Шифр «Надія»

Наукова робота на конкурс за напрямом:

Комп`ютерні науки

на тему:

«Оптимізація процесів забезпечення надійності баз даних розподілених інформаційних систем в умовах негативного зовнішнього впливу»

2019

Зміст

ВСТУП

Актуальність. Негативний зовнішній вплив, кібератаки, віруси суттєво впливають на надійність інформаційного забезпечення розподілених інформаційних систем. На жаль, не існує універсального апробованого методу, який би унеможливий виникнення матеріальних збитків від таких небезпек. Основним невирішеним питанням залишається питання оптимізації процесів забезпечення надійності баз даних.

Мета. Розробити моделі і інформаційну технологію для оптимізації процесів забезпечення надійності баз даних розподілених інформаційних систем в умовах негативного зовнішнього впливу .

Об'єкт дослідження. Надійність баз даних розподілених інформаційних систем в умовах негативного зовнішнього впливу.

Предмет дослідження. Оптимізація процесів забезпечення надійності баз даних.

*Наукова новизна***.** На відміну від існуючих методів, основаних на принципах нечіткої логіки, експетних оцінок та статичних моделей, запропонований метод враховує динаміку процесів виникнення і усунення проблемних ситуацій, а також вижливість інформації, що зберигається в розподілених базах диних.

*Практичне значення***.** Результати можуть використовуватись в системі забезпечення надійності розподілених баз даних автоматизованих систем управління різноманітними процесами і забезпечують прийняття рішень на основі об'єктивних кількісних показників.

Апробація. X Міжнародна студентська конференція «Перший крок у науку».

Кількість публікацій за темою -3.

1 АНАЛІЗ ПРОБЛЕМ ЗАБЕЗПЕЧЕННЯ НАДІЙНОСТІ БАЗ ДАНИХ РОЗПОДІЛЕНИХ ІНФОРМАЦІЙНИХ СИСТЕМ В УМОВАХ НЕГАТИВНОГО ЗОВНІШНЬОГО ВПЛИВУ

1.1 Аналіз вразливостей в розподілених інфомаційних системах

Програмне забезпечення (ПЗ), встановлене в інформаційну систему (ІС), являє собою операційну систему, утиліти і різні прикладні програми, у тому числі засоби захисту інформації (ЗЗІ. Успіх негативного впливу (НВ) на ІС практично повністю визначається моментом часу в який використовується вразливість ПО. У зв'язку з цим особливо важливим стає точне визначення і дослідження життєвого циклу вразливостей [1-7]. Цей життєвий цикл описується певними подіями (або датами цих подій). Списки таких подій різні як за кількістю подій, так і по їх складу, крім того, деяких важливих подій немає ні в одному з таких списків (або вони включені в інші події). Далі приведений список подій, які визначають цикл вразливостей, що представляють об'єднання вже існуючих списків[2-6]:

- дата ін'єкції - дата, коли вразливий код був вперше зареєстрований в репозиторії вихідного коду розробника. Якщо репозиторій не використовується, то це перша дата, коли вразливий код був доданий до поточної збірки.
- дата випуску - дата загальнодоступного випуску системи, яка вперше містить певну вразливість.
- дата виявлення - дата, коли була вперше виявлена вразливість.
- дата розкриття - дата, коли людина або організація, яка виявила вразливість, вперше про неї повідомила вендора (постачальника ПО) або спеціальні установи, що займаються розкриттям вразливостей.
- дата публікації - дата, коли існування вразливості становиться публічно відома (наприклад, через загальнодоступні форуми або випуск патчу).

Дата публікації вразливості часто збігається з датою випуску патчу, який її виправляє.

- дата випуску тимчасового рішення - дата, коли випускається перший тимчасове рішення, яке описує, як усунути уразливість, незалежно від того, офіційно воно випущено (від постачальника).
- дата випуску патчу (оновлення ПЗ, який закриває уразливість) - дата, коли випускається перше виправлення для вразливості, незалежно від того, чи офіційно виправлено (постачальником).
- дата інсталяції патчу або застосування тимчасового рішення - дата, коли на ІС був встановлений патч, що закриває уразливість, або було використане тимчасове рішення, що також усуває вразливість.
- дата створення експлойту (програми або скрипта, що використовує вразливість ПО для НВ на ІС) - дата, коли був випущений перший автоматизований експлойт, що використовує певну вразливість, для негативного впливу на ІС.

В залежності від того, настала та чи інша подія, вразливостям можна привласнити наступні статуси:

- невідома вразливість: вразливість існує в програмному забезпеченні, але ще не була виявлена.
- секретна вразливість: вразливість була виявлена, але той, хто її виявив, що не повідомив про неї вендору (постачальнику ПО), громадськості або спеціальній установі, що займається розкриттям вразливостей. Якщо людина, що виявила вразливість - джерело НВ (ДНВ), вона може бути використана для негативного впливу на ІС.
- розкрита вразливість: вразливість була виявлена, і той, хто її виявив, розкрив інформацію про неї вендору або установі, займається розкриттям вразливостей.
- опублікована вразливість: вразливість була виявлена і оприлюднена або через патч або через загальнодоступний інтернет-ресурс (сайт, форум тощо) або через засоби масової інформації.

- вразливість, для якої існує тимчасове рішення: це вразливість, для якої було створено хоча б одне тимчасове рішення, що закриває її.
- вразливість, для якої існує патч: вразливість, для якої був створений хоча б один патч, що виправляє її.
- закрита вразливість: вразливість, яка була видалена з інформаційної системи за допомогою інсталяції патча, або ж за допомогою застосування тимчасового рішення.
- вразливість, для якої існує експлойт: вразливість, яку ДНВ виявив та створив для неї експлойт.

При цьому одна вразливість може володіти відразу декількома статусами. Наприклад, вразливість може одночасно мати патч і експлойт. Визначальною умовою для потенційної надійності тієї чи іншої ІС є 2 ключових події в життєвому циклі вразливості – виявлення вразливості і її виправлення.

1.2 Методи оцінювання надійності баз даних в умовах негативних впливів

На цей час існує велика кількість підходів до аналізу надійності в умовах негативних впливів як в цілому інформаційних систем, так і в окремих інформаційних технологій [8-19]. На відміну від відомих робіт [8-10], присвячених оцінці впливу на надійність будь-яких дефектів ПЗ, в даній роботі основна увага приділена аналізу підходів [11-19], які так чи інакше зачіпають питання можливості використання вразливостей ПЗ для зовнішніх негативних впливів, що порушують працездатність ІС. Ці підходи можна розділити на 3 категорії.

1. підходи, які офіційно закріплені нормативними документами і мають державний або міжнародний статус.
2. підходи, які використовуються на ринку послуг комп'ютерного безпеки (у бізнесі).

3. підходи, що мають на даний момент тільки науковий додаток.

При порівнянні різних підходів має сенс брати до уваги наскільки повно вони враховують реальні умови функціонування ІС при наявності навмисних негативних впливів (ННВ): скільки факторів вони враховують, які це чинники і яким чином вони враховуються. Підходи до аналізу надійності інформаційних систем при навмисних негативних впливах слід порівнювати за наступними критеріями:

1. чи враховується динаміка надійності інформаційної системи (чи враховуються процеси або враховуються конкретні стани ІС).
2. які процеси враховуються.
3. чи враховується недетермінований характер процесів.
4. які параметри, від яких залежать процеси, враховуються.
5. як оцінюються та враховуються параметри (оцінювання або на основі наявної статистики або на основі прогнозу).

Порівнюючи підходи до аналізу надійності інформаційних систем при цілеспрямованих негативних впливах по 1 критерію, їх можна розділити на статичні і динамічні підходи.

До категорії статичного відносяться такі підходи, які враховують тільки конкретний стан ІС, в основному - це поточний стан [13-14]. Тобто аналізуються поточні умови функціонування ІС, і на основі цього аналізу виконується оцінювання надійності ІС. При цьому передбачається, що поточні умови функціонування ІС змінюватися не будуть, а якщо вони все-таки будуть змінюватися, то ці зміни будуть санкціоновані системними адміністраторами ІС, внаслідок чого вони зможуть при таких змінах оперативно оцінити надійність ІС в нових умовах. Головна проблема такого підходу полягає в тому, що далеко не всі зміни в умовах функціонування ІС залежать від її системних адміністраторів.

Динамічні підходи [17-19], на відміну від статичних, розглядають показники, що характеризують процеси, які впливають на надійність ІС, а не показники, що характеризують конкретний стан ІС. Як приклад можна привести модель, що описує динаміку появи вразливостей в ІС, засновану на теорії масового обслуговування

[17], модель конфлікту ДНВ і ІС [18] і модель оцінки надійності системи захисту інформації від несанкціонованого доступу [19].

2 МОДЕЛІ РОЗВИТКУ НЕГАТИВНИХ ВПЛИВІВ В ІНФОРМАЦІЙНИХ СИСТЕМАХ

2.1 Аналіз сутності конфліктів

В якості універсальної синтетичної методології досліджень в сфері надійності інформаційних систем і технологій доцільно розглядати методологію математичного та комп'ютерного моделювання динаміки конфлікту.

Пропонується розглянути 4 типові ситуації розгортання такого конфлікту:

S1. Інформаційна система без ЗЗІ знаходиться в конфліктному взаємодії з одним зовнішнім ДНВ.

S2. Інформаційна система з ЗЗІ знаходиться в конфліктному взаємодії з одним зовнішнім ДНВ.

S3. Інформаційна система без ЗЗІ знаходиться в конфліктній взаємодії з коаліцією ДНВ у відсутності інсайдера (людини, що має прямий доступ до ПЗ ІС і поставляє інформацію про це ПЗ і вразливості в ньому ДНВ), що працює в даній організації.

S4. Інформаційна система без ЗЗІ знаходиться в конфліктній взаємодії з коаліцією ДНВ при наявності інсайдера, що працює в організації, на ІС якої планується НВ.

Дані ситуації не вичерпують всіх можливих варіантів конфлікту ІС – ДНВ, проте, на їх основі, виходячи з тих же принципів, можуть бути розглянуті і інші варіанти.

Крім того, варто відзначити, що при розрахунках, що ілюструють можливості розробленої моделі, враховувалися вразливості з будь-яким ступенем серйозності,

при використанні цих моделей на практиці слід або виконувати окремий аналіз вразливостей різного ступеня серйозності, або привласнювати вразливостям різного ступеня серйозності коефіцієнти, що відображають їх вплив на надійність ІС. Для прикладу, розглянемо один з випадків, наприклад S1.

2.2. Моделювання конфліктів в інформаційних системах

Нехай є ІС з встановленим ПО. ДНВ, яке навмисно впливає на ІС, як правило, проводить попередній аналіз (комп'ютерну розвідку) програмного забезпечення, встановленого на ІС, досліджує вразливості в цьому програмному забезпеченні і можливі способи використання цих вразливостей, при цьому ДНВ може володіти різною кваліфікацією і можливостями, які описуємо як середній час, який потрібен ДНВ.

Математична модель конфлікту ґрунтується на представленні процесу зміни станів об'єднаної системи ІС - ДНВ в вигляді ланцюга Маркова з кінцевим числом станів, переходи між якими здійснюються за експоненціальним (пуассонівського) закону розподілу [23,24]. Дана модель є розширенням найпростішої математичної моделі ІС в плані обліку дій ДНВ в залежності від його обізнаності та кваліфікації. На рис. 2.2.1 представлені стани, в яких може перебувати ДНВ.

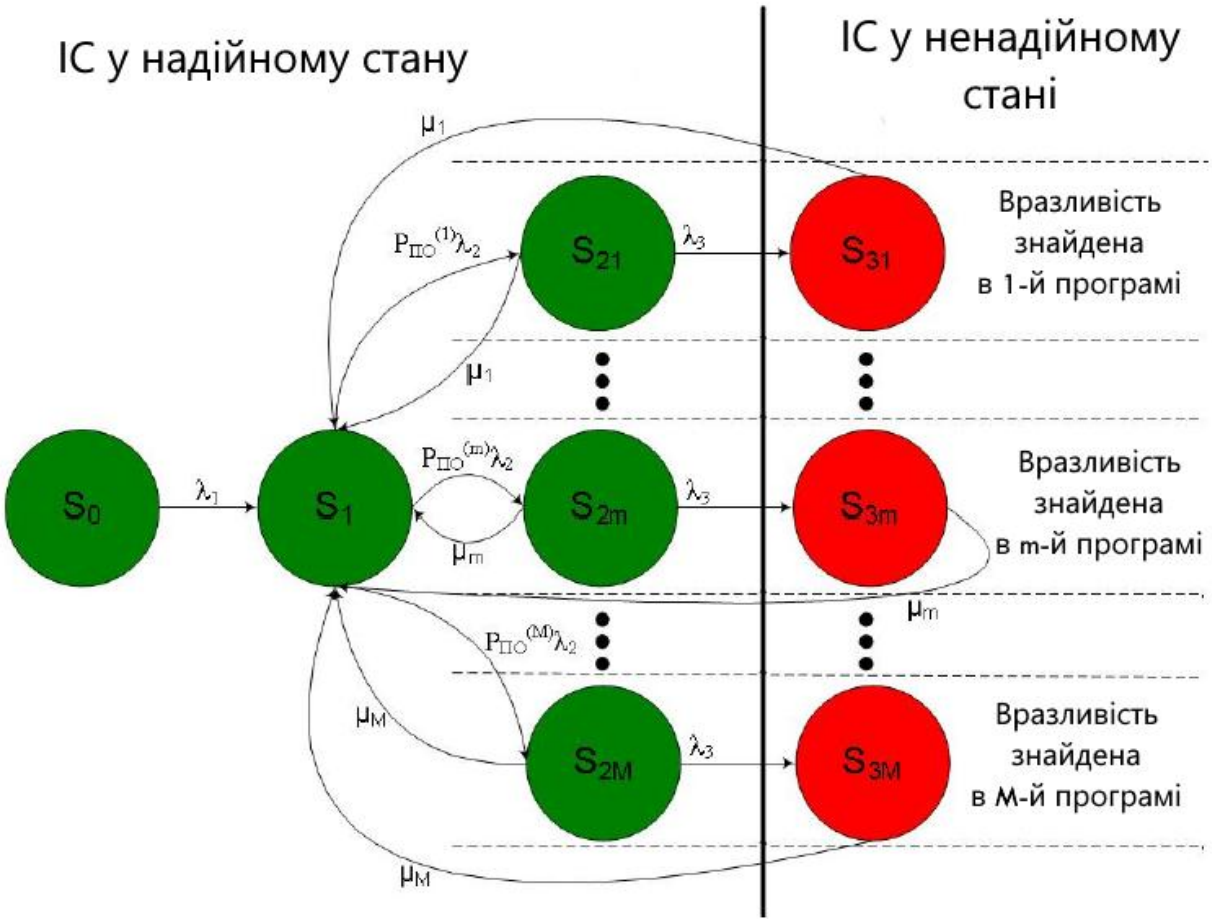

Рисунок 2.2.1 – Математична модель конфлікту ІС без ЗЗІ та ДНВ,

де μ- інтенсивність переходів із станів $S_{2m}(m \in 1 \dots M)$ та $S_{3m}(m \in 1 \dots M)$ до стану S_1;

$Р_{ПО}$ – ймовірності знаходження в зазначених станах;

λ – інтенсивність перехід із стану S_i в S_{i+1};

Вузли ланцюга відповідають наступним станам:

S_0- у ДНВ відсутня будь-яка інформація про ІС;

S_1- - у ДНВ є інформація про ПО ІС;

S_{2m} - у ДНВ є інформація про ПО ІС і про одну вразливість в цьому ПО, де m – номер програми, в якій була знайдена вразливість (m є 1..M), а M – кількість програм в ІС;

S_{3m} (m є 1..M) - у ДНВ є інформація про ПО ІС про одну вразливості в цьому ПО, а також про спосіб використання цієї вразливості для здійснення НВ на ІС. Ймовірності знаходження в зазначених станах позначено відповідно P_0, P_0, P_{21}, ..., P_{2m}, ..., P_{2M}, P_{31}, ..., P_{3m}, ..., P_{3M}.

3 ВИБІР СТРАТЕГІЙ ДЛЯ ЗАБЕЗПЕЧЕННЯ ЦІЛІСНОСТІ ІНФОРМАЦІЙНОЇ СИСТЕМИ В УМОВАХ КОНФЛІКТНИХ ВЗАЄМОДІЙ

3.1 Ранжування баз даних за ступенем важливості

Задачу ранжування приведено до автоматичної класифікації на основі транзитивного замикання нечіткого відношення подібності.

Відомі підходи до ранжування елементів системи розділяють на два класи, виходячи з типу причинно-спадкових зв’язків між відмовами. Назвемо ці підходи зовнішнім і внутрішнім.

