Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ В.Ф.УТКИНА»
Рязанский станкостроительный колледж РГРТУ

Лабораторная работа 4-6

Дополнительная профессиональная программа повышения квалификации «Тестирование программного обеспечения (с учетом стандарта Ворлдскиллс по компетенции «Программные решения для бизнеса»)»

Мазуренко Валерия Витальевна

Рязань 2022

Практическая работа №4-6

Введение в тестовую документацию. Техники тест-дизайна, написание тест-кейсов

Цель работы: Научиться использовать техники тест-дизайна при написании тестовых сценариев. Продолжительность работы - 4 часа.

Содержание

Тестовые артефакты

В соответствие с процессами или методологиями разработки ПО, во время проведения тестирования создается и используется определенное количество тестовых артефактов (документы, модели и т.д.). Наиболее распространенными тестовыми артефактами являются:

- **План тестирования (Test Plan)** - это документ описывающий весь объем работ по тестированию, начиная с описания объекта, стратегии, расписания, критериев начала и окончания тестирования, до необходимого в процессе работы оборудования, специальных знаний, а также оценки рисков с вариантами их разрешения.
- **Набор тест кейсов и тестов (Test Case & Test suite)** - это последовательность действий, по которой можно проверить соответствует ли тестируемая функция установленным требованиям.
- **Дефекты / Баг Репорты (Bug Reports / Defects)** - это документы, описывающие ситуацию или последовательность действий приведшую к некорректной работе объекта тестирования, с указанием причин и ожидаемого результата.

Тестовая документация бывает двух видов: внешняя и внутренняя. И та, и друга— инструмент, облегчающий жизнь проектной команде. Не более и не менее.

Внешняя документация:

• Замечание — короткая записка, комментарий о небольшой неточности в реализации продукта.

• Баг-репорт — описание выявленного случая несоответствия производимого продукта требованиям, к нему выдвигаемым — ошибки или ее проявления. Он обязательно должен содержать следующие элементы:

• Идею тестового случая, вызвавшего ошибку.

• Описание исходного состояния системы для выполнения кейса.

• Шаги, необходимые для того, чтобы выявить ошибку или ее проявление.

• Ожидаемый результат, т. е. то, что должно было произойти в соответствии с требованиями.

• Фактический результат, т. е. то, что произошло на самом деле.

• Входные данные, которые использовались во время воспроизведения кейса.

• Прочую информацию, без которой повторить кейс не получится.

• Критичность и/или приоритет. Экранный снимок (скрин).

• Версию, сборку, ресурс и другие данные об окружении.

• Запрос на изменение (улучшение) — описание неявных/некритичных косвенных требований, которые не были учтены при планировании/реализации продукта, но несоблюдение, которых может вызвать неприятие у конечного потребителя. И пути/рекомендации по модификации продукта для соответствия им.

• Отчет о тестировании (тест репорт) — документ, предоставляющий сведения о соответствии/ несоответствии продукта требованиям. Может так же содержать описание некоторых подробностей проведенной сессии тестирования, например, затраченное время, использованные виды тестирования, перечень проверенных случаев и т. п. В идеальном варианте фраза вида «Тест пройден. Ошибка не воспроизводится/Функционал работает корректно/Соответствует требованиям» означает, что продукт или его часть полностью соответствует требованиям прямым и косвенным (в производстве ПО).

Внутренняя документация:

• Тест-план (план тестирования) формализованное и укрупненное описание одной сессии тестирования по одному или нескольким направлениям проверок. Т.е. перечень направлений проверок, которые должны быть проведены в рамках сессии тестирования (и, сообразных этим направлениям, требований). Также может содержать в себе необходимую информацию об окружении, методике, прочих условиях важных для показательности данной сессии тестирования. Под направлением проверок также может пониматься более детализированная тестовая документация (в виде ссылки на нее): чек листы, тестовые комплекты, тестовые сценарии, на которую необходимо опираться при проведении сессии тестирования. Основная цель документа — описать границы сессии

тестирования, стабилизировать показательность данной сессии.

