КОЛЛЕДЖ МНОГОУРОВНЕВОГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

МДК 01.02 Поддержка и тестирование программных модулей

Лабораторная работа №3,4

**Тестирование «Чёрным ящиком»**

|  |  |
| --- | --- |
| Выполнил(а) студент группы 32ИС-22 | Проверил(а) |
| Арутюнян С.К. | Калашникова О.А. |

Москва 2023 г.

**Форма отчета:**

**Цель работы.** Отработать навыки составления и тестирования программ как «черного ящика».

Особенность тестировании программ как «черный ящик» заключается в следующем:

**Известны:** функции программы.

**Исследуется:** работа каждой функции на всей области определения.

Как показано на рисунке, основное место приложения тестов «черного ящика» – интерфейс ПО.

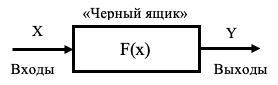


Рисунок 6.1 – Тестирование «черного ящика»

Эти тесты демонстрируют:

• как выполняются функции программ;

• как принимаются исходные данные;

• как вырабатываются результаты;

• как сохраняется целостность внешней информации.

При тестировании «черного ящика» рассматриваются системные характеристики программ, игнорируется их внутренняя логическая структура.

**Практическое задание:**

Построить таблицу значений функции *y*=*f(x)*, *x* изменяется от *xmin* до *xmax* с шагом *dx*. Проконтролировать правильность ввода *xmin*, *xmax*, *dx* и корректность вычисляемого выражения.

Примечание. В протоколе необходимо указать порядок выполнения операций в соответствии с их приоритетом.



**1. Перечень причин и следствий:**

Причины:

1. Неправильное задание порядка выполнения операций может привести к неверным результатам вычислений.

2. Некоторые значения, например, под знаком корня, могут быть отрицательными, что приведет к ошибке выполнения программы.

3. Некоторые значения, например, логарифм от нуля, могут быть неопределенными, что также приведет к ошибке выполнения программы.

Следствия:

1. Неверные результаты вычислений могут привести к неправильным выводам и решениям.

2. Ошибки выполнения программы могут привести к ее аварийному завершению или непредсказуемому поведению.

3. Неправильный порядок выполнения операций может привести к неэффективности программы и затратам на ее исправление.



Функция «не» устанавливает, что если значение c1 есть 1, то значение e1 есть 0. В противном случае значение есть 1.

*import math*

*a = float(input("Введите значение a: "))*

*b = float(input("Введите значение b: "))*

*xmin = float(input("Введите значение xmin: "))*

*xmax = float(input("Введите значение xmax: "))*

*dx = float(input("Введите значение dx: "))*

*x = xmin*

*while x <= xmax:*

*if 19\*x-3.5 < 0: # проверка на отрицательное значение под корнем*

*print("При x =", x, "значение корня меньше нуля")*

*else:*

*y = x\*\*2 - a\*\*3 - b\*\*2\*math.sqrt(19\*x-3.5) + math.log(x)*

*print("При x =", round(x, 2), "значение y =", round(y, 2)) # округление до сотых*

*x += dx*

Порядок выполнения операций в соответствии с их приоритетом:

1. Вычисление значения выражения под знаком корня (19\*x-3.5).

2. Вычисление квадратного корня из полученного значения.

3. Умножение полученного значения на b^2.

4. Вычисление значения выражения под знаком логарифма (x).

5. Вычисление натурального логарифма от полученного значения.

6. Вычитание из полученного значения a^3.

7. Вычисление значения x^2.

