

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Факультет безопасности информационных технологий

Дисциплина:

«Технологии и методы программирования»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

«Тестирование программного обеспечения»

Выполнил:

Чу Ван Доан – Студент группы N3347



(подпись)

Проверил:

Ищенко Алексей Петрович

(отметка о выполнении)

(подпись)

Санкт-Петербург

2024 г.

СОДЕРЖАНИЕ

1. Задание.....	3
2. Ход работы.....	3
2.1. Тестирование "черного ящика" (Black-box Testing).....	3
2.2. Тестирование "белого ящика" (White-box Testing).....	3
2.3. Тестирование на основе требований (Requirement-based Testing).....	4
2.4. Сценарии тестирования базовой функциональности.....	4
2.4.1. Тестирование арифметических операций.....	4
2.4.2. Тестирование сложных операций.....	6
2.4.3. Тестирование некорректного ввода.....	7
2.5. Сценарии тестирования пользовательского интерфейса (UI Testing).....	7
2.6. Сценарии тестирования производительности (Performance Testing).....	8
2.7. Сценарии тестирования совместимости (Compatibility Testing).....	9
2.8. Сценарии тестирования граничных значений (Boundary Testing).....	11
2.9. Сценарии тестирования специальных функций.....	12
Заключение.....	13

1. Задание

Цель работы: Научиться применять методы тестирования программного обеспечения, разрабатывать тестовые сценарии и проводить тестирование на примере простого приложения.

Задачи:

1. Ознакомиться с основными методами тестирования (черный ящик, белый ящик, тестирование на основе требований).
2. Разработать тестовые сценарии для заданного приложения.
3. Провести функциональное тестирование приложения.
4. Зафиксировать результаты тестирования и выявленные дефекты.
5. Подготовить отчет о проведенной лабораторной работе.

2. Ход работы

- Я выбираю сайт: <https://www.desmos.com/scientific> для выполнения тестирования.

2.1. Тестирование "черного ящика" (Black-box Testing)

- Описание: Метод тестирования, при котором проверяется функциональность системы без учета внутренней структуры или кода. Тестировщик рассматривает систему как "черный ящик" и работает только с входными и выходными данными.
- Особенности:
 - + Используется для проверки соответствия системы заданным требованиям.
 - + Не требует знаний о внутреннем устройстве приложения.
 - + Фокусируется на том, что система должна делать, а не на том, как она это делает.
- Примеры:
 - + Проверка правильности вычислений на калькуляторе.
 - + Проверка интерфейса пользователя: работа кнопок, полей ввода и отображение данных.

2.2. Тестирование "белого ящика" (White-box Testing)

- **Описание:** Метод тестирования, при котором проверяется внутренняя структура, логика и код приложения. Тестировщик имеет доступ к исходному коду и тестирует систему на уровне кода

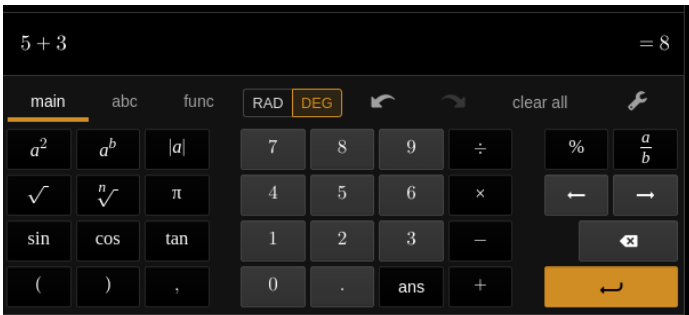
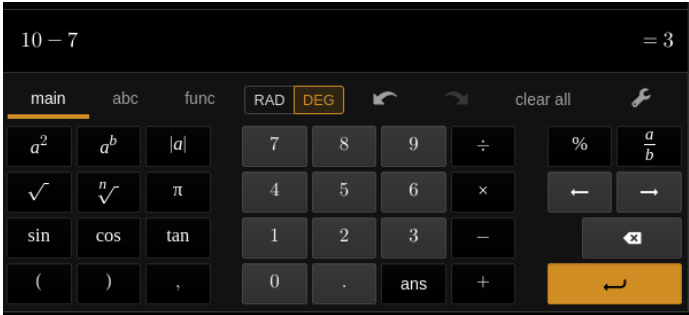
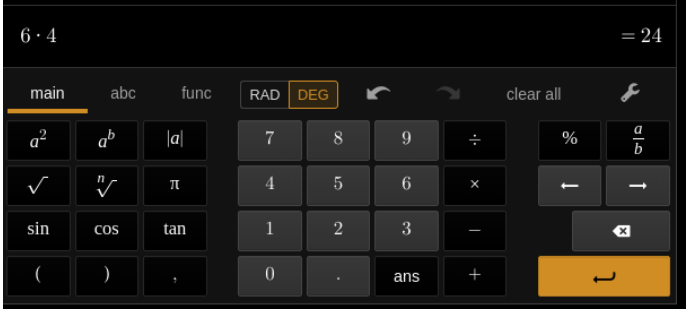
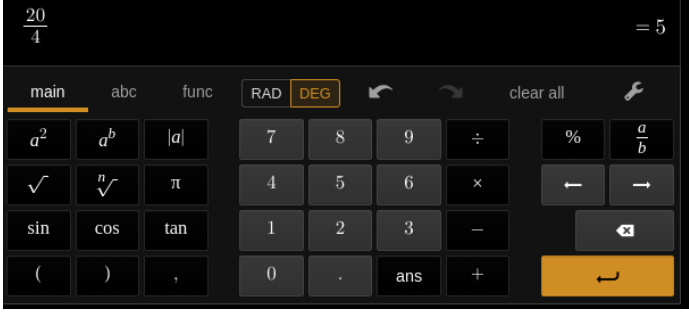
- Особенности:
 - + Требуется знание о внутренней реализации приложения.
 - + Направлено на проверку логики, структуры и возможных ошибок в коде.
 - + Используется для написания модульных и интеграционных тестов.
- Примеры:
 - + Проверка работы алгоритмов: вычисление сложных математических операций.
 - + Тестирование условий и циклов в коде.
 - + Покрытие кода тестами (Coverage Testing).