- Тестовый сценарий — последовательность действий над продуктом, которые связаны единым ограниченным бизнес-процессом использования, и сообразных им проверок корректности поведения продукта в ходе этих действий. Может содержать информацию об исходном состоянии продукта для запуска сценария, входных данных и прочие сведения, имеющие определяющее значение для успешного и показательного проведения проверок по сценарию. Особенностью является линейность действий и проверок, т.е. зависимость последующих действий и проверок от успешности предыдущих. Цель документа — стабилизация покрытия аспектов продукта, необходимых для выполнения функциональной задачи, показательными необходимыми и достаточными проверками. Фактически при успешном прохождении всего тестового сценария мы можем сделать заключение о том, что продукт может выполнять ту или иную возложенную на него функцию.
- Тестовый комплект — некоторый набор формализованных тестовых случаев объединенных между собой по общему логическому признаку.
- Чек-лист (лист проверок) — перечень формализованных тестовых случаев в виде удобном для проведения проверок. Тестовые случаи в чек-листе не должны быть зависимыми друг от друга. Обязательно должен содержать в себе информацию о: идеях проверок, наборах входных данных, ожидаемых результатах, булевую отметку о прохождении/непрохождении тестового случая, булевую отметку о совпадении/несовпадении фактического и ожидаемого результата по каждой проверке. Может так же содержать шаги для проведения проверки, данные об особенностях окружения и прочую информацию необходимую для проведения проверок. Цель — обеспечить стабильность покрытия требований проверками необходимыми и достаточными для заключения о соответствии им продукта. Особенностью является то, что чек-листы компонуются теми тестовыми случаями, которые показательны для определенного требования.
- Тестовый случай (тест-кейс) — формализованное описание одной показательной проверки на соответствие требованиям прямым или косвенным. Обязательно должен содержать следующую информацию:
    - Идея проверки.
    - Описание проверяемого требования или проверяемой части требования.
    - Используемое для проверки тестовое окружение.
    - Исходное состояние продукта перед началом проверки.
    - Шаги для приведения продукта в состояние, подлежащее проверке.

- Входные данные для использования при воспроизведении шагов.
- Ожидаемый результат.
- Прочую информацию, необходимую для проведения проверки.

Тестовый случай (Test Case)

Тестовый случай (Test Case) - это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части.

Под тест кейсом понимается структура вида:

Action > Expected Result > Test Result Пример:

Виды Тестовых Случаев

Тест кейсы разделяются по ожидаемому результату на позитивные и негативные:

- **Позитивный** тест кейс использует только корректные данные и проверяет, что приложение правильно выполнило вызываемую функцию.
- **Негативный** тест кейс оперирует как корректными так и некорректными данными (минимум 1 некорректный параметр) и ставит целью проверку исключительных ситуаций (срабатывание валидаторов), а также проверяет, что вызываемая приложением функция не выполняется при срабатывании валидатора.

Структура Тестовых Случаев (Test Case Structure)

На просторах интернета вы сможете найти очень много информации о структуре тест кейсов, уровне их детализации и количестве проверок в них.

Каждый тест кейс должен иметь 3 части:

PreConditions

Список действий, которые приводят систему к состоянию пригодному для проведения основной проверки. Либо список условий, выполнение которых говорит о том, что система находится в пригодном для проведения основного теста

Test Case Description

Список действий, переводящих систему из одного состояния в другое, для получения результата, на основании которого можно сделать вывод о удовлетворении реализации, поставленным требованиям

PostConditions

Список действий, переводящих систему в первоначальное состояние (состояние до проведения теста - initial state)

Примечание: Post Conditions не является обязательной частью. Это скорее всего - правило хорошего тона: "намусорил - убери за собой". Это особенно актуально при автоматизированном тестировании, когда за один прогон можно наполнить базу данных сотней или даже тысячей некорректных документов.

