Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационных технологий и анализа данных Центр программной инженерии

ОТЧЕТ

к лабораторной работе №3 по дисциплине:

|  |
| --- |
| «Технология разработки программных комплексов» |
| Тестирование программ |

наименование темы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент |  | | | | |
| группы: | ИСТб-19-2 |  |  |  | Комогорцева Ю.В. |
| Проверил: | шифр группы  доцент |  | подпись  15 баллов |  | Фамилия ИО  Бахвалова З.А. |
|  | должность |  | подпись |  | Фамилия ИО |

Иркутск. 2022 г.

**Оглавление**

[Общее задание 3](#_Toc100018318)

[1 Постановка задачи 4](#_Toc100018319)

[2 Описание задачи и алгоритма ее решения 6](#_Toc100018320)

[3 Структурное тестирование 7](#_Toc100018321)

[4 Белый ящик 9](#_Toc100018322)

[5 Черный ящик 15](#_Toc100018323)

[6 Тестирование пользовательского интерфейса 21](#_Toc100018324)

[7 Протокол по результатам тестирования 24](#_Toc100018325)

[8 Выводы о качестве тестируемого программного продукта и наиболее удачных тестах 25](#_Toc100018326)

[9 Таблицы спецификаций 26](#_Toc100018327)

[Литература 28](#_Toc100018328)

# Общее задание

**Цель работы -** знакомство с существующими стратегиями тестирования,

приобретение навыков выбора стратегии и разработки тестов для отдельных задач, сравнение и оценка различных методов тестирования и их возможностей.

**Задание:**

Написать программу расчета площади заштрихованной области геометрической фигуры. Программа должна:

* строить геометрические фигуры и выводить их на экран,
* считать площадь заштрихованной области фигуры,
* изменять размеры построенной фигуры в пределах области окна,
* изменять размеры построенной фигуры пропорционально изменению размеров окна,
* фигуры, выводимые на экран, должны соответствовать введенным размерам,
* параметры фигур вводятся в сантиметрах(стороны) и градусах (углы),

**Индивидуальный вариант № 8**

Рисунок 1 – Геометрические фигуры

# 1 Постановка задачи

**Бизнес-требования**

Программа должна строить указанные на Рис.1 геометрические фигуры и считать площадь заштрихованной области фигуры.

**Пользовательские требования**

Пользовательские требования определяют набор пользовательских задач, которые должна решать программа, а также способы (сценарии) их решения в системе. Можно представить в виде диаграммы прецедентов (use case), наглядно демонстрирующей функции программы и действия пользователя;

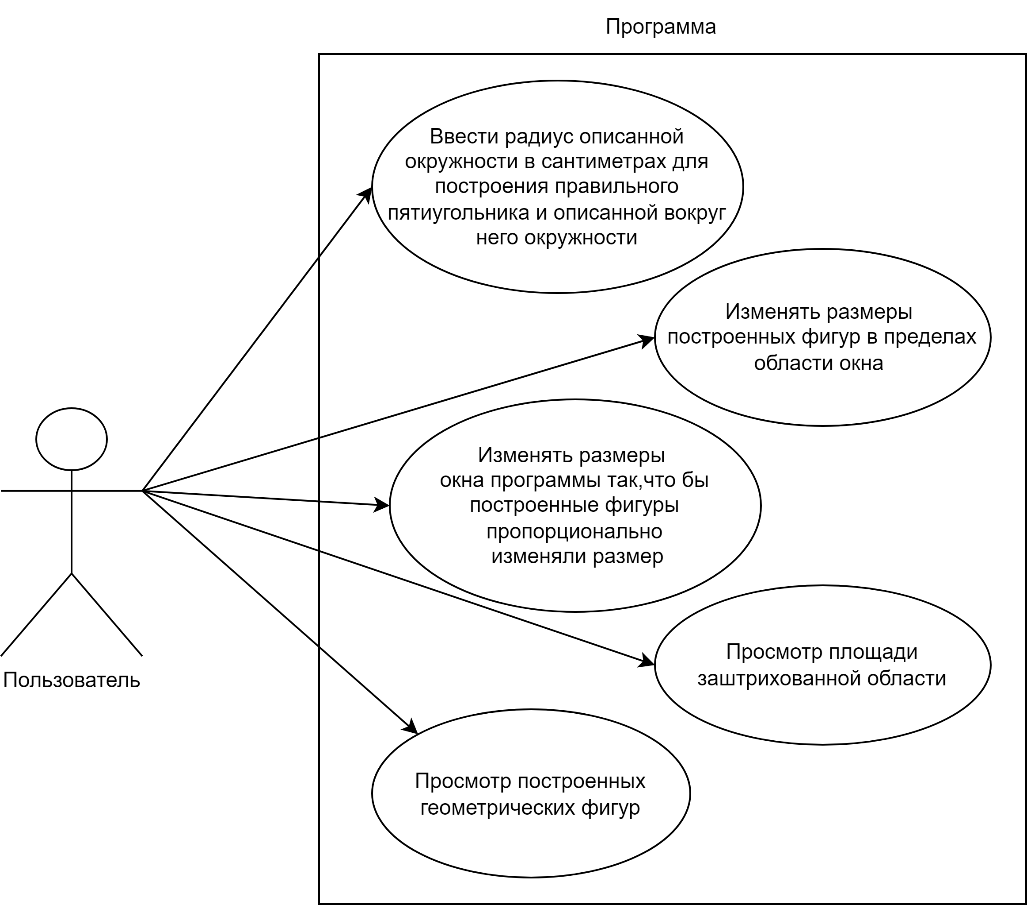


Рисунок 2 – Use case

**Функции программы должны включать:**

Просмотр построенных геометрических фигур – правильный пятиугольник и описанная вокруг него окружность. Область вне пятиугольника и внутри окружности должна быть выделена цветом.

Построение геометрических фигур на основе заданного радиуса описанной окружности. Радиус задается в сантиметрах. Радиус — это положительное число. Верхняя граница радиуса должна быть не больше 13 см, так как фигура начинает выходить за границы приложения и ее становится не видно. Центр геометрических фигур должен располагаться в центре элемента для отображения фигур. Первый раз при запуске фигуры строятся по размерам R=2 см.

Если пользователь в поле радиуса введет не положительное число, то программа должна сообщить об ошибке.