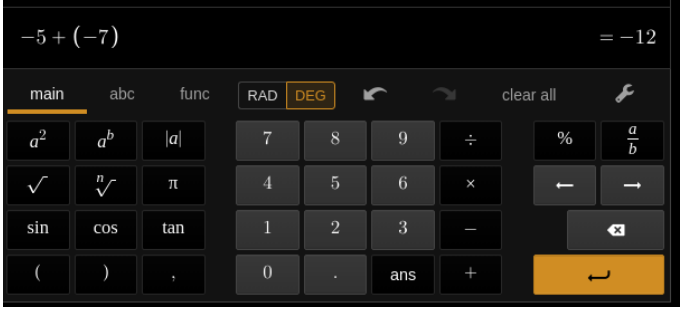
2.3. Тестирование на основе требований (Requirement-based Testing)

- **Описание:** Метод тестирования, который основывается на требованиях и спецификациях, предоставленных заказчиком или разработчиком. Его цель — убедиться, что приложение соответствует функциональным и нефункциональным требованиям.
- Особенности:
 - + Помогает определить, насколько приложение соответствует ожиданиям пользователя.
 - + Используется как в "черном ящике", так и в "белом ящике".
 - + Может включать тестирование производительности, безопасности и функциональности.
- Примеры:
 - + Если требование гласит, что калькулятор должен правильно выполнять операции сложения, вы проверяете это.
 - + Если в требованиях указано, что пользователь должен видеть результат за 2 секунды, тестируется время отклика системы.

2.4. Сценарии тестирования базовой функциональности

2.4.1. Тестирование арифметических операций

№	Название теста	Входные данные	Ожидаемый результат	Результат
1	Сложение целых чисел	5+3	Результат 8	
2	Вычитание целых чисел	10-7	Результат 3	
3	Умножение целых чисел	6×4	Результат 24	
4	Деление целых чисел	20÷4	Результат 5	

5	Использование десятичных чисел	$3.5+2.5$	Результат 6	
6	Использование отрицательных чисел	$-5+(-7)$	Результат -12	

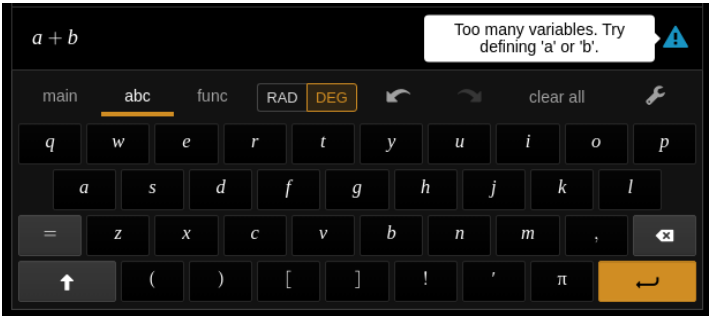
-> Программа выдает правильные результаты при тестировании арифметических операций.

