коли ж вхідні дані передаються в підпорядковану процедуру за посиланням, зміни можуть пройти шлях назад до процедури, що викликала першу, якщо джерелом вхідного значення в ній є змінна, а не константа або вираз. У VBScript є можливість явного оголошення аргументів як переданих за значенням ByVal або як переданих за посиланням ByRef. Як вже зазначалося вище, за замовчуванням у VBScript використовується оголошення ByRef.

В описі формату компонента списку параметрів показано, що після імені змінної-параметра можуть одна за одною слідувати круглі дужки відкривання і закривання. Ці знаки ставляться тоді, коли ім’я вхідного параметра є ім’ям масиву даних. Однак коли вхідними даними для процедури, що викликається, є значення масиву, то вони можуть передаватися в неї тільки за посиланням. Тому наявність круглих дужок після імені вхідного параметра і ключове слово ByVal перед ним виключають один одного. Такого поєднання не можна допускати.

На відміну від процедури-функції, метою виконання якої є отримання деякого значення (числа або набору символів), процедуру-підпрограму можна розглядати як частину коду сценарію, призначену для багаторазового виконання при різних вхідних даних і в різних фрагментах основного коду сценарію.

Задаються процедури-функції і процедури-підпрограми в тексті сценарію VBScript майже однаково. Формат процедури-підпрограми такий:

```
Sub <ім'я підпрограми>( [<параметри>] )
[<оператори>]
[Exit Sub]
[<оператори>]
End Sub
```
,

де <ім'я підпрограми> — ім'я підпрограми, складене за правилами складання ідентифікаторів в VBScript;

інші позначення такі самі, що і для процедури-функції.

Слід зауважити, що всі однаково задані компоненти у форматах обох процедур мають однакові властивості, правила використання та обмеження.

Розглянемо ще правила, які слід виконувати при задаванні процедур у сценаріях VBScript. Процедура (як функція, так і підпрограма) не може бути описана всередині іншої процедури (тобто не допускається використання вкладених процедур).

Оператори Exit Function та Exit Sub можна використовувати для виходу з будь-якого місця функції та підпрограми відповідно з поверненням дії в місце їх виклику з керуючого ними програмного модуля.

Виклик процедури-функції в тексті сценарію оформляється також як виклик стандартної функції, наприклад MsgBox, тобто двома способами. Перший спосіб: вставлення виклику функції в обчислювальний вираз, заданий у тексті процедури, що викликає функцію. У цьому випадку запис виклику задається ідентифікатором функції, а праворуч від нього в круглих дужках наводиться список реальних вхідних параметрів, які будуть використовуватися при даному виконанні функції. Другий спосіб: виклик функції задається як самостійний оператор у тексті процедури, що її викликає. Обидва способи були вже докладно розглянуті раніше.

Оформлення викликів процедур-підпрограм теж може бути виконано у двох варіантах. Перший шляхом задавання оператора виклику з ключем Call (англ. call — виклик). Формат завдання цього оператора такий:

Call <ім’я підпрограми> **(** [<вхідні параметри>] **)** **,**

де <ім’я підпрограми> — ім’я підпрограми, складене за правилами складання ідентифікаторів у VBScript, тобто має те саме значення, що й відповідний атрибут в описі формату задавання процедури-функції;

<вхідні параметри> — список з констант, імен змінних і виразів, значення яких використовуються як реальні вхідні дані для обчислення процедури, ім’я якої задано атрибутом <ім’я підпрограми>.

Можна не використовувати ключове слово Call при виклику підпрограми (це другий варіант виклику), тоді список <вхідні параметри> не береться в круглі дужки.

Елементи списку <вхідні параметри> в обох випадках (як при виклику функції, так і при виклику підпрограми) відокремлюються один від одного комами.

Слід звернути увагу на дуже важливу особливість узгодження апарату передачі вхідних даних між викликами процедур і їх задаванням у тексті сценарію. Список <параметри> в задаванні процедур-функцій і процедур-підпрограм є списком імен локальних змінних, яким у момент виклику процедур передаються значення з елементів списку <вхідні параметри>, визначених у процедурі, що робить виклик, шляхом їх прямого копіювання або ототожнення їхніх адрес в ОП. Інакше кажучи, співставлення значень вхідних даних між списком <параметри> і списком <вхідні параметри> відбувається тільки через взаємне розташування їхніх елементів всередині списків. Тому порядок проходження елементів в обох списках і їхня кількість мають завжди збігатися.

Слід також зауважити, що значення локальних змінних у процедурах не зберігаються між повторними викликами процедур (області пам’яті під локальні змінні використовуються тільки під час роботи процедури).

10 ПРИВ'ЯЗКА ОБРОБНИКІВ ПОДІЙ ДО КОМПОНЕНТІВ ВЕБ-СТОРІНКИ HTML-ДОКУМЕНТА

Тепер розглянемо безпосередньо питання про те, як обмінюватися даними з обчислювальним процесом, реалізованим інтерпретатором VBScript згідно зі сценарієм, у якому відсутні виклики вбудованих функцій MsgBox і InputBox. Ми знаємо, що для цього треба організувати роботу допоміжних процедур, які називають обробниками подій. Відстеження потрібних для нас подій буде забезпечувати спеціальна служба браузера IE, що активізується при візуалізації сторінки HTML-документа, куди вставляється код сценарію.

Подій, за якими можна так стежити, досить багато. Але ми розглянемо тільки дві з них: onclick і ondblclick, оскільки користуватися ними в наших цілях найбільш зручно. Подія onclick настає, коли відбувається натискання (англ. click) кнопки вказівки на пристрої «миша». Подія ondblclick (dbl від англ. double — подвійний) має місце при подвійному натисканні на кнопку вказівки на корпусі «миші».

Однак щоб оголосити обробник події, мало вказати для нього ім’я події, при настанні якої той має виконатися. Необхідно обов’язково визначити ще компонент веб-сторінки, на який повинен вказувати курсор «миші» в момент настання цієї події. Все це задається в першому рядку опису обробника події, формат якого такий:

```
Sub <компонент сторінки>_<подія>
    [<оператори>]
End Sub
```

,

де <компонент сторінки> — ідентифікатор компонента сторінки HTML-документа. На цей компонент повинен вказувати курсор «миші» в момент настання заданої події, щоб процедура обробника події почала виконуватися;

<подія> — ім’я події, яка має настати, щоб процедура обробника події почала виконуватися;

<оператори> — група операторів, яка буде використовуватися в процедурі обробника події.

Вище вже розглядалася структура і загальні правила побудови HTML-документа. Відомо, що в такому документі компоненти веб-сторінки, яку має візуально відобразити браузер у своєму вікні, записують за допомогою тегів — парних і непарних. Тегами в HTML-документі задаються як пасивні компоненти веб-сторінок (текст, картинки, фотографії, елементи декору), так й активні компоненти (віртуальні клавіші, кнопки, поля для введення символів і т. п.), тобто ті, що дозволяють приймати від користувача-клієнта в середовище браузера якісь дані для подальшої передачі їх серверу. Такі компоненти веб-сторінок зазвичай називають елементами управління.

Однак дані, які надходять від елементів управління, можна і не відсилати на веб-сервер, а залишити в програмному середовищі браузера, як це кажуть, для «внутрішнього використання». Таким прийомом ми й будемо користуватися для забезпечення процесу обміну даними між обчислювальним сценарієм VBScript і людиною-користувачем без залучення для цього функцій MsgBox та InputBox.

Елементів управління в арсеналі HTML досить багато, але ми розглянемо тільки два: поле для введення (а також виведення) текстового рядка та кнопку. Застосування цих елементів у різних поєднаннях і умовах дозволяє організувати зручний графічний інтерфейс між користувачем і обчислювальним процесом, активізованим інтерпретатором VBScript.

Елементи управління, від яких надходять дані для формування відсильного пакета, щоб передати їх однією послідовністю на веб-сервер, збираються в HTML-документі всередині парного тегу <FORM> (англ. form — форма). Але в нашому випадку, коли дані, одержувані від елементів управління, нікуди не відсилаються, можна взагалі не застосовувати в тексті HTML-документа теги <FORM>, </FORM>, оскільки останні версії IE автоматично визначають компоненти форми і відкривають доступ до них тільки за їхніми ідентифікаторами.

Самі елементи управління записуються в HTML-документ за допомогою непарного тегу <INPUT> (ввести), який, як зазначено

вище, у загальному випадку має розміщуватися між тегами <FORM> та </FORM>. Тег <INPUT> обов'язково повинен мати атрибут TYPE (тип), значення якого, власне, і визначає, який елемент управління виявиться відображеним на веб-сторінці. У нашому випадку будемо користуватися тільки одним з двох значень: Button (кнопка) або Text (текст).

Введення значення Button для атрибута TYPE в тегу <INPUT> забезпечує вставлення на відповідне місце веб-сторінки віртуальної кнопки, яка буде «натискатися», якщо на неї навести покажчик «миші» і при цьому натиснути кнопку вказівки (зазвичай ліва кнопка) на корпусі цього пристрою.

А присвоєння значення Text тому самому атрибуту того самого тегу призведе до відображення у відповідному місці веб-сторінки текстового поля, куди можна встановити «мишею» курсор і ввести з клавіатури потрібну послідовність символів.

В обох випадках зазначених значень атрибута TYPE в тегу <INPUT> слід встановити значення ще для кількох атрибутів: ID, NAME і VALUE.

Значення атрибута ID служать для внутрішньої ідентифікації компонентів веб-сторінки. Кожен парний тег у HTML-документі може мати атрибут ID (ідентифікатор) зі значенням, яке являє собою унікальну для даного HTML-документа послідовність символів, що дозволяє формально відрізняти один компонент веб-сторінки від іншого.

Ця послідовність, за правилами HTML, може складатися з латинських літер, цифр, символів дефіса та підкреслення. Крайнім лівим знаком у ній має бути обов'язково літера. Зауважимо, що в нашому випадку ці загальні правила складання ідентифікаторів у HTML треба посилити, оскільки далі зазначені ідентифікатори використовуються і в сценаріях VBScript. Тому застосування дефіса в значеннях атрибутів ID для нас неприпустимо (цей знак використовується в скриптах VBS для задавання операції віднімання).

Але головним у застосуванні ідентифікаторів для елементів управління є те, що значення атрибута ID може служити як <компонент сторінки> при прив'язці виклику процедури обробника події до відповідного елемента (див. задавання формату обробника події).

