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](#_Toc99552624)

[1 Постановка задачи 4](#_Toc99552625)

[**2 Описание задачи и алгоритма ее решения** 6](#_Toc99552626)

[**3 Структурное тестирование** 7](#_Toc99552627)

[**4 Белый ящик** 9](#_Toc99552628)

[**5 Черный ящик** 12](#_Toc99552629)

[**6 Тестирование пользовательского интерфейса** 15](#_Toc99552630)

[**7 Результаты тестирования, представленные в виде таблиц** 16](#_Toc99552631)

[8 Протокол по результатам тестирования 18](#_Toc99552632)

[9 Выводы о качестве тестируемого программного продукта и наиболее удачных тестах 19](#_Toc99552633)

[**10 Таблицы спецификаций** 20](#_Toc99552634)

[Литература 23](#_Toc99552635)

# 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;

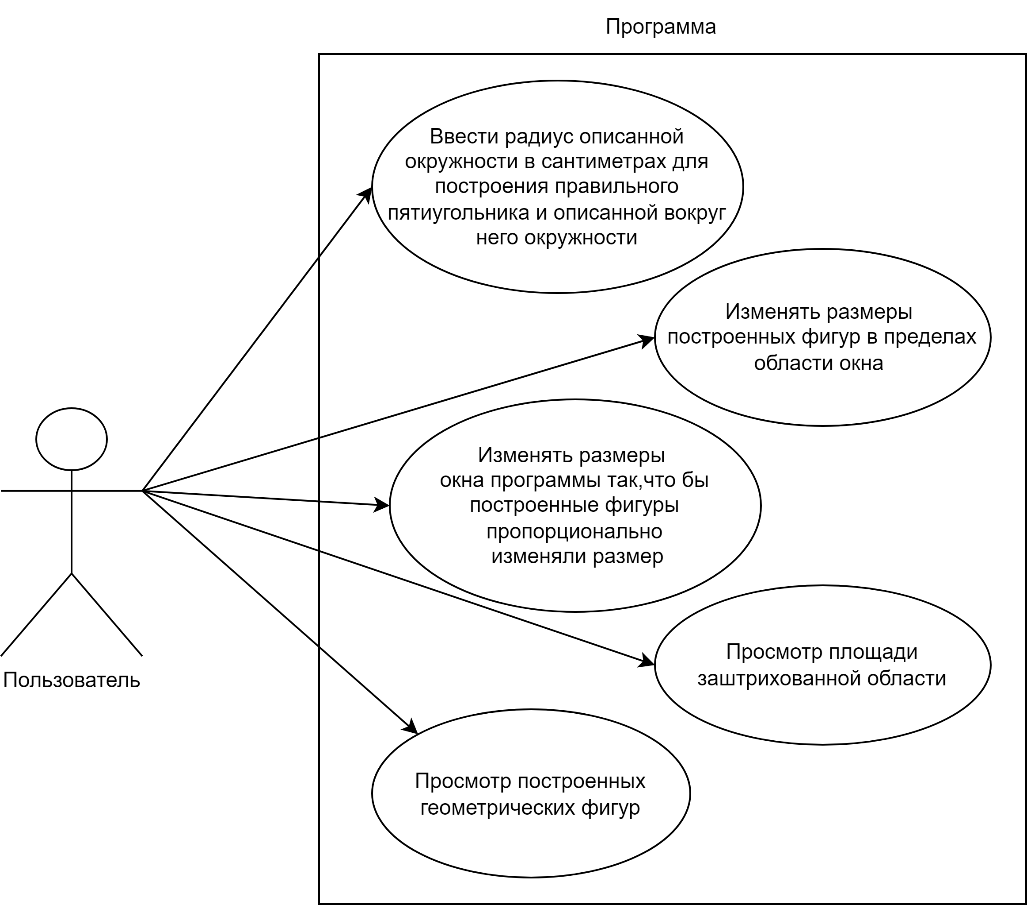


Рисунок 2 – Use case

**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 фигуры на основе заданного радиуса описанной окружности. Радиус задается в сантиметрах. Радиус — это положительное число. Верхняя граница радиуса должна быть не больше 25 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. |
| 2.6 |  |  |  |
| 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 | | |
| 4.1 |  |  |  |
| 4.2 |  |  |  |
| 4.3 |  |  |  |

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.