2.4.2. Тестирование сложных операций

№	Название теста	Входные данные	Ожидаемый результат	Результат
1	Последовательность операций	$2+3 \times 4$	Результат 14	
2	Приоритет скобок	$(2+3) \times 4$	Результат 20	


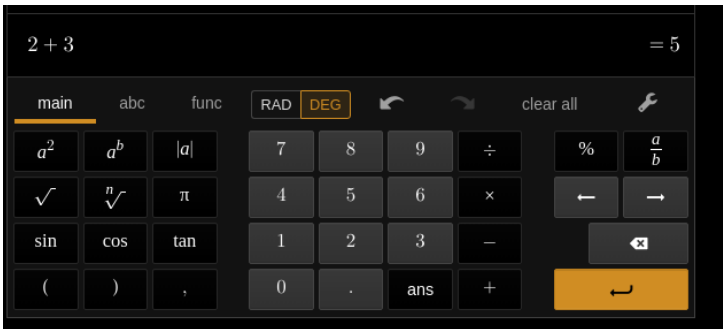
-> Программа выдает правильные результаты при тестировании сложных операций.

2.4.3. Тестирование некорректного ввода

№	Название теста	Входные данные	Ожидаемый результат	Результат
1	Ввод нечисловых символов	$a+b$	Отображается сообщение об ошибке	
2	Деление на 0	$5 \div 0$	Отображается сообщение "Нельзя делить на ноль"	

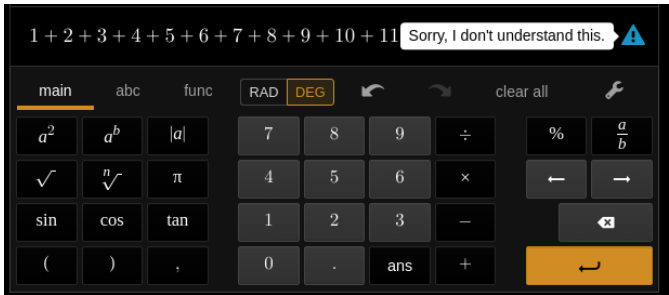
-> Программа выдает правильные результаты при тестировании некорректного ввода.

2.5. Сценарии тестирования пользовательского интерфейса (UI Testing)

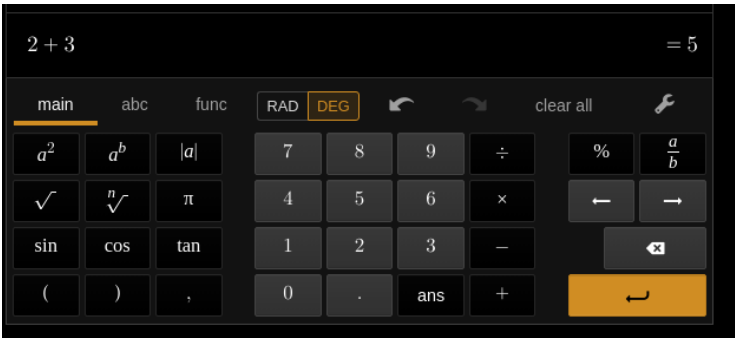
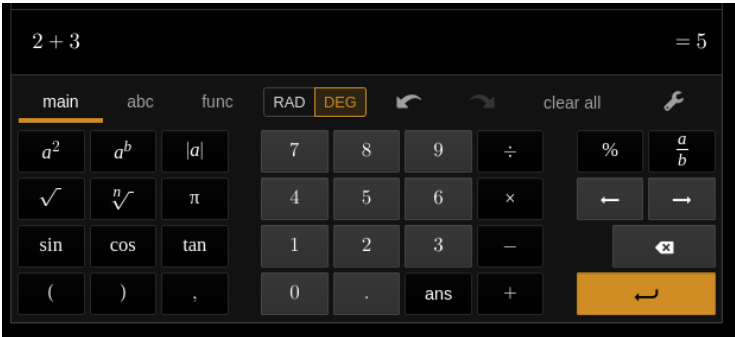
№	Название теста	Входные данные	Ожидаемый результат	Результат
1	Проверка главного интерфейса	-	Все кнопки и поля работают корректно	
2	Отображение результата	2+3	Результат отображается четко и правильно	

