УДК 004.9:378.147
DOI https://doi.org/10.24144/2616-7700.2025.46(1).247-255

М. М. Повідайчик[1], М. М. Шаркаді[2], Р. А. Кацала[3], Р. М. Годя[4], І. В. Янчій[5]

[1] ДВНЗ «Ужгородський національний університет»,
професор кафедри кібернетики і прикладної математики,
доктор педагогічних наук
mykhailo.povidaichyk@uzhnu.edu.ua
ORCID: https://orcid.org/0000-0003-1554-2067

[2] ДВНЗ «Ужгородський національний університет»,
доцент кафедри кібернетики і прикладної математики,
кандидат економічних наук
marianna.sharkadi@uzhnu.edu.ua
ORCID: https://orcid.org/0000-0002-1850-996X

[3] ДВНЗ «Ужгородський національний університет»,
доцент кафедри кібернетики і прикладної математики,
кандидат фізико-математичних наук
roman.katsala@uzhnu.edu.ua
ORCID: https://orcid.org/0009-0000-4766-5235

[4] ДВНЗ «Ужгородський національний університет»,
аспірант 2-го року навчання
roman.hodia@uzhnu.edu.ua
ORCID: https://orcid.org/0009-0008-7546-5263

[5] ДВНЗ «Ужгородський національний університет»,
студент 4-го курсу
yanchii.ivan@student.uzhnu.edu.ua
ORCID: https://orcid.org/0009-0003-2280-7219

РОЗРОБЛЕННЯ ОБОЛОНКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ «КОМП'ЮТЕРНИЙ ТРЕНАЖЕР З ФІНАНСОВОЇ МАТЕМАТИКИ»

Описано розроблену оболонку інформаційної системи «Комп'ютерний тренажер з фінансової математики». Особливістю розробленої системи є орієнтація на бази знань, а не на бази даних, що дозволяє генерувати значну кількість рівнозначних тестових завдань. Відмітимо можливість відносно легко розширювати систему новими типами завдань з фінансової математики, а також переорієнтовувати оболонку для інших розділів математики чи інформатики. Розроблена система може використовуватися як для контролю знань, так і як інструмент для самонавчання.

Ключові слова: комп'ютерний тренажер, бази знань, фінансова математика, VBA.

1. Вступ. Однією із задач, яку вирішує викладач ЗВО, є об'єктивна перевірка знань здобувачів. Незважаючи на те, що за допомогою комп'ютерного тестування не можна перевірити комплексний рівень знань, тим не менше саме за допомогою тестів можна швидко здійснити його оцінку. Так, на сьогодні комп'ютерне тестування використовується:

– при проведенні поточного оцінювання знань здобувачів вищої освіти;
– при написанні модульних та залікових контрольних робіт;
– при складанні вступних іспитів до магістратури та ін.

Слід зазначити, що при цьому від викладача вимагається розроблення набору тестових завдань у різних форматах:

– із відкритою формою відповіді, де результатом розв'язання є число;
– із закритою формою відповіді, коли у тесті поряд із 1 правильною приводяться декілька неправильних відповідей;
– тестові завдання на встановлення відповідності між декількома запитаннями та переліком можливих відповідей.

Все це вимагає від викладача значних затрат на розроблення завдань, вивірення правильності відповідей та ін. Тому актуальною є задача розроблення оболонки комп'ютерної системи генерації тестових завдань, яка буде вирішувати декілька завдань [1; 2]:

– слугувати інструментом для викладача при формуванні заданої кількості варіантів контрольної роботи;
– формувати файли заданого шаблону, наприклад, система Moodle дозволяє імпортувати тестові завдання, записані у спеціальних форматах AIKEN, GIFT та ін.;
– використовуватися у якості комп'ютерного тренажера для самопідготовки здобувачів вищої освіти.