Вычисление площади для области вне пятиугольника, пересекающейся с областью внутри описанной вокруг пятиугольника окружности. Вычисление площади происходит в реальном времени после каждого изменения размера фигур. Значение площади округляется до 3 чисел после запятой.

Изменение размера построенных фигур должно осуществляться с помощью ползунка(каретки). Как только значение ползунка изменяется, программа умножает радиус на измененный промежуток, перестраивает геометрические фигуры и пересчитывает площадь. Нижняя граница ползунка может быть положительным числом.

Изменение размера окна вызывает пропорциональное изменение размера геометрических фигур, а также перестроение фигуры и пересчет площади.

**Нефункциональные требования:**

Окно приложения должно быть разбито на 2 части: изображение с геометрическими фигурами и рабочая область. Изображение с геометрическими фигурами прикреплено к верхней левой части окна и является квадратным. Рабочая область располагается справа от изображения и содержит: каретку для изменения размера, поле для ввода радиуса и поле для вывода площади.

При изменении размера окна, изображение остается квадратным, но меняет размер, а всю остальную часть занимает рабочая область.

Сообщения об ошибке ввода должно выводиться справа от поля ввода.

Интерфейс программы должен быть выдержан в стандартных цветах.

**2 Описание задачи и алгоритма ее решения**

Для построения описанной окружности нужен только радиус.

Для построения правильного пятиугольника вписанного в окружность с использованием только радиуса окружности понадобиться формула вычисления координат X и Y для каждой из вершин пятиугольника:

, где R-радиус, α-угол поворота, который должен увеличиваться на 72 градуса.

Формула вычисления стороны правильного пятиугольника:

Для вычисления площади правильного пятиугольника нужна сторона пятиугольника:

Для вычисления площади заштрихованной области необходимо из площади окружности вычесть площадь пятиугольника по формуле:

**3 Структурное тестирование**

Структурный контроль исходного текста представляет собой набор процедур и приемов обнаружения ошибок при изучении текста группой специалистов, в которую входят автор программы, проектировщик, специалист по тестированию и координатор (компетентный программист, но не автор программы).

Программа была проанализирована по списку вопросов для выявления исторически сложившихся общих ошибок программирования. Результаты изложены в таблице структурного контроля.

Таблица структурного контроля

|  |  |  |  |
| --- | --- | --- | --- |
| Номер вопроса | Строки, подлежащие проверке | Результат проверки | Вывод |
| 1.1 | 16-20 | vertices = 5  pix = 37.936  proportion = 1f  \_oldWidth, \_oldHeight, \_oldTrackBar – инициализируются при запуске | Все переменные инициализированы |
| 1.2 | 36 | Массив verticies состоящий из Point содержит только 5 точек (вершин прямоугольника) | Размеры массивов не превышены |
| 1.3 | Матрицы не используются | | |
| 1.4 | Переменные со сходными именами не используются | | |
| 1.5 | Файлы не используются | | |
| 1.6 | 34,36 | Индексы за границы массивов не выходят, типы переменных соответствуют при наложении формата | Не типизированные переменные, открытые массивы не используются, переменные создаются в динамической памяти, верно. |
| 2.1 | 83,88,93,94,74,75 |  |  |
| 2.2 | Арифметические переменные не используются | | |
| 2.3 | Вычислений с переменными различных типов нету | | |
| 2.4 | Переполнение разрядной сетки или ситуация машинного нуля невозможна | | |
| 2.5 | 83 | Происходит округление до 3 цифр после запятой | Вычисление соответсвует заданному требованию точности. |
| 3.1 | 70-78 |  | Цикл будет завершен корректно |
| 3.2 |  |  | Программа будет завершена только пользователем |
| 3.3 | 70-78 |  | Циклы продолжают вычисления даже после нарушения условия входа |
| 3.4 | Поисковых циклов нет | | |

Структурный контроль позволяет найти ошибки, увидеть сделанные программистом ошибки, получить возможность оценить свой стиль программирования, выбор алгоритмов и методов тестирования. Структурный контроль является способом раннего выявления частей программы, с большей вероятностью содержащих ошибки, что позволяет при тестировании уделить внимание именно этим частям.

**Преимущества структурного тестирования:**

* Обеспечивает тщательное тестирование программного обеспечения.
* Это помогает в обнаружении дефектов на ранней стадии.
* Это помогает в устранении мертвого кода.
* Это не отнимает много времени, поскольку это в основном автоматизировано.

**Недостатки структурного тестирования:**

* Требуется знание кода для выполнения теста.
* Требуется обучение инструменту, используемому для тестирования.
* Иногда это дорого.

**4 Белый ящик**

Для тестирования стратегии методом «Белый ящик» был выбран метод radiusChanged, который отвечает за изменения радиуса если изменится значение каретки. На вход в метод поступает две переменные: a – значение каретки после изменения значения каретки, b – значение каретки до изменения значения каретки. Соответственно если значение увеличилось относительно прошлого значения, то и размер необходимо увеличить.

Функции, выполнение которых зависит от условия внутри метода:

1. Увеличить размер фигуры, выполняется если a>b
2. Уменьшить размер фигуры, выполняется если a<b
3. Оставить размер фигуры прежним, выполняется если a=b

Код метода:

private void radiusChanged(int a,int b) {

if (a > b) {

numericUpDown1.Value = numericUpDown1.Value+(numericUpDown1.Value \* (decimal)0.1);

}

else if(a < b)

{

numericUpDown1.Value = numericUpDown1.Value- (numericUpDown1.Value \* (decimal)0.1);

}

}

Для тестирования методом белого ящика была составлена схема алгоритма и ее граф передачи управления.

