

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт  информационных технологий** | **Кафедра информационных систем** |

|  |  |  |
| --- | --- | --- |
| **Основная образовательная программа 09.03.02 «Информационные системы и технологии»**  **Отчет по дисциплине «Управление рисками и надежностью информационных систем»** | | |
| **по лабораторной работе №1** | | |
| **Тема: «Тестирование»** | | |
|  | | |
| **Проверил**  **преподаватель** |  | **Петруша А.О.** |
|  | подпись |  |
| **Выполнил**  **студент группы ИДБ-21-06** |  | **Музафаров К.Р.** |
|  | подпись |  |

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc148698029)

[ГЛАВА 1. практическая часть. 4](#_Toc148698030)

[1.1. Тестирование программы для определения вида треугольника по трём сторонам 4](#_Toc148698031)

[ЗАКЛЮЧЕНИЕ 12](#_Toc148698032)

ВВЕДЕНИЕ

Тестирование является неотъемлемой частью разработки программного обеспечения и играет важную роль в обеспечении его качества. Тестирование представляет собой процесс исследования/испытания программного обеспечения (ПО) с целью определения соответствия между реальным и ожидаемым поведением программы, осуществляемый на основе набора тестов, выбранных определенным образом. Результаты тестов записываются для последующей оценки ПО на соответствие заданным требованиям и принятия решения о том, что с ним делать дальше. Тестирование имеет свои методики, подходы и инструменты, которые позволяют эффективно и систематически проверить функциональность, надежность, безопасность и производительность программы. Результаты тестирования помогают разработчикам исключить ошибки и гарантировать, что программное обеспечение будет работать корректно и надежно.

# Теоритическая часть.

## Составление краткого конспекта для устного ответа

В широком смысле тестирование программного обеспечения включает в себя планирование, проектирование, выполнение тестов, анализ полученных результатов.

В более узком тестирование программного обеспечения (software testing) это процесс исследования/испытания программного обеспечения (ПО) с целью определения соответствия между реальным и ожидаемым поведением программы, осуществляемый на основе набора тестов, выбранных определенным образом. Результаты тестов записываются для последующей оценки ПО на соответствие заданным требованиям и принятия решения о том, что с ним делать дальше.

QA (Quality Assurance) – список мер по улучшению качества продукта на всех этапах разработки, включая описание проекта, тестирование, релиз и иногда и пост-релизный этап. Quality Assurance обязательно включает в себя:

1. Процессы по улучшению качества продукта: предотвращение ошибок.
2. Контроль качества продукта – Quality Control.
3. Раннее тестирование(даже до того как продукт прошел ревью).
4. Управление изменениями, т. к. даже самое маленькое изменение может сломать систему, ее логику или какие-то компоненты.

 QC (Quality Control) – контроль качества продукта – это часть комплекса QA, которая отвечает за тестирование как таковое, начиная еще с этапа кодирования. QC-специалист проводит тестирование с целью представления информации о том, что происходит в разработке. Он ищет ошибки, анализирует результаты.

Quality Control включает в себя:

1. code review;
2. модульное тестирование;
3. непосредственное тестирование качества;
4. аудит (выполняется какой-то внешней организацией).

Работа тестировщика заключается в планировании и выполнении тестирования программного продукта, проверке его работоспособности и соответствию заявленным требованиям.

Этапы работы тестировщика:

1. Работа с требованиями перед непосредственным тестированием: Тестировщик анализирует требования для того, чтобы проверить их соответствие последующей реализации в программе.
2. Подготовка тестовых данных (test data).
3. Разработка тестовых сценариев (test scenarios) декомпозиция, mind-map.
4. Прогон тестов.
5. Нахождение дефектов (багов – bugs).
6. Оформление отчета о выполненной работе.
7. Активное взаимодействие с другими участниками процесса разработки ПО: с разработчиками, аналитиками, продукт-оунерами, это тоже является частью работы.
8. Написание, отладка, запуск и поддержка автотестов.

Тест-кейс – это пошаговая инструкция к функционалу приложения. Он дает представление о том, какое поведение ожидается от приложения, и любое отклонение от предполагаемого результата будет являться багом.  Для этого необходимо пошагово (1.. 2.. 3.. и т. д.) описать необходимые действия и их результаты (1 - результат на 1 шаг, 2 - результат на 2 шаг и т. д.)  для воспроизведения определенного пользовательского сценария (use case). В тест-кейсе всегда каждому действию (шагу) обязательно соответствует ожидаемый результат. Пример тест-кейса: «Успешная авторизация пользователя в приложении, используя почту».

test suit – это комплект тест-кейсов, относящихся к одному тестируемому модулю, функциональности, приоритету или одному типу тестирования.

Тестировщик пишет тест-кейсы для того, чтобы продукт можно было проверить без ознакомления со всей документацией.