У даній статті описується розроблена оболонка інформаційної системи «Комп'ютерний тренажер з фінансової математики». Особливостями розробленої оболонки є:

– відсутність баз даних, орієнтація на бази знань, що дозволяє генерувати значну кількість рівнозначних тестових завдань;
– можливість відносно легко розширювати систему новими типами завдань з фінансової математики, а також переорієнтовувати оболонку для інших розділів математики чи інформатики;
– у більшості випадків сформовані файли можуть безпосередньо використовуватися при тестуванні, але при потребі їх можна доопрацьовувати у текстових редакторах.

2. Аналіз останніх досліджень і публікацій. Зазначені проблеми досліджують як вітчизняні, так і зарубіжні науковці. Так, задачу розроблення комп'ютерних систем тестування вивчають вчені З. Бондаренко, О. Радкевич, О. Музика, О. Шинкаренко, О. Басалкевич; розроблення комп'ютерних тренажерів описують А. Шурдук, В. Мандрика, Ю. Олексійчук, О. Ємець та ін. Проте, недостатньо дослідженою залишається проблема розроблення комп'ютерної оболонки, орієнтованої на роботу з базами знань.

3. Постановка задачі. Розробити оболонку інформаційної системи «Комп'ютерний тренажер з фінансової математики» на основі шаблонів MS Word з використанням середовища Visual Basic for Applications.

4. Розроблення оболонки інформаційної системи. При розробленні оболонки інформаційної системи «Комп'ютерний тренажер з фінансової математики» за основу було обрано шаблони MS Word, оскільки вони дозволяють формувати документи заданої структури, а вбудована мова програмування Visual Basic for Applications дозволяє ефективно керувати всіма об'єктами MS Word.

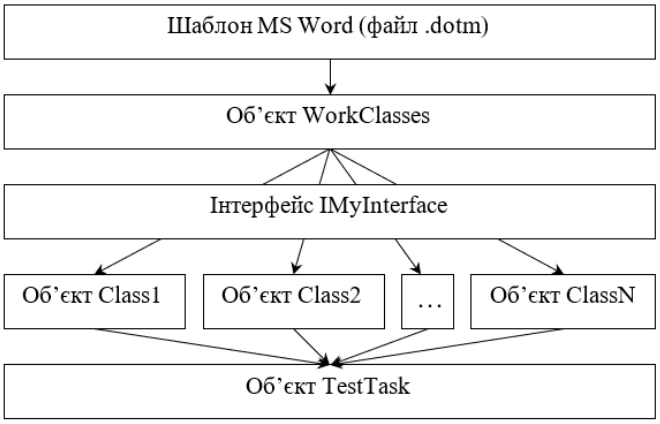

Рис. 1. Загальна структура системи.

Розглянемо загальну структуру системи (рис. 1).

Опишемо спрощену систему, яка буде демонструвати роботу основних модулів:

– запуск програми і налагодження діалогу з користувачем;
– створення об'єкта «WorkClasses», який на основі інтерфейсу «IMyInterface» надає доступ до класів, у яких описані завдання;
– створення колекції об'єктів Class1, Class2, на основі якої створюються варіанти завдань та таблиця з відповідями.

Отже, роботу системи розпочнемо із шаблону MS Word (файл «Фінансова математика.dotm»). У цьому файлі будуть описані демо варіанти об'єктів, приведених на рис. 1. Для створення нового документу на основі шаблону використовується обробник за домовленістю Document_New(). У цій процедурі отримаємо від користувача дані про кількість варіантів та тип завдання. У результаті виконання процедури буде створено новий файл з варіантами завдань та таблицею з відповідями до них.