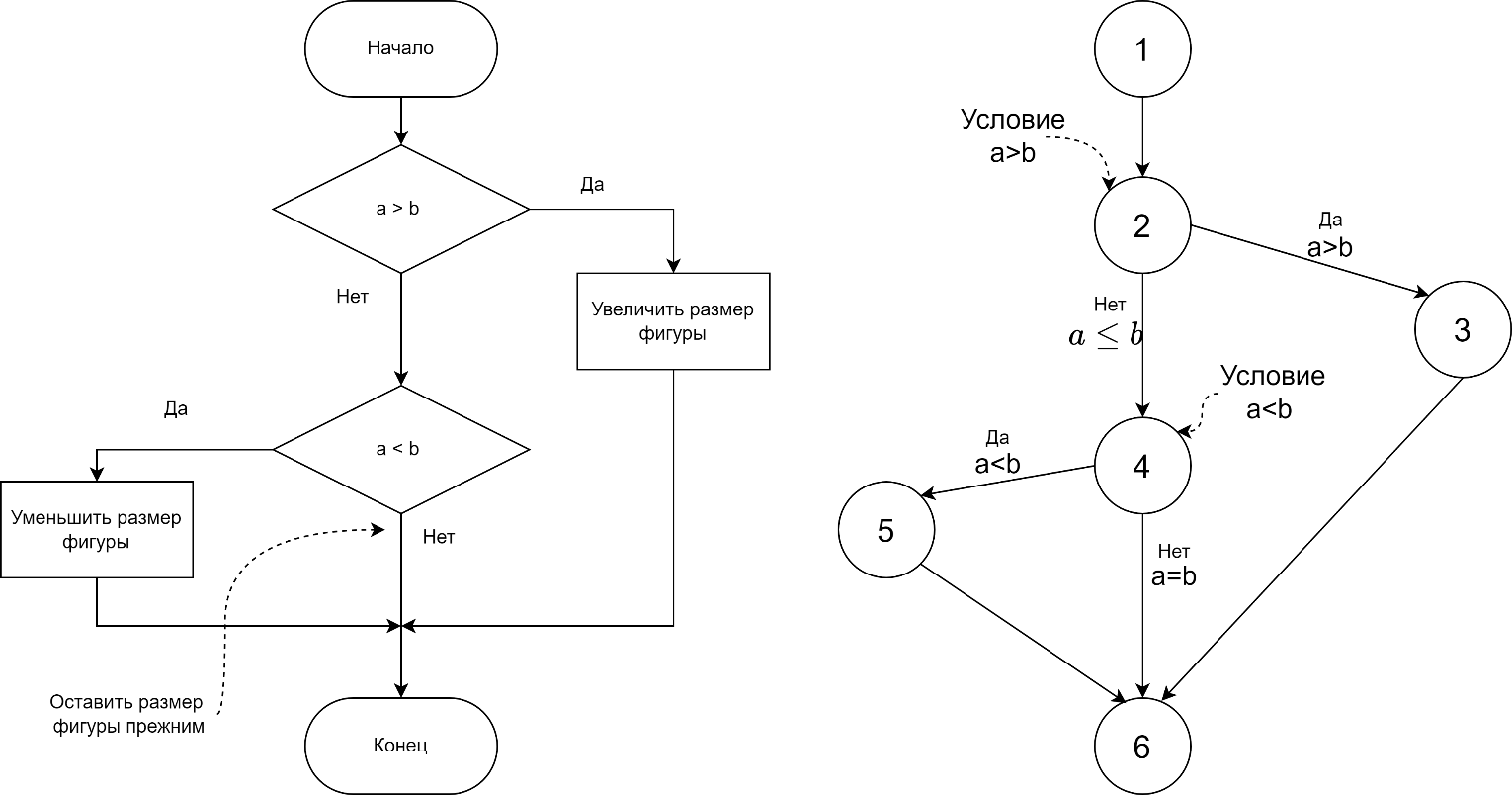


Рисунок 3 – Схема алгоритма метода и ее граф передач управления

**Метод покрытия операторов** подразумевает выполнение каждого оператора программы, по крайней мере, один раз. Поэтому нужно проверить все ли операторы выполняются при выполнении функций метода, что даст нам возможность убедиться в необходимости каждого из операторов.

Составим тесты:

1. Для выполнения функции увеличить размер фигуры потребуются входные значения a=5, b=4. Если мы воспроизведём выполнение алгоритма с такими входными условиями, то пройдем по пути 1236. Следовательно операторы 1,2,3,6 являются необходимыми.
2. Для выполнения функции уменьшить размер фигуры потребуются входные данные a=3, b=8. Если мы воспроизведём выполнение алгоритма с такими входными условиями, то пройдем по пути 12456. Следовательно операторы 1,2,4,5,6 являются необходимыми.

На данном методе достаточно двух тестов, потому что мы убедились в необходимости всех операторов алгоритма.

Таблица тестов стратегии «белого ящика» по методу покрытия операторов.

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Назначение теста | Значения исходных данных | Ожидаемый результат |
| 1 | Увеличить размер фигуры | a=5  b=4 | Путь 1236 |
| 2 | Уменьшить размер фигуры | a=3  b=8 | Путь 12456 |

**Метод покрытия решений** подразумевает, что каждое решение на этих тестах принимает значение «истина» или «ложь», по крайней мере, один раз. Поэтому в данном методе целью составления тестов является проверка наличия и необходимости решений при выполнения заданных для этого метода функций.

Составим тесты:

1. Для выполнения функции увеличить размер фигуры нужно проверить есть ли условие, которое проверит увеличилось ли значение. Для теста на увеличение размера фигуры потребуются входные значения a=5, b=4. Если мы воспроизведём выполнение алгоритма с такими входными условиями, то пройдем по пути 1236. Оператор 2 является решением, которое проверяет a>b, где а – новое значение, b – старое значение. Значит оператор 2 проверяет стало ли больше новое значение, именно это условие и требовалось найти. Вывод: условие для увеличения размера фигуры существует.
2. Для выполнения функции уменьшить размер фигуры нужно проверить есть ли условие, которое проверит уменьшилось ли значение. Для теста на уменьшение размера фигуры потребуются входные значения a=3, b=8. Если мы воспроизведём выполнение алгоритма с такими входными условиями, то пройдем по пути 12456. В данном пути встретились два решения 2 и 4, значит нужно проверить оба. Оператор 2 является решением, которое проверяет стало ли больше новое значение, что не подходит для нашего теста. Однако в необходимости этого решения мы убедились в предыдущем тесте. Оператор 4 является решением, которое проверяет a<b, где а – новое значение, b – старое значение. Значит оператор 4 проверяет стало ли меньше новое значение, именно это условие и требовалось найти. Вывод: условие для уменьшения размера фигуры существует.
3. Для выполнения функции оставить размер фигуры прежним нужно проверить есть ли условие, которое проверит что значение не изменилось. Для теста на оставление размера фигуры прежним потребуются входные значения a=3, b=3. Если мы воспроизведём выполнение алгоритма с такими входными условиями, то пройдем по пути 1246. В данном пути встретились два решения 2 и 4, значит нужно проверить оба. Оператор 2 является решением, которое проверяет стало ли больше новое значение, однако при наших входных данных значение не стало больше и решение принимает значение ложь. Значит можно сделать вывод что решение 2 является условием, которое учитывает ситуацию данного теста, однако не покрывает полностью. Оператор 4 является решением, которое проверяет a<b, где а – новое значение, b – старое значение. Значит оператор 4 проверяет стало ли меньше новое значение. Но в нашем случае решение принимает значение ложь, и тем самым произошла проверка на ситуацию данного теста. Вывод: условие для оставления размера фигуры прежним существует.