Зовнішній підхід використовує чутливість надійності системи до зміни надійності її елементів. Розглянемо функцію надійності:

$$P_s = f(P_1, P_2, \dots, P_n) \quad (1)$$

Дана функція пов'язує ймовірності безвідмовної роботи системи (P_s) і її елементів (P_i). Уявімо цю функцію у вигляді ряду:

$$P_s = b_0 + \sum_{i=1}^{n} b_i P_i + \sum_{i=1}^{n} b_{ij} P_i P_j + \cdots, \quad (2)$$

Коефіцієнти якого мають сенс часткових похідних:

$$b_i = \frac{dP_s}{dP_i}, b_{ij} = \frac{d^2 P_s}{dP_i P_j} \quad (3)$$

Коефіцієнт b_i в (3) відповідає індексу важливості і-го елемента (reliability importance index). Коефіцієнт b_{ij} в (3) відповідає індексу важливості спільного впливу і-го і j-го елементів (Joint reliability importance). Для систем, надійність яких моделюється методом Monte-Carlo. Обчислення важливості елементів системи безпосередньо на основі логічної (або структурної) функції має наступний вигляд:

$$\alpha_s = f_L(\alpha_1, \alpha_2, \dots, \alpha_n) \quad (4)$$

де α_s (α_i) = 1 (0), якщо система (і-й елемент) працює (не працює), f_L - булева функція.

Внутрішній підхід пов’язаний з оцінкою значимості елементів на основі теорії відносин і графів.

Внутрішній підхід не вимагає побудови структурної функції (4) і функції надійності (1). Він спирається на інформацію про структуру системи, тобто склад її елементів і зв'язки між ними. При цьому можуть використовуватися знання про вплив порушень в одних елементах на виникнення порушень в інших елементах. Наприклад, «догляд параметрів і-го елемента призводить до відтоку параметрів j-го елемента, що в свою чергу призводить до відмови k-го елемента, і т.д.». Таким чином, може враховуватися «ефект доміно». Носієм інформації для обчислення рангів в служить матриця зв'язків.

$$A = [a_{ij}], i, j = 1,2, \dots, n \quad (5)$$

В якій a_{ij} = 1 (0), якщо і-й елемент пов'язаний (не пов'язаний) з j-м елементом. Ранг і-го елемента обчислюється як сума елементів і-го рядка матриці

$$D = A + A^2 \quad (6)$$

Яка враховує однокрокові і двокрокові впливи порушень в і-м елементі системи.

Нижче пропонується рішення задачі ранжування елементів на основі нечіткого транзитивного замикання і спеціальних процедур побудови нечітких відносин впливу і подібності, запропонований А.П. Ротштейном [25].

Нехай $\{x_1, x_2, \dots, x_n\}$ – множина елементів системи. Вплив елементу $x_i \in$ X на інші елементи задаємо нечіткою множиною:

$$I_i = \left\{\frac{\mu_{i1}}{x_1}, \frac{\mu_{i2}}{x_2}, \dots, \frac{\mu_{in}}{x_n}\right\} \quad (7)$$

де μij - число в інтервалі [0,1], яке характеризує ступінь впливу елемента $x_i \in$ X на елемент $x_i \in$ X; i,j = 1, 2, ..., n.

Будемо припускати, що вплив елемента $x_i \in$ X на самого себе відсутній, тобто

$$\mu_{ii} = 0, i = 1,2, \dots, n. \quad (8)$$

Сукупність ступенів впливу μ_{ij} з (7) для всіх елементів $x_i \in$ X утворює нечітке відношення впливу I, яке задане на декартовому добутку X × X, тобто I ⊂ X × X:

$$I = \left[\frac{\mu_{ij}}{(x_i, x_j)}\right], i, j = 1,2, \dots, n \quad (9)$$

Число μ_{ij}, яке ставиться у відповідність кожній парі елементів (x_i, x_j), може здаватися експертно, або методом найменшого впливу, який пропонується нижче. Зауважимо, що подібна процедура обчислення ступенів належності використовувалася раніше в роботі [22].

Нехай f_{ij} - сила впливу елемента $x_i \in$ X на елемент $x_j \in$ X, причому виконується умова: «чим більше сила f_{ij}, тим більше ступінь впливу μ_{ij} », тобто має місце

співвідношення:

$$\frac{\mu_{i1}}{f_{i1}} = \frac{\mu_{i2}}{f_{i2}} = \cdots = \frac{\mu_{il}}{f_{il}} = \cdots = \frac{\mu_{in}}{f_{in}} \tag{10}$$

Передбачається, що відповідно до (8):

$$f_{ii} = 0, i = 1,2, \dots, n \tag{11}$$

Нехай x_l - елемент, на який елемент x_i має найменший вплив. З (10) маємо:

$$\mu_{ij} = \mu_{il}\frac{f_{i1}}{f_{il}}, \mu_{i2} = \mu_{il}\frac{f_{i2}}{f_{il}}, \dots, \mu_{ij} = \mu_{il}\frac{f_{in}}{f_{il}} \tag{12}$$

Підставляючи (12) у вимогу:

$$\mu_{ij} + \mu_{ij} + \cdots + \mu_{ij} = 1, i = 1,2, \dots, n,$$

Отримуємо найменший ступінь впливу елемента $x_i \in$ X в системі:

$$\mu_{i1} = \left(\frac{f_{i1}}{f_{il}} + \frac{f_{i1}}{f_{il}} + \cdots + \frac{f_{in}}{f_{il}}\right)^{-1} \tag{13}$$

Співвідношення (13) і (12) дозволяють обчислювати ступеня впливу в нечіткому відношенні (9) шляхом порівняння сил впливів f_{ij} з найменшою силою впливу f_{il} для кожного елемента $x_i \in$ X. Для цього використовується 9-бальна шкала Сааті.

$$\frac{f_{ii}}{f_{il}} = 1,3,5,7,9, \tag{14}$$

якщо вплив «ij» (елемента x_i на елемент x_j) по порівнянню з найменшим впливом «iℓ» (елемента x_i на елемент x_l): 1 – вплив однаковий, 3 - вплив трохи більший, 5 - вплив більший, 7 -вплив значно більший, 9 - вплив абсолютно більший (можливі проміжні оцінки: 2, 4, 6, 8).

Міру подібності за ступенем впливу між елементами $x_i \in$ X і $x_j \in$ X визначимо величиною

$$r_{ij} = 1 - d_{ij}, \tag{15}$$

де d_{ij} - відстань між нечіткими множинами впливу елементів x_i і x_j:

$$I_i = \left\{\frac{\mu_{i1}}{x_1}, \frac{\mu_{i2}}{x_2}, \dots, \frac{\mu_{in}}{x_n}\right\}, I_j = \left\{\frac{\mu_{j1}}{x_1}, \frac{\mu_{j2}}{x_2}, \dots, \frac{\mu_{jn}}{x_n}\right\}$$

Для обчислення d_{ij} можуть використовуватися відносні відстані по Хеммінгу ($d_{ij}^{(h)}$) або по Евкліду ($d_{ij}^{(e)}$):

$$d_{ij}^{(h)} = \frac{1}{n}\sum_{k=1}^{n}\left|\mu_{ik} - \mu_{jk}\right| \tag{16}$$

$$d_{ij}^{(e)} = \frac{1}{n}\sqrt{\sum_{k=1}^{n}\left(\mu_{ik} - \mu_{jk}\right)^2} \tag{17}$$

Сукупність величин r_{ij} для всіх пар $(x_i,\ x_j) \in X \times X$ утворює нечітке відношення подібності $R \subset X \times X$:

$$R = \left[\frac{r_{ij}}{x_i, x_j}\right] \tag{18}$$

яке має властивості:

(А) рефлексивність, тобто $r_{ij} = 1$, для всіх $x_i \in X$,

(Б) симетричність, тобто $r_{ij} = r_{ji}$, для всіх $x_i \in X$, $x_j \in X$.

Для розбиття множини X на непересічні класи елементів, подібних за ступенем впливу, необхідно надати вихідного нетранзитивному відношенню подібності R властивість транзитивності. Дане перетворення забезпечується операцією транзитивного замикання нечіткого відношення.

Нехай стоїть задача розподілу обмежених щоденних ресурсів на підтримку цілісності баз даних інформаційної системи керування залізничним транспортом. Структура обраної системи зображена на рисунку 3.1.1.

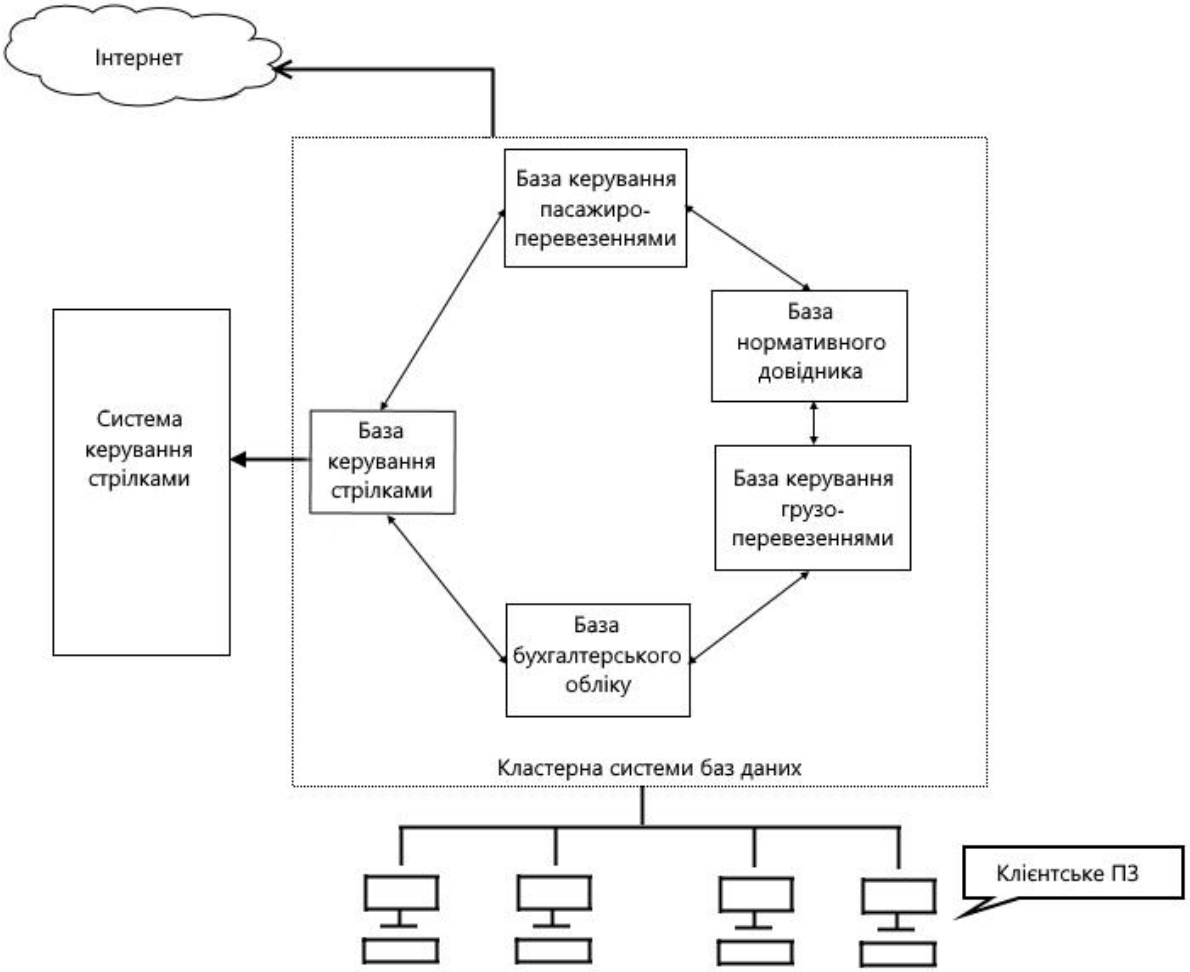

Рисунок 3.1.1 – Спроектована модель ІС залізничного транспорту

Таким чином розглядається інформаційна система, яка складається з 5 баз даних, тобто $X = \{x_1, x_2, x_3, x_4, x_5\}$. Експерта інформація, необхідна для обчислення відношення методом найменшого впливу, представлена у таблиці 3.1.1

Таблиця 3.1.1 – Вихідні дані для методу найменшого впливу

x_i	x_l	*ij/ il*				
		x_1	x_2	x_3	x_4	x_5
x_1	x_2	0	1	5	3	1
x_2	x_1	1	0	9	3	1
x_3	x_5	9	3	0	2	1
x_4	x_2	1	1	7	0	3
x_5	x_1	1	5	9	7	0

Другий стовпець таблиці 3.1.1 (x_l) містить елементи, на які відповідні елементи першого стовпчика (x_i) мають найменший вплив: x_1 найменш впливає на x_2, x_2 найменш впливає на x_2, ..., x_5 найменш впливає на x_1. Джерелом цієї інформації являється експерт.

Користуючись даними з таблиці 3.1.1 і формулами, обчислимо ступені впливу μ_{ij} для відношення. Тут $\mu_{ii} = 0$.

Для елемента x1 (i = 1, ℓ = 2) маємо:

$$\mu_{11} = 0$$

$$\mu_{12} = \left(\frac{f_{11}}{f_{12}} + \frac{f_{12}}{f_{12}} + \frac{f_{13}}{f_{12}} + \frac{f_{14}}{f_{12}} + \frac{f_{15}}{f_{12}}\right)^{-1} = \frac{1}{0+1+5+3+1} = \frac{1}{10}$$

$$\mu_{13} = \mu_{12}\frac{f_{13}}{f_{12}} = \frac{1}{10}5 = \frac{5}{10}$$

$$\mu_{12} = \mu_{12}\frac{f_{14}}{f_{12}} = \frac{1}{10}3 = \frac{3}{10}$$

$$\mu_{12} = \mu_{12}\frac{f_{15}}{f_{12}} = \frac{1}{10}1 = \frac{1}{10}$$

Аналогічно обчислюємо інші ступені приналежності, які утворюють нечітке відношення впливу:

Таблиця 3.1.2 – значення нечітких відношень

	x_1	x_2	x_3	x_4	x_5	
x_1	0	1/10	5/10	3/10	1/10	max=5/10
x_2	1/14	0	9/14	3/14	1/14	max=9/14
x_3	9/15	3/15	0	2/15	1/15	max=9/15
x_4	1/12	1/12	7/12	0	3/12	max=7/12
x_5	1/22	5/22	9/22	7/22	0	max=9/22

Для нормалізації відношення розділимо елементи кожного рядка на максимальне значення і записуємо результати у таблицю 3.1.3.

Таблиця 3.1.3 – нормалізоване значення нечітких відношень

	x_1	x_2	x_3	x_4	x_5
x_1	0	0.2	1	0.6	0.2
x_2	0.11	0	1	0.33	0.11
x_3	1	0.33	0	0.22	0.11
x_4	0.14	0.14	1	0	0.43
x_5	0.11	0.56	1	0.78	0

Нечітке відношення подібності має вигляд таблиці 3.1.4.

Таблиця 3.1.4 – нечітке відношення подібності

	x_1	x_2	x_3	x_4	x_5
x_1	1	0.87	0.48	0.76	0.83
x_2	0.87	1	0.53	0.83	0.44
x_3	0.48	0.53	1	0.48	0.44
x_4	0.79	0.83	0.48	1	0.67
x_5	0.83	0.78	0.44	0.67	1

Ступені приналежності із використанням відстані по Хеммінгу. Наприклад, $r_{12} = 1 - d_{12}$, де

$$d_{12} = \frac{1}{5}[(0{,}0{,}2{,}1{,}0{,}6{,}0{,}2) - (0{,}11{,}0{,}1{,}0{,}33{,}0{,}1)]$$

$$= \frac{1}{5}[|0 - 0{,}11| + |0{,}2 - 0| + |1 - 1| + |0{,}6 - 0{,}33| + |0{,}2 - 0{,}1|]$$

$$= \frac{1}{5}[0{,}11 + 0{,}1 + 0 + 0{,}27 + 0{,}09] = 0{,}13$$

Для отримання транзитивного замикання відношення подібності використовуємо нечітку композицію (таблиця 3.1.5).

Таблиця 3.1.5 – Обчислення значення транзитивного замикання відношення подібності

	x_1	x_2	x_3	x_4	x_5
x_1	1	0,87	0,53	0,83	0,83
x_2	0,87	1	0,53	0,83	0,83
x_3	0,53	0,53	1	0,53	0,53
x_4	0,83	0,83	0,53	1	0,79
x_5	0,83	0,83	0,53	0,79	1

У результаті бачимо, що транзитивне замикання в нашому випадку має вигляд:

$$\hat{R} = R \cup R^2 \cup R^3 \cup \ldots \cup R^k \cup \ldots = R^3,$$

тобто збігається з відношенням. Підсумовуючи значення рядків матриці, отримуємо кількісні значення рангів елементів:

$$p_1 = 0 + 0{,}87 + 0{,}53 + 0{,}83 = 4{,}06$$

$$p_2 = 4.06, p_3 = 3.12, p_4 = 4.02, p_5 = 4.02$$

Нечітке відношення (26) можна розкласти по α-рівням наступним чином:

$$\hat{R} = \bigcup_{\alpha} \alpha R_{\alpha} = 0{,}53R_{0{,}53} \bigcup 0{,}83R_{0{,}83} \bigcup 0{,}87R_{0{,}87} \bigcup R_1,$$

Для стислості елемент x_i позначається символом i, i=1,2,..., 5. Чіткі відносини α-рівня утворюють класи елементів, еквівалентних за важливістю (таблиця. 3.1.6).