Лістинг 1. Процедура Document_New().

```
Private Sub Document_New()
  Dim numTaskVariants As Integer
  Dim typeQuestion As Integer
  Dim txt As String ' Текст завдань
  Dim ans() As String ' Таблиця з відповідями
  Dim tbl As Table
  Dim rng As Range
  Dim i As Integer
  Dim j As Integer
  numTaskVariants = InputBox("Введіть кількість варіантів:")
  typeQuestion = InputBox("Введіть тип завдань (1-відкрита форма, 2-закрита
   форма, 3-встановлення відповідності):")
  Dim ob As New WorkClasses
  ob.Constructor numTaskVariants, typeQuestion
  txt = ob.Text
  ActiveDocument.Range.Text = txt
  ans = ob.Table
  ' Вставити таблицю в кінець документа
  Set rng = ActiveDocument.Content
```

```
  rng.Collapse Direction:=wdCollapseEnd
  ' Створити таблицю
  Set tbl = ActiveDocument.Tables.Add(Range:=rng, NumRows:=UBound(
   ans, 1), NumColumns:=UBound(ans, 2))
  ' Заповнити таблицю даними з масиву
  For i = 1 To UBound(ans, 1)
    For j = 1 To UBound(ans, 2)
      tbl.Cell(i, j).Range.Text = ans(i, j)
    Next j
  Next i
End Sub
```

На рис. 2 приведено створений файл «Документ1.docx» на основі шаблону «Фінансова математика.dotm», при заданих значеннях змінних

– numTaskVariants = 3 (3 варіанти);
– typeQuestion = 1 (завдання з відкритою формою).

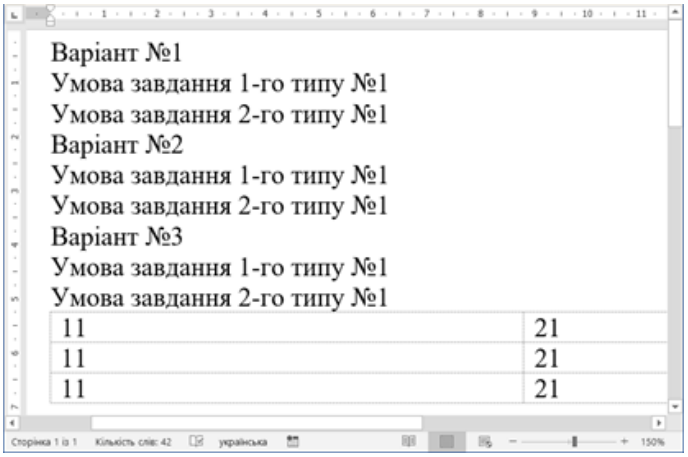

Рис. 2. Загальний вигляд створеного документу.

Фрагмент процедури Document_New() (див. лістинг 1)

```
Dim ob As New WorkClasses
ob.Constructor numTaskVariants, typeQuestion
```

створює об'єкт класу WorkClasses, який керує іншими об'єктами системи. Так, конструктор класу циклічно звертається до об'єктів середнього рівня на рис. 1 і, в залежності від вхідних параметрів, формує вказану кількість варіантів завдань вказаного типу.

Лістинг 2. Конструктор класу WorkClasses.

```
Public Sub Constructor(ByVal numVariants As Integer, ByVal
    typeQuestion As Integer)
  ' Колекція завдань, які утворюють варіант
  Dim colTasksVariant As Collection
  ' Колекція однотипних завдань
  Dim colTestTasks As Collection
  Dim obj1 As IMyInterface
  Dim obj2 As IMyInterface
  Dim item As IMyInterface
  Dim i As Integer
```

```
  Dim j As Integer
  pNumVariants = numVariants
  pTypeQuestion = typeQuestion
  pText = ""
  ReDim pTable(1 To pNumVariants, 1 To m) As String
  For i = 1 To pNumVariants
    pText = pText & "Варіант №" & CStr(i) & vbCrLf
    Set colTasksVariant = New Collection
    Set obj1 = New Class1
    Set obj2 = New Class2
    colTasksVariant.Add obj1
    colTasksVariant.Add obj2
    j = 0
    For Each item In colTasksVariant
      Set colTestTasks = item.TestTasks
      j = j + 1
      Select Case pTypeQuestion
        Case 1 ' Завдання відкритої форми
          pText = pText & colTestTasks.item(1).QuestionPrompt &
    vbCrLf  ' Умова першого завдання з
    колекції
          pTable(i, j) = CStr(colTestTasks.item(1).NumericalAnswer)
     ' Відповідь до першого завдання з
    колекції
        Case 2 ' Завдання закритої форми
          ' Тут описується формування варіанту, якщо вибрані завдання закритої
    форми
        Case 3 ' Завдання на встановлення відповідності
          ' Тут описується формування варіанту, якщо вибрані завдання на
    встановлення відповідності
      End Select
    Next item
  Next i
End Sub
```

Зауважимо, що для циклічного звертання до властивостей об'єктів необхідно створити один інтерфейс для всіх класів середнього рівня (див. лістинг 3).