Значення атрибута NAME задає ім’я даними, що передаються від відповідного елемента управління на веб-сервер. Однак це ім’я можна використовувати і в процесі відображення веб-сторінки браузером: воно може служити дублером значенню атрибута ID, наприклад при тій самій прив’язці обробника подій до елемента управління. Тому часто значення атрибутів NAME і ID для одного елемента управління задають однаковими.

Атрибут VALUE задає набір символів, який є написом на віртуальній кнопці в разі, коли значення атрибута TYPE є Button, і початковими даними для текстового поля, які вносяться туди при ініціалізації останнього в разі, коли значення атрибута TYPE є Text.

Крім того, для випадку, коли елементом управління є текстове поле, слід у тегу <INPUT> задавати ще атрибут SIZE, значенням якого має бути натуральне число, що вказує довжину текстового поля у знакомісцях.

Тепер розглянемо питання, як за допомогою елементів управління, оголошених у HTML-документі, можна здійснювати обмін даними між відображенням веб-сторінки у вікні браузера і обчислювальним процесом, який виконується інтерпретатором VBScript за сценарієм, що вставлений у цей же HTML-документ.

Для цього згадаємо, що VBScript є об’єктно-орієнтованою мовою програмування. Пояснимо, що це означає.

Вже більше трьох десятків років як з’явилася і увійшла в застосування концепція програмування комп’ютерних систем будь-якої складності, призначених для передачі, зберігання і обробки даних, що отримала назву об’єктно-орієнтоване програмування (ООП). Відповідно до цієї концепції обчислювальний процес, програмно активізований, розглядається як відкрите віртуальне середовище, у якому функціонують деякі об’єкти. У кожного об’єкта є свої властивості, що визначають його стан, і свої методи впливу на інші об’єкти свого середовища і на себе самого. Деякі об’єкти можуть самі розглядатися як середовище для функціонування інших, поміщених у неї, об’єктів, що мають свої властивості і методи. Більш того, обчислювальний процес, який у цьому поясненні розглядається як первинне віртуальне середовище, в іншому випадку може бути теж об’єктом

зі своїми властивостями та методами в середовищі, яке є обчислювальним процесом більш високого рангу ієрархії.

Функціонально об’єкт є обчислювальним процесом, ініційованим і пов’язаним з іншими процесами. Однак обчислювальний процес завжди задається (описується) програмним кодом, тому зазвичай під терміном «об’єкт» розуміється саме програмний блок, який обробляє і переносить якісь дані. Зауважимо, що об’єкти завжди є програмним поданням чогось реального: дією або зміною стану якого-небудь пристрою, компонента або вузла пристрою, що належить до апаратної конфігурації комп’ютерної системи.

Відповідно до концепції ООП при відкритті будь-якої веб-сторінки у вікні браузера IE у віртуальному середовищі, яке, власне, і являє собою обчислювальний процес, активізований програмним кодом браузера, виникають два об’єкти, які називають Document (документ) і Window (вікно). Об’єкт Document є відображенням поточної веб-сторінки, що показує браузер у своєму вікні, а об’єкт Window являє собою поточне вікно самого Web-оглядача. Обидва об’єкти є середовищами для функціонування інших об’єктів. Document містить у собі об’єкти, що є всіма компонентами поточної веб-сторінки, Window — набір об’єктів, пов’язаних з розмірами вікна браузера, видимістю і складом його рядка меню і т. п.

До всіх цих об’єктів можна отримувати доступ із сценаріїв VBScript. Причому не тільки отримувати від них дані, але й, змінюючи значення їхніх властивостей, застосовуючи до них відповідні методи, управляти частинами, вузлами, елементами реальних пристроїв комп’ютерної системи, відображеннями яких вони є.

Однак у рамках цього курсу будуть розглянуті тільки способи читання та запису зі сценарію VBScript властивості Value для об’єктів, що становлять на веб-сторінці елементи управління, які задаються тегом <INPUT> при значеннях Button або Text для атрибута TYPE.

Нагадаємо, що значення властивості Value відповідає тексту напису на віртуальній кнопці в разі, коли об’єкт, що має цю властивість, є відображенням такої кнопки на поточній веб-сторінці у вікні браузера. А в разі, коли об’єкт, що має властивість

Value, є відображенням текстового поля, то значенням цієї властивості буде послідовність символів, яка знаходиться в поточний момент у тому текстовому полі.

Для обох випадків у сценарії VBScript це значення може бути задано як значення змінної, ім’я якої побудовано так:

[<ідентифікатор форми>**.**] <ім’я або ідентифікатор об’єкта>**.Value** **,**

де <ідентифікатор форми> — значення атрибута ID в тегу <FORM>. Якщо такого тегу в HTML-документі нема, то <ідентифікатор форми> і наступна за ним крапка відсутні;

<ім’я або ідентифікатор об’єкта> — значення атрибута NAME або ID у відповідному тегу <INPUT>.

Розглянемо нижче приклад складання коду, у якому продемонструємо задавання обробника події та обмін даними між елементами управління веб-сторінки і сценарієм.

```
<HTML><HEAD>
<TITLE> Обробник події та введення-виведення даних </TITLE>
</HEAD><BODY>
   <H2 ALIGN="Center">Додавання двох чисел</H2>
   <P>Доданок 1         Доданок 2  
              Результат<BR>
        <INPUT TYPE="Text" NAME="Slog_1" ID="Chisl_1"
        SIZE="10"> +
        <INPUT TYPE="Text" NAME="Slog_2" ID="Chisl_2"
        SIZE="10"> =
        <INPUT TYPE="Text" NAME="Summ" ID="Rezul"
        SIZE="12"><BR>
        <INPUT TYPE="Button" NAME="Knopk_1" ID="Klavish_1"
        VALUE="Обчислити !"></P>
<SCRIPT LANGUAGE="VBScript"><!--
   SUB Knopk_1_onclick
        Rezul.Value = 0 + Slog_1.Value + Chisl_2.Value
   END SUB
--></SCRIPT></BODY></HTML>
```

Цей код реалізує простий HTML-додаток, що складає два числа. Рисунок 10.1 показує, як приблизно виглядає його інтерфейс у вікні браузера. Користувач вводить у текстові поля з написами «Доданок 1» і «Доданок 2» необхідні числа з клавіатури, попередньо встановлюючи в них за допомогою «миші» текстовий курсор. Після натискання ним віртуальної кнопки «Обчислити !» у текстовому полі з написом «Результат» з'являється відповідь обчислення.

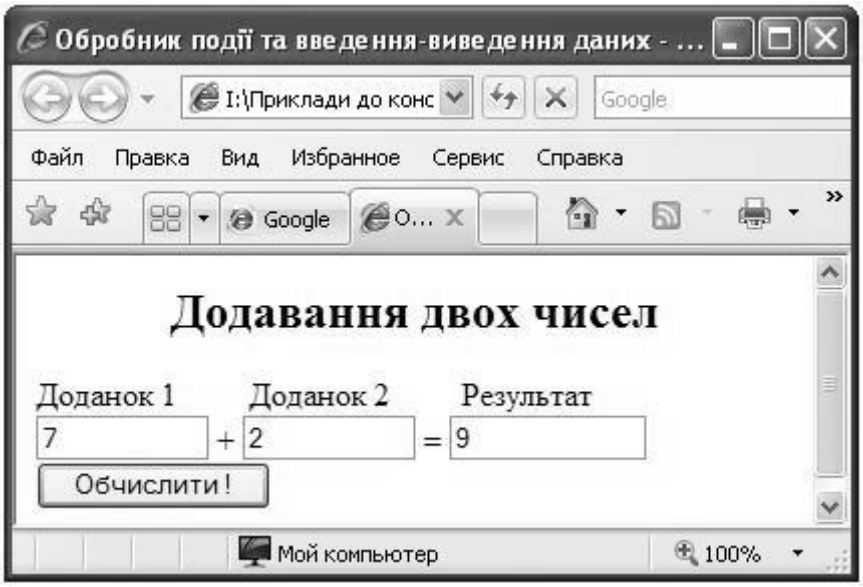

Рисунок 10.1 — Вигляд інтерфейсу HTML-додатка,
що складає два введених числа, у вікні IE

Наведений вище код HTML-документа показує, як задавати обробник події, як читати і писати символьні дані в сценарії VBScript, користуючись текстовим полем з веб-сторінки (наводиться найпростіший спосіб: без використання тегу <FORM>), демонструє можливість вільного застосування знака підкреслення в іменах та ідентифікатори елементів управління.

Крім того, у прикладі коду наводиться ще один парний тег <P> ... </P>. Цей тег служить для організації абзацу тексту (від англ. paragraph — параграф). Абзац завжди починається з нового рядка, а абзаци тексту, що йдуть один за одним, розділяються між собою проміжком. Текст в абзаці завжди форматується відповідно до ширини вікна браузера.

Втім, у всіх сучасних версіях браузерів тег закривання </P> можна не застосовувати. Більш того, сучасний браузер будь-який текст у HTML-документі, що знаходиться поза парними тегами, «сприймає» як абзац. Однак тег <P> іноді застосовувати необхідно, наприклад, щоб задати абзацу ідентифікатор. Тоді можна текст абзацу перетворити в «кнопку», прив'язавши до нього обробник події onclick або ondblclick.

Текст абзацу на сторінці подається у два або більше рядків тільки в результаті автоматичного форматування відповідно до зазначення розташування та ширини вікна браузера. Щоб у тексті абзацу на веб-сторінці забезпечити примусовий перехід на новий рядок, потрібно використовувати непарний тег
 (англ. break — розрив). У місці його розміщення рядок закінчується, а частина тексту, що залишилася, відображується з нового рядка.

Зауважимо, що в наведеному прикладі один парний тег <P> ...</P> абзацу застосовується для задавання написів для текстових полів (те, що знаходиться в ньому між тегом <P> і першим по порядку читання тегом
), для установлення текстових полів і прилеглих до них символів в один ряд (те, що знаходиться між двома тегами
), для відділення стартової кнопки від інших елементів сторінки (те, що знаходиться між другим тегом
 і тегом </P>). Цей прийом дозволяє задати візуально зручний інтерфейс програми — трьома рядками.

У коді прикладу неодноразово зустрічається символьна послідовність   — це вставлення символу «нерозривний пробіл» у текст HTML-документа. Символи нерозривних пробілів використовуются тут для розсунення написів до текстових полів один від одного на потрібну відстань, оскільки застосування з цією метою послідовностей із звичайних пробілів неможливо. Справа в тому, що звичайний пробіл у HTML-документі застосовується як нейтральний розподільник між будь-якими елементами в документі, тому будь-яка послідовність з двох і більше звичайних пробілів сприймається браузерам при візуалізації веб-сторінки завжди як один звичайний пробіл, навіть якщо ця послідовність знаходиться в тексті абзацу.

