Українська Академія державного управління
при Президентові України

Використання MS Excel при прийнятті рішень

Методичні рекомендації

Схвалено
Вченою радою
Української Академії
державного управління
при Президентові України

Київ
Видавництво УАДУ
2000

ББК 67.9(4**УКР**)300
Д36

Д36 Використання MS Excel при прийнятті рішень: Метод. реком. / Уклад А.М. Панчук. - К.: Вид-во УАДУ, 2000. - 84 с.

У збірнику наукових праць розглядаються теоретичні й практичні проблеми формування та функціонування виконавчої влади в Україні, розкриваються проблеми її становлення, розвитку та ресурсного забезпечення, теоретико-правові засади виконавчої влади з використанням нових методологічних підходів. Особливу увагу приділено громадянинові України як суб'єкту та об'єкту влади. Висвітлено нові аспекти адміністративно-правових форм діяльності органів виконавчої влади України.

Видання розраховане на наукових і практичних працівників, викладачів та студентів вузів.

Рецензенти:
доктор з державного управління, професор ***В.П. Тронь***,
кандидат економічних наук ***В.М. Сидоренко***.

Д 1203020200 - 004 / **УАДУ** - 2000 **Без оголош.**

ISBN 966-7353-50-8

Вступ

Серед слухачів Академії побутує думка, що знати тонкощі програми Excel не обов'язково, і що це потрібно лише бухгалтерам. Дійсно, можна погодитися з тим, що для бухгалтера програма Excel є чудовим інструментом. Але і державному службовцеві, і керівникові іноді доводиться вирішувати такі завдання, для яких також може згодитися цей інструмент. Ці завдання можуть бути відносно простими (наприклад: складання переліків з подальшою класифікацією або порівнянням значень, ведення та накопичення різноманітної первісної інформації - за регіонами, напрямами, фаховою спеціалізацією з подальшою її інтеграцією та обробкою) або більш складними (про деякі з них буде йтися у цих методичних рекомендаціях).

Головною особливістю програми Excel є табличне подання даних, яке є типовим для відображення інформації різноманітного призначення: економічної, демографічної, територіальної, виробничої або персональної. У простих випадках можна скористатися поширеним текстовим редактором MS Word, завдяки якому можна складати таблиці, сортувати дані і навіть виконувати нескладні обчислення. Але якщо обробка ускладнюється, наприклад необхідно синхронно обробляти дані з кількох таблиць, важко знайти кращий інструмент, ніж редактор електронних таблиць.

До додаткових засобів програми Excel, які, як правило, менш відомі користувачам, слід віднести [1; 2]:

1) зведені таблиці;

2) консолідацію даних та автоматичний підрахунок підсумків;

3) обробку інформації з використанням файлів на зовнішніх носіях;

4) використання так званих "макросів", які дають змогу проводити довільну, досить складну обробку даних.

Використання макросів

Як відомо, основним у використанні Excel є застосування формул та стандартних функцій. У версії програми Excel 95 (або Excel 7.0), яка включена до складу комплекту програм Microsoft Office 95, користувачеві доступні понад 200 стандартних функцій: від широко відомих математичних і не дуже складних внутрішніх (доступ до елементів робочого листка, інформаційні функції і т. ін.) до більш складних (таких, як функції роботи з базами даних, фінансові та статистичні функції)[1]. Але досить часто виникає потреба у виконанні більш складних дій, які поєднують послідовні стандартні дії над агрегатами даних. Спеціально для таких випадків (інакше кажучи, для автоматизації робочих процесів) до складу можливостей Excel включено **макроси**.

Макрос - це програма або група взаємопов'язаних програм, що автоматично виконують дії, які користувач загалом може здійснити також у ручному режимі. Для складання таких програм використовується одна з версій дуже поширеної сьогодні мови програмування Visual Basic, яка є складовою частиною всіх компонент Microsoft Office і має спеціальну назву Visual Basic for Application (Visual Basic для застосувань) [2]. У подальшому ми використовуватимемо загальновідоме скорочення VBA.

Елементи VBA

Мова програмування Basic належить до класу простих комп'ютерних мов. Її назва тлумачиться досить просто - це абревіатура англійської назви "Beginners Aided System Instruction Code" (код системних інструкцій, орієнтований на початківців)[2]. Додаток "Visual" тут означає використання сучасних інструментальних підходів, які дозволяють не просто програмувати, а, так би мовити, "збирати" програми з

---

[1] У подальших версіях цієї програми Excel 97 та Excel 2000 кількість та спектр стандартних функцій продовжує зростати.

[2] Бажано не плутати з Basic English (British-American Scientific International English), тобто у базову англійську мову, яка свого часу була запропонована Огденом як міжнародна: її словник налічував 850 слів, головною властивістю цієї мови було те, що з її допомогою нібито можна було визначити всі поняття.

набору стандартизованих візуальних елементів, у ролі яких виступають “об’єкти” - інше концептуальне поняття, яке також характеризує сучасну методологію (мовою професіоналів - “парадигма об’єктно-орієнтованого програмування”). З точки зору цієї методології робочий листок у документі Excel - це об’єкт, таблиця або група клітин на робочому листку - також об’єкт, вікно або його елемент - об’єкти тощо.

Основною “цеглинкою” VBA-програми є **процедура** (або підпрограма), яка задає логічно завершену послідовність дій над величинами, що характеризують об’єкти. Тому програма фактично є послідовністю визначень величин (констант або змінних) та процедур. Як правило, визначення величин передують визначенням процедур. А у випадку, коли одна процедура (головна) виконує інші (допоміжні) процедури, визначення допоміжних процедур включаються у програму раніше, ніж визначення головної. Всі визначення у Excel розміщуються у спеціальних робочих листках, які належать до

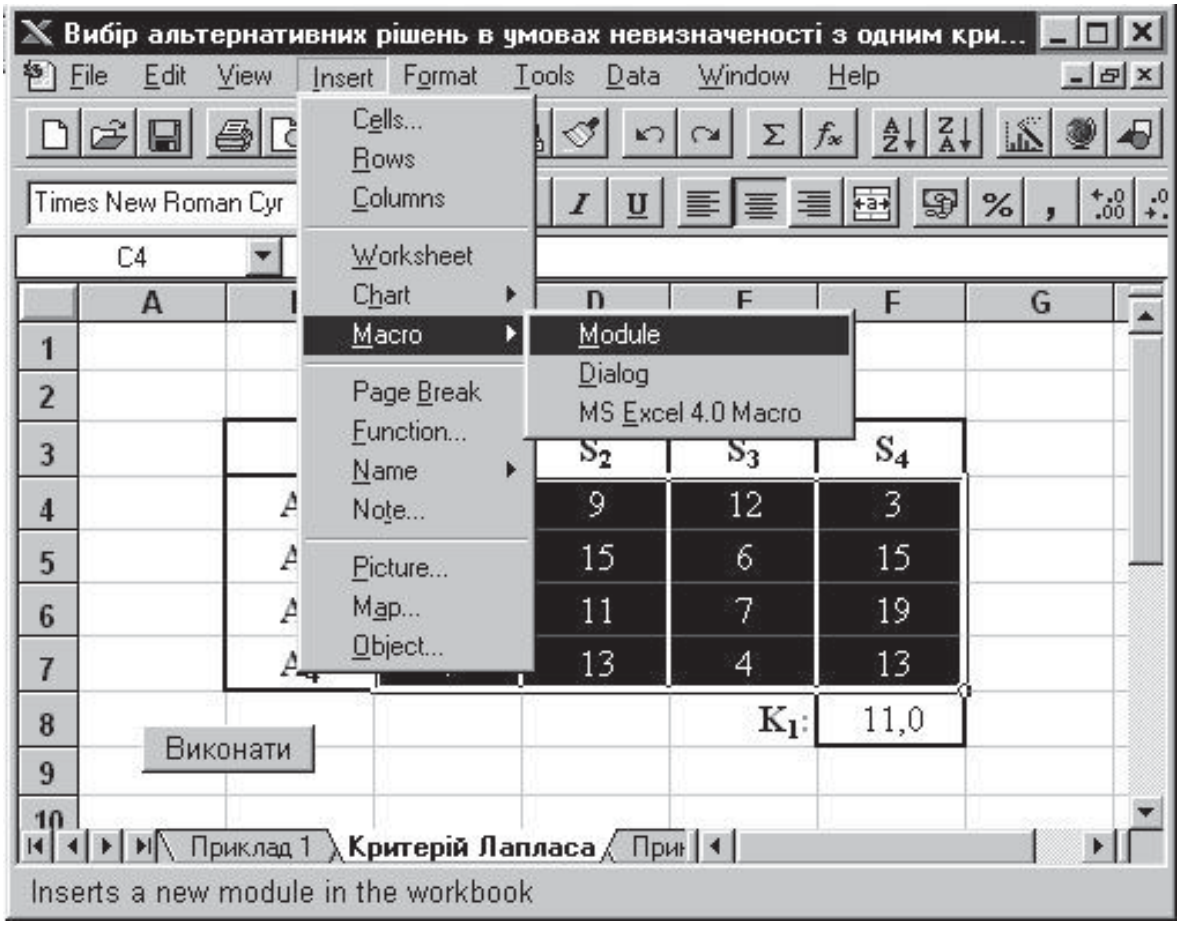


Рис. 1. ***Вставка Модуля***

класу “модулів”. Для включення в робочу книгу листка модулів у Excel 95 необхідно виконати команду меню Insert/Macro/Module[1] (рис. 1). У робочій книзі з’являється листок з назвою “Модуль *n*”, до якого користувач може включати макроси[2]. (Як і інші робочі листки, листок макросів можна перейменувати).

Процедури та змінні

Окремий макрос оформлюється як процедура:

```
Sub ім’я_процедури (аргументи)
операторі_процедури
End Sub
```

Якщо процедура не має аргументів, до першого рядка, який називається “заголовком процедури”, необхідно включити порожні дужки. Кожний аргумент має бути визначений за допомогою конструкції

```
ім’я_аргументу As тип_аргументу
```

Ім’я, або **ідентифікатор** аргументу, починається з літери, а **тип** аргументу - це один з припустимих стандартних типів даних VBA (табл.1).

Boolean - логічний тип даних, який може зберігати значення True (істина) False (хибність) і використовується у так званих умовних виразах.

Integer - звичайне ціле значення, або ціле значення одинарної довжини.

Long - ціле значення подвійної довжини.

Single - дійсне число з плаваючою крапкою.

Double - дійсне число з плаваючою крапкою подвійної точності.

---

[1] У русифікованій версії Excel 95 ця послідовність команд має вигляд як **Вставка**/**Макро**/**Модуль**. У подальшому будемо наводити еквівалентні команди в дужках.

[2] У Excel 97 та Excel 2000 суттєво змінено технологію використання VBA. Робота з макросами здійснюється за допомогою редактора VBA, який викликається командою **меню** Tools/Macro/Visual Basic Editor (**Сервис**/**Макрос**/**Редактор** Visual Basic). Користуючись командами редактора, можна включати у робочу книгу нові модулі, розробляти та редагувати вже існуючі макроси, проводити перевірку та налагодження процедур.

Таблиця 1

***Стандартні типи даних** VBA*

Тип даних	Розмір, байтів	Розрядність, цифр	Діапазон значень
Boolean	2	1	True або False
Integer	2	5	від -32768 до 32767
Long	4	10	від -2147483648 до 2147483647
Single	4	7	від -3.402823E+38 до -1.401298E-45 та від 1.401298E-45 до 3.402823E+38
Double	8	15	від -1.79769313486232E+308 до -4.94065645841247E-324 та від 4.94065645841247E-324 до 1.79769313486232E+308
Currency	8	19	від -922337203685477.5808 до 922337203685477.5807
Date	8		від 01.01.1900 до 31.12.2078
String	1 + довжина		від 0 до 65535 символів
Object	4		будь-який визначений об'єкт
Array	Визначається кількістю та розміром елементів		
Variant	Визначається даними		Будь-який стандартний тип даних

Currency - значення, які відтворюють грошові суми і дають змогу правильно виконувати округлення.

Date - значення, що відтворюють календарні дати. Цей тип даних дозволяє також відтворювати час як дробову частку дня.

String - текстове (символьне) значення. Відтворює текст посимвольно. У першому байті зберігається довжина тексту.

Object - спеціальний тип даних, який дає змогу посилатися (адресувати) на будь-який об'єкт VBA.

Array - визначає групу (масив) даних одного стандартного типу. Кожне з даних називають елементом масиву. До елемента масиву можна дістати доступ за допомогою **індексів**. Реально службове слово Array не використовується. Для визначення масиву достатньо зазначити індекси.

Variant - стандартний тип, який використовують у випадках, коли не можна визначити конкретний тип даних (так би мовити, будь-який тип). Коли виконуються обчислення з такими величинами, VBA визначає тип виходячи із самих даних. (Цей тип рекомендується застосовувати тільки у крайніх випадках, оскільки його використання призводить до уповільнення обчислень).

Крім аргументів у процедурі можуть бути застосовані інші величини - **змінні**. Якщо такі величини використовуються тільки у конкретній процедурі, їх називають допоміжними, або **внутрішніми** (**локальними**) змінними. У тому випадку, коли виникає необхідність використання однієї змінної для кількох різних процедур, їй надається статус **зовнішньої** (**глобальної**). Це досягається розміщенням опису ззовні будь-якої процедури (як правило, на початку модуля).

Змінні визначаються описами, які схожі з визначенням аргументів процедури, але починаються спеціальним оператором Dim:

```
Dim ім'я_змінної As тип_змінної
```

Наприклад:

```
Dim nRows, nCols As Integer
```

Якщо змінні використовуються у процедурах, які розміщені у різних модулях, до їх опису треба додати визначник Public, наприклад:

```
Public Dim Size As Integer
```

На відміну від змінних, **процедури** вважаються відомими і доступними у всіх модулях робочої книги. Тому, якщо виникає необхідність обмежити доступність процедури рамками поточного модуля, до її опису додається визначник Private:

```
Private Sub Compare()
```

Спеціальний визначник Const дозволяє визначати **іменовані константи**:

```
Const Pi = 3.14159265358979
Const Blue = 5
```

Масиви

Тип Array, на відміну від всіх інших стандартних типів, задається не визначником, а за допомогою так званої **вимірності**:

```
Dim ім'я_масиву (індекс_1, індекс_2...) As тип_масиву
```

Індекси задають вимірність масиву. Масив з одним індексом - аналог вектора, з двома - матриці тощо. Індекс масиву може бути заданий просто числом, або діапазоном.

Наприклад:

```
Dim Sales(5) As Currency
Dim AcctNo(700 To 799) As Integer
Dim theCoords(4, 1 To 3) As Single
Dim Birthdays(3 To 23, 5 To 7) As Date
```

У першому прикладі кожний елемент масиву може бути позначений змінною з індексом:

```
Sales(0), Sales(1), Sales(2), Sales(3), Sales(4), Sales(5)
```

Зверніть увагу на те, що у простішому випадку початковий індекс дорівнює **нулю**. Це правило може бути змінено використанням спеціального оператора

```
Option Base 1
```

який повинен починати модуль, має як область дії цей модуль і визначає, що першим елементом масиву буде елемент з індексом **один**.

У другому прикладі нам відомий діапазон значень індексів (наприклад, діапазон номерів рахунків). Цей факт явно задається визначенням індексу першого та останнього елементів.

Третій та четвертий приклади показують комбінації варіантів визначення індексів для багатовимірних масивів. Звертання до елементів багатовимірних масивів здійснюється за допомогою кількох індексів:

```
theCoords(0, 1),... theCoords(3, 2),... theCoords(4, 3)
```

У деяких випадках розмір масиву може залежати від умов виконання макросу. Типовою в Excel є ситуація обробки виділеного діапазону клітин, який, звичайно, може змінюватися. Для такого випадку у VBA передбачено спеціальний оператор

```
ReDim ім'я_масиву (індекс_1, індекс_2...)
```

який дозволяє змінити розміри масиву. Наприклад:

```
Dim mArray() As Single
' Визначення виділеної області клітинок
nRows = Selection.Rows.Count
nCols = Selection.Columns.Count
ReDim mArray(nRows, nCols)
```

Зверніть увагу на те, що третій та четвертий рядки містять звернення до об'єктів робочої книги: Selection означає виділений діапазон клітинок, Rows - рядки, Count - лічильник (кількість). Разом: **кількість рядків виділеного діапазону клітинок**.

Доступ до об'єктів Excel

Кожний об'єкт робочої книги (та інших робочих книг) можна буде використати в програмі на VBA. Загалом, кожний об'єкт має низку **властивостей** та **методів**, за допомогою яких можна дістати доступ до нього. Повний перелік цих характеристик досить великий. Тому розглянемо тільки деякі з них, до яких звертаються першочергово і найчастіше. Більш докладну, а головне - повну, інформацію можна одержати в книзі [2] або в довідкових матеріалах до Excel.

Початковими (основними) об'єктами є Workbooks - робочі книги та Worksheets - робочі листки.

Звернутися до конкретного робочого листка можна за номером або за допомогою імені (назви):

```
Workbooks("Sales").Worksheets(2)
Workbooks("Sales").Worksheets("West")
```

Імена робочої книги та робочого листка обов'язково треба брати в лапки: це загальне правило для визначення так званих **текстових констант**.

Конструкція Workbooks("Sales") у даному прикладі називається **префіксом** і показує, до якої робочої книги ми звертаємося. Якщо префікс відсутній, мається на увазі поточна робоча книга, тобто та, у якій розміщено макрос.

Щоб скористатися характеристиками робочих листків, можна звернутися до їх **властивостей**. Наприклад:

```
Worksheets.Count ' Кількість робочих листків
Worksheets.Parent.Name ' Ім'я робочої книги, до якої належать робочі листки
```

Після апострофу у VBA-програмі розміщують **коментарі**.

Щоб звернутися до конкретної клітини робочого листка або групи клітин, можна наприклад, записати:

```
Worksheets("Sheet1").Range("B2").Value
Worksheets("Sheet1").Range("B7:C9").Value
```

Range - це приклад методу (стандартної для об'єкта процедури), який визначено для об'єкта Worksheet. Range (діапазон) дозволяє послатися на діапазон клітин у стилі "колонка-буква-рядок-число". Ще один метод - Cells - дає змогу робити те саме, але в іншому стилі - "рядок-колонка".

