Лабораторна робота №7 (семестр 5)

ТЕМА: Організація тестування розробленої програми-додатку.

МЕТА: Отримати практичні навики в тестуванні програмного забезпечення.

ЗНАТИ: Мову програмування С або Pascal.

ВМІТИ: Інсталювати RAD (наприклад Builder або Delphi). Знати основи розробки програм під ОС Windows.

ТЕОРЕТИЧНІ ВІДОМОСТІ.

Теоретичні відомості наведені у лекції № 6.

Тестуванням називається процес виконання програми з метою виявлення помилки. Ніяке тестування не може довести відсутність помилок в програмі. Тест – це сукупність вхідних та вихідних даних, отриманих до виконання програми.

Вихідними даними для етапу тестування є технічне завдання, специфікація та розроблені на попередніх етапах структурна і функціональна схеми програмного продукту. Під час тестування рекомендується дотримуватися таких основних принципів:

1. Передбачувані результати повинні бути відомі до тестування.
2. Необхідно досконально вивчати результати кожного тесту.
3. Необхідно перевіряти дії програми на невірних даних.
4. Необхідно перевіряти програму на несподівані побічні ефекти.
5. Вдалим вважається тест, який виявляє хоча б одну ще не виявлену помилку.
6. Імовірність наявності помилки в частині програми пропорційна кількості помилок, вже виявлених в цій частині.

Процес розробки ПЗ передбачає **три стадії тестування**:

– Автономне тестування компонентів ПЗ;

– Комплексне тестування розробленого ПЗ;

– Системне або оцінкове тестування.

Існує **два підходи до формування тестів**:

– Структурний; Тестування за принципом "білого ящика";

– Функціональний; Тестування за принципом «чорної ящика».

Тестування за принципом "білого ящика"

Стратегія тестування за принципом "білого ящика", або стратегія тестування, керована логікою програми (з урахуванням алгоритму), дозволяє перевірити внутрішню структуру програми. У цьому випадку тестує отримує тестові дані шляхом аналізу логіки програми.

Тестування на основі потоку управління

Стратегія «білого ящика» включає в себе наступні критерії структурного тестування:

– C0 – критерій покриття операторів;

– C1 – критерій покриття рішень;

– C2 – критерій покриття усіх шляхів у керуючому графові програми.

Тестування на основі потоку даних

Стратегія «білого ящика» включає в себе наступні критерії структурного тестування:

CP – покриття пар досяжних дуг у керуючому графові програми.

Оформлення результатів тестування

Результати тестування за принципом "білого ящика» можна оформити у вигляді таблиці:

Номер тесту	Критерій тестування	Призначення тесту	Значення вихідних даних	Очікуваний результат	Отриманий результат

Тестування за принципом «чорного ящика»

Одним із способів перевірки програм є стратегія тестування, звана стратегією "чорного ящика" або тестуванням з керуванням за даними. У цьому випадку програма розглядається як "чорний ящик" і таке тестування має на меті з'ясування обставин, в яких поведінка програми не відповідає специфікації.

Стратегія "чорного ящика" включає в себе такі методи формування тестових наборів:

– еквівалентне розбиття;

– аналіз граничних значень;

– аналіз причинно-наслідкових зв'язків;

– припущення про помилку.

Виділення класів еквівалентності

Класи еквівалентності виділяються шляхом вибору кожної вхідної умови (зазвичай це пропозиція або фраза з специфікації) та розбивкою його на дві або більше груп. Для цього використовується таблиця наступного вигляду:

Вхідна умова	Правильні класи еквівалентності	Неправильні класи еквівалентності

Правильні класи включають правильні дані, неправильні класи – неправильні дані.

Виділення класів еквівалентності є евристичним процесом, проте при цьому існує ряд правил:

1. Якщо вхідні умови описують *область* значень (наприклад «ціле може приймати значення від 1 до 20»), то виділяють один правильний клас і два неправильних X <1 і X> 20.

2. Якщо вхідна умова описує ситуацію "повинно бути" (наприклад, «першим символом ідентифікатора повинна бути буква»), то визначається один правильний клас еквівалентності (перший символ – буква) і один неправильний (перший символ – не буква).