Таблиця 3.1.6 – Класи елементів, еквівалентних по важливості

Рівень	Число класів	Класи елементів
$\alpha = 0{,}53$	1	$\{x_1, x_2, x_3, x_4, x_5\}$
$\alpha = 0{,}83$	2	$\{x_1, x_2, x_4, x_5\}, \{x_3\}$
$\alpha = 0{,}87$	4	$\{x_1, x_2\}, \{x_3\}, \{x_4\}, \{x_5\}$
$\alpha = 0{,}1$	5	$\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}$

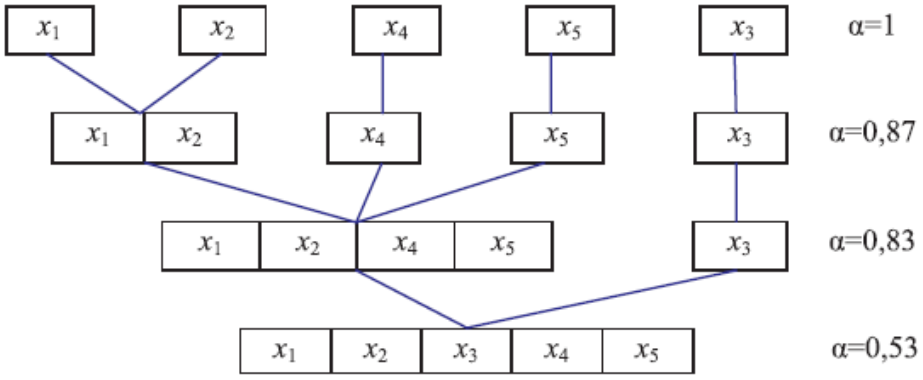

Рисунок 3.1.2 – Дерево декомпозиції на класи еквівалентності

Число α може інтерпретуватися як рівень визначеності наших знань про систему, а (1 - α) - рівень невизначеності. Тобто чим складніше система і чим більше число реалій не враховується при моделюванні, тим більше невизначеність і нижче число α.

З рис. 3.1.2 видно, що при максимальній визначеності (α = 1) кожен з елементів x_i являє собою унікальний клас важливості. Однак на рівні α = 0,53 всі елементи системи не помітні по рангах. З урахуванням кількісних оцінок для практичних розрахунків можна вибрати рівень визначеності α = 0,83, на якому:

$$p_1 = p_2 = p_4 = p_5 \approx 4, p_3 \approx 3$$

Якщо C_0 - допустимі витрати на забезпечення надійності системи, то з урахуванням рангів елементів ці витрати повинні розподілятися так:

$$\sum_{i=1}^{5} C_i = C_0\,,\ \ C_1 = C_2 = C_4 = C_5 = \frac{4}{19} C_0\,, C_3 = \frac{3}{19} C_0$$

3.2 Розробка моделі оптимізації процесу забезпечення надійності

Розв'язком задачі, що поставлено, є значення фінансових витрат на забезпечення цілісності конкретної бази даних. Якщо V_0 - допустимі витрати на забезпечення надійності системи, то з урахуванням рангів елементів ці витрати повинні розподілятися наступним чином:

$$V_i = \lambda V_0,$$

де $(i = i, n) -$ кількість баз даних

$\lambda -$ коефіцієнт розподілення ресурсів.

Для попереднього прикладу:

$$\sum_{i=1}^{5} V_i = V_0\,,\ \ V_1 = V_2 = V_4 = V_5 = \frac{4}{19}V_0\,, V_3 = \frac{3}{19}V_0$$

Дані витрати визначають параметри:

k_1 – кількість основного і допоміжного персоналу

t_v – технологію виявлення порушень

t_y – технологію усунення наслідків НВ, створеного ДНВ

Параметри k_1, t_v, t_y фактично визначають характеристики моделей ІС (рис 2.2.1). Це дозволяє вирішити задачу розподілу обмеженого ресурсу на забезпечення цілісності баз даних в умовах конфліктних взаємодій.

Для вирішення завдання вибору оптимальних стратегій процес усунення негативного впливу на базу даних представимо у вигляді напівмарковських процесу (НМП) – рисунок 3.2.1.

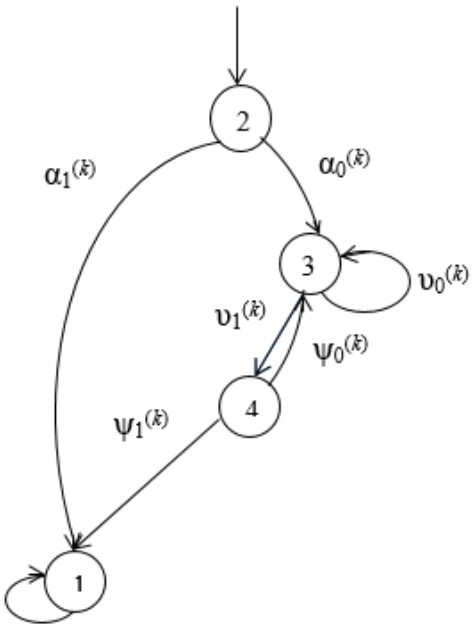

Рис. 3.2.1. - Граф подій процесу усунення негативного впливу на базу даних

де S_1 - працездатний стан

S_2 -стан, який відповідає початку негативного впливу;

S_3 - стан порушення в базі даних і невиявлений фактом порушення;

S_4 - стан порушення в базі даних і виявлений фактом порушення;

У даному документі введено такі показники якості виконання операцій по усуненню негативного впливу

$\alpha_1^{(k)}$– ймовірність автоматичного усунення негативного впливу при застосуванні *k*-ї стратегії, $k \in K_a$, K_a– множина стратегій для автоматичного усунення негативного впливу;

$\alpha_0^{(k)}$ – ймовірність порушення ($\alpha_0^{(k)}=1-\alpha_1^{(k)}$) при застосуванні *k*-ї стратегії, $k \in K_a$;

$\upsilon_1^{(k)}$– ймовірність виявлення порушення ДНВ при застосуванні *k*-й стратегії $k \in K_p$

K_p– безліч стратегій виявлення порушення системним адміністратором;

$\upsilon_0^{(k)}$– ймовірність не виявлення порушення системним адміністратором;

$\psi_1^{(k)}$ – ймовірність безпомилкового усунення порушення системним адміністратором при застосуванні *k*-ї стратегії $k \in K_p$,

K_p– множина стратегій виявлення порушення системним адміністратором;

$\psi_0^{(k)}$ – ймовірність усунення порушення вендором з помилкою *k*-ї стратегії *k* ϵK_p.

Матриця НМП в позначеннях показників якості виконання операцій задачі оптимізації прийняття рішення про організацію процесу усунення негативного впливу на базу даних має вигляд:

$$\left[P_{ij}^{k}\right]=\begin{pmatrix} 1 & 0 & 0 & 0 \\ \alpha_1^{(k)} & 0 & \alpha_0^{(k)} & 0 \\ 0 & 0 & \upsilon_0^{(k)} & \upsilon_1^{(k)} \\ \psi_1^{(k)} & 0 & \psi_0^{(k)} & 0 \end{pmatrix}$$

Вектор початкових ймовірностей:

$a = (a_2, a_3, ,a_4)=(1,0,0)$

K - безліч всіх стратегій для усунення негативного впливу на базу даних.

$K=K_a U K_p$, де

K_a– безліч стратегій для автоматичного усунення негативного впливу на базу даних;

K_p– безліч стратегій виявлення порушення людиною;

Кожному варіанту закінчення функціонування процесу усунення негативного впливу на базу даних на графі подій ставимо у відповідність поглинаючий стан, наприклад, «працездатний стан». Поглинаючу вершину нумеруємо першим *s* натуральним числом (*s=1*). Для початкових вершин, які нумеруються числами з числової послідовності після первої *s* поглинає вершини, задаємо вектор початкових ймовірностей:

$a = (a_{s+1}, a_{s+2}, \ldots, a_n)$, при цьому : $\sum_{i=s+1}^{N} a_i = 1$,

де *N* – загальна кількість вершин графа, з яких перша *s*- поглинаюча.

У кожній вершині і може бути K_i альтернатив або рішень, $K_i \epsilon K$. Кожному рішенню відповідає свій набір переходів, який характеризується ймовірністю переходу з вершини *i* в вершину *j* при виборі *k*-го рішення, $k \in K_i$.

В умовах задачі прийняття рішення про організацію усунення негативного впливу на базу даних *k*-е рішення означає застосування стратегії $k \in K_i \in K$ на етап технологічного процесу, який моделюється вершиною *i* графа подій.

$P_{ij}^{(k)}$ – ймовірність переходу системи з вершини *i* в вершину *j* при виборі *k* -го рішення. Причому:

$\sum_j p_{ij}^{(k)} = 1$ при всіх *i* и при всіх $k \in K_i$.

$T_{ij}^{(k)}$ – середній час *i*-ї роботи при *k*-му рішенні при переході в вершину *j*. В цьому випадку середнє очікуване час *i*-ї роботи при *k*-й стратегії $T_i^{(k)}$ обчислюються за наступною формулою:

$$T_i^{(k)} = \sum_j P_{ij}^{(k)} T_{ij}^{(k)}$$

$r_{ij}^{(k)}$ –середня вартість *i*-ї роботи по усуненню негативного впливу на базу даних при *k*-й стратегії при переході в вершину *j*. У цьому випадку середня очікувана вартість *i*-ї роботи при *k*-й стратегії обчислюються за наступною формулою:

$$r_i^{(k)} = \sum_j P_{ij}^{(k)} r_{ij}^{(k)}$$

Під оптимізацією системи розуміється вибір в кожній вершині такого рішення (альтернативи), щоб цільовій функції доставлявся екстремум.

В якості цільової функції беремо *T* - середній час процесу усунення негативного впливу на базу даних до поглинання (вершина *s=1* на графі подій):

$$T = \sum_{i=2}^{N} \sum_{k \in K_i} P_{i1}^{(k)} T_{i1}^{(k)} x_i^{(k)}$$

У формулі $x_i^{(k)}$ – змінна, яка характеризує вибір рішення:

$x_i^{(k)} > 0$ у тому випадку, якщо в *i*-й вершині для виконання роботи по усуненню негативного впливу на базу даних обрано *k*-е рішення та $x_i^{(k)} = 0$, у протилежному випадку.

У задачі оптимізації враховуємо той факт, що середня вартість роботи по усуненню негативного впливу на базу даних V не повинна перевищувати заданого значення V_0.

$$V = \sum_{i=2}^{N} \sum_{k \in K_i} P_{i1}^{(k)} r_{i1}^{(k)}$$

Для обліку обмеження за вартістю, вводяться булеві змінні $\delta_i^{(k)}$ (для застосування на етапі технологічного процесу, який моделюється вершиною *i* графа подій, *k*-ї стратегії) і додаються нові обмеження.

Результатами оптимізації є:

Значення цільової функції T_i – значення середньої вартості роботи по усуненню негативного впливу на базу даних *V*;

Номер стратегії в кожній вершині, який доставляє екстремум цільової функції.

Таким чином, завдання оптимізації прийняття рішення про організацію процесу усунення негативного впливу на базу даних з урахуванням обмеження на вартість виконання операцій *V*, має вигляд:

$$T = \sum_{i=2}^{N} \sum_{k \in K_i} P_{i1}^{(k)} T_{i1}^{(k)} x_i^{(k)} \rightarrow \min \qquad (19)$$

$$\sum_{i=2}^{N} \sum_{k \in K_i} P_{i1}^{(k)} r_{i1}^{(k)} x_i^{(k)} \le V_0; \qquad (20)$$

$$\sum_{k \in K_i} x_j^{(k)} - \sum_{i=r+1}^{N} \sum_{k \in K_i} P_{ij}^{(k)} x_i^{(k)} = a_j, \ \text{j=2,3,...}N; \qquad (21)$$

$$\sum_{i=2}^{N} \sum_{k \in K_i} P_{i1}^{(k)} x_i^{(k)} = 1; \qquad (22)$$

$x_i^{(k)} \ge 0$ для всіх *i* та всіх $k \in K_i$; (23)

$\sum_{k \in K_i} \delta_i^{(k)} = 1,$ для всіх *i*; (24)

$x_i^{(k)} - M * \delta_i^{(k)} \le 0$, для всіх *i* та $k \in K_i$ (26)

$x_i^{(k)} - m * \delta_i^{(k)} \ge 0$, для всіх *i* та $k \in K_i$ (26)

$\delta_i^{(k)} \in \{0;1\}$ для всіх *i i* всіх $k \in K_i$; (27)

де *m* і *M* - досить мале і досить велике число.

Рішення даної задачі лінійного програмування (19) - (27) має таку властивість, що для кожного і лише один $x_i^{(k)}$ відмінний від нуля. Це означає, що в вершині *i* для виконання роботи приймається *k*-е рішення, $k \in K_i$, тобто застосовується стратегія $k \in K_i \in K$.

Сенс зазначених обмежень наступний:

Обмеження (22) є нормованою умовою, що вимагає, щоб з ймовірністю 1 процес поглинувся.

Обмеження (24) для булевої змінної $\delta_i^{(k)}$ вимагає, щоб для кожного *i* лише одне $\delta_i^{(k)}$ було рівне одиниці. Це означає, що в кожній вершині *i* для виконання роботи приймається тільки одне *k*-е рішення, $k \in K_i$;

Обмеження (25) вимагає, щоб при кожному *i* не більше ніж одне $x_i^{(k)}$ було відмінно від нуля (спільно з обмеженням (5));

Обмеження (27) вимагає, щоб при кожному *i* одне або більше $x_i^{(k)}$ було відмінно від нуля. Спільно з цим, обмеження (25) і (26) вимагають, щоб лише одне $x_i^{(k)}$ було відмінне від нуля.

Зв'язок між кожним із завдань оптимізації здійснений через обмеження типу (20), в якому V_0 – розв'язок задачі ранжування важливості баз даних.

3.3 Інформаційна технологія для підтримки прийняття рішень

Метод ранжування був реалізований у середовищі Visual Studio версії 2010 за допомогою мови програмування C#. Обрана мова програмування аргументовата тим, що вона допомагає створювати віконні програмні продукти майже автоматично.

Для кращого розуміння коду та його роботи, програма була поділена на декілька функцій – частин коду, які викликаються у основній функції програми. На таблиці 3.3.1 наведений опис усіх присутніх функцій.

Таблиця 3.3.1 – Опис функцій

№	Заголовок	Назва	Пояснення
1.	Form1()	Завантаження форми	Описуються дії, які виникнуть під час завантаження вікна програми
2.	RoundTo2()	Округлення масиву	Округлення обраного масиву значень до 2 чисел після коми
3.	button1_Click()	Обробник подій елементу «кнопку»	Описані дії, які виконуються після натиснення на елемент «кнопка» з назвою «Расчитать»
4.	dataGridView1_RowPostPaint	Створення стилю заголовків таблиці	Використовується для видалення чорного трикутника у заголовках рядів таблиці
5.	value_size_ValueChanged	Зміна розмірності таблиці	Після зміни числа у полі вводу, таблиця змінює свій розмір на заданий
6.	Normalization	Нормалізація відношення	Виконання нормалізації відношення масиву значень степенів приналежності
7.	Composition	Транзитивне замикання	Виконання транзитивного замикання відношення
8.	fuzzy_sim	Нечітке відношення	Виконання перетворення масиву значень до нечіткого відношення
9.	Find_Classes	Класи елементів	Останній етап виконання алгоритму ранжування
10.	button3_Click()	Кнопка «Очистить таблицы»	Видалення усіх даних з усіх таблиць
11.	min_i_CellContentClick	Введення значень	Занесення коректних даних до таблиці
12.	button2_Click()	Довідка	Виведення інформації про заповнення таблиць
13.	array_check()	Перевірка масиву	Перевірка масиву на наявність чисельного типу

У додатку А наведений лістинг програми, у додатку Б – приклад роботи програми, у додатку В – комп’ютерні експерименти.

ВИСНОВКИ

Для моделювання процесу виникнення та усунення негативних впливів на бази даних інформаційної системи зручно використовувати апарат напівмарківських процесів.

Побудована напівмарківська модель процесу прийняття рішення про вибір заходів по забезпеченню надійності баз даних дозволяє обирати стратегії виявлення атак і помилок, а також усунення їх наслідків.

Побудована відповідна модель зведена до задачі лінійного програмування, що є передумовою вирішення її в будь-яких програмних середовищах.