Чек-лист (checklist)–это текстовый документ, содержащий перечень формализованных проверок(может заменять тест кейсы, может быть подготовкой к ним, проще поддерживать. Один пункт чек-листа описывает одно направление для проверки.).

Свойства чек-листа:

1. Логичность.
2. Последовательность и структурированность. Структурированность достигается при оформлении чек-листа в виде многоуровневого списка. Последовательность предполагает подачу информации в виде небольших групп идей, переход между которыми является понятным и очевидным (сначала позитивные, потом негативные).
3. Полнота(сухая выжимка, в ней все важное).

Чек-лист содержит:

1. список проверок с требуемой степенью детализации;
2. статус проверок (результат);
3. сборка, на которой проводилось тестирование;
4. тестовое окружение (если применимо);
5. кто проводил тестирование.

Любой тест-кейс имеет следующие для заполнения поля:

1. Уникальный номер
2. Заголовок (Title) - суть теста.
3. Приоритет (Priority)
4. Описание (Description) - что проверяет тест кейс.
5. Предусловия (Preconditions).
6. Требования (Requirements).
7. Шаги (Steps).
8. Ожидаемые результаты (Expected results).

Уровни тестирования:

1. Модульное (компонентное) тестирование – самая ранняя стадия тестирования, проверка отдельно взятого модуля (единицы исходного кода программы: отдельной функции, метода или объекта) на работоспособность.
2. Интеграционное тестирование – проводится после модульного, сосредоточено в основном на интерфейсах и потоке данных между модулями и представляет собой довольно широкую область проверок взаимодействия модулей в связке друг с другом. Приоритет проверки присваивается интегрирующим ссылкам, а не функциям блока, которые уже проверены разработчиком на модульном уровне.
3. Системное тестирование(функциональное, тестирование безопасности, тестирование юзабилити, тестирование производительности, нагрузочное тестирование и т. д. ).
4. Приемочное тестирование - соответствие требованиями заказчика.

Виды тестирования

Знание видов тестирования помогает оценить:

1. Функциональные характеристики качества продукта (полнота, правильность, целесообразность);
2. Нефункциональные характеристики качества продукта (надежность, производительность, безопасность, совместимость, удобство и т. д.);
3. Структуру и архитектуру продукта;
4. Последствия изменений.

Виды тестирования обычно классифицируются по следующим параметрам:

1. По позитивности
2. По целям тестирования функциональное и не функциональное
3. По способу исполнения тестов: ручное и автоматизированное
4. По времени применения тестов:

Дымовое тестирование

Санитарное тестирование

Регрессивное тестирование

Повторное/подтверждающее тестирование

1. По подходу выделяют:
   * Исследовательское тестирование
   * Свободное тестирование
2. По объекту тестирования:
   * Модульное
   * Интеграционное
   * Системное
3. По исполнению кода:
   * Динамическое(dynamic) - запуск и вод данных.
   * Статическое(static).

# Практическая часть

## Тестирование программы для определения вида треугольника по трём сторонам

На собеседованиях начинающим тестировщикам часто дают задание на проверку работы формы. Одно из наиболее популярных – тестирование программы, которая определяет тип треугольника по трем его сторонам. Каждая из сторон задается в отдельном текстовом поле.

В этом тренажере спрятаны несколько багов. Также он подсчитывает разные варианты кейсов, которые можно использовать для тестирования.

В ходе выполнения данного практического задания были найдены 4 из 4 багов и 12 из 12 кейсов.

Баг «Форма не валидирует поле C» представлен на рис. 1.1.

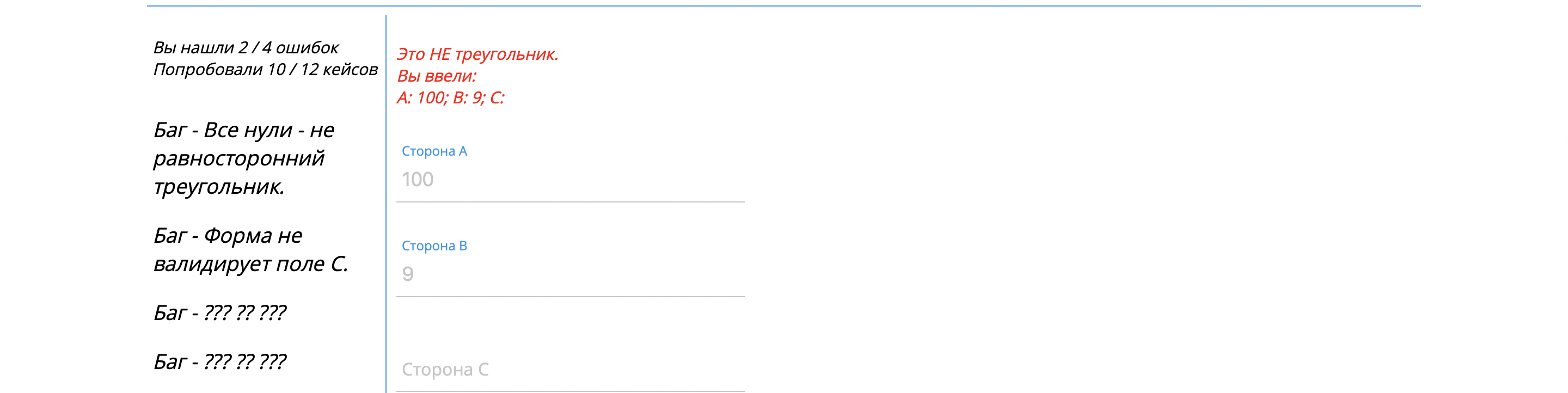


Рис. 1.1. Баг «Форма не валидирует поле C»

