РОЗДІЛ 4
ПРОГРАМНІ ЗАСОБИ ІДЕНТИФІКАЦІЇ ПРИХОВАНИХ ПОМИЛОК ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1.Структура системи ідентифікації прихованих помилок програмного забезпечення

На основі розробленого методу та алгоритмів ідентифікації прихованих помилок на базі НІТ розроблено структурну схему системи [131, 132] ідентифікації прихованих помилок програмного забезпечення (рис.4.1).

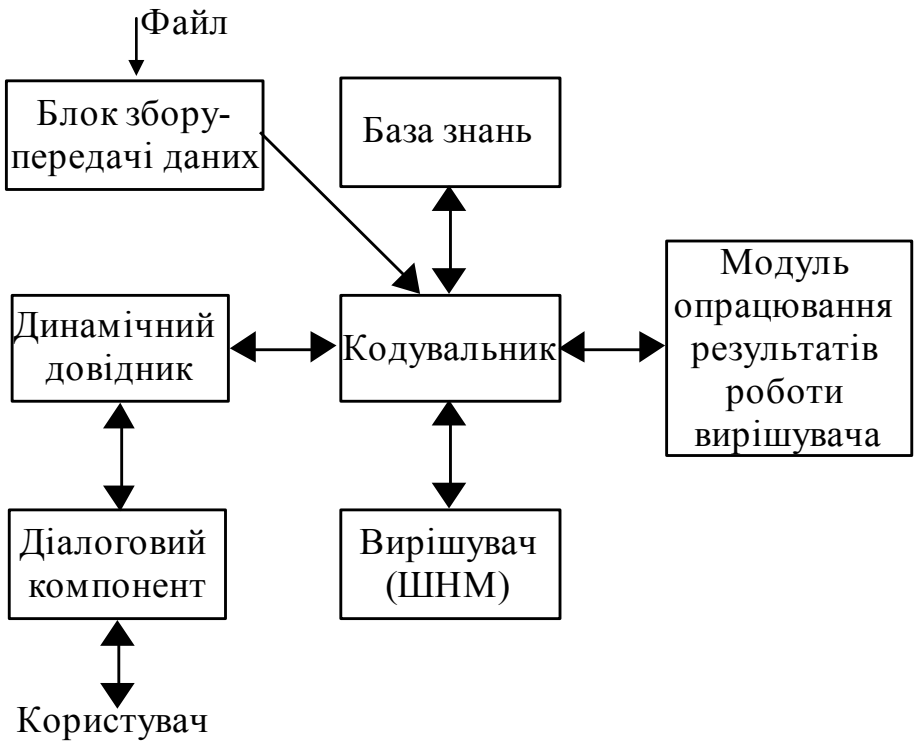

Рис. 4.1 Структурна схема системи ідентифікації прихованих помилок програмного забезпечення

Система ідентифікації прихованих помилок програмного забезпечення складається з наступних компонентів:

1) блок збору – передачі даних – підключає наданий користувачем файл з результатами основного тестування, представленими у вигляді журналу “Метод тестування – Операція тестування – Тип виявленої помилки”;

2) кодувальник – виконує перетворення вхідних даних з лінгвістичної форми представлення в кількісну форму за допомогою таблиць 1-3 бази знань (додаток А), заповнення таблиці 5 бази знань (додаток А) вхідними даними та формування вхідних векторів для модуля вирішувача. Кодувальник також перевіряє вхідні дані на правильність та достатність при формуванні вхідних векторів; якщо дані недостатні або вони невірні, то кодувальник передає динамічному довіднику своє повідомлення з пропозицією сформувати ще один файл такої ж форми, як і попередній, з додатковими результатами, що перетворюються в кількісну форму аналогічно даним основного файлу, після чого заносяться в базу знань. Здійснюється заповнення бази знань вихідними даними, перетворення результуючих векторів вирішувача з кількісної в лінгвістичну форму за допомогою таблиці 4 бази знань (додаток А) та передачу їх модулю опрацювання результатів роботи вирішувача;

3) база знань – містить таблиці присвоєння номерів методам і операціям основного тестування, типам виявлених помилок та присвоєння номерів рівням категорійності прихованих помилок; таблицю кількісного представлення вхідних даних, в якій містяться вхідні дані, перетворені кодувальником в кількісну форму; таблицю текстового представлення результуючих векторів вирішувача (ШНМ), в якій представлені результуючі вектори, перетворені кодувальником в лінгвістичну форму; таблиці відповідності методу основного тестування, операцій основного тестування, типів виявлених під час основного тестування помилок, відповідності між номером методів тестування ПЗ та рівнем категорійності прихованих помилок ПЗ, відповідності між операціями тестування ПЗ та рівнем категорійності прихованих помилок, на основі яких система формує висновок про метод, яким рекомендується проводити повторне тестування прикладного ПЗ, а також правила для формування висновку про необхідність та метод повторного тестування;

4) вирішувач – штучна нейронна мережа, на входи якої подається інформація про методи і операції основного тестування та типи виявлених під час основного тестування помилок, а на виході одержується рівень категорійності прихованих помилок;

5) модуль опрацювання результатів роботи вирішувача – на основі правил та таблиці результатів роботи вирішувача, взятих з бази знань, генерує висновок про необхідність та метод повторного тестування, який передається користувачу через кодувальник, динамічний довідник та діалоговий компонент;

6) динамічний довідник – надає користувачу довідку про формат вхідного файлу, про відомі системі методи і операції основного тестування ПЗ, типи виявлених під час основного тестування помилки ПЗ, а також представляє в наглядній формі всі повідомлення будь-якого з компонентів системи;

7) діалоговий компонент –візуалізує повідомлення динамічного довідника та видає їх користувачу в зрозумілій для сприйняття формі.

Запропонована система ідентифікації прихованих помилок програмного забезпечення дозволяє користувачу, на основі звіту про результати основного тестування, одержати висновок про необхідність повторного тестування, а саме: про наявність у програмному забезпеченні прихованих помилок та про метод, яким рекомендується здійснювати повторне тестування. В даній системі кодувальник виступає в якості інтерфейсу між користувачем і системою, а динамічний довідник – у якості інтерфейсу між користувачем та кодувальником.

4.2. Реалізація системи ідентифікації прихованих помилок програмного забезпечення

Систему ідентифікації прихованих помилок програмного забезпечення було реалізовано в Borland C++ Builder 6.0 із застосуванням системи управління базами даних Paradox7. Лістинги всіх програм даного проекту знаходяться в додатку Г.