Розроблені модель і програмне забезпечення виявлення коефіцієнтів важливості баз даних використовує технологію обробки експертних оцінок, але є вільною від обмежень відомих експертних методів типу аналізу ієрархій.

Приведені комп'ютерні експерименти довели конструктивність підходу і доцільність використання моделей і програмного забезпечення в практиці експлуатації інформаційних систем.

*Наукова новизна***.** На відміну від існуючих методів, основаних на принципах нечіткої логіки, експетних оцінок та статичних моделей, запропонований метод враховує динаміку процесів виникнення і усунення проблемних ситуацій, а також вижливість інформації, що зберигається в розподілених базах диних.

*Практичне значення***.** Результати можуть використовуватись в системі забезпечення надійності розподілених баз даних автоматизованих систем управління різноманітними процесами і забезпечують прийняття рішень на основі об'єктивних кількісних показників.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ozment, A. Vulnerability discovery & software security: dissertation for the degree of Ph.D. / Andy Ozment. – University of Cambridge, 2007. – 139 p.
2. Frei, S. Security econometrics - the dynamics of (in)security: dissertation for the degree of Doctor of Science / Stefan Frei. – ETH Zurich, 2009. – 184 p.
3. Rescorla, E. Is finding security holes a good idea? / E. Rescorla // Security and Privacy, Jan-Feb 2005. – P. 14-19.
4. Ozment, A. Improving Vulnerability Discovery Models: Problems with Definitions and Assumptions / A. Ozment // In the proceedings of the Third Workshop on Quality of Protection (QoP’ 07). October 29, 2007 - P. 6
5. Joh, H. Defining and Assessing Quantitative Security Risk Measures Using Vulnerability Lifecycle and CVSS Metrics / H. Joh, Y.K. Malaiya // SAM'11, The 2011 International Conference on Security and Management, 2011.- P.10-16.
6. Okamura, H. Quantitative Security Evaluation for Software System from Vulnerability Database / H. Okamura, M. Tokuzane, T. Dohi // Journal of Software Engineering and Applications, Vol. 6 No. 4A, 2013. – P. 15-23.
7. Frei, S. Large-scale vulnerability analysis / S. Frei, M. May, U. Fiedler, B. lattner // In LSAD '06: Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense. – New York, NY, USA, 2006. - P. 131–138
8. Липаев, В.В. Надежность программного обеспечения /В.В. Липаев – М.: Радио и связь, 1998. - 200 с.
9. Musa, J.D. Software Reliability: Measurement, Prediction, Application / J.D. Musa, A. lannino, K. Okumoto // N.Y. McGraw Hill, 1990. – 621 p.
10. Нестеров, С. Анализ и управление рисками в информационных системах на базе операционных систем Microsoft / С. Нестеров [Електронний ресурс]. – Режим доступу: http://www.intuit.ru/studies/courses/531/387/info.

11. Симонов, С. Анализ рисков, управление рисками / С. Симонов // Jet Info. Информационный бюллетень, 1999. - С. 2-28.

12. Симонов, С.В. Технологии и инструментарий для управления рисками / С.В. Симонов // Jet Info. - № 2. - 2003. – С. 3-32.

13. The logic behind CRAMM's assessment of measures of risk and determination of appropriate countermeasures [Електронний ресурс]. – Режим доступу: http://www.cramm.com/downloads/techpapers.htm.

14. Thomas R. Peltier. Information security risk analysis / R. Peltier Thomas. – Auerbach Pub, 2001. – 281 p.

15. OCTAVE (Operationally Critical Threat, Asset, and Vulnerability EvaluationSM) [Електронний ресурс] // CERT. – Режим доступу: www.cert.org/octave.

16. Using vulnerability assessment tools to develop an OCTAVE Risk Profile [Електронний ресурс] // SANS Institute. InfoSec Reading Room. – Режим доступу: http://www.sans.org/reading-room/whitepapers/auditing/vulnerability-assessment-toolsdevelop-octave-risk-profile-1353.

17. Щеглов, А.Ю. Безопасность современных ОС «в цифрах» [Електронний ресурс] / А. Ю. Щеглов. – Режим доступу: http://www.itsec.ru/ articles2 / Inf_security / bezopasnost-OS.

18. Застрожнов, И.И. Модель конфликта злоумышленника и системы защиты информации / И.И. Застрожнов, Д.И. Коробкин, А.А. Окрачков, Е.А. огозин. - Вестник Воронежского государственного технического университета. 2009. – С. 142-149.

19. Климов, И. З. Оценка надежности систем защиты информации от несанкционированного доступа / И. З. Климов, А. А. Пономарев. - Вестник Ижевского государственного технического университета. 2008. - С. 102-103.

20. Hong J.S. and Lie C.H. Joint reliability importance of two edges in undirected network. IEEE transaction on reliability 42 (1), 1993 - P.17-23

21. Gertsbakh I. B. and Shpungin Y. Combinatorial approach to component importance indexes in coherent systems. Probability in the Engineering and Information Sciences, 2011. – P..1-12

22. Rotshtein A., Shnaider E., Schneider M. and Kandel A. Fuzzy multicriterial selection of alternatives : The worst-case method, International journal of intelligent systems, 01/2010 - P. 948-957

23. Вялых, А.С. Оценка возможностей атаки на информационную систему / Вялых А.С., Вялых С.А. // Кибернетика и высокие технологии XXI века : матер. XII международ. науч.-тех. конф., Воронеж, 11-12 мая 2011 г. – Воронеж : ИПЦ ВГУ, 2011. – Т.1. – С. 91-96.

24. Вялых, А.С. Оценка уязвимости информационной системы на основе ситуационной модели динамики конфликта / А.С. Вялых, С.А. Вялых, А.А. Сирота // Информационные технологии. - 2012. - № 9. - С. 16-21.

25. Ротштейн А.П. Ранжирование элементов системы на основе нечетких отношений: метод наименьшего влияния [Електронний ресурс] А. П. Ротштейн. – Режим доступу: https://www.dependability.ru/jour/article/view/100/273

ДОДАТОК А

Лістинг програми:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace System_Ranging
{
    public partial class Form1 : Form
    {
        int start = 0;

        public Form1()
        {
            InitializeComponent();
            this.input_values.RowHeadersDefaultCellStyle.Padding =
            new Padding(this.input_values.RowHeadersWidth);
            input_values.RowPostPaint +=
            new DataGridViewRowPostPaintEventHandler(dataGridView1_RowPostPaint);

            this.min_i.RowHeadersDefaultCellStyle.Padding =
            new Padding(this.min_i.RowHeadersWidth);
            min_i.RowPostPaint +=
            new DataGridViewRowPostPaintEventHandler(dataGridView1_RowPostPaint);
            value_size.Value = 5;
            value_size.Value = 4;
        }

        static double[] finanses(double[,] mas)
        {
            double[] p = new double[mas.GetLength(0)];

            int div_max = 0;
            for (int i = 0; i < mas.GetLength(0); i++)
            {
                p[i] = 0;
                for (int j = 0; j < mas.GetLength(1); j++)
                {
                    p[i] += mas[i, j];
                }
                p[i] = Math.Round(p[i], 0);
                div_max += (int)p[i];

            }

            double[] answer = new double[p.Count()];

            for (int i = 0; i < answer.Count(); i++)
            {
                answer[i] = p[i] / div_max;
                answer[i] = Math.Round(answer[i], 4);
            }

            return answer;
        }

        static void RoundTo2(double[,] mas)
        {
```

```
    double[,] temp = mas;
    for (int i = 0; i < mas.GetLength(0); i++)
        for (int j = 0; j < mas.GetLength(1); j++)
        {
            double multiplier = Math.Pow(10, Convert.ToDouble(2));
            mas[i, j] = Math.Round(mas[i, j], 2);

        }
}

static bool array_check(int[,] mas)
{
    int check = 0;
    for (int i = 0; i < mas.GetLength(0); i++)
    {
        for (int j = 0; j < mas.GetLength(1); j++)
        {
            if (i == j)
            {
                check++;
            }
            else if (mas[i, j] >= 1 && mas[i, j] <= 9 && mas[i,j].GetType()==typeof(int))
            {
                check++;
            }
        }
    }
    if (check == Math.Pow(mas.GetLength(0), 2))
        return true;
    else
        return false;
}


static object[,] Find_Classes(double[,] mas)
{

    System.Collections.ArrayList unique = new System.Collections.ArrayList();

    for (int i = 0; i < mas.GetLength(0); i++)
        for (int j = 0; j < mas.GetLength(1); j++)
            if (unique.IndexOf(mas[i, j]) == -1)
                unique.Add(mas[i, j]);
    unique.Sort();
    double[] temp = new double[unique.Count];
    for (int i = 0; i < unique.Count; i++)
    {
        temp[i] = Convert.ToDouble(unique[i]);
    }
    string[] elements_classes = new string[temp.GetLength(0)];
    string bonus;
    for (int class_count = 0; class_count < elements_classes.GetLength(0); class_count++)
    {
        bonus = "";
        elements_classes[class_count] = "{";

        for (int i = 0; i < mas.GetLength(0); i++)
        {

            int help_count = 0;
            for (int j = 0; j < mas.GetLength(1); j++)
            {
                if (temp[class_count] == mas[i, j])
                    help_count++;
            }
```

```
                    if (help_count > 0)
                    {
                        elements_classes[class_count] += "x" + (i + 1) + ", ";

                    }
                    else
                        bonus += ", {x" + (i + 1) + "}";
                }
                elements_classes[class_count] =
elements_classes[class_count].Remove(elements_classes[class_count].Length - 2);
                elements_classes[class_count] += "}" + bonus;

            }

            int last = elements_classes.Count() - 1;
            elements_classes[last] = "";
            for (int i = 0; i < mas.GetLength(0); i++)
            {
                if (i != mas.GetLength(0) - 1)
                    elements_classes[last] += "{x" + (i + 1) + "}, ";
                else
                    elements_classes[last] += "{x" + (i + 1) + "}";
            }
            char search = Convert.ToChar("{");
            int[] result_count = new int[elements_classes.Count()];
            for (int i = 0; i < result_count.Count(); i++)
                for (int j = 0; j < elements_classes[i].Length; j++)
                    if (elements_classes[i][j] == search)
                        result_count[i]++;

            result_count[result_count.Count() - 1] = mas.GetLength(0);
            object[,] result = new object[elements_classes.GetLength(0), 3];
            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < elements_classes.GetLength(0); j++)
                {
                    switch (i)
                    {
                        case 0:
                            result[j, i] = unique[j];
                            break;

                        case 1:
                            result[j, i] = result_count[j];
                            break;

                        case 2:
                            result[j, i] = elements_classes[j];
                            break;
                    }
                }
            }

            return result;
        }



        static double[,] Composition(double[,] mas)
        {
            double[,] norm_mas = new double[mas.GetLength(0), mas.GetLength(1)];
            for (int i = 0; i < mas.GetLength(0); i++)
            {
                for (int j = 0; j < mas.GetLength(1); j++)
```

```
            {
                norm_mas[i, j] = 0;
            }
        }

        double[] min_value = new double[mas.GetLength(1)];
        double max_value = 0;
        for (int i = 0; i < mas.GetLength(0); i++)
        {
            for (int j = 0; j < mas.GetLength(1); j++)
            {
                if (i != j)
                {
                    for (int k = 0; k < mas.GetLength(0); k++)
                    {
                        if (mas[i, k] < mas[k, j])
                        {
                            min_value[k] = mas[i, k];
                        }
                        else
                        {
                            min_value[k] = mas[k, j];
                        }
                    }
                    max_value = min_value[0];
                    for (int k = 0; k < mas.GetLength(0) - 1; k++)
                    {
                        if (max_value < min_value[k + 1])
                        {
                            max_value = min_value[k + 1];
                        }
                    }
                    norm_mas[i, j] = max_value;

                }
                else
                    norm_mas[i, j] = 1;
            }
        }
        return norm_mas;

    }


    static double[,] Normalization(double[,] mas)
    {
        double[] max_value = new double[mas.GetLength(0)];
        double[,] result = new double[mas.GetLength(0), mas.GetLength(1)];
        for (int i = 0; i < mas.GetLength(0); i++)
        {
            max_value[i] = mas[i, 0];
            for (int j = 1; j < mas.GetLength(1); j++)
            {
                if (max_value[i] < mas[i, j])
                    max_value[i] = mas[i, j];

            }
        }

        for (int i = 0; i < mas.GetLength(0); i++)
        {
            for (int j = 0; j < mas.GetLength(1); j++)
            {
                result[i, j] = mas[i, j] / max_value[i];
            }
```

```
    }
    RoundTo2(result);
    return result;
}

static double[,] fuzzy_sim(double[,] mas)
{
    double[,] result = new double[mas.GetLength(0), mas.GetLength(1)];
    double sum = 0;
    for (int i = 0; i < mas.GetLength(0); i++)
    {
        for (int j = 0; j < mas.GetLength(1); j++)
        {
            if (i != j)
            {
                for (int k = 0; k < mas.GetLength(0); k++)
                {
                    sum += Math.Abs(mas[i, k] - mas[j, k]);
                }
                sum = sum / mas.GetLength(0);
                sum = 1 - sum;
                result[i, j] = sum;
                sum = 0;
            }
            else
                result[i, j] = 1;
        }
    }
    RoundTo2(result);

    return result;


}


private void value_size_ValueChanged(object sender, EventArgs e)
{
    start++;
    input_values.Rows.Clear();
    input_values.Columns.Clear();
    min_i.Rows.Clear();
    for (int i = 0; i < value_size.Value; i++)
    {
        DataGridViewTextBoxColumn col = new DataGridViewTextBoxColumn();
        col.Width = 30;
        col.Name = "x" + (i + 1);
        input_values.Columns.Add(col);
        input_values.Rows.Add();
        input_values.Rows[i].HeaderCell.Value = "x" + (i + 1);
        min_i.Rows.Add();
        min_i.Rows[i].HeaderCell.Value = "x" + (i + 1);

    }

    for (int i = 0; i < input_values.Columns.Count; i++)
    {
        input_values.Rows[i].Cells[i].Value = 0;
        input_values.Rows[i].Cells[i].ReadOnly = true;
    }

}

void dataGridView1_RowPostPaint(object sender, DataGridViewRowPostPaintEventArgs e)
{
```

```
            object o = input_values.Rows[e.RowIndex].HeaderCell.Value;

            e.Graphics.DrawString(
                o != null ? o.ToString() : "",
                input_values.Font,
                Brushes.Black,
                new PointF((float)e.RowBounds.Left + 2, (float)e.RowBounds.Top + 4));
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Result_Table.Rows.Clear();
            int[,] input_array = new int[input_values.Rows.Count, input_values.Columns.Count];

            for (int x = 0; x < input_array.GetLength(0); x++)
                for (int i = 0; i < input_array.GetLength(1); i++)
                {
                    input_array[x, i] = Convert.ToInt32(input_values.Rows[x].Cells[i].Value);
                }

            if (!array_check(input_array))
            {
                MessageBox.Show("Проверьте массив со входными данными!\n(Значения должны быть
целочисленными и в диапазон [1;9])");
            }
            else
            {


            int[] min_inf = new int[min_i.Rows.Count];

            for (int i = 0; i < min_inf.Length; i++)
            {
                min_inf[i] = Convert.ToInt32(min_i.Rows[i].Cells[0].Value);
            }

            double[,] fuz_rel = new double[input_array.GetLength(0), input_array.GetLength(1)];

            double sum = 0;
            int helper = 0;
            for (int i = 0; i < input_array.GetLength(0); i++)
            {
                for (int j = 0; j < input_array.GetLength(1); j++)
                {
                    if (i != j)
                    {
                        if (j == min_inf[i] && helper == 0)
                        {
                            for (int k = 0; k < input_array.GetLength(0); k++)
                            {
                                sum += input_array[i, k] / input_array[i, j];
                            }
                            sum = Math.Pow(sum, -1);
                            fuz_rel[i, j] = sum;
                            j = 0;
                            helper++;
                        }
                        else
                            fuz_rel[i, j] = sum * input_array[i, j];
                    }
                }
                helper--;
            }

            double[,] norm_ratio = Normalization(fuz_rel);
```

```
            RoundTo2(norm_ratio);

            double[,] ham_dist = fuzzy_sim(norm_ratio);
            RoundTo2(ham_dist);
            double[,] fuzzy_tranz;

            double[,] temp = ham_dist;

            int check, iter = 0;
            do
            {
                check = 0;
                fuzzy_tranz = Composition(temp);

                for (int i = 0; i < fuzzy_tranz.GetLength(0); i++)
                    for (int j = 0; j < fuzzy_tranz.GetLength(1); j++)
                        if (fuzzy_tranz[i, j] != temp[i, j])
                            check++;
                temp = fuzzy_tranz;
                iter++;
                if (iter > 1000)
                {
                    MessageBox.Show("Внимание! Количество транзитывных замиканий превысило
1000.\nПроверьте значения входных значений");
                    return;
                }
            }
            while (check != 0);

            object[,] result = Find_Classes(fuzzy_tranz);

            for (int i = 0; i < result.GetLength(0); i++)
            {
                Result_Table.Rows.Add();
                for (int j = 0; j < result.GetLength(1); j++)
                {
                    Result_Table[j, i].Value = result[i, j];
                }
            }

            double[] results = finanses(fuzzy_tranz);


            int C0_Value=Convert.ToInt32(C0.Text);

            for (int i = 0; i < results.Count(); i++)
            {
                Financee.Rows.Add();
                Financee[0, i].Value = (i + 1);
                Financee[1, i].Value = results[i];
                Financee[2, i].Value = C0_Value;
                Financee[3, i].Value = results[i] * C0_Value;
            }
            }