Таблица тестов стратегии «белого ящика» по методу покрытия решений.

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Назначение теста | Значения исходных данных | Ожидаемый результат |
| 3 | Увеличить размер фигуры | a=5  b=4 | Путь 1236 |
| 4 | Уменьшить размер фигуры | a=3  b=8 | Путь 12456 |
| 5 | Оставить размер фигуры прежним | a=3  b=3 | Путь 1246 |

Используя **метод покрытия условий,** формируют некоторое количество тестов, достаточное для того, чтобы все возможные результаты каждого условия в решении были выполнены, по крайней мере, один раз. В связи с тем, что в представленном коде программы решения имеют только одно условие, то решение и условие совпадает, значит нет необходимости использовать данный метод.

**Метод покрытие решений/условий** требует составить тесты так, чтобы все возможные результаты каждого условия выполнились, по крайней мере, один раз, все результаты каждого решения выполнились, по крайней мере, один раз и каждой точке входа управление передается, по крайней мере, один раз. В выбранном методе как указывалось ранее условия и решения совпадают. Поэтому в данном методе целью составления тестов является проверка соответствия каждого решения и результата каждого решения при выполнении заданных для этого метода функций.

Составим тесты:

1. Для выполнения функции увеличить размер фигуры потребуются входные значения a=5, b=4. В данном тесте необходимо проверить что при увеличении значения размер действительно увеличится. Если мы воспроизведём выполнение алгоритма с заданными входными условиями, то пройдем по пути 1236. Из тестов, описанных выше, мы выяснили что за проверку увеличилось ли значение отвечает оператор 2. В данном случае решение оператора 2 принимает значение истина, значит следующим шагом должно идти увеличение размера. Из рисунка 3 видно, что оператор 3 соответствует действию увеличить размер. Вывод: при выполнении функции увеличить размер фигуры решения и результаты решения соответствуют.
2. Для выполнения функции уменьшить размер фигуры потребуются входные данные a=3, b=8. В данном тесте необходимо проверить что при уменьшении значения размер действительно уменьшится. Если мы воспроизведём выполнение алгоритма с такими входными условиями, то пройдем по пути 12456. Из тестов, описанных выше, мы выяснили что за проверку уменьшилось ли значение отвечает оператор 4. В данном случае решение оператора 4 принимает значение истина, значит следующим шагом должно идти уменьшение размера. Из рисунка 3 видно, что оператор 5 соответствует действию уменьшить размер. Вывод: при выполнении функции уменьшить размер фигуры решения и результаты решения соответствуют.
3. Для выполнения функции оставить размер фигуры прежним потребуются входные данные a=3, b=3. В данном тесте необходимо проверить что если не изменить размер фигуры, то после выполнения метода он останется неизменным. Если мы воспроизведём выполнение алгоритма с такими входными условиями, то пройдем по пути 1246. Из тестов, описанных выше, мы выяснили что за проверку данной ситуации отвечает оператор 2 и 4. Решение оператора 2 принимает значение ложь и следующим шагом идет оператор 4. В данном случае решение оператора 4 принимает значение ложь, значит следующим шагом должно идти окончание программы. Из рисунка 3 видно, что после принятия решения ложь в операторе 4 программа стремится к концу алгоритма. Вывод: при выполнении функции оставить размер фигуры прежним решения и результаты решения соответствуют.

Таблица тестов стратегии «белого ящика» по методу покрытия решений/условий.

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Назначение теста | Значения исходных данных | Ожидаемый результат |
| 6 | Проверить, что при увеличении значения размер действительно увеличится | a=5  b=4 | Путь 1236 |
| 7 | Проверить, что при уменьшении значения размер действительно уменьшится | a=3  b=8 | Путь 12456 |
| 8 | Проверить что если не изменить размер фигуры, то после выполнения метода он останется неизменным | a=3  b=3 | Путь 1246 |

**Метод комбинаторного покрытия условий** требует создания такого числа тестов, чтобы все возможные комбинации результатов условий в каждом решении и все точки входа выполнялись, по крайней мере, один раз. В связи с тем, что в представленном коде программы решения имеют только одно условие, то решение и условие совпадает, значит нет необходимости использовать данный метод.

Подводя итог, можно все тесты, созданные выше объединить в одну общую таблицу тестов стратегии «белого ящика».

Таблица тестов стратегии «белого ящика».

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Метод | Назначение теста | Значения исходных данных | Ожидаемый результат |
| 1 | Покрытия операторов | Увеличить размер фигуры | a=5  b=4 | Путь 1236 |
| 2 | Уменьшить размер фигуры | a=3  b=8 | Путь 12456 |
| 3 | Покрытия решений | Увеличить размер фигуры | a=5  b=4 | Путь 1236 |
| 4 | Уменьшить размер фигуры | a=3  b=8 | Путь 12456 |
| 5 | Оставить размер фигуры прежним | a=3  b=3 | Путь 1246 |
| 6 | Покрытия решений/условий | Проверить, что при увеличении значения размер действительно увеличится | a=5  b=4 | Путь 1236 |
| 7 | Проверить, что при уменьшении значения размер действительно уменьшится | a=3  b=8 | Путь 12456 |
| 8 | Проверить что если не изменить размер фигуры, то после выполнения метода он останется неизменным | a=3  b=3 | Путь 1246 |

После составленных тестов можно выделить некоторые итоги: при проектирование тестовых наборов белого ящика мы проверили структуру исходного кода отдельного метода.

Стратегия «белого ящика» состоит из методов:

1. Метод покрытия операторов не является достаточным для проверки программы.
2. Метод покрытия решений, который гарантирует покрытие операторов.
3. Метод покрытия условий, который не гарантирует покрытия решений
4. Метод покрытия решений/условий гарантирует покрытие и операторов, и решений, и условий.
5. Метод комбинаторного покрытия условий гарантирует покрытие и операторов, и решений, и условий.

