

Документ подписан простой электронной подписью

Информация о владельце:

ФИО: Фурсов Виталий Сергеевич

Должность: И.о. директора Пятигорского института (филиал) Северо-Кавказского
федерального университета

Дата подписания: 08.06.2026 13:50:45

Уникальный программный ключ:

1c378726a41fd0143ae5bccc8ba81860b00daa62

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Пятигорский институт (филиал) СКФУ

Колледж Пятигорского института (филиал) СКФУ

УТВЕРЖДАЮ

И.О. Директора
Пятигорского института
(филиал) СКФУ
В.А. Фурсов

ПМ.03 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ИНФОРМАЦИОННЫХ СИСТЕМ
МДК.03.04 ТЕСТИРОВАНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ
МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ

Специальности СПО

09.02.11 Разработка и управление программным обеспечением

Пятигорск 2026

Методические указания для выполнения лабораторных работ по дисциплине МДК.03.04 Тестирование информационных систем, составлены в соответствии с требованиями ФГОС СПО. Предназначены для студентов, обучающихся по специальности 09.02.11 Разработка и управление программным обеспечением

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Методические указания для выполнения лабораторных работ по дисциплине «Тестирование информационных систем» составлены в соответствии с требованиями ФГОС СПО. Предназначены для студентов, обучающихся по специальности 09.02.07 «Информационные системы и программирование».

В результате изучения дисциплины студенты должны:

Знать:

основные понятия, принципы и стратегии тестирования;
классификацию ошибок программного обеспечения;
методы тестирования «черного» и «белого» ящика;
инструментальные средства тестирования и автоматизации;
требования к документированию процесса тестирования.

Уметь:

осуществлять тестирование информационной системы на этапе опытной эксплуатации с фиксацией выявленных ошибок;
разрабатывать тестовые сценарии и тестовую документацию;
применять методы ручного и автоматизированного тестирования;
анализировать результаты тестирования и оценивать качество ПО.

Владеть:

навыками проектирования тестовых наборов;
технологиями проведения сертификации программного средства;
методами идентификации сбоев и ошибок.

В результате освоения учебной дисциплины студент должен овладеть:

Общими компетенциями:

ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам.

ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности.

ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по правовой и финансовой грамотности в различных жизненных ситуациях.

ОК 04. Эффективно взаимодействовать и работать в коллективе и команде.

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста.

ОК 06. Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных российских духовно-нравственных ценностей, в том числе с учетом гармонизации межнациональных и межрелигиозных отношений, применять стандарты антикоррупционного поведения.

ОК 07. Содействовать сохранению окружающей среды, ресурсосбережению, применять знания об изменении климата, принципы бережливого производства, эффективно действовать в чрезвычайных ситуациях.

ОК 08. Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности.

ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках.

Профессиональными компетенциями:

ПК 3.1. Собирать исходные данные для разработки проектной документации на информационную систему.

ПК 3.2. Разрабатывать проектную документацию на разработку информационной системы в соответствии с требованиями заказчика.

ПК 3.3. Разрабатывать подсистемы безопасности информационной системы в соответствии с техническим заданием.

ПК 3.4. Производить разработку модулей информационной системы в соответствии с техническим заданием.

ПК 3.5. Интегрировать информационную систему с существующими информационными системами заказчика.

ПК 3.6. Осуществлять модульное и интеграционное тестирование информационной системы.

ПК 3.7. Разрабатывать техническую документацию на эксплуатацию информационной системы.

ПК 3.8. Производить оценку информационной системы для выявления возможности ее модернизации.

ОБЩИЕ МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

По дисциплине «Тестирование информационных систем» лабораторные работы содержат задачи и теоретические вопросы. Варианты для каждого обучающегося – индивидуальные. Задачи и ответы на вопросы, выполненные не по своему варианту, не засчитываются.

Лабораторная работа выполняется студентами на персональном компьютере с использованием соответствующего программного обеспечения. В процессе выполнения задания обучающиеся выполняют практические действия, направленные на освоение изучаемых технологий и инструментов. Полученные результаты работы, а также выполненные операции и промежуточные этапы фиксируются и оформляются в виде отчёта по лабораторной работе.

Отчёт должен содержать условия задания, описание хода выполнения работы, используемые инструменты и методы, а также полученные результаты. При необходимости в отчёт включаются схемы, таблицы, иллюстрации и другие материалы, подтверждающие выполнение задания. Все результаты должны быть представлены в структурированном виде и соответствовать требованиям оформления учебных работ.

При выполнении лабораторной работы необходимо следовать методическим указаниям: повторить краткое содержание теории, запомнить основные формулы и законы, проанализировать пример выполнения аналогичного задания, затем приступить непосредственно к решению задачи. К зачету допускаются студенты, получившие положительные оценки по всем лабораторным работам.

ПРАВИЛА ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ

Студент должен прийти на лабораторную работу подготовленным к выполнению лабораторной работы.

Каждый студент после проведения работы должен представить отчет о проделанной работе с анализом полученных результатов и выводом по работе.

Таблицы и рисунки следует выполнять с помощью чертежных инструментов (линейки, циркуля и т.д.) карандашом с соблюдением ЕСКД.

Расчет следует проводить с точностью до двух значащих цифр.

Исправления проводить на обратной стороне листа. При мелких исправлениях неправильное слово (буква, число и т.п.) аккуратно зачеркивается и над ним пишется правильное пропущенное слово (букву, число и т.п.).

Вспомогательные расчеты можно выполнять на отдельных листах, а при необходимости на листах отчета.

Если студент не выполнит лабораторную работу или часть работы, то он выполнит ее во внеурочное время, согласованное с преподавателем.

Оценку по лабораторной работе студент получает с учетом срока выполнения работы, если: расчеты выполнены правильно и в полном объеме;

сделан анализ проделанной работы и вывод по результатам работы;

студент может пояснить выполнение любого этапа работы;

отчет выполнен в соответствии с требованиями к выполнению работы.

