Государственное бюджетное профессиональное образовательное учреждение

«Нижегородский Губернский колледж»

**Лабораторная работа №6**

**на тему: Разработка и тестирование программного продукта.**

|  |  |
| --- | --- |
| Руководитель Васильева П.А. | 09.04.2022 |
| Студент Герасимов Н.А. | 09.04.2022 |
| Специальность, группа: 09.02.07, 31П |  |

Нижний Новгород

2022

**Цель работы:** разработать и протестировать программный продукт в соответствии с указанными заданием.

**Задание:**

1. Разработать программу с графическим интерфейсом, которая принимает целые числа - длины сторон треугольника, после чего определяется какого он типа: равнобедренный, равносторонний или разносторонний.
2. Разработать различные механизмы защиты программы. (ноль, отрицательное число, символ или знак, теорема сторон треугольника).
3. Написать чек-лист требований качества продукта. (какие пункты должны быть выполнены, чтобы считать продукт качественным)
4. Разработать тестовый план и тестовую стратегию. (Основные элементы тестирования (по приоритету), соответствие элементов тестирования видам тестирования и методам отслеживания тестирования (документы), риски тестирования)
5. Протестировать программный продукт пятью видами тестирований (GUI & Ux-testing, Positive, Negative, White-box, Black-box). Для каждого тестирования разработать отслеживающую документацию (тест-кейсы, чек-листы, баг - репорты).
6. Разработать презентацию для защиты. Защитить работу.

**1 ЧЕК-ЛИСТ ТРЕБОВАНИЙ КАЧЕСТВА ПРОДУКТА**

Качество ПП — это совокупность его черт и характеристик, которые влияют на способность ПП удовлетворять заданные потребности пользователя.

Исходя из вышеописанного определения для того, чтобы отнести данный программный продукт к качественному, необходимо составить чек-лист с требованиями качества продукта.

Первый чек-лист требований качества продукта графического интерфейса:

* интерфейс выдержан в постельных тонах, тем самым, создавая атмосферу уюта;
* интуитивно понятный интерфейс;
* отсутствие грамматических и орфографических ошибок;
* интерфейс удобен для пользователя;
* интерфейс выдержан в едином стиле;
* отсутствии воздуха в интерфейсе.

Второй чек-лист требований качества продукта функциональной части программы:

* функциональная часть программы правильно выполняет свою предназначенную роль;
* программный продукт доступна для пользователя любой возрастной категории;
* программный продукт содержит в себе механизм «poka-yoke» (защита от дурака);
* программный продукт содержит в себе механизм, позволяющий проверить корректность выполнения результатов вычисления типа треугольника.

**2 РАЗРАБОТКА ТЕСТОВОГО ПЛАНА И ТЕСТОВОЙ СТРАТЕГИИ**

**2.1 Разработка тестового плана**

Тест-план – это документ, который чаще всего описывает весь объем работ по тестированию проекта либо части проекта.

В данном случае, документом будет являться данный отчет, который содержит в себе подпункт с разработкой тестового плана.

Тестовый план содержит в себе описание тестирования программного продукта.

Данный программный продукт представляет из себя программу, которая позволяет рассчитывать тип треугольника из трех существующих (равносторонний, разносторонний и равнобедренный).

Для составления тестового плана будет использоваться следующий функциональный перечень программы:

1. Функциональные свойства.

* Корректность вывода результата;
* Скорость отклика программы;
* Наличие защиты от «дурака»;

1. Эргономичность.

* Удобство использования программы;
* Интуитивность работы с программой;
* Наличие подсказок для каждой кнопки;
* Наличие инструкции для пользователя;
* Наличие кнопок для быстрого взаимодействия с программой (кнопка генерации случайных чисел и очистка полей).

1. Долговечность.

* Срок службы программы;

1. Стрессовое тестирование.

* Ввод символов, букв (как русских, так и латинских) в поля для ввода значений, ввод отрицательных значений, ввод нуля как первое число;

**2.2 Риски тестирования. Оценка рисков**

Риск – сочетание вероятности наступления события и последствий, вызванных определенным событием.