Техники тест-дизайна

Рассмотрим программу, которая должна принять шесть символов на вход и убедиться, что первый символ является числовым и а остальные буквенно-цифровыми. Посчитаем только количество вариантов тестов с позитивным результатом:

0-9 = 10 чисел, 62 алфавитно-цифровых символа (верхнего и нижнего регистра) и 10 чисел составляют (10 * 62 ^ 5) = 9 161 328 320 вариантов. При условии, что мы затратим 10 секунд для теста, чтобы проверить все допустимые случаи нам понадобится 2905 лет непрерывного тестирования.

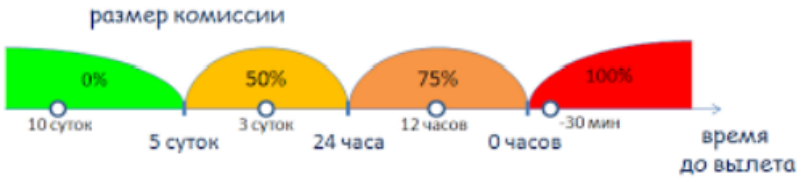
Как вывод, полное тестирование (exhaustive testing, complete testing) не возможно. Вместо исчерпывающего тестирования, мы используем анализ рисков и расстановку приоритетов и техники тест дизайна. Процесс тест-дизайна включает в себя анализ и трансформацию имеющийся документации и требований в тест-кейсы и чек-листы.

Выделяют следующие техники тест-дизайна:

- Эквивалентное разделение
- Анализ граничных значений
- Причинно-следственный анализ
- Предугадывание ошибки

- **Эквивалентное Разделение (Equivalence Partitioning — EP).** Как пример, у вас есть диапазон допустимых значений от 1 до 10, вы должны выбрать одно верное значение внутри интервала, скажем, 5, и одно неверное значение вне интервала — 0.

Давайте рассмотрим пример: функцию подсчета комиссии при отмене бронирования авиабилетов.



Предположим, что размер комиссии зависит от времени до вьшета, когда совершена отмена:

• За 5 суток до вылета комиссия составляет 0%

• Меньше 5 суток, но больше 24 часов — 50%

• Меньше 24 часов, но до вылета — 75%

• После вылета — 100% Теперь давайте пойдем по шагам:

1. **Определим классы эквивалентности** (для каждого теста из этих классов мы ожидаем получить одинаковый результат):

• 1 класс: время до вылета > 5 суток

• 2 класс: 24 часа < время до вылета < 5 суток

• 3 класс: 0 часов < время до вылета < 24 часа

• 4 класс: время до вылета < 0 часов (вылет уже состоялся)

2. **Выберем представителя от каждого класса.** Здесь мы можем поступить, как нам хочется, и выбрать любые значения из класса. Ведь, если предположить, что мы правильно разбили на классы эквивалентности, то нет разницы, какое значение из диапазона мы выберем.

• время до вылета = 10 суток (тест из 1-го класса)

• время до вылета = 3 суток (тест из 2-го класса)

• время до вылета = 12 часов (тест из 3-го класса)

• время до вылета = -30 мин (тест из 4-го класса)

3. **Выполним тесты:**

• Отменим бронь за 10 суток до вылета и проверим, что комиссия составила 0%.

• Отменим бронь за 3 суток до вылета и проверим, что комиссия составила 50%.

• Отменим бронь за 12 часов до вылета и проверим, что комиссия составила 75%.

• Отменим бронь через 30 мин после вылета и проверим, что комиссия составила 100%.

Мы видим, что у нас осталось всего 4 теста. А сколько возможных тестов

существует?

Даже если мы введем ограничение, что отмена бронирования может произойти в рамках 10 суток до вылета и 1 суток после вылета, то у нас будет около 950400 возможных тестов (я посчитал количество секунд в 11 сутках).

Мы рассмотрели очень простой пример. Редко бывает так, что функция зависит только от одной переменной. Обычно классов эквивалентности больше и выделить их сложнее.

Плюсы и минусы техники анализа классов эквивалентности

Как и любая другая техника, анализа классов эквивалентности имеет достоинства и недостатки.