ЛАБОРАТОРНАЯ РАБОТА № 1

ИЗУЧЕНИЕ ПРИНЦИПОВ И СТРАТЕГИЙ ТЕСТИРОВАНИЯ ПРОГРАММ

Цель работы: Изучить понятия, основные принципы и стратегии тестирования программного обеспечения

Теоретическая часть

1. Определение термина «тестирование»

Тестирование – это процесс исполнения программы с целью обнаружения ошибок.

Из приведенного определения тестирования вытекает несколько следствий. Например, одно из них состоит в том, что тестирование – процесс деструктивный (т. е. обратный созидательному, конструктивному). Именно этим и объясняется, почему многие программисты и тестировщики считают его трудным. Большинство людей склонны к конструктивному процессу созидания объектов и в меньшей степени – к деструктивному процессу разделения на части.

2. Экономика тестирования

В общем случае невозможно обнаружить все ошибки программы. А это в свою очередь порождает экономические проблемы, задачи, связанные с функциями человека в процессе отладки, способы построения тестов.

2.1 Тестирование программы как черного ящика

При использовании этой стратегии программа рассматривается как черный ящик. Иными словами, такое тестирование имеет целью выяснение обстоятельств, в которых поведение программы не соответствует ее спецификации. Тестовые же данные используются только в соответствии со спецификацией программы (т. е. без учета знаний о ее внутренней структуре).

2.2 Тестирование программы как белого ящика

Стратегия белого ящика, или стратегия тестирования, управляемого логикой программы, позволяет исследовать внутреннюю структуру программы. В этом случае тестирующий получает тестовые данные путем анализа логики программы.

3. Принципы тестирования

1. Описание предполагаемых значений выходных данных или результатов должно быть необходимой частью тестового набора.
2. Следует избегать тестирования программы ее автором.
3. Программирующая организация не должна сама тестировать разработанные ею программы.
4. Необходимо досконально изучать результаты применения каждого теста.
5. Тесты для неправильных и непредусмотренных входных данных следует разрабатывать так же тщательно, как для правильных и предусмотренных.
6. Необходимо проверять не только, делает ли программа то, для чего она предназначена, но и не делает ли она то, что не должна делать.
7. Не следует выбрасывать тесты, даже если программа уже не нужна.
8. Нельзя планировать тестирование в предположении, что ошибки не будут обнаружены.
9. Вероятность наличия необнаруженных ошибок в части программы пропорциональна числу ошибок, уже обнаруженных в этой части.
10. Тестирование – процесс творческий.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, одной из систем программирования Microsoft Visual Studio, IDLE (Python 3.*) или др., а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Порядок выполнения работы:

1. Изучить теоретический материал.

2. В таблице 1 приведены варианты индивидуальных заданий, а в листинге 1 - программы на языке Python для решения этих задач, содержащие, по крайней мере, две логические ошибки.
3. Для обнаружения ошибок в программе составьте не менее 10 тестов в виде таблицы 2 с использованием стратегии черного ящика.
4. Наберите текст предложенной программы и выполните ее на каждом из подготовленных тестов.
5. Заполните таблицу с результатами тестирования (по образцу таблицы 3).
6. Найдите и исправьте обнаруженные в процессе тестирования ошибки (нужно именно внести исправления в предложенную программу, а не написать свой вариант ее решения).
7. Оформить отчет в электронном виде. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:
 - титульный лист с темой лабораторной работы;
 - цель работы;
 - вариант задания и исходную программу;
 - таблицу с составленными тестами;
 - результаты первого тестирования;
 - исправленный вариант программы;
 - результаты повторного тестирования;
 - выводы о проделанной работе.

Таблица 1
Варианты заданий

№ варианта	Задание
1	Требовалось написать программу, которая получает на вход натуральное число N , не превосходящее 10^9 , и выводит число, которое получается из N после удаления всех единиц; порядок остальных цифр при этом не меняется.
2	Дано натуральное число N , не превосходящее 10^8 . Необходимо найти и вывести число, которое получится при записи N справа налево и удалении всех единиц.
3	Требовалось написать программу, при выполнении которой с клавиатуры считывается натуральное число x , не превосходящее 1000, и выводится количество значащих цифр в двоичной записи этого числа.
4	Для заданного положительного вещественного числа A необходимо найти минимальное целое число K , при котором выполняется неравенство.
5	На обработку поступает натуральное число, не превышающее 10^9 . Нужно написать программу, которая выводит на экран максимальную цифру числа, кратную 3.
6	Даны целые положительные числа M и N . Необходимо определить количество таких натуральных чисел K , для которых выполняется неравенство $M \leq K^2 \leq N$.
7	Дано целое положительное число N , не превосходящее 1000. Необходимо определить, является ли это число степенью числа 7.
8	Дано целое положительное число A . Требуется вывести такое минимально возможное нечётное натуральное число K , при котором сумма квадратов первых нечётных чисел окажется больше A .
9	Дано целое положительное число A . Требуется вывести такое минимально возможное натуральное число K , при котором сумма окажется больше A .
10	Числа Фибоначчи задаются рекуррентным соотношением. Требовалось написать программу, которая по входному целому неотрицательному значению n определяет соответствующее число Фибоначчи.

Таблица 2

Таблица тестов

№ теста	Значения входных переменных	Ожидаемый результат
1		
2		
3		
...		

Таблица 3

Таблица результатов тестирования

№ теста	Значения входных переменных	Ожидаемый результат	Фактический результат	Результат тестирования (успешно / неуспешно)
1				
2				
3				
...				