Баг «Форма неправильно работает с нецелыми числами» представлен на рис. 1.2.

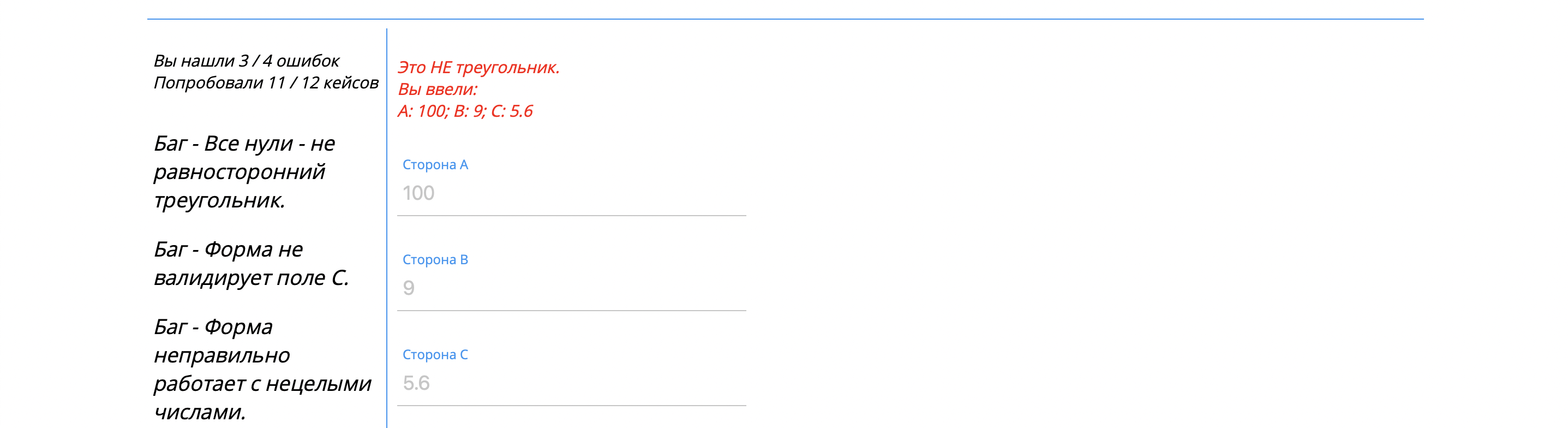


Рис. 1.2. Баг «Форма неправильно работает с нецелыми числами»

Баг «Все нули – не равносторонний треугольник» представлен на рис. 1.3.

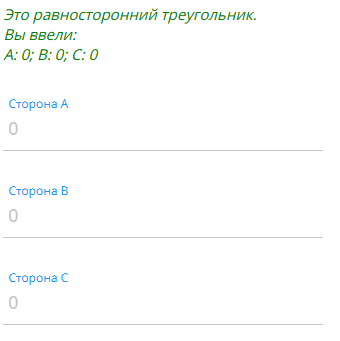


Рис. 1.3. Баг «Все нули – не равносторонний треугольник»

Баг «Вы нашли XSS» представлен на рис. 1.4.



Рис. 1.4. Баг «Вы нашли XSS»

Кейс «Равнобедренный треугольник» представлен на рис. 1.5.