- К плюсам можно отнести заметное сокращение времени и улучшение структурированности тестирования.
- К минусам можно отнести то, что при неправильном использовании техники мы рискуем потерять баги.

Еще раз напомню важный принцип разбиения на классы эквивалентности:

- Слишком большое количество эквивалентных классов увеличивает вероятность, что большинство тестов будет лишним (избыточным)
- Слишком малое число эквивалентных классов, хоть и уменьшает время тестирования, но увеличивает вероятность, что ошибки продукта будут пропущены.

- **Анализ Граничных Значений (Boundary Value Analysis — BVA).** Если взять пример выше, в качестве значений для позитивного тестирования выберем минимальную и максимальную границы (1 и 10), и значения больше и меньше границ (0 и 11). Анализ Граничный значений может быть применен к полям, записям, файлам, или к любого рода сущностям имеющим ограничения.

Примерный алгоритм использования техники анализа граничных значений:

1. Во-первых, нужно **выделить классы эквивалентности**. Опять же, это очень важный шаг, и от правильности разбиения на классы эквивалентности зависит эффективность тестов граничных значений.

2. Далее нужно **определить граничные значения** этих классов.
3. Нам нужно **понять, к какому классу будет относиться каждая граница**.
4. Для каждой границы нам нужно **провести тесты по проверке значения до границы, на границе, и сразу после границы.**

Можно сказать, что количество тестов для проверки граничных значений будет равно количеству границ, умноженному на 3. Причем в литературе по тестированию рекомендуется проверять значения вплотную к границе. Скажем, если мы имеем диапазон целых значений, и граница у нас находится в числе 10, то мы будем проводить тесты с числом 9 (вплотную бо границы), 10 (саму границу) и 11 (сразу после границы).

Вернемся к нашему примеру с бронированием и проведем тестирование граничных значений.

Пойдем по шагам:

1. **Выделим классы эквивалентности:**
   - время до вылета > 5 суток
   - 24 часа =+ время до вылета =< 5 суток
   - 0 часов < время до вылета < 24 часа
   - время до вылета =< 0 часов (вылет уже состоялся)

2. **Определим границы:**

   1. 5 суток
   2. 24 часа
   3. 0 часов

3. **Определим, к какому классу относятся границы:**

(Примечание: На этом шаге спорный момент, на который нужно обратить внимание. При составлении задачи не известно, какая логика должна быть на самих границах. Это типичный пример того, как составители требований не задумываются о границах. И поэтому программист может трактовать их по- своему.)

1. 5 суток — к 2-му классу
2. 24 часа — к 32-му классу [спасибо Анастасии за исправление, см. комментарий ниже]
3. 0 часов — к 4-му классу

4. Протестируем значения на границах, до и после них:

1. Отменим бронь за 5 суток + 1 секунду до вылета (или просто постараемся выполнить бронь как можно ближе к границе, но слева от нее) и проверим, что комиссия равна 0%.
2. Отменим бронь ровно за 5 суток до вылета и проверим, что комиссия равна 50%.
3. Отменим бронь за 5 суток — 1 секунду до вылета и проверим, что комиссия равна 50%.
4. Отменим бронь за 24 часа + 1 секунду до вылета и проверим, что комиссия равна 50%.
5. Отменим бронь ровно за 24 часа до вылета и проверим, что комиссия равна-50%.
6. Отменим бронь за 24 часа - 1 секунду до вылета и проверим, что комиссия равна 75%.
7. Отменим бронь за 1 секунду до вылета и проверим, что комиссия равна 75%.
8. Отменим бронь ровно во время вылета и проверим, что комиссия равна 100%.
9. Отменим бронь спустя 1 секунду после вылета и проверим, что комиссия равна 100%.

Мы получили 9 тестов, по 3 теста на каждую границу. Если суммировать тесты, необходимые для проверки классов эквивалентности и граничных значений, получим 4 + 9 =13 тестов.