11 ЗАДАВАННЯ РОЗГАЛУЖЕНИХ ОБЧИСЛЮВАЛЬНИХ ПРОЦЕСІВ У СЦЕНАРІЯХ VBSCRIPT

Для задавання розгалужених обчислювальних процесів у сценаріях VBScript частіше за все застосовують оператор умовного переходу, який ще іноді називають за його ключами оператором If...Then...Else.

У мові VBScript він вживається у двох формах синтаксису: лінійній (рядковій) і блоковій формах.

Формат рядкової форми можна записати так:

If <умова> **Then** <оператори 1> [**Else** <оператори 2>] ,

де <умова> — ім’я змінної, відношення або логічний вираз, тобто все те, що в результаті читання або обчислення може набувати значення істини (True) або хибності (False). Як умову можна використовувати і арифметичний вираз, тоді його будь-який ненульовий результат обчислення буде сприйматися як істина, а нуль — як хибність;

<оператори 1> — один або кілька операторів (тоді вони відокремлюються один від одного двокрапкою), які виконуються, коли <умова> є істинною (True);

<оператори 2> — один оператор або більше (в останньому випадку відокремлюються один від одного двокрапкою), що виконуються, якщо <умова> не була істинною.

Зауважимо, що ця форма оператора умовного переходу обов’язково має бути задана «одним рядком». Звісно, застосування перенесення частини символьної послідовності оператора на наступний рядок коду сценарію при цьому не заборонено. Однак лінійну форму оператора If...Then...Else візуально добре можна сприймати в тексті сценарію тільки тоді, коли оператор дійсно записаний одним рядком. Тому рекомендується застосовувати її лише для простих випадків розгалуження.

Часто лінійна форма оператора умовного переходу застосовується в скороченому вигляді (без ключа Else). Тоді

оператори, розташовані після ключа Then і до кінця рядка, виконуються при істинному значенні умови, а в іншому випадку — починає виконуватися оператор, записаний у наступному рядку сценарію.

Для більш складних розгалужень обчислювального процесу (коли, щоб прийняти рішення про наступне обчислення, перевіряються дві і більше умов) зазвичай використовується блокова форма синтаксису оператора умовного переходу.

Формат блокової форми оператора умовного переходу такий:

```
If <умова 1> Then
[<оператори 1>]
[ ElseIf <умова 2> Then
[<оператори 2>]]
. . .
[ ElseIf <умова  n > Then
[<оператори n >]]
[ Else
[<оператори для випадку else>]]
End If
```

,

де <умова 1> — ім’я логічної змінної, відношення або логічний вираз, які в результаті читання або обчислення можуть набувати значення істини (True) або хибності (False). Ще <умова 1> може бути арифметичним виразом, ненульовий результат обчислення якого вважається істиною, а нульовий — хибністю;

<умова 2>, ..., <умова *n*> — те саме, що і <умова 1>;

<оператори 1>, <оператори 2>, ..., <оператори *n*> — один або більше операторів, які виконуються інтерпретатором, якщо остання з попередніх умов має значення істини (True);

<оператори для випадку *else*> — один або більше операторів, які виконуються інтерпретатором, якщо всі попередні умови не були істинними.

Дії, що виробляються інтерпретатором при виконанні блокової форми оператора If...Then...Else, можна описати в такій послідовності.

Після ідентифікації форми оператора умовного переходу як блокової перевіряється <умова 1>. Якщо вона істинна, то виконуються <оператори 1>, розташовані між рядком, де знаходиться ключ If, і найближчим рядком з ключем ElseIf, а якщо такий рядок відсутній, то рядком з ключем Else. Якщо ж відсутній і такий, то виконуються всі оператори аж до рядка з ключем End If. Якщо <умова 1> не є істинною, то перевіряються по черзі <умова 2>, ..., <умова *n*> , які йдуть за кожним ключем ElseIf. У разі виявлення серед них істинної умови виконуються оператори, розташовані між рядком з цією умовою і наступним рядком з ключем ElseIf, а за відсутності останнього, рядком з ключем Else або ж до кінця блока (рядок End If). Після виявлення однієї істинної умови інші умови, що залишилися (якщо вони є), вже не перевіряються. Якщо ж істинної умови не було знайдено, виконуються оператори, що йдуть за рядком із ключем Else (якщо такий рядок і оператори є).

Розглянемо приклад використання блочного оператора If...Then...Else. Нехай необхідно реалізувати обчислення величини

$$y = \begin{cases} 1, \text{ якщо } -0,5 < x < 2,5, \\ 2, \text{ якщо } \ 2,5 \le x, \\ 3, \text{ в інших випадках}. \end{cases}$$

Нижче наводяться два варіанти фрагмента коду VBScript із застосуванням зазначеного оператора, що розв’язують цю задачу. Імена змінних у фрагментах коду збігаються з іменами відповідних їм величин в умові завдання.

Варіант 1

```
IF ( 1.5 - Abs(x - 1) + Abs( 1.5 _
  - Abs(x - 1))) /2 THEN
  y = 1
ELSEIF x >= 2.5 THEN
  y = 2
ELSE
  y = 3
    END IF
```

Варіант 2

```
 IF x > -0.5 And x < 2.5 THEN
    y = 1
ELSEIF x >= 2.5 THEN
  y = 2
ELSE
  y = 3
END IF
```

У першому варіанті коду умова, що йде за ключем If, задана арифметичним виразом, у другому варіанті — за допомогою логічного виразу. Можливо, що це не зовсім очевидно, але функціонують обидва фрагменти однаково. У цьому легко переконатися, перевіривши їхню роботу на практиці.

Крім оператора умовного переходу, для організації розгалужених обчислень у мові VBScript використовується оператор вибору Select Case (з англ. «вибрати випадок»). Він дозволяє виконувати ті чи інші оператори залежно від множини «дозволених» значень деякого заданого тестового виразу. Формат цього оператора для сценаріїв VBScript можна задати так:

```
Select Case <тестовий вираз>
  [Case <список виразів 1> [ : <оператор 1>]
        [<оператори 1>]]
              . . .
  [Case <список виразів n> [ : <оператор n>]
        [<оператори n>]]
  [Case Else  [ : <оператор для випадку else>]
      [<оператори для випадку else>]]
 End Select
```

,

де <тестовий вираз> — будь-який вираз, значення якого порівнюється зі значеннями виразів із списків, розташованих правіше ключів Case;

<список виразів 1>, ..., <список виразів *n* > — списки з одного або більше виразів (якщо виразів кілька, то вони відокремлюються один від одного комами), значення яких порівнюються зі значенням тестового виразу. У разі виявлення збігу в одному зі списків відповідний ключ Case, праворуч від якого розташований список, вважається обраним;

<оператор 1>, <оператори 1>, ..., <оператор *n* >, <оператори *n*> — один оператор або група операторів, що виконуються інтерпретатором, якщо пов’язаний з ними ключ Case став обраним;

<оператор для випадку *else*>, <оператори для випадку *else*> — один оператор або група операторів, які виконуються інтерпретатором у разі, якщо жоден з ключів Case не став обраним.

Описати процес виконання оператора вибору можна так. Інтерпретатор після виявлення оператора Select Case в тексті сценарію обчислює значення тестового виразу. Далі послідовно від верхнього рядка до нижнього шукаються рядки коду з ключем Case. Знайшовши такий рядок, інтерпретатор обчислює значення кожного виразу з послідовності виразів, яка розташовується правіше ключа Case в тому самому рядку.

Кожне обчислене значення елемента послідовності порівнюється зі значенням тестового виразу. Якщо знайдено збіг, то подальші порівняння вже не проводяться, а інтерпретатор виконує оператор, що стоїть правіше знака двокрапки в тому самому рядку, що й послідовність виразів, після чого переходить до аналізу рядків сценарію, що йдуть нижче рядка з ключем End Select, тобто завершує виконання оператора вибору. Якщо ж у рядку з ключем Case двокрапка відсутня, то інтерпретатор виконує ті оператори, які розташовуються в рядках нижче наступного рядка з ключем Case або ж до рядка End Select. Далі інші рядки з ключами Case (якщо вони є) вже не розглядаються, а інтерпретатор переходить до виконання рядків коду сценарію, що йдуть нижче рядка End Select.

Якщо ж збіг між значенням тестового виразу і значенням будь-якого виразу з послідовності виразів при ключі Case не було знайдено, то інтерпретатор переходить до розгляду наступного рядка з послідовністю виразів при ключі Case, який розташовується нижче. У разі відсутності такого рядка інтерпретатор виконує оператори, розташовані нижче рядків Case Else (якщо вони є), і завершує процес виконання оператора вибору.

Слід зазначити, що в ранніх діалектах Basic оператор вибору був менш «гнучким» і поступався в можливостях блоковій формі оператора умовного переходу. Однак у VBScript, коли допускається в операторі вибору використовувати як тестовий вираз і вирази у списках при ключах Case вирази будь-якого типу, можливості оператора Select Case і блокової форми оператора If...Then...Else стали практично рівними.

12 ОРГАНІЗАЦІЯ ЦИКЛІЧНИХ ОБЧИСЛЕНЬ У СЦЕНАРІЯХ VBSCRIPT

Оператор організації циклу дозволяє повторити виконання деякої кінцевої послідовності інших операторів (цю послідовність зазвичай називають тілом циклу) кілька разів відповідно до заданих умов повтору.

У мові VBScript застосовуються три способи організації циклів: організація циклу за параметром, організація циклу за умовою, організація циклу повного перебору групи (множини).

Організація циклу за параметром завжди пов’язана зі зміною значення деякої числової змінної (однієї змінної, а не кількох), яка кожного разу змінює своє значення від виконання поточної «петлі» тіла циклу до наступної «петлі». Таку змінну зазвичай називають змінною циклу або параметром циклу. Задається початкове значення параметра циклу, значення зміни параметра циклу від одного повтору тіла циклу до іншого і число, за яким можна визначити кількість повторень тіла циклу. У VBS такий оператор лише один, його так і називають — оператор циклу за параметром, або ж за його ключами — оператор For...Next. Цей оператор звичайно добре пристосований для організації арифметичних (регулярних) обчислювальних циклів.