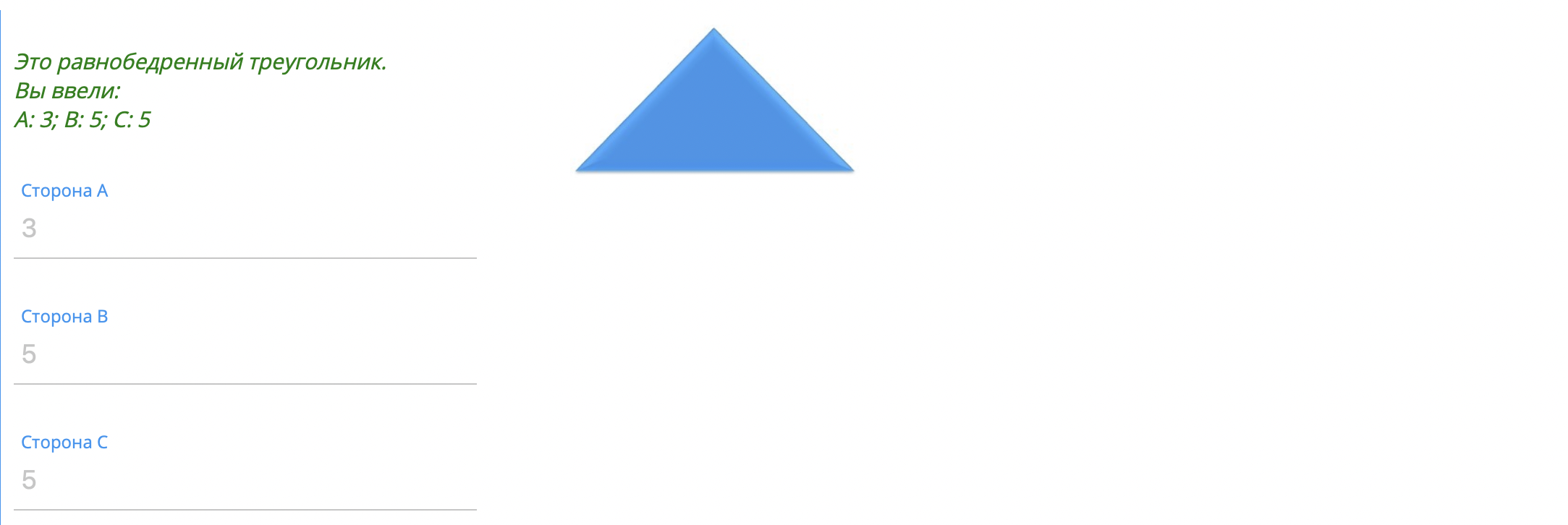


Рис. 1.5. Кейс «Равнобедренный треугольник»

Кейс «Не все поля заданы» представлен на рис. 1.6.

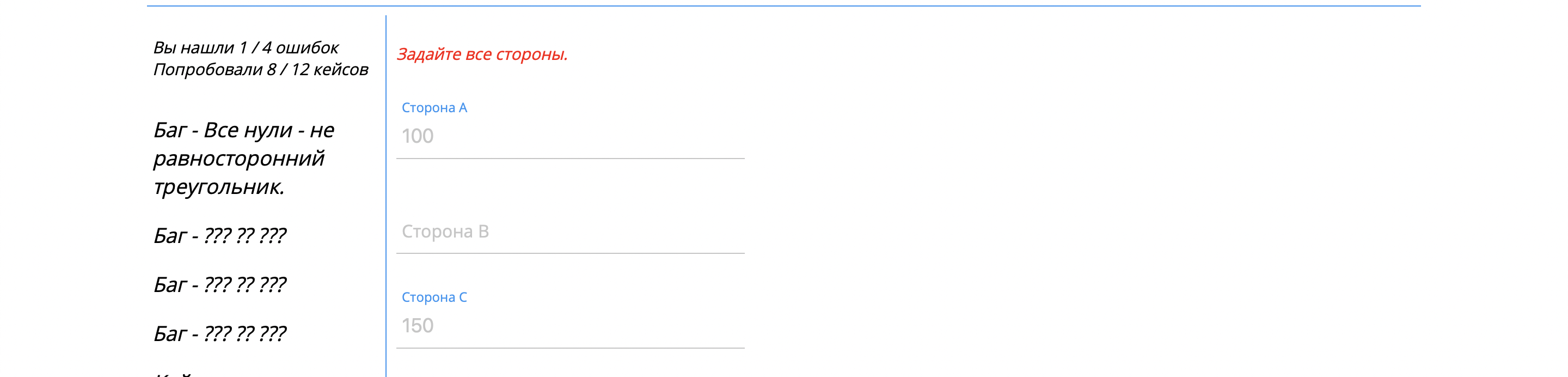


Рис. 1.6. Кейс «Не все поля заданы»

Кейс «Прямоугольный треугольник» представлен на рис. 1.7.

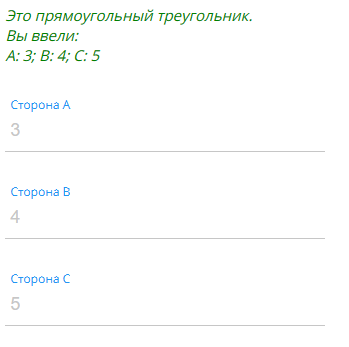


Рис. 1.7. Кейс «Прямоугольный треугольник»

Кейс «Равносторонний треугольник» представлен на рис. 1.8.

