Лабораторная работа

"Разработка и тестирование программного продукта"

Задание: разработать и протестировать программный продукт.

План работы:

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).
6. Для каждого тестирования разработать отслеживающую документацию (тест-кейсы, чек-листы, баг - репорты).
7. Разработать документ. содержащий документацию по тестированию или ссылки на документы

Тест план

Тестируемый объект: программный продукт под названием “Определение типа 🔺”. Приложение носит следующий смысл и поставленные для решения ему задачи: ввод данных о сторонах треугольника, проверка вводимых данных на ошибки, сравнение стороны и отображение данных пользователю. Результатов работы программы всего может быть четыре, а это: “Разносторонний”, “Равнобедренный”, “Равносторонний” или же “ERROR” - в случае ошибки вводимых данных, дополнительно выводя окно с уточнением ошибки.

Тестирование будет проходить: функциональная часть и пользовательский интерфейс приложения. В функциональной части акцентировать внимание на работоспособность модуля определения типа треугольника и ввод некорректных данных пользователем. Пользовательский интерфейс протестировать удобство пользования, а также интуитивное понимание визуальность части приложения.

Тестирование функционального фрагмента будет проводится путем ввода различных данных, будут использованы следующие виды тестирования Ux-testing, positive, negative, white-box, black-box с последующим описание результата выполнения программного продукта.

Риски тестирования:

* неверная оценка трудозатрат;
* изменение требований со стороны заказчика в ходе реализации;
* увольнение/перераспределение людей;
* низкая продуктивность;
* несогласованность плана релиза.

Управление рисками:

* выявления рисков, которые могут помешать целям проекта;
* анализа;
* планирования;
* мониторинга и контроля.

**Проведение различного вида тестирования.**

1. **GUI & UX tests**

1.1 GUI

|  |  |
| --- | --- |
| **Тест** | **Результат** |
| Проверка работоспособности масштабирования, при помощи базового функционала WPF. | Внутренние элементы приложения соответственно масштабируются и принимают более удобную в соответствие. |
| Проверка содержащего контента в демонстрируемом окне. | Орфографических, пунктуационных и других ошибок не было обнаружено. |
| Используемый элемент выделяется, при использование. | При наведении курсора на элемент, он выделяется синим контуром, а при вводе данных появляется вертикальный штрих. |
| Масштабируемость размера текстового контента в окне приложения. | Размер шрифта не увеличивается, но окно с расположенным текстом меняет позцию и размер. |

1.2 UX

* Преимущества функционала:
  + - * Ввод данных для сторон треугольника;
      * Автоматическая проверка данных;
      * Подсказки в случае введения неправильного значения;
      * Возможность перезагрузить приложение “изнутри”;
* Недостатки функционала:
  + - * Нету возможности сохранения результата;
      * Отсутствует возможность просмотра последних вычислений;

1. **Positive**

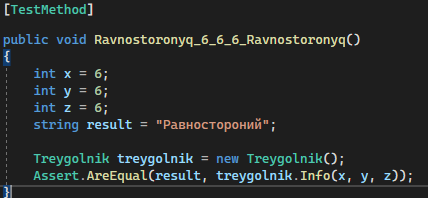
Тестирование по позитивному сценарию. Пользователь запускает программу, следом с соответствующими “заголовками” для ввода, вводит данные и нажимает на кнопку “Вычислить” - следом получает соответствующий результат от вводимых данных.

1. **Negativ**

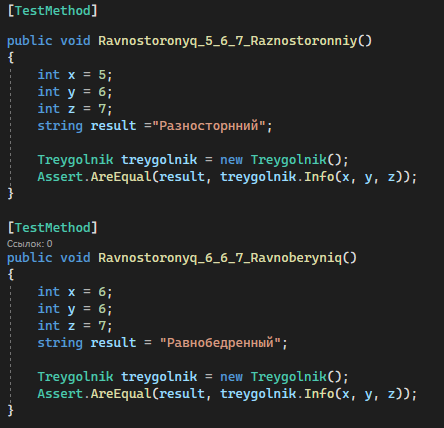
Тестирование по негативному сценарию. Пользователь запускает программу, вводит не соответствующие данные, либо символы в окна для получения данных и следом нажимает на кнопку “Вычислить” - получает уведомление об ошибки и поле вывода результата “ERROR”.