Лістинг 3. Створення інтерфейсу для доступу до властивостей об'єктів.

```
' Клас IMyInterface
Public Property Get TestTasks() As Collection
  ' Це опис, реалізація буде в інших класах
End Property
```

Залишилося цей інтерфейс імплементувати у класах, які відповідають за формування конкретного завдання. На рис. 1 це класи Class1, Class2, ..., ClassN. У лістингу 4 подано описання класу Class1, у якому імплементовано інтерфейс IMyInterface, приведено демо конструктор по домовленості Class_Initialize() та властивість IMyInterface_TestTasks(), яка повертає колекцію однотипних завдань, яка власне і використовується у класі вищого рівня WorkClasses (див. лістинг 2).

Лістинг 4. Демо версія класу Class1.

```
Implements IMyInterface
' У класі Class1 описано N тестових завдань
Private Const N = 4 ' Кількість однотипних тестових завдань

```

```
' Властивості
Private pTestTasks As New Collection

' Конструктор класу
Private Sub Class_Initialize()
  ' Тут сформується N завдань і N відповідей до них
  Dim i As Integer
  Dim t As TestTask
  For i = 1 To N
    Set t = New TestTask
    ' Приклад завдання та відповіді до нього
    t.QuestionPrompt = "Умова завдання 1-го типу №" & i
    t.NumericalAnswer = 10 + i
    pTestTasks.Add t
  Next i
End Sub

' Повернення колекції завдань
Private Property Get IMyInterface_TestTasks() As Collection
  Set IMyInterface_TestTasks = pTestTasks
End Property
```

На кінець зауважимо, що для забезпечення цілісності умови завдання та відповіді до неї, а також можливості наповнення описаної оболонки новими методами та властивостями, кожний клас середнього рівня на рис. 1 звертається до класу найнижчого рівня TestTask.

Лістинг 5. Клас TestTask.

```
' Властивості
Private pQuestionPrompt As String ' Умова тестового завдання
Private pNumericalAnswer As Double ' Числова відповідь

' Методи встановлення значень
Public Property Let QuestionPrompt(Value As String)
  pQuestionPrompt = Value
End Property
Public Property Get QuestionPrompt() As String
  QuestionPrompt = pQuestionPrompt
End Property
  Public Property Let NumericalAnswer(Value As Double)
  pNumericalAnswer = Value
End Property
Public Property Get NumericalAnswer() As Double
  NumericalAnswer = pNumericalAnswer
End Property
```

Власне колекція pTestTasks у лістингу 4 містить об'єкти класу TestTask.

5. Наповнення розробленої оболонки завданнями з фінансової математики. Для практичного використання системи необхідно розробити базу знань для певної прикладної області. Що стосується фінансової математики, то було розроблено алгоритми для таких тестових завдань:

1. За перший рік банк нарахував p_1, а за другий p_2 складних відсотків. На скільки відсотків нарощена сума більша відносно початкової?
2. Суму a грн. розділено на два вклади: x грн. та y грн. під p_1 та p_2 річних відсотків відповідно. Знайдіть x, якщо відомо, що за рік початкова сума

зросла на p_3 відсотків.