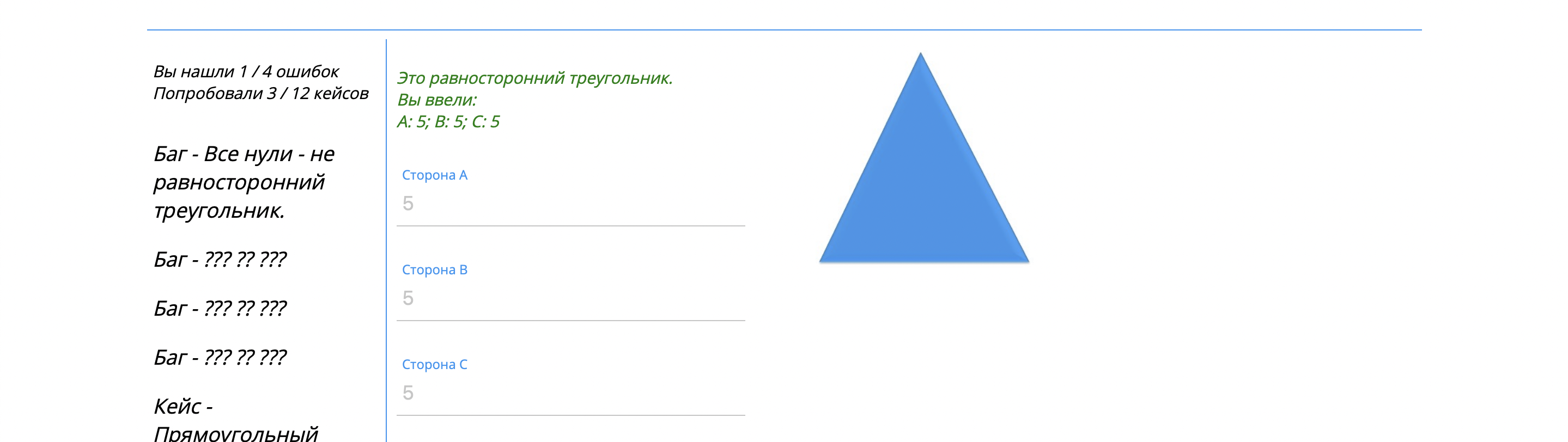


Рис. 1.8. Кейс «Равносторонний треугольник»

Кейс «Это не треугольник» представлен на рис. 1.9.

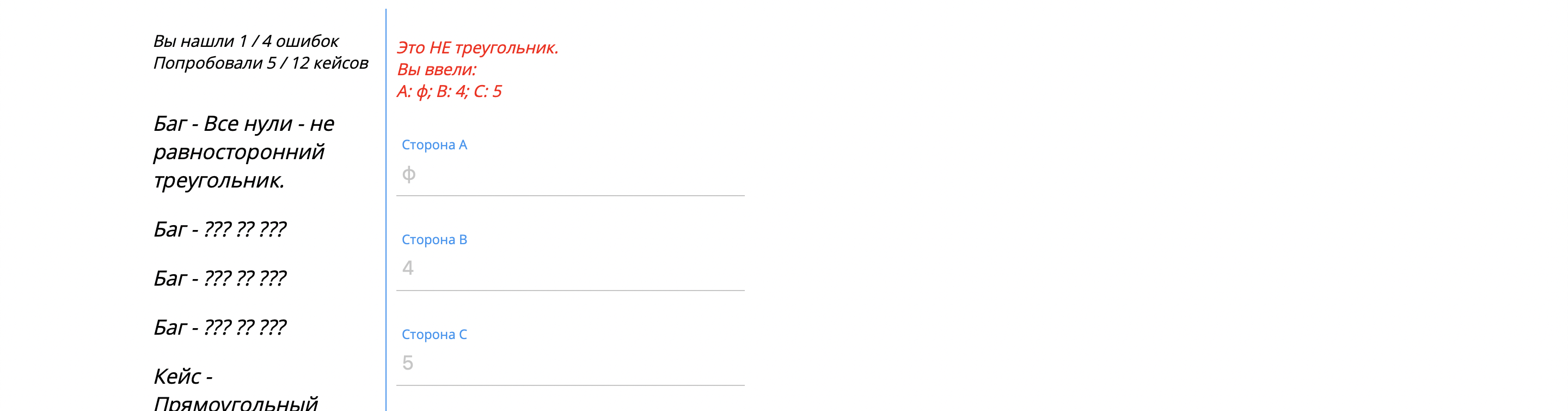


Рис. 1.9. Кейс «Это не треугольник»

Кейс «Не выполнились условия треугольника» представлен на рис. 1.10.