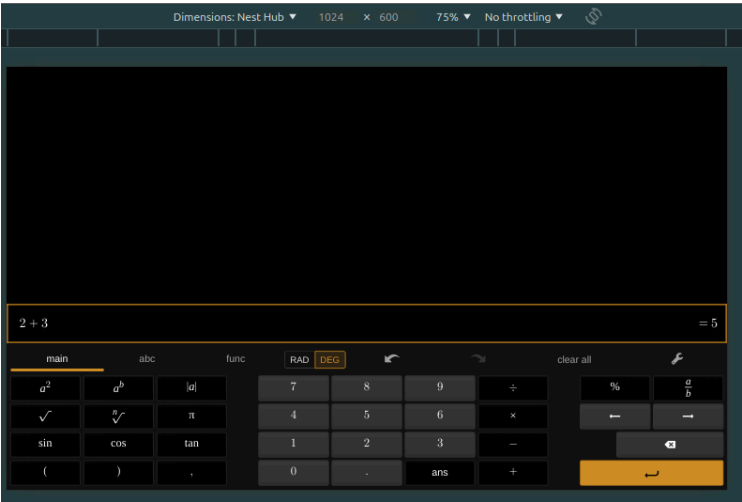
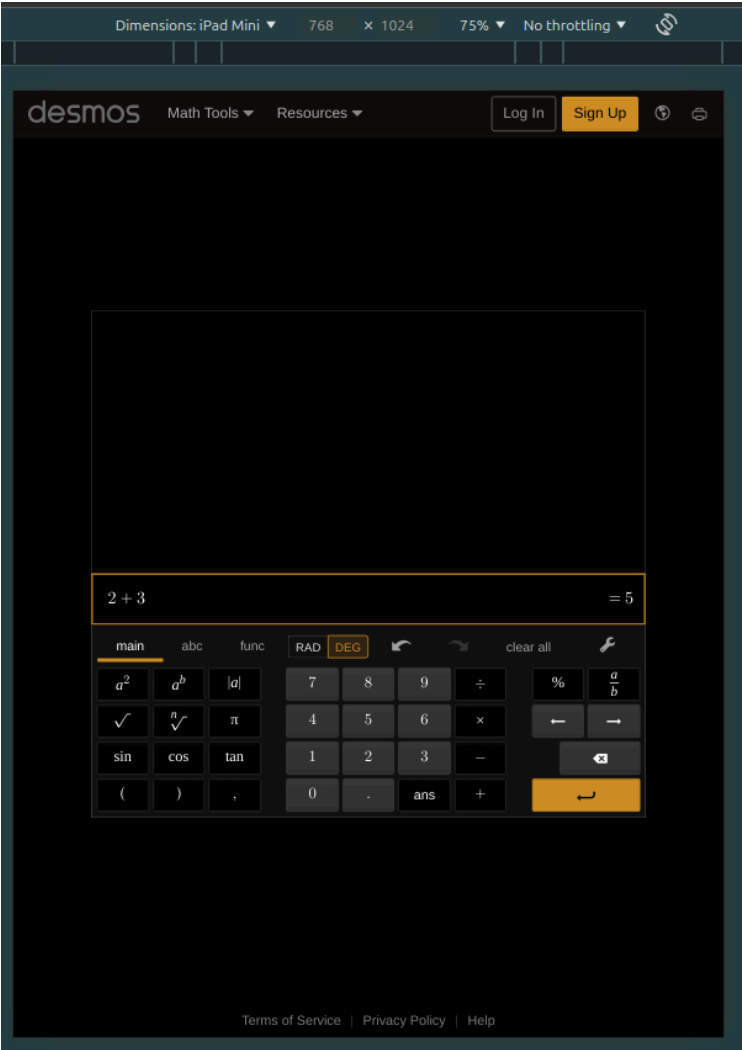
-> Программа выдает правильные результаты при сценариях тестирования пользовательского интерфейса (UI Testing).