Головною перевагою другого методу є можливість задавати координати як вирази (тобто обчислювати їх):

```
Worksheets("Sheet1").Cells(L + 5, 2).Value
```

Щоб послатися на діапазон клітин за допомогою Cells, треба комбінувати обидва методи:

```
Range(Cells(7, 2), Cells(9, 3)).Value
```

Ще один метод посилання на діапазон був застосований в одному з попередніх прикладів: мається на увазі властивість Selection. Можна, наприклад, послатися на конкретну клітину виділеної області у такий спосіб:

```
Selection.Cells(i, j).Value
```

причому адреса клітини, яка визначається методом Cells, у даному випадку задається **відносно початку** виділеної області Selection.

Оператори VBA

Більша частина того, що ми досі розглядали, стосується **даних**, тобто того, над чим виконуються дії. Для визначення дій використовуються **оператори**.

Основним у мові VBA є оператор **призначення**, за допомогою якого **змінній** може бути задано **значення**. Загальний вигляд оператора призначення:

```
ім'я_змінної = вираз
```

Ліва частина оператора призначення може бути простою змінною (змінною стандартного типу, крім типу Object, - для останнього оператор призначення має спеціальну форму), елементом масиву або властивістю об'єкта. Наприклад:

```
theFileName = "C:\VBA\EXAMPLES\example.xls"
Sales = Units * Price
Profit = Sales - Cost
Coords(3,2) = 19.37
Selection.Value = 25
Range("B5").Formula = "B4*B3-1"
```

Права частина оператора призначення - це вираз відповідного типу (у тому розумінні, що він має відповідати типу змінної у лівій частині). Найпростішим виразом є **константа** або інша проста змінна. У більш складних випадках використовуються елементарні операції та стандартні функції.

У табл. 2 і 3 наведено операції та функції для математичних обчислень і роботи з текстами. У колонці 3 табл. 2 визначається пріоритет операцій.

Можна також використовувати стандартні функції Excel.

Об'єктні змінні можуть використовуватися для маніпулювання префіксними частинами складних конструкцій доступу до об'єктів Excel. Для цього застосовується спеціальний оператор Set. Наприклад:

```
Dim theRange As Object
Set theRange = ActiveSheet.Range("B5")
theRange.Value = 10
```

Оператор Set є аналогом оператора призначення для об'єктів.

У складніших випадках застосовуються так звані оператори-конструкції, до складу яких відносять: оператори If, оператори Select та групу операторів циклу.

При конструюванні операторів виникає потреба побудови **умовних**, або **логічних** виразів. Значення таких виразів є значеннями типу Boolean. Згадані вирази будуються за допомогою операцій **порівняння** та **булевих** (операцій логіки), перелік яких наведено у табл. 4, 5.

Операції порівняння використовуються для порівняння значень двох виразів. Булеві операції іноді називають **зв'язками**, які дають змогу формулювати більш складні умовні вирази на основі порівнянь. У колонці 3 табл. 5 наведено пріоритети булевих операцій.

Приклади умовних виразів:

```
Sales > 150
Sum < 100 Or Rest >= 3
B1 And (B2 Or B3 And (B4 Xor B5) Imp B6) Or B7
```

Існують три способи запису оператора If.

1. ***Скорочений***:

```
If умовний_вираз Then
    блок_операторів
End If
```

Блок операторів буде виконано тільки в тому разі, якщо умовний вираз має значення "істина".

Таблиця 2

Математичні операції та функції *VBA*

Операції та функції	Визначення дії	Пріоритет
+	Додавання	7
-	Віднімання	7
Mod	Остача від ділення	6
\	Ділення без остачі	5
*	Множення	4
/	Ділення	4
-	Від'ємність (інверсія знаку)	3
^	Піднесення до степеня (степенювання)	2
Atn	Арктангенс числа	1
Sin	Синус кута	1
Cos	Косинус кута	1
Tan	Тангенс кута	1
Exp	Експонента $\mathbf{e}^x$	1
Log	Натуральний логарифм числа (основа **e** = 2.71828…)	1
Sqr	Корінь квадратний числа	1
Randomize	Ініціація генератора випадкових чисел	1
Rnd	Випадкове число	1
Abs	Абсолютна величина	1
Sgn	Знак числа (у вигляді +1, -1 або 0)	1
Fix	Виділення цілої частини дійсного числа	1
Int	Округлення дійсного числа	1

Таблиця 3

Операції та функції *VBA* ***для роботи з текстами***

Операції та функції	Визначення дії
&	Конкатенація (склеювання) текстів
Len	Довжина тексту
Lset	Вирівнювання вліво
Rset	Вирівнювання направо
Left	Ліва частина тексту
Right	Права частина тексту
Mid	Виділення або переміщення фрагменту тексту
Str	Перетворення числа на текст
Format	Перетворення числа на текст за форматом
Val	Перетворення текстового відображення числа на число

Таблиця 4

***Операції порівняння** VBA*

Операції та функції	Визначення дії
=	Дорівнює
<>	Не дорівнює
<	Менше
>	Більше
<=	Більше або дорівнює
>=	Менше або дорівнює
Is	Ідентичний (тільки для об'єктів)

Таблиця 5

***Булеві операції** VBA*

Операції та функції	Визначення дії	Пріоритет
Not	Інверсія (True на False і навпаки)	1
And	Логічне "І"	2
Or	Логічне "Або"	3
Xor	Виключне "Або"	3
Imp	Імплікація	3
Eqv	Еквівалентність	3

2. Стандартний, або ***повний***:

```
If умовний_вираз Then
    блок_операторів
Else
    альтернативний_блок_операторів
End If
```

Альтернативний блок операторів буде виконано тільки тоді, коли умовний вираз має значення "хибність".

3. Розширений, або ***багатоблочний***:

```
If умовний_вираз_1 Then
    блок_операторів_1
ElseIf умовний_вираз_2
    блок_операторів_2
...
Else
    альтернативний_блок_операторів
End If
```

Виконується тільки той блок операторів, для першого з яких умовний вираз має значення “істина”. У випадку, коли всі умовні вирази “хибні”, буде виконано альтернативний блок. Підконструкцій ElseIf може бути кілька, а альтернатива Else може бути взагалі відсутня.

Оператор Select є варіантом останньої форми оператора Else для того випадку, коли умовні вирази є різними варіантами значень однієї величини. Конструкція оператора така:

```
Select Case величина
    Case порівняння_1
    блок_операторів_1
    Case порівняння_2
    блок_операторів_2
    ...
    Case Else
    альтернативний_блок_операторів
End Select
```

Оператори циклу дають змогу повторювати певні послідовності дій залежно від конкретних умов. Розрізняють: **обчислюваний** цикл, чотири типи **ітеративних** циклів та цикл **об’єктного** типу.

Обчислюваний цикл застосовується, якщо треба повторити дії наперед задану кількість разів. Цей оператор має вигляд

```
For змінна_циклу = початок To кінець Step крок
    блок_операторів
Next змінна_циклу
```

Змінна циклу перебігає послідовність значень від “початку” до “кінця”, змінюючись щоразу на величину “крок”. Для кожного значення виконується блок операторів. Якщо підконструкція Step не використовується, крок вважається таким, що дорівнює +1 або -1 залежно від співвідношення початкового та кінцевого значення змінної циклу. Серед операторів блока можуть траплятися інші оператори циклу (звичайно, з іншими змінними циклу) - такі випадки називають “вкладенням циклів”. Типовою ситуацією вкладених циклів можна вважати дії, пов’язані з обробкою матриць. Наприклад:

```
For i = 1 To nRows
    For j = 1 To nCols
    mArray(i, j) = 0
    Next j
Next i
```

Цей фрагмент програми заповнює нульовими значеннями матрицю mArray.

Ітеративні цикли використовують, коли кількість повторень наперед невідома. У цьому випадку, як правило, відома умова, за якої дії мають повторюватися, або навпаки, перериватися.

Перший з чотирьох варіантів називають циклом з **передумовою**:

```
Do While умовний_вираз
    блок_операторів
Loop
```

Блок операторів буде виконуватися (і повторюватися), якщо умовний вираз має значення “істина”. В противному разі повторення закінчується.

Другий варіант оператора називають циклом з **післяумовою**:

```
Do
    блок_операторів
Loop While умовний_вираз
```

На відміну від попереднього варіанта, блок операторів буде виконано хоча б один раз і повторено, якщо умовний вираз має значення “істина”.

Дві інші форми оператора ітеративного циклу є, так би мовити, логічними доповненнями перших двох. Вони мають вигляд

```
Do Until умовний_вираз
    блок_операторів
Loop
```

та

```
Do
    блок_операторів
Loop Until умовний_вираз
```

Заміна конструкції While на конструкцію Until призводить до того, що блок операторів буде повторюватися, якщо умовний вираз має значення “хибність”.

Оператор циклу об’єктного типу у деяких випадках може бути записано більш компактно, ніж розглянутими раніше способами. Наприклад:

```
For Each c in Worksheets("Sheet1").Range("A1:D10")
    If c.Value < 0.001 Then
    c.Value = 0
    End If
Next c
```

Конструкція

```
For Each змінна_циклу in
```

яка буквально перекладається: “для всіх значень змінної циклу з...” досить зрозуміла. Якщо переписати цей цикл із застосуванням конструкції обчислюваного циклу, то потрібно було б побудувати два вкладених цикли з двома змінними циклів.

Власні функції користувача

У VBA користувач має можливість описувати власні функції - більш складні або специфічні дії, які закінчуються обчисленням певного значення. Функція дуже схожа на процедуру, але має дві істотні особливості:

1. Функція має тип значення, яке вона “повертає”.

2. Кінцеве значення має бути призначене ідентифікатору функції, який може розглядатися позначенням спеціальної змінної у середині функції.

Опис функції має такий вигляд:

```
Function ім’я_функції (аргументи) As тип_функції
    оператори_функції
    ім’я_функції = кінцеве_значення
End Function
```

Наведемо приклад функції, яка перемножує два значення:

```
Function MultiplyEm(Value1 As Single, Value2 As Single) As
Single
    MultiplyEm = Value1 * Value2
End Function
```

Оскільки функція має статус Public, її можна використати в інших процедурах та функціях для перемноження величин:

```
Res = MultiplyEm(2, 2) / 2 + 4
```

Нові функції, якщо вони не мають сторонніх ефектів, можуть бути використані для побудови робочих таблиць поряд із стандартними функціями Excel. Їх можна знайти за допомогою діалогового вікна Function Wizard (**Мастер Функций**), для виклику якого слід виконати команду Insert/Function**...** (**Вставка**/**Функция...**). Нові функції з'являються у категорії User Defined (**Пользовательские**) (рис. 2).

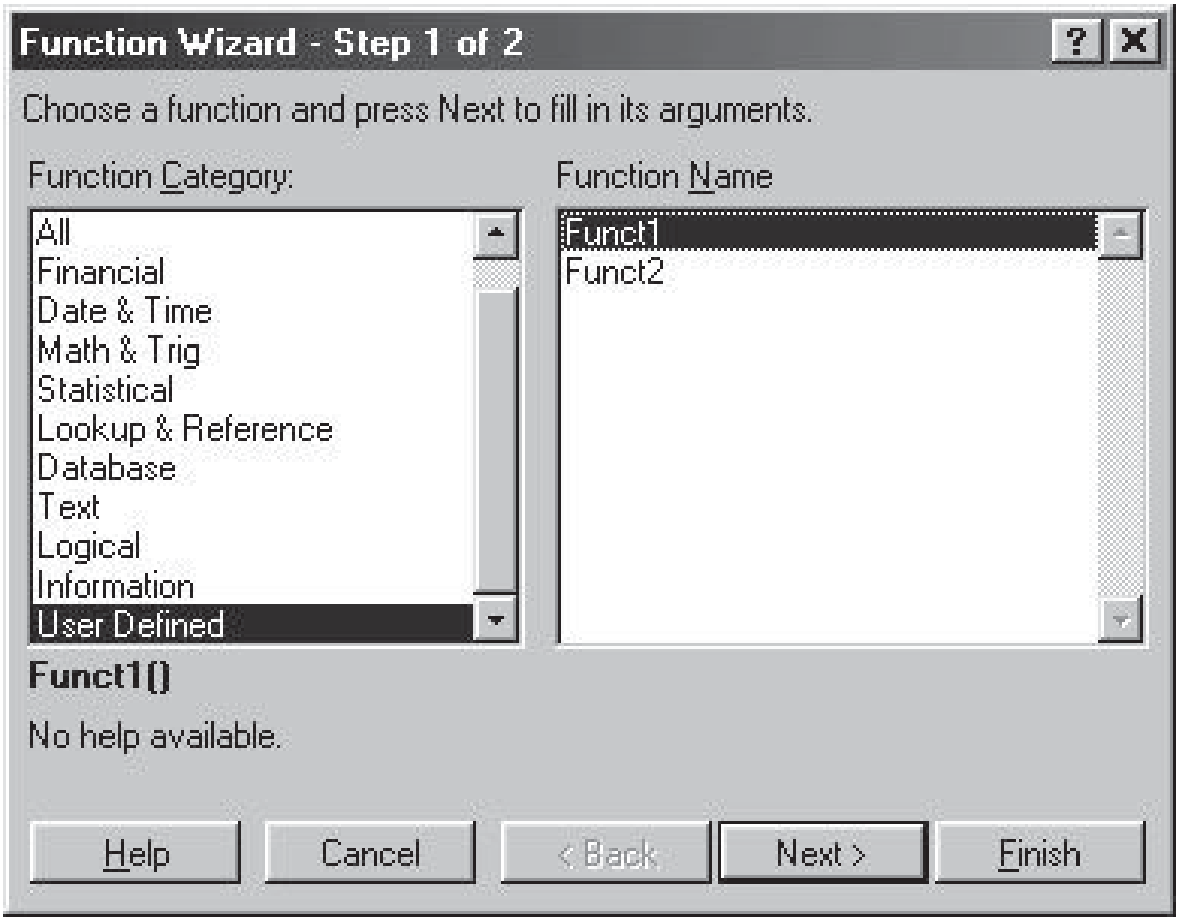


Рис. 2. ***Місце пошуку нових функцій***

Використання готових макросів

Макроси (процедури), які визначені у модулях робочої книги, можна застосовувати одним з трьох способів: **безпосереднім** запуском, **прив'язуванням** до кнопки на робочому листку і визначенням спеціального **елемента меню**. Розглянемо кожний з трьох способів.

Безпосередній запуск макросу

Всі макроси, які визначено у робочій книзі, розміщені в діалоговому вікні Macro (**Макрос**), яке можна викликати командою Tools/Macro**...** (**Сервис**/**Макрос...**) (рис. 3). Після вибору конкретного макросу (рис. 4) можна скористатися однією з кнопок у правій частині діалогового вікна. Кнопка Run (**Выполнить**) дає змогу запустити макрос у автоматичному (звичайному) режимі, кнопка Cancel (**Отмена**) - припинити діалог, кнопка Step (**Шаг**) - виконати макрос по кроках (цей режим використовують, коли виникає потреба налагодити макрос). Кнопка Edit (**Редактировать**) відкриває модуль, у якому визначено процедуру, з допомогою якої можна вносити зміни в текст програми. Кнопка Delete (**Удалить**) дає змогу вилучити макрос з робочої книги, а кнопка Options**...** (**Параметры...**) - визначити деякі додаткові властивості макросу.