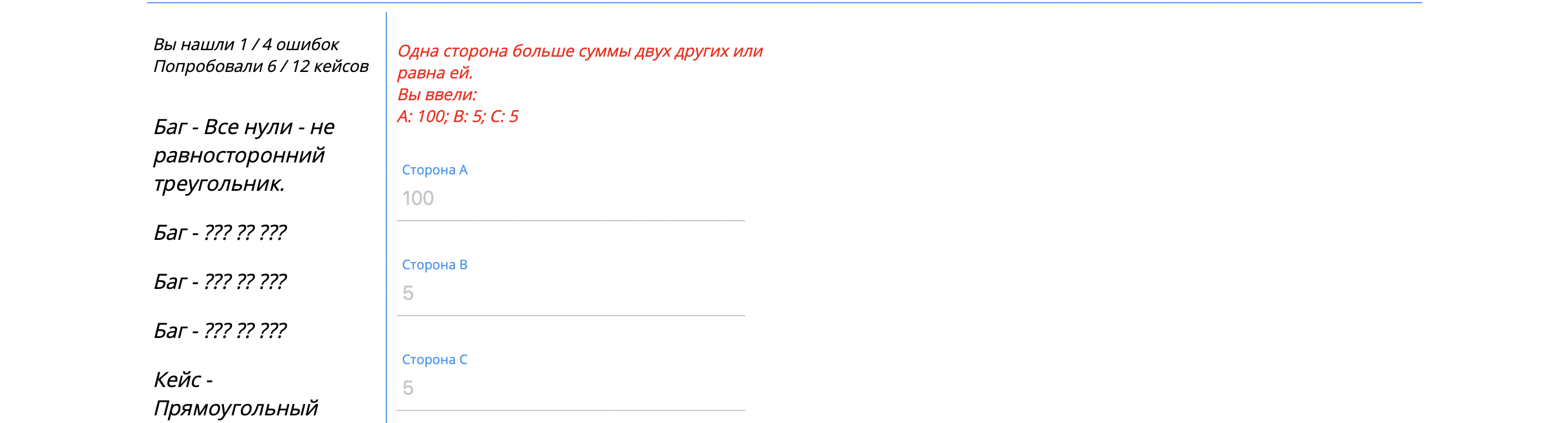


Рис. 1.10. Кейс «Не выполнились условия треугольника»

Кейс «Тупоугольный треугольник» представлен на рис. 1.11.

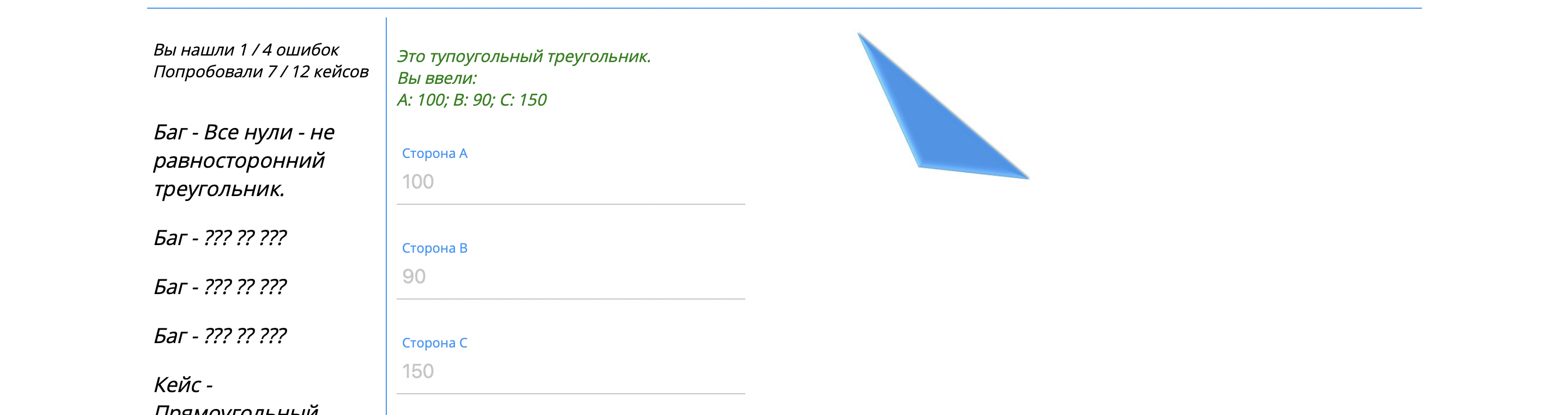


Рис. 1.11. Кейс «Тупоугольный треугольник»

Кейс «Все поля пустые» представлен на рис. 1.12.

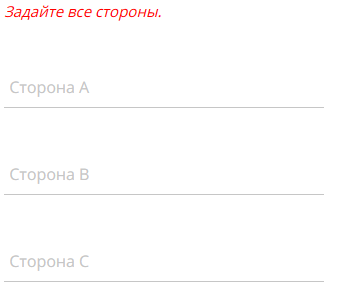


Рис. 1.12. Кейс «Все поля пустые»

Кейс «Попробовали большие числа» представлен на рис. 1.13.