3. Якщо є будь-яка підстава вважати, що різні елементи класу еквівалентності трактуються програмою неоднаково, то даний клас розбивається на менші класи еквівалентності.

Побудова тестів

Цей крок полягає у використанні класів еквівалентності для побудови тестів. Цей процес включає в себе:

– призначення кожному класу еквівалентності унікального номера.

– проектування нових тестів, кожен з яких покриває як можна більше число непокритих класів еквівалентності, до тих пір, поки всі правильні класи не будуть покриті (тільки не загальними) тестами.

– запис тестів, кожен з яких покриває один і тільки один з непокритих неправильних класів еквівалентності, до тих пір, поки всі неправильні класи не будуть покриті тестами.

– розробка індивідуальних тестів для неправильних класів еквівалентності обумовлено тим, що певні перевірки з помилковими входами приховують або замінюють інші перевірки з помилковими входами.

Недоліком методу еквівалентного розбиття є те, що він не досліджує комбінації вхідних умов.

Аналіз граничних значень

Граничні умови – це ситуації, що виникають на, вище або нижче меж вхідних класів еквівалентності. Аналіз граничних значень відрізняється від еквівалентного роздроблення наступним:

Вибір будь-якого елемента в класі еквівалентності в якості представницького при аналізі граничних умов здійснюється таким чином, щоб перевірити тестом кожну границю цього класу.

При розробці тестів розглядаються не тільки вхідні умови (*простір входів*), але і *простір результатів*.

Застосування методу аналізу граничних умов вимагає певної міри творчості і спеціалізації в розглядаємій проблемі. Тим не менш, існує кілька загальних правил цього методу:

1. Побудувати тести для границь області та тести з неправильними вхідними даними для ситуацій незначного виходу за межі області, якщо вхідна умова описує область значень (наприклад, для області вхідних значень від -1.0 до +1.0 необхідно написати тести для ситуацій -1.0, +1.0, -1.001 і +1.001).

2. Побудувати тести для мінімального і максимального значень умов і тести, більше і менше цих двох значень. Якщо вхідна умова задовольняє дискретного ряду значень, наприклад, якщо вхідний файл може містити від 1 до 255 записів, то перевірити 0,1,255 і 256 записів.

3. Якщо вхід або вихід програми є впорядкована множина (наприклад, послідовний файл, лінійний список, таблиця), то зосередити увагу на першому і останньому елементах цієї множини.

Припущення про помилку

Часто програміст з великим досвідом вишукує помилки "без всяких методів". При цьому він підсвідомо використовує метод "припущення про помилку". Процедура методу припущення про помилку в значній мірі заснована на інтуїції. Основна ідея методу полягає в тому, щоб перерахувати в деякому списку можливі помилки або ситуації, в яких вони можуть з'явитися, а потім на основі цього списку скласти тести. Іншими словами, потрібно перерахувати ті спеціальні випадки, які можуть бути не враховані при проектуванні.

Оформлення результатів тестування

Результати тестування за принципом «чорного ящика» можна оформити у вигляді таблиці:

Номер тесту	Призначення тесту	Значення вихідних даних	Очікуваний результат	Реакція програми	Висновок

Тестування для користувацького інтерфейсу

При розробці набору тестів для аналізу користувальницького інтерфейсу необхідно оцінити наступні характеристики інтерфейсу програми:

1. Функціональність:
   – правильне виконання базових функцій;
   – наявність пропущених функцій;
   – робота в реальному часі;
   – надмірність або недостатність вихідної інформації;
   – способи її збереження і представлення.
2. Організація екрану:
   – естетичне оформлення екрана;
   – наявність меню;
   – організація діалогових вікон;
   – можливість управляти надлишковою інформацією на екрані.
3. Організація меню:
   – складність ієрархії меню;
   – правила переходу по меню (вверх, вниз, швидкий вибір);
   – однозначна відповідність команд і пунктів меню;
   – наявність контекстного меню.
4. Довідкова система:
   – складність і повнота викладу;
   – багатослівність і емоційність;

– помилки викладу.

5. Обробка помилок користувача:

– за вхідними даними;

– виконання функцій в неправильному контексті.