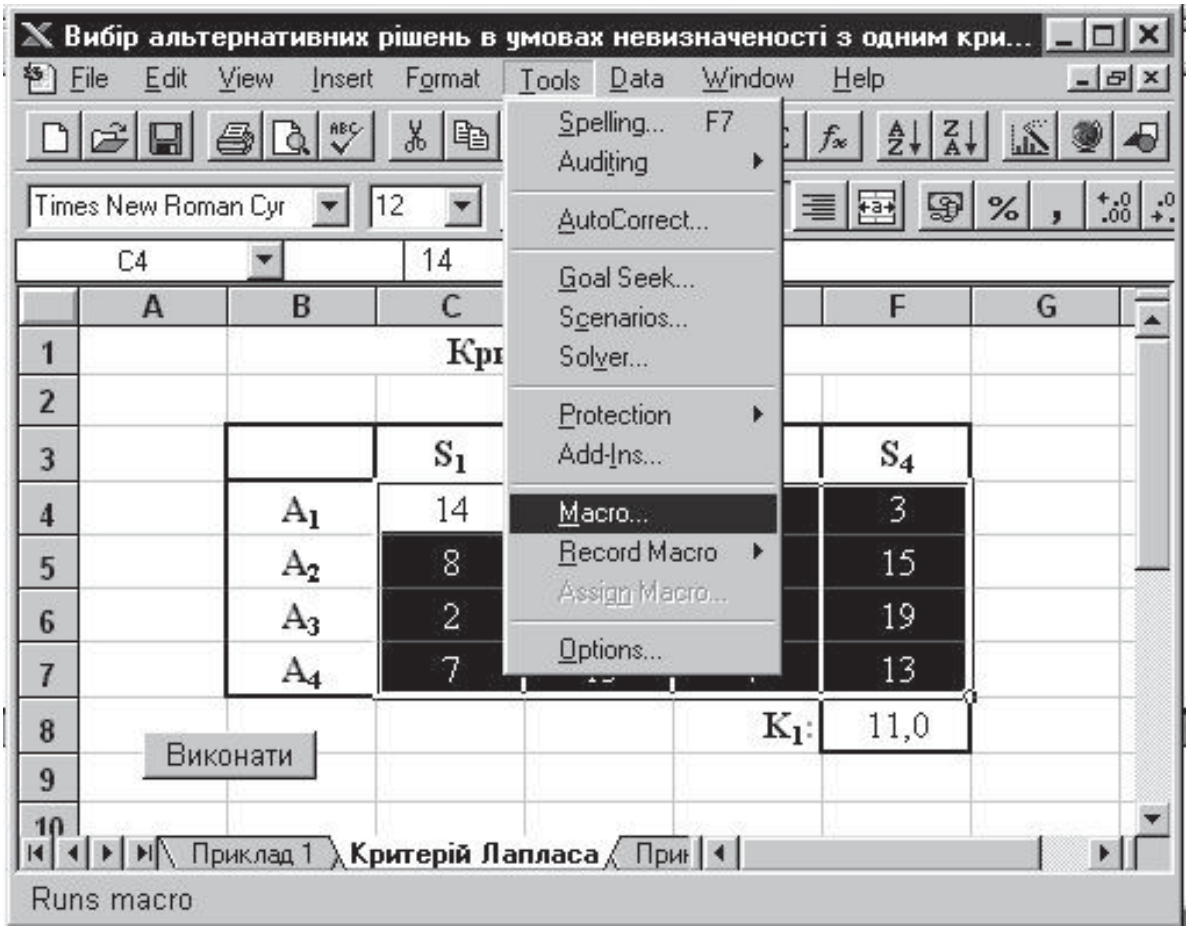


Рис. 3. ***Вибір макросу***

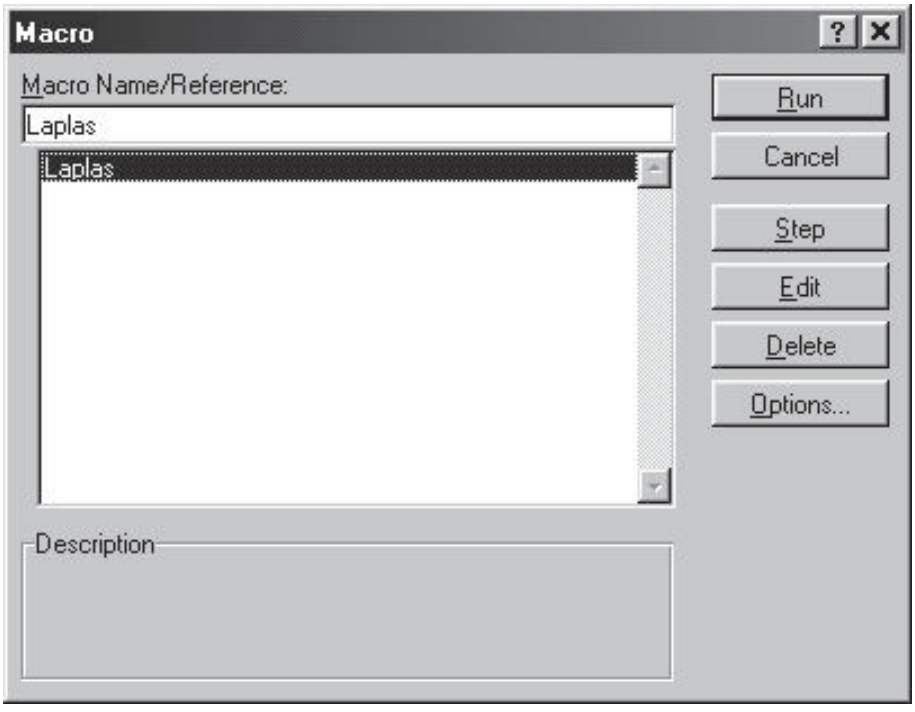


Рис. 4. ***Діалогове вікно*** *Macro*

Більш органічним видається другий спосіб, використовуючи який, можна прив'язувати макрос до кнопки, розміщеної на робочому листку.

Прив'язування макросу

Для розміщення кнопки на робочому листку необхідно виконати такі дії.

1. Зробити видимою панель інструментів Forms (**Формы**), для чого виконати команду View/Toolbars**...** (**Вид/Панели инструментов**) і позначити необхідну панель інструментів у діалоговому вікні Toolbars (**Панели инструментов**) (рис. 5). Після натискання кнопки OK позначена панель інструментів з'явиться в робочому вікні (рис. 6).

2. Вибрати на панелі інструментів Forms елемент-кнопку (Create Button - **Создать кнопку**), а також позицію в робочому листку та, не відпускаючи лівої кнопки мишки, намалювати кнопку.

3. Назву кнопки можна відредагувати, якщо зробити два послідовних (окремих) кліки (натискання) лівою кнопкою мишки. Якщо ж скористатися правою кнопкою мишки, можна викликати так зване "контекстне меню" і за його допомогою уточнити деякі параметри кнопки, головним з яких є Assign Macro**...** (**Назначить макрос**).

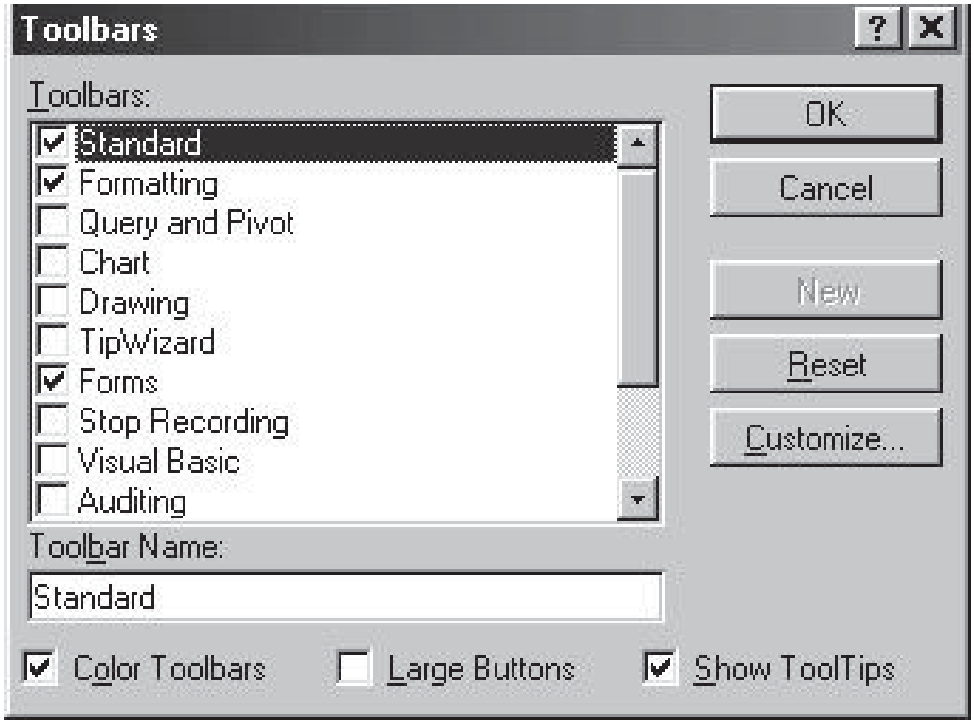


Рис. 5. ***Діалогове вікно*** *Toolbars*

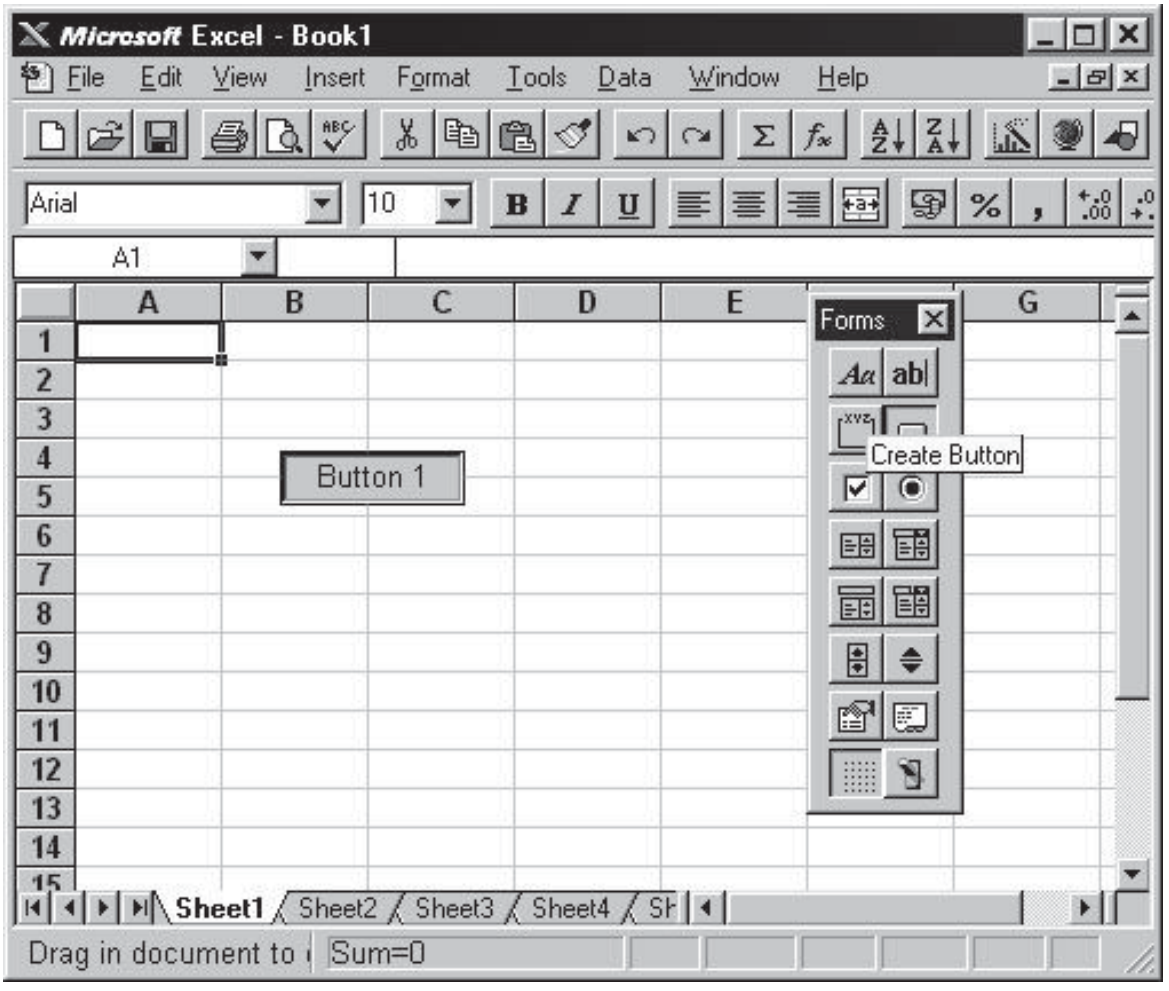


Рис. 6. ***Створення кнопки***

Визначення пунктів меню

Якщо в робочій книзі існує кілька макросів, посилання на них можна розмістити в спеціально організованому меню (точніше, у доповненні до головного меню Excel). Для цього необхідно:

1. Зробити видимою панель інструментів Visual Basic (у таким самий спосіб, як у попередньому випадку для панелі Forms (див. рис. 5).

2. Вибрати на панелі інструментів Visual Basic елемент з назвою Menu Editor (**Редактор меню**) (рис. 7).

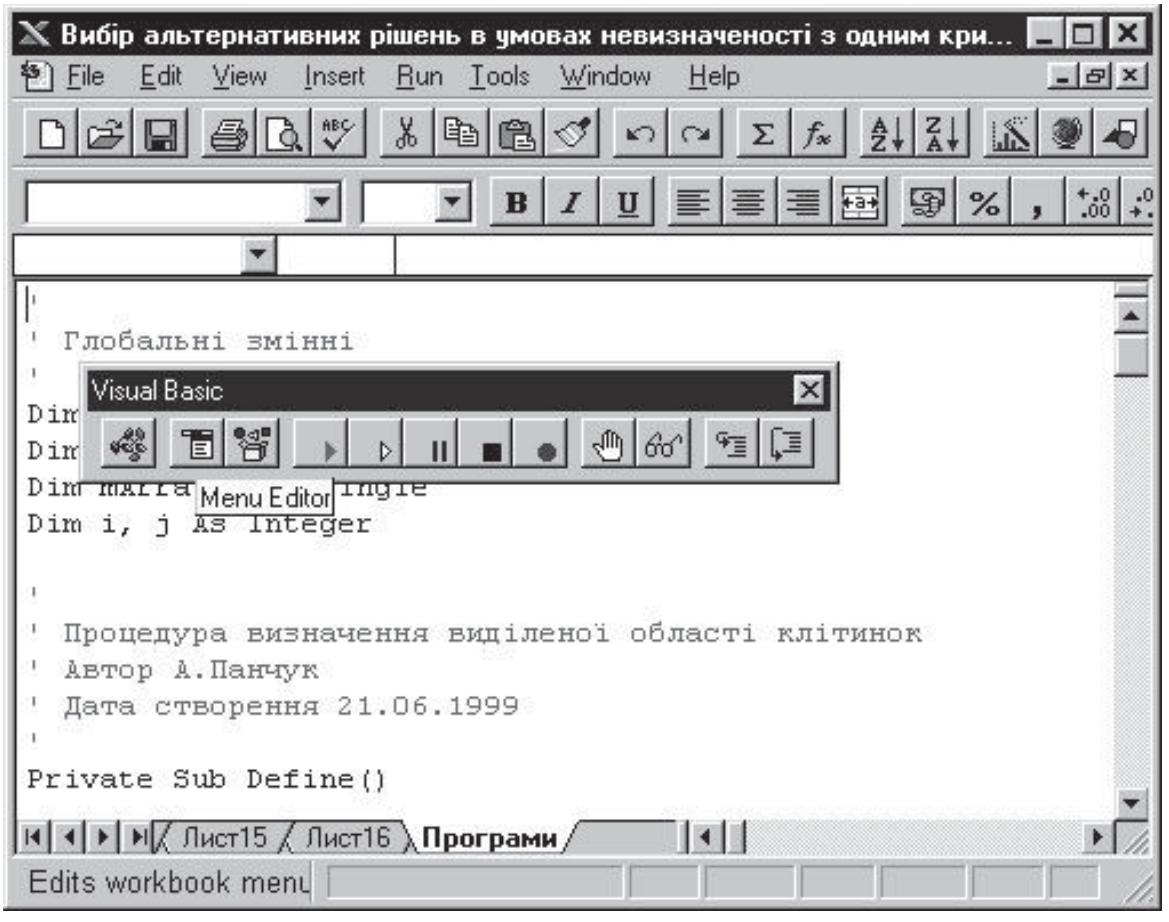


Рис. 7. ***Виклик редактора меню***

3. У діалоговому вікні Menu Editor визначити відповідний елемент меню та прив'язати до нього відповідний макрос (рис.8).

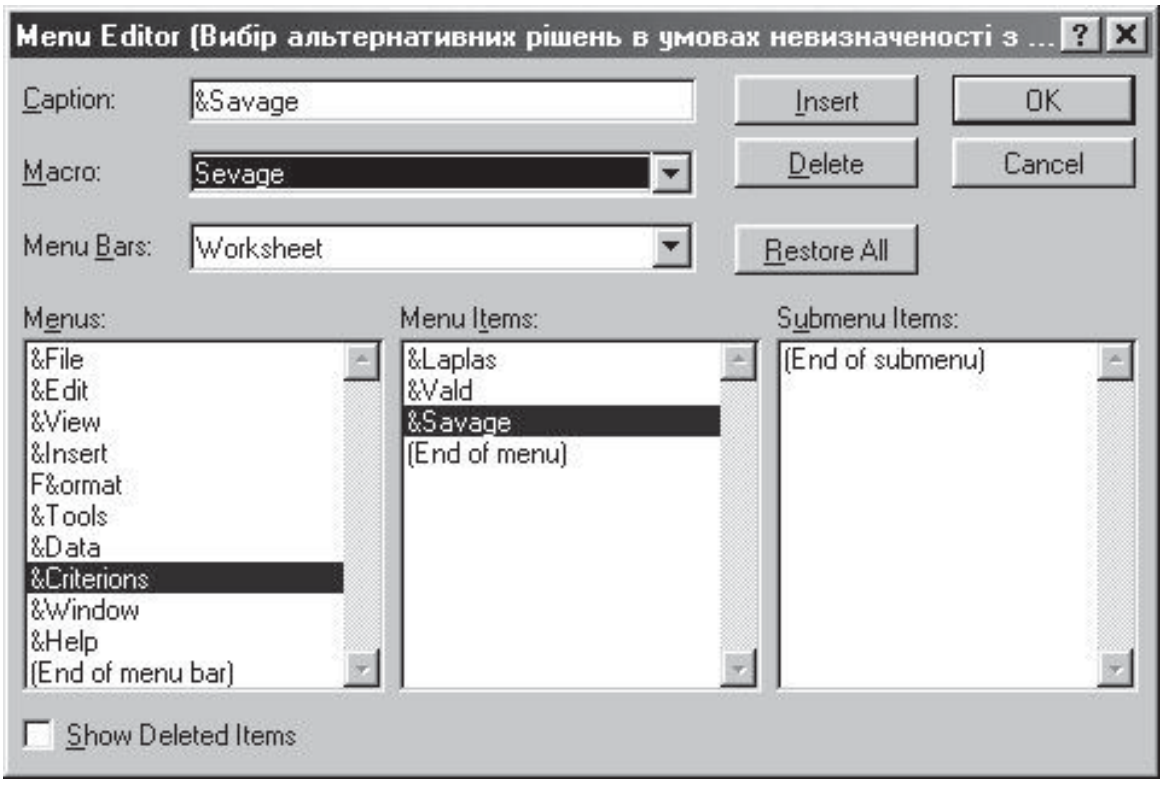


Рис.8. ***Редагування меню***

Зверніть увагу на те, що меню має кілька рівнів підпорядковування: головне меню (Menus - **Меню**), підлеглі елементи меню (Menu Items - **Элементы меню**) та елементи підменю (Submenu Items - **Пункты подменю**). Кнопкою Insert (**Вставить**) можна включити новий елемент меню на будь-якому рівні: він передуватиме виділеному у даний момент елементу. У полі Macro (**Макрос**) можна визначити відповідний макрос, вибравши його з переліку.