Після активізації системи користувачем на екрані з’являється наступне вікно (рис.4.2):

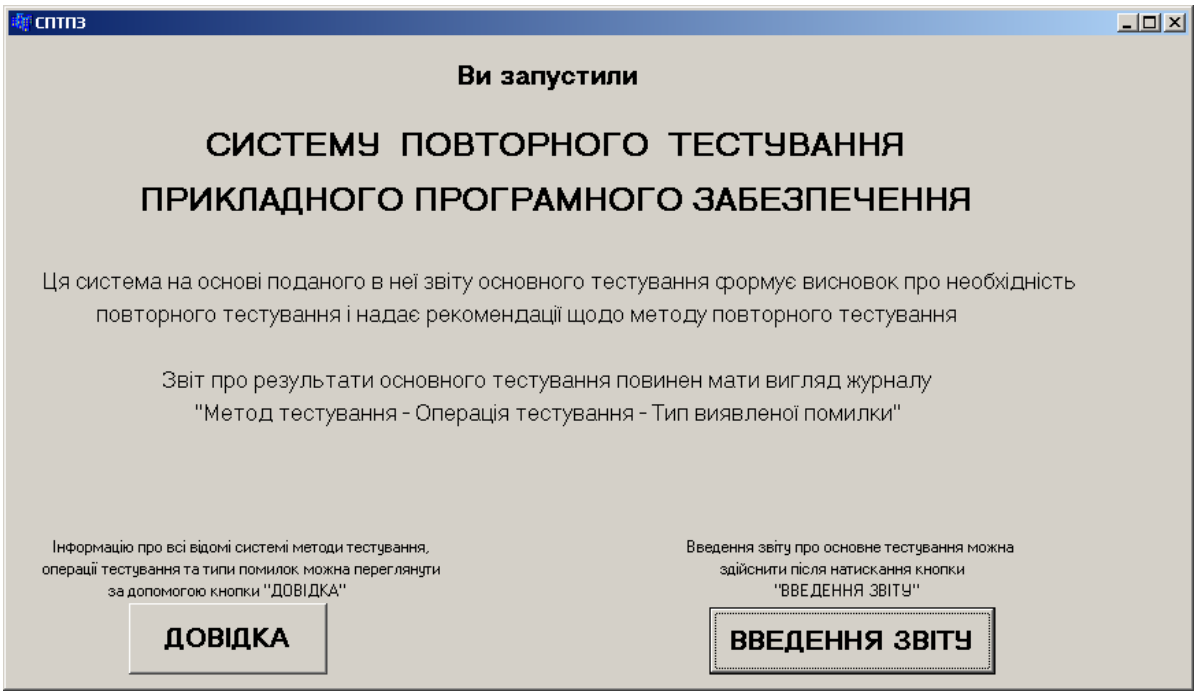

Рис.4.2. Перше діалогове вікно системи ідентифікації прихованих помилок програмного забезпечення

Після натискання кнопки “Довідка” підключається довідка про всі відомі системі методи тестування, операції тестування та типи помилок, а також таблиця залежності операцій основного тестування і типів виявлених під час основного тестування помилок від методу основного тестування та таблиця відповідності номерів методу основного тестування, операцій основного тестування і типів виявлених під час основного тестування помилок (рис.4.3), які допомагають користувачу вірно сформувати звіт про основне тестування, який надалі подаватиметься в систему для формування висновку про необхідність та метод повторного тестування.

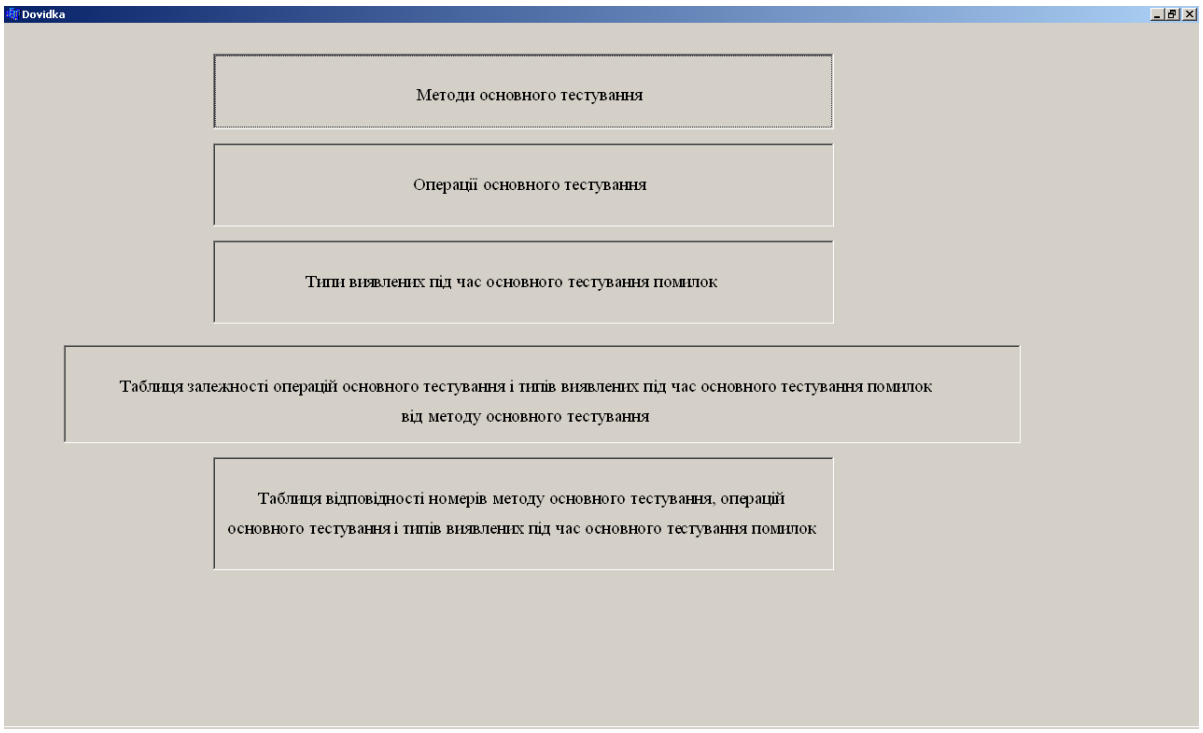

Рис.4.3. Вікно довідника

При виборі будь-якого з компонентів виникає вікно типу (рис.4.4):