Оціночне тестування

Оціночне тестування включає такі види:

– тестування зручність використання;

– тестування на граничних обсягах;

– тестування на граничних навантаженнях;

– тестування захисту та надійності;

– тестування продуктивності при різній апаратурі;

– тестування зручності установки і обслуговування, сумісності;

Ручне тестування

Ручне тестування полягає у виконанні документування процедури, де описана методика виконання тестів, що задає порядок тестів і для кожного тесту список значень параметрів, які подаються на вхід і список результатів, очікуваних на виході. Контроль відповідності результатів очікуваних (тестів) виконує людина.

Автоматизоване тестування

Автоматизоване тестування передбачає створення тестового драйвера, де описана методика виконання тестів, що задає порядок тестів і для кожного тесту список значень параметрів, який подається на вхід і список результатів, очікуваних на виході. Контроль відповідності результатів очікуваних (тестів) повинен виконуватися автоматично (порівняти вміст двох файлів) і видати повідомлення Так / Ні і який тест не пройшов у разі Ні.

Особливості тестування для об'єктно-орієнтованих програм

Основними методами тестування для об'єктно-орієнтованих програм є:

– модульне тестування;

– інтеграційне тестування;

– системне тестування.

Модульне тестування – це тестування класів (за принципом білого ящика), використовуючи написання тестових драйверів для перевірки функціональності, що входять в клас методів.

Інтеграційне тестування – тестування за принципом білого ящика правильності взаємодії класів, що входять у різні модулі. Взаємодія об'єктів являє собою просто запит одного об'єкта на виконання іншим об'єктом однієї з операцій одержувача і всіх видів обробки, необхідних для завершення цього запиту. Способи взаємодії класів:

– загальнодоступна операція має параметри об'єктного типу;

– загальнодоступна операція повертає значення об'єктного типу;

– метод одного класу створює екземпляр іншого класу як частина соєю реалізації;

– метод одного класу посилається на глобальний екземпляр іншого класу.

Системне тестування – тестування за принципом чорної ящика правильності функціонування системи в цілому.

ПРИКЛАД ПРОТОКОЛУ ЗА РЕЗУЛЬТАТАМИ ТЕСТУВАННЯ

Звіт про виявлену невідповідність при тестуванні

Програма Калькулятор **Випуск 2 Дата XX.XX.XX**

Група розробки: Інтерфейс

Тип помилки

1. **Серйозна** Невірне число відображається у правому нижньому куті

2. **Незначна** Не можу зробити ширину стовпця рівної 17

Група розробки: Обчислення

1. **Фатальна** Збій програми, якщо результат обчислення виявляється довше п'яти цифр.

2. **Фатальна** Нескінченний цикл по таблицях в яких більше 10 рядків.

Таблиця проведених тестів

Номер тесту	Призначення тесту	Значення вихідних даних	Очікуваний результат	Реакція програми	Висновок
1					
2					
3					
4					
…	…	…	…	…	…
N					

Тестування проводив: П.І.Б.

ВИСНОВКИ ЗА РЕЗУЛЬТАТАМИ ТЕСТУВАННЯ

Програма вимагає подальшої доробки для усунення виявлених при тестуванні помилок.

Можливе використання програми, але в наступних версіях потрібно усунути, виявлені при тестуванні помилки.

ЗАВДАННЯ:

Скласти набір тестів для перевірки розробленої програми в ЛР № 6, виконати тести і проаналізувати отримані результати. Ступінь складності та кількість тестів визначається, виходячи з написаних процедур і функцій.

Звіт з лабораторної роботи повинен містити наступні елементи

- Оформлена титульна сторінка.
- Завдання.
- Назва проекту, короткий його опис.
- Складені тести, результат тестування, супутній графічний матеріал.

Лабораторна робота вважається зарахованою при виконанні наступних умов:

- Написана на позитивну оцінку летуча контрольна робота.
- Наявність звіту, оформленого згідно наведених вище вимог.
- Скласти тести.
- Виконати тестування, заповнити протокол результатів тестування.
- Співбесіда з викладачем, який приймає лабораторну роботу на наявність знань з даної тематики. (Або відповідь на контрольні питання наведені нижче.

КОНТРОЛЬНІ ЗАПИТАННЯ:

1. Що таке тестування?
2. Навіщо тестування необхідно?
3. У яких випадках можна обійтися без тестування?
4. Що таке модульне тестування?
5. Що таке інтеграційне тестування?