8. Сложение полученных значений в соответствии с порядком записи.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **Xmin** | **Xmax** | **Dx** | **При x = …значение y…** | |
| 1 | 2 | 3 | 4 | 5 | 3.0 | -20.16 |
| -2 | -3 | -4 | -5 | -6 | *Process finished with exit code 0*  Ошибку не выдаёт, но пример не подсчитан. | |
| 0 | -9 | 3 | 0.7 | 2 | *Process finished with exit code 0*  Ошибку не выдаёт, но пример не подсчитан. | |
| 78 | 67 | 88 | 100 | -120 | *чтение корня меньше нуля*  *При x = -152652035.0 значение корня меньше нуля*  Это крайний вывод решения  Выполняется наше условие программы. | |
| 0 | 0 | 0 | 0 | 0 | *При x = 0.0 значение корня меньше нуля*  Выполняется наше условие программы. | |
| 1 | 1 | 0 | 3 | 7 | *При x = 0.0 значение корня меньше нуля*  Выполняется наше условие программы. | |

Практическая часть

**Цель работы**: получить навыки тестирования методом «черного ящика», используя технику тестирования *по сценариям пользователя*.

**Задание 1.** Создать приложение для определения вида треугольника и вычисления его периметра и площади. При создании использовать шаблон Windows Forms.

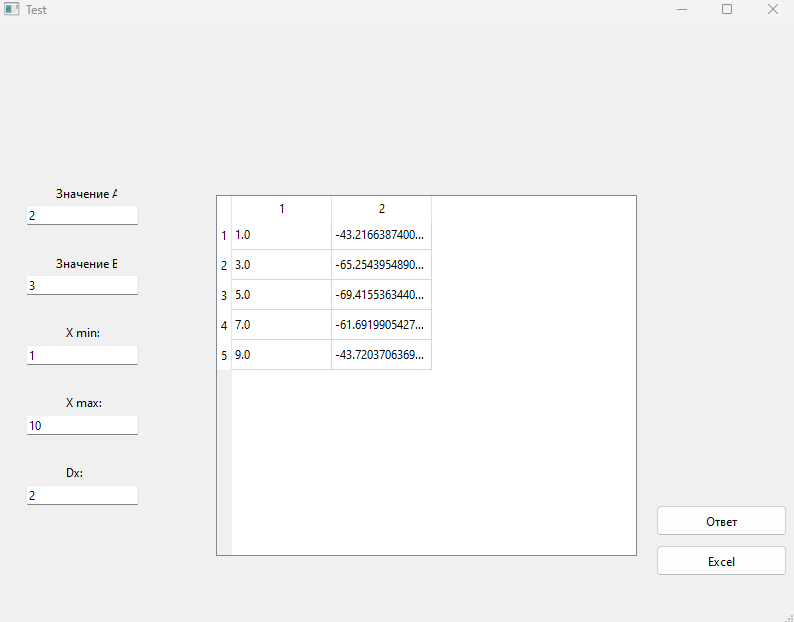
**Требования к приложению**.

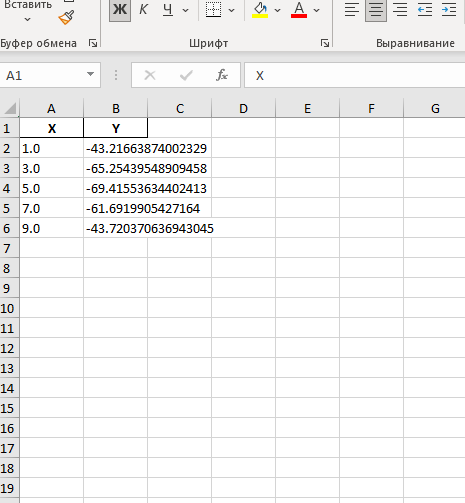
Приложение должно:

1. Проверять существование треугольника
2. Определять тип треугольника (разносторонний, равносторонний, равнобедренный, прямоугольный, остроугольный или тупоугольный) по длинам его сторон.
3. Вычислять площадь треугольника.
4. Вычислять периметр треугольника.

**Ограничения:**

1. При оставлении любого поля для ввода пустым должно обрабатываться исключение.
2. При заполнении любого поля для ввода некорректными данными должно обрабатываться исключение.
3. xmin не должен быть больше xmax.
4. Программа должна строить таблицу значений функции y=f(x) с шагом dx от xmin до xmax.
5. Программа должна выводить таблицу значений функции в Excel.