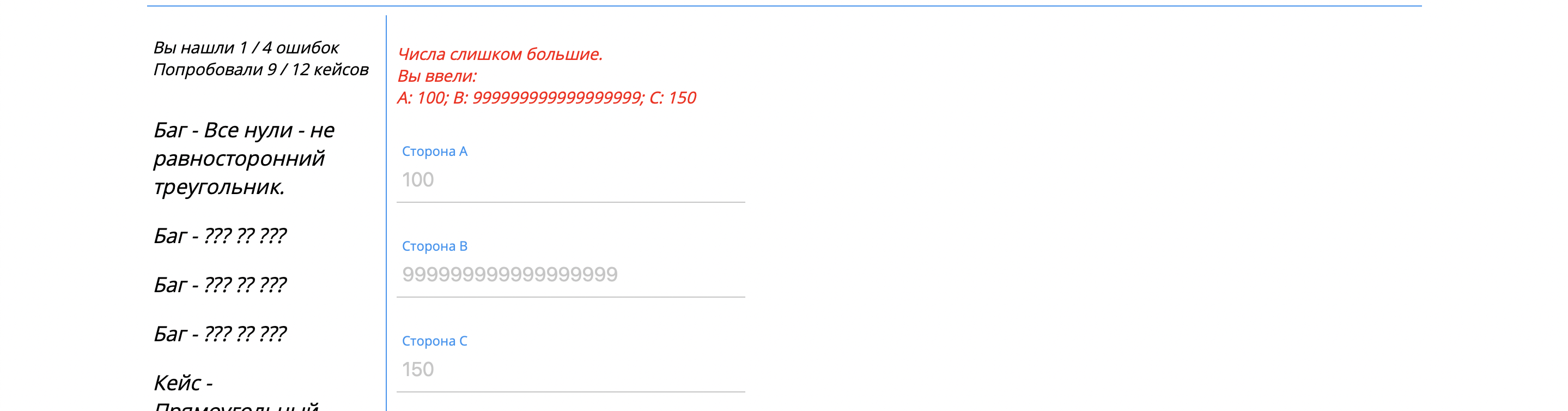


Рис. 1.13. Кейс «Попробовали большие числа»

Кейс «Попробовали XSS» представлен на рис. 1.15.

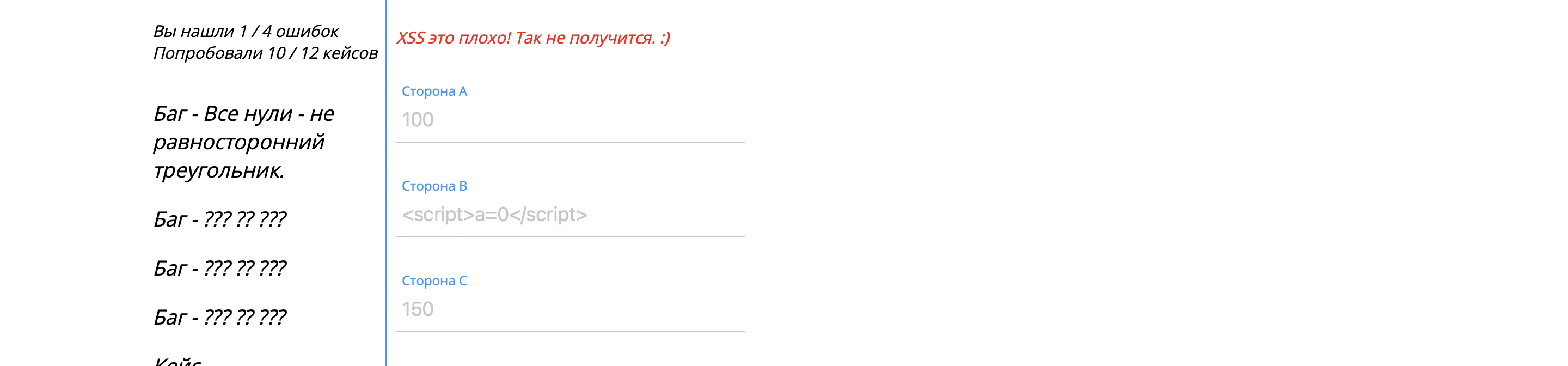


Рис. 1.15. Кейс «Попробовали XSS»

Кейс «Попробовали SQL-инъекцию» представлен на рис. 1.16.