Перечень возможных рисков при взаимодействии с программой, которые могут возникнуть в процессе тестирования представлены в таблице 1.

Таблица 1 – Описание возможных рисков

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Описание риска** | **Вероятность возникновения** | **Оценка угрозы** |
| 1 | В процессе тестирования, программа выдала критическую ошибку, в следствии чего требуется перезапуск. | Низкая | Средняя. После выхода с программы, пользователю придется заново производить запуск, что занимает время, хоть и не долгое. Так же, если остались стороны в полях и пользователях захочет проверить достоверность выведенного типа треугольника, а программа выдала ошибку, что повлекло за собой выход, введенные ранее данные не сохранятся. |
| 2 | В процессе тестирования, кнопка для очистки полей не сработала. | Низкая | Средняя. Если при какой-то ситуации, поля на очистились, пользователю придется вручную переходить на каждое поле удалять значения или заменять на новые. |
| 3 | В процессе тестирования, кнопка со генерацией случайных чисел, повторяет свои значения. | Средняя | Высокая. Проверять типы треугольников с повторяющимися числами – бессмысленно и продолжать тестирование данной кнопки бесполезно, ведь числа будут идти по кругу.  Для предотвращения данного риска, следует проверить N раз корректность генерации чисел. |
| 5 | Не сработали подсказки на кнопках/текстовых полях | Средняя | Подсказки для кнопок функционируют в полном объеме, но для того, чтобы увидеть подсказку на текстовом поле, необходимо либо переключаться между ними при помощи клавиши TAB или ввести число, перейти на другое поле и зайти обратно на то поле, где было введено число или ввести число на одном поле, подождать N времени, после чего подсказка появится.  Для предотвращения такого риска была создана кнопка «Инструкция по использованию». |

**2.3 Стратегия тестирования**

Стратегия тестирования — это схема, описывающая подход к тестированию в рамках цикла разработки программного обеспечения.

Стратегия тестирования содержит следующие методы тестирования:

1. Функциональное тестирование.
2. Тестирование отдельных компонентов.
3. Тестирование производительности:

* Нагрузочное тестирование
* Стресс-тестирование
* Тестирование стабильности и надежности

1. Негативное тестирование
2. Тестирование совместимости
3. Тестирование безопасности
4. Юзабилити тестирование