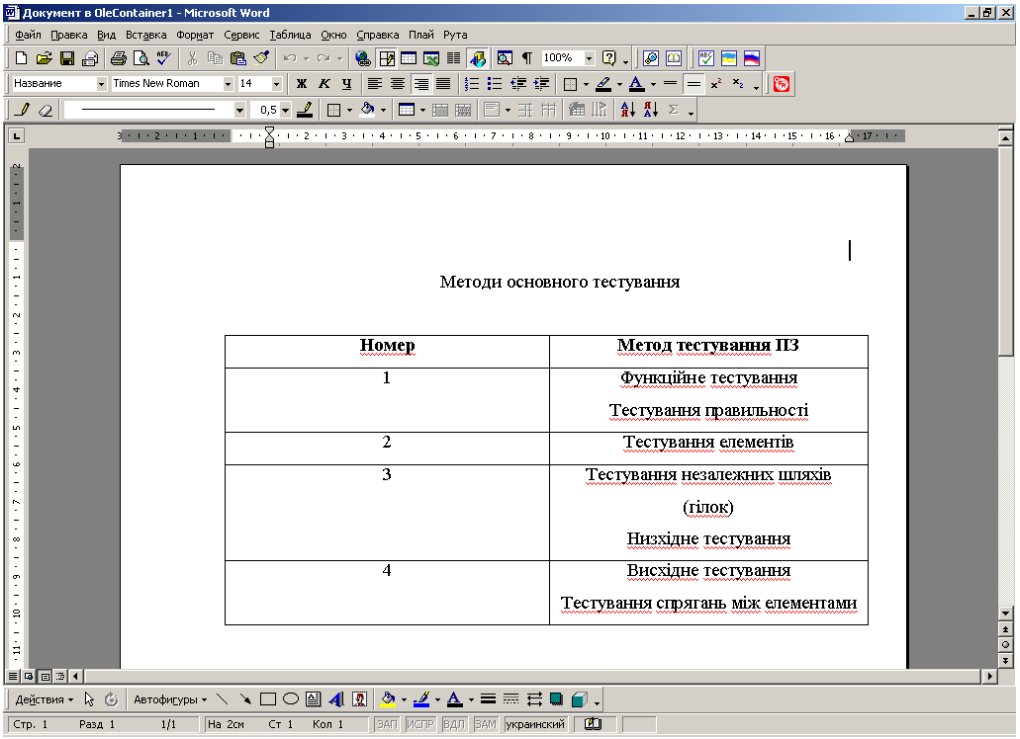

Рис. 4.4. Довідка про методи основного тестування

При натисканні в першому діалоговому вікні кнопки “ВВЕДЕННЯ ЗВІТУ” користувач має змогу подати по рядку звіт в систему повторного тестування прикладного програмного забезпечення (рис.4.5).

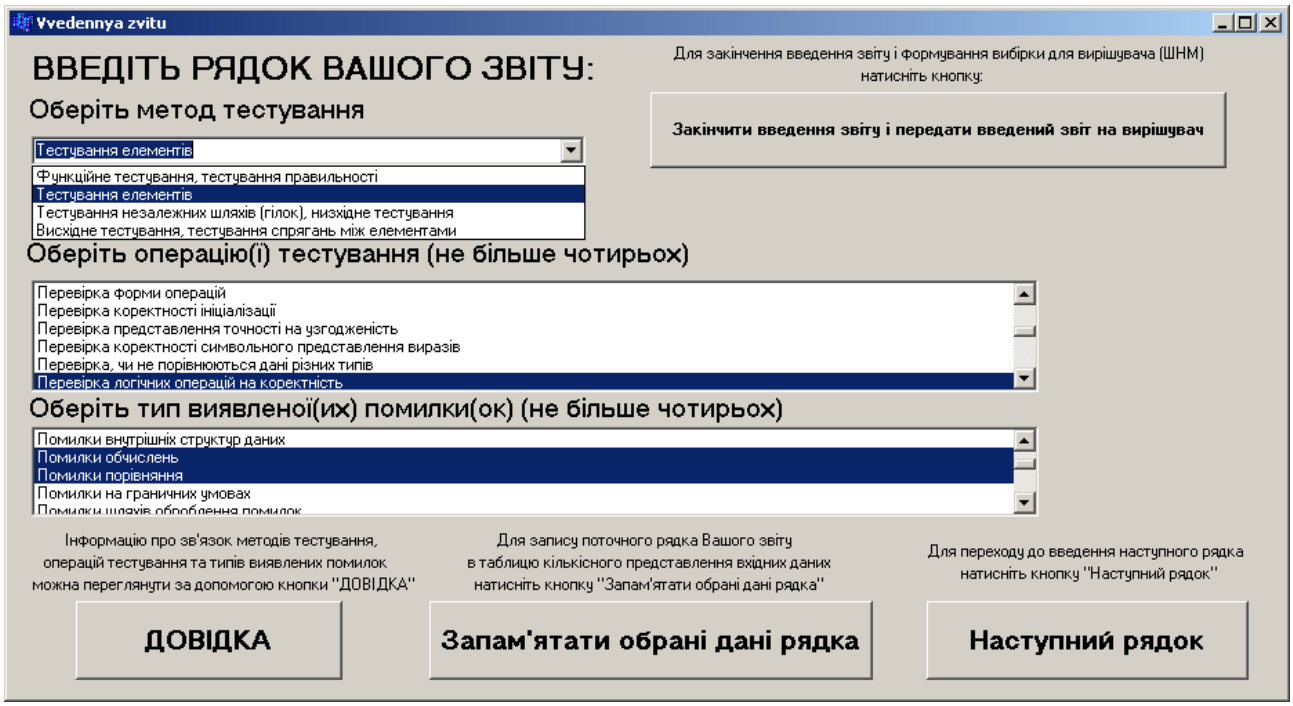

Рис.4.5. Введення рядка звіту

Для вибору методу тестування підключається поле Method таблиці бази знань Methodi.db, для вибору операції(й) тестування підключається поле Oper таблиці бази знань Operacii.db, для вибору типу(ів) виявленої(их) помилки(ок) підключається поле Type таблиці бази знань Tipi.db. Для операцій тестування та типів виявлених помилок можливий множинний вибір.