1. **White-box**

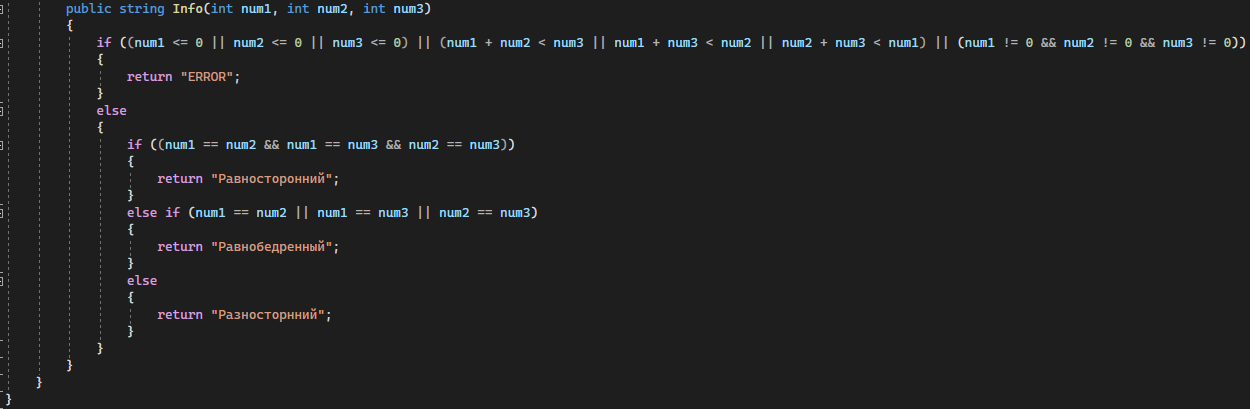
4.1 Разработано несколько методов, по структуре структуре юнит-теста: определить, сделать, проверить.



4.2 Созданы наборы тестовых методов для функции, которая проверяет треугольник на принадлежность к так типам как: равносторонний, разносторонний, несуществующий, равнобедренный.



4.3 Сама тестируемая функция, они принимает 3 аргумента (длину каждой стороны треугольника) после начинается сравнение сторон, для определения принадлежности типа треугольника.



4.4 Вводимые данные в unit-test и ожидаемый результат:

* Треугольник равнобедренный – ожидаемый результат «равнобедренный», значения длин сторон треугольника 6, 6, 7;
* Треугольник равносторонний – ожидаемый результат «равносторонний», значения длин сторон треугольника 6, 6, 6;
* Треугольник разносторонний – ожидаемый результат «равнобедренный», значения длин сторон треугольника 5, 6, 7;

1. Black-box
   1. Чек-лист

|  |  |
| --- | --- |
| Функционал | Проведено |
| Запуск приложения | Успешно |
| Ввод всех длин сторон треугольника | Успешно |
| Возможность пользователя самому начать работу функционала | Успешно |
| Вывод результата работы приложения | Успешно |
| Перезагрузка приложения, при нажатие на кнопку | Успешно |
| Защита на ввод некорректных данных | Успешно |

**Чек-лист требований качества продукта**.

1. Функциональность
   * функциональная неисправность;
   * соответствие стандартам;
   * функциональная совместимость;
   * безопасность.
2. Эффективность
   * эффективность по времени;
   * эффективность использования ресурсов.
3. Удобство использования
   * простота и удобство использования;
   * понятность.
4. Надежность
   * завершенность;
   * устойчивость к отказам.
5. Портативность
   * удобство установки;
   * совместимость.