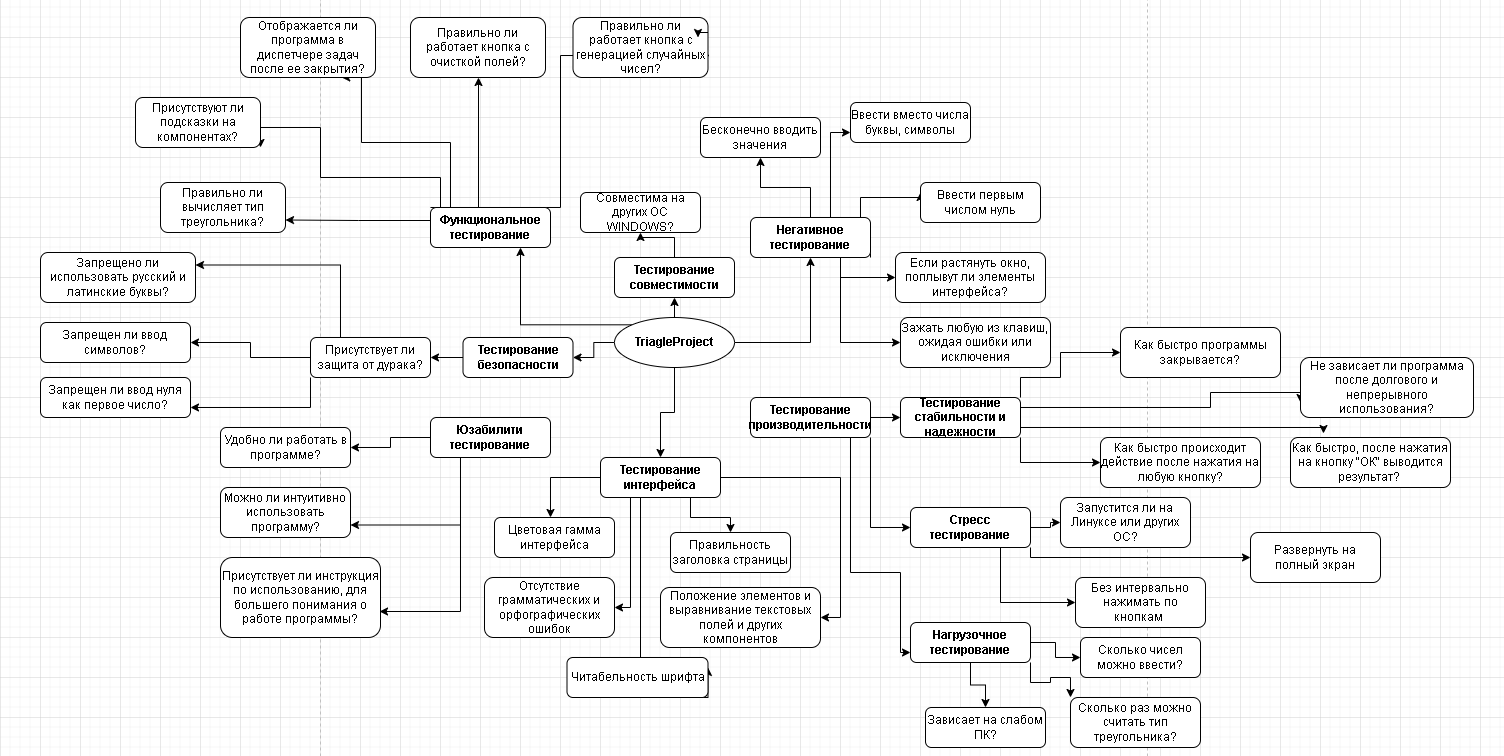
Сконструированная стратегия тестирования представлена на рисунке 1.

Рисунок 1 – Стратегия тестирование программного продукта TriagleProject

**3 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА**

**3.1 Тестирование GUI & UX**

Тестирование пользовательского интерфейса (GUI) – тестирование, выполняемое путем взаимодействия с системой через графический интерфейс пользователя.

Перед началом проведения тестирования, необходимо определить целевую аудиторию пользователей программы.

Возрастные ограничения на использование программы отсутствуют.

Проведенное тестирование пользовательского интерфейса, оно же GUI-тестирование, представлено ниже.

Преимущества:

* интерфейс выполнен в постельных тонах;
* графические элементы выдержаны в едином стиле;
* заголовки имеют корректное расположение;
* орфографические и пунктуационные ошибки отсутствуют;
* шрифт читабелен;
* отсутствие воздуха в интерфейсе.

Недостатки не выявлены.

UX тестирование (также юзабилити-тестирование, Usability testing) – комплекс мер, направленных на выявление любых проблемных мест на вашем ресурсе: достаточно ли он понятен, логичен, удобен, правильно ли работают все его технические элементы.

Проведенное Usability-тестирование представлено ниже.

Преимущества:

* присутствует инструкция по использованию программы;
* присутствует анимация активной кнопки при наведении. пример указан на рисунке 2;

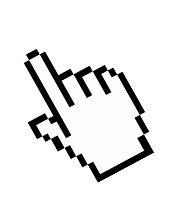


Рисунок 2 – Эффект при наведении на кнопку

* присутствуют подсказки для каждого графического компонента;
* присутствует кнопка с генерацией случайного числа;
* присутствует кнопка с очисткой всех трех полей;
* подробная инструкция по использованию приложения.

Недостатки:

* зафиксированный размер приложения: невозможно применить полноэкранный режим, увеличить или уменьшить окно (это конечно можно отнести и к положительной черте данного недостатка, ведь надобности в использовании полного экранного режима попросту нет).

Для двух видов тестирования были разработаны опросники.

Опросник по интерфейсу программы и результаты ответов представлен на рисунке 3.