Контрольные вопросы:

1. Дайте определение понятию «тестирование».
2. Сформулируйте характеристики удачного теста.
3. Поясните, почему тестирование является деструктивным процессом.
4. Почему при тестировании необходимо желать, чтобы программа дала сбой?
5. Придумайте, каким образом два программиста, создающие одну программу, могут протестировать ее, не нарушая принципов тестирования.
6. Покажите и докажите на конкретной программе, что абсолютное тестирование невозможно.
7. Перечислите принципы тестирования.
8. Почему в участке кода, где было обнаружено больше всего ошибок, может содержаться еще большее их количество?
9. К чему ведет планирование теста в предположении отсутствия ошибок?
10. Приведите пример, когда отсутствие тестов, проверяющих, не делает ли программа лишних действий (что она не должна делать), может повлечь ошибки в работе?
11. Что дает проверка программы на неправильных входных данных?
12. Зачем необходимо хранить тесты, если программа уже выпущена?
13. В чем заключается принцип тестирования программы как белого ящика?
14. В чем заключается принцип тестирования программы как черного ящика?

Список литературы, рекомендуемый к использованию по данной теме:

1. Луиза Тамре. Введение в тестирование программного обеспечения. - М.: Издательский дом «Вильямс», 2013.
2. Майерс, Г. Искусство тестирования программ. – М.: Финансы и статистика, 2001.

ЛАБОРАТОРНАЯ РАБОТА № 2

ТИПЫ ОШИБОК И АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ ПРОГРАММ

Цели работы:

1. изучить основные типы ошибок в программах;
2. научиться автоматизировать прогон тестов и составление отчета о результатах тестирования

Теоретическая часть

1. Классификация ошибок

Задача любого тестировщика заключается в нахождении наибольшего количества ошибок, поэтому он должен хорошо знать наиболее часто допускаемые ошибки и уметь находить их за минимально короткий период времени.

Ошибка – это расхождение между вычисленным, наблюдаемым и истинным, заданным или теоретически правильным значением.

Итак, по времени появления ошибки можно разделить на три вида:

1. структурные ошибки набора;
2. ошибки компиляции;
3. ошибки периода выполнения.

Структурные ошибки возникают непосредственно при наборе программы.

Ошибки компиляции возникают из-за ошибок в тексте кода. Они включают ошибки в синтаксисе, неверное использование конструкций языка, использование несуществующих объектов или свойств, методов у объектов.

Ошибки периода выполнения возникают, когда программа выполняется и компилятор (или операционная система, виртуальная машина) обнаруживает, что оператор делает попытку выполнить недопустимое или невозможное действие.

В теоретической информатике программные ошибки классифицируют по степени нарушения логики на:

1. синтаксические;
2. семантические;
3. прагматические.

Синтаксические ошибки заключаются в нарушении правописания или пунктуации в записи выражений, операторов и т. п., т. е. в нарушении грамматических правил языка.

Семантические ошибки заключаются в нарушении порядка операторов, параметров функций и употреблении выражений.

Прагматические ошибки (или логические) заключаются в неправильной логике алгоритма, нарушении смысла вычислений и т. п.

2. Автоматизация тестирования и составления отчета

Если разработанные тесты записать в текстовый файл и написать программу, которая будет последовательно считывать очередной тест из файла, прогонять на нем программу и записывать ожидаемый и фактический результаты в файл отчета, то процесс тестирования и документирования будет выполняться автоматически.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, одной из систем программирования Microsoft Visual Studio, IDLE (Python 3.*) или др., а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Порядок выполнения работы:

1. Записать построчно в текстовый файл tests.txt тесты, составленные в лабораторной работе 1. Значения входных переменных и ожидаемый результат должны быть заданы в одной строке файла через пробел.
2. Оформить первоначальный вариант программы из лабораторной 1 в виде функции program1(), а исправленный – в виде функции program2().
3. Написать функцию auto_test() на языке Python, которая будет считывать из текстового файла tests.txt подготовленные тесты, выполнять программу program1() на каждом из тестов, сравнивать полученный результат с ожидаемым, и записывать выводы по каждому тесту в файл report1.txt.
4. Выполнить функцию auto_test для исходного и исправленного вариантов программы. Результаты тестирования исправленного варианта записать в файл report2.txt.

5. Оформить отчет в электронном виде. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:

- титульный лист с темой лабораторной работы;
- цель работы;
- вариант задания и исходную программу;
- скриншоты функций program1() и auto_test;
- скриншот файла tests.txt с тестами;
- скриншот файла report1.txt;
- скриншоты функций program2() и auto_test;
- скриншот файла report2.txt;
- выводы о проделанной работе.

Контрольные вопросы

1. Дайте определение термина «ошибка».
2. Приведите классификацию ошибок по времени их появления.
3. Приведите классификацию ошибок по степени нарушения логики.
4. Определите вид ошибки: if $x > 3$ and $x < 2$...
5. Какие типовые ошибки встречаются в программах?
6. Приведите пример ошибки обращения к данным.
7. Приведите пример ошибки описания данных.
8. Приведите пример ошибки интерфейса.
9. Приведите пример ошибки передачи управления.
10. Приведите пример ошибки при сравнениях.
11. Приведите пример ошибки вычисления.
12. Приведите пример ошибки ввода-вывода.

Список литературы, рекомендуемый к использованию по данной теме:

1. Луиза Тамре. Введение в тестирование программного обеспечения. - М.: Издательский дом «Вильямс», 2013.

ЛАБОРАТОРНАЯ РАБОТА № 3 ПРОВЕДЕНИЕ ИНСПЕКЦИИ КОДА В ГРУППЕ

Цель работы: Освоение методов ручного тестирования, получение навыков проведения инспекций исходного текста.

Теоретическая часть

1. Инспекции и сквозные просмотры

Инспекции исходного текста и сквозные просмотры являются основными методами ручного тестирования. Так как эти два метода имеют много общего, они рассматриваются здесь совместно. Инспекции и сквозные просмотры включают в себя чтение или визуальную проверку программы группой лиц. Эти методы развиты из идей Вейнберга. Оба метода предполагают некоторую подготовительную работу. Завершающим этапом является «обмен мнениями» – собрание, проводимое участниками проверки. Цель такого собрания – нахождение ошибок, но не их устранение (т. е. тестирование, а не отладка).