        }
        private void button2_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Для выполнения ранжирования элементов системы необходимо:"
                + "\n1)Выбрать количество элементов"
                + "\n2)Ввести експертную оценку влияния элементов друг на друга"
                + "\n3)Ввести индексы элементов с найменшим влияниям на другие элементы"
                + "\n(например: для x1-индекс 2, это значит, что x1 найменее влияет на x2)"
                + "\n4)Нажать на кнопку `Расчитать`");
        }
```

```
        private void min_i_CellContentClick(object sender, DataGridViewCellEventArgs e)
        {
            int help;
            if (start > 0)
            {
                for (int i = 0; i < input_values.Rows.Count; i++)
                {
                    for (int j = 0; j < input_values.Columns.Count; j++)
                    {
                        if (i != j)
                            input_values.Rows[i].Cells[j].ReadOnly = false;
                    }
                }
                for (int i = 0; i < input_values.Columns.Count; i++)
                {
                    if (min_i.Rows[i].Cells[0].Value != null)
                    {
                        help = Convert.ToInt32(min_i.Rows[i].Cells[0].Value);
                        if (help <= 0 || help > (input_values.Columns.Count))
                        {
                            MessageBox.Show("Введено некоректное значение!");
                            min_i.Rows[i].Cells[0].Value =
min_i.Rows[i].Cells[0].DefaultNewRowValue;
                            break;
                        }
                        else
                        {
                            help--;
                            if (i != help)
                            {
                                input_values.Rows[i].Cells[help].Value = 1;
                                input_values.Rows[i].Cells[help].ReadOnly = true;
                            }
                            if (i == help || help < 0)
                            {
                                MessageBox.Show("Ошибка ввода элемента " + (i + 1) + "!",
"Ошибка");
                                min_i.Rows[i].Cells[0].Value =
min_i.Rows[i].Cells[0].DefaultNewRowValue;
                                break;
                            }
                        }
                    }
                }
            }
        }
        private void button3_Click(object sender, EventArgs e)
        {
            input_values.Rows.Clear();
            input_values.Columns.Clear();
            min_i.Rows.Clear();
            Result_Table.Rows.Clear();
            value_size.Value = 4;