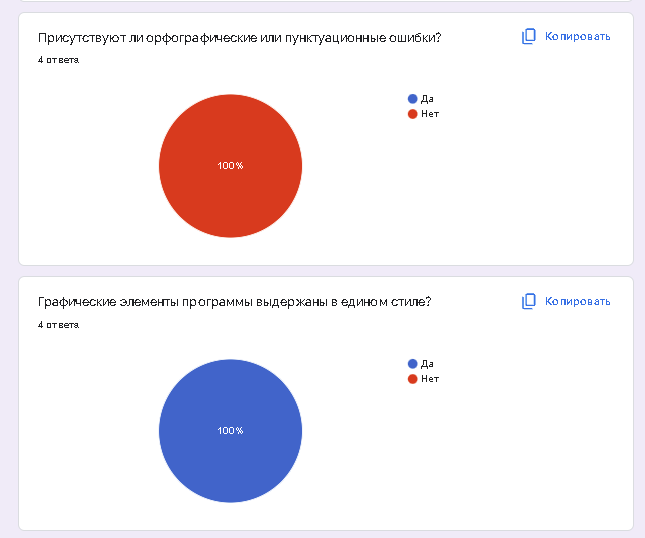
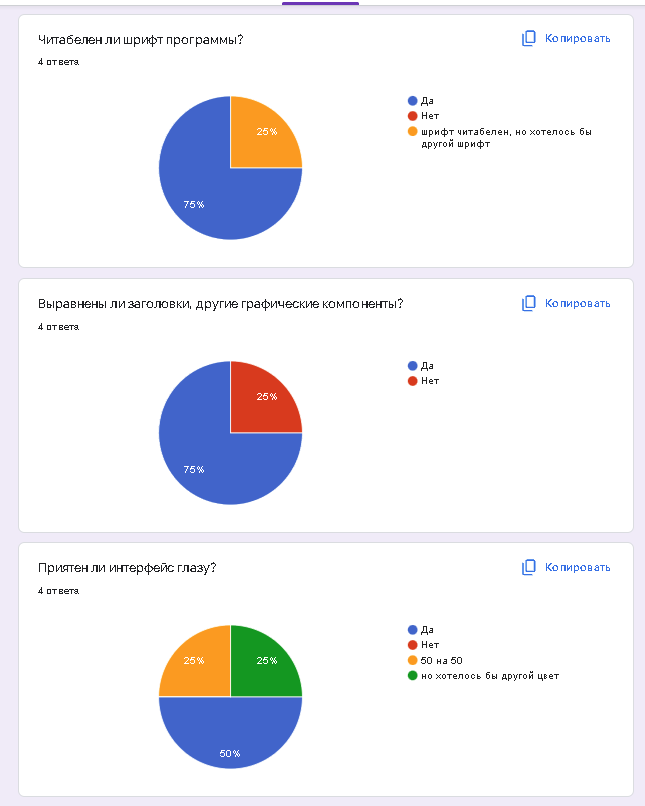
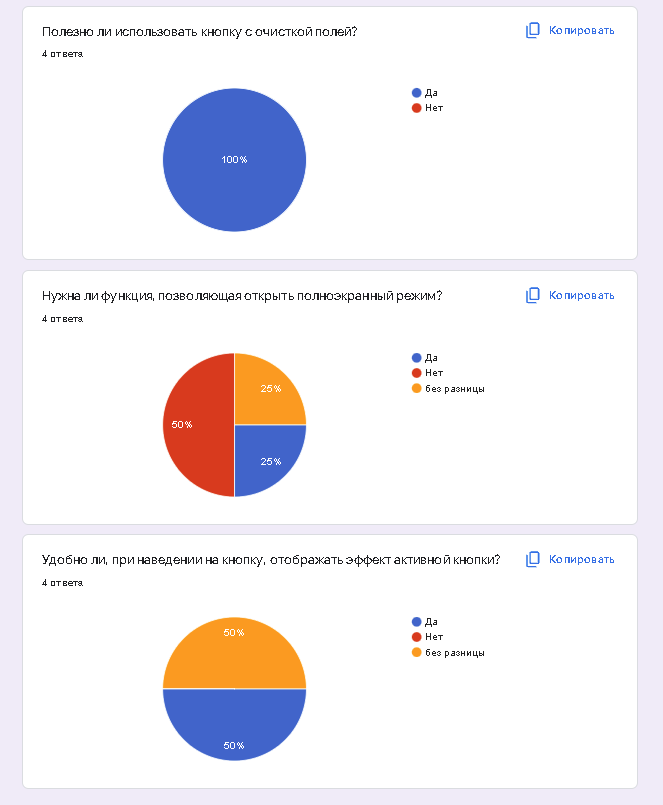