Спосіб організації циклу за умовою полягає в задаванні тіла циклу і деякої умови, яка може виконуватися або не виконуватися в певний момент процесу обчислень. Акт перевірки умови розглядається як компонент тіла циклу. Якщо умова виконується (або не виконується), приймається рішення про виконання іншої частини тіла циклу або про виконання наступного витка циклу. Останнє залежить від того, на початку або в кінці тіла циклу знаходиться дія з перевірки умови. У разі розташування перевірки умови на початку тіла циклу згідно з однією з класифікацій обчислювальних циклів організується цикл з передумовою, а в разі розташування перевірки умови в кінці тіла циклу — цикл з післяумовою. У VBS є два оператори організації циклів за умовою. Їх зазвичай називають у відповідності з їхніми ключами: оператор циклу Do...Loop і оператор циклу While...Wend. Перший з них може організовувати цикли як з передумовою, так і післяумовою, другий — тільки з передумовою. Звичайно вони добре організують

ітераційні циклічні обчислення, але і арифметичні цикли за їх допомогою задаються досить легко.

Організація циклу повного перебору елементів групи забезпечує однакову обробку кожного члена групи. При цьому порядок вибору членів групи для обробки ніяк не визначається, задається тільки акт самої обробки. У VBS організує такий циклічний обчислювальний процес лише один оператор, який, як правило, називають за його ключами оператором циклу For Each...Next. Він обробляє масиви даних і колекції об'єктів.

Оператор For...Next (оператор циклу за параметром) задається в сценаріях VBScript у відповідності з таким форматом:

```
For <парам. циклу> = <поч. знач.> To <кін. знач.> [ Step <крок>]
      [<оператори>]
    [Exit For]
      [<оператори>]
Next                                                              ,
```

де <парам. циклу> — ім'я змінної (параметра) циклу, значення якої показує, наскільки близький поточний циклічний процес до завершення;

<поч. знач.> — початкове значення параметра циклу;

<кін. знач.> — граничне (максимальне або мінімальне) значення, яке дозволено приймати параметру циклу, щоб цикл повторювався;

<крок> — крок зміни параметра циклу; на цю величину автоматично змінюється параметр циклу після кожного виконання тіла циклу; якщо крок не вказано, він дорівнює 1;

<оператори> — оператори, що складають тіло циклу;

Exit For — оператор альтернативного виходу з циклу; зазвичай використовується з перевіркою умови виходу в операторі If...Then...Else; вихід виконується на рядок скрипту, який йде за рядком з ключем Next.

Крок зміни параметра циклу може бути як додатною, так і від'ємною величиною. Якщо крок додатний, то початкове значення параметра циклу має бути менше його граничного значення, при від'ємному кроці — навпаки. Якщо різниця між

граничним і початковим значеннями параметра циклу має знак відмінний від знака кроку, то оператори тіла циклу жодного разу не будуть виконані. Завдання нульового кроку циклу призводить до «зациклення» сценарію, тобто він не може завершитися, циклічне обчислення в ньому буде повторюватися і повторюватися. Причому так триватиме, поки не буде аварійно зупинено роботу IE або ж поки не завершить роботу операційна система. Іноді при ситуаціях зациклення доводиться користуватися послугою засобу Windows Диспетчер Задач (Task Manager).

З попередньої частини курсу відомо, що за допомогою організації арифметичного циклу зазвичай розв'язується одна з задач комп'ютерного дослідження математичних функцій — табуляція функції. Потрібно розрахувати значення деякої аналітично заданої функціональної залежності, наприклад $y = ax^2 + bx + c$ на деякій ділянці зміни її аргументу x для набору рівномірно заданих на цій ділянці значень x. Нижче наводиться текст HTML-документа з відповідним сценарієм VBScript, реалізація яких в IE забезпечує розв'язання цієї задачі.

```
<HTML>
<HEAD>
<TITLE>Табуляція функції за допомогою For-Next</TITLE>
</HEAD>
<BODY ID="Ispolnit">
  <P><INPUT TYPE="Text" NAME="ImyaA"
       ID="paramA" SIZE="7"> параметр A<BR>
  <INPUT TYPE="Text" NAME="ImyaB"
       ID="paramB" SIZE="7"> параметр B<BR>
  <INPUT TYPE="Text" NAME="ImyaC"
       ID="paramC" SIZE="7"> параметр C<BR>
  <INPUT TYPE="Text" NAME="ImyaD"
       ID="paramD" SIZE="7"> поч. знач.<BR>
  <INPUT TYPE="Text" NAME="ImyaE"
       ID="paramE" SIZE="7"> кін. знач.<BR>
  <INPUT TYPE="Text" NAME="ImyaF"
       ID="paramF" SIZE="7"> крок</P>
<SCRIPT LANGUAGE="VBScript">
```

Продовження програми:

```
<!--
SUB Ispolnit_ondblclick
        a = 0 + paramA.Value
        b = 0 + paramB.Value
        c = 0 + paramC.Value
        d = 0 + paramD.Value
        e = 0 + paramE.Value
        f = 0 + paramF.Value
        x1 = "x = "
        y1 = "  y = "
        v = ""
        FOR x = d TO e STEP f
           y = a * x ^ 2 + b * x + c
           v = v & x1 & x & y1 & y & vbLF
        NEXT
        MsgBox v, , "Вікно виведення результату"
END SUB
-->
</SCRIPT>
</BODY></HTML>
```

Пояснимо деякі моменти коду. Оператор v = "" присвоює змінній v «порожній рядок», тобто символьну константу, у якій нема жодного символу. Далі значення змінної v змінюється шляхом з’єднання її вмісту зі значеннями інших символьних змінних і стандартної константи vbLF. Ця константа (vbLF) являє собою код переходу на наступний рядок. Крім того, тут для активації розрахунку застосовується дуже простий спосіб: обробник події ondblclick (на подвійне натискання кнопки вибору «миші»), який, власне, і здійснює всі розрахунки в скрипті, прив’язується до тегу <BODY>. Таким чином, «кнопкою» для запуску розрахунків стає вся область вікна браузера.

У наведеному коді виведення кожної пари значень аргументу і функції (суть табуляції в цьому!) здійснюється шляхом формування символьної змінної v, значення якої виводиться за допомогою функції MsgBox. На рисунку 12.1 показано, як

приблизно виглядає розрахунок задачі табуляції функції у вікні браузера.

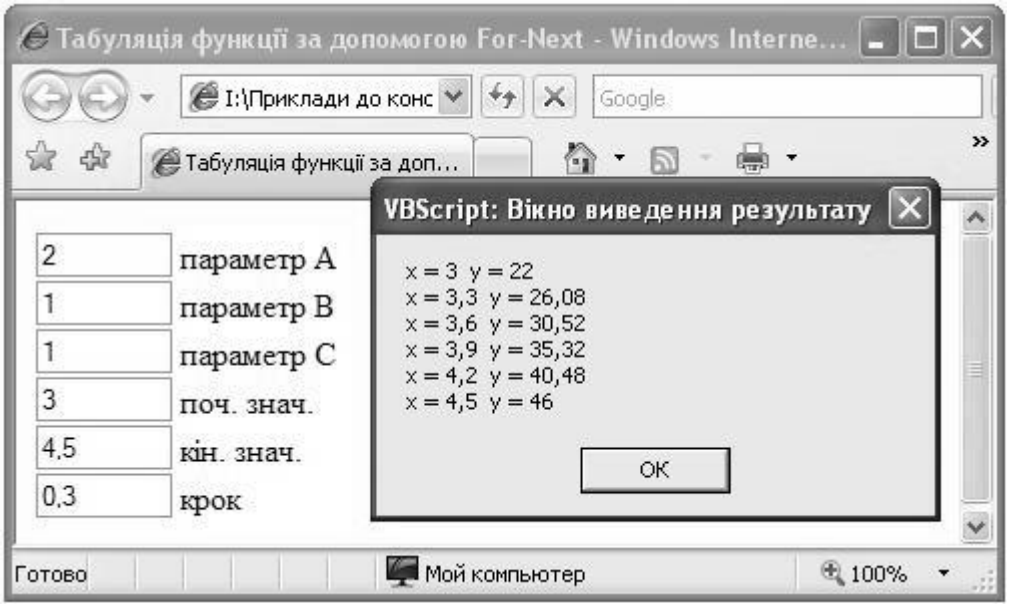

Рисунок 12.1 — Зовнішній вигляд інтерфейсу HTML-додатка, що робить табуляцію квадратних поліномів, і накладеного на нього зверху вікна виведення результату, відкритого за допомогою функції MsgBox

Використовуваний у наведеному прикладі спосіб багаторядкового виведення, як ми знаємо, має недоліки. По-перше, довжина символьного рядка, яку можна вивести у вікні функції MsgBox, досить обмежена. По-друге, відкрите вікно повідомлень блокує роботу браузера з іншими вікнами.

Але є більш прийнятний варіант організації такого виведення. Нижче демонструється приклад коду сценарію, який розв’язує ту саму задачу, що і попередній. Однак у ньому не використовується для виведення даних функція MsgBox.

```
SUB Ispolnit_ondblclick
    a = 0 + paramA.Value
    b = 0 + paramB.Value
    c = 0 + paramC.Value
    d = 0 + paramD.Value
    e = 0 + paramE.Value
    f = 0 + paramF.Value
```

Продовження програми:

```
    x1 = "x = "
    y1 = "  y = "
    v = "<BODY><P>"
    Document.Write _
    "<TITLE> Вікно виведення результату </TITLE>"
    FOR x = d TO e STEP f
       y = a * x ^ 2 + b * x + c
       v = v & x1 & x & y1 & y & "<BR>"
    NEXT
    v = v & "</P></BODY>"
    Document.Write v
END SUB
```

Наведений вище код є новою редакцією процедури Ispolnit для обробки події подвійного натискання на кнопку вибору «миші», що використовувалася в сценарії попереднього прикладу коду. Якщо цією новою редакцією замінити стару редакцію процедури Ispolnit, то оновлений HTML-додаток при візуалізації в IE буде функціонувати інакше. Спочатку у вікні браузера відкриється такий самий інтерфейс з текстовими полями для введення параметрів, як і раніше (рисунок 12.1). Однак після занесення в поля потрібних даних і подвійного натискання на кнопку «миші» вже не спливе вікно повідомлень, а повністю зміниться вміст вікна браузера: на ньому рядок за рядком будуть виведені результати розрахунку (рисунок 12.2).