            Financee.Rows.Clear();
        }
        private void numericUpDown1_ValueChanged(object sender, EventArgs e)
        {
            int C0_Value=Convert.ToInt32(C0.Text);
            for (int i = 0; i < Financee.Rows.Count; i++)
            {
                Financee[2, i].Value = C0.Value;
                Financee[3, i].Value = C0_Value * Convert.ToDouble(Financee[1, i].Value);
            }
        }
    }}
```

ДОДАТОК Б

Приклад роботи комп’ютерної програми

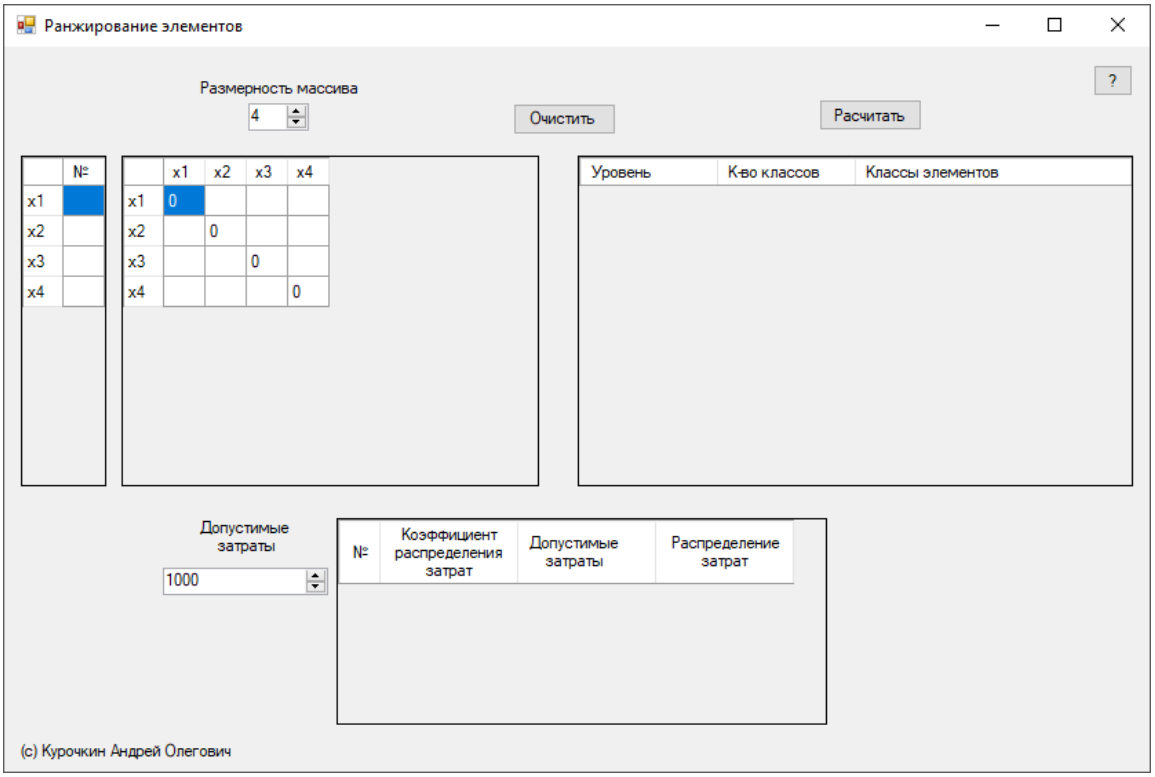

Рисунок Б.1 – Запуск програми

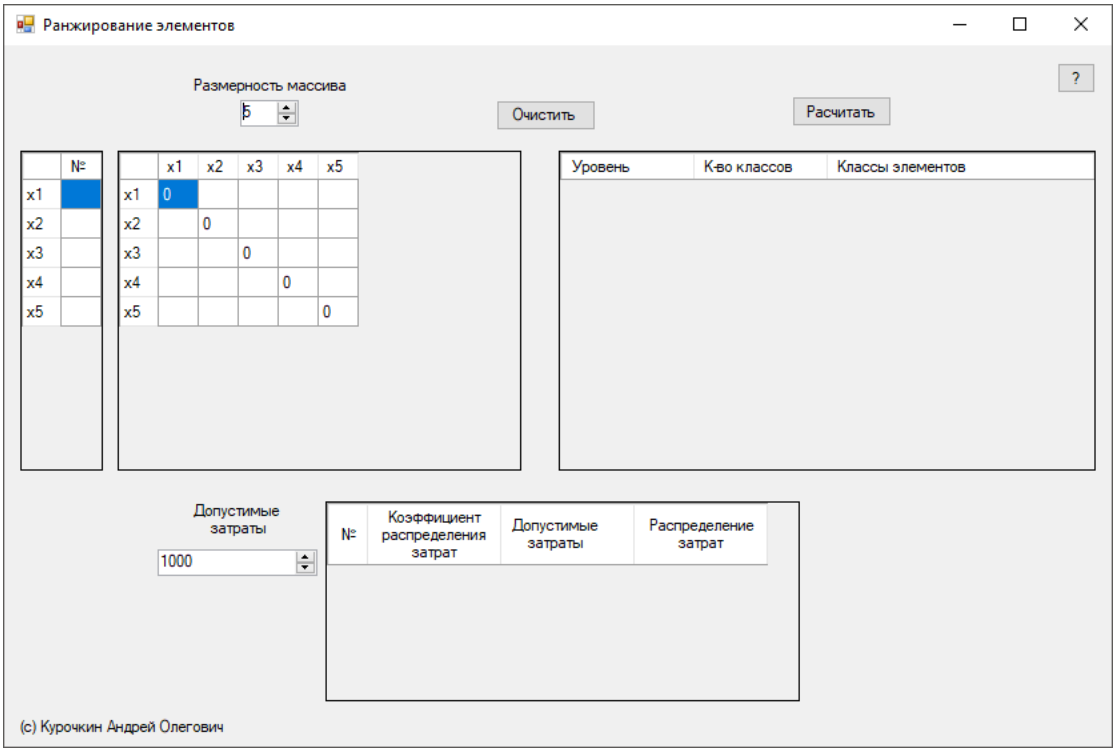

Рисунок Б.2 – Введення кількості баз даних

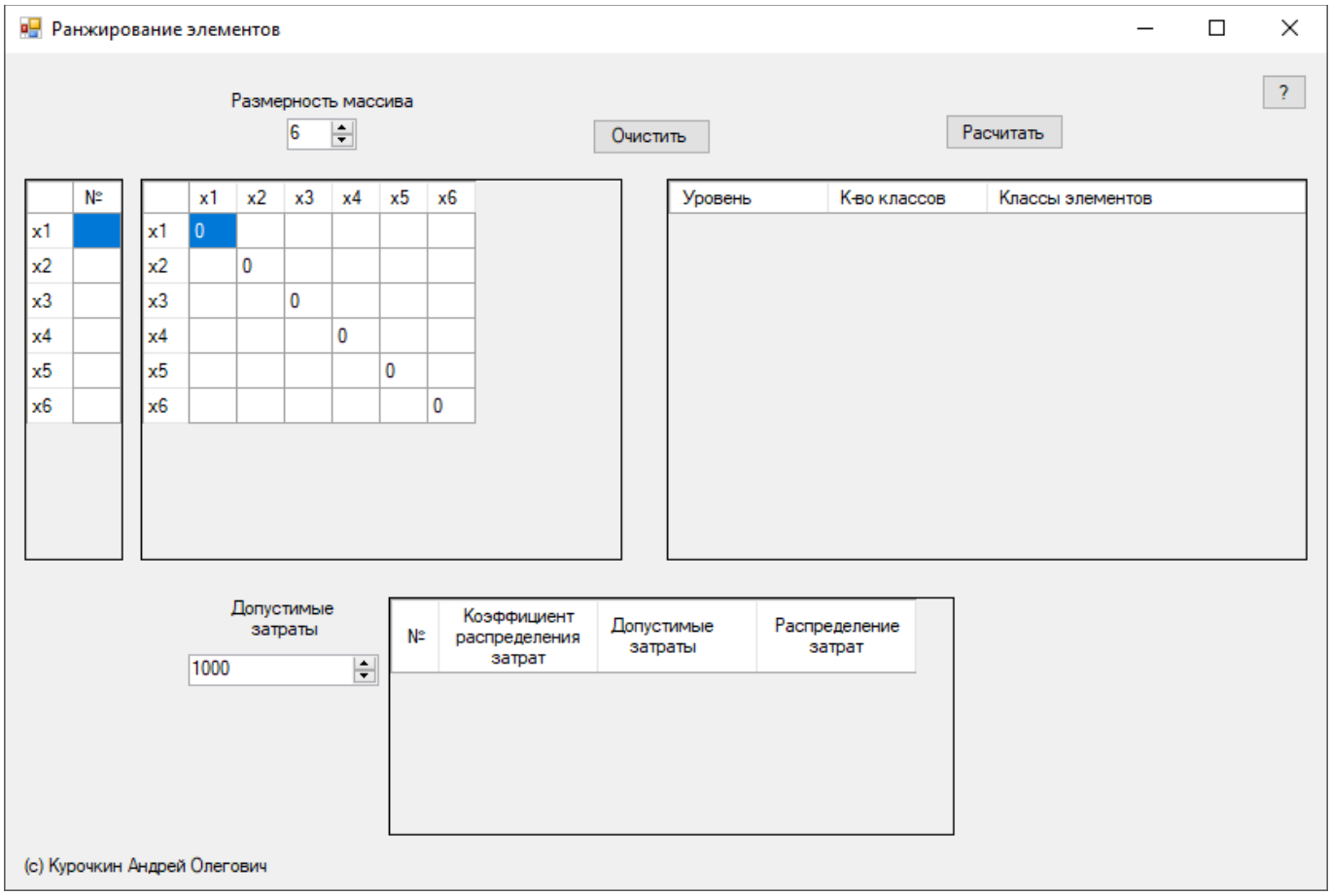

Рисунок Б.3 - Введення кількості баз даних

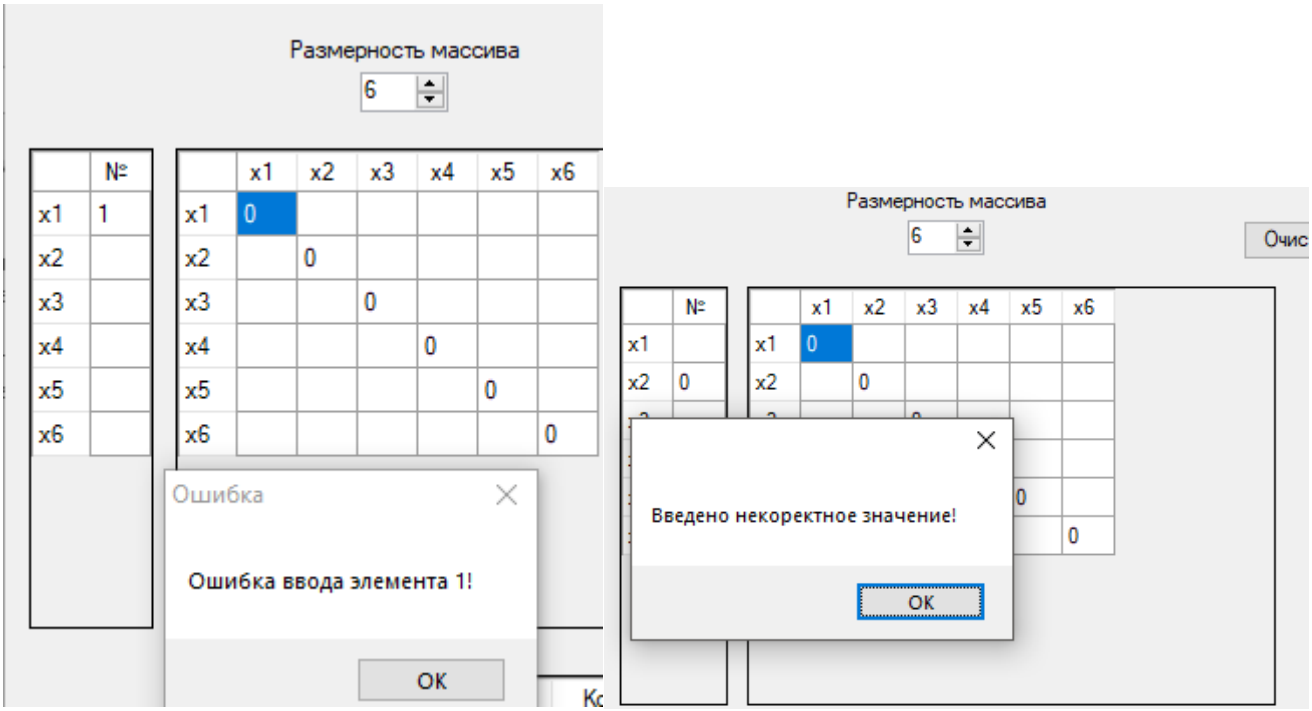

Рисунок Б.4 - Введення некоректного числа індексу найменшого впливу

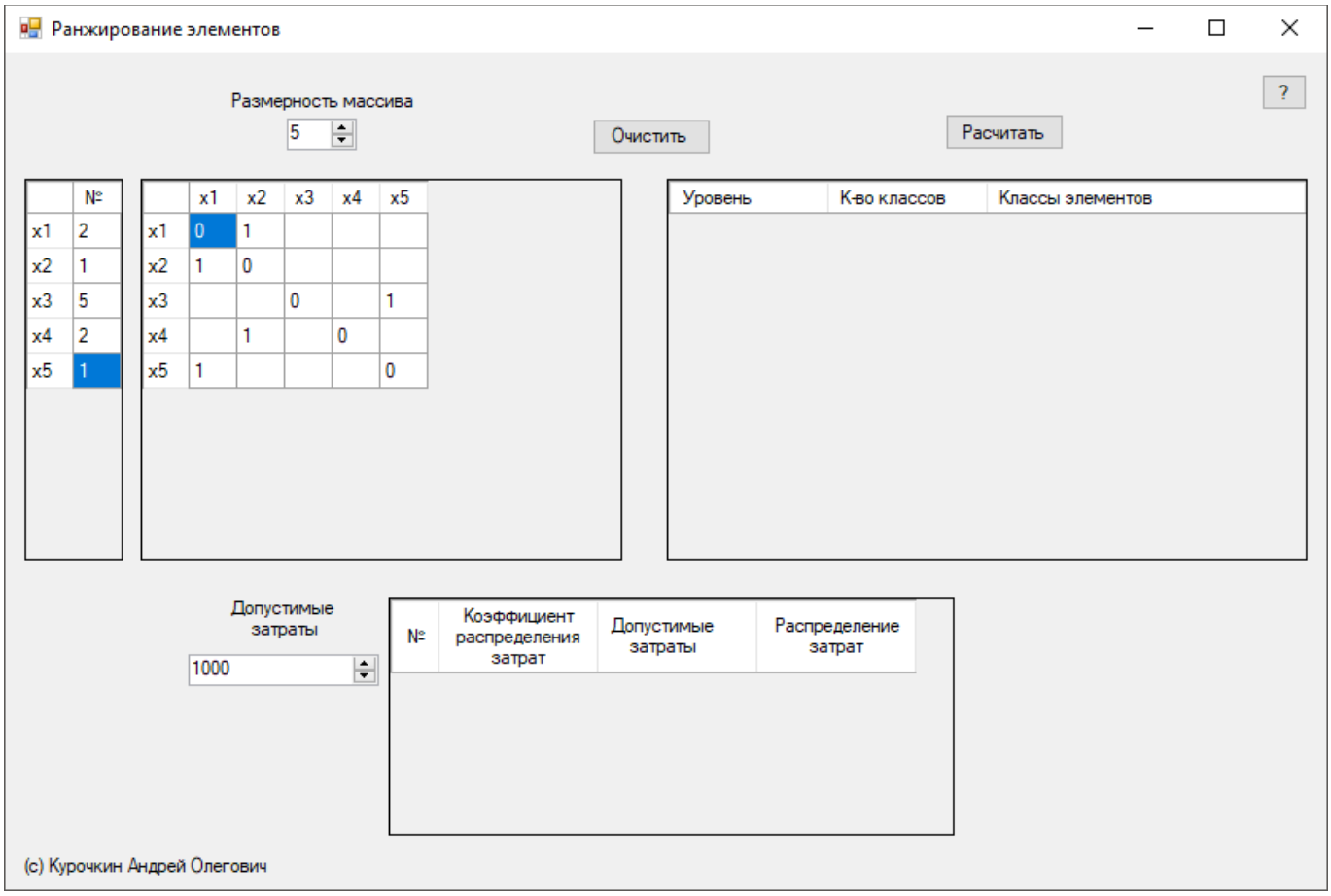

Рисунок Б.5 - Введення коректного числа індексу найменшого впливу

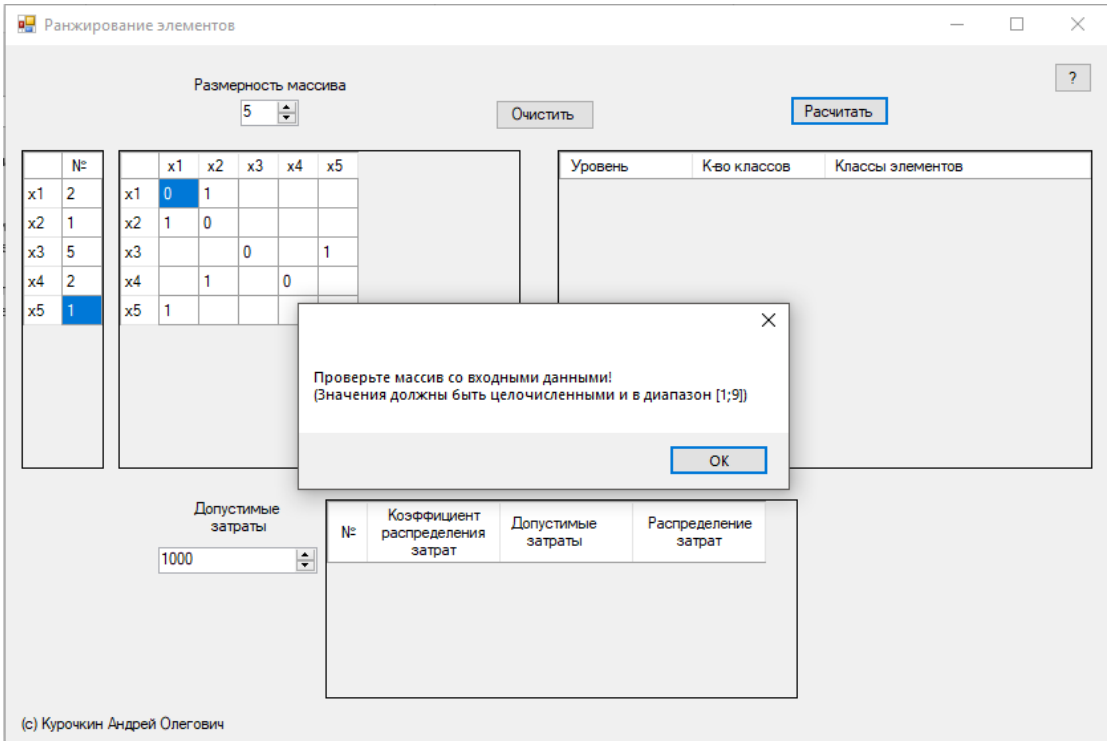

Рисунок Б.6 – Натиснення на кнопку «Расчитать» із неправильно заповненим масивом вхідних значень

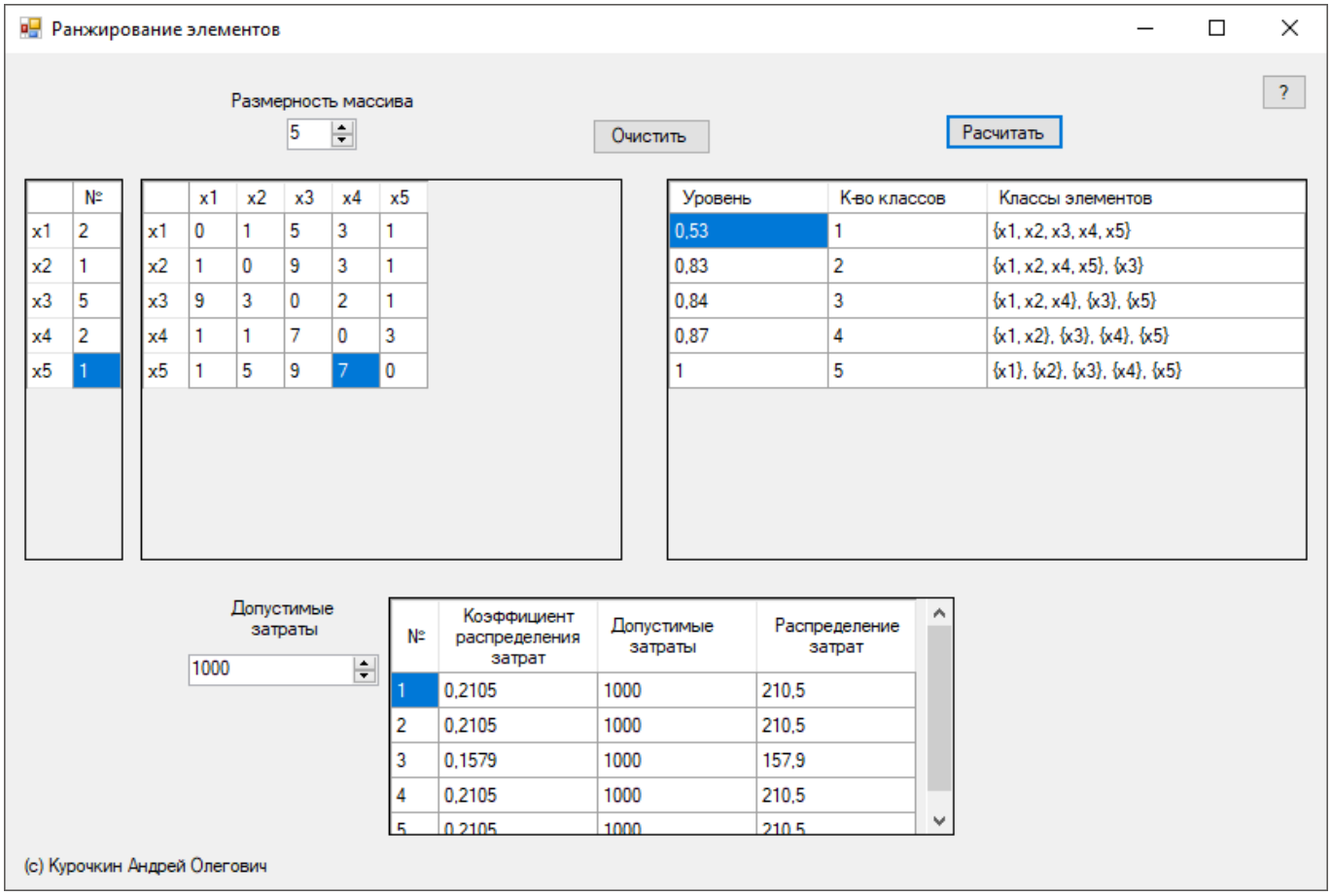

Рисунок Б.7 - Натиснення на кнопку «Расчитать» із правильно заповненим масивом

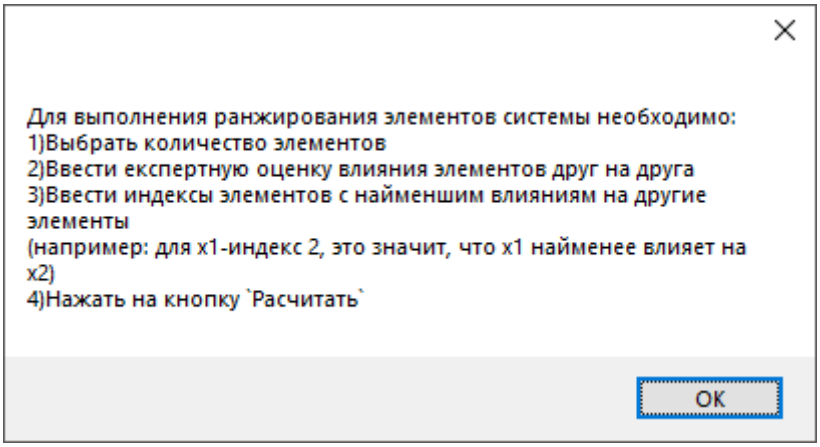

Рисунок Б.8 – Натиснення на кнопку «?»

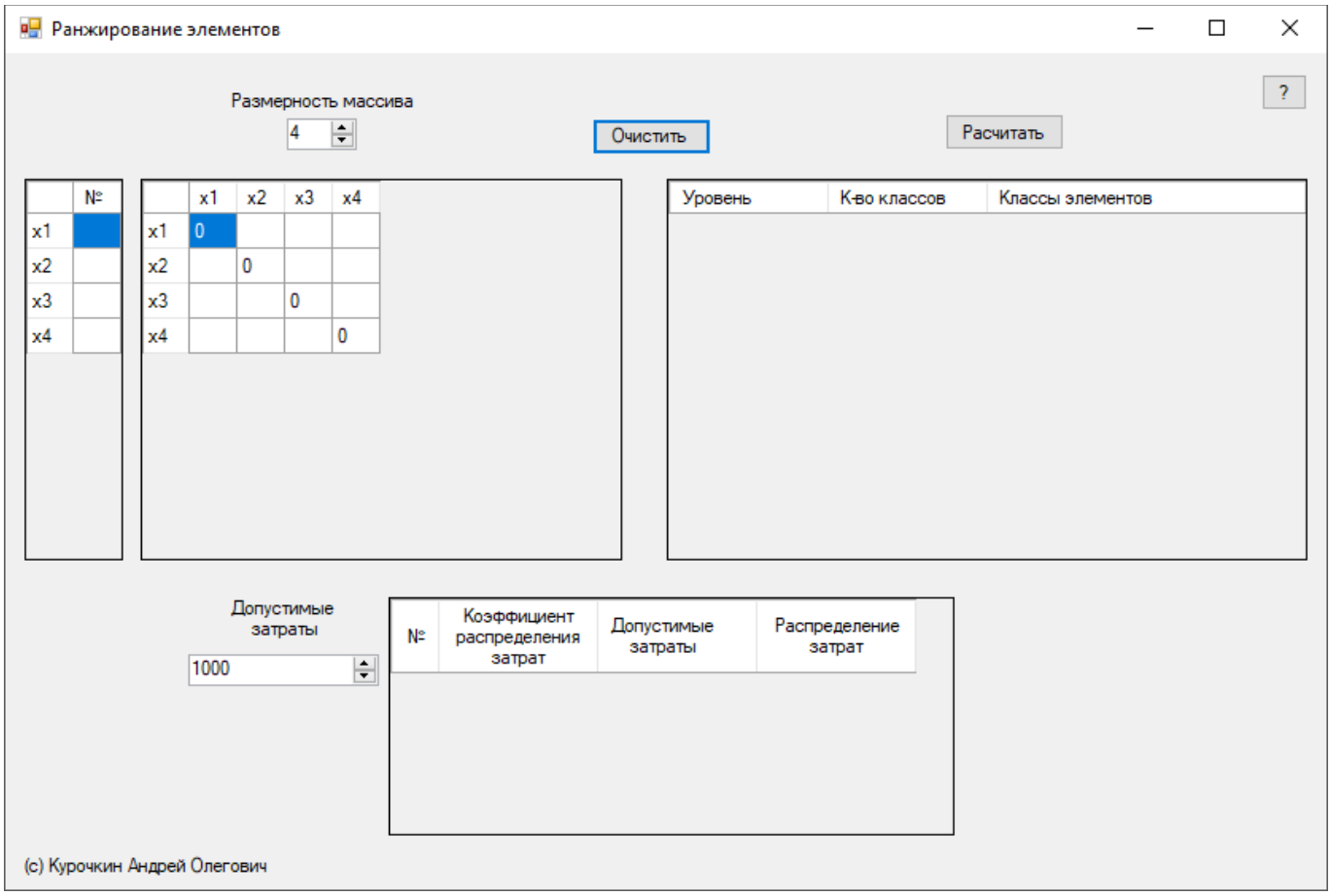