В некоторых источниках рекомендуется использовать классы эквивалентности и граничные условия вместе по следующим соображениям:

- Техника анализа классов эквивалентности говорит о том, что мы должны выбрать минимум по одному значению из каждого класса.
- Так как граница обычно относится к какому-то классу, то можно использовать ее как представителя этого класса.
- Тогда мы сэкономим определенное количество тестов.

Если следовать этой рекомендации, то в нашем случае останется 9 тестов. Но эту

рекомендацию нужно использовать на свой страх и риск. Если считаете, что необходимо все-таки проверить обычные (не граничные) тесты — сделайте это. Лучше перестраховаться, чем пропустить ошибки. Опять же, мы рассмотрели очень простой пример, в реальных продуктах обычно граничные значения найти сложнее.

Плюсы и минусы техники

Можно выделить следующее преимущество техники анализа граничных значений:

Эта техника добавляет в технику анализа классов эквивалентности ориентированность на конкретный тип ошибок. То есть, техника анализа классов эквивалентности просто говорит нам о том, что нужно разбить все тесты на классы и провести тестирование всех классов. А техника граничных значений ориентирована на обнаружение конкретной проблемы — возникновения ошибок на границах классов эквивалентности.

Но, как и для техники анализа классов эквивалентности, эффективность техники анализа граничных значений зависит от правильности ее использования. Мы должны приложить усилия, чтобы правильно определить классы эквивалентности и их границы. Если мы отнесемся к этому поверхностно

— мы рискуем пропустить ошибки.

- **Причина / Следствие (Cause/Effect — CE).** Это, как правило, ввод комбинаций условий (причин), для получения ответа от системы (Следствие). Например, вы проверяете возможность добавлять клиента, используя определенную экранную форму. Для этого вам необходимо будет ввести несколько полей, таких как «Имя», «Адрес», «Номер Телефона» а затем, нажать кнопку «Добавить» — эта «Причина». После нажатия кнопки «Добавить», система добавляет клиента в базу данных и показывает его номер на экране — это «Следствие».

- **Предугадывание ошибки (Error Guessing — EG).** Это когда тест аналитик использует свои знания системы и способность к интерпретации спецификации на предмет того, чтобы «предугадать» при каких входных условиях система может выдать ошибку. Например, спецификация говорит: «пользователь должен ввести код». Тест аналитик, будет думать: «Что, если я не введу код?», «Что, если я введу неправильный код? », и так далее. Это и есть предугадывание ошибки.

• **Исчерпывающее тестирование (Exhaustive Testing — ET)** — это крайний случай. В пределах этой техники вы должны проверить все возможные комбинации входных значений, и в принципе, это должно найти все проблемы. На практике применение этого метода не представляется возможным, из- за огромного количества входных значений.

Порядок выполнения лабораторной работы.

Задание на лабораторную работу:

Разработать набор тестовых сценариев для тестирования программы определения типа треугольника с использованием техник тест-дизайна и написать баг-репорты о выявленных ошибках.

Спецификация Программа "треугольник"

==========

Программа принимает на вход три целых или вещественных числа в виде аргументов командной строки и в стандартном потоке вывода сообщает пользователю о том, существует ли треугольник с указанными длинами сторон или нет. Если существующий треугольник равнобедренный, равносторонний или прямоугольный, то программа также сообщает об этом.

Описание входных данных

==========

Три целых или вещественных числовых параметра, например, triangle 1 2 3. Вещественные значения длин сторон обрабатываются с точностью 5* l0^-6.

Описание выходных данных

==========

Программа выводит следующие сообщения: Если число аргументов отлично от трёх:

- "Необходимо указать ровно три входных параметра"

Если хотя бы один из аргументов не задаёт длину стороны:

- "Значение длин сторон треугольника должно принадлежать множеству положительных вещественных или целых чисел"

Если треугольник с заданными длинами сторон невозможен:

- "Треугольник не существует"

Если треугольник существует:

- "Треугольник существует".
- "Треугольник существует"."Равносторонний треугольник"
- "Треугольник существует". "Равнобедренный треугольник"
- "Треугольник существует"."Прямоугольный треугольник"

Контрольные вопросы:

1. В чем цель тестирования?
2. Из каких этапов состоит процесс тестирования?
3. Что такое баг-репорт?
4. Что такое тест-кейсы и для чего они нужны?
5. Почему необходимо использование техник тест-дизайна?
6. Какие техники тест-дизайна вы знаете?
7. Приведите плюсы и минусы использования эквивалентного разделения.
8. Приведите примеры использования техники анализа граничных значений.