Після введення одного рядка звіту потрібно натиснути кнопку “Запам’ятати обрані дані”, яка дозволяє перетворити обрані текстові дані в кількісну форму. Після цього натискаємо кнопку “Наступний рядок” для введення наступного рядка звіту. Якщо ж введення звіту закінчено, потрібно натиснути кнопку “Закінчити введення звіту і передати введений звіт на вирішувач”.

Отже, якщо на вхід системи буде подано звіт 1 про результати основного тестування (таблиця 4.1), то після кодування цей звіт матиме вигляд, представлений на рис.4.6.

Таблиця 4.1

Звіт 1 про результати основного тестування

Метод тестування	Операція тестування	Тип виявленої помилки
(1)	(2)	(3)
Тестування елементів	Перевірка коректності кожної гілки програми (графу керування)	Помилки незалежних маршрутів програми
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка пріоритету арифметичних операцій на правильність і зрозумілість	Помилки обчислень
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка коректності роботи “заглушок”	Некоректна робота “заглушок”
Висхідне тестування, тестування спрягань між елементами	Перевірка правильності розробки та функціювання драйверів	Помилки драйверів та їх розробки
Висхідне тестування, тестування спрягань між елементами	Перевірка даних на втрати при проходженні через спрягання	Помилки спрягань модуля
Висхідне тестування, тестування спрягань між елементами	Перевірка, чи немає несприятливого впливу одного модуля на інший	Помилки спрягань модуля
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка форми операцій	Помилки обчислень

(1)	(2)	(3)
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка логічних операцій на коректінсть Перевірка пріоритету арифметичних операцій на правильність і зрозумілість	Помилки порівняння Помилки обчислень

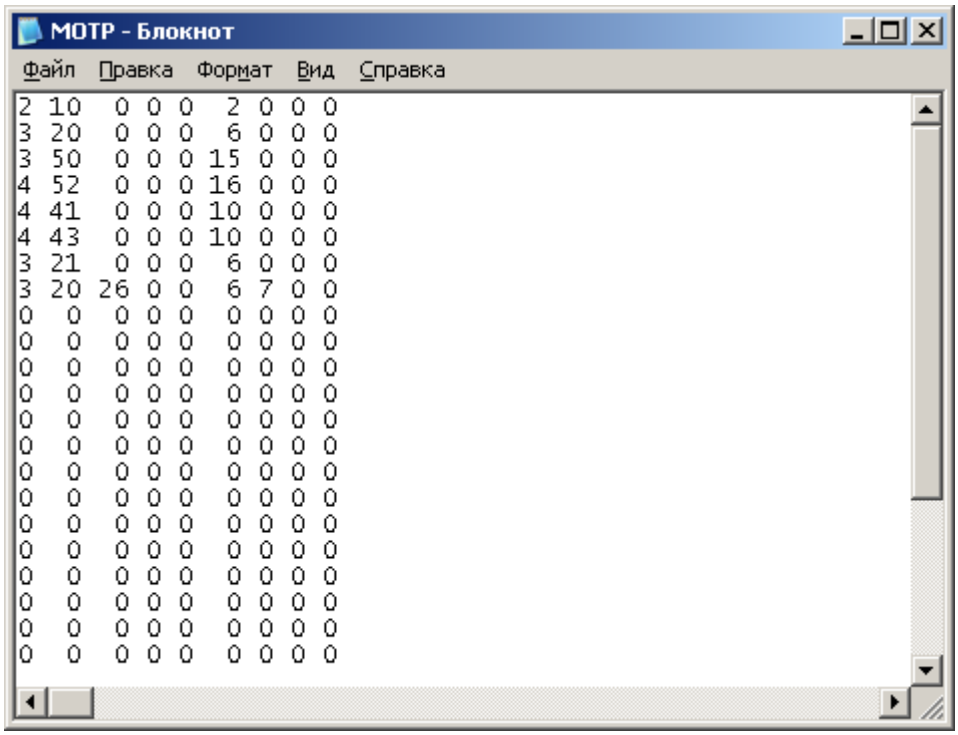

Рис.4.6. Файл МОТР.dat кількісного представлення вхідних даних звіту 1

Таблицю кількісного представлення побудовано таким чином, що в неї можна розмістити звіт, який складається з 1000 рядків. Оскільки звіт 1 складається з 8 рядків, то 9-й – 1000-й рядки таблиці дорівнюють нулям.

Після кодування вхідних даних формується вхідна вибірка для ШНМ за принципами, описаними в 3.6.

Для звіту, представленого на рис.4.6, вхідна вибірка для входу Input1 ШНМ матиме вигляд, представлений на рис.4.7.

Inp1 - Блокнот

Файл Правка Формат Вид Справка

```
 0  0  0  0   0  0  0  0  2  0  0  0  0  0  0  0
 6  0  0  0   0  0  0  0  0  0  0  0  0  0  0  0
15  0  0  0   0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  16  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  10  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  10  0  0  0  0  0  0  0  0  0  0  0
 6  0  0  0   0  0  0  0  0  0  0  0  0  0  0  0
 6  7  0  0   0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0   0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0   0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0   0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0   0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0   0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0   0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0   0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0   0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0   0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0   0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0   0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0   0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0   0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0   0  0  0  0  0  0  0  0  0  0  0  0
```

Рис.4.7. Файл Inp1.dat з вхідною вибіркою для входу Input1 ШНМ, сформованою за звітом 1

Для звіту, представленого на рис.4.6, вхідна вибірка для входу Input2 ШНМ матиме вигляд, представлений на рис.4.8.

Inp2 - Блокнот

Файл Правка Формат Вид Справка

```
 0   0  0  0
20   0  0  0
50   0  0  0
 0   0  0  0
 0   0  0  0
 0   0  0  0
21   0  0  0
20  26  0  0
 0   0  0  0
 0   0  0  0
 0   0  0  0
 0   0  0  0
 0   0  0  0
 0   0  0  0
 0   0  0  0
 0   0  0  0
 0   0  0  0
 0   0  0  0
 0   0  0  0
 0   0  0  0
 0   0  0  0
 0   0  0  0
```

Рис.4.8. Файл Inp2.dat з вхідною вибіркою для входу Input2 ШНМ, сформованою за звітом 1

Для звіту, представленого на рис.4.6, вхідна вибірка для входу Input3 ШНМ матиме вигляд, представлений на рис.4.9.