Рисунок Б.9 - Натиснення на кнопку «Очистить»

ДОДАТОК В

Комп’ютерні експерименти

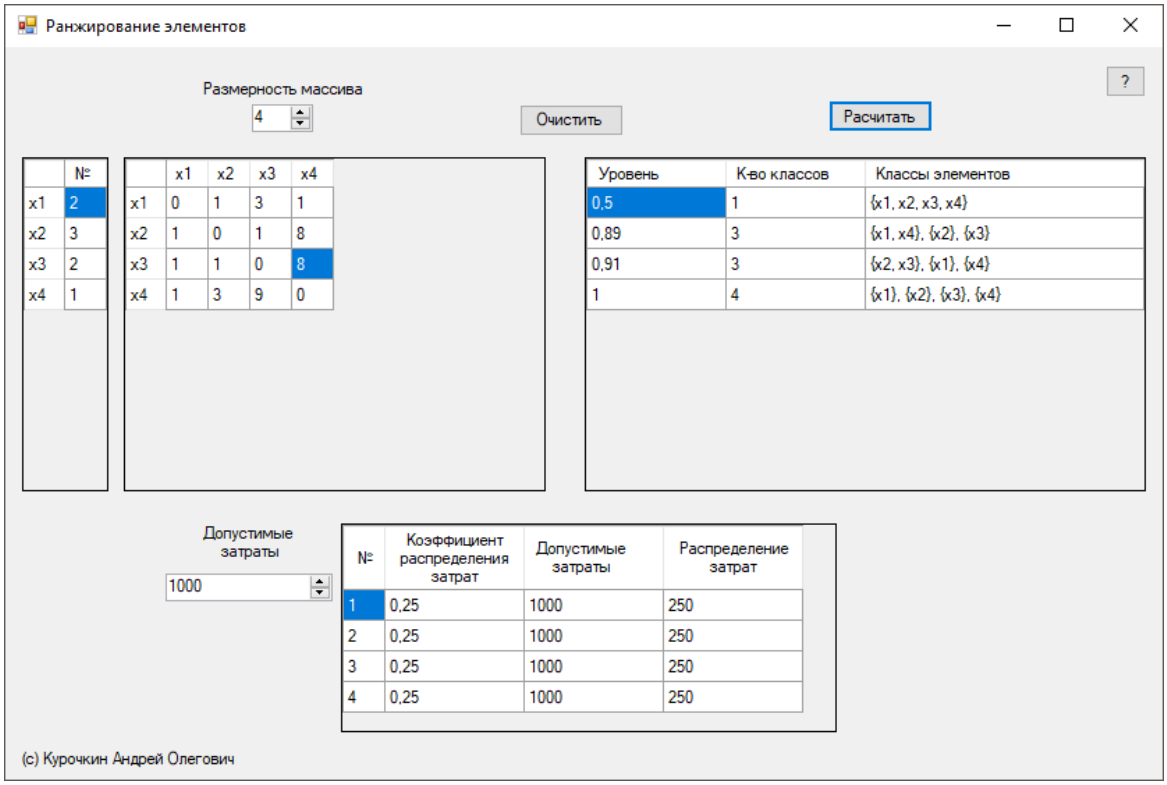

Рисунок В.1 – Виконання експерименту із кількістю баз даних 4

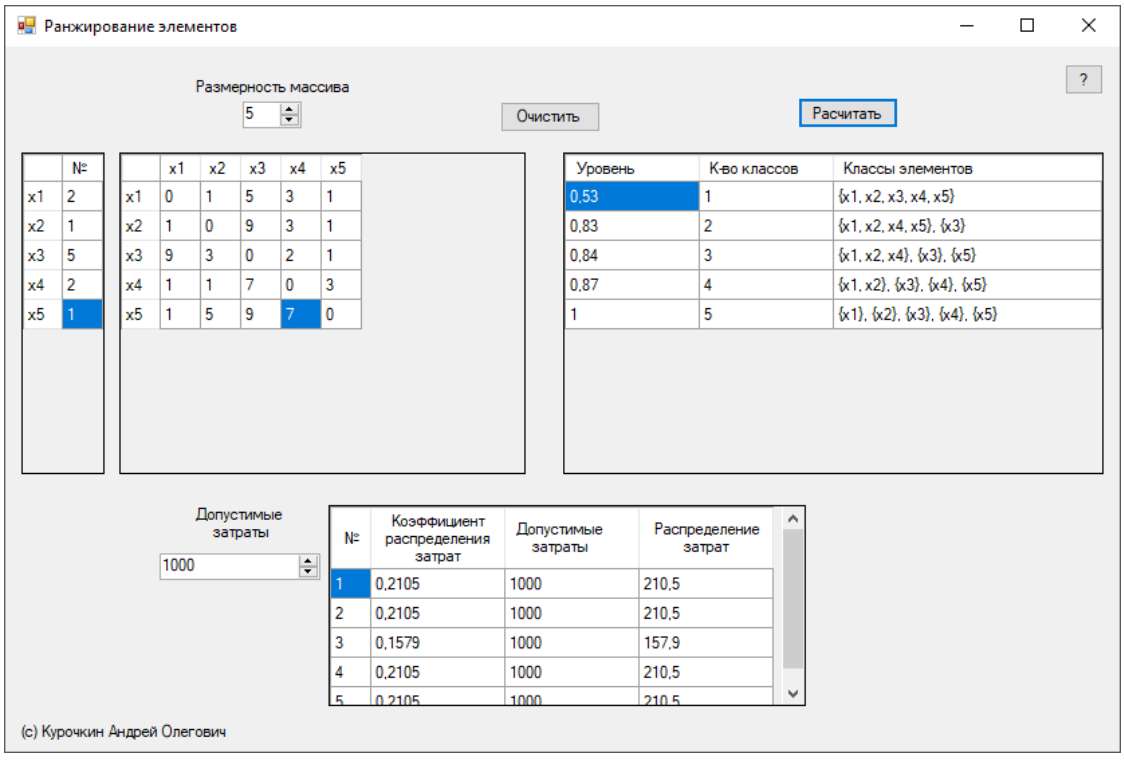

Рисунок В.2 - Виконання експерименту із кількістю баз даних 5

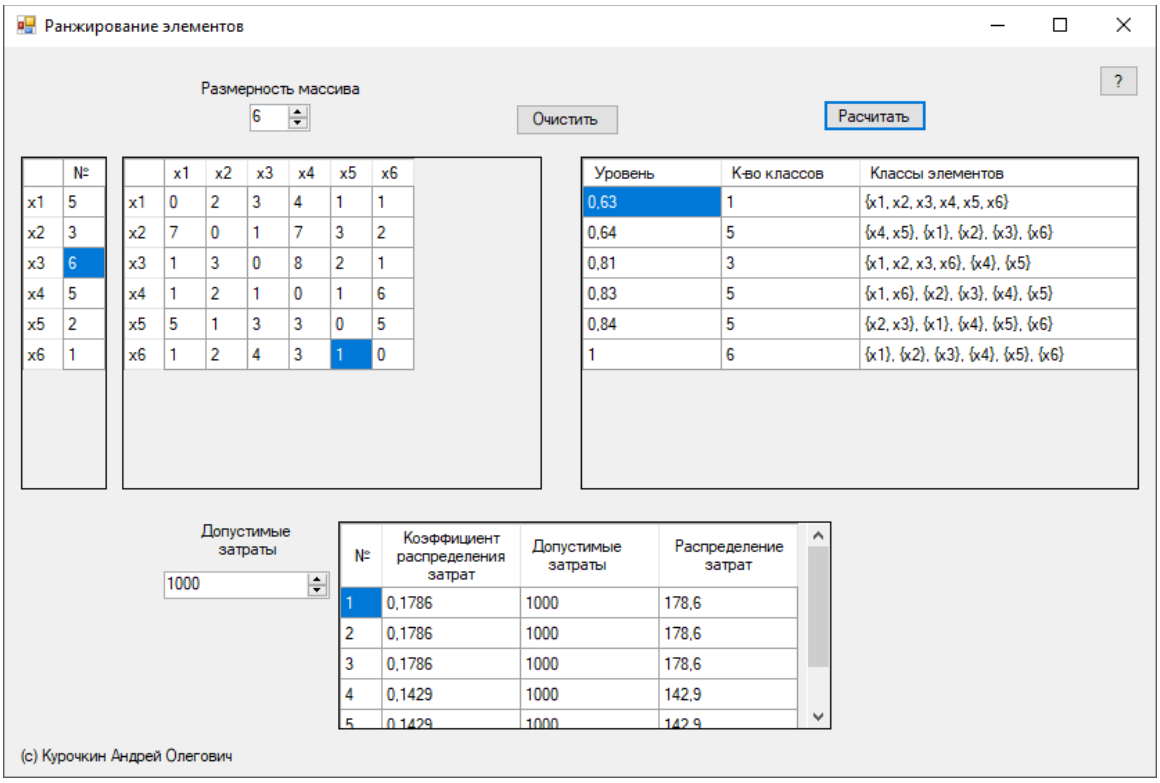

Рисунок В.3 - Виконання експерименту із кількістю баз даних 6

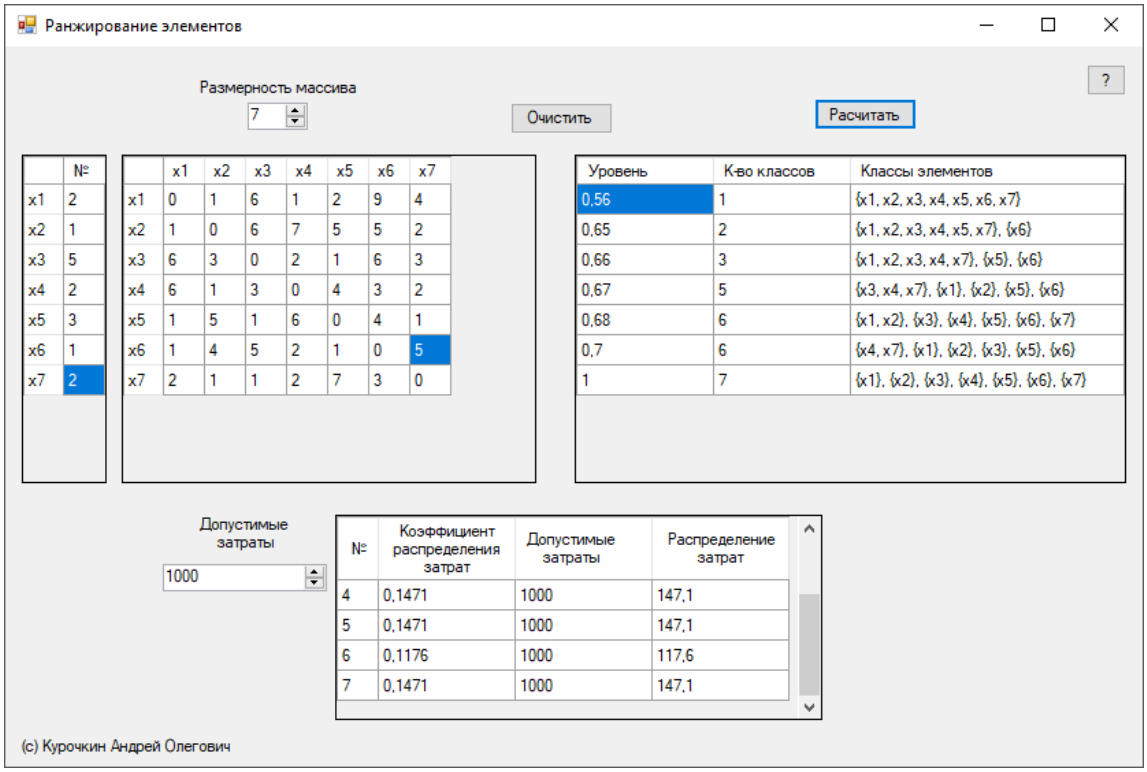

Рисунок В.4 - Виконання експерименту із кількістю баз даних 7

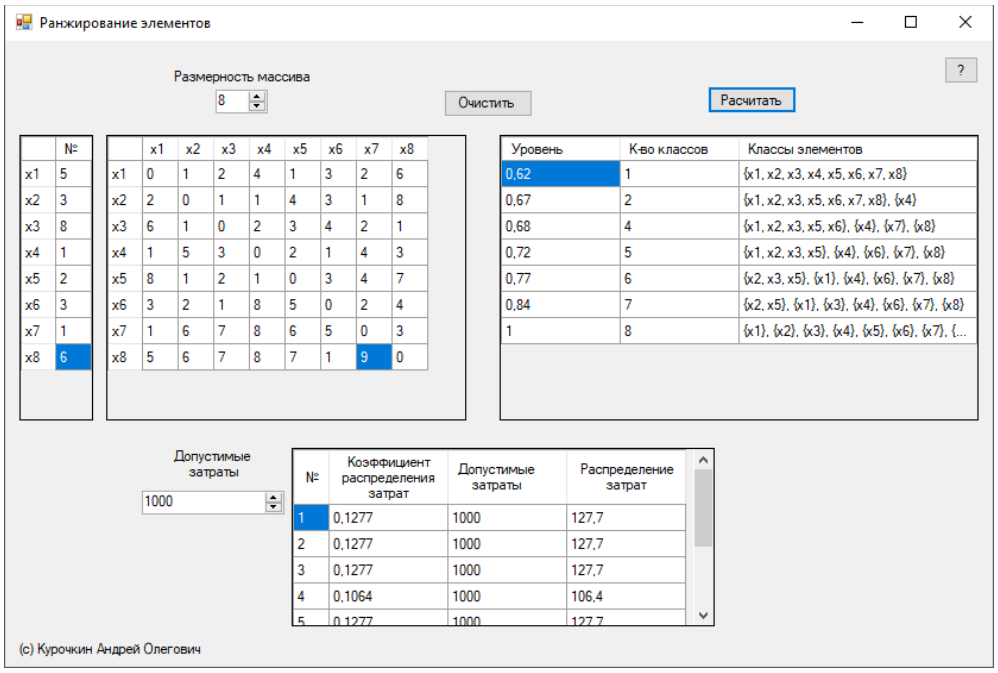

Рисунок В.5 - Виконання експерименту із кількістю баз даних 8

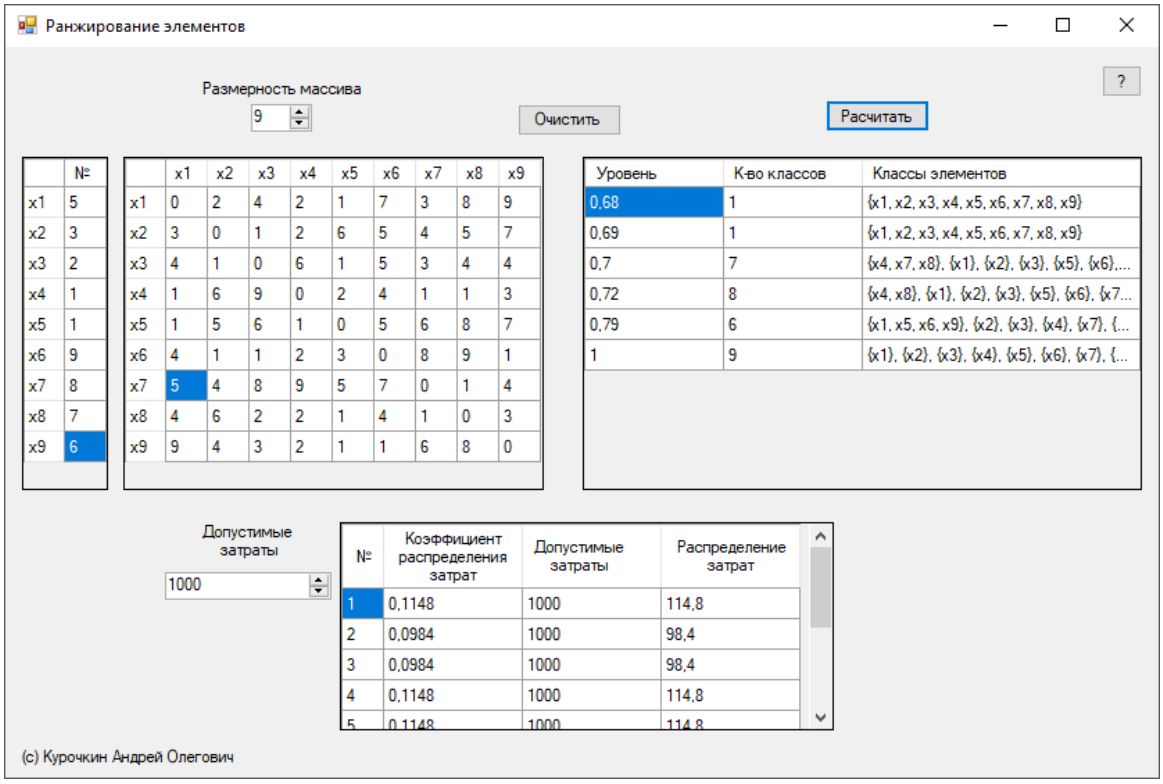

Рисунок В.6 - Виконання експерименту із кількістю баз даних 8

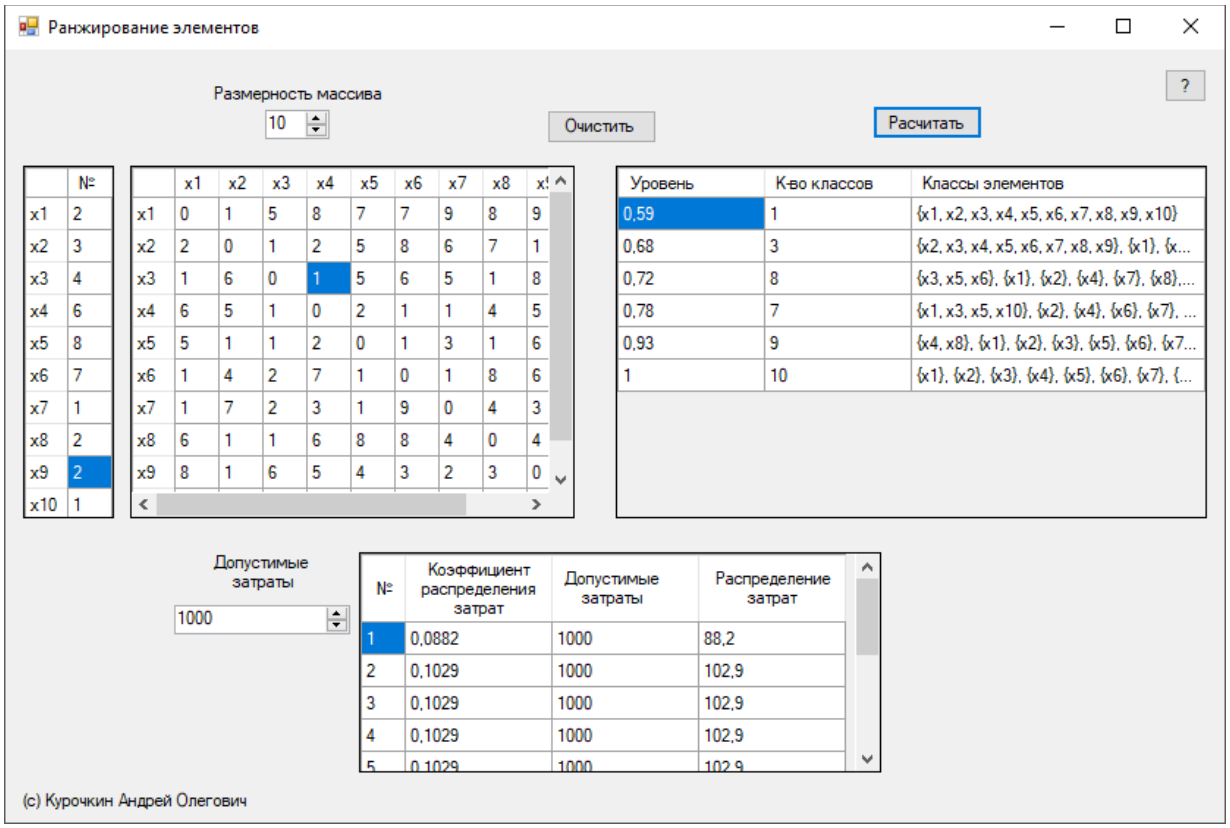

Рисунок В.7 - Виконання експерименту із кількістю баз даних 10

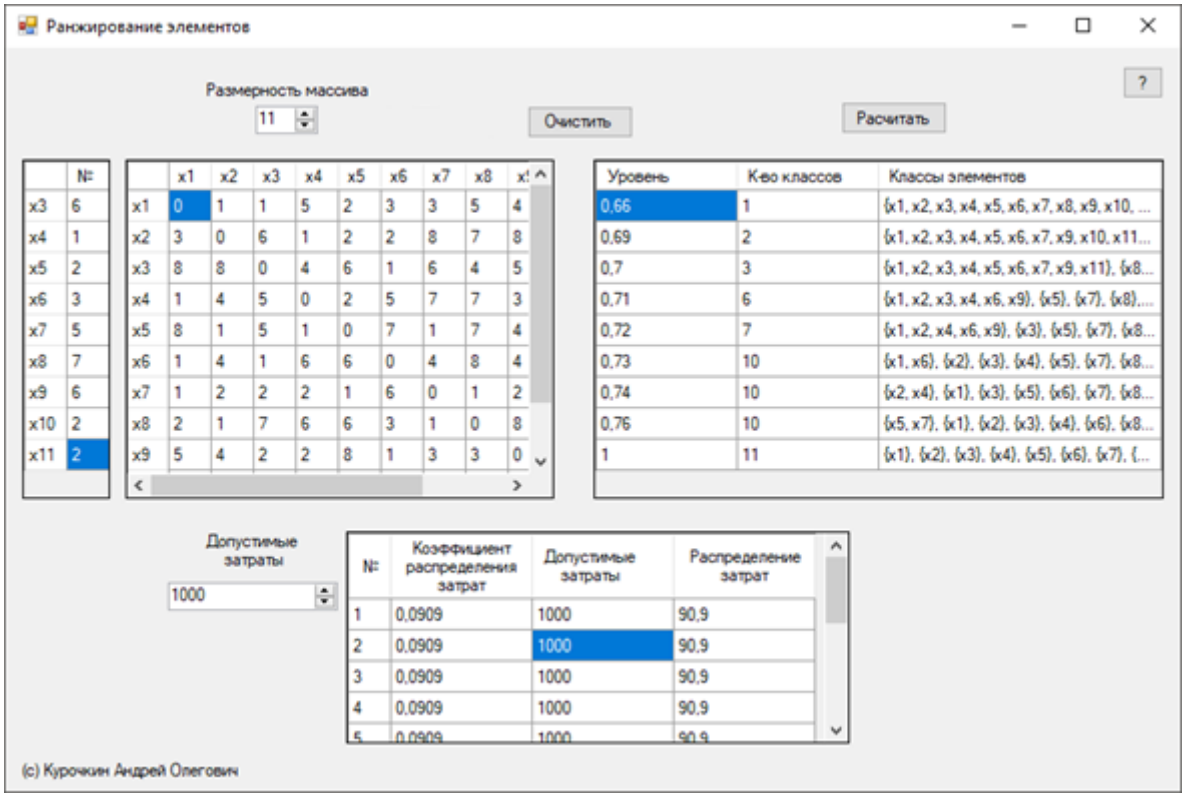

Рисунок В.8 - Виконання експерименту із кількістю баз даних 11

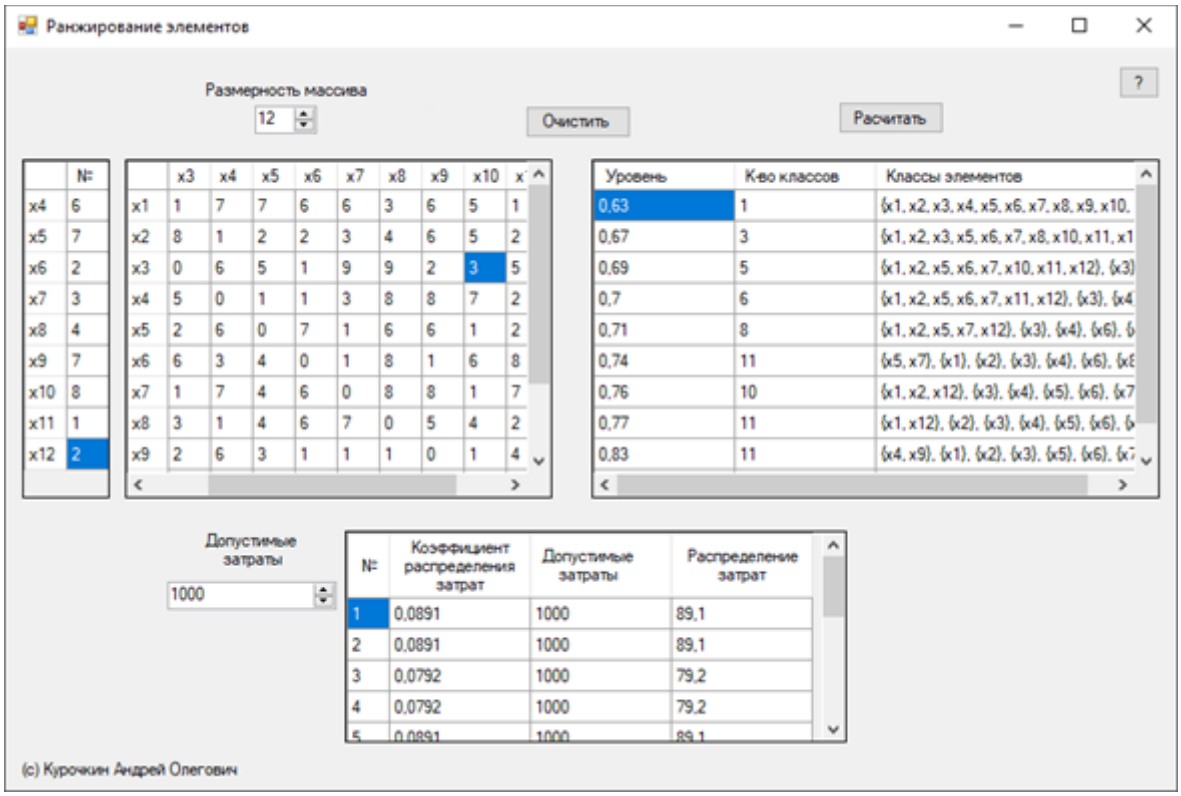

Рисунок В.9 - Виконання експерименту із кількістю баз даних 12

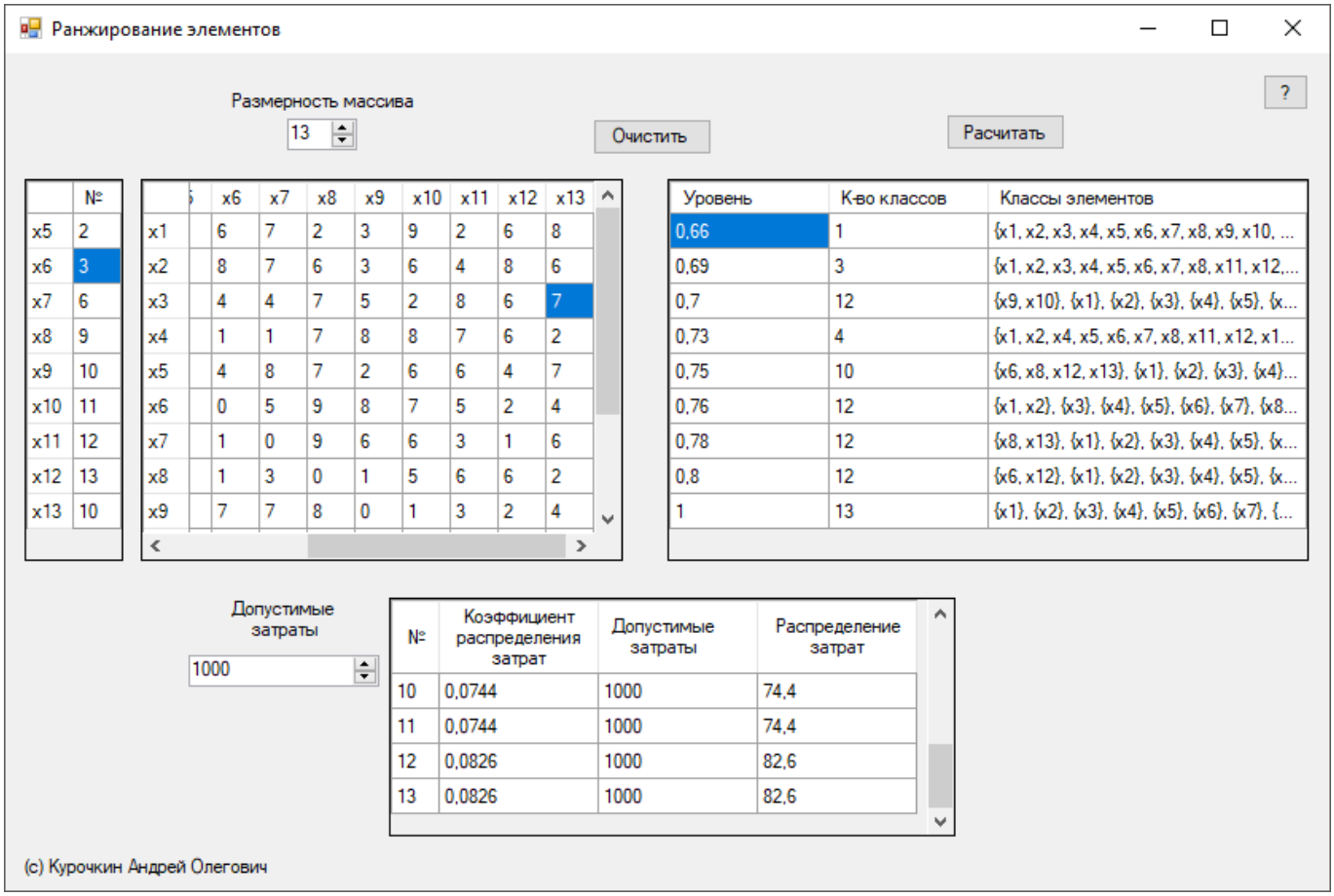

Рисунок В.10 - Виконання експерименту із кількістю баз даних 13

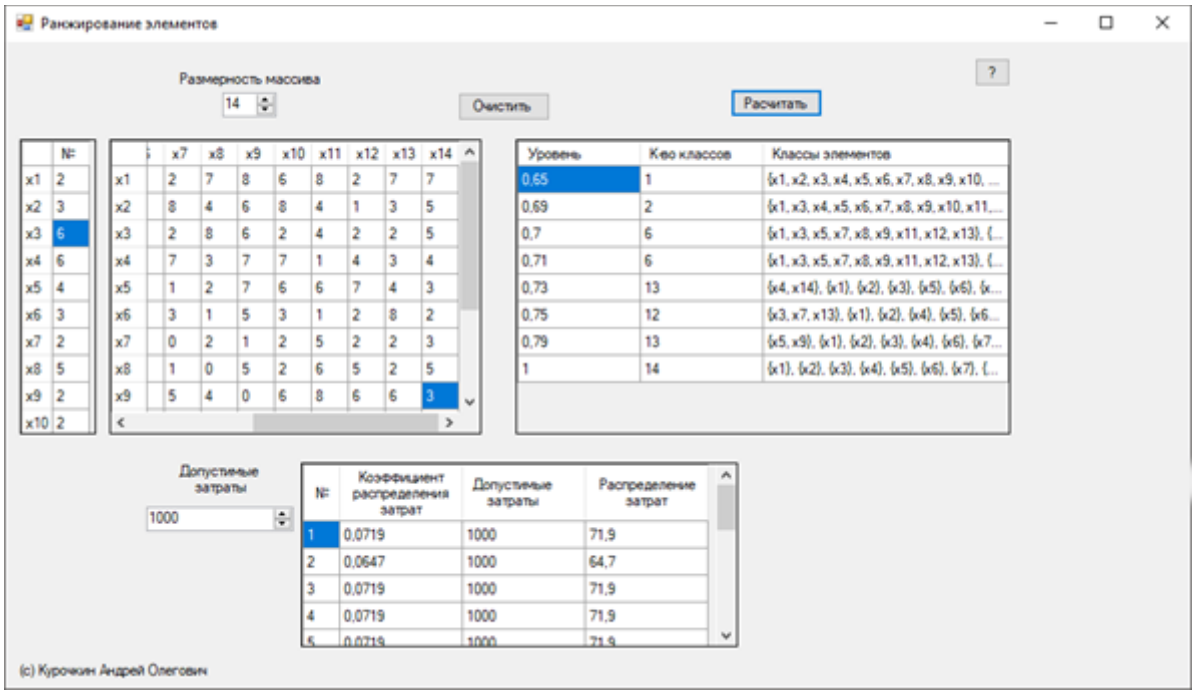

Рисунок В.11 - Виконання експерименту із кількістю баз даних 14

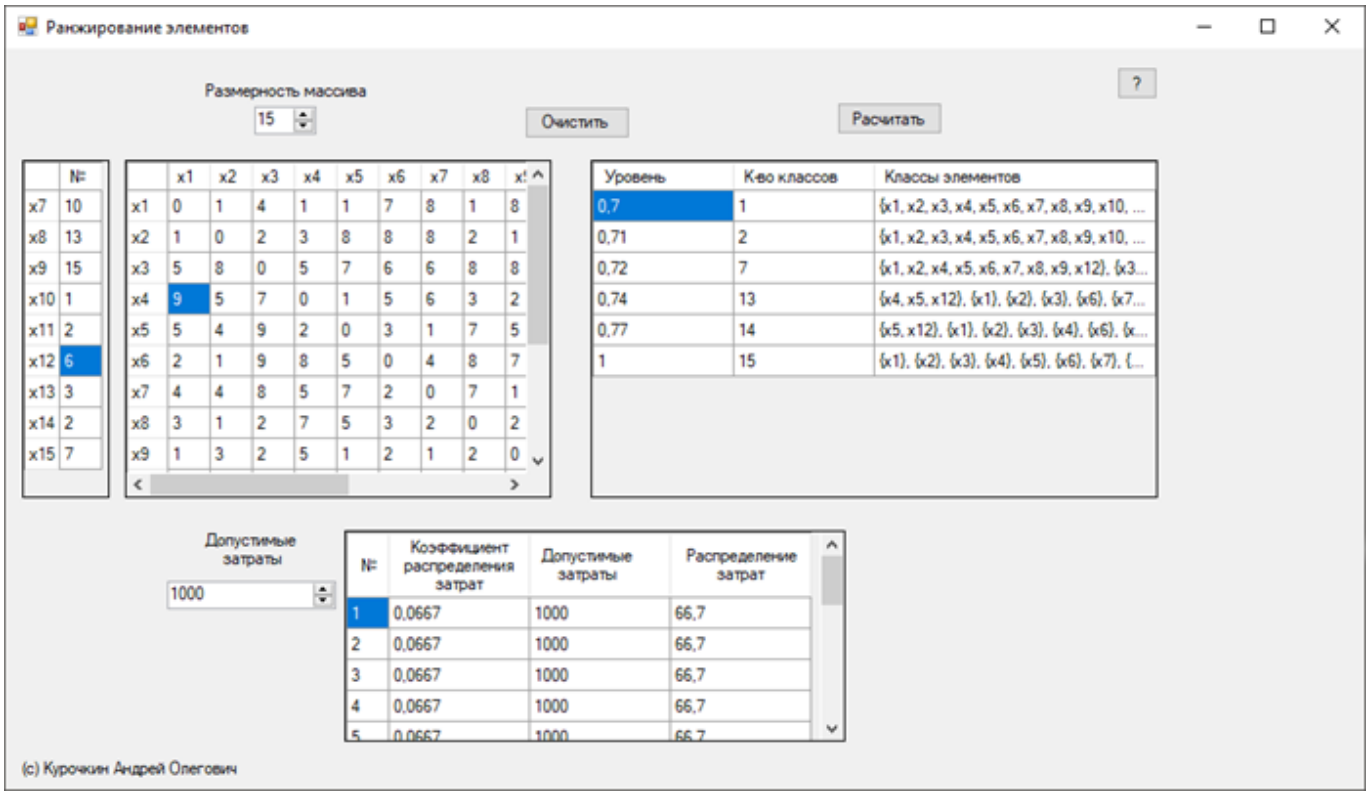

Рисунок В.12 - Виконання експерименту із кількістю баз даних 15