1. http://www.protestinд.ru/
2. https://habrahabr.ru/post/254209/
3. http://qa-he1per.com/test-design-techniques/
4. http://33testers.b1ogspot.ru/2013/07/bloд-Post 27.htm1
5. https://bloц.skil1factory.ru/ц1ossary/baд-report/
6. https://habr.com/ru/post/156099/

ЛАБОРАТОРНАЯ РАБОТА.

Программа принимает на вход три целых или вещественных числа в виде аргументов командной строки и в стандартном потоке вывода сообщает пользователю о том, существует ли треугольник с указанными длинами сторон или нет. Если существующий треугольник равнобедренный, равносторонний или прямоугольный, то программа также сообщает об этом.

Рассмотрим программу, которая должна принять 3 символа на вход и убедиться, что первый символ является числовым и а остальные буквенно-цифровыми. Посчитаем только количество вариантов тестов с позитивным результатом:

- **Эквивалентное Разделение (Equivalence Partitioning — EP).** Как пример, у вас есть диапазон допустимых значений от 1 до 10, вы должны выбрать одно верное значение внутри интервала, скажем, 5, и одно неверное значение вне интервала —

Рассмотрим типы треугольников в условии:

• Равнобедренный (две стороны равны)

• Равносторонний (все стороны равны)

• Прямоугольный (один угол равен 90)

1. **Определим классы эквивалентности** (для каждого теста из этих классов мы ожидаем получить одинаковый результат):

•1 класс Равнобедренный (две стороны равны)

2 класс Равносторонний (все стороны равны)

3 класс Прямоугольный (один угол равен 90)

4 класс Треугольник (любой другой не подходящий под наше условие)

2. **Выберем представителя от каждого класса.** Здесь мы можем поступить, как нам хочется, и выбрать любые значения из класса. Ведь, если предположить, что мы правильно разбили на классы эквивалентности, то нет разницы, какое значение из диапазона мы выберем.

a=2, b=2, c=4 (тест из 1-го класса)

a=2, b=2, c=2 (тест из 2-го класса)

a=2, b=3, c=4 (тест из 3-го класса)

a=3, b=4, c=5 (тест из 4-го класса)

3. **Выполним тесты:**

1. a=2, b=2, c=4, получим «равнобедренный треугольник»

2. a=2, b=2, c=2 получим «равносторонний треугольник»

3. a=2, b=3, c=4 получим «прямоугольный треугольник»

4. a=3, b=4, c=5 получим «треугольник существует»

Получилось 4 позитивных теста

Важный принцип разбиения на классы эквивалентности:

• *Слишком большое количество эквивалентных классов увеличивает вероятность, что большинство тестов будет лишним (избыточным)*

• *Слишком малое число эквивалентных классов, хоть и уменьшает время тестирования, но увеличивает вероятность, что ошибки продукта будут пропущены.*

• **Анализ Граничных Значений (Boundary Value Analysis — BVA).** Если взять пример выше, в качестве значений для позитивного тестирования выберем минимальную и максимальную границы (1 и 10), и значения больше и меньше границ (0 и 11). Анализ Граничный значений может быть применен к полям, записям, файлам, или к любого рода сущностям имеющим ограничения.

Алгоритм использования техники анализа граничных значений:

1. Выделим классы эквивалентности:

Все переменные должны быть <=0

2. Определим границы:

Минимально количество чисел от 1

Максимальное количество чисел 2 147 483 647.