За необхідності вміст вікна з результатами обчислень можна роздрукувати за допомогою стандартних засобів IE на принтері. Слід зауважити, що розрахунки можна повторити з новими вхідними даними без перезавантаження додатка в IE. Для цього потрібно повернути попередній вміст вікна браузера за допомогою активізації віртуальної кнопки «Назад», розташованої у верхній частині панелі IE, і повторити операції введення даних і пуску розрахунку.

Пояснимо деякі моменти коду нової редакції процедури Ispolnit. Тут ми ширше, ніж звичайно, використовуємо об’єктно-орієнтовану спрямованість застосування засобів мови VBScript. Вище вже йшлося, що при візуалізації веб-сторінки у вікні IE в програмному середовищі браузера створюється об’єкт Document, який є відображенням вмісту всієї сторінки.

Деякими властивостями цього об’єкта ми користуємося для отримання зв’язку між елементами управління з веб-сторінки та обчислювальним процесом сценарію, що виконується.

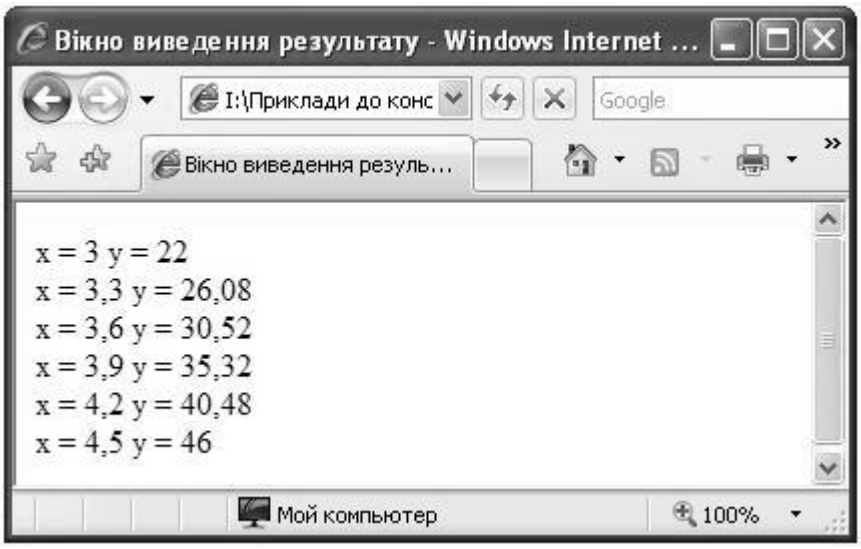

Рисунок 12.2 — Форма виведення результатів розрахунку додатка для табуляції квадратних поліномів, виконаного шляхом формування нової веб-сторінки у вікні IE

Однак в об’єкта Document є ще методи. Один з них — метод Write. Він забезпечує додавання у відкриту для запису веб-сторінку нових компонентів, про які не було зазначено в тегах початкового HTML-документа. Формат виклику цього методу такий:

Document .Write <символьний рядок> ,

де <символьний рядок> — послідовність символів, взята в подвійні лапки, або символьний вираз, значенням якого є послідовність символів, що задає собою запис за допомогою тегів компонентів веб-сторінки, призначених для додавання.

Слід зауважити, що метод Write дозволяє писати символьний рядок тільки у відкриту для додавання компонентів сторінку. Але сторінка закривається для додавання відразу ж після завантаження HTML-документа в обчислювальне середовище браузера та візуалізації веб-сторінки на екрані. Тому з обробників подій будь-які додавання до вже показаної браузером сторінки робити не можна. Однак користуватися методом Write в обробниках подій не

заборонено. У цьому випадку браузер закриває стару веб-сторінку і в тому самому вікні виводить нову.

За допомогою методу Write можна в нову сторінку помістити навіть елементи управління форми, однак не вдасться прив'язати до них новий код сценарію, тоді як колишній код, який був прив'язаний до старої сторінці, у новій сторінці став недоступним.

Приклад ще показує, що для формування веб-сторінки у вікні браузера не обов'язково вводити в документ все установчі теги. Дійсно, для простих випадків, як цей, такі «вільності» допустимі, оскільки сучасні браузери досить «інтелектуальні». Однак у HTML-документа зі складною структурою так робити не слід.

Оператор організації циклу Do...Loop можна назвати найзатребуванішим з групи операторів циклу за умовою, оскільки за його допомогою можна організовувати як циклічні обчислення з передумовою, так і організовувати обчислювальні цикли з післяумовою. Формат цього оператора задається у двох варіантах запису: 1) перевірка умови на початку циклу

```
Do [ While | Until <умова>]
[<оператори>]
[Exit Do]
[<оператори>]
Loop
```
;

2) перевірка умови в кінці циклу

```
Do
[<оператори>]
 [Exit Do]
[<оператори>]
Loop [ While | Until <умова>]
```
,

де While | Until — два ключі, що встановлюють два режими задавання критерію, за яким приймається рішення про продовження виконання тіла циклу: якщо використовується ключ While (з англ. «поки»), то цикл триватиме в разі істинності умови, що за ним слідує, в іншому випадку — відбувається вихід з циклу; якщо використовується ключ

Until (з англ. «доти»), то цикл триватиме в разі хибності умови і відбудеться вихід з циклу в разі істинності останньої. Вибирають завжди тільки один ключ з двох і відповідно тільки один режим або не вибирають ніякого;

<умова> — відношення або логічний вираз, яке має значення істина (True) або хибність (False);

<оператори> — оператори, що складають тіло циклу;

Exit Do — оператор альтернативного виходу з циклу (на наступний рядок скрипту після рядка з ключем Loop); будь-яку кількість цих операторів може бути поміщено всередині циклу. Зазвичай оператор Exit Do використовується спільно з оператором If...Then...Else.

Оператор While...Wend має більш простий синтаксис. Формат його такий:

While <умова>
[<оператори>]
Wend ,

де <умова> і <оператори> позначають те саме, що і для оператора Do...Loop.

При використанні оператора While...Wend виконання операторів тіла циклу повторюється, поки умова, розташована в тому самому рядку, що і ключ While, є істинною (True).

Приклад використання оператора Do...Loop. Розрахунок числа π за допомогою ряду:

$$\frac{\pi}{4} = \sum_{n=0}^{\infty} (-1)^n \frac{1}{2n+1}.$$

Обчислення зупиняється, коли абсолютна величина доданка в ряду не стане менше або рівною величині «точність».

Нижче наводиться текст HTML-документа додатка.

```
<HTML>
<HEAD><TITLE> Розрахунок числа пі </TITLE></HEAD>
<BODY ID="Ispolnit">
  <P><INPUT TYPE="Text" NAME="ImyaA" ID="paramA"
       SIZE="10" VALUE="0,01"> точність</P>
  <P><INPUT TYPE="Text" NAME="ImyaB" ID="paramB"
       SIZE="15"> результат<BR>
  <INPUT TYPE="Text" NAME="ImyaC" ID="paramC"
       SIZE="15"> кількість ітерацій</P>
<SCRIPT LANGUAGE="VBScript"> <!--
SUB Ispolnit_ondblclick
       a = 0 + paramA.Value
       p = 1 : n = 0 : z = 1 : s = 1
       DO
          n = n + 1 : z = -z
          s = 1 / (2 * n + 1)
          p = p + z * s
       LOOP UNTIL s <= a
       ImyaB.Value = 4 * p
       paramC.Value = n
END SUB
--> </SCRIPT>
</BODY></HTML>
```

Результат роботи цього HTML-додатка наводиться на рисунку 12.3. Число π обчислено з точністю до 6 розрядів після коми.

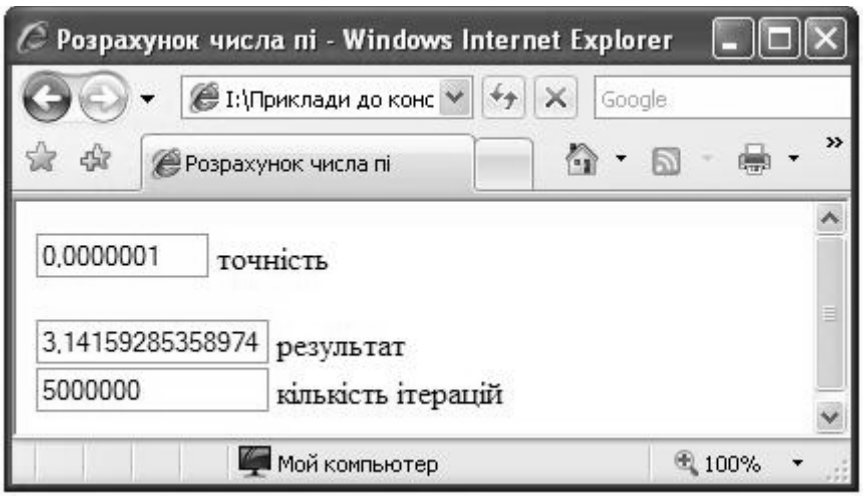

Рисунок 12.3 — Вигляд інтерфейсу HTML-додатка
для обчислення числа π з результатами розрахунків у вікні IE

Приклад використання оператора While...Wend. Обчислення значень арктангенса від x для діапазону $-1 \le x \le 1$ за формулою $\operatorname{arctg} x = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2x+1}}{2x+1}$. Розрахунок суми триває, поки черговий доданок не опиниться менше величини «точність». Перед розрахунком перевіряється, чи належить значення x діапазону збіжності. Якщо введене значення x є поза діапазоном, то його текстове поле очищується та видається відповідне повідомлення. Розрахунок проводиться тільки при задаванні коректного значення для аргументу x. Код цього HTML-додатка наводиться нижче.

```
<HTML><HEAD>
<TITLE> Розрахунок арктангенса від X при |X| ≤ 1</TITLE>
</HEAD><BODY ID="Ispolnit">
  <P><INPUT TYPE="Text" NAME="ImyaA" ID="paramA"
       SIZE="10"> знач. аргументу<BR>
       <INPUT TYPE="Text" NAME="ImyaB" ID="paramB"
       SIZE="10" VALUE="0,01"> точність</P>
  <P> <INPUT TYPE="Text" NAME="ImyaC" ID="paramC"
       SIZE="15"> знач. арктангенса</P>
<SCRIPT LANGUAGE="VBScript"> <!--
SUB Ispolnit_ondblclick
       a = 0 + paramA.Value
       IF a >= -1 And a <=1 THEN
         n = 0 : z = 1 : s = a : f = 0
         b = 0 + paramB.Value
         WHILE Abs(s) >= b
            f = f + s : n = n + 1 : z = -z
            k = 2 * n + 1 : s = z * a ^ k / k
         WEND
         ImyaC.Value = f
       ELSE
         ImyaA.Value = ""
         MsgBox "Значення аргументу поза діапазону.", , _
         "Повідомлення про помилку"
       END IF
END SUB
--> </SCRIPT></BODY></HTML>
```