Каждый из методов по отдельности не являются достаточными для тестирования программы, поэтому необходимо использовать их в совокупности.

**Преимущества тестирования белого ящика**

* Очень большая точность
* Просто автоматизировать
* Позволяет определить, где в коде произошла ошибка
* Для выполнения тестирования не нужен UI

**Недостатки тестирования белого ящика**

* Инженеры, выполняющие тестирование, должны знать программирование на достаточном уровне
* Иногда бывает невозможно протестировать каждый участок кода
* Фокус на тестировании кода может привести к ситуации, когда пропускаются баги, связанные с функциональностью. При этом код может работать правильно, но приложение не будет соответствовать требованиям.

**5 Черный ящик**

Рассмотрим создание тестов разными методами стратегии "черного ящика" с описанием и заполнением тестов в таблице, которая находится в конце раздела.

**По методу эквивалентных разбиений:**

Исходными данными программы является только радиус, именно он отвечает за все функции: построение фигуры, расчет площади и изменение размеров разными способами. Поэтому для радиуса были выделены классы эквивалентности на основе функциональных требований.

Таблица эквивалентности

|  |  |  |
| --- | --- | --- |
| Тип класса | Номер класса | Классы эквивалентности |
| Правильный | 1 | R – число |
| 2 | R – вещественное число |
| 3 | 0≤ R ≤ 13 |
| Неправильный | 4 | R – не число |
| 5 | R < 0 |
| 6 | R > 13 |

Одним тестом мы должны покрыть все правильные классы эквивалентности:

* При R=4,25 покрываются классы 1,2,3, результатом этого теста будет корректная работа программы: построенная фигура, выделенная область, рассчитанная и выведенная площадь S= 13,799

Но на каждый неправильный класс эквивалентности мы создадим отдельный тест:

* При R= абвг покрывается класс 4, результатом будет сообщение о неправильном вводе
* При R= -2 покрывается класс 5, результатом будет сообщение о неправильном вводе
* При R= 15 покрывается класс 6, результатом будет сообщение о неправильном вводе

**По методу граничных условий:**

Границы для входной величины – радиуса: 0 ≤ R ≤ 13.

Внутри рассматривать не будем значения, зато можно рассмотреть значения, которые выходят за границы на 0.001. Тем самым необходимо создать тесты, в которых проверяются значения:

* R= -0,001, результат будет сообщение о неправильном вводе
* R= 13,001, результат будет сообщение о неправильном вводе

**По методу анализа причинно-следственных связей:**

Метод применяется для создания тестов, которые максимально будут учитывать все ситуации логики программы (Например: при пересечение прямых нужны тесты для выполнения условия пересечения, что не относится к функциональности). Однако в данной программе отсутствует ситуация, для тестирования которой можно было бы применить данный метод.

**По методу предположения об ошибке:**

Метод предполагает создание соответствий неправильного класса эквивалентности и сообщения об ошибке, которое будет отображаться в программе, при вводе значений, которые входят в заданный класс эквивалентности.

Таблица сообщений об ошибке

|  |  |  |
| --- | --- | --- |
| № сообщения | Номер класса | Сообщение об ошибке |
| 1 | 5 | Вы ввели не число! Повторите ввод. |
| 2 | 6 | Вы ввели отрицательное число! Повторите ввод. |
| 3 | 7 | Вы ввели радиус фигуры, которая не влезет в экран! Повторите ввод. |

На основе всех выделенных тестов была составлена таблица тестов с результатами тестирования по стратегии «черный ящик».

**Таблица тестов с результатами тестирования по стратегии «черный ящик»**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Номер теста | Назначение теста | Значения исходных данных | Ожидаемый результат | Ожидаемая реакция программы | Результат | Реакция  программы |
| 9 | Построить фигуру, выделить область, рассчитать и вывести площадь | R=4,25 | S= 13,799 | Построенная фигура, выделенная область, выведенная площадь | S= 13,799 | Построенная фигура, выделенная область, выведенная площадь.  См. рисунок 4 |
| 10 | Проверка на правильность работы программы при вводе текста (неправильный класс эквивалентности № 4) | R= абвг |  | Программа выдает сообщение об ошибке 1 |  | Не удалось в поле ввода ввести текст, буквы не печатаются, поэтому сообщение об ошибке 1 не было выведено. |
| 11 | Проверка на правильность работы программы при вводе отрицательного числа (неправильный класс эквивалентности № 5) | R= -2 |  | Программа выдает сообщение об ошибке 2 | S=0 | Программа не выдала сообщение об ошибке, окно с изображением фигуры пустое, после ввода R = 0  См. рисунок 5 |
| 12 | Проверка на правильность работы программы при вводе текста (неправильный класс эквивалентности № 6) | R=15 |  | Программа выдает сообщение об ошибке 3 | S=129,108 | Программа не выдала сообщение об ошибке, потому что автоматически подставила в значения радиуса R= 13 |
| 13 | Проверка на корректность работы программы при вводе числа не входящего в диапазон радиуса (выход за граничные условия) | R= -0,001 |  | Программа выдает сообщение об ошибке 2 | S=0 | Программа не выдала сообщение об ошибке, окно с изображением фигуры пустое, после ввода R = 0  См. рисунок 5 |
| 14 | Проверка на корректность работы программы при вводе числа не входящего в диапазон радиуса (выход за граничные условия) | R=13,001 |  | Программа выдает сообщение об ошибке 3 | S=129,108 | Программа не выдала сообщение об ошибке, потому что автоматически подставила в значения радиуса R= 13 |

После составленных тестов можно выделить некоторые итоги: Тестирование по принципу «черного ящика» - тестирование для выяснения обстоятельств, в которых поведение программы не соответствует функциональным требованиям.

Стратегия "черного ящика" включает в себя следующие методы:

1. Метод эквивалентных разбиений, который не исследует комбинации входных условий.
2. Метод анализа граничных значений, который не позволяет проверять сочетания исходных данных и не всегда можно определить границы.
3. Метод анализа причинно-следственных связей, который неадекватно исследует граничные условия
4. Метод предположения об ошибке, работает только на интуиции для выявления ситуаций, которые могут быть не учтены при проектировании.