**Задание 2**. Составить спецификацию переменных.

|  |  |  |  |
| --- | --- | --- | --- |
| Имя переменной в программе | Назначение переменной в программе | Тип переменной | Диапазон типа |
| a | Сторона треугольника | int | От -2 147 483 648 до 2 147 483 647 |
| … | … | … | … |

**Задание 3.** Подготовить тестовый сценарий (набор тест-кейсов) для обнаружения ошибок в программе.

Результат оформить в следующем виде:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **А** | **В** | **С** | **Ожидаемый**  **результат** | **Реальный результат** |
| Значение | Значение | Значение | Что должно получится | Что получилось |
| … | … | … | … | … |

**Задание 4.** На основании проведенных тестов составьте отчет об ошибках (*баг-репот*) Пример оформления баг-репота:

|  |  |  |  |
| --- | --- | --- | --- |
| **№ теста** |  | **Описание ошибки** | **Рекомендации** |
| 1 | В ходе проведения первого теста было обнаружено, что при введении отрицательных значений площадь все равно вычисляется. | | 1) Добавить в программу проверку введенных значений на соответствие ограничения. 2) В случае, если пользователь введет некорректные данные, следует выводить сообщение с просьбой исправить введенные значения. |

*Приложение 1* **Чек-лист для проверки проведенного тестирования.** В графе «Отметка о прохождении» поставьте «+» напротив тестов, которые вы провели.

**Позитивные тесты**

|  |  |  |
| --- | --- | --- |
| № | Тест | Отметка о прохождении |
| 1. | 4 5 6 – разносторонний треугольник |  |
| 2. | 3 3 5 - равнобедренный треугольник |  |
| 3. | 3 5 3 - равнобедренный треугольник |  |
| 4. | 5 3 3 - равнобедренный треугольник |  |
| 5. | 5 5 5 - равносторонний треугольник |  |
| 6. | прямоугольный (3, 4, 5) |  |
| 7. | тупоугольный (2, 3, 4) |  |
| 8. | остроугольный (66, 67, 68) |  |

**Негативные тесты**

|  |  |  |  |
| --- | --- | --- | --- |
| № |  | Тест | Отметка о прохождении |
|  | 1. | 5 2 1 - две стороны слишком короткие (вариант 1) |  |
|  | 2. | 2 5 1 - две стороны слишком короткие (вариант 2) |  |
|  | 3. | 2 1 5 - две стороны слишком короткие (вариант 3) |  |
|  | 4. | 2 2 4 - две стороны слишком короткие - проверка строгих неравенств (вариант 1) |  |
|  | 5. | 2 4 2 - две стороны слишком короткие - проверка строгих неравенств (вариант 2) |  |
|  | 6. | 4 2 2 - две стороны слишком короткие - проверка строгих неравенств (вариант 3) |  |
|  | 7. | тесты на сторону, имеющую длину меньше нуля (возможно избыточные, т.к. код, проверяющий слишком короткие стороны должен отсечь такие данные) |  |
|  | 8. | тесты на проверку нулевых значений (возможно избыточные, т.к. код, проверяющий слишком короткие стороны по строгим неравенствам должен отсечь такие данные |  |
|  | 9. | Три теста на проверку ввода дробных значений |  |
|  | 10. | Три теста на проверку нечисловых значений |  |
|  | 11. | 2 2 - недостаточно данных - 2 числа |  |
|  | 12. | 2 - недостаточно данных - 1 число |  |
|  | 13. | "пустой ввод" |  |
|  | 14. | Проверяем слишком большое число. Большим числом называем такое, которое больше максимального значения INT. Например, число 4294967295. |  |

*Вывод(ы)*.

Код работает корректно и проходит базовые тесты, предусматривая следующую последовательность действий, построена таблица значений функции *y*=*f(x)*, *x* изменяющаяся от *xmin* до *xmax* с шагом *dx*. В протоколе указаны порядок выполнения операций в соответствии с их приоритетом.