13 ЗАСТОСУВАННЯ МАСИВІВ У СЦЕНАРІЯХ VBSCRIPT

Мова VBScript підтримує роботу з масивами. З попередньої частини курсу відомо, що масивом називають кілька однорідних (тобто таких, що зберігають дані одного типу) змінних, об'єднаних в одну іменовану групу. Однак можна також інтерпретувати масив як змінну особливої властивості, у якій зберігається більше одного окремого значення. Так чи інакше, але масив схожий на багатоквартирний будинок, де в кожній квартирі є свій мешканець. Так само, як кожна квартира має свій номер (тобто деяке число), окремі значення в масиві ідентифікуються цілими додатними числами, які називаються індексами. Кожен елемент масиву може визначатися одним індексом, двома індексами, трьома і так далі, тоді масив називають відповідно одновимірним, двовимірним, тривимірним і т. д. В одному масиві кожен елемент ідентифікується однаковою кількістю індексів, причому індекси при кожному елементі розглядаються як список. Наприклад, дві пари однакових за значеннями індексів можуть ідентифікувати два різних елементи у двовимірному масиві, якщо однакові за значеннями індекси знаходяться в парах на різних місцях. Синтаксис задавання ідентифікаторів для елементів масивів в мові VBScript такий:

<ім'я масиву> **(**<індекси>**)** **,**

де <ім'я масиву> — ідентифікатор (ім'я) конкретного масиву, до якого входить елемент, що ідентифікується;

<індекси> — список індексів, що уточнює, який саме елемент вказується в масиві. Якщо в списку два або більше індексів, то вони розділяються комами.

Самі індекси можуть задаватися за допомогою конкретних чисел, імен змінних і арифметичних виразів. У разі двох останніх інтерпретатор VBS розглядає як індекси їхні значення. Мінімальне значення кожного індексу в масиві завжди дорівнює нулю. Імена масивів задаються за тими самими правилами, що й імена простих змінних. Приклади звернення до елементів масивів:

```
j = 2  :  k = 1
x( 5) = 0.25 :  y( j ) = x(5) + k : s( k, j + k +1) = 5
y( k - j ) = 1
```

Пояснення до прикладу. Нехай максимальні значення індексів в усіх трьох масивах є 5. Тоді у другому рядку прикладу елемент x (5) отримує значення 0,25; елемент y (2) отримує значення 1,25; а елемент s (1, 4) отримує значення 5. При виконанні третього рядка інтерпретатор видасть повідомлення про помилку: «Індекс виходить за межі допустимого діапазону», оскільки елемент y (-1) не існує.

Звичайну змінну у VBScript можна створити простим присвоюванням їй значення, а перед першим використанням масиву в сценарії потрібно обов'язково повідомити про його існування інтерпретатору VBScript. Масив оголошуються в сценарії так само, як проста змінна, тобто за допомогою оператора Dim, тільки в рядку оператора після імені масиву в круглих дужках має йти список із максимальних значень індексів, використовуваних у цьому масиві. Оголошувати масиви і прості змінні можна разом в одному рядку під одним ключем Dim, відокремлюючи оголошення одне від одного комами.

Приклад задавання масивів:

```
Dim  x (99), y (24, 24), z (99, 99, 99)
```
.

Рядок коду прикладу повідомляє інтерпретатору VBScript, що в сценарії буде використовуватися одновимірний масив з ім'ям «x», що має розмір у 100 елементів, двовимірний масив з ім'ям «y» і розміром у 625 елементів і тривимірний масив з ім'ям «z» і розміром у 1000000 елементів.

Слід зауважити, що дані в масивах, як і дані в простих змінних, після оголошень їхніх носіїв у сценаріях VBS мають загальний тип Variant, тому елементам масивів у мові VBScript можна присвоювати значення різних типів. Такої можливості не мали ранні діалекти Basic.

У наведеному вище прикладі всі три масиви є статичними. Однак у VBScript масив може бути і динамічним (змінюваних розмірів). Щоб ввести в сценарій динамічний масив, потрібно його

обов’язково оголосити в операторі Dim особливим способом: після імені масиву в круглих дужках нічого не вказувати.

Приклад масив з ім’ям dinam оголошений динамічним масивом:

```
Dim  dinam( )
```

.

Для ініціалізації динамічного масиву слід використовувати оператор ReDim. Формат цього оператора такий:

ReDim [**Preserve**] <ім’я масиву> **(**<макс. індекси>**)**[, …] ,

де <ім’я масиву> — ім’я динамічного масиву;

<макс. індекси> — список заново встановлених максимальних значень індексів, що розділяються комами, довжиною відповідно з потрібною розмірністю масиву.

Оператор ReDim застосовується перед першим використанням динамічних масивів для запису даних і в будь-який час після цього, розміри масивів можуть багаторазово переустановлюватися. Ключове слово Preserve використовується, якщо виконується повторна зміна розміру масиву і необхідно зберегти значення елементів масиву, які вже були присвоєні. За відсутності Preserve всі елементи масиву очищуються від значень (зрозуміло, навіть при використанні ключового слова Preserve вкорочення масиву призводить до втрати всієї інформації, що зберігалася в елементах замикання, яких масив позбавляється).

Приклад:

```
Dim  x ( )
ReDim x(10, 10, 10)
   . . .
ReDim Preserve x(10, 10, 15)
```

Масиви часто використовуються, якщо є необхідність обробляти велику кількість даних однаковим чином, оскільки значною перевагою використання масивів у таких ситуаціях є те, що вони дозволяють створювати програми, які є незалежними від кількості елементів, що обробляються. Для цього застосовуються

оператори циклу, що забезпечують послідовний перебір елементів масиву і проведення послідовної обробки кожного елемента.

Раніше нами вже розглядалися оператори циклів, які звичайно можуть організувати такі обчислювальні процеси, але для їхньої роботи потрібно задавати деякі підготовчі дії. Однак у мові VBScript є спеціальний оператор циклу, який дозволяє провести повний перебір всіх елементів масиву і виконати потрібну над кожним обробку без будь-яких підготовчих дій. Зазвичай його називають за ключами — оператор циклу For Each...Next. Формат цього оператора можна задати так:

```
For Each <елемент> In <група>
[<оператори>]
  [ Exit For]
[<оператори>]
Next
```

,

де <елемент> — ім’я змінної, значення якої по черзі зв’язується посиланням зі значеннями всіх елементів масиву або колекції в процесі їх повного перебору;

<група> — ім’я масиву даних або колекції об’єктів;

<оператори> — оператори, що складають тіло циклу;

Exit For — оператор альтернативного виходу з циклу (на наступний рядок сценарію після рядка з ключем Next).

При застосуванні цього оператора організації циклу вміст тіла циклу виконується один раз для кожного елемента групи. Якщо група порожня (у ній нема елементів), то вміст тіла циклу пропускається без виконання.

Нижче наводиться фрагмент коду, який демонструє використання оператора For Each...Next для організації виведення даних з деякого масиву з ім’ям mas у вікно браузера.

```
FOR EACH elem IN mas
    Document.Write elem & "<BR>"
NEXT
```

У такій мовній конструкції звернення до кожного елемента масиву здійснюється через ім’я змінної-посередника.

14 ВВЕДЕННЯ-ВИВЕДЕННЯ ТА ОБРОБКА ДАНИХ В ОДНОВИМІРНИХ МАСИВАХ

Що стосується виведення даних з масиву (як одновимірного, так і двовимірного), то найкраще його робити останньою дією після процесу обчислень і для цього скористатися методом Write об’єкта Document (див. останній приклад з попереднього розділу). Оператор циклу можна використовувати будь-який. Однак у разі вибору оператора For Each...Next треба врахувати, що, як показує практика, одновимірні масиви він обробляє в порядку зростання індексів, а двовимірні масиви — по стовпцях (якщо розглядати масив як матрицю і вважати, що лівий індекс кожного елемента масиву вказує на рядок, де знаходиться елемент, а правий — на стовпець).

Розглянемо введення даних в одновимірний масив. З попередньої частини курсу відомі два способи передачі символьної послідовності в обчислювальне середовище VBScript від клавіатури. Перший спосіб — використовувати стандартну функцію InputBox; другий — застосувати елементи управління форми мови HTML. Обидва способи дозволяють вирішити поставлене завдання.

Покажемо це на простому прикладі. Нехай є вхідний числовий одновимірний масив з n елементів, який заповнюється довільно числами. Потрібно знайти в масиві всі числа, що не менше деякого числа a й не більше деякого числа b, вивести їх і підрахувати кількість.

Для розв’язання задачі застосуємо простий поетапний алгоритм: 1) вводимо дані і формуємо масив; 2) аналізуємо вхідні дані, отримуємо вихідні дані та готуємо їх для виведення; 3) виводимо вихідні дані.

Використання для введення даних стандартної функції InputBox має відомі недоліки. Однак застосування її дозволяє скласти код сценарію за класичною схемою алгоритму введення даних у масив: задається розмір масиву, потім організується

арифметичний цикл, тілом якого є операція введення даних у конкретний елемент масиву.