Застосування Excel для розв'язування задач прийняття рішень

Необхідність прийняття рішення виникає тоді, коли є потреба покращати певну ситуацію. У задачі прийняття рішення можна визначити [3]:

- особу, яка приймає рішення (ОПР);
- параметри, які визначають ситуацію і можуть бути змінені (керовані змінні);
- параметри, які також визначають стан, але не можуть бути змінені ОПР (характеристики середовища);
- показники, за допомогою яких ОПР може оцінити стан;
- принцип, згідно з яким ОПР оцінює задовільність станом, що залежить від значень керованих змінних та характеристик середовища.

Те, що називають "прийняттям рішення", насправді є останнім кроком довгострокового процесу вироблення рішення, у якому можна виділити щонайменше три етапи.

1. Передусім необхідно визначити, які дії загалом можуть бути виконані, тобто побудувати множину можливих рішень.

2. Проаналізувати певні наслідки (позитивні або негативні), до яких може призвести кожна дія. Ці наслідки ОПР оцінює на основі визначених показників, кожний з яких забезпечує кількісну оцінку наслідків та в сукупності з іншими показниками дає змогу порівнювати альтернативи і робити вибір.

3. Для вибору рішення необхідно визначити метод.

Найпростіша постановка задачі прийняття рішення може мати вигляд, наведений у табл. 6.

Таблиця 6

Постановка задачі прийняття рішення

Альтернатива	Параметр 1	Параметр 2	...	Параметр n
1	q_{11}	q_{12}	...	q_{1n}
2	q_{21}	q_{22}	...	q_{2n}
...	...	...	...	...
m	q_{m1}	q_{m2}	...	q_{mn}

Символом q_{ij} позначено показники. Рядок значень показників характеризує альтернативу, а колонка - критерій або ситуацію.

Дуже важливим для ОПР є поняття управлінського ризику, який є наслідком невизначеності, притаманній будь-якому вибору. Джерелом невизначеності вибору може бути:

- невизначеність поведінки протилежної сторони, дії якої неможливо повністю врахувати або передбачити;
- невизначеність бажань або цілей;
- слабка структуризація поведінки ОПР у момент вибору;
- невизначеність оцінки можливих наслідків реалізації рішень.

Таким чином, залишається ризик прийняття не найкращого рішення у конкретній ситуації. Ризик - це не збитки, а навпаки, ефект, який не був реалізований, або користь, якою не скористалися. “Управлінський ризик - це різниця між ефектом, що очікується, при виборі альтернатив за умовою невизначеності, та оптимальним вибором, який було б зроблено, якщо б ці умови були визначені точно. Управлінський ризик - це своєрідна сплата за відсутність інформації” [3].

Якщо показник критерію q_{ij} визначає так звану “корисність” (див., наприклад, [4]), то ризик за умови вибору альтернативи i для критерію j визначається як різниця між максимальним значенням показника для критерію q_j^{max} та його поточним значенням:

$$r_{ij} = q_j^{max} - q_{ij} . \qquad (1)$$

Різні ситуації для одного критерію можна порівнювати з урахуванням імовірності того, що ця ситуація існуватиме.

Розглянемо, як деякі з типових задач прийняття рішення можна реалізувати за допомогою програми MS Excel.

Задачі вибору альтернативних рішень в умовах невизначеності з одним критерієм

Нагадаємо, що можуть бути такі умови задач вибору:

- повна визначеність, коли стан середовища (стан природи) відомий точно;
- невизначеність, коли відомий розподіл імовірностей станів природи;
- повна невизначеність, коли відомі тільки можливі стани природи і немає жодної підстави вважати якийсь з них більш імовірним.

Далі розглядатимемо задачі з одним критерієм та кількома можливими станами природи, кожний з яких може мати свій вплив на кожну з альтернатив. У виразах будемо використовувати такі позначення:

A - альтернатива;
S - стан природи;
m - кількість альтернатив;
n - кількість станів природи;
i - індекс альтернативи;
j - індекс стану природи;
q - значення показника (корисність);
r - ризик;
p - імовірність.

Критерій Лапласа

Цей критерій застосовується в умовах повної невизначеності (його ще іноді називають "принципом недостатньої підстави") і може бути визначений так:

$$K_1 = \max_i \frac{1}{n} \sum_{j=1}^{n} q_{ij} . \qquad (2)$$

Критерій Лапласа доцільно застосовувати у випадках, коли різниця між впливом окремих станів природи досить велика. Приклад застосування критерію Лапласа наведено в табл. 7.

Таблиця 7

Приклад застосування критерію Лапласа

A	S_1	S_2	S_3	S_4	Σ	K_1
A_1	14	9	12	3	38	9,5
A_2	8	15	6	15	44	11,0
A_3	2	11	7	19	39	9,75
A_4	7	13	4	13	37	9,25
					max:	11,0

Такий не дуже складний приклад можна розв'язати стандартними засобами Excel. Для цього достатньо у колонку Σ записати формулу підсумовування, у колонку K_1 - результат ділення суми на кількість станів природи, а в клітини, що містить результат, - формулу визначення максимального значення в стовпчику.

Але більш гнучким рішенням буде створення спеціального макросу, або процедури, складеної за допомогою VBA (лістинг 1).

```
' Лістинг 1
'
' Програма застосування критерію Лапласа
' Автор А.Панчук
' Дата створення 23.05.1999
'
Sub Laplas()
    Dim nRows, nCols As Integer
    Dim rRow, rCol, rAlt As Integer
    Dim i, j As Integer
    Dim rSum, rAver As Single
    Dim mArray() As Single
    ' Визначення виділеної області клітинок
    nRows = Selection.Rows.Count
    nCols = Selection.Columns.Count
    ReDim mArray(nRows, nCols)
    ' Позиція результату
    rRow = Selection.Row + nRows
    rCol = Selection.Column + nCols - 1
```

```
' Копіювання вмісту клітин у масив
For i = 1 To nRows
For j = 1 To nCols
mArray(i, j) = Selection.Cells(i, j).Value
Next j
Next i
' Обчислення критерію Лапласа
rAver = 0
For i = 1 To nRows
rSum = 0
For j = 1 To nCols
rSum = rSum + mArray(i, j)
Next j
If rSum / nCols > rAver Then
rAver = rSum / nCols
rAlt = i
End If
Next i
```

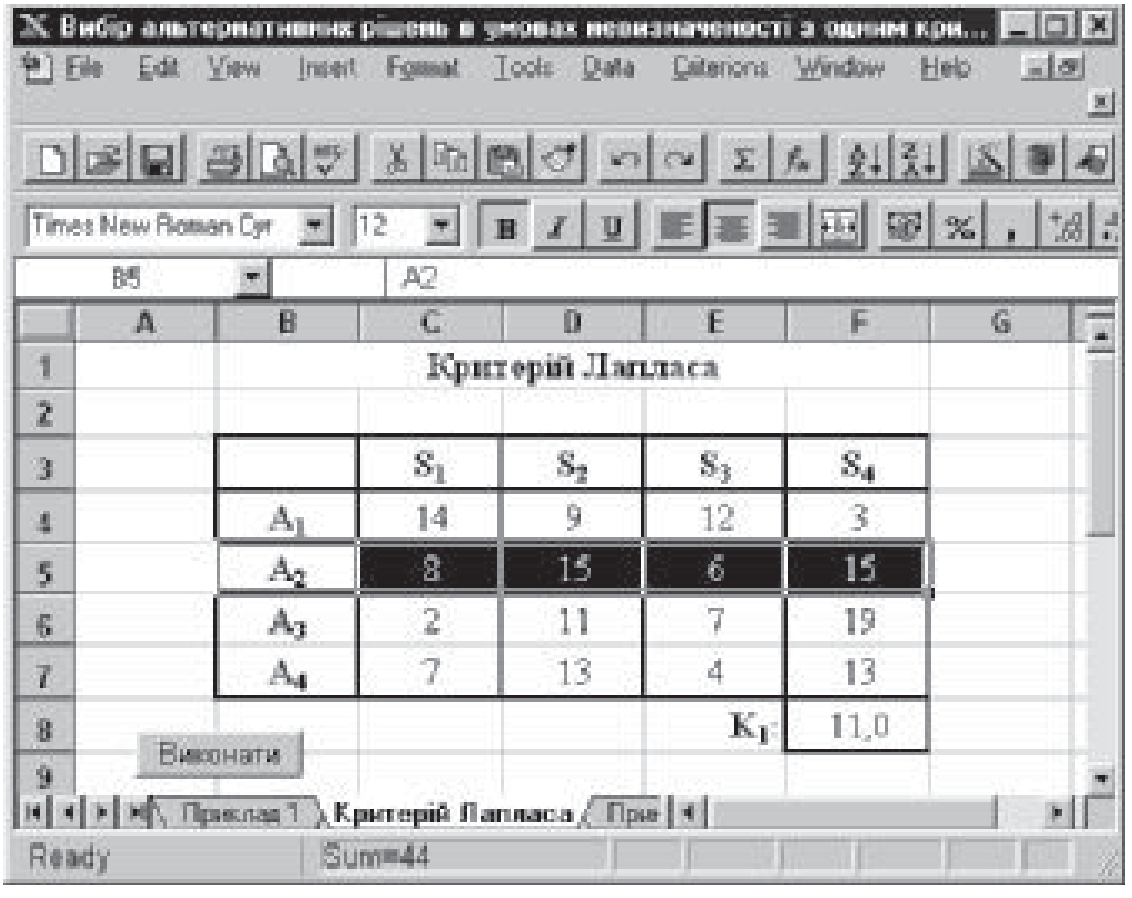


Рис. 9. ***Критерій Лапласа***

```
    ' Запис результату
    Cells(rRow, rCol) = rAver
    ' Визначення альтернативи
    rAlt = rRow - nRows + rAlt - 1
    Range(Cells(rAlt, rCol - nCols), Cells(rAlt,
rCol)).Activate
End Sub
```

Оскільки всі обчислення за формулою (2) запрограмовані у процедурі Laplas, немає потреби відтворювати послідовність обчислень та проміжні результати в робочому листку. Але за бажанням розробника програми це може бути реалізовано без особливих складнощів.

В результаті розв'язок задачі згідно з лістингом 1 набере вигляду, як показано на рис. 9: досить задати таблицю і заповнити її початковими даними, виділити дані та виконати макрос Laplas, щоб здобути результат. У даному випадку макрос прив'язано до кнопки **Виконати**. Шукана альтернатива виділяється.

Критерій Вальда (максимінний)

Застосування цього критерію в умовах повної невизначеності є характерним для обережних ОПР, котрі орієнтуються на найгірші умови, у яких вибираються альтернативи. Цей критерій часто називають максимінним, оскільки він визначається так:

$$K_2 = \max_i \min_j q_{ij} \ . \qquad (3)$$

Його іноді називають ще критерієм песимізму, або перестраховки. Приклад цього критерію наведено в табл. 8.

Таблиця 8

Приклад застосування критерію Вальда

A	S_1	S_2	S_3	S_4	min
A_1	14	9	12	3	3
A_2	8	15	6	15	6
A_3	2	11	7	19	2
A_4	7	13	4	13	4
				max:	6

Легко бачити, що вибір альтернативи A_2 дає змогу одержати не найгірший виграш за будь-яких умов: якщо настане стан природи S_3, цей виграш буде найменшим з можливих, а в решті ситуацій - значно більшим. Фахівці вважають, що критерій Вальда дає **найкращий гарантований** результат прийняття рішення.

Процедура, яка реалізує критерій Вальда, наведена у лістингу 2.

```
' Лістинг 2
'
' Програма застосування критерію Вальда
' Автор А.Панчук
' Дата створення 14.06.1999
    '
Sub Vald()
    Dim nRows, nCols As Integer
    Dim rRow, rCol, rAlt As Integer
    Dim i, j As Integer
    Dim iMax, jMin As Single
    Dim mArray() As Single
    ' Визначення виділеної області клітинок
    nRows = Selection.Rows.Count
    nCols = Selection.Columns.Count
    ReDim mArray(nRows, nCols)
    ' Позиція результату
    rRow = Selection.Row + nRows
    rCol = Selection.Column + nCols - 1
    ' Копіювання вмісту клітин у масив
    For i = 1 To nRows
        For j = 1 To nCols
            mArray(i, j) = Selection.Cells(i, j).Value
        Next j
    Next i
    ' Обчислення критерію Вальда
    iMax = 0
    For i = 1 To nRows
        jMin = 99999 ' Умовне максимальне значення виграшу
```

```
        For j = 1 To nCols
            If jMin > mArray(i, j) Then
                jMin = mArray(i, j)
            End If
        Next j
        If iMax < jMin Then
            iMax = jMin
            rAlt = i
        End If
    Next i
    ' Запис результату
    Cells(rRow, rCol) = iMax
    ' Визначення альтернативи
    rAlt = rRow - nRows + rAlt - 1
    Range(Cells(rAlt, rCol - nCols), Cells(rAlt, rCol)).Activate
End Sub
```

В результаті розв'язок задачі згідно з лістингом 2 набере вигляду, який наведено на рис. 10: досить задати таблицю і заповнити її початковими даними, виділити дані та виконати макрос Vald, щоб здобути результат. У даному випадку, як і у попередньому, макрос прив'язано до кнопки **Виконати**. Шукана альтернатива виділяється.

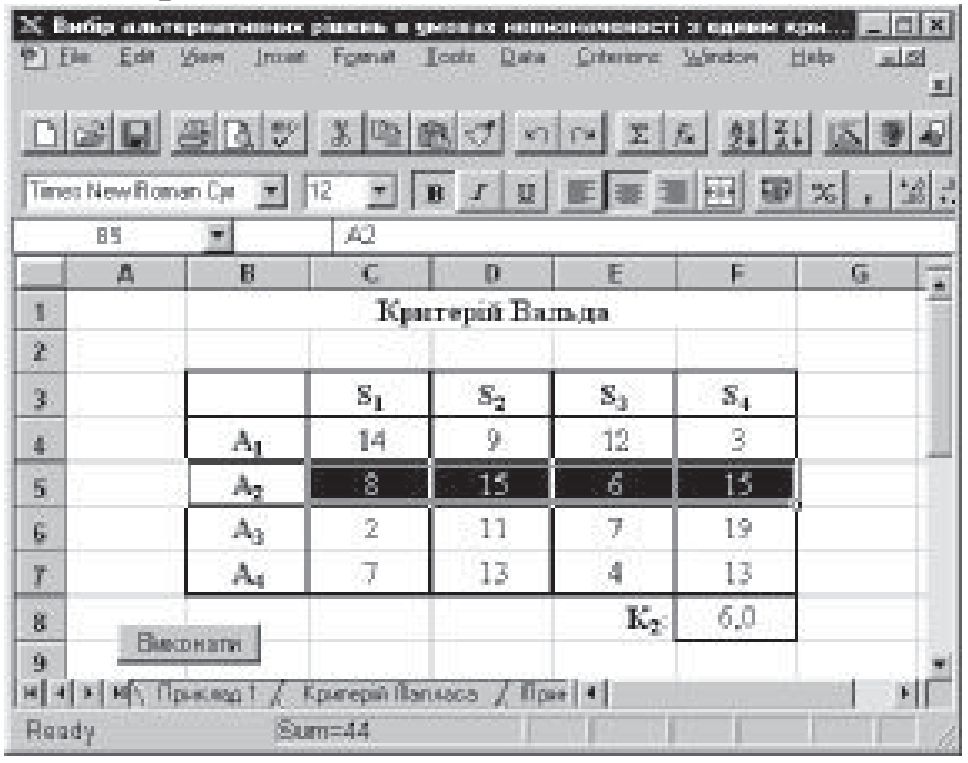


Рис.10. ***Критерій Вальда***

Критерій Севіджа (мінімального ризику)

Цей критерій грунтується на принципі мінімаксу наслідків помилкового рішення, тобто робиться спроба мінімізувати "втрачену користь". Неважко бачити (див. табл. 8), що для альтернативи A_2 максимальний виграш буде як у випадку стану природи S_2, так і стану природи S_4. У кожному випадку отримаємо 15 одиниць виграшу. Але, якщо б напевно було відомо, що стан природи буде S_4, ми, не вагаючись, вибрали б альтернативу A_4 і мали б виграш у 19 одиниць. Таким чином, за незнання інформації ми розплачуємося втратою 19 - 15 = 4 одиниць. Це - втрачена користь, тобто "ризик" (1). Щоб визначити розмір ризику, треба у кожній колонці знайти максимальний виграш, відносно якого обчислити ризик (табл.9). При цьому критерій Севіджа визначається так:

$$K_3 = \min_i \max_j r_{ij} = \min_i \max_j (\max_i q_{ij} - q_{ij}) \ . \quad (4)$$

Критерій Севіджа дає змогу не допустити надмірно високих втрат, до яких можуть призвести помилкові рішення. Цей критерій є досить популярним і його можна рекомендувати в разі прийняття рішень на довгостроковий період (наприклад, розподіл капіталовкладень на перспективу).

Приклад застосування критерію Севіджа наведено в табл.9.