2. Инспекции исходного текста

Инспекции исходного текста представляют собой набор процедур и приемов обнаружения ошибок при изучении (чтении) текста группой специалистов. При рассмотрении инспекций исходного текста внимание будет сосредоточено в основном на методах, процедурах, формах выполнения и т. д.

Инспектирующая группа включает обычно четыре человека, один из которых выполняет функции председателя. Председатель должен быть компетентным программистом, но не автором программы; он не должен быть знаком с ее деталями.

3. Список вопросов для выявления ошибок при инспекции

3.1 Ошибки обращения к данным

1. Используются ли переменные с неустановленными значениями?
2. Лежат ли индексы вне заданных границ?
3. Есть ли нецелые индексы?
4. Соответствуют ли друг другу определение структуры и ее использование в различных методах?
5. Превышены ли границы строки?

3.2 Ошибки описания данных

1. Все ли переменные описаны?
2. Правильно ли инициализированы объекты, массивы и строки?
3. Понятны ли имена переменных?
4. Нельзя ли обойтись без переменных со сходными именами?
5. Корректно ли произведено описание класса?

3.3 Ошибки вычислений

1. Производятся ли вычисления с использованием данных разного типа?
2. Возможно ли переполнение или потеря промежуточного результата при вычислении?
3. Есть ли деление на ноль?
4. Существуют ли неточности при работе с двоичными числами?
5. Не выходит ли значение переменной за пределы установленного диапазона?

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, одной из систем программирования Microsoft Visual Studio, IDLE (Python 3.*) или др., а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Порядок выполнения работы:

1. Изучить теоретические сведения об инспекциях.
2. Сформировать группу из 3-4 человек и распределить роли (председатель, автор, тестировщик, проектировщик).
3. Получить код для инспекции у преподавателя.
4. Провести заседание инспекционной группы (90-120 минут).
5. Составить список найденных ошибок по категориям.
6. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:
 - титульный лист с темой лабораторной работы;
 - цель работы;
 - состав инспекционной группы и роли;
 - список найденных ошибок по категориям;
 - рекомендации по исправлению;
 - вывод.

Таблица 1

Список найденных ошибок

№ п/п	Категория ошибки	Описание ошибки	Строка кода	Рекомендация по исправлению
1				
2				
3				

...				
-----	--	--	--	--

Контрольные вопросы

1. В чем заключается сущность инспекции?
2. Какие этапы включает метод инспекции?
3. Кто входит в состав инспекционной группы?
4. Какова оптимальная продолжительность инспекционного заседания?
5. Почему инспекция эффективнее проверки за столом?
6. Какие категории ошибок существуют?
7. Приведите пример ошибки обращения к данным.
8. Приведите пример ошибки описания данных.
9. Приведите пример ошибки вычислений.
10. Приведите пример ошибки при сравнениях.
11. Приведите пример ошибки в передачах управления.
12. Приведите пример ошибки интерфейса.
13. Приведите пример ошибки ввода-вывода.

Список литературы, рекомендуемый к использованию по данной теме:

1. Вейнберг, Дж. Психология программирования для ЭВМ. - М.: Радио и связь, 1981.
2. Майерс, Г. Искусство тестирования программ. – М.: Финансы и статистика, 2001.

ЛАБОРАТОРНАЯ РАБОТА № 4 РАЗРАБОТКА ТЕСТОВ МЕТОДОМ ЭКВИВАЛЕНТНОГО РАЗБИЕНИЯ

Цель работы: Освоение метода эквивалентного разбиения для проектирования тестовых наборов.

Теоретическая часть

1. Эквивалентное разбиение

В лекции 1 отмечалось, что хороший тест имеет приемлемую вероятность обнаружения ошибки, и что исчерпывающее входное тестирование программы невозможно. Следовательно, тестирование программы ограничивается использованием небольшого подмножества всех возможных входных данных.

Эквивалентное разбиение – методология тестирования по принципу черного ящика, основанная на разбиении входной области программы на конечное число классов эквивалентности.

2. Выделение классов эквивалентности

Классы эквивалентности выделяются путем выбора каждого входного условия (обычно это предложение или фраза в спецификации) и разбиением его на две или более групп.

Различают два типа классов эквивалентности:

- правильные классы эквивалентности, представляющие правильные входные данные программы;
- неправильные классы эквивалентности, представляющие все другие возможные состояния условий (т. е. ошибочные входные значения).

3. Правила выделения классов эквивалентности

1. Если входное условие описывает область значений – один правильный и два неправильных класса.
2. Если входное условие описывает число значений – один правильный и два неправильных класса.
3. Если входное условие описывает множество значений – правильный класс для каждого значения.
4. Если входное условие описывает ситуацию «должно быть» – один правильный и один неправильный класс.

4. Построение тестов

1. Назначение каждому классу эквивалентности уникального номера.

2. Проектирование новых тестов, каждый из которых покрывает как можно большее число непокрытых правильных классов эквивалентности.
3. Запись тестов, каждый из которых покрывает один и только один из непокрытых неправильных классов эквивалентности.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, одной из систем программирования Microsoft Visual Studio, IDLE (Python 3.*) или др., а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Порядок выполнения работы:

1. Изучить теоретические сведения о методе эквивалентного разбиения.
2. Получить спецификацию тестируемой функции у преподавателя.
3. Составить таблицу классов эквивалентности.
4. Разработать тестовые сценарии для каждого класса.
5. Выполнить тестирование.
6. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:
 - титульный лист с темой лабораторной работы;
 - цель работы;
 - таблица классов эквивалентности;
 - список тестовых сценариев;
 - результаты выполнения тестов;
 - вывод.

Таблица 1

Классы эквивалентности

Входные условия	Правильные классы эквивалентности	Неправильные классы эквивалентности
1		
2		
3		
...		

Таблица 2

Тестовые сценарии

№ теста	Класс эквивалентности	Входные данные	Ожидаемый результат	Фактический результат	Результат
1					
2					
3					
...					