Каждый из методов по отдельности не являются достаточными для тестирования программы, поэтому необходимо использовать их в совокупности.

**Преимущества тестирования черного ящика**

* Позволяет найти несоответствия между спецификацией и разработанным приложением
* При тестировании «черным ящиком» тестировщик не должен глубоко погружаться в особенности разработки и архитектуры, значит не нужно иметь специальных знаний.
* Тесты могут быть автоматизированы

**Недостатки тестирования черного ящика**

* Требования могут быть неточными и/или непонятными, поэтому могут быть трудности с написанием тестов.
* Может быть сложно протестировать все части приложения.

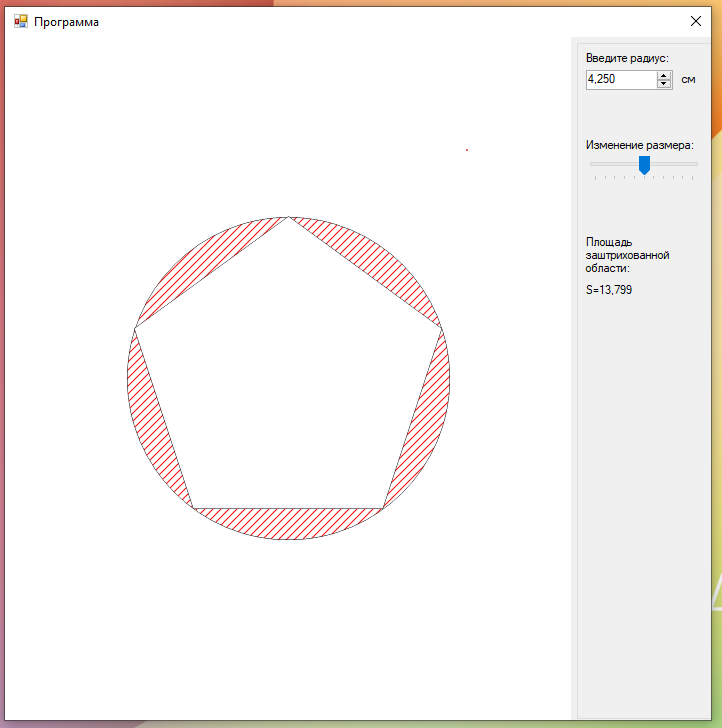


Рисунок 4 – Корректная работа программы

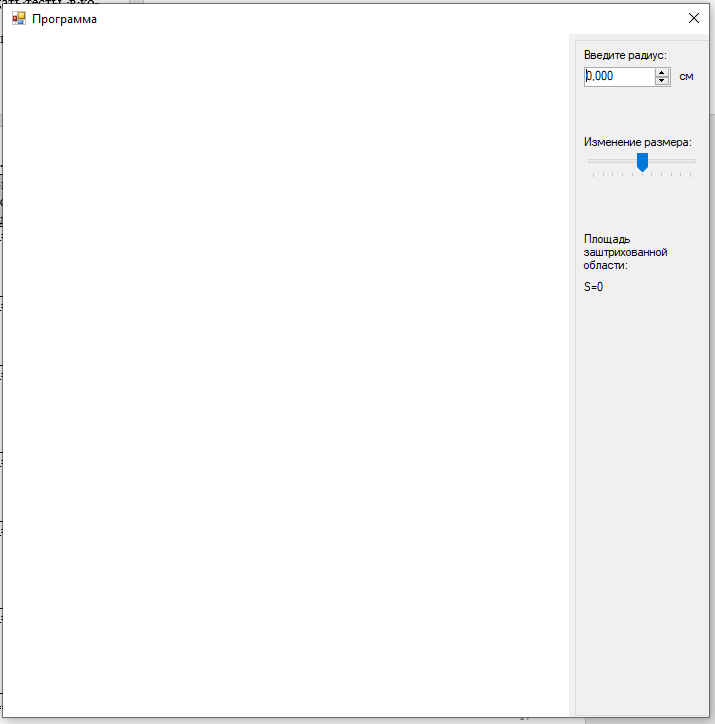


Рисунок 5 – Реакция программы при вводе отрицательных чисел

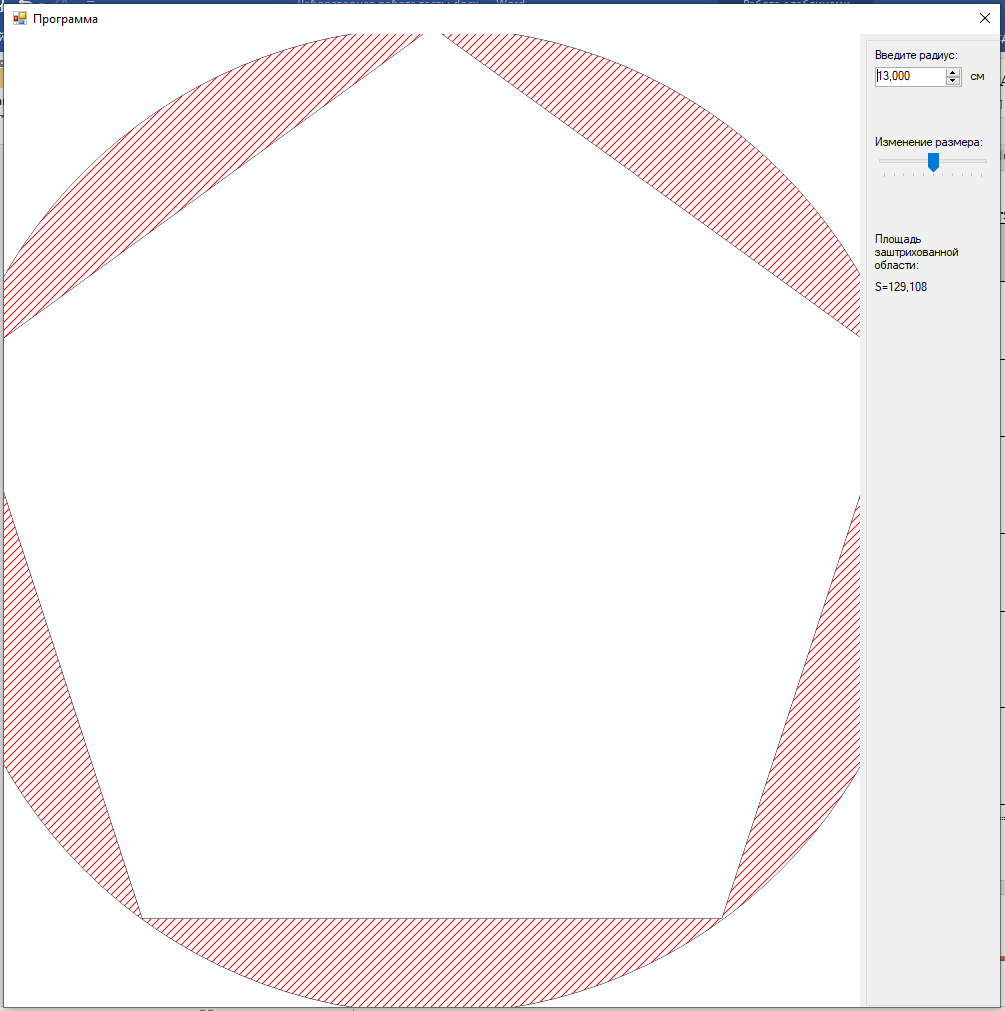


Рисунок 6 – Реакция программы при вводе R>13

**6 Тестирование пользовательского интерфейса**

Для создания тестов необходимо учесть все указанные в разделе постановки задачи требования, которые можно разделить на группы:

* требования к внешнему виду пользовательского интерфейса и формам взаимодействия с пользователем;
* требования по доступу к внутренней функциональности системы при помощи пользовательского интерфейса

Создадим таблицу тестов и результатов тестирования пользовательского интерфейса.

Таблица тестов и результатов тестирования пользовательского интерфейса

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Требование | Результат тестирования | Примечание |
| 15 | Окно приложения должно быть разбито на 2 части: изображение с геометрическими фигурами и рабочая область. | Требование выполнено | См. рисунок 4 |
| 16 | Изображение с геометрическими фигурами прикреплено к верхней левой части окна | Требование выполнено | См. рисунок 4 |
| 17 | Изображение с геометрическими фигурами является квадратным | Требование выполнено | См. рисунок 4 |
| 18 | Рабочая область располагается справа от изображения | Требование выполнено | См. рисунок 4 |
| 19 | Сообщения об ошибке ввода должно выводиться справа от поля ввода | Требование не выполнено, сообщения не отображаются нигде | См. рисунок 5 и тесты 10-14 |
| 20 | Интерфейс программы должен быть выдержан в стандартных цветах | Требование выполнено | См. рисунок 4 |
| 21 | Интерфейс позволяет выполнить функцию ввода радиуса | Требование выполнено, интерфейс для этого имеет поле для ввода радиуса. | См. рисунок 4.  Не очень удобно что поле маленького размера. |