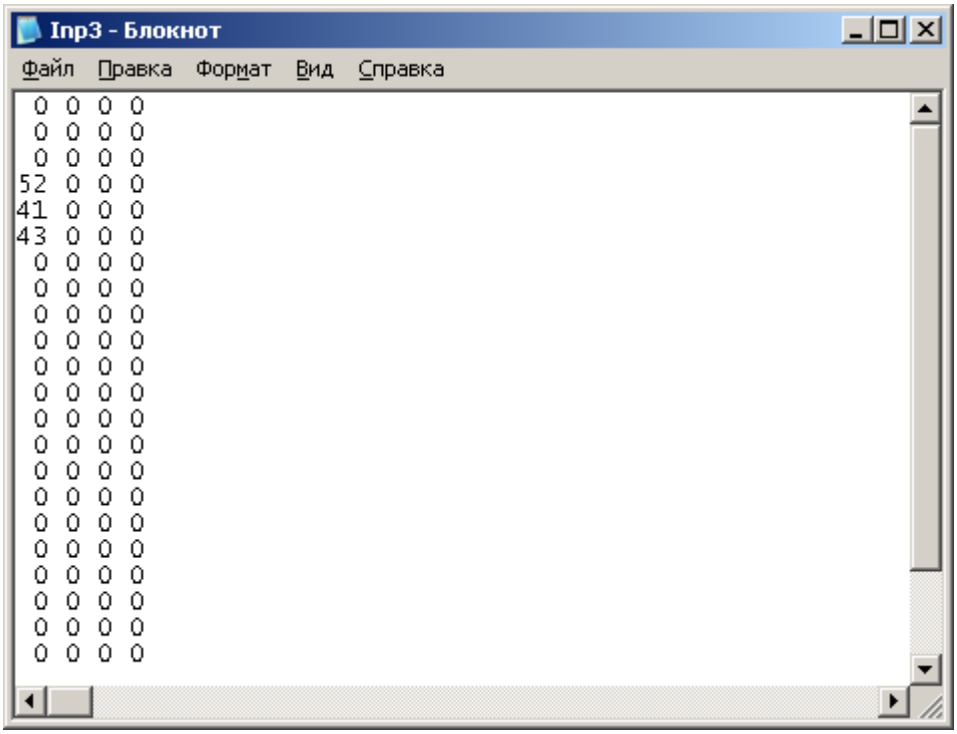

Рис.4.9. Файл Inp3.dat з вхідною вибіркою для входу Input3 ШНМ, сформованою за звітом 1

Для звіту, представленого на рис.4.6, вхідна вибірка для входу Input4 ШНМ матиме вигляд, представлений на рис.4.10.

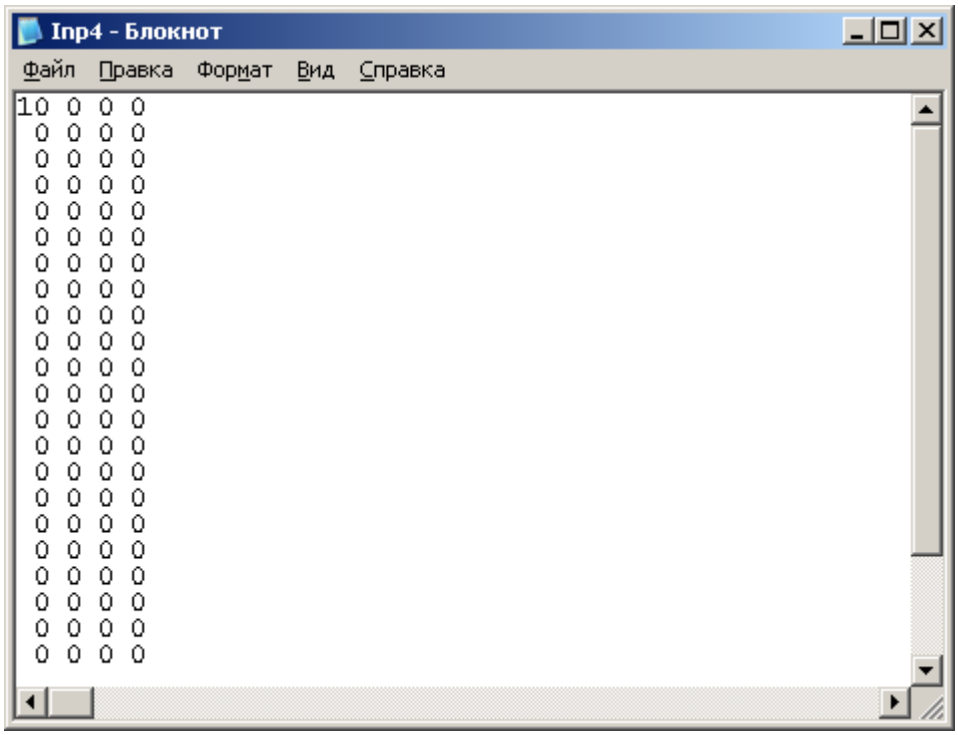

Рис.4.10. Файл Inp4.dat з вхідною вибіркою для входу Input4 ШНМ, сформованою за звітом 1

Для звіту, представленого на рис.4.6, вхідна вибірка для входу Input5 ШНМ матиме вигляд, представлений на рис.4.11.