Контрольные вопросы

1. Что такое класс эквивалентности?
2. Какие правила выделения классов эквивалентности существуют?
3. Почему неправильные классы тестируются индивидуальными тестами?
4. Приведите пример класса эквивалентности.
5. В чем недостаток метода эквивалентного разбиения?
6. Чем правильный класс эквивалентности отличается от неправильного?
7. Сколько тестов нужно для покрытия одного правильного класса?
8. Сколько тестов нужно для покрытия одного неправильного класса?

9. Что такое спецификация программы?
10. Как определить входные условия для разбиения?

Список литературы, рекомендуемый к использованию по данной теме:

1. Майерс, Г. Искусство тестирования программ. – М.: Финансы и статистика, 2001.
2. Луиза Тамре. Введение в тестирование программного обеспечения. - М.: Издательский дом «Вильямс», 2013.

ЛАБОРАТОРНАЯ РАБОТА № 5 ТЕСТИРОВАНИЕ МЕТОДОМ ГРАНИЧНЫХ ЗНАЧЕНИЙ

Цель работы: Освоение метода анализа граничных значений для проектирования эффективных тестов.

Теоретическая часть

1. Анализ граничных значений

Как показывает опыт, тесты, исследующие граничные условия, приносят большую пользу, чем тесты, которые их не исследуют.

Граничные условия – это ситуации, возникающие непосредственно на, выше или ниже границ входных и выходных классов эквивалентности.

2. Правила метода

1. Построить тесты для границ области и тесты с неправильными входными данными для ситуаций незначительного выхода за границы области.

2. Построить тесты для минимального и максимального значений условий и тесты, большие и меньшие этих значений.

3. Использовать первое правило для каждого выходного условия.

4. Использовать второе правило для каждого выходного условия.

5. Если вход или выход программы есть упорядоченное множество, то сосредоточить внимание на первом и последнем элементах этого множества.

3. Алгоритм использования техники граничных значений

1. Выделить классы эквивалентности.

2. Определить граничные значения этих классов.

3. Нужно понять, к какому классу будет относиться каждая граница.

4. Нужно провести тесты по проверке значения до границы, на границе и сразу после границы.

Количество тестов для проверки граничных значений будет равно количеству границ, умноженному на 3.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, одной из систем программирования Microsoft Visual Studio, IDLE (Python 3.*) или др., а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Порядок выполнения работы:

1. Изучить теоретические сведения о методе граничных значений.
2. Получить спецификацию тестируемой функции у преподавателя.
3. Определить граничные значения для каждого условия.
4. Разработать тестовые сценарии (до границы, на границе, после границы).
5. Выполнить тестирование.

6. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:

- титульный лист с темой лабораторной работы;
- цель работы;
- таблица граничных значений;
- список тестовых сценариев;
- результаты выполнения тестов;
- вывод.

Таблица 1

Граничные значения

Условие	Минимальное значение	Граница	Максимальное значение	Тесты

Таблица 2

Тестовые сценарии

№ теста	Входные данные	Ожидаемый результат	Фактический результат	Результат
1				
2				
3				
...				

Контрольные вопросы

1. Что такое граничные значения?
2. Чем анализ граничных значений отличается от эквивалентного разбиения?
3. Сколько тестов нужно для проверки каждой границы?
4. Приведите пример граничных значений для диапазона 1-100.
5. Почему граничные условия важны при тестировании?
6. Какие значения нужно проверять для каждой границы?
7. Что такое пространство результатов?
8. Как определить граничные значения для выходных данных?
9. Почему тесты на границах более эффективны?
10. Приведите пример ошибки, обнаруживаемой на границе.

Список литературы, рекомендуемый к использованию по данной теме:

1. Майерс, Г. Искусство тестирование программ. – М.: Финансы и статистика, 2001.
2. Луиза Тамре. Введение в тестирование программного обеспечения. - М.: Издательский дом «Вильямс», 2013.

ЛАБОРАТОРНАЯ РАБОТА № 6 ПОСТРОЕНИЕ ГРАФА ПОТОКА УПРАВЛЕНИЯ. ТЕСТЫ НА ПОКРЫТИЕ ОПЕРАТОРОВ

Цель работы: Освоение метода структурного тестирования «белого ящика», построение графа потока управления.

Теоретическая часть

1. Тестирование по принципу белого ящика

Тестирование по принципу белого ящика характеризуется степенью, в какой тесты выполняют или покрывают логику (исходный текст) программы.

2. Покрытие операторов

Если отказаться полностью от тестирования всех путей, то можно показать, что критерием покрытия является выполнение каждого оператора программы, по крайней мере, один раз. Это метод покрытия операторов.

К сожалению, это слабый критерий, так как выполнение каждого оператора, по крайней мере, один раз есть необходимое, но не достаточное условие для приемлемого тестирования по принципу белого ящика.

3. Покрытие решений (покрытие переходов)

Более сильный критерий покрытия логики программы известен как покрытие решений, или покрытие переходов. Согласно данному критерию должно быть записано достаточное число тестов, такое, чтобы каждое условие на этих тестах принимало значение истина и ложь по крайней мере один раз.

4. Граф потока управления

Граф потока управления – представление программы в виде вершин (участки кода) и дуг (передачи управления).

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, одной из систем программирования Microsoft Visual Studio, IDLE (Python 3.*) или др., а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Порядок выполнения работы:

1. Изучить теоретические сведения о стратегии белого ящика.
2. Получить исходный код программы у преподавателя.
3. Построить граф потока управления.
4. Определить минимальный набор тестов для покрытия операторов.
5. Выполнить тестирование.
6. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:
 - титульный лист с темой лабораторной работы;
 - цель работы;
 - исходный код программы;
 - граф потока управления;
 - список тестовых сценариев;
 - результаты выполнения тестов;
 - вывод.

Таблица 1

Покрытие операторов

№ оператора	Код оператора	Покрыт (да/нет)	№ теста
1			
2			
3			
...			

Таблица 2

Тестовые сценарии

№ теста	Входные данные	Путь в графе	Покрытые операторы	Результат
1				
2				
3				
...				