| 22 | Интерфейс позволяет выполнить функцию вывода рассчитанной площади | Требование выполнено, интерфейс для этого имеет поле для вывода площади | См. рисунок 4 |
| 23 | Значение площади округляется до 3 чисел после запятой | Требование выполнено, результат всегда округляется | См. рисунок 4 |
| 24 | Вычисление площади происходит в реальном времени после каждого изменения размера фигур. | Требование выполнено |  |
| 25 | Построение геометрических фигур происходит в реальном времени после каждого изменения размера фигур. | Требование выполнено |  |
| 26 | Интерфейс позволяет выполнить функцию изменения размера построенных фигур | Требование выполнено, интерфейс для этого имеет каретку для изменения размера фигуры | См. рисунок 4, не удобно работает, большой шаг изменения размера, возможно лучше заменить другим элементом интерфейса |
| 27 | Интерфейс позволяет выполнить функцию изменения размера фигур за счет изменения размера окна | Требование выполнено | Работает плохо и не удобно |
| 28 | Интерфейс позволяет выполнить функцию просмотра построенных геометрических фигур и выделенной области | Требование выполнено | См. рисунок 4 |
| 29 | При изменении размера окна, изображение остается квадратным, но меняет размер, а всю остальную часть занимает рабочая область | Требование выполнено | См. рисунок 4 |
| 30 | Центр геометрических фигур должен располагаться в центре элемента для отображения фигур. | Требование выполнено | См. рисунок 4 |
| 31 | Первый раз при запуске фигуры строятся по размерам R=2 см. | Требование выполнено | См. рисунок 7 |
| 32 | На интерфейсе размер фигур задается сантиметрами и размер соответствует реальному масштабу | Требование выполнено | См. рисунок 4 |
| 33 | Интерфейс не дает возможность пользователю увеличивать размер изображения фигур, если они будет превышать размеры окна, отображающего фигуры (выступать за границы) | Требование не выполнено | При увеличение кареткой размера фигуры, когда R>13, вылазят ошибки о недопустимом значении поля ввода. Аналогичные ошибки при растягивании окна. |
| 34 | Интерфейс не дает возможность писать в поле радиус, значение выходящее за заданные границы | Требование выполнено | Автоматически подставляет в значения радиуса R=13 или R=0 |

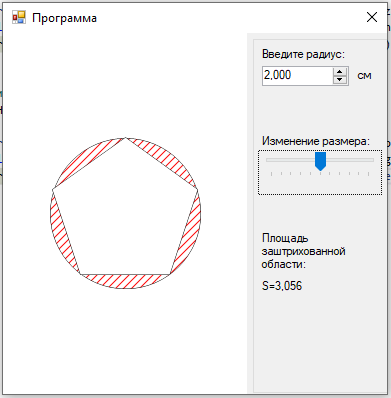


Рисунок 7 – Программа при первом запуске

По результатам созданных тестов и проведенного тестирования пользовательского интерфейса, можно сделать вывод: не все требования были выполнены, однако наиболее важные (подсчет площади, отрисовка фигуры и задание размера через поле ввода) выполняются без ошибок.