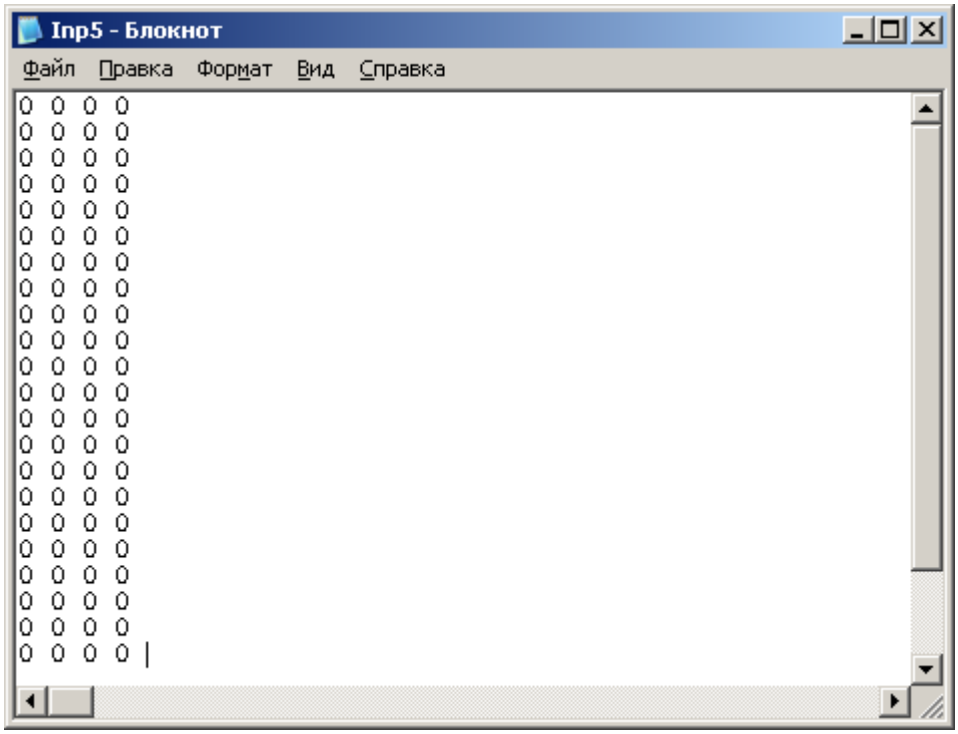

Рис.4.11. Файл Inp5.dat з вхідною вибіркою для входу Input5 ШНМ, сформованою за звітом 1

З вхідних вибірок для кожного з входів Input1 – Input5 (рис.4.7 – 4.11) формуємо загальну вхідну вибірку ШНМ (4.1) і передаємо її на входи ШНМ.

{[0; 0; 0; 0; 0; 0; 0; 0; 2; 0; 0; 0; 0; 0; 0; 0]
[6; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0]
[15; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0]
[0; 0; 0; 0; 16; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0]
[0; 0; 0; 0; 10; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0]
[0; 0; 0; 0; 10; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0]
[6; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0]
[6; 7; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0]
[0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0]
[0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0]
…….;
[0; 0; 0; 0] [20; 0; 0; 0] [50; 0; 0; 0] [0; 0; 0; 0] [0; 0; 0; 0] [0; 0; 0; 0]
[21; 0; 0; 0] [20; 26; 0; 0] [0; 0; 0; 0] [0; 0; 0; 0] …….;
[0; 0; 0; 0] [0; 0; 0; 0] [0; 0; 0; 0] [52; 0; 0; 0] [41; 0; 0; 0] [43; 0; 0; 0]
[0; 0; 0; 0] [0; 0; 0; 0] [0; 0; 0; 0] [0; 0; 0; 0] ……;
[10; 0; 0; 0] [0; 0; 0; 0] [0; 0; 0; 0] [0; 0; 0; 0] [0; 0; 0; 0] [0; 0; 0; 0]
[0; 0; 0; 0] [0; 0; 0; 0] [0; 0; 0; 0] [0; 0; 0; 0] ……;
[0; 0; 0; 0] [0; 0; 0; 0] [0; 0; 0; 0] [0; 0; 0; 0] [0; 0; 0; 0] [0; 0; 0; 0]
[0; 0; 0; 0] [0; 0; 0; 0] [0; 0; 0; 0] [0; 0; 0; 0] ……}; (4.1)

ШНМ опрацьовує вхідну вибірку і заносить результати у таблицю розміром n x 20, де n – кількість рядків у вхідному звіті. Наприклад, для звіту 1 – **n=8**. Тобто, після опрацювання загальної вибірки (4.1), на виході ШНМ одержано:

{[0;0;0;0;0] [0;0;0;0;0] [1;1;1;1;1] [0;0;0;0;0]
[1;1;1;1;1] [0;0;0;0;0] [0;0;0;0;0] [0;0;0;0;0]
[1;1;1;1;1] [0;0;0;0;0] [0;0;0;0;0] [0;0;0;0;0]

[0;0;0;0;0] [1;1;1;1;1] [0;0;0;0;0] [0;0;0;0;0]
[0;0;0;0;0] [1;1;1;1;1] [0;0;0;0;0] [0;0;0;0;0]
[0;0;0;0;0] [1;1;1;1;1] [0;0;0;0;0] [0;0;0;0;0]
[1;1;1;1;1] [0;0;0;0;0] [0;0;0;0;0] [0;0;0;0;0]
[1;1;1;1;1] [0;0;0;0;0] [0;0;0;0;0] [0;0;0;0;0]
[0;0;0;0;0] [0;0;0;0;0] [0;0;0;0;0] [0;0;0;0;0]
[0;0;0;0;0] [0;0;0;0;0] [0;0;0;0;0] [0;0;0;0;0]
… … … … … … … … … … … … … … …}

Після розшифрування цих даних є зрозумілим, що діагностовано одну помилку 3-го рівня категорійності, чотири помилки 1-го рівня категорійності та три помилки 2-го рівня категорійності.

Для формування висновку вводяться величини порогових значень кількостей помилок кожного рівня категорійності.

4.3. Опис порогових значень кількостей помилок кожного рівня категорійності

Щодо розподілу помилок ПЗ взагалі за видами і впливом на роботу комп’ютерних систем, то в літературі відомий їх розподіл за пріоритетами і категоріями [68]. Оскільки уточнення цього підходу щодо опису прихованих помилок з введенням концепції категорійності помилок проведено в дисертаційній роботі, то порогові значення кількостей помилок кожного рівня категорійності, по перевищенню яких приймається висновок про необхідність повторного тестування, в відомих літературних джерелах не описані.

Для встановлення цих порогових значень проводились дослідження кількості помилок програмного забезпечення, яке складалось з різної кількості операторів, з врахуванням і без врахування впливу помилок одного типу (рівня категорійності) на виникнення помилок наступного типу (рівня категорійності). Результати такого дослідження відображені в таблиці 4.2.