3. Два платежі (векселі) номіналом a грн. та b грн., які підлягають оплаті через n_1 та n_2 днів відповідно, сторони домовилися об'єднати в один зі стоком платежу через n_3 днів. Знайдіть суму консолідованого платежу, якщо застосовується проста річна ставка $p\%$ (365/365).
4. Платіж (вексель) номіналом a грн. зі строком платежу через n_1 років сторони домовилися замінити таким чином: через n_2 роки виплатити b грн., а ще через n_3 роки погасити борг повністю. Знайдіть суму останнього платежу, якщо застосовується складна річна ставка $p\%$.
5. Наявні дві альтернативи: вкласти a євро під p_1 простих річних відсотків на n_1 місяців (360/360), або конвертувати вказану суму у гривні (b_1 грн. за 1 євро), вкласти її під p_2 простих річних відсотків на той самий строк та знову конвертувати у євро (b_2 грн. за 1 євро). Яка із вказаних схем більш прибуткова?
6. Знайдіть річний платіж скінченної річної ренти (постнумерандо), у якої сучасна величина складає a грн., а накопичена — b грн., якщо застосовується складна річна ставка $p\%$.
7. «Вічна» рента з річним платежем a грн. замінена на n-річну (постнумерандо). Знайдіть річний платіж нової ренти, якщо застосовується складна річна ставка $p\%$.
8. Кредит у розмірі a грн. видано на n років під p складних річних відсотків. Знайдіть розмір одного платежу, якщо за умовами договору погашення відбувається рівними щорічними виплатами.
9. Інвестиційний процес заданий потоком платежів: $\{(a_1; 0),\ (a_2; 1),\ (a_3; 2),\ (a_4; 3)\}$. Знайдіть термін окупності проєкту, якщо складна річна ставка складає $p\%$.
10. Проєкт характеризується початковими інвестиціями a грн. та наступним щорічним доходом b грн. протягом n років. Визначити індекс дохідності (рентабельність) інвестиційного процесу, якщо складна річна ставка складає $p\%$.

Ці тестові завдання реалізуються у класах середнього рівня на рис. 1. Власне після розроблення оболонки тільки ці класи і потрібно змінювати під конкретну прикладну область. Тут при розробленні завдань закритого типу із вибором однієї правильної відповіді або при встановленні відповідності між варіантами завдань та відповідями до них виникає ще одна підзадача: як забезпечити унікальність варіантів відповідей? У нашій системі це було реалізовано на основі таких підходів. По-перше, ми можемо генерувати n однотипних завдань, розв'язки яких потрапляють у діапазони, що не перетинаються. Тоді, наприклад, при виборі однієї відповіді будуть приведені 1 правильна та $n-1$ неправильні, які не будуть співпадати. По-друге, можна об'єднувати у один клас завдання, які мають схожі умови, але їхнє розв'язання передбачає різні обчислювальні процедури, що забезпечує різницю у відповідях. Розглянемо другий підхід на прикладі обчислення характеристик ренти. Так, параметрами ренти є річний платіж R, річна процентна ставка i та її тривалість n років. Якщо платежі здійснюються у кінці року, то рента називається постнумерандо (звичайна), на початку — пренумерандо (авансова). Розглядають майбутню та поточну вартість ренти. Так, для звичайної ренти поточна вартість (PV) — це дисконтована

сума всіх платежів [3]:

$$PV = \frac{R}{1+i} + \frac{R}{(1\ +\ i)^2} + \ldots + \frac{R}{(1\ +\ i)^n} = R \cdot \sum_{k=1}^{n} \frac{1}{(1+i)^k} = R \cdot \sum_{k=1}^{n} \frac{1}{(1+i)^k} =$$

$$= R \cdot \frac{(1+i)^{-1} \cdot \left(1 - (1+i)^{-n}\right)}{1 - (1+i)^{-1}} = R \cdot \frac{(1+i)^{-1} \cdot \left(1 - (1+i)^{-n}\right)}{1 - (1+i)^{-1}} = \frac{R \cdot \left(1 - (1+i)^{-n}\right)}{i}.$$