Таблиця 9

Матриця ризиків та критерій Севіджа

A	S_1	S_2	S_3	S_4	max
A_1	0	6	0	16	16
A_2	6	0	6	4	6
A_3	12	4	5	0	12
A_4	7	2	8	6	8
				min:	6

Процедура, яка реалізує критерій Севіджа, наведена у лістингу 3.

```
' Лістинг 3
'
' Глобальні змінні
'
Dim nRows, nCols As Integer
Dim rRow, rCol, rAlt As Integer
Dim mArray() As Single
Dim i, j As Integer
'
' Процедура визначення виділеної області клітинок
' Автор А.Панчук
' Дата створення 21.06.1999
'
Private Sub Define()
    ' Визначення виділеної області клітинок
    nRows = Selection.Rows.Count
    nCols = Selection.Columns.Count
    ReDim mArray(nRows, nCols)
    ' Позиція результату
    rRow = Selection.Row + nRows
    rCol = Selection.Column + nCols - 1
    ' Копіювання вмісту клітин у масив
    For i = 1 To nRows
        For j = 1 To nCols
            mArray(i, j) = Selection.Cells(i, j).Value
        Next j
    Next i
End Sub
'
' Процедура визначення результату
' Автор А.Панчук
' Дата створення 21.06.1999
'
Private Sub Result(rValue As Single)
```

```
    ' Запис результату
    Cells(rRow, rCol) = rValue
    ' Визначення альтернативи
    rAlt = rRow - nRows + rAlt - 1
    Range(Cells(rAlt, rCol - nCols), Cells(rAlt, rCol)).Activate
End Sub
'
' Програма застосування критерію Севіджа
' Автор А.Панчук
' Дата створення 21.06.1999
'
Sub Savage()
    Dim iMin, jMax As Single
    Dim rArray() As Single
    Dim qMax As Single
    ' Виклик допоміжної процедури Define
    Define
    ' Визначення матриці ризику
    ReDim rArray(nRows, nCols)
    For j = 1 To nRows
        qMax = 0
        For i = 1 To nCols
            If qMax < mArray(i, j) Then
                qMax = mArray(i, j)
            End If
        Next i
        For i = 1 To nCols
            rArray(i, j) = qMax - mArray(i, j)
        Next i
    Next j
    ' Обчислення критерію Севіджа
    iMin = 99999 " Умовне максимальне значення виграшу
    For i = 1 To nRows
        jMax = 0
        For j = 1 To nCols
            If jMax < rArray(i, j) Then
```

```
                jMax = rArray(i, j)
            End If
        Next j
        If iMin > jMax Then
            iMin = jMax
            rAlt = i
        End If
    Next I
    ' Запис результату
    Result( iMin)
End Sub
```

На відміну від двох попередніх прикладів, у цьому макросі використані додаткові процедури для визначення виділеної області клітинок та запису результату, для чого кілька змінних перейшли в розряд глобальних.

Критерій Байєса (максимуму середнього виграшу)

Цей критерій можна застосувати, якщо відомий розподіл імовірностей станів природи (чого, до речі, ОПР має завжди прагнути). Знання ймовірності дає змогу визначити математичне сподівання користі для кожної альтернативи. При цьому критерій Байєса визначається за формулою

$$K_4 = \max_i \sum_{j=1}^{n} p_j \, q_{ij} \qquad (5)$$

за умови, що

$$0 \le p_j \le 1 \; ; \; \sum_{j=1}^{n} p_j = 1 \; . \qquad (6)$$

Приклад з урахуванням певних імовірностей наведено у табл.10.

Таблиця 10

Приклад застосування критерію Байєса

A	S_1		S_2		S_3		S_4		Σ
	q	pq	q	pq	q	pq	q	pq	
A_1	14	2,1	9	3,6	12	2,4	3	0,75	8,85
A_2	8	1,2	15	6,0	6	1,2	15	3,75	12,15
A_3	2	0,3	11	4,4	7	1,4	19	4,75	10,85
A_4	7	1,05	13	5,2	4	0,8	13	3,25	10,3
p	0,15		0,4		0,2		0,25		1,0
								max:	12,15

Процедура, що реалізує критерій Байєса, наведена у лістингу 4.

```
' Лістинг 4
'
' Програма застосування критерію Байєса
' Автор А.Панчук
' Дата створення 24.06.1999
'
Sub Bayes()
    Dim iMax, jSum As Single
    Dim pqArray() As Single
    ' Виклик допоміжної процедури Define
    Define
    ' Перевірка коректності умов для ймовірностей
    jSum = 0
    For j = 1 To nCols
        If (mArray(nRows, j) < 0#) Or (mArray(nRows, j) > 1#)
            Then
            Rsp = MsgBox("Значення ймовірності не може
                бути < 0 або > 1.", _ vbCritical, "Помилка у даних")
            Exit Sub
        End If
        jSum = jSum + mArray(nRows, j)
    Next j
```

```
    If jSum <> 1# Then
        Rsp = MsgBox("Сума ймовірностей має дорівнювати
            одиниці.", vbCritical,_ "Помилка у даних")
        Exit Sub
    End If
    ' Обчислення критерію Байєса
    ReDim pqArray(nRows - 1, nCols)
    iMax = 0
    For i = 1 To nRows - 1
        jSum = 0
        For j = 1 To nCols
            pqArray(i, j) = mArray(i, j) * mArray(nRows, j)
            jSum = jSum + pqArray(i, j)
        Next j
        If iMax < jSum Then
            iMax = jSum
            rAlt = i
        End If
    Next i
    ' Запис результату
    Result (iMax)
End Sub
```

Зверніть увагу на те, що в цьому прикладі реалізована перевірка коректності даних для значень імовірності згідно з умовами (5). Наявність помилки фіксується викликом стандартної функції VBA - MsgBox (Message Box - блок повідомлення). Якщо користувач припускається помилки, робити обчислення не можна. Тому виконання макросу припиняється завдяки виклику спеціального оператора Exit Sub (вийти з процедури).

Критерій Байєса-Севіджа (мінімального середнього ризику)

Цей критерій аналогічний попередньому, тільки замість максимізації середнього виграшу ОПР мінімізує середній ризик. Тому критерій Байєса - Севіджа можна визначити за формулою

$$K_1 = \min_i \sum_{j=1}^{n} p_j r_{ij} = \min_i \sum_{j=1}^{n} (\max_i q_{ij} - q_{ij}) \quad (7)$$

з урахуванням обмежень (6).

Приклад з урахуванням ризиків та певних імовірностей наведено у табл.11.

Таблиця 11

Матриця ризиків та критерій Байєса - Севіджа

A	S_1		S_2		S_3		S_4		Σ
	r	pr	r	pr	r	pr	r	pr	
A_1	0	0,0	6	2,4	0	0,0	16	4,0	6,4
A_2	6	0,9	0	0,0	6	1,2	4	1,0	3,1
A_3	12	1,8	4	1,6	5	1,0	0	0,0	4,4
A_4	7	1,05	2	0,8	8	1,6	6	1,5	4,95
p	0,15		0,4		0,2		0,25		1,0
								max:	3,10

Процедура, яка реалізує критерій Байєса - Севіджа, наведена в лістингу 5.

```
' Лістинг 5
'
' Програма застосування критерію Байєса - Севіджа
' Автор А.Панчук
' Дата створення 24.06.1999
'
Sub BayesSavage()
    Dim iMin, qMax As Single
    Dim jSum As Single
    Dim rArray() As Single
```

```
' Виклик допоміжної процедури Define
Define
' Перевірка коректності умов для ймовірностей
jSum = 0
For j = 1 To nCols
    If (mArray(nRows, j) < 0#) Or (mArray(nRows, j) > 1#)
       Then
        Rsp = MsgBox("Значення ймовірності не може
           бути < 0  або > 1.", _vbCritical, "Помилка у даних")
        Exit Sub
    End If
    jSum = jSum + mArray(nRows, j)
Next j
If jSum <> 1# Then
    Rsp = MsgBox("Сума ймовірностей має дорівнювати
       одиниці.", vbCritical, _ "Помилка у даних")
    Exit Sub
End If
' Визначення матриці ризику
ReDim rArray(nRows - 1, nCols)
For j = 1 To nCols
    qMax = 0
    For i = 1 To nRows - 1
        If qMax < mArray(i, j) Then
            qMax = mArray(i, j)
        End If
    Next i
    For i = 1 To nRows - 1
        rArray(i, j) = qMax - mArray(i, j)
    Next i
Next j
' Обчислення критерію Байєса - Севіджа
iMin = 99999 " Умовне максимальне значення ризику
For i = 1 To nRows - 1
    jSum = 0
    For j = 1 To nCols
```

```
            rArray(i, j) = rArray(i, j) * mArray(nRows, j)
            jSum = jSum + rArray(i, j)
        Next j
        If iMin > jSum Then
            iMin = jSum
            rAlt = i
        End If
    Next i
    ' Запис результату
    Result (iMin)
End Sub
```

Критерій виграшу за Гурвіцем (оптимізму-песимізму)

У деяких випадках вибору альтернативного рішення видається логічним керуватися не ортодоксальними міркуваннями (від максимально оптимістичних до надто песимістичних), а міркуваннями з позиції компромісу. Саме таке правило вибору в умовах повної невизначеності запропонував математик Гурвіц. Згідно з цим правилом оптимальний вибір визначається так:

$$K_6 = \max_i \left\{ \lambda \max_j q_{ij} + (1 - \lambda) q_{ij} \right\} \quad (8)$$

за умови

$$0 \le \lambda \le 1 , \quad (9)$$

де λ - ступінь упевненості: чим складніше ситуація, чим більше хоче "підстраховатися" ОПР, тим ближчим до одиниці вибирається значення коефіцієнта.

Застосування цього критерію може ускладнитись, якщо немає достатнього уявлення про значення коефіцієнта λ, оскільки воно може спричинити неоднакові розв'язки. Розглянемо варіанти розв'язування нашої задачі для трьох значень:

$$\lambda_1 = 0{,}4,\ \lambda_2 = 0{,}6 \text{ та } \lambda_3 = 0{,}8.$$

Результати обчислень наведено в табл. 12.

Таблиця 12

Варіанти розв'язків згідно з критерієм виграшу за Гурвіцем

A	S_1	S_2	S_3	S_4	max	min	λ_1=0,4	λ_2=0,6	λ_3=0,8
A_1	14	9	12	3	14	3	7,4	9,6	11,8
A_2	8	15	6	15	15	6	9,6	11,4	13,2
A_3	2	11	7	19	19	2	8,8	12,2	15,6
A_4	7	13	4	13	13	4	7,6	9,4	11,2
						max:	9,6	12,2	15,6

Процедура, яка реалізує критерій виграшу за Гурвіцем, наведена в лістингу 6.

```
' Лістинг 6
'
' Програма застосування критерію виграшу за Гурвіцем
' Автор А.Панчук
' Дата створення 25.06.1999
'
Sub GainHurwicz()
    Dim iMax, jMax, jMin As Single
    Dim Lambda, Expect As Single
    ' Виклик допоміжної процедури Define
    Define
    ' Введення Lambda
    Lambda = CSng(InputBox("Введіть значення коефіцієнта
       впевненості:", _ "Критерій виграшу за Гурвіцем"))
    If (Lambda < 0#) Or (Lambda > 1#) Then
       Rsp = MsgBox("Значення коефіцієнта впевненості не
          може бути < 0 "_ & "або > 1.", vbCritical, "Помилка
             у даних")
       Exit Sub
    End If
    ' Обчислення критерію виграшу за Гурвіцем
    iMax = 0
    For i = 1 To nRows
       jMax = 0
```

```
            jMin = 99999 ' Умовне максимальне значення виграшу
            For j = 1 To nCols
                If jMax < mArray(i, j) Then
                    jMax = mArray(i, j)
                End If
                If jMin > mArray(i, j) Then
                    jMin = mArray(i, j)
                End If
            Next j
            Expect = Lambda * jMax + (1 - Lambda) * jMin
            If iMax < Expect Then
                iMax = Expect
                rAlt = i
            End If
        Next i
        ' Запис результату
        Result (iMax)
    End Sub
```

У цьому прикладі вперше застосована стандартна функція VBA InputBox (блок введення даних). Оскільки значення цієї функції має тип String, використовується функція перетворення даних з типу String на тип Single - CSng (Convert to Single). Після введення значення коефіцієнта перевіряється.

Критерій ризику за Гурвіцем

Цей критерій аналогічний попередньому, тільки грунтується не на виграшах, а на ризиках (як і критерій Севіджа). Тому він визначається за формулою

$$K_7 = \min_i \left\{ \lambda \min_j r_{ij} + (1 - \lambda) r_{ij} \right\} \quad (10)$$

з урахуванням обмеження (8).

Приклад з урахуванням ризиків наведено в табл.13.

Таблиця 13

Варіанти розв'язків згідно з критерієм ризику за Гурвіцем

A	S_1	S_2	S_3	S_4	min	max	λ_1=0,4	λ_2=0,6	λ_3=0,8
A_1	0	6	0	16	0	16	9,6	6,4	3,2
A_2	6	0	6	4	0	6	3,6	2,4	1,2
A_3	12	4	5	0	0	12	7,2	4,8	2,4
A_4	7	2	8	6	2	8	5,6	4,4	3,2
						min:	3,6	2,4	1,2

Процедура, яка реалізує критерій ризику за Гурвіцем, наведена у лістингу 7.

```
' Лістинг 7
'
' Програма застосування критерію ризику за Гурвіцем
' Автор А.Панчук
' Дата створення 27.06.1999
'
Sub RiskHurwicz()
    Dim iMin, jMin, jMax As Single
    Dim Lambda, Expect As Single
    Dim rArray() As Single
    Dim qMax As Single
    ' Виклик допоміжної процедури Define
    Define
    ' Введення Lambda
    Lambda = CSng(InputBox("Введіть значення коефіцієнта
       впевненості:", _ "Критерій ризику за Гурвіцем"))
    If (Lambda < 0#) Or (Lambda > 1#) Then
       Rsp = MsgBox("Значення коефіцієнта впевненості не
          може бути < 0 " & "або > 1.", vbCritical, "Помилка
             у даних")
       Exit Sub
    End If
    ' Визначення матриці ризику
    ReDim rArray(nRows, nCols)
    For j = 1 To nCols
```

```
            qMax = 0
            For i = 1 To nRows
                If qMax < mArray(i, j) Then
                    qMax = mArray(i, j)
                End If
            Next i
            For i = 1 To nRows
                rArray(i, j) = qMax - mArray(i, j)
            Next i
        Next j
        ' Обчислення критерію ризику за Гурвіцем
        iMin = 99999 " Умовне максимальне значення
        For i = 1 To nRows
            jMin = 99999 " Умовне максимальне значення
            jMax = 0
            For j = 1 To nCols
                If jMin > rArray(i, j) Then
                    jMin = rArray(i, j)
                End If
                If jMax < rArray(i, j) Then
                    jMax = rArray(i, j)
                End If
            Next j
            Expect = Lambda * jMin + (1 - Lambda) * jMax
            If iMin > Expect Then
                iMin = Expect
                rAlt = i
            End If
        Next i
        ' Запис результату
        Result (iMin)
    End Sub
```

Критерій компромісу за Гурвіцем

Якщо ОПР важко визначити ступінь упевненості λ, але бажано знайти компроміс між оптимістичним та песимістичним рішеннями, можна скористатися формулою

$$K_8 = \max_i \left\{ \frac{\max_j q_{ij} + \min_j q_{ij}}{2} \right\}. \quad (11)$$

Приклад застосування критерію наведено в табл.14.

Таблиця 14

Приклад застосування критерію компромісу за Гурвіцем

A	S_1	S_2	S_3	S_4	max	min	K_8
A_1	14	9	12	3	14	3	8,5
A_2	8	15	6	15	15	6	10,5
A_3	2	11	7	19	19	2	10,5
A_4	7	13	4	13	13	4	8,5
						max:	10,5

У даному випадку дві альтернативи одночасно задовольняють умови критерію, тому подальший вибір однієї з них покладається на ОПР.

Процедура, яка реалізує критерій ризику за Гурвіцем, наведена у лістингу 8.

```
' Лістинг 8
'
Dim nAlt, iAlt() As Integer
'
' Процедура визначення кількох результатів
' Автор А.Панчук
' Дата створення 27.06.1999
'
Private Sub mResult(rValue As Single)
    Dim rRange As Range
    Dim i As Integer
```

```
    ' Запис результату
    Cells(rRow, rCol) = rValue
    ' Визначення альтернатив
    iAlt(1) = rRow - nRows + iAlt(1) - 1
    Set rRange = Range(Cells(iAlt(1), rCol - nCols), Cells(iAlt(1),
      rCol))
    For i = 2 To nAlt
        iAlt(i) = rRow - nRows + iAlt(i) - 1
        Set rRange = Union(rRange, Range(Cells(iAlt(i), rCol -
          nCols), _Cells(iAlt(i), rCol)))
    Next i
    rRange.Activate
End Sub
'
' Програма застосування критерію компромісу за Гурвіцем
' Автор А.Панчук
' Дата створення 27.06.1999
'
Sub ComprGainHurwicz()
    Dim iMax, jMax, jMin As Single
    Dim Aver() As Single
    ' Виклик допоміжної процедури Define
    Define
    ' Обчислення критерію компромісу за Гурвіцем
    iMax = 0
    ReDim Aver(nRows)
    For i = 1 To nRows
        jMax = 0
        jMin = 99999 ' Умовне максимальне значення виграшу
        For j = 1 To nCols
            If jMax < mArray(i, j) Then
                jMax = mArray(i, j)
            End If
            If jMin > mArray(i, j) Then
                jMin = mArray(i, j)
```

```
            End If
        Next j
        Aver(i) = (jMax + jMin) / 2#
        If iMax < Aver(i) Then
            iMax = Aver(i)
            rAlt = i
        End If
    Next i
    nAlt = 1
    ' Визначення кількох однакових розв'язків
    For i = rAlt + 1 To nRows
        If iMax = Aver(i) Then
            nAlt = nAlt + 1
        End If
    Next i
    If nAlt = 1 Then
        ' Запис результату
        Result (iMax)
    Else
        ' Запис кількох результатів
        ReDim iAlt(nAlt)
        i = 0
        For j = 1 To nRows
            If Aver(j) = iMax Then
                i = i + 1
                iAlt(i) = j
            End If
        Next j
        mResult (iMax)
    End If
End Sub
```

У результаті розв'язок задачі згідно з лістингом 8 набере вигляду, як показано на рис. 11.