Рисунок 3 – Пользовательский опрос по интерфейсу программы

Опросник по удобству использования программы представлен на рисунке 4.

В результате проведенного опроса, большую часть пользователей устраивает программа: как по функциональной части, так и по графической.

Рисунок 4 – Пользовательский опрос по удобству использования программы



**3.2 Позитивное и негативное тестирование**

Позитивное тестирование – это процесс проверки на корректное поведение согласно техническим требованиям и документации. Позитивное тестирование выполняется для обеспечения того, что система делает именно то, что ожидается.

Тестирование, в соответствии с позитивным сценарием представлено в таблице 3.

Таблица 3 – Тестирование по позитивному сценарию

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Проверка** | **Ожидаемый результат** | **Фактический результат** |
| **1.** | Необходимо в текстовые поля написать любые значения | Программа выдаст тип треугольника | Фактический результат соответствует ожидаемому |
| **2.** | Необходимо написать в текстовые поля одинаковые значения | Программа выдаст сообщение, что по трем сторонам получен равносторонний треугольник | Фактический результат соответствует ожидаемому |
| **3.** | Необходимо написать в текстовые поля два одинаковых значения, а третье любое | Программа выдаст сообщение, что по трем сторонам получен равнобедренный треугольник | Фактический результат соответствует ожидаемому |
| **4** | Необходимо написать в текстовые поля три разных значения | Программа выдаст сообщение, что по трем сторонам получен разносторонний треугольник | Фактический результат соответствует ожидаемому |
| **5** | Необходимо нажать на кнопку «Случайные числа» | В текстовые поля подставятся случайные числа | Фактический результат соответствует ожидаемому |
| **6** | Необходимо нажать на кнопку «Очистка полей» | Все значения в текстовых полях сотрутся | Фактический результат соответствует ожидаемому |
| **7** | Необходимо нажать на кнопку «Инструкция по использованию» | Откроется окно, в котором написана инструкция по использованию программы | Фактический результат соответствует ожидаемому |

Негативное тестирование — это тестирование на данных или сценариях, которые соответствуют нештатному поведению ПО.

Негативное тестирование гарантирует, что приложение продолжит работу в случае ошибки или непредвиденного поведения со стороны пользователя. С его помощью можно определить, как система реагирует на неожиданности.

Тестирование программы по негативному сценарию представлено в таблице 4.

Таблица 4 – Тестирование по негативному сценарию

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Проверка** | **Ожидаемый результат** | **Фактический результат** |
| **1.** | Необходимо открыть программу, а далее развернуть на полноэкранный режим | Программа не откроется на полный экран. Вместо этого появится в левом углу экрана | Фактический результат соответствует ожидаемому |
| **2.** | Необходимо открыть программу, далее в любое текстовое поле ввести символ | Программа не даст пользователю ввести символ | Фактический результат соответствует ожидаемому |
| **3.** | Необходимо открыть программу, далее в любое текстовое поле ввести букву (русскую или латинскую) | Программа не даст пользователю ввести букву | Фактический результат соответствует ожидаемому |
| **4.** | Необходимо открыть программу, далее в любое текстовое поле ввести ноль как первое число | Программа не даст пользователю ввести ноль, как первое число | Фактический результат соответствует ожидаемому |

**3.3 Тестирование методом белого и черного ящика**

WHITE BOX TESTING — это тестирование внутренней структуры, дизайна и кодирования программного решения. В этом типе тестирования код виден тестеру. Основное внимание уделяется проверке потока входных и выходных данных через приложение, улучшению дизайна и удобства использования, усилению безопасности.