Майбутню вартість (FV) отримуємо з PV:

$$FV = PV \cdot (1+i)^n = \frac{R \cdot ((1+i)^n - 1)}{i}.$$

Аналогічно отримуємо майбутню (FV_{due}) та поточну вартість (PV_{due}) авансової ренти, враховуючи зміщення платежів на 1 рік:

$$PV_{\text{due}} = PV \cdot (1+i)\,;$$

$$FV_{\text{due}} = FV \cdot (1+i)\,.$$

Отже, для $R > 0$, $i > 0$ та $n > 1$ у колекцію pTestTasks (див. лістинг 4) записуємо 4 «однорідні» тестові завдання:

– знайти поточну вартість звичайної ренти;
– знайти майбутню вартість звичайної ренти;
– знайти поточну вартість авансової ренти;
– знайти майбутню вартість авансової ренти.

Зважаючи на обчислювальні процедури, отримаємо різні числові відповіді для кожного завдання. Тому це використовуємо при формуванні завдання з вибором 1 правильної відповіді — вибираємо один з 4 елементів колекції, інші використовуються для формування набору неправильних відповідей. Аналогічний підхід використовується при формуванні тестових завдань на встановлення відповідності.

6. Висновки. Описану оболонку інформаційної системи можна використовувати при формуванні обчислювальних задач із довільних предметних областей. На нашу думку, перевагою такого підходу є орієнтація на бази знань, а не на бази даних. Це дозволяє викладачу швидко формувати довільну кількість варіантів завдань як для поточного чи контрольного оцінювання, так, наприклад, і для потреб приймальних комісій та ін. Ще однією із можливостей використання системи є формування завдань для різноманітних освітніх платформ, наприклад Moodle. Серед недоліків системи є певна обмеженість мови VBA, що ускладнює використання, наприклад, бібліотек.

У подальшому систему можна розвивати як інструмент для самонавчання студентів — це і розробка методів автоматичного розв'язування задач, побудова довідникової системи та ін.

Список використаної літератури

1. Повідайчик М. М., Кацала Р. А., Годя Р. М., Янчій І. В. Розроблення інформаційної системи з фінансової математики. *Актуальні проблеми сучасної науки та освіти :* матеріали XIV Міжнародної науково-практичної конференції. Львів, 19–20 березня 2025 року. Львів : Львівський науковий форум, 2025. С. 59–60.

2. Повідайчик М. М., Повідайчик О. С., Герич М. С., Попович А. О. Розробка автоматизованих систем навчання та контролю знань учнів і студентів: навчальний посібник. Ужгород : Видавництво УжНУ «Говерла», 2022. 84 с.
3. Фінансова математика: підручник / О. В. Зайцев. Суми: СумДУ, 2022. 610 с.

Povidaichyk M. M, Sharkadi M. M., Katsala R. A., Hodia R. M., Yanchii I. V. Development of the Information System Shell "Computer-Based Trainer in Financial Mathematics".

The article describes the development of the information system shell "Computer-Based Trainer in Financial Mathematics". A key feature of the developed system is its focus on knowledge bases rather than databases, which enables the generation of a significant number of equivalent test tasks. The system allows for relatively easy expansion with new types of financial mathematics tasks and can be adapted for other branches of mathematics or computer science. The developed system can be used both for knowledge assessment and as a tool for self-learning.

Keywords: computer-based trainer, knowledge bases, financial mathematics, VBA.

References

1. Povidaychyk, M. M., Katsala, R. A., Hodia, R. M., & Yanchii, I. V. (March 19–20, 2025). Development of an Information System in Financial Mathematics. In *Actual Problems of Modern Science and Education.* Proceedings of the 14th International Scientific and Practical Conference. Lviv: Lviv Scientific Forum [in Ukrainian].
2. Povidaychyk, M. M., Povidaychyk, O. S., Herych, M. S., & Popovych, A. O. (2022). *Development of Automated Learning and Knowledge Control Systems for Pupils and Students: Textbook.* Uzhhorod: UzhNU "Hoverla" Publishing House [in Ukrainian].
3. Zaitsev, O. V. (2022). *Financial Mathematics: Textbook.* Sumy: Sumy State University [in Ukrainian].

Одержано 25.03.2025