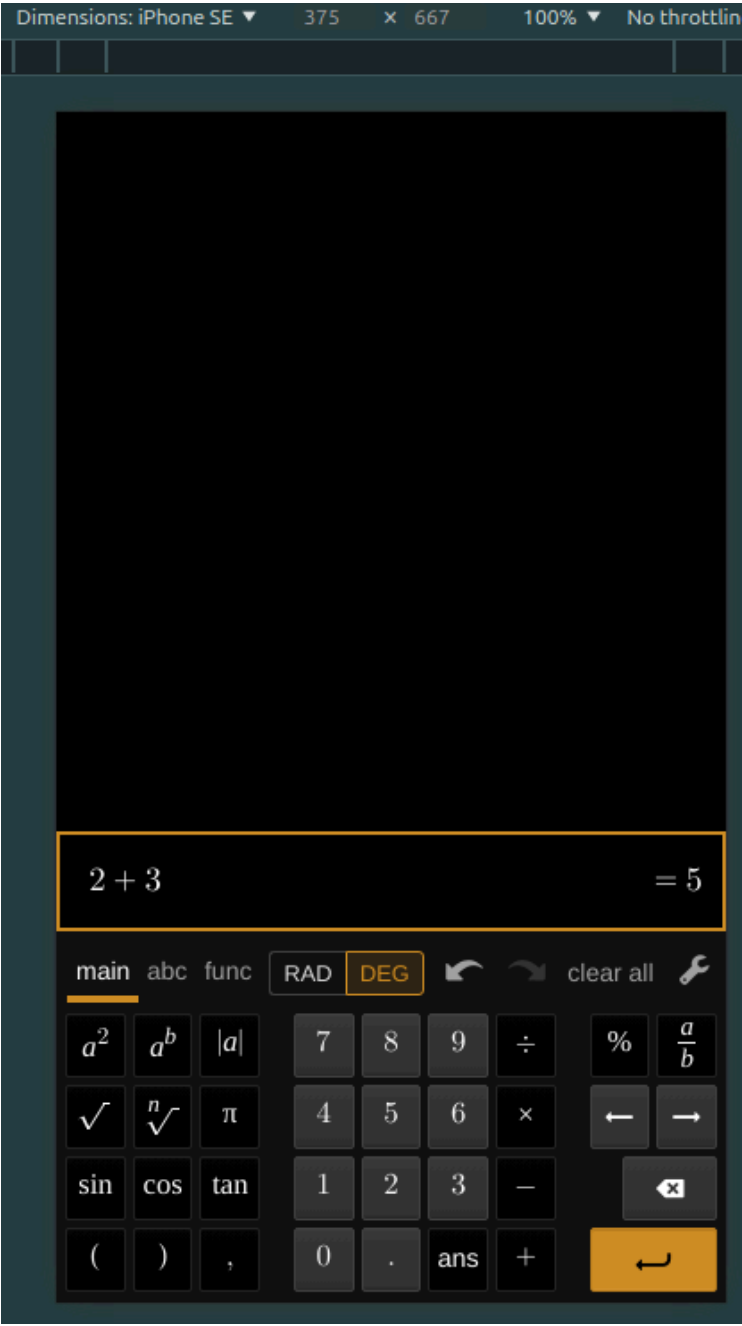
2.6. Сценарии тестирования производительности (Performance Testing)

№	Название теста	Входные данные	Ожидаемый результат	Результат
1	Длинная последовательность операций	1+2+3+...+1000	Результат отображается четко и правильно	

2.7. Сценарии тестирования совместимости (Compatibility Testing)

№	Название теста	Входные данные	Ожидаемый результат	Результат
1	Проверка на разных браузерах	2+3	Корректная работа на Chrome, Firefox, Yandex	 <p>Chrome</p>  <p>Firefox</p>

				Yandex
2	Проверка на разных устройствах	2+3	Корректная работа на ПК, планшетах, телефонах	 <p>на ПК</p>  <p>на планшете</p>

				 <p>на телефоне</p>
--	--	--	--	--

-> Программа выдает правильные результаты при сценариях тестирования совместимости (Compatibility Testing)

2.8. Сценарии тестирования граничных значений (Boundary Testing)

№	Название теста	Входные данные	Ожидаемый результат	Результат
1	Максимально возможное число	$10^{308} + 1$	Корректный результат или сообщение об ограничении	
2	Минимально возможное число	$-10^{308} - 1$	Корректный результат или сообщение об ограничении	

-> Программа отображает правильный результат, но не полностью соответствует ожиданиям, так как не отображает все цифры в результате вычисления.

2.9. Сценарии тестирования специальных функций

№	Название теста	Входные данные	Ожидаемый результат	Результат
1	Использование других функций (Функция построения графиков)	$(x^2 + y^2 - 1)^3 - x^2 y^3 = 0$	Отображение графика в форме сердца.	
2	Функция сохранения вычисления	$(x^2 + y^2 - 1)^3 - x^2 y^3 = 0$	Успешное сохранение графика и уравнения.	

- > Программа выдает правильные результаты при сценариях тестирования специальных функций

ЗАКЛЮЧЕНИЕ

Для тестирования веб-сайта calculator.net были применены методы "черного ящика", тестирования производительности и тестирования на основе требований. Тестовые сценарии включали проверку базовой функциональности, обработки ошибок, производительности, совместимости и пользовательского интерфейса. Сайт успешно прошел большинство тестов, показав корректную работу арифметических операций, устойчивость при высоких нагрузках и совместимость с различными устройствами и браузерами. Выявленные мелкие недостатки касаются обработки крайних случаев и сообщений об ошибках. В целом, сайт соответствует требованиям и демонстрирует высокую стабильность.