Для проведения тестирования методом белого ящика, будет использоваться юнит-тестирование.

Юнит-тестирование — процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы.

Актуальность использования юнит-тестов заключается в том, что достаточно с оперативной скоростью можно проверить, привели ли очередное изменение кода к негативным последствиям, т.е. – к появлению ошибок уже в тех местах, где было произведено тестирование программы. Помимо этого, юнит тесты облегчают поиск и устранение ошибочного результата.

В результате детальнейшего и фундаментального анализа теоретического и практического материала, были составлены следующие юнит-тесты для программы, вычисляющая тип треугольника. Для каждого описанного действия были созданы тестовые сценарии.

Тестовый сценарий был создан как отдельный проект, в котором была добавлена ссылка на основной проект. (ПКМ – «Ссылки» – «Добавить ссылку»).

Методы и тестовый сценарий функций указаны ниже.

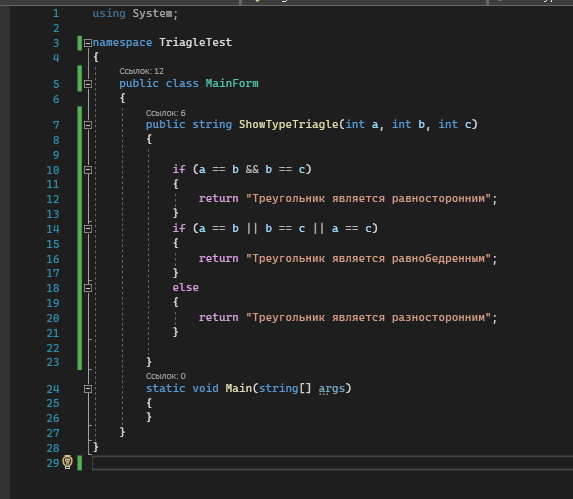


Рисунок 4 – Метод для программы, определяющий тип треугольника

Тестовые сценарии, возвращающие значения TRUE – т.е., те сценарии, которые возвращают корректный тип треугольника при указанных значения представлены на рисунке 5.



Рисунок 5 – Тестовые сценарии, возвращающие значения TRUE

Тестовые сценарии, возвращающие значения FALSE – т.е., те сценарии, которые возвращают некорректный тип треугольника при указанных значения представлены на рисунке 6.

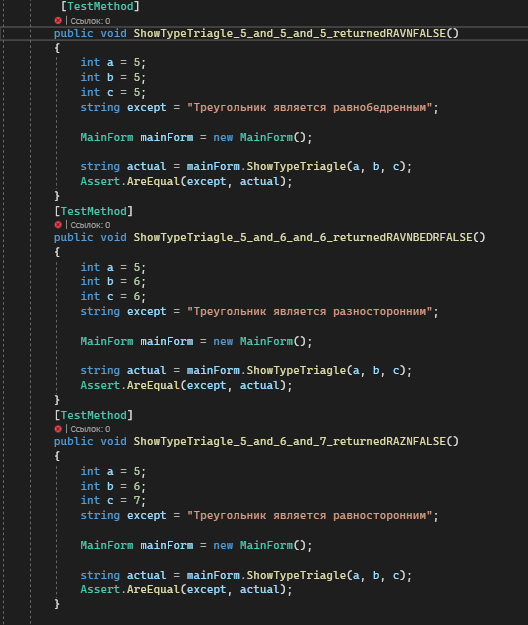


Рисунок 7 – Тестовые сценарии, возвращающие значения FALSE

Результат выполнения тестов представлен на рисунке 7. Данный обозреватель тестов отображает результат пройденного тестирования (сколько тестов было пройдено/не пройдено), а также показывает ожидаемый и фактический результат для сценариев с FALSE.