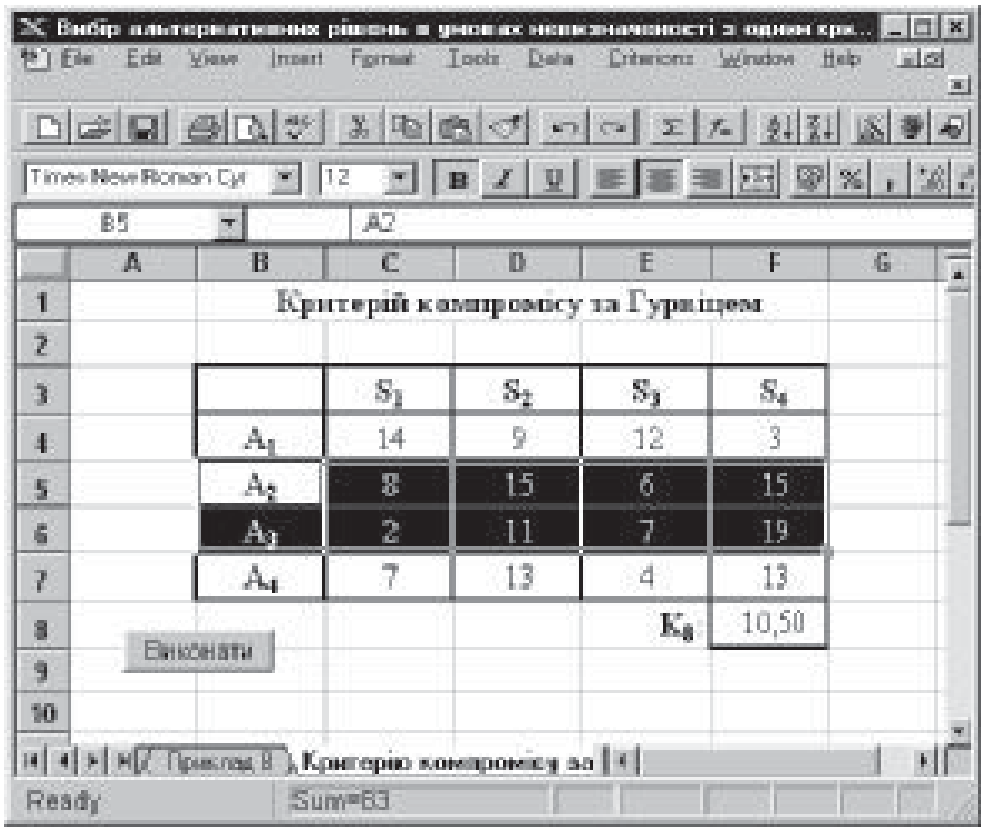


Рис. 11. ***Критерій компромісу за Гурвіцем***

Зверніть увагу на те, що до цієї програми включено визначення глобальних змінних та допоміжної процедури mResults, яка знаходить та виділяє на робочому листку всі відповідні альтернативи.

Критерій компромісу згідно з ризиком

Цей критерій аналогічний попередньому, тільки вибір грунтується не на виграшах, а на ризиках. Звичайно, що ризики необхідно мінімізувати. Тому критерій може бути визначений так:

$$K_9 = \min_i \left\{ \frac{\min_j r_{ij} + \max_j r_{ij}}{2} \right\}. \quad (12)$$

Приклад застосування критерію наведено в табл.15.

Таблиця 15

Матриця ризиків та критерій компромісу згідно з ризиком

A	S_1	S_2	S_3	S_4	min	max	K_9
A_1	0	6	0	16	0	16	8
A_2	6	0	6	4	0	6	3
A_3	12	4	5	0	0	12	6
A_4	7	2	8	6	2	8	5
						min:	3,0

Процедура, яка реалізує критерій ризику за Гурвіцем, наведена в лістингу 9.

```
' Лістинг 9
'
' Програма застосування критерію компромісу згідно з
    ризиком за Гурвіцем
' Автор А.Панчук
' Дата створення 27.06.1999
'
Sub ComprRiskHurwicz()
    Dim iMin, jMin, jMax As Single
    Dim rArray() As Single
    Dim qMax As Single
    Dim Aver() As Single
    ' Виклик допоміжної процедури Define
    Define
    ' Визначення матриці ризику
    ReDim rArray(nRows, nCols)
    For j = 1 To nCols
        qMax = 0
        For i = 1 To nRows
            If qMax < mArray(i, j) Then
                qMax = mArray(i, j)
            End If
        Next i
        For i = 1 To nRows
            rArray(i, j) = qMax - mArray(i, j)
```

```
    Next i
Next j
' Обчислення критерію компромісу згідно з ризиком
  за Гурвіцем
ReDim Aver(nRows)
iMin = 99999 " Умовне максимальне значення
For i = 1 To nRows
    jMin = 99999 " Умовне максимальне значення
    jMax = 0
    For j = 1 To nCols
        If jMin > rArray(i, j) Then
            jMin = rArray(i, j)
        End If
        If jMax < rArray(i, j) Then
            jMax = rArray(i, j)
        End If
    Next j
    Aver(i) = (jMax + jMin) / 2#
    If iMin > Aver(i) Then
        iMin = Aver(i)
        rAlt = i
    End If
Next i
nAlt = 1
' Визначення кількох однакових розв'язків
For i = rAlt + 1 To nRows
    If iMin = Aver(i) Then
        nAlt = nAlt + 1
    End If
Next i
If nAlt = 1 Then
    ' Запис результату
    Result (iMin)
Else
    ' Запис кількох результатів
    ReDim iAlt(nAlt)
```

```
        i = 0
        For j = 1 To nRows
            If Aver(j) = iMin Then
                i = i + 1
                iAlt(i) = j
            End If
        Next j
        mResult (iMin)
    End If
End Sub
```

Критерій Ходжеса - Лемана

Критерій Ходжеса - Лемана використовує фактори, які можна вважати суб'єктивними: по-перше, це ймовірність станів природи, а, по-друге, це ступінь упевненості ОПР, з якою ми вже стикалися у критеріях за Гурвіцем (див. (9)).

Цей критерій може бути визначено за формулою

$$K_{10} = \max_i \left\{ \lambda \sum_{j=1}^{n} p_{ij}\ q_{ij} + (1- \lambda)\ \min\ q_{ij} \right\} . \quad (13)$$

Приклад застосування критерію Ходжеса - Лемана наведено в табл. 16.

Таблиця 16

Варіанти розв'язки згідно з критерієм Ходжеса - Лемана

A	S_1		S_2		S_3		S_4		Sum	min	λ		
	q	pq	q	pq	q	pq	q	pq			0,4	0,6	0,8
A_1	14	2,1	9	3,6	12	2,4	3	0,75	8,85	3	5,34	6,51	7,68
A_2	8	1,2	15	6,0	6	1,2	15	3,75	12,2	6	8,46	9,69	10,92
A_3	2	0,3	11	4,4	7	1,4	19	4,75	10,9	2	5,54	7,31	9,08
A_4	7	1,05	13	5,2	4	0,8	13	3,25	10,3	4	6,52	7,78	9,04
p	0,15		0,4		0,2		0,25			max:	8,46	9,69	10,92

Процедура, яка реалізує критерій ризику за Гурвіцем, наведена у лістингу 10.

```
' Лістинг 10
'
' Програма застосування критерію Ходжеса - Лемана
' Автор А.Панчук
' Дата створення 27.06.1999
'
Sub HodgesLehman()
    Dim iMax, jSum, jMin As Single
    Dim pqArray() As Single
    Dim Lambda, Expect As Single
    ' Виклик допоміжної процедури Define
    Define
    ' Перевірка коректності умов для ймовірностей
    jSum = 0
    For j = 1 To nCols
        If (mArray(nRows, j) < 0#) Or (mArray(nRows, j) > 1#)
           Then
            Rsp = MsgBox("Значення ймовірності не може
               бути < 0 або > 1.", _ vbCritical, "Помилка у даних")
            Exit Sub
        End If
        jSum = jSum + mArray(nRows, j)
    Next j
    If jSum <> 1# Then
        Rsp = MsgBox("Сума ймовірностей має дорівнювати
           одиниці.", vbCritical, _ "Помилка у даних")
        Exit Sub
    End If
    ' Введення Lambda
    Lambda = CSng(InputBox("Введіть значення коефіцієнта
       впевненості:", _ "Критерій виграшу за Гурвіцем"))
    If (Lambda < 0#) Or (Lambda > 1#) Then
        Rsp = MsgBox("Значення коефіцієнта впевненості не
           може бути < 0 "_ & "або > 1.", vbCritical, "Помилка
              в даних")
        Exit Sub
```

```
        End If
        ' Обчислення критерію Ходжеса - Лемана
        ReDim pqArray(nRows - 1, nCols)
        iMax = 0
        For i = 1 To nRows - 1
            jSum = 0
            jMin = 99999 " Умовне максимальне значення
            For j = 1 To nCols
                pqArray(i, j) = mArray(i, j) * mArray(nRows, j)
                jSum = jSum + pqArray(i, j)
                If jMin > mArray(i, j) Then
                    jMin = mArray(i, j)
                End If
            Next j
            Expect = Lambda * jSum + (1 - Lambda) * jMin
            If iMax < Expect Then
                iMax = Expect
                rAlt = i
            End If
        Next i
        ' Запис результату
        Result (iMax)
    End Sub
```

Критерій Ходжеса - Лемана вважається складним. Головним його недоліком є використання великої кількості суб'єктивних факторів.

Критерій крайнього оптимізму

Неважко бачити, що критерій виграшу за Гурвіцем за умови $\lambda = 1$ вироджується у формулу виду

$$K_{11} = \max_i \max_j q_{ij}, \qquad (14)$$

яка визначає критерій крайнього оптимізму.

Приклад застосування цього критерію наведено у табл. 17.

Таблиця 17

Приклад застосування критерію крайнього оптимізму

A	S_1	S_2	S_3	S_4	max
A_1	14	9	12	3	14
A_2	8	15	6	15	15
A_3	2	11	7	19	19
A_4	7	13	4	13	13
				max:	19

Процедура, яка реалізує критерій крайнього оптимізму, наведена у лістингу 11.

```
' Лістинг 11
'
' Програма застосування критерію крайнього оптимізму
' Автор А.Панчук
' Дата створення 28.06.1999
'
Sub Optimism()
    Dim iMax, jMax As Single
    ' Виклик допоміжної процедури Define
    Define
    ' Обчислення критерію крайнього оптимізму
    iMax = 0
    For i = 1 To nRows
        jMax = 0
        For j = 1 To nCols
            If jMax < mArray(i, j) Then
                jMax = mArray(i, j)
            End If
        Next j
        If iMax < jMax Then
            iMax = jMax
            rAlt = i
        End If
    Next i
    ' Запис результату
    Result (iMax)
End Sub
```

Критерій “боягуза”

Критерій “боягуза” є, так би мовити, повною протилежністю критерію крайнього оптимізму і визначається за формулою:

$$K_{12} = \min_i \min_j q_{ij} . \qquad (15)$$

Приклад його застосування наведено у табл.18.

Таблиця 18

Приклад застосування критерію “боягуза”

A	S_1	S_2	S_3	S_4	min
A_1	14	9	12	3	3
A_2	8	15	6	15	6
A_3	2	11	7	19	2
A_4	7	13	4	13	4
				min:	2

Процедура, яка реалізує критерій “боягуза”, наведена у лістингу 12.

```
' Лістинг 12
'
' Програма застосування критерію боягуза
' Автор А.Панчук
' Дата створення 28.06.1999
'
Sub Coward()
    Dim iMin, jMin As Single
    ' Виклик допоміжної процедури Define
    Define
    ' Обчислення критерію боягуза
    iMin = 99999 ' Умовне максимальне значення
    For i = 1 To nRows
        jMin = 99999 ' Умовне максимальне значення
        For j = 1 To nCols
            If jMin > mArray(i, j) Then
                jMin = mArray(i, j)
```

```
            End If
        Next j
        If iMin > jMin Then
            iMin = jMin
            rAlt = i
        End If
    Next i
    ' Запис результату
    Result (iMin)
End Sub
```

Критерій обережності згідно з ризиком

Цей критерій на перший погляд схожий на попередній: критерій "боягуза". Але рішення приймається з урахуванням розміру ризику, тому він видається більш раціональним. Критерій обережності згідно з ризиком визначається за формулою

$$K_{13} = \min_i \min_j r_{ij} . \qquad (16)$$

Приклад його застосування наведено в табл.19.

Таблиця 19

Матриця ризиків та критерій обережності згідно з ризиком

A	S_1	S_2	S_3	S_4	min
A_1	0	6	0	16	0
A_2	6	0	6	4	0
A_3	12	4	5	0	0
A_4	7	2	8	6	2
				min:	0

Легко передбачити, що в більшості ситуацій цей критерій визначає кілька альтернатив (за кількістю рядків, які містять нульові значення ризику).

Процедура, яка реалізує критерій обережності згідно з ризиком, наведена у лістингу 13.

```
' Лістинг 13
'
Dim rArray() As Single
Dim qMax As Single
'
' Процедура визначення матриці ризику
' Автор А.Панчук
' Дата створення 03.07.1999
'
Private Sub Risk()
    ' Визначення матриці ризику
    ReDim rArray(nRows, nCols)
    For j = 1 To nCols
        qMax = 0
        For i = 1 To nRows
            If qMax < mArray(i, j) Then
                qMax = mArray(i, j)
            End If
        Next i
        For i = 1 To nRows
            rArray(i, j) = qMax - mArray(i, j)
        Next i
    Next j
End Sub
'
' Програма застосування критерію обережності згідно з
  ризиком
' Автор А.Панчук
' Дата створення 03.07.1999
'
Sub Caution()
    Dim iMin, jMin() As Single
    ' Виклик допоміжної процедури Define
    Define
    ' Визначення матриці ризику
    Risk
    ' Обчислення критерію обережності згідно з ризиком
```

```
    ReDim jMin(nRows)
    iMin = 99999 ' Умовне максимальне значення
    For i = 1 To nRows
        jMin(i) = 99999 ' Умовне максимальне значення
        For j = 1 To nCols
            If jMin(i) > rArray(i, j) Then
                jMin(i) = rArray(i, j)
            End If
        Next j
        If iMin > jMin(i) Then
            iMin = jMin(i)
            rAlt = i
        End If
    Next i
    ' Визначення кількох однакових розв'язків
    nAlt = 1
    For i = rAlt + 1 To nRows
        If iMin = jMin(i) Then
            nAlt = nAlt + 1
        End If
    Next i
    If nAlt = 1 Then
        ' Запис результату
        Result (iMin)
    Else
        ' Запис кількох результатів
        ReDim iAlt(nAlt)
        i = 0
        For j = 1 To nRows
            If jMin(j) = iMin Then
                i = i + 1
                iAlt(i) = j
            End If
        Next j
        mResult (iMin)
    End If
End Sub
```

Критерій Гермеєра

Критерій Гермеєра нагадує критерій Вальда, але додатково враховує ймовірності станів природи. Він може бути визначений за формулою

$$K_{14} = \max_i \min_j q_{ij} p_j . \qquad (17)$$

Приклад його застосування наведено у табл.20.

Таблиця 20

Приклад застосування критерію Гермеєра

A	S_1		S_2		S_3		S_4		min
	q	pq	q	pq	q	pq	q	pq	
A_1	14	2,1	9	3,6	12	2,4	3	0,75	0,75
A_2	8	1,2	15	6,0	6	1,2	15	3,75	1,20
A_3	2	0,3	11	4,4	7	1,4	19	4,75	0,30
A_4	7	1,05	13	5,2	4	0,8	13	3,25	0,80
p	0,15		0,4		0,2		0,25		1,0
								max:	1,20

Процедура, яка реалізує критерій Гермеєра, наведена у лістингу 14.

```
' Лістинг 14
'
Dim jSum As Single
'
' Функція перевірки умов для ймовірностей
' Автор А.Панчук
' Дата створення 03.07.1999
'
Private Function Probability() As Boolean
    ' Перевірка коректності умов для ймовірностей
    Probability = True
```

```
    jSum = 0
    For j = 1 To nCols
        If (mArray(nRows, j) < 0#) Or (mArray(nRows, j) > 1#)
           Then
            Rsp = MsgBox("Значення ймовірності не може
               бути < 0 або > 1.", _vbCritical, "Помилка у даних")
            Probability = False
            Exit Function
        End If
        jSum = jSum + mArray(nRows, j)
    Next j
    If jSum <> 1# Then
        Rsp = MsgBox("Сума ймовірностей має дорівнювати
           одиниці.", vbCritical, _"Помилка у даних")
        Probability = False
    End If
End Function
'
' Програма застосування критерію Гермеєра
' Автор А.Панчук
' Дата створення 03.07.1999
'
Sub Germayer()
    Dim iMax, jMin As Single
    Dim pqArray() As Single
    ' Визначення виділеної області клітинок
    Define
    ' Перевірка коректності умов для ймовірностей
    If Not (Probability) Then
        Exit Sub
    End If
    ' Обчислення критерію Гермеєра
    ReDim pqArray(nRows - 1, nCols)
    iMax = 0
    For i = 1 To nRows - 1
        jMin = 99999 ' Умовне максимальне значення виграшу
```

```
            For j = 1 To nCols
                pqArray(i, j) = mArray(i, j) * mArray(nRows, j)
                If jMin > pqArray(i, j) Then
                    jMin = pqArray(i, j)
                End If
            Next j
            If iMax < jMin Then
                iMax = jMin
                rAlt = i
            End If
        Next i
        ' Запис результату
        Result (iMax)
    End Sub
```

Агрегування критеріїв

Коли вирішуються реальні завдання, ОПР альтернативи оцінює за багатьма критеріями. Разом з тим для спрощення проблеми вибору бажано зменшувати кількість критеріїв, тобто якось їх **агрегувати**. У подальшому окремі критерії позначатимемо Q_j, а агрегований єдиний критерій - Q_a.

Треба розрізняти дві проблеми.

Перша пов'язана з необхідністю враховувати випадкові величини: у цьому разі і сам показник ефективності також буде випадковою величиною. У подібній ситуації фахівці рекомендують перетворити ймовірнісну задачу на детерміновану шляхом заміни випадкових факторів на їх математичні сподівання (такий прийом називають "оптимізацією у середньому").

Друга проблема пов'язана з тим, що тільки у виняткових ситуаціях ОПР вдається одночасно наблизитись до всіх окремих цілей, оскільки критерії можуть бути взаємовиключними. Зведення багатокритеріальної задачі до однокритеріальної в такому випадку реалізується введенням єдиної функції (наприклад, у вигляді відношення, чисельником якого

є критерії, які бажано збільшити, а знаменником - ті, які бажано зменшити).