# 7 Протокол по результатам тестирования

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Название компании | | | | | | | | | | | | Конфиденциально | | | | | | | | | Отчет о проблеме | | | | | | | | | | | | | | | | | | | | | |
| № |  | | |
| Программа | |  | | | | | | | | | | | | | Выпуск | | | | |  | | | | | | Версия | | | | | | | | | | |  | | | | |
| Тип проблемы (1-6) **1** | | | | | | | | | | | | | | | Степень важности **2** | | | | | | | | | | | | | | | Приложения(Д/Н): | | | | | | | | | | | | | |
| 1 – Ошибка кодирования | | | | | | | | | | | | | | | 1 – Фатальная | | | | | | | | | | | | | | | Если да, то какие | | | | | | | | | | | | | |
| 2 – Ошибка проектирования | | | | | | | | | | | | | | | 2 – Серьезная | | | | | | | | | | | | | | |  | | | | | | | | | | | | | |
| 3 – Предложение | | | | | | | | | | | | | | | 3 – Незначительная | | | | | | | | | | | | | | |  | | | | | | | | | | | | | |
| 4 – Расхождение с документацией | | | | | | | | | | | | | | |  | | | | | | | | | | | | | | |  | | | | | | | | | | | | | |
| 5 – Взаимодействие с аппаратурой | | | | | | | | | | | | | | |  | | | | | | | | | | | | | | |  | | | | | | | | | | | | | |
| 6 – Вопрос | | | | | | | | | | | | | | |  | | | | | | | | | | | | | | |
| ПРОБЛЕМА | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Можете ли вы воспроизвести проблемную ситуацию? (Д/Н) | | | | | | | | | | | | | | | | | | | | | | | | | **Да** | | | | | | | | | | | | | | | | | |
| Подробное описание проблемы и способа ее воспроизведения: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | **Запустить** | | | | | | | | | | | | | |
| **программу, ввести в поле ввода значения радиуса R= -2. Программа не выдает** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **сообщение об ошибке, окно с изображением фигуры пусто, потому что после** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| **ввода отрицательного значения в поле радиуса R = 0.** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Предлагаемое исправление (необязательно): | | | | | | | | | | | | | | | | | | **Сделать уведомление пользователя** | | | | | | | | | | | | | | | | | | | | | | | | |
| **с сообщением об ошибке.** | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Отчет представлен сотрудником | | | | | | | | | | | | |  | | | | | | | | Дата | | | | | 05 | | | | | | | / | | 04 | | | / | 2022 | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Заполняется разработчиком | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Функциональная область | | | | | | | | |  | | | | | | | | | | Ответственный | | | | | | | | | | | | |  | | | | | | | | | | |
| Комментарии | | | | | |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Состояние (1-2) | | | | | | | 1 | | | | | | | | | Приоритет (1-5) | | | | | | | | | | | | | | | 1 | | | | | | | | | | | |
| 1 - Открыто | | | | | 2 - Закрыто | | | | | |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Резолюция (1-9) | | | | | | | 1 | | | | | | | Исправленная версия | | | | | | | | | | | | | |  | | | | | | | | | | | | | | |
| 1 - Рассматривается | | | | | | | | | | | | 6- Не может быть исправлено | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 - Исправлено | | | | | | | | | | | | 7 - Отозвано составителем | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3- Не воспроизводится | | | | | | | | | | | | 8 - Нужна дополнительная информация | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 - Отложено | | | | | | | | | | | | 9 - Не согласен с предложением | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 - Соответствует проекту | | | | | | | | | | | |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Рассмотрено | | |  | | | | | | | | | | | | | | Дата | | | | |  | | / | | |  | | | | | | | / | |  | | | |
| Проконтролировано | | | | | | | |  | | | | | | | | | Дата | | | | |  | | / | | |  | | | | | | | / | |  | | | |
| Считать отложенным(Д/Н) | | | | | | | | | |  | | | | | | |  | | | | | |  | | | | | | | | | | | | | | | | | | | |

# 8 Выводы о качестве тестируемого программного продукта и наиболее удачных тестах

Для составления тестов были использованы следующие методы тестирования:

1. Структурное
2. Белый ящик (покрытие операторов, покрытие решений, покрытие решений / условий)
3. Черный ящик (эквивалентные классы, граничные условия, предположение об ошибке)
4. Тестирование пользовательского интерфейса

Составлено и протестировано 34 теста, из них 7 с неудовлетворительной реакцией программы. Было выявлено что программа не обрабатывает входные данные, поэтому не может выдавать сообщения об ошибке ввода. Так же не совсем корректно работают функции изменения размера через растягивания окна и каретку.

Трудно выделить наиболее удачные тесты, потому что каждые из методов и стратегий тестирования направлены на поиск ошибок разного рода.

На основе результатов тестирования можно сделать вывод что качество тестируемой программы далеко от идеального, возможно это связано с неполнотой или неточностью постановки задачи.

**9 Таблицы спецификаций**

**Поля и методы класса Form1:**

Private int vertices = 5 –

private double pix = 37.936

int \_oldWidth, \_oldHeight

int \_oldTrackBar

float proportion = 1f

Функция DrawPolygon\_Ellipse

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| Входные величины | | |
| radius | Радиус | double |

Функция CalculateVertices

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| Входные величины | | |
| radius | Радиус | double |
| startingAngle | Угол начала поворота | int |
| center | Центр фигуры | Point |
| Выходные величины | | |
| points |  | Point[] |

Функция AreaCalculation

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| Входные величины | | |
| radius | Радиус | double |
| Выходные величины | | |
|  |  | double |

Функция AreaCalculationEllipse

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| Входные величины | | |
| radius | Радиус | double |
| Выходные величины | | |
|  |  | double |

Функция AreaCalculationPolygon

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| Входные величины | | |
| radius | Радиус | double |
| Выходные величины | | |
|  |  | double |

Функция trackBar1\_ValueChanged

Функция numericUpDown1\_ValueChanged

Функция Form1\_Resize

Функция Form1\_ResizeBegin

Функция Form1\_ResizeEnd

Функция radiusChanged

|  |  |  |
| --- | --- | --- |
| Имя | Назначение | Тип/Диапазон |
| Входные величины | | |
| a | Переменная для значения бегунка сейчас | int |
| b | Переменная для старого значения бегунка | int |

# Литература

* 1. Эрик Фримен, Элизабет Робсон Head First Паттерны проектирования обновленное юбилейное издание, 2020. 656 с. ISBN: 978-5-4461-1034-6 Серия: Head First O’Reilly (дата обращения: 17.02.2022)
  2. Репозиторий с исходным кодом проекта // GitHub URL: https://github.com/KomogortsevaYulia/TRPK/tree/main/Testing (дата обращения: 12.03.2022)
  3. Модульное тестирование - стратегии проектирования тестовых наборов белого ящика. [Электронный ресурс] URL: https://pandia.ru/text/78/481/44389.php (дата обращения 26.03.2022)