3. Определим, к какому классу относятся границы:

1. 0 часов — 2 147 483 6473. к 1 классу

4. Протестируем значения на границах, до и после них:

1. Введем значение 0 (введено некорректное значение)
2. Введем значение 1 (введено корректное значение)
5. Введем значение 2 147 483 647 (введено корректное значение
6. Введем значение 2 147 483 648 (введено некорректное значение)
7. Введем значение Ф (введено некорректное значение)
8. Введем значение -1 (введено некорректное значение)

Мы получили 8 тестов, по 4 теста на каждую границу. Если суммировать тесты, необходимые для проверки классов эквивалентности и граничных значений, получим 4 + 8 =12 тестов.

- **Причина / Следствие (Cause/Effect — CE).**

1. Причина a=2, b=2, c=4, получим следствие равнобедренный треугольник

2.Причина a=2, b=2, c=2 получим следствие равносторонний треугольник

3.Причина a=2, b=3, c=4получим следствие прямоугольный треугольник

4. Причина a=3, b=4, c=5 получим следствие «треугольник существует

- **Предугадывание ошибки (Error Guessing — EG).** Это когда тест аналитик использует свои знания системы и способность к интерпретации спецификации на предмет того, чтобы «предугадать» при каких входных условиях система может выдать ошибку.

1. a=4, b=4(пробел)44, c=5 - Вводим в одну из переменных после чего ставим пробел

- Ошибка. Необходимо указать ровно три входных параметра

2. a=2, b=3, c=(Пробел) - Вводим в одну из переменных пробел

-Ошибка. Значение длин сторон треугольника должно принадлежать множеству положительных вещественных или целых чисел

3. a=4, b=g, c=4 - Вводим в одну из переменных букву

– Ошибка. Значение длин сторон треугольника должно принадлежать множеству положительных вещественных или целых чисел

4. a=*, b=7, c=4 - Вводим в одну из переменных символ

- Ошибка. Значение длин сторон треугольника должно принадлежать множеству положительных вещественных или целых чисел

5. Не вводить никаких значений в поля

- Ошибка. Необходимо указать ровно три входных параметра

6. a=0, b=0, c=0 – Вводим во все переменные 0

-Ошибка. Необходимо указать ровно три входных параметра

7. a=4, b=-2, c=4 - Вводим в одну из переменных отрицательное число

-Ошибка. Значение длин сторон треугольника должно принадлежать множеству положительных вещественных или целых чисел

8. a=6, b=7, c=5 - Вводим числа проверяем существует ли такой треугольник

- Треугольник существует

6. a=6, b=6, c=5 – Проверяем правило, существует Равнобедренный треугольник

Треугольник существует Равнобедренный треугольник

7. a=2, b=2, c=2 - Проверяем правило, существует Равносторонний треугольник

- Треугольник существует Равносторонний треугольник

8. a=2, b=2, c=9 - Проверяем правило, существует Прямоугольный треугольник

- Треугольник существует Прямоугольный треугольник

5. Баг-репорты о выявленных ошибках

	1	2
Название баг-репорта	Ввод данных в переменную при значении 0	Ввод данных в переменную букв, пробелов, символов, отрицательных значений
Краткое описание ошибки	Любя из сторон не может ровняться нулю	При вводе данных в одну из переменных, программа не выдает просчет
Проект	Расчёт треугольника	Расчёт треугольника
Номер версии	V.14.584	V.15.547
Серьезность бага	S3	S3
Приоритет бага	P2	P1
Статус бага	Отсрочен	Отклонен
Назначен на	Климов А.П.	Пяточкин М.А.
Окружение	Google Chrome	Google Chrome
Шаги воспроизведения	Ввести данные a=5, b=3, c=0	Ввести данные 1.a=5, b=3, c=w 2.a=5, b=3, c=@ 3.a=5, b=3, c=
Фактический результат	Существующий треугольник	Существующий треугольник
Ожидаемый результат	Ошибка. Необходимо указать ровно три входных параметра	Ошибки. Значение длин сторон треугольника должно принадлежать множеству положительных вещественных или целых чисел
Дополнения	Скриншот	Скриншот