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.

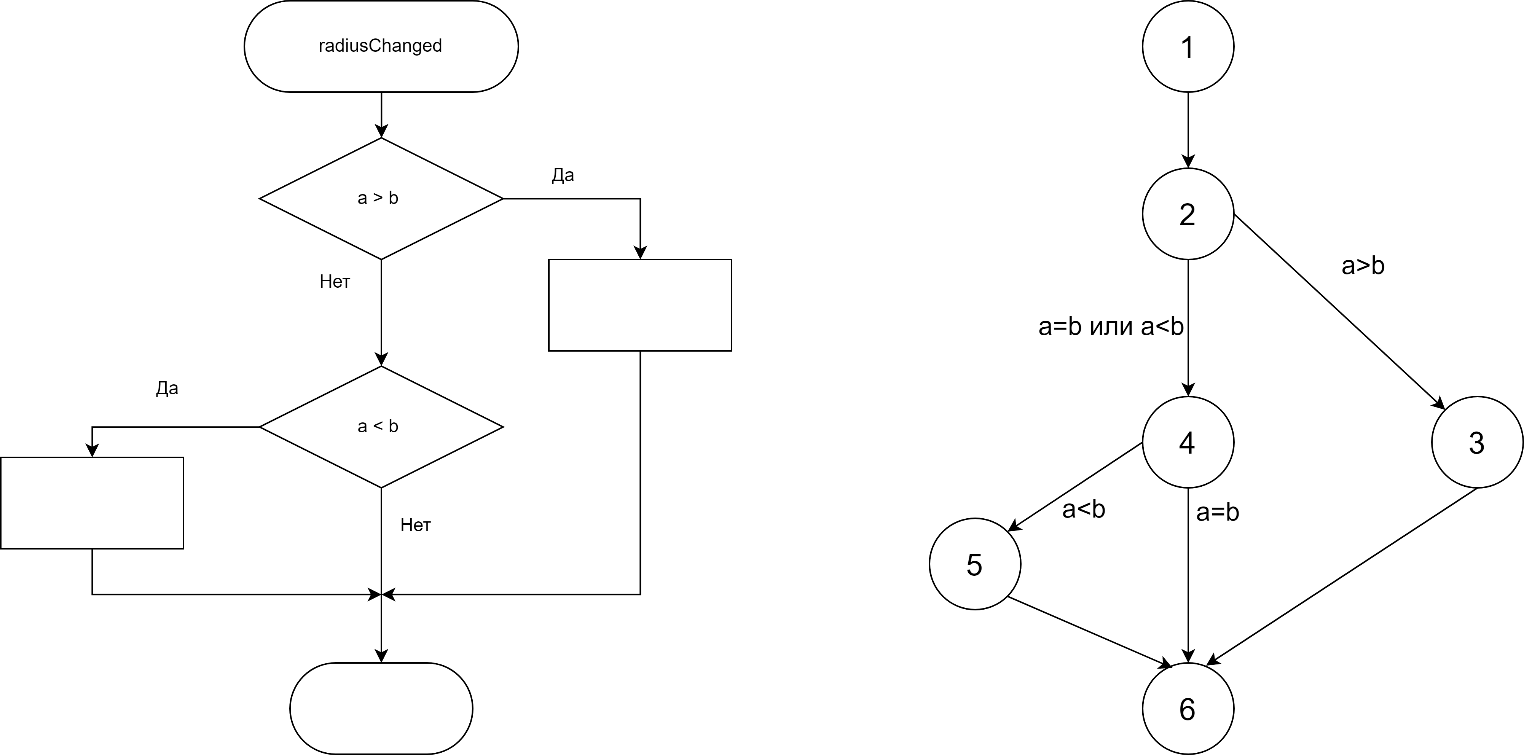


Рисунок 3 – Схема алгоритма метода и ее граф передач управления

Что бы убедиться в необходимости операторов, составим тесты по методу покрытия операторов. Покрытие операторов подразумевает выполнение каждого оператора программы, по крайней мере, один раз. Такому выполнению соответствует путь 1236 и путь 12456.