Контрольные вопросы

1. В чем суть стратегии белого ящика?
2. Опишите процесс тестирования методом покрытия операторов.
3. Что такое граф потока управления?
4. Почему покрытие операторов является слабым критерием?
5. Какие этапы входят в проектирование теста?
6. Что такое покрытие решений?
7. Чем покрытие решений отличается от покрытия операторов?
8. Как построить граф потока управления?
9. Что такое вершина графа?
10. Что такое дуга графа?

Список литературы, рекомендуемый к использованию по данной теме:

1. Майерс, Г. Искусство тестирования программ. – М.: Финансы и статистика, 2001.
2. Антамошкин, О.А. Программная инженерия. Теория и практика: учебник. – Красноярск: Сибирский федеральный университет, 2012.

ЛАБОРАТОРНАЯ РАБОТА № 7 РАЗРАБОТКА ТЕСТОВ ДЛЯ ПОКРЫТИЯ УСЛОВИЙ СЛОЖНОГО АЛГОРИТМА

Цель работы: Освоение методов комбинаторного покрытия условий при структурном тестировании.

Теоретическая часть

1. Покрытие условий

Покрытие условий – записывается число тестов, достаточное для того, чтобы все возможные результаты каждого условия в решении выполнялись, по крайней мере, один раз.

2. Покрытие решений/условий

Покрытие решений/условий – все возможные результаты каждого условия в решении и все результаты каждого решения выполняются хотя бы один раз.

3. Комбинаторное покрытие условий

Комбинаторное покрытие условий – все возможные комбинации результатов условия в каждом решении выполняются хотя бы один раз.

4. Критерии покрытия

Для программ, содержащих только одно условие на каждое решение, минимальным является критерий, набор тестов которого:

1. вызывает выполнение всех результатов каждого условия, по крайней мере, один раз;
2. передает управление каждой точке входа по крайней мере один раз.

Для программ, содержащих решения, каждое из которых имеет более одного условия, минимальный критерий состоит из набора тестов, вызывающих выполнение всех возможных комбинаций результатов условий в каждом решении.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, одной из систем программирования Microsoft Visual Studio, IDLE (Python 3.*) или др., а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Порядок выполнения работы:

1. Изучить теоретические сведения о покрытии условий.
2. Получить исходный код программы у преподавателя.
3. Выделить все условия и решения.
4. Составить таблицу комбинаций условий.
5. Разработать тестовые сценарии.
6. Выполнить тестирование.
7. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:

- титульный лист с темой лабораторной работы;
- цель работы;
- исходный код программы;
- таблица комбинаций условий;
- список тестовых сценариев;
- результаты выполнения тестов;
- вывод.

Таблица 1
Комбинации условий

№ комбинации	Условие 1	Условие 2	Условие 3	Результат
1	Истина	Истина	Истина	
2	Истина	Истина	Ложь	
3	Истина	Ложь	Истина	
4	Истина	Ложь	Ложь	
5	Ложь	Истина	Истина	
6	Ложь	Истина	Ложь	
7	Ложь	Ложь	Истина	
8	Ложь	Ложь	Ложь	

Таблица 2
Тестовые сценарии

№ теста	Входные данные	Покрытые комбинации	Результат
1			
2			
3			
...			

Контрольные вопросы

1. Опишите процесс тестирования методом покрытия условий.
2. Опишите процесс тестирования методом покрытия решений/условий.
3. Опишите процесс тестирования методом комбинаторного покрытия условий.
4. В чем недостаток критерия покрытия решений/условий?

5. Сколько тестов нужно для комбинаторного покрытия двух условий?
6. Что такое условие в программе?
7. Что такое решение в программе?
8. Чем покрытие условий отличается от покрытия решений?
9. Почему комбинаторное покрытие более полное?
10. Какие ограничения существуют для комбинаторного покрытия?

Список литературы, рекомендуемый к использованию по данной теме:

1. Майерс, Г. Искусство тестирования программ. – М.: Финансы и статистика, 2001.
2. Антамошкин, О.А. Программная инженерия. Теория и практика: учебник. – Красноярск: Сибирский федеральный университет, 2012.

ЛАБОРАТОРНАЯ РАБОТА № 8 ПОИСК И ИСПРАВЛЕНИЕ ЛОГИЧЕСКИХ ОШИБОК С ИСПОЛЬЗОВАНИЕМ ОТЛАДЧИКА VISUAL STUDIO

Цель работы: Закрепление навыков использования инструментов отладки для поиска логических ошибок.

Теоретическая часть

1. Логическая ошибка

Логическая ошибка – это ошибка в программе, приводящая к тому, что программа выдает неверные результаты. Большинство усилий при отладке тратится именно на поиск и исправление логических ошибок, совершенных программистом.

2. Способы обнаружения логических ошибок

1. Тщательный анализ кода.

2. Использование инструментов для отладки:

- точек останова;

- специальных окон для проверки значений переменных во время выполнения программы.

3. Использование режима останова

Режим останова позволяет выполнять код программы по одной строке и изучать изменение значений одной или нескольких переменных.

Для установки точки останова переместите указатель мыши к полосе Указатель поля напротив строки программы, в которой надо установить точку останова, а затем щелкните на этой полосе.

4. Использование окон «Видимые» и «Локальные»

Окно «Видимые» показывает состояние переменных и свойств, которые используются в текущий момент.

Окно «Локальные» позволяет видеть содержимое переменных и свойств на протяжении всего выполнения программы.

5. Использование окна «Контрольные значения»

Окно «Контрольные значения» позволяет видеть изменение значений не только переменных, но и любых выражений и свойств объектов в процедуре.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, Microsoft Visual Studio, а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Порядок выполнения работы:

1. Запустить Visual Studio и открыть проект с кодом.