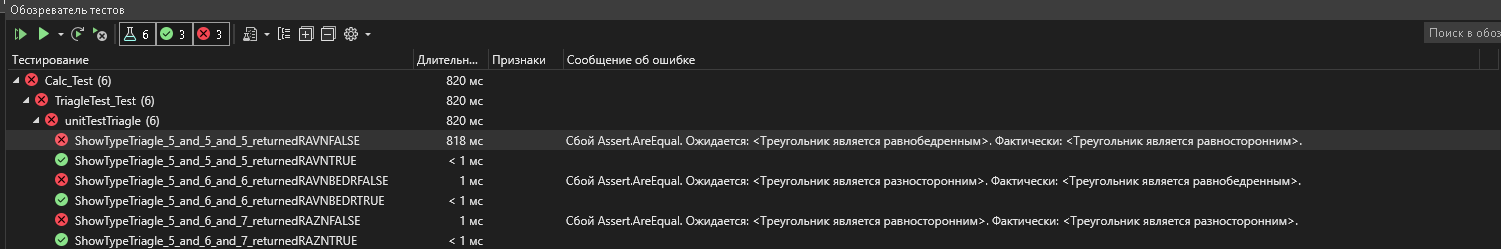


Рисунок 7 – Обозреватель тестов

Тестирование методом «черного ящика» – это функциональное и нефункциональное тестирование без доступа к внутренней структуре компонентов системы. Метод тестирования «черного ящика» – процедура получения и выбора тестовых случаев на основе анализа спецификации (функциональной или нефункциональной), компонентов или системы без ссылки на их внутреннее устройство. Тестирование методом «черного ящика» указано в таблице 5.

Таблица 4 – Тестирование методом «Черного ящика»

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Проделанное действие** | **Ожидаемый результат** | **Фактический результат** |
| 1 | Открыть программу | Программа должна открыться | Программа открылась |
| 2 | Нажать на кнопку «Инструкция по использованию» | После нажатия на эту кнопку, появится окно, в котором будет прописана инструкция по использованию приложения | После нажатия на эту кнопку, появилось окно, в котором прописана инструкция по использованию приложения |
| 3 | Ввести одну сторону, а остальные две оставить пустыми | Программа выдаст ошибку о том, что поля/поле не были заполнены | Программа выдала ошибку о том, что поля/поле не были заполнены |
| 4 | Необходимо нажать на клавишу «TAB» | При нажатии на клавишу «TAB», произойдет перемещение по элементам программы | При нажатии на клавишу «TAB», происходит перемещение по элементам программы |
| 5 | Необходимо ввести символ в любое текстовое поле | Программа не отреагирует на действия (т.к. разрешено использовать только цифры) | Программа не отреагировала на действия |
| 6 | Необходимо ввести любую букву в любое текстовое поле (русскую или латинскую) | Программа не отреагирует на действия (т.к. разрешено использовать только цифры) | Программа не отреагировала на действия |
| 7 | Необходимо открыть программу в полноэкранном режиме | Программа откроется на полный экран. Все элементы станут адаптивны, независимо от масштаба окна. | Программа не открылась на полный экран |
| 8 | Необходимо нажать на кнопку «Случайные числа» | После нажатия, сгенерируются в три текстовых поля случайные числа | После нажатия, сгенерировались в три текстовых поля случайные числа |
| 9 | Необходимо ввести ноль как первое число | В текстовое поле введется первым числом ноль | В текстовое поле не ввелось первым число ноль, т.к. при установке нуля как первое число, невозможно будет сгенерировать тип треугольника |
| 10 | Необходимо заполнить текстовые поля, а далее нажать на кнопку «Очистка полей» | Текстовые поля будут очищены | Текстовые поля очистились |
| 11 | Необходимо свернуть программу | Программа свернется | Программа свернулась |
| 12 | Необходимо заполнить текстовые поля, а далее нажать на кнопку «ОК» | Программа сгенерирует тип треугольника и в новом окне выведет результат | Программа сгенерировала тип треугольника и в новом окне вывела результат |

В результате тестирования методом «черного ящика» в таблице были выявлены соответствия в виде ожидаемого и фактического результата.