Слід зважати на те, що розмірність такого складеного критерію найчастіше не матиме реального фізичного змісту. Дуже часто помилково вважають, що брак одного із складових критеріїв буде компенсовано за рахунок іншого (наприклад, низька продуктивність - за рахунок меншої вартості). Як правило, це не так.

Найпоширенішими алгоритмами "згортки" критеріїв є адитивний (так звана **зважена сума**) та мультиплікативний (який грунтується на перемножуванні показників). Інші методи спираються переважно на два принципи оптимізації, які визначають кращими альтернативи:

- перший - ту, яка є ближчою до певної (так званої "ідеальної", або в якийсь інший спосіб вибраної) альтернативи; у цьому випадку необхідно визначити поняття **відстані** між альтернативами (тобто встановити метрику в просторі критеріїв);

- другий - ту, яка переважає інші в більшості окремих критеріїв.

Розглянемо більш докладно деякі алгоритми агрегування критеріїв.

Адитивний критерій

Вибір альтернативи за адитивним критерієм можна здійснити за формулою

$$A_1 = \max_i \sum_{j=1}^{n} \pm w_{ij}\, q_{ij} \ , \qquad (18)$$

за умови що $\sum_{j=1}^{n} w_j = 1$,

де w_j - вага відповідного критерію, а знак "+" чи "-" вибирається залежно від того, чого необхідно досягти: **максимізації** чи **мінімізації** окремого критерію.

Найчастіше під значеннями q_{ij} розуміють не абсолютні величини, які мають конкретні розмірності і можуть бути несумісними щодо окремих критеріїв, а безрозмірні (відносні) величини, які є результатом нормалізації - ділення абсолютних значень критерію на максимальне з них:

$$q_{ij}^{\text{norm}} = \frac{q_{ij}}{q_{ij}^{\max}} \quad . \qquad (19)$$

Приклад застосування адитивного критерію наведено у табл.21.

Таблиця 21

Приклад застосування адитивного критерію

A	Q_1			Q_2			Q_3			Q_a
	q	q^{norm}	dwq	q	q^{norm}	dwq	q	q^{norm}	dwq	
A_1	100,0	1,00	0,30	3,0	0,43	0,13	0,15	0,75	-0,30	0,129
A_2	80,0	0,80	0,24	7,0	1,00	0,30	0,09	0,45	-0,18	0,360
A_3	80,0	0,80	0,24	5,0	0,71	0,21	0,03	0,15	-0,06	0,394
A_4	60,0	0,60	0,18	3,0	0,43	0,13	0,20	1,00	-0,40	-0,091
w	0,3			0,3			0,4			1,0
d	max			max			min			
									max:	0,394

Процедура, яка реалізує адитивний критерій, наведена у лістингу 15.

```
' Лістинг 15
'
Dim nRows, nCols As Integer
Dim rRow, rCol, rAlt As Integer
Dim mArray() As Single
Dim dArray() As String
Dim qArray() As Single
Dim qFirst, qLast As Integer
Dim jDir() As Single
Dim jSum As Single
Dim qMax, qOpt As Single
Dim nAlt, iAlt() As Integer
```

```
Dim i, j As Integer
'
' Процедура визначення виділеної області клітинок
' Автор А.Панчук
' Дата створення 14.08.1999
'
Private Sub Define()
    ' Визначення виділеної області клітинок
    nRows = Selection.Rows.Count
    nCols = Selection.Columns.Count
    ReDim mArray(nRows - 1, nCols)
    ' Агрегований критерій та його позиція в робочому листку
    ReDim qArray(nRows - 1)
    qFirst = Selection.Row
    qLast = qFirst + nRows - 2
    ' Позиція результату
    rRow = Selection.Row + nRows
    rCol = Selection.Column + nCols - 1
    ' Копіювання вмісту клітин у масив
    For i = 1 To nRows - 1
        For j = 1 To nCols
            mArray(i, j) = Selection.Cells(i, j).Value
        Next j
    Next i
    ReDim dArray(nCols)
    ReDim jDir(nCols)
    For j = 1 To nCols
        dArray(j) = Selection.Cells(nRows, j).Value
        If dArray(j) = "max" Then
            jDir(j) = 1#
        Else
            jDir(j) = -1#
        End If
    Next j
End Sub
'
' Функція перевірки коректності умов для вагових коефіцієнтів
```

```
' Автор А.Панчук
' Дата створення 14.08.1999
'
Private Function Weight() As Boolean
    ' Перевірка коректності вагових коефіцієнтів
    Weight = True
    jSum = 0
    For j = 1 To nCols
        If (mArray(nRows - 1, j) < 0#) Or (mArray(nRows - 1, j) >
           1#) Then
            Rsp = MsgBox("Значення коефіцієнта не може
               бути < 0 або > 1.", _vbCritical, "Помилка в даних")
            Weight = False
            Exit Function
        End If
        jSum = jSum + mArray(nRows - 1, j)
    Next j
    If jSum <> 1# Then
        Rsp = MsgBox("Сума вагових коефіцієнтів має
           дорівнюватиодиниці.", _vbCritical, "Помилка в даних")
        Weight = False
    End If
    qArray(nRows - 1) = jSum
End Function
'
' Процедура визначення результату
' Автор А.Панчук
' Дата створення 14.08.1999
'
Private Sub Result(rValue As Single)
    ' Запис результату
    For i = 1 To nRows - 1
        Cells(qFirst + i - 1, rCol) = qArray(i)
    Next i
    Cells(rRow, rCol) = rValue
    ' Визначення альтернативи
```

```
    rAlt = rRow - nRows + rAlt - 1
    Range(Cells(rAlt, rCol - nCols), Cells(rAlt, rCol)).Select
End Sub
'
' Програма застосування адитивного критерію
' Автор А.Панчук
' Дата створення 06.07.1999
'
Sub Additive()
    Dim iMax As Single
    Dim dwqArray() As Single
    ' Визначення виділеної області клітинок
    Define
    ' Перевірка коректності вагових коефіцієнтів
    If Not (Weight) Then
        Exit Sub
    End If
    ' Обчислення адитивного критерію
    ReDim dwqArray(nRows - 2, nCols)
    iMax = -99999# ' Умовне мінімальне значення
    For i = 1 To nRows - 2
        jSum = 0
        For j = 1 To nCols
            dwqArray(i, j) = mArray(i, j) * mArray(nRows - 1, j) *
              jDir(j)
            jSum = jSum + dwqArray(i, j)
        Next j
        If iMax < jSum Then
            iMax = jSum
            rAlt = i
        End If
        qArray(i) = jSum
    Next i
    ' Запис результату
    Result (iMax)
End Sub
```

На рис.12 наведено результат застосування адитивного критерію без попередньої нормалізації значень. Для запуску програми необхідно виділити клітини, що містять початкові дані, включаючи вагові коефіцієнти та рядок напрямів оптимізації. Остання колонка таблиці Q_a заповнюється автоматично. Шукана альтернатива виділяється.

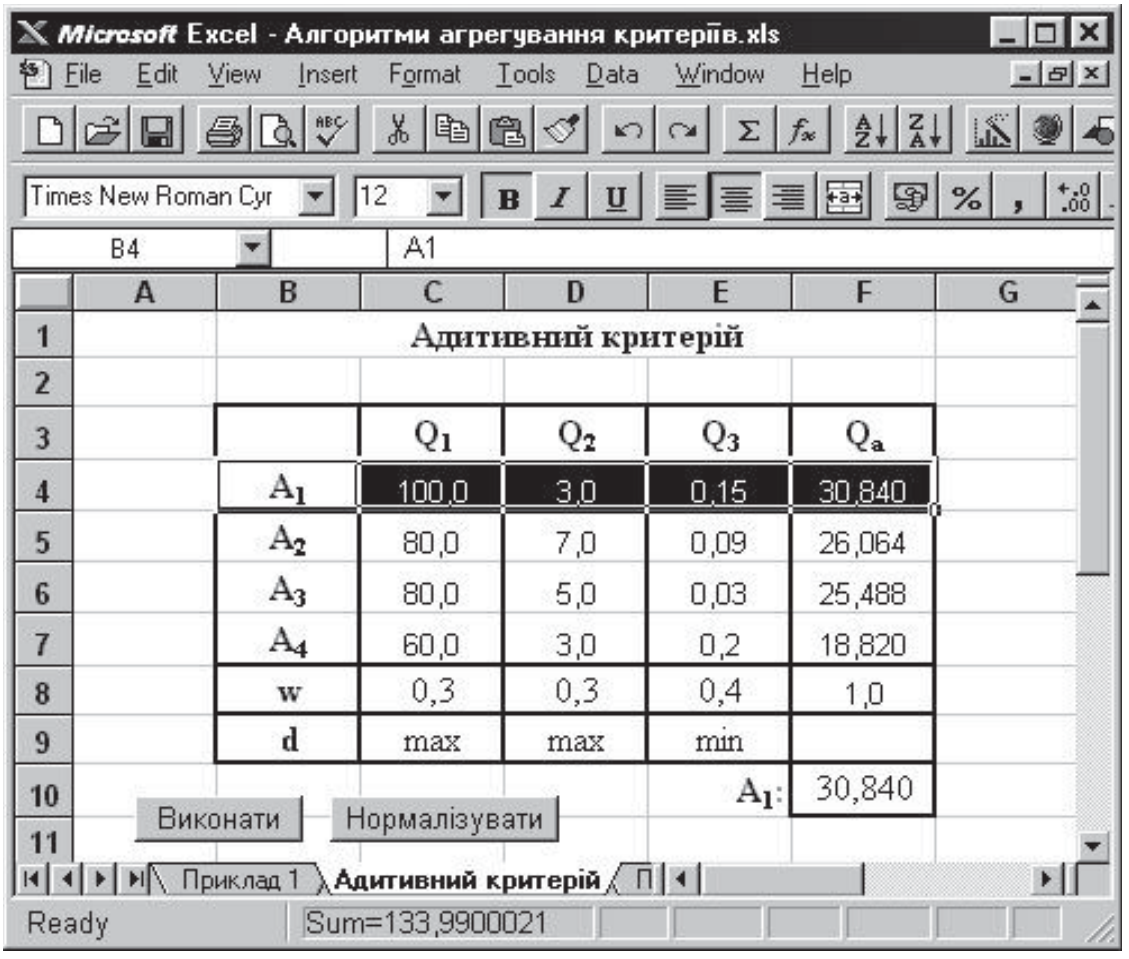
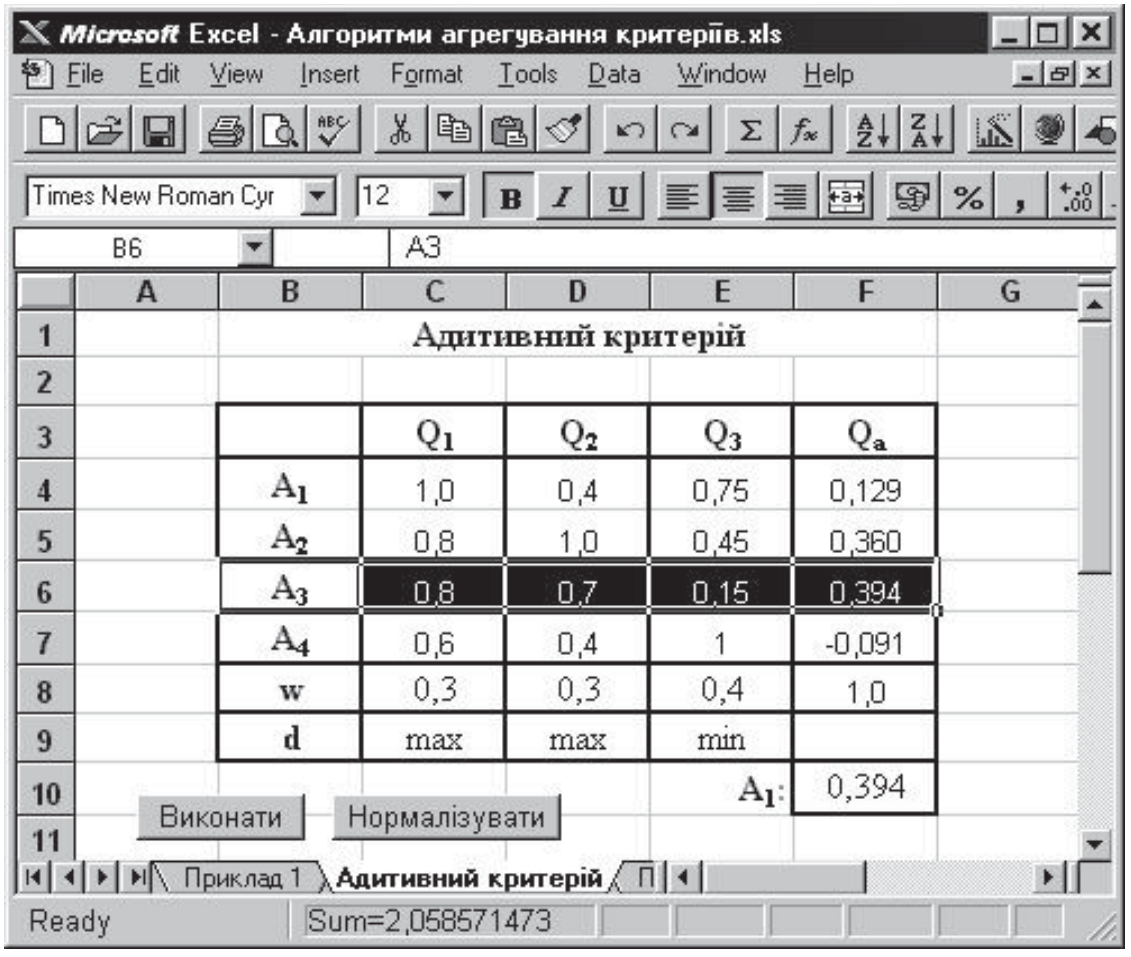


Рис. 12. ***Адитивний критерій***

Кнопка **Нормалізувати** дає змогу автоматично виконати нормалізацію, після якої результат застосування адитивного критерію буде іншим (рис. 13). Зауважимо, що для виконання нормалізації необхідно виділити клітини, які містять тільки дані по альтернативах (не враховуючи вагових коефіцієнтів та рядку напрямів оптимізації).



		Q_1	Q_2	Q_3	Q_a
A_1		1,0	0,4	0,75	0,129
A_2		0,8	1,0	0,45	0,360
A_3		0,8	0,7	0,15	0,394
A_4		0,6	0,4	1	-0,091
w		0,3	0,3	0,4	1,0
d		max	max	min	
				A_1:	0,394

Рис. 13. ***Адитивний критерій після нормалізації***

Програму нормалізації критеріїв наведено у лістингу 16.

```
' Лістинг 16
'
' Програма нормалізації критеріїв

' Автор А.Панчук
' Дата створення 04.08.1999
'
Sub Normal()
    ' Визначення виділеної області клітинок
    nRows = Selection.Rows.Count
    nCols = Selection.Columns.Count
    ReDim mArray(nRows, nCols)
    ' Копіювання вмісту клітин у масив
    For i = 1 To nRows
```

```
        For j = 1 To nCols
            mArray(i, j) = Selection.Cells(i, j).Value
        Next j
    Next i
    ' Нормалізація критеріїв
    For j = 1 To nCols
        qMax = -99999# ' Умовне мінімальне значення
        For i = 1 To nRows
            If qMax < mArray(i, j) Then
                qMax = mArray(i, j)
            End If
        Next i
        For i = 1 To nRows
            mArray(i, j) = mArray(i, j) / qMax
        Next i
    Next j
    ' Повернення нормалізованих значень у виділену область
    For i = 1 To nRows
        For j = 1 To nCols
            Selection.Cells(i, j).Value = mArray(i, j)
        Next j
    Next i
End Sub
```

Мультиплікативний критерій

Мультиплікативний критерій відрізняється від адитивного тим, що замість суми використовується добуток (значення перемножуються). Формула мультиплікативного критерію має такий вигляд:

$$A_2 = \max_i \prod_{j=1}^{n} q_{ij}^{\pm w_j}, \qquad (20)$$

за умови що $\sum_{j=1}^{n} w_j = 1$.

З точки зору обчислень між формулами (18) та (20) немає суттєвої різниці, оскільки логарифмуючи формулу (20),

дістанемо (18). Але важливою властивістю мультиплікативного критерію є те, що при його застосуванні немає потреби спеціально нормалізувати величини.

Приклад застосування адитивного критерію наведено у табл. 22.