Текст фрагмента сценарію, який реалізує цей етап, наводиться нижче.

```
DIM mas( )
n = InputBox("Введіть розмір вхідного масиву:", _
        "Вікно введення даних")
 REDIM mas(n)
 FOR i = 0 TO n - 1
     mas(i) = 0 + InputBox("Введіть значення для " & i & _
                 "-го елемента масиву:", "Вікно введення даних")
 NEXT
```

Для початку обробки значень елементів масиву треба ввести ще значення параметрів *a* та *b*. Це можна зробити, знову двічі застосувавши функцію InputBox. Нижче наводиться фрагмент коду, у якому задається названа дія разом з простою перевіркою коректності значень параметрів, що вводяться.

```
 DO
       a = 0 + InputBox("Введіть значення для параметра A:", _
            "Вікно введення даних")
        b = 0 + InputBox("Введіть значення для параметра B:", _
            "Вікно введення даних")
       IF a > b THEN MsgBox "Помилка!" & vbLF &  _
       "Значення для A більше, ніж значення для B." _
       & vbLF & "Повторіть введення значень цих параметрів!", _
       , "Важливе повідомлення"
 LOOP WHILE a > b
```

Другий етап алгоритму, тобто аналіз даних у масиві, виявлення шуканих значень, підрахунок їх кількості та підготовка до виведення, можна реалізувати нижченаведеними фрагментом коду.

```
k = 0 : v = ""
FOR i = 0 TO n - 1
   IF mas(i) >= a And mas(i) <= b THEN
      v = v & mas(i) & "   " : k = k + 1
   END IF
     NEXT
```

У ньому для пошуку потрібних значень у масиві організується арифметичний цикл. За допомогою оператора умовного переходу, укладеного в тіло циклу, проводиться виявлення необхідних значень і «спрацьовування» лічильника (змінної з ім'ям «k»). Підготовка даних до виведення зводиться до формування символьного рядка (змінна з ім'ям «v»), до якої поступово приєднуються символьні представлення виявлених значень, відділяючись одне від одного трьома символами «нерозривний пробіл».

Третій етап алгоритму — виведення вихідних даних, виконують два звернення до методу Write об'єкта Document.

```
Document.Write "Кількість шуканих значень в масиві = " _
    & k & ".<BR><BR>"
Document.Write v
```

Якщо послідовно з'єднати всі чотири наведених фрагменти в один сценарій і вставити його відомим чином у порожню форму HTML-документа, яка була розроблена нами раніше, то отримаємо потрібний HTML-додаток, що розв'язує задачу прикладу.

Тепер розглянемо другий спосіб введення даних в одновимірний масив з використанням елементів управління мови HTML. Нижче наводиться майже повністю текст HTML-додатка для розв'язання тієї самої задачі прикладу. До нього увійшли повністю два останніх з чотирьох вищенаведених фрагментів коду, і тому з метою скорочення розмірів загального запису HTML-документа їхній зміст у ньому замінено прямокутниками з відповідними короткими написами.

```
<HTML><HEAD>
<TITLE>Обробка даних в одновимірному
      масиві</TITLE></HEAD>
<BODY>
    <P><INPUT TYPE="Text" NAME="dlina"
        ID="dlina" SIZE="6">
        : довжина масиву<BR>
        <INPUT TYPE="Button" NAME="knopka1"
        ID="knopka1" VALUE="Ввести довжину!"></P>
    <P><INPUT TYPE="Text" NAME="element"
        ID="element" SIZE="25">
        <INPUT TYPE="Text" NAME="znachen"
        ID="znachen" SIZE="7"><BR>
        <INPUT TYPE="Button" NAME="knopka2"
        ID="knopka2" VALUE="Ввести значення!"></P>
    <P><INPUT TYPE="Text" NAME="param_a"
        ID="param_a" SIZE="6" VALUE="0"> : параметр A<BR>
        <INPUT TYPE="Text" NAME="param_b"
        ID="param_b" SIZE="6" VALUE="0"> : параметр B<BR>
        <INPUT TYPE="Button" NAME="knopka3"
        ID="knopka3" VALUE="Виконати розрахунок!">
 <SCRIPT LANGUAGE="VBScript"><!--
    DIM mas( ), n, m
    s1 = "Значення " : s2 = "-го елемента:"


    SUB knopka1_onclick
        n = 0 + dlina.Value
        REDIM mas(n - 1)
        m = 1
        element.Value = s1 & m & s2
    END SUB
```

Продовження програми:

```
SUB knopka2_onclick
        mas(m - 1) = 0 + znachen.Value
        znachen.Value = ""
        m = m + 1
       IF m > n THEN
           element.Value = "Усі значення введені!"
           znachen.Value = ""
        ELSE
           element.Value = s1 & m & s2
        END IF
    END SUB
    SUB knopka3_onclick
        a = 0 + param_a.Value
        b = 0 + param_b.Value
        IF a < b THEN
```

Аналіз вхідних даних і формування вихідного рядка символів

Демонстрація вихідного рядка символів у вікні IE

```
        ELSE
           param_a.Value = "Помилка!"
           param_b.Value = "Помилка!"
        END IF
    END SUB
--></SCRIPT></BODY></HTML>
```

У цьому коді нема викликів функцій MsgBox та InputBox, що зупиняють обчислювальний процес в очікуванні відповіді від людини-користувача. Код сценарію завантажується, і його головна процедура відразу виконується після завантаження, але вона являє собою буквально два рядки, у яких визначаються глобальні масиви і змінні. Всі інші дії сценарію розподілені між трьома допоміжними процедурами — обробниками подій.

Таку структуру формування сценарію слід вважати досить зручною при організації обчислювального процесу, коли є введення одного або декількох масивів початкових даних, подальша їх обробка й виведення шуканих результатів. Розглянемо з цих позицій вищенаведений код.

Головна процедура сценарію прикладу має невеликі розміри, але це дуже важлива частина всього завдання обчислень, оскільки оголошення глобальних величин створює єдину загальну область даних для спільного використання всіма процедурами сценарію. Втім, установчі дії головної процедури сценарію не були б достатніми без відкриття веб-сторінки з набором елементів управління, до яких мають «прив'язуватися» обробники подій, тому ініціалізацією обчислювального процесу слід вважати не тільки визначення загальної області даних головною процедурою, але і формування необхідних елементів управління веб-сторінки у вікні браузера.

Далі послідовно застосовуються три процедури обробників подій, пов'язані з «натисканнями» віртуальних кнопок, тобто подій, ініційованих безпосередньо людиною-користувачем. Перша процедура («прив'язана» до кнопки «knopka1») приймає від текстового поля, сформованого у вікні браузера, ціле число, яке встановлює розмір вхідного масиву, і відповідним чином перетворює організацію відкритого глобального масиву даних. Наступною дією вона повідомляє користувача, що час ввести дані в перший елемент масиву. Це здійснюється за допомогою подачі потрібного повідомлення в спеціальне текстове поле.

Отримавши це повідомлення, користувач вносить числові дані в інше спеціально відведене для цього текстове поле і «натискає» на віртуальну кнопку «knopka2», чим активізує другу за порядком опису в скрипті процедуру обробки події. Ця процедура виконує три дії: 1) вводить потрібне число в черговий елемент масиву даних, що заповнюється; 2) підраховує за допомогою спеціальної змінної-лічильника кількість заповнених даними елементів масиву; 3) видає повідомлення користувачу про необхідність введення нового значення для наступного «порожнього» елемента масиву, якщо такий є, а в іншому випадку повідомляє користувачеві, що масив вже заповнений і пора починати його обробляти. Таким чином, другий обробник подій буде активізований користувачем стільки разів, скільки елементів має вхідний масив.

Третя процедура обробки події onclick, що «пов'язана» з кнопкою «knopka3», власне, і є тією обробкою значень масиву та виведенням результатів обробки, заради яких організовувався

процес введення даних до масиву. Ця процедура активізується користувачем після того, як він отримає повідомлення від попередньої процедури про те, що вхідний масив вже повністю заповнений даними.

15 РОБОТА З ДВОВИМІРНИМИ МАСИВАМИ ДАНИХ У СЦЕНАРІЯХ VBSCRIPT

За своїми підходами обробка даних у двовимірних масивах нічим не відрізняється від аналогічних операцій з даними в одновимірних масивах. Треба тільки пам’ятати, що кожен елемент двовимірного масиву визначається при зверненнях до нього значенням не одного індексу (якщо розглядати індекс як числову величину), а значеннями двох індексів. Тому якщо деяка операція з обробки одновимірного масиву потребує організації простого циклу для перебору значень індексів елементів масиву, то подібна операція з двовимірним масивом даних потребує організації подвійного циклу (цикл, що забезпечує перебір значень одного індексу, вкладається в тіло циклу, що забезпечує перебір значень іншого індексу). Що стосується розбіжностей у загальних алгоритмах обробки одновимірних і двовимірних масивів даних, то їх не існує.

Якщо не використовувати обробники подій і елементи управління HTML, то будь-який обчислювальний процес, пов’язаний зі зверненнями до елементів двовимірного масиву, точно відповідає «класичним» схемами їхніх алгоритмів, які вивчалися в попередній частині курсу. На підтвердження цього розглянемо код розв’язання такої задачі.

Дано числову матрицю, що має n рядків і m стовпців. Потрібно знайти в ній максимальне і мінімальне числа, а також вказати індекси елементів матриці, значеннями яких ці числа є (передбачається, що матриця має тільки по одному елементу з максимальним і мінімальним значеннями).

Звичайно як матриця в сценарії досліджується відповідний двовимірний масив, який треба попередньо заповнити за допомогою введення з клавіатури числами. Нижче наводиться текст цього сценарію. Щоб його виконати, достатньо ввести код сценарію відомим способом в HTML-документ і «відкрити» останній браузером IE.

```
DIM mas( )
n = 0 + InputBox( _
       "Введіть кількість рядків у вхідному масиві:", _
       "Вікно введення даних")
m = 0 + InputBox( _
       "Введіть кількість стовпців у вхідному масиві:", _
       "Вікно введення даних")
REDIM mas(n - 1 , m -1)
 FOR i = 0 TO n - 1
   FOR j = 0 TO m - 1
     mas(i, j) = 0 + InputBox( _
         "Введіть значення для елемента масиву," _
         & vbLF & "який розташований на " & (i + 1) & _
         "-му рядку і на " & (j + 1) & "-му стовпці:", _
         "Вікно введення даних")
   NEXT
NEXT
ma = mas(0, 0) : mi = mas(0, 0)
ima = 0 : jma = 0 : imi = 0 : jmi = 0
FOR i = 0 TO n - 1
   FOR j = 0 TO m - 1
     IF mas(i, j) >= ma THEN
       ma = mas(i, j) : ima = i : jma = j
     ELSEIF mas(i, j) <= mi THEN
       mi = mas(i, j) : imi = i : jmi = j
     END IF
   NEXT
NEXT
MsgBox "Максимальний елемент дорівнює " & ma & vbLF _
& " і він знаходиться на " & (ima +1) & "-му рядку й " & _
(jma + 1) & "-му стовпці." & vbLF _
& "Мінімальний елемент дорівнює " & mi & vbLF _
& " і він знаходиться на " & (imi +1) & "-му рядку й " & _
(jmi + 1) & "-му стовпці.", , _
"Вікно виведення результату"
```

У коді використовується динамічний двовимірний масив. Однак його можна задати і статичним, вказавши в операторі Dim конкретні максимально допустимі для розмірів масиву числа.