Что бы убедиться в необходимости решений, составим тесты по методу покрытия решений. Покрытие решений подразумевают, что каждое решение на этих тестах принимает значение «истина» или «ложь», по крайней мере, один раз. При выполнении программы по пути 1246 или 12456 решение на вершине 2 и на вершине 4 примет значение «истина» или «ложь» по одному разу.

В связи с тем, что в представленном коде программы решения имеют только одно условие, то решение и условие совпадает, значит нет необходимости использовать метод покрытия условий.

В связи с тем, что в представленном коде программы решения имеют только одно условие, то решение и условие совпадает, значит по методу покрытия решений/условий нам необходимо составить такие пути выполнения программы, где все результаты каждого решения выполнились, по крайней мере один раз: путь 1236, путь 1246, путь 12456.

В связи с тем, что в представленном коде программы решения имеют только одно условие, то решение и условие совпадает, значит нет необходимости использовать метод комбинаторного покрытия условий.

Таблица тестов стратегии «белого ящика».

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Назначение теста | Значения исходных данных | Ожидаемый результат |
| 1 | Покрытие операторов,  Покрытие решений/условий | a=5  b=4 | Путь 1236 |
| 2 | Покрытие операторов, Покрытие решений,  Покрытие решений/условий | a=5  b=6 | Путь 12456 |
| 3 | Покрытие решений/условий | a=6  b=6 | 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

|  |  |  |  |
| --- | --- | --- | --- |
| Номер класса | Правильные  классы эквивалентности | Номер класса | Неправильные классы эквивалентности |
| 1 | R – число | 4 | R – не число |
| 2 | R – вещественное число | 5 | R < 0 |
| 3 | 0≤R ≤ 25 | 6 | R > 25 |
|  |  |  |  |

Одним тестом мы должны покрыть все правильные классы эквивалентности: при R=4.25 покрываются классы 1,2,3,4.

Но на каждый неправильный класс эквивалентности мы создадим отдельный тест. Для класса 5 подойдет R= абвг, для класса 6 подойдет R= -0,001, для класса 7 подойдет R= 25,001.

**По методу граничных условий:**

Границы для входной величины – радиуса: 0 ≤ R ≤ 25. Внутри рассматривать не будем значения, зато можно рассмотреть значения, которые выходят за границы на 0. 001.Тем самым необходимо проверить значения R= -0,001 и R=25,001.Однако такие значения входных данных уже используются в тестах из эквивалентных разбиений, поэтому мы не будем делать дублирование а соединим в один тест.

**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 методов была создана таблица тестов стратегии «Черный ящик».

Таблица тестов стратегии «черного ящика».

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер теста | Назначение теста | Значения исходных данных | Ожидаемый результат | Реакция программы | |
| 4 | Проверка на правильные классы эквивалентности 1,2,3,4 | R=4,25 | S= 13,799 | Построенная фигура, выделенная область, выведенная площадь | |
| 5 | Проверка на неправильный класс эквивалентности 5 | R= абвг | Программа выдает сообщение об ошибке 1  Программа выдает сообщение об ошибке 2 | | Программа выдает сообщение об ошибке 3 |
| 6 | Проверка на неправильный класс эквивалентности 6, проверяем граничное значение | R= -0,001 |  | |  |
| 7 | Проверка на неправильный класс эквивалентности 7, проверяем граничное значение | R=25,001 |  | |  |

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.