Таблиця 4.2

Кількість помилок програмного забезпечення з врахуванням і без врахування впливу помилок одного типу на виникнення помилок наступного типу

	Кількість виявлених помилок без врахування впливу помилок одного типу на виникнення помилок наступного типу					Реальна кількість помилок				
Кількість операторів в програмі	100	500	1000	5000	10000	100	500	1000	5000	10000
Загальна кількість помилок	10	18	25	42	68	16	24	36	42	85
Незначні помилки (1 РК)	8	14	19	31	51	8	14	19	31	51
Помірні помилки (2 РК)	2	4	5	11	16	5	9	14	11	33
Серйозні помилки (3 РК)	0	0	1	0	1	2	1	2	0	1
Катастрофічні помилки (4 РК)	0	0	0	0	0	1	0	1	0	0

В результаті аналізу дослідження таблиці 4.2 можна зробити наступні висновки:

1) для програмного коду зі 100 операторів кількість незначних помилок становить 80% (перевищує 75%) загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу і через це виникли 3 помірні помилки; кількість помірних помилок складає 50% загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу і через це виникли 2 серйозні помилки, які спричинили появу катастрофічної помилки;

2) для програмного коду з 500 операторів кількість незначних помилок становить 78% (перевищує 75%) загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу і через це виникло 5 помірних помилок; кількість помірних помилок складає 50% загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу і через це виникла 1 серйозна помилка, яка не спричинила появу катастрофічних помилок;

3) для програмного коду з 1000 операторів кількість незначних помилок становить 76% (перевищує 75%) загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу і через це виникло 9 помірних помилок; кількість помірних помилок складає 56% (перевищує 50%) загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу і через це виникла 1 серйозна помилка, 2 серйозні помилки спричинили появу катастрофічної помилки;

4) для програмного коду з 5000 операторів кількість незначних помилок становить 74% (не перевищує 75%) загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу, тому помірних помилок з причини накопичення незначних помилок не виникло; кількість помірних помилок складає 26% загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу, тому серйозних помилок з причини накопичення помірних помилок не виникло, а, отже, не виникли й катастрофічні помилки;

5) для програмного коду з 10000 операторів кількість незначних помилок становить 75% загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу і через це виникло 17 помірних помилок; кількість помірних помилок складає 49% (не перевищує 50%) загальної кількості виявлених без врахування взаємовпливу помилок одного типу на помилки іншого типу, тому серйозних помилок з причини накопичення помірних помилок не виникло; одна ж серйозна помилка не спричинила появу катастрофічної помилки.

Порогові значення вводяться на основі евристичних оцінок (за таблицею 4.2) наступним чином:

1. якщо кількість помилок 4-го рівня категорійності (катастрофічних) перевищує 1, то повторне тестування необхідне за причини можливості відмови програмної системи;
2. якщо кількість помилок 3-го рівня категорійності (серйозних) перевищує 2, то повторне тестування необхідне за причини виникнення помилок вищого рівня категорійності;
3. якщо кількість помилок 2-го рівня категорійності (помірні) дорівнює або перевищує 50% від загальної кількості виявлених під час основного тестування помилок, то повторне тестування необхідне за причини виникнення помилок вищих рівнів категорійності;

4. якщо кількість помилок 1-го рівня категорійності (незначні) дорівнює або перевищує 75% від загальної кількості виявлених під час основного тестування помилок, то повторне тестування необхідне за причини виникнення помилок вищих рівнів категорійності.

При аналізі таблиці 4.2 видно, що при перевищенні саме таких значень виникають приховані помилки більш високих рівнів категорійності, тому ці значення використовуються в якості порогових при формуванні висновку про необхідність повторного тестування прикладного програмного забезпечення.

4.4. Приклади формування висновку про необхідність та метод повторного тестування програмного забезпечення

Після аналізу звіту 1 із застосуванням введених порогових значень система видає висновок, що повторне тестування не потрібне (рис.4.12).

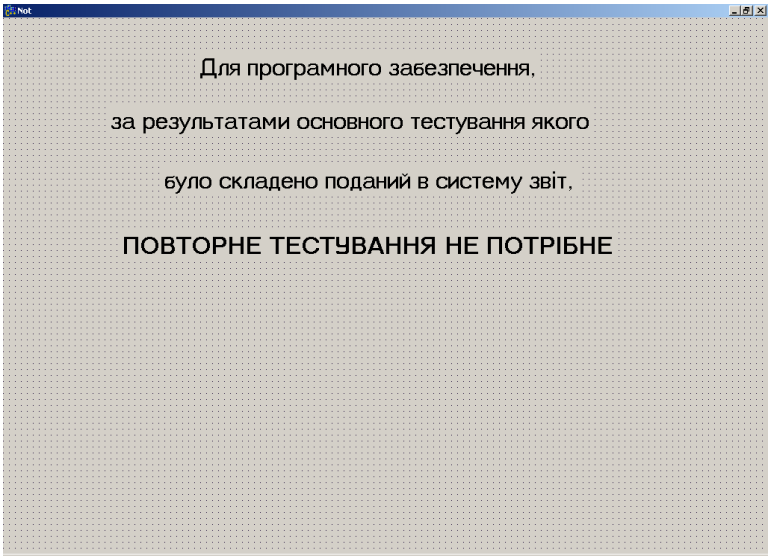

Рис.4.12. Висновок системи ідентифікації прихованих помилок після аналізу звіту 1

Подамо на вхід системи наступний звіт.

Таблиця 4.3

Звіт 2 про результати основного тестування

Метод тестування	Операція тестування	Тип виявленої помилки
(1)	(2)	(3)
Функційне тестування, тестування правильності	Перевірка співпадання вихідних результатів з наперед відомими еталонними результатами	Некоректні чи відсутні функції
Функційне тестування, тестування правильності	Перевірка, чи виконує ПС поставлені вимоги	Програма та її функціювання не відповідає специфікації вимог до ПЗ
Тестування елементів	Перевірка коректності кожної гілки програми (графу керування)	Помилки незалежних маршрутів програми
Тестування елементів	Перевірка булевих операторів на коректність, відсутність чи надлишковість	Помилки логічних умов
Тестування елементів	Перевірка цілісності збережуваних даних	Помилки внутрішніх структур даних
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка пріоритету арифметичних операцій на правильність і зрозумілість	Помилки обчислень

(1)	(2)	(3)
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка, чи не порівнюються дані різних типів	Помилки порівняння
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка коректності роботи “заглушок”	Некоректна робота “заглушок”
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка коректності роботи в ситуаціях, коли на верхніх рівнях необхідні результати з нижніх рівнів	Помилки в ситуаціях, коли на верхніх рівнях необхідні результати з нижніх рівнів, які ще не розроблені
Висхідне тестування, тестування спрягань між елементами	Перевірка правильності розробки та функціювання драйверів	Помилки драйверів та їх розробки
Висхідне тестування, тестування спрягань між елементами	Перевірка коректності об’єднання кластерів в загальну структуру	Помилки об’єднання кластерів в загальну структуру
Висхідне тестування, тестування спрягань між елементами	Перевірка, чи не відбувається перехід окремих допустимих неточностей за допустиму межу при інтеграції	Помилки спрягань модуля

Після аналізу звіту 2 система видає висновок, що повторне тестування потрібне. Проводити його система рекомендує методом функційного тестування і тестування правильності, тестування елементів, оскільки висновок про необхідність повторного тестування зроблено по перевищенню порогового значення кількістю помилок 3-го та 4-го рівня категорійності, а помилки 3-го рівня категорійності виявляє метод тестування елементів і помилки 4-го рівня категорійності виявляє метод функційного тестування і тестування правильності (ця відповідність наведена у таблиці відповідності між номером методів тестування ПЗ та рівнем категорійності прихованих помилок ПЗ бази знань (додаток А)).