Якщо задавати введення і виведення даних у двовимірних масивах без застосування стандартних функцій InputBox і MsgBox (тобто шляхом організації обробників подій і використовуючи елементи управління HTML), то модифікацію коду таких сценаріїв слід проводити так само, як показано в попередньому розділі при організації цих дій з одновимірними масивами. Обов'язково при цьому мають враховуватися операції з другим індексом для кожного звернення до елементів таких масивів.

Нижче наводиться фрагмент HTML-коду, який забезпечує задавання потрібного інтерфейсу користувача.

```
<P><INPUT TYPE="Text" NAME="dlstr"
   ID="dlstr" SIZE="6">
   : довжина масиву в рядках<BR>
   <INPUT TYPE="Text" NAME="dlstb"
   ID="dlstb" SIZE="6">
   : довжина масиву в стовпцях<BR>
   <INPUT TYPE="Button" NAME="knopka1"
   ID="knopka1" VALUE="Ввести розміри!"></P>
<P><INPUT TYPE="Text" NAME="element"
   ID="element" SIZE="54">
   <INPUT TYPE="Text" NAME="znachen"
   ID="znachen" SIZE="7"><BR>
   <INPUT TYPE="Button" NAME="knopka2"
   ID="knopka2" VALUE="Ввести значення!"></P>
```

Цей фрагмент коду має міститися всередині парного тегу <BODY> в HTML-документі. Після відображення цього документа браузером користувач побачить у вікні останнього два текстових поля (їхні ідентифікатори — dlstr та dlstb), у які йому слід ввести розміри вхідного масиву в рядках і стовпцях. Нижче текстових полів знаходитиметься кнопка з написом «Ввести розміри!» (ідентифікатор кнопки — «knopka1»), яку слід активізувати після того, як у згаданих текстових полях будуть вказані потрібні розміри. Крім названих компонентів інтерфейсу, нижче у вікні браузера будуть перебувати ще два текстових поля і кнопка з написом «Ввести значення!». Через одне з цих текстових полів (його ідентифікатор — «element») інтерпретатором VBScript

будуть виводитися вказівки для користувача, що тому треба вводити в друге текстове поле (ідентифікатор — «znachen») перед тим, як «натиснути» кнопку «Ввести значення!» (ідентифікатор — «knopka2»). Зрозуміло, що текстове поле «znachen» використовується для введення символьних даних за допомогою клавіатури в елементи двовимірного масиву, а в полі «element» використовується виключно для виведення коментарів до цього вводу.

Для реалізації вищеописаних дій задається сценарій VBScript, що складається з головної процедури і двох обробників події onclick. У головній процедурі оголошується як глобальний динамічний масив mas, який далі буде перетворений у двовимірний масив. Також оголошуються глобальними кілька змінних величин, деякі з них отримують початкові значення. За допомогою цих величин будуть передаватися дані до обробників подій. Відповідний фрагмент коду головної процедури сценарію наводиться нижче.

```
DIM mas( ), n, m, str, stb
 s1 = "Значення елемента з "
 s2 = "-го рядку й " : s3 = "-го стовпця:"
```

Два обробники події, які, власне, і забезпечують весь процес введення даних у двовимірний масив, за своїми діями повністю збігаються з однойменними процедурами обробки подій попереднього розділу, де розглядалося введення даних в одновимірний масив. Відмінності, які все ж таки є в їхньому коді, обумовлені виключно додаванням дій, пов’язаних з обробкою значень другого індексу при зверненнях до елементів вхідного масиву.

Тому без додаткових пояснень просто наведемо далі код цих процедур.

```
SUB knopka1_onclick
   n = 0 + dlstr.Value : m = 0 + dlstb.Value
   REDIM mas(n - 1, m - 1)
   str = 1 : stb = 1
   element.Value = s1 & str & s2 & stb & s3
END SUB
```

```
SUB knopka2_onclick
    mas(str - 1, stb - 1) = 0 + znachen.Value
    znachen.Value = ""
    IF stb = m THEN
       stb = 1
       IF str = n  THEN
         element.Value = "Усі значення введені!"
         znachen.Value = ""
       ELSE
         str = str + 1
         element.Value = s1 & str & s2 & stb & s3
       END IF
    ELSE
      stb = stb + 1
      element.Value = s1 & str & s2 & stb & s3
    END IF
END SUB
```

Оскільки двовимірні масиви часто використовуються для обробки числових матриць, тобто прямокутних числових таблиць, іноді дуже важливо, щоб виведення значень таких масивів у вікні браузера виглядало би як таблиця, де числа шикуються в рівні стовпці і рядки. Розв’язати таку задачу тільки засобами VBScript було б важко, але все спрощується, якщо для цього залучити засоби HTML.

Таблиця в HTML задається за допомогою парного тегу <TABLE>. Між його початковим і кінцевим тегами поміщуються рядки таблиці, що задаються за допомогою парних тегів <TR>. Кожен рядок таблиці містить елементи таблиці, які можуть належати до двох різних типів. Комірки для заголовків стовпців і рядків задають парним тегом <TH>, а звичайні комірки — парним тегом <TD>. Кожна комірка може містити довільний текст, а також будь-які теги HTML, допустимі в «тілі» документа. Зокрема комірка таблиці може містити вкладену таблицю або зображення. При відображенні таблиці у вікні браузера відбувається її автоматичне форматування з підбором розмірів комірок відповідно до обсягу розміщуваних там даних і заданих атрибутів.

Зауважимо, що в нашому випадку, коли теги таблиці будуть використовуватися для відображення числової матриці, для задавання елементів таблиці зручніше використовувати парний тег <TH>, оскільки за замовчуванням браузер його вміст виводить напівжирним шрифтом і центрує.

Нижче наводиться фрагмент сценарію VBScript, який забезпечує виведення вмісту числового масиву mas, що має *n* рядків і *m* стовпців, у вигляді числової матриці на окремо відкриту веб-сторінку у вікні браузера.

```
Document.Write "<TABLE>"
FOR i = 0 TO n - 1
    v = "<TR>"
    FOR j = 0 TO m - 1
        v = v & "<TH>" _
            & " " & mas(i, j) & " " & "</TH>"
    NEXT
    v = v & "</TR>"
    Document.Write v
NEXT
Document.Write "</TABLE>"
```

Слід сказати, що мова HTML має багато різних засобів подання та оформлення текстових даних. Зокрема існує допоміжна для мов розмітки (тобто і для мови HTML теж) формальна мова CSS (Cascading Style Sheets — каскадні таблиці стилів), що служить для опису зовнішнього вигляду електронних документів і яку добре «розуміють» браузери.

Продемонструємо одну з можливостей CSS: додамо в кожен тег <TABLE> і <TH> із наведеного вище фрагмента коду VBScript як атрибут зі значенням підрядок

```
STYLE = 'border: thin dotted black'
```

і перевіримо, як IE відобразить перетворений таким способом код. Ми побачимо, що браузер відкриє нове вікно і виведе в нього матрицю, яка сама і кожна в ній комірка будуть обрамлені лініями з дрібних точок.

Вивчення засобів CSS виходить за межі цього курсу, однак бажаючі можуть ознайомитися з їхніми основами з відповідної літератури, поданої в загальному списку літератури.

СПИСОК ЛІТЕРАТУРИ

1 MS Project Standart [CDR] 2002/ Pus. Disc 1885 November. 2002.

2 MS Windows 98 [CDR] 2001/ Pus. Disc 0101 January. 2001.

3 MS Visual Studio 6 PRO [CDR] 2001/ Eng Disc 0934 July. 2001.

4 Бізюк А. В., Бізюк І. Г. Створення Web-сторінок : конспект лекцій. Харків : УкрДАЗТ, 2008. 67 с.

5 Буров Є. В. Компютерні мережі : підручник. Львів : «Магнолія 2006», 2010. 260 с.

6 Глушаков С. В., Сурядный А. С. Программирование на Visual Basic 6.0. : конспект лекцій. Харьков : Фолио, 2002. 214 с.

7 Основи алгоритмізації базових обчислювальних процесів : навч. посіб. / В. С. Меркулов [та ін.]. Харків : УкрДАЗТ, 2008. 147 с.

8 Дибкова Л. М. Інформатика і комп’ютерна техніка : навч. посіб. Вид. 2-ге, перероб. Київ : Академвидав, 2007. 415 с.

9 Інженерна та комп’ютерна графіка : підручник / за ред. В. Є. Михайленка. Вид. 3-тє, перероб. і допов. Київ : Видавничий дім «Слово», 2011. 352 с.

10 Меркулов В. С., Бізюк І. Г., Чаленко О. В. Програмування в середовищі VISUAL BASIC 6.0 : конспект лекцій. Харків : УкрДАЗТ, 2014. Ч. 1. 65 с.

11 Меркулов В. С., Бізюк І. Г., Чаленко О. В. Програмування в середовищі VISUAL BASIC 6.0 : конспект лекцій. Харків : УкрДАЗТ, 2014. Ч. 2. 67 с.

12 Меркулов В. С., Бізюк І. Г., Завгородня Н. М., Чаленко О. В. Програмування в середовищі VISUAL BASIC 6.0 : конспект лекцій. Харків : УкрДАЗТ, 2014. Ч. 3. 57 с.

13 Форкун Ю. В., Длугунович Н. А. Інформатика : навч. посіб. Львів : Новий Світ, 2012. 463 с.

14 Шеховцов В. А. Операційні системи : підручник. Київ : Видавнича група BHV, 2005. 576 с.

ОСНОВИ ПРОГРАМУВАННЯ
ІНЖЕНЕРНО-ТЕХНІЧНИХ РОЗРАХУНКІВ
АЛГОРИТМІЧНОЮ МОВОЮ ВИСОКОГО РІВНЯ:
ВВЕДЕННЯ ДО VBSCRIPT

Конспект лекцій

з дисциплін

«ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА ПРОГРАМУВАННЯ»,
«ІНФОРМАТИКА»

Відповідальний за випуск Бізюк І. Г.

Редактор Ібрагімова Н. В.

Підписано до друку 03.04.19 р.

Формат паперу 60х84 1/16. Папір писальний.

Умовн.-друк.арк. 4,0. Тираж 50. Замовлення №

Видавець та виготовлювач Український державний університет
залізничного транспорту,
61050, Харків-50, майдан Фейєрбаха, 7.
Свідоцтво суб'єкта видавничої справи ДК № 6100 від 21.03.2018 р.