**6 Тестирование пользовательского интерфейса**

Таблица тестов пользовательского интерфейса

|  |  |  |
| --- | --- | --- |
| Номер теста | Действие | Реакция системы |
| 8 | Щелкните на пиктограмме System и выберите пункт меню 'System Management Applet'. | Появится окно ввода логина и пароля |
| 9 | Введите в появившееся окно ввода имя пользователя 'guest1' и пароль 'guest'. Затем нажмите кнопку 'Login'. | Появится окно 'System Management Applet'. В верхнем правом углу должно быть выведено имя вошедшего пользователя guest1 Все опции в окне должны быть отключены (выведены серым цветом) |
| 10 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**7 Результаты тестирования, представленные в виде таблиц**

Таблица результатов тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Назначение теста |  |  | Результат выполнения теста |
| 4 | Проверка на правильные классы эквивалентности 1,2,3,4 |  |  | Верно  Рисунок 4 |
| 5 | Проверка на неправильный класс эквивалентности 5 |  |  |  |
| 6 | Проверка на неправильный класс эквивалентности 6, проверяем граничное значение |  |  |  |
| 7 | Проверка на неправильный класс эквивалентности 7, проверяем граничное значение |  |  |  |
| 8 | Проверка на правильные классы эквивалентности 1,2,3,4 |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

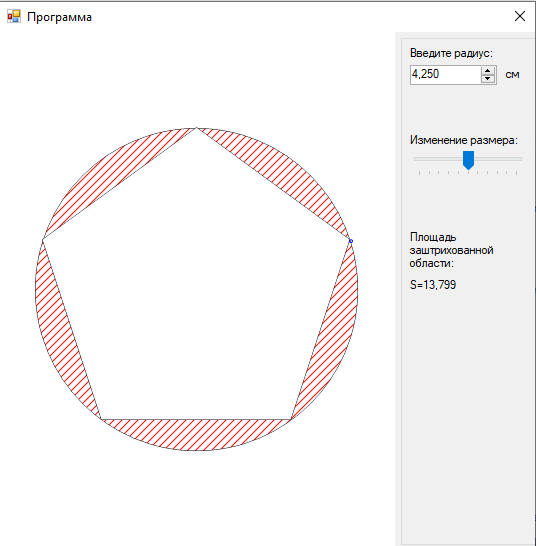


Рисунок 4 – Результат выполнения теста №4

# 8 Протокол по результатам тестирования

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Название компании | | | | | | | | | | | | Конфиденциально | | | | | | | | | Отчет о проблеме | | | | | | | | | | | | | | | | | | | | | |
| № |  | | |
| Программа | |  | | | | | | | | | | | | | Выпуск | | | | |  | | | | | | Версия | | | | | | | | | | |  | | | | |
| Тип проблемы (1-6) | | | | | | | | | | | | | | | Степень важности (1-3) | | | | | | | | | | | | | | | 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) | | | | | | | | | | | | | | | | | | | | | | | | |  | | | | | | | | | | | | | | | | | |
| Подробное описание проблемы и способа ее воспроизведения: | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Предлагаемое исправление (необязательно): | | | | | | | | | | | | | | | | | |  | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Отчет представлен сотрудником | | | | | | | | | | | | |  | | | | | | | | Дата | | | | |  | | | | | | | / | |  | | | / |  | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Заполняется разработчиком | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Функциональная область | | | | | | | | |  | | | | | | | | | | Ответственный | | | | | | | | | | | | |  | | | | | | | | | | |
| Комментарии | | | | | |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Состояние (1-2) | | | | | | |  | | | | | | | | | Приоритет (1-5) | | | | | | | | | | | | | | |  | | | | | | | | | | | |
| 1 - Открыто | | | | | 2 - Закрыто | | | | | |  | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Резолюция (1-9) | | | | | | |  | | | | | | | 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) | | | | | | | | | |  | | | | | | |  | | | | | |  | | | | | | | | | | | | | | | | | | | |

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

**10 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/Pattern (дата обращения: 12.02.2022)
  3. <https://pandia.ru/text/78/481/44389.php>