Таблиця 22

Приклад застосування мультиплікативного критерію

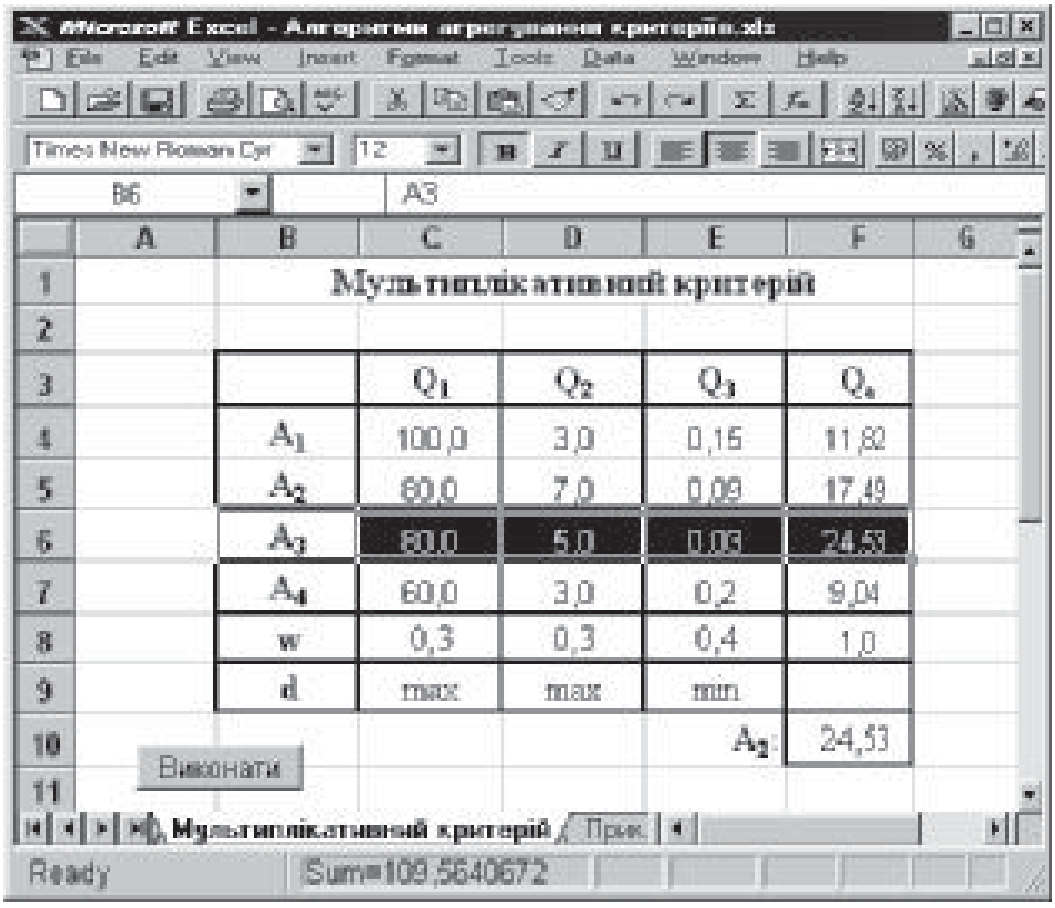
A	Q_1		Q_2		Q_3		Q_a
	q	q^{dw}	q	q^{dw}	q	q^{dw}	
A_1	100,0	3,981	3,0	1,39	0,15	2,136	11,82
A_2	80,0	3,723	7,0	1,793	0,09	2,620	17,49
A_3	80,0	3,723	5,0	1,621	0,03	4,066	24,53
A_4	60,0	3,415	3,0	1,390	0,20	1,904	9,04
w	0,3		0,3		0,4		1,0
d	max		max		min		
						max:	24,53

Процедура, яка реалізує адитивний критерій, наведена у лістингу 17.

```
' Лістинг 17
'
' Програма застосування мультиплікативного критерію
' Автор А.Панчук
' Дата створення 14.08.1999
'
Sub Multiplic()
    Dim iMax, jPro As Single
    Dim qdwArray() As Single
    ' Визначення виділеної області клітинок
    Define
    ' Перевірка коректності вагових коефіцієнтів
    If Not (Weight) Then
        Exit Sub
    End If
    ' Обчислення мультиплікативного критерію
    ReDim qdwArray(nRows - 2, nCols)
    iMax = -99999# ' Умовне мінімальне значення
    For i = 1 To nRows - 2
```

```
        jPro = 1#
    For j = 1 To nCols
        qdwArray(i, j) = Exp(Log(mArray(i, j)) * mArray(nRows -1, j) *
          jDir(j))
        jPro = jPro * qdwArray(i, j)
        Next j
        If iMax < jPro Then
            iMax = jPro
            rAlt = i
        End If
        qArray(i) = jPro
    Next i
    ' Запис результату
    Result (iMax)
End Sub
```

Розв'язок задачі згідно з лістингом 17 має вигляд, наведений на рис. 14. (Зверніть увагу на те, що вибрана та сама **третя** альтернатива, що і у випадку застосування адитивного критерію за умови нормалізації даних).



		Q_1	Q_2	Q_3	Q_4
A_1		100,0	3,0	0,15	11,82
A_2		60,0	7,0	0,09	17,49
A_3		80,0	5,0	0,09	24,53
A_4		60,0	3,0	0,2	9,04
w		0,3	0,3	0,4	1,0
d		max	max	min	
				A_3:	24,53

Рис. 14. ***Мультиплікативний критерій***

Перетворимо початкові дані так: для критеріїв, які мають бути мінімізовані, помножимо всі значення на -1. Тепер напрям оптимізації для всіх окремих критеріїв буде однаковим - **максимізувати**. На таких даних дуже просто вибрати так звану **ідеальну альтернативу**[1], для якої справедлива формула

$$q_{ij}^{\text{ideal}} = \max_j q_{ij} . \qquad (21)$$

Знайдемо відхилення кожної з альтернатив від ідеальної:

$$\Delta_{ij} = q_{ij}^{\text{ideal}} - q_{ij} . \qquad (22)$$

Тоді метод мінімаксу може бути задано формулою

$$A_3 = \min_i \max_j \Delta_{ij} . \qquad (23)$$

Приклад застосування методу мінімаксу наведено у табл. 23.

Таблиця 23

Приклад застосування методу мінімаксу

A	Початкова			Перетворена			Відхилення			Q_a (max)
	Q_1	Q_2	Q_3	Q_1	Q_2	Q_3	Δ_1	Δ_2	Δ_3	
A_1	100,0	3,0	0,15	100,0	3,0	-0,15	0,000	0,571	0,600	0,600
A_2	80,0	7,0	0,09	80,0	7,0	-0,09	0,200	0,000	0,300	0,300
A_3	80,0	5,0	0,03	80,0	5,0	-0,03	0,200	0,286	0,000	0,286
A_4	60,0	3,0	0,20	60,0	3,0	-0,20	0,400	0,571	0,850	0,850
d	max	max	min	100,0	7,0	-0,03				
				Ідеальна альтернатива					min:	0,286

Процедура, яка реалізує метод мінімаксу, наведена у лістингу 18.

```
' Лістинг 18
'
' Програма застосування методу мінімаксу
```

[1] Як правило, ідеальна альтернатива не входить до множини альтернатив, що досліджуються.

```
' Автор А.Панчук
' Дата створення 14.08.1999
'
Sub MiniMax()
    Dim iMin, jMax As Single
    Dim dAlt() As Single
    ' Визначення виділеної області клітинок
    Define
    ReDim dAlt(nRows - 1, nCols)
    ' Перетворення матриці
    For i = 1 To nRows - 1
        For j = 1 To nCols
            mArray(i, j) = mArray(i, j) * jDir(j)
        Next j
    Next i
    ' Визначення матриці відхилень
    For j = 1 To nCols
        qMax = -99999# ' Умовне мінімальне значення
        For i = 1 To nRows - 1
            If qMax < mArray(i, j) Then
                qMax = mArray(i, j)
            End If
        Next i
        For i = 1 To nRows - 1
            dAlt(i, j) = qMax - mArray(i, j)
        Next i
    Next j
    ' Обчислення за методом мінімаксу
    iMin = 99999# ' Умовне максимальне значення
    For i = 1 To nRows - 1
        jMax = -99999# ' Умовне мінімальне значення
        For j = 1 To nCols
            If jMax < dAlt(i, j) Then
                jMax = dAlt(i, j)
            End If
        Next j
```

```
            If iMin > jMax Then
                iMin = jMax
                rAlt = i
            End If
            qArray(i) = jMax
        Next i
        ' Запис результату
        Result (iMin)
    End Sub
```

Метод мінімальної відстані

Можна визначити відстань між двома альтернативами *s* та *t* у звичайному евклідовому розумінні:

$$d_{st} = \sqrt{\sum_{j=1}^{n} (q_{sj} - q_{tj})^2} \ . \qquad (24)$$

Тоді метод мінімальної відстані може бути визначено за формулою

$$A_4 = \min_i d_{it} \ , \qquad (25)$$

де *t* позначає, наприклад, ідеальну альтернативу.

Приклад застосування методу мінімальної відстані наведено в табл. 24.

Таблиця 24

Приклад застосування методу мінімальної відстані

A	Q_1			Q_2			Q_3			Q_a
	q	q^{norm}	d	q	q^{norm}	d	q	q^{norm}	d	
A_1	100,0	1,0	0,0	3,0	0,43	0,57	0,15	0,75	0,60	0,829
A_2	80,0	0,8	0,2	7,0	1,00	0,00	0,09	0,45	0,30	0,361
A_3	80,0	0,8	0,2	5,0	0,71	0,29	0,03	0,15	0,00	0,349
A_4	60,0	0,6	0,4	3,0	0,43	0,57	0,20	1,00	0,85	1,100
d	max			max			min			
									min:	0,349

Процедура, яка реалізує метод мінімальної відстані, наведена у лістингу 19.

```
' Лістинг 19
'
' Програма застосування методу мінімальної відстані
' Автор А.Панчук
' Дата створення 14.08.1999
'
Sub MiniDist()
    Dim iMin As Single
    Dim idAlt(), dAlt As Single
    ' Визначення виділеної області клітинок
    Define
    ReDim idAlt(nCols)
    ' Визначення ідеальної альтернативи
    For j = 1 To nCols
        If dArray(j) = "max" Then
            qOpt = -99999# ' Умовне мінімальне значення
            For i = 1 To nRows - 1
                If qOpt < mArray(i, j) Then
                    qOpt = mArray(i, j)
                End If
            Next i
        Else
            qOpt = 99999# ' Умовне максимальне значення
            For i = 1 To nRows - 1
                If qOpt > mArray(i, j) Then
                    qOpt = mArray(i, j)
                End If
            Next i
        End If
        idAlt(j) = qOpt
    Next j
    ' Обчислення за методом мінімальної відстані
    iMin = 99999# ' Умовне максимальне значення
    For i = 1 To nRows - 1
```

```
            jSum = 0
            For j = 1 To nCols
                dAlt = mArray(i, j) - idAlt(j)
                jSum = jSum + dAlt * dAlt
            Next j
            jSum = Sqr(jSum)
            If iMin > jSum Then
                iMin = jSum
                rAlt = i
            End If
            qArray(i) = jSum
        Next i
        ' Запис результату
        Result (iMin)
    End Sub
```

Метод кращої суми місць

Нормалізація значень - не єдиний можливий засіб зближення критеріїв, які мають різні розмірності та порядок. Іншим таким засобом може стати використання рангів (наприклад, місць у послідовності). На цьому грунтується метод кращої суми місць.

Упорядкуємо значення певного критерію за зростанням або за спаданням (залежно від того, максимізується чи мінімізується цей критерій). Вищим рангом будемо вважати число m, яке дорівнює кількості альтернатив. Альтернативі з кращим значенням критерію призначимо максимальний ранг, наступною за нею - ранг, на одиницю менший і т.д. Зрозуміло, що альтернатива з найгіршим значенням критерію матиме ранг, що дорівнює одиниці. Якщо кілька альтернатив мають однакові значення критерію, їх ранги усереднюються. Повторення цієї процедури для всіх критеріїв призведе до перетворення початкової матриці критеріїв на матрицю рангів. Агрегування критеріїв досягається усередненням рангів для кожної альтернативи.

Приклад застосування методу кращої суми місць наведено у табл.25.

Таблиця 25

Приклад застосування методу кращої суми місць

A	Початкова			Ранги			Q_a
	Q_1	Q_2	Q_3	R_1	R_2	R_3	
A_1	100,0	3,0	0,15	4	1,5	2	2,500
A_2	80,0	7,0	0,09	2,5	4	3	3,167
A_3	80,0	5,0	0,03	2,5	3	4	3,167
A_4	60,0	3,0	0,20	1	1,5	1	1,167
d	max	max	min	10	10	10	10
						max:	3,167

Процедура, яка реалізує метод мінімальної відстані, наведена у лістингу 20. Для цього методу не виключена можливість появи еквівалентних альтернатив. Це призводить до необхідності включення в програму процедури визначення кількох результатів.

```
' Лістинг 20
'

' Процедура визначення кількох результатів
' Автор А.Панчук
' Дата створення 14.08.1999
'

Private Sub mResult(rValue As Single)
    Dim rRange As Range
    Dim i As Integer
    ' Запис результату
    For i = 1 To nRows - 1
        Cells(qFirst + i - 1, rCol) = qArray(i)
    Next i
    Cells(rRow, rCol) = rValue
    ' Визначення альтернатив
    iAlt(1) = rRow - nRows + iAlt(1) - 1
    Set rRange = Range(Cells(iAlt(1), rCol - nCols - 1),
       Cells(iAlt(1), rCol))
    For i = 2 To nAlt
        iAlt(i) = rRow - nRows + iAlt(i) - 1
        Set rRange = Union(rRange, Range(Cells(iAlt(i), rCol -
           nCols - 1), _Cells(iAlt(i), rCol)))
```

```
    Next i
    rRange.Activate
End Sub
'
' Програма застосування методу кращої суми місць
' Автор А.Панчук
' Дата створення 14.08.1999
'
Sub BestSum()
    Dim iMax, jSum As Single
    Dim rnAlt(), cRang, rSum As Single
    Dim eAlt() As Integer
    Dim k, n As Integer
    ' Визначення виділеної області клітинок
    Define
    ReDim rnAlt(nRows - 1, nCols)
    ReDim eAlt(nRows - 1)
    ' Визначення матриці рангів
    For j = 1 To nCols
    qOpt = mArray(1, j)
    rnAlt(1, j) = nRows - 1
    For i = 2 To nRows - 1
        cRang = nRows - 1
        If (dArray(j) = "max") And (qOpt < mArray(i, j)) Or _
           (dArray(j) = "min") And (qOpt > mArray(i, j)) Then
            qOpt = mArray(i, j)
            rnAlt(i, j) = cRang
        Else
            For k = 1 To i - 1
                If (dArray(j) = "max") And (mArray(i, j) <=
                   mArray(k, j)) Or _ (dArray(j) = "min") And
                      (mArray(i, j) >= mArray(k, j)) Then
                    cRang = cRang - 1
                End If
            Next k
            rnAlt(i, j) = cRang
        End If
```

```
    For k = 1 To i - 1
        If rnAlt(k, j) <= cRang Then
            rnAlt(k, j) = rnAlt(k, j) - 1
        End If
    Next k
Next i
For i = 1 To nRows - 1
    eAlt(i) = -i
Next i
For i = 1 To nRows - 2
    If eAlt(i) = 0 Then
        Exit For
    End If
    eAlt(i) = -eAlt(i)
    qMax = mArray(i, j)
    rSum = rnAlt(i, j)
    n = 1
        For k = i + 1 To nRows - 1
            If qMax = mArray(k, j) Then
                eAlt(k) = -eAlt(k)
                rSum = rSum + rnAlt(k, j)
                n = n + 1
            End If
        Next k
        If n > 1 Then
            rSum = rSum / n
            For k = i To nRows - 1
                If eAlt(k) > 0 Then
                    rnAlt(k, j) = rSum
                    eAlt(k) = 0
                End If
            Next k
        End If
    Next i
Next j
' Обчислення за методом кращої суми місць
iMax = -99999# ' Умовне мінімальне значення
```

```
    For i = 1 To nRows - 1
        jSum = 0
        For j = 1 To nCols
            jSum = jSum + rnAlt(i, j)
        Next j
        jSum = jSum / nCols
        If iMax < jSum Then
            iMax = jSum
            rAlt = i
        End If
        qArray(i) = jSum
    Next i
    ' Визначення кількох однакових розв'язків
    nAlt = 1
    For i = rAlt + 1 To nRows - 1
        If iMax = qArray(i) Then
            nAlt = nAlt + 1
        End If
    Next i
    If nAlt = 1 Then
        ' Запис результату
        Result (iMax)
    Else
        ' Запис кількох результатів
        ReDim iAlt(nAlt)
        i = 0
        For k = 1 To nRows - 1
            If qArray(k) = iMax Then
                i = i + 1
                iAlt(i) = k
            End If
        Next k
        mResult (iMax)
    End If
End Sub
```

Висновки

Розглянуті приклади ілюструють, що за допомогою редактора електронних таблиць MS Excel (або іншої аналогічної програми) можна досить ефективно розв'язувати різноманітні задачі, пов'язані з прийняттям рішень. Тим більше, що дані, які при цьому використовуються, як правило, мають структуру матриць або векторів, що досить просто і навіть органічно можуть бути реалізовані стандартними засобами програми Excel.

Необхідність використання для цього мови програмування Visual Basic не потребує спеціальної серйозної підготовки користувача у галузі програмування: мова Basic була спеціально розроблена для початківців і правила її застосування дуже просто засвоюються за умови знання особливостей системи Windows та програми Excel. Практика показує, що одним з найефективніших прийомів освоєння програмування є використання готових прикладів. Тому посібник за задумом автора має використовуватися, як набір прикладів для розв'язування задач у галузі прийняття рішень.

Розглядаючи та розробляючи такі програми, керівник вимушений аналізувати особливості задачі, порівнювати різні методи її розв'язування, знаходити методи відображення результатів тощо. Кінцевим результатом цієї роботи буде не тільки готовий інструмент для вирішення у подальшому практичних завдань, а й всебічне розуміння проблеми, її особливостей та знаходження шляхів досягання кіцевої мети.

У майбутньому розроблені програми можуть бути вдосконалені професіоналами з погляду ефективності, математичної коректності, і навіть включені до більш складних програмних комплексів, орієнтованих на вирішення широких класів проблем у галузі державного управління.

Список використаної літератури

1. *Колесников А., Пасько В.* Microsoft Office для Windows 95 в бюро. - К.: BHV, 1996.

2. *Орвис В. Дж.* Visual Basic for Application на примерах / Пер. с англ. - М.: БИНОМ, 1995.

3. *Тронь А.П., Тронь В.П.* Критерии оптимального выбора в неопределенных условиях. Учеб. пособие. - К.: ИУНХ, 1986.

4. *Нейман Дж., Моргенштерн О.* Теория игр и экономическое поведение. - М.: Наука, 1970.

ЗМІСТ

Навчальне видання

Використання MS Excel при прийнятті рішень

Методичні рекомендації

Укладач Панчук Анатолій Миколайович

Київ, видавництво УАДУ

Відповідальна за випуск *Н.А.Гегеля*
Редактор *В.Г.Шевельова*
Коректор *С.М.Шиманська*
Комп'ютерна верстка *Є.О.Слободян*

Підписано до друку 1.10.1999.
Формат 60 х 84 $^{1}/_{16.}$ Тираж 500 прим.
Обл.-вид.арк. 5,64.

Видавництво
Української Академії державного управління
при Президентові України.

03057, Київ-57, вул. Ежена Потьє, 20.