Подамо на вхід системи звіт 3.

Таблиця 4.4

Звіт 3 про результати основного тестування

Метод тестування	Операція тестування	Тип виявленої помилки
(1)	(2)	(3)
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка пріоритету арифметичних операцій на правильність і зрозумілість	Помилки обчислень
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка форми операцій	Помилки обчислень
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка коректності ініціалізації	Помилки обчислень
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка, чи не порівнюються дані різних типів	Помилки порівняння
Тестування незалежних шляхів (гілок), низхідне	Перевірка зміни змінних циклу	Помилки порівняння

(1)	(2)	(3)
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка коректності роботи “заглушок”	Некоректна робота “заглушок”
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка коректності роботи в ситуаціях, коли на верхніх рівнях необхідні результати з нижніх рівнів	Помилки в ситуаціях, коли на верхніх рівнях необхідні результати з нижніх рівнів, які ще не розроблені
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка представлення точності на узгодженість	Помилки обчислень
Тестування незалежних шляхів (гілок), низхідне тестування	Перевірка коректності пріоритетності	Помилки порівняння
Тестування елементів	Перевірка виконання всіх циклів	Помилки в циклах у межах їх границь і діапазонів
Висхідне тестування, тестування спрягань між елементами	Перевірка правильності розробки та функціювання драйверів	Помилки драйверів та їх розробки
Висхідне тестування, тестування спрягань між елементами	Перевірка, чи не відбувається перехід окремих допустимих неточностей за допустиму межу при інтеграції	Помилки спрягань модуля

Після аналізу звіту 3 система видає висновок, що повторне тестування потрібне. Проводити його система рекомендує методом тестування незалежних шляхів (гілок) та низхідного тестування, оскільки висновок про необхідність повторного тестування зроблено по перевищенню порогового значення кількістю помилок 1-го рівня категорійності, а помилки 1-го рівня категорійності виявляє метод тестування незалежних шляхів (гілок) та низхідного тестування (ця відповідність наведена у таблиці відповідності між номером методів тестування ПЗ та рівнем категорійності прихованих помилок ПЗ бази знань (додаток А)).

Подамо на вхід системи звіт 4 (таблиця 1 додатку В). Після його аналізу система видає висновок, що повторне тестування потрібне. Проводити його система рекомендує методом висхідного тестування та тестування спрягань між елементами, оскільки висновок про необхідність повторного тестування зроблено по перевищенню порогового значення кількістю помилок 2-го рівня категорійності (ця відповідність наведена у таблиці відповідності між номером методів тестування ПЗ та рівнем категорійності прихованих помилок ПЗ бази знань (додаток А)).

Подамо на вхід системи звіт 5 (таблиця 2 додатку В). Після аналізу цього звіту система видає висновок, що повторне тестування потрібне. Проводити його система рекомендує методом функційного тестування і тестування правильності, тестування елементів, оскільки висновок про необхідність повторного тестування зроблено по перевищенню порогового значення кількістю помилок 3-го та 4-го рівня категорійності, а помилки 3-го рівня категорійності виявляє метод тестування елементів і помилки 4-го рівня категорійності виявляє метод функційного тестування і тестування правильності (ця відповідність наведена у таблиці відповідності між номером методів тестування ПЗ та рівнем категорійності прихованих помилок ПЗ бази знань (додаток А)).

Подамо на вхід системи звіт 6 (таблиця 3 додатку В). Після його аналізу система видає висновок, що повторне тестування потрібне. Проводити його система рекомендує методом висхідного тестування та тестування спрягань між елементами, оскільки висновок про необхідність повторного тестування зроблено по перевищенню порогового значення кількістю помилок 2-го рівня категорійності (ця відповідність наведена у таблиці відповідності між номером методів тестування ПЗ та рівнем категорійності прихованих помилок ПЗ бази знань (додаток А)).

Подамо на вхід системи звіт 7 (таблиця 4 додатку В). Після аналізу звіту 7 система видає висновок, що повторне тестування потрібне. Проводити його система рекомендує методами функційного тестування і тестування правильності, тестування незалежних шляхів (гілок) та низхідного тестування, оскільки висновок про необхідність повторного тестування зроблено по перевищенню порогового значення кількістю помилок 4-го рівня категорійності та 1-го рівня категорійності, а помилки 4-го рівня категорійності виявляє метод функційного тестування і тестування правильності і помилки 1-го рівня категорійності виявляє метод тестування незалежних шляхів (гілок) та низхідного тестування (ця відповідність наведена у таблиці відповідності між номером методів тестування ПЗ та рівнем категорійності прихованих помилок ПЗ бази знань (додаток А)).

4.5. Висновки

1. Вперше запропоновано структуру системи ідентифікації прихованих помилок програмного забезпечення, яка дозволяє користувачу на основі звіту про результати основного тестування одержати висновок про необхідність повторного тестування, а саме: про наявність у програмному забезпеченні прихованих помилок та про метод, яким рекомендується здійснювати повторне тестування.

2. На підставі евристичних оцінок визначено величини порогових значень кількостей помилок рівнів категорійності: 1-й рівень категорійності – 75% і більше від загальної кількості виявлених помилок; 2-й рівень категорійності – 50% і більше від загальної кількості виявлених помилок; 3-й рівень категорійності – 2 і більше помилок; 4-й рівень категорійності – 1 і більше помилок.

3. На основі методу та алгоритмів ідентифікації прихованих помилок розроблено програмну реалізацію системи ідентифікації прихованих помилок програмного забезпечення, котра дає рекомендації користувачу щодо доцільності та методів повторного тестування ПЗ.