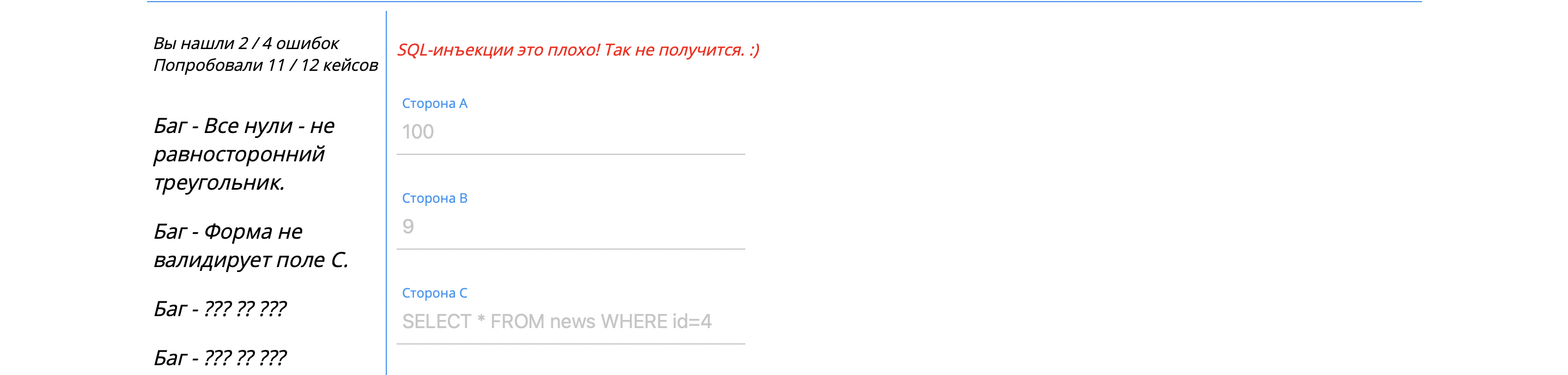


Рис. 1.15. Кейс «Попробовали SQL-инъекцию»

Кейс «Остроугольный треугольник» представлен на рис. 1.16.

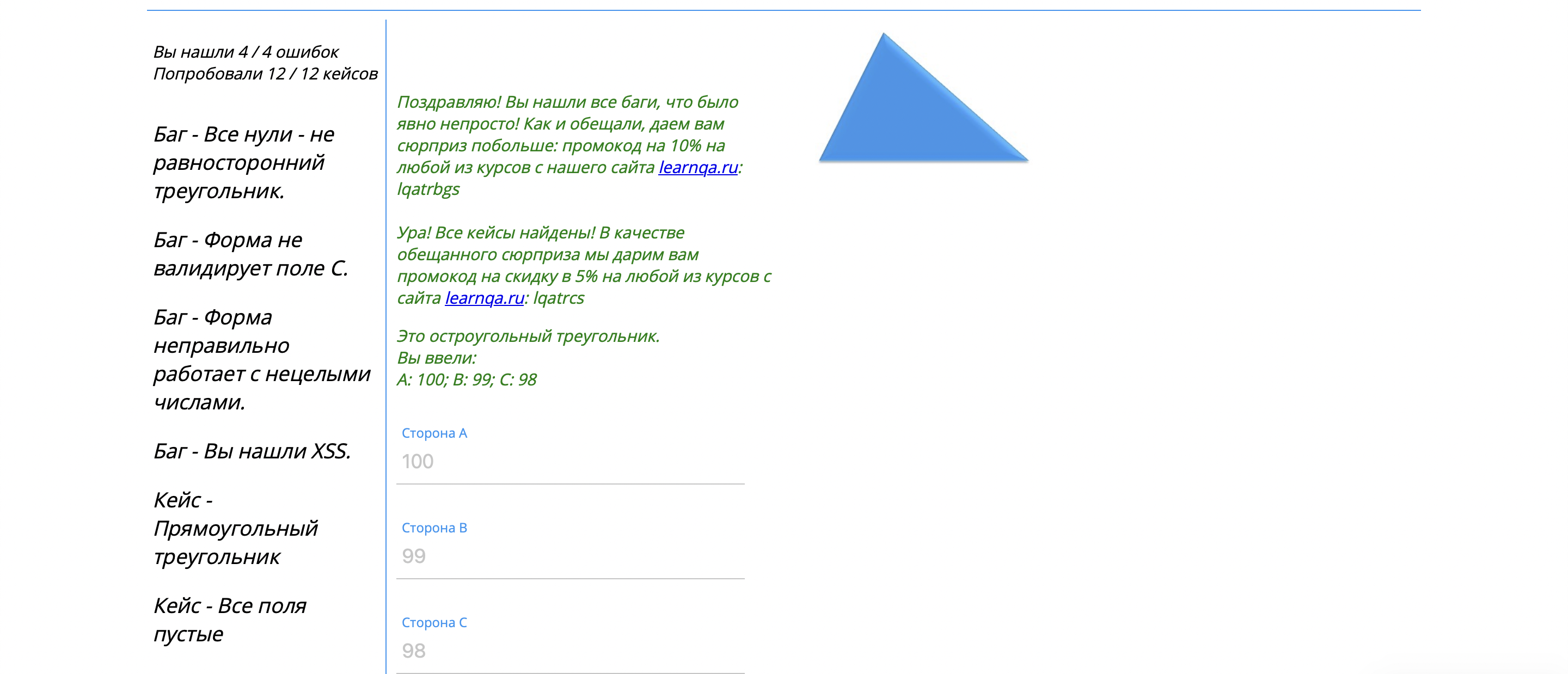


Рис. 1.16. Кейс «Остроугольный треугольник»

ЗАКЛЮЧЕНИЕ

В данной лабораторной работе были изучены теоретические аспекты тестирования, также было проведено ручное тестирование программы, которая определяет тип треугольника по трем его сторонам.