2. Установить точки останова в подозрительных участках кода.
3. Запустить программу в режиме отладки.
4. Использовать окна отладки для анализа значений переменных.
5. Исправить обнаруженные ошибки.
6. Проверить исправленный код.
7. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:
 - титульный лист с темой лабораторной работы;
 - цель работы;
 - описание найденных логических ошибок;
 - скриншоты окон отладки;
 - листинг исправленного кода;
 - вывод.

Таблица 1
Найденные ошибки

№ ошибки	Тип ошибки	Строка кода	Описание	Способ исправления
1				
2				
3				
...				

Контрольные вопросы

1. Что такое логическая ошибка?
2. Какие инструменты отладки существуют в Visual Studio?
3. Как использовать окно «Контрольные значения»?
4. Что такое пошаговое выполнение?
5. Как исправить ошибку в условном операторе?
6. Что такое точка останова?
7. Как установить точку останова?
8. Чем отличается окно «Видимые» от окна «Локальные»?
9. Как запустить программу в режиме отладки?
10. Какие клавиши используются для пошагового выполнения?

Список литературы, рекомендуемый к использованию по данной теме:

1. Microsoft Visual Studio Documentation. - Microsoft, 2024.
2. Зубкова, Т.М. Технология разработки программного обеспечения: учебное пособие. – Оренбург: ОГУ, 2017.

ЛАБОРАТОРНАЯ РАБОТА № 9 НАПИСАНИЕ ТЕСТОВ, ПРОВОЦИРУЮЩИХ ИСКЛЮЧЕНИЯ. ПРОВЕРКА ОБРАБОТКИ

Цель работы: Освоение механизмов обработки исключительных ситуаций в программах.

Теоретическая часть

1. Обработка ошибок с помощью try...catch

Для обработки ошибок используется инструкция try...catch. Например, если ошибка времени исполнения произошла при загрузке файла с диска, обработчик ошибок может отобразить собственное сообщение об ошибке.

Синтаксис инструкции try...catch:

```
``csharp
try
{
    // Инструкции, которые могут вызвать исключение
}
catch
{
    // Инструкции, которые выполняются, если происходит исключение
}
finally
{
    // Дополнительные инструкции
}
...
```

2. Основные типы исключений

Тип исключения	Описание
System.NullReferenceException	Возникает при попытке обратиться к объекту, который равен null
System.DivideByZeroException	Деление на ноль
System.IndexOutOfRangeException	Выход за границы массива
System.IO.IOException	Ошибка ввода-вывода
System.ArgumentException	Неверный аргумент метода
System.FormatException	Ошибка преобразования строки в число

3. Блок catch

Блок catch выполняется, когда внутри блока try возникает ошибка. Блок catch имеет доступ к объекту исключения (Exception), который содержит информацию об ошибке.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, Microsoft Visual Studio, а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

Порядок выполнения работы:

1. Изучить теоретические сведения об обработке исключений.
2. Получить исходный код программы у преподавателя.
3. Добавить блоки try...catch.
4. Написать тесты для каждого типа исключения.
5. Проверить обработку исключений.
6. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:
 - титульный лист с темой лабораторной работы;
 - цель работы;
 - исходный код программы;
 - код с обработкой исключений;
 - список тестовых сценариев;
 - результаты выполнения тестов;
 - вывод.

Таблица 1

Типы исключений

№ теста	Тип исключения	Входные данные	Ожидаемая реакция	Фактическая реакция	Результат
1					
2					
3					
...					

Контрольные вопросы

1. Для чего используется конструкция try...catch?
2. Какие типы исключений существуют в .NET?
3. Что такое блок finally?
4. Как обработать несколько типов исключений?
5. Приведите пример деления на ноль.
6. Что такое исключительная ситуация?
7. Чем исключение отличается от ошибки?
8. Когда следует использовать блок finally?
9. Как получить информацию об исключении?
10. Можно ли выбросить собственное исключение?

Список литературы, рекомендуемый к использованию по данной теме:

1. Microsoft .NET Framework Documentation. - Microsoft, 2024.
2. Зубкова, Т.М. Технология разработки программного обеспечения: учебное пособие. – Оренбург: ОГУ, 2017.

ЛАБОРАТОРНАЯ РАБОТА № 10

НАПИСАНИЕ МОДУЛЬНЫХ ТЕСТОВ В VISUAL STUDIO (MSTEST/NUNIT)

Цель работы: Освоение технологий модульного (Unit) тестирования в среде Visual Studio.

Теоретическая часть**1. Модульный тест**

Модульный тест – это вспомогательный компонент программы, предназначенный для имитации её работы в целях тестирования. Модульный тест – это программный сценарий, который вызывает те или иные функции тестируемой программы и анализирует результаты их работы.

2. Структура класса модульного теста

```
```csharp
using Microsoft.VisualStudio.TestTools.UnitTesting;
```

```
[TestClass]
public class UnitTest1
{
 [TestMethod]
 public void TestMethod1()
 {
 // тело теста
 }
}
```

}  
...

### 3. Атрибуты

- [TestClass] – класс модульного теста.
- [TestMethod] – тестовый метод.

### 4. Достоинства модульного тестирования

1. Модульные тесты значительно оптимизируют процесс тестирования.
2. Более высокий уровень контроля качества по сравнению с ручным тестированием.
3. Решается проблема с описанием тестовых случаев.

### 5. Недостатки модульного тестирования

1. Модульные тесты сами являются программами и не защищены от возможных ошибок.
2. Написание модульных тестов требует времени.
3. Модульные тесты не способны полностью заменить ручное тестирование.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, Microsoft Visual Studio, а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

### Порядок выполнения работы:

1. Изучить теоретические сведения о модульном тестировании.
2. Создать новый проект «Проект модульного теста».
3. Добавить ссылку на тестируемый проект.
4. Написать тестовые методы.
5. Запустить тесты (Тест → Выполнить → Все тесты).
6. Проанализировать результаты.
7. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:

- титульный лист с темой лабораторной работы;
- цель работы;
- структура проекта модульного теста;
- листинг тестовых методов;
- результаты выполнения тестов (скриншоты);
- вывод.

Таблица 1

Результаты тестирования

№ теста	Имя теста	Ожидаемый результат	Фактический результат	Статус
1				
2				
3				
...				

### Контрольные вопросы

1. Что такое модульный тест?
2. Какие атрибуты используются для модульных тестов?
3. Как создать проект модульного теста в Visual Studio?
4. В чем достоинства модульного тестирования?
5. В чем недостатки модульного тестирования?
6. Что такое тестовый метод?
7. Что такое тестовый класс?
8. Как запустить все тесты?

9. Как интерпретировать результаты тестов?
10. Какое покрытие кода тестами считается оптимальным?

**Список литературы, рекомендуемый к использованию по данной теме:**

1. Microsoft Visual Studio Documentation. - Microsoft, 2024.
2. Зубкова, Т.М. Технология разработки программного обеспечения: учебное пособие. – Оренбург: ОГУ, 2017.

## **ЛАБОРАТОРНАЯ РАБОТА № 11**

### **РАЗРАБОТКА СЦЕНАРИЕВ СИСТЕМНОГО ТЕСТИРОВАНИЯ УЧЕБНОГО ПРИЛОЖЕНИЯ**

Цель работы: Освоение методов системного тестирования, разработка комплексных тестовых сценариев.

#### **Теоретическая часть**

##### **1. Системное тестирование**

Системное тестирование – выполнение ПО в его окончательной конфигурации, интегрированного с другими программными и аппаратными системами.

##### **2. Виды системного тестирования**

1. Функциональное тестирование.
2. Нагрузочное тестирование.
3. Тестирование безопасности.
4. Тестирование совместимости.
5. Тестирование производительности.

##### **3. Этапы системного тестирования**

1. Создание тестового набора путем ручной разработки или автоматической генерации.
2. Прогон программы на тестах с получением протокола тестирования.
3. Оценка результатов выполнения программы на наборе тестов.

##### **4. Комплексное тестирование**

Комплексное тестирование – процесс поиска несоответствия системы ее исходным целям. В процессе тестирования участвует система, описание целей продукта и вся документация, поставляемая вместе с системой.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, Microsoft Visual Studio, а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

#### **Порядок выполнения работы:**

1. Изучить теоретические сведения о системном тестировании.
2. Получить учебное приложение для тестирования.
3. Разработать тестовые сценарии для каждого вида тестирования.
4. Выполнить тестирование.
5. Зафиксировать результаты.
6. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:
  - титульный лист с темой лабораторной работы;
  - цель работы;
  - описание тестируемого приложения;

- тестовые сценарии;
- результаты выполнения тестов;
- вывод.

Таблица 1  
Тестовые сценарии

№ сценария	Название сценария	Предусловия	Шаги выполнения	Ожидаемый результат
1				
2				
3				
...				

Таблица 2  
Результаты тестирования

№ сценария	Фактический результат	Статус	Примечание
1			
2			
3			
...			

### Контрольные вопросы

1. Что такое системное тестирование?
2. Какие виды системного тестирования существуют?
3. Чем системное тестирование отличается от модульного?
4. Что такое приемочное тестирование?
5. Кто пишет приемочные тесты?
6. Что такое комплексное тестирование?
7. Какие этапы включает системное тестирование?
8. Что такое тестовый сценарий?
9. Как фиксировать результаты тестирования?
10. Что делать при обнаружении ошибки?

### Список литературы, рекомендуемый к использованию по данной теме:

1. Антамошкин, О.А. Программная инженерия. Теория и практика: учебник. – Красноярск: Сибирский федеральный университет, 2012.
2. Зубкова, Т.М. Технология разработки программного обеспечения: учебное пособие. – Оренбург: ОГУ, 2017.

## ЛАБОРАТОРНАЯ РАБОТА № 12 ОФОРМЛЕНИЕ ТЕСТОВОЙ ДОКУМЕНТАЦИИ ПО ШАБЛОНУ (ТЕСТ-ПЛАН, ТЕСТ-КЕЙС)

**Цель работы:** Освоение требований к документированию процесса тестирования.

### Теоретическая часть

1. Тестовый план

Тестовый план – это документ, или набор документов, который содержит тестовые ресурсы, перечень функций и подсистем, подлежащих тестированию, тестовую стратегию, расписание тестовых циклов, фиксацию тестовой конфигурации.

## 2. Тест-кейс

Тест-кейс – документ, описывающий набор шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции.

## 3. Структура тест-кейса

- Идентификатор.
- Название.
- Предусловия.
- Шаги выполнения.
- Ожидаемый результат.
- Фактический результат.
- Статус.

## 4. Стандарты документации

Тестирование имеет много общего с процессами верификации и валидации (V&V). Общность процесса тестирования с процессами V&V заключается в единстве состава и структуры планов, рекомендуемых стандартом IEEE Std. 829.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, Microsoft Word, а также программой MS Excel для подготовки таблиц.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

### Порядок выполнения работы:

1. Изучить теоретические сведения о тестовой документации.
2. Получить шаблон документов у преподавателя.
3. Разработать тест-план.
4. Оформить тест-кейсы (минимум 10).
5. Проверить соответствие стандартам (IEEE 829).
6. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:
  - титульный лист с темой лабораторной работы;
  - цель работы;
  - тест-план;
  - тест-кейсы;
  - отчет об ошибках (шаблон);
  - вывод.

Таблица 1

Структура тест-плана

Раздел	Описание
1. Введение	
2. Объем тестирования	
3. Стратегия тестирования	
4. Ресурсы	
5. Расписание	
6. Критерии начала/окончания	

Таблица 2

Структура тест-кейса

Поле	Значение
Идентификатор	

Название	
Предусловия	
Шаги выполнения	
Ожидаемый результат	
Фактический результат	
Статус	

### Контрольные вопросы

1. Что такое тестовый план?
2. Что такое тест-кейс?
3. Какие стандарты документации тестирования существуют?
4. Что должно входить в отчет об ошибках?
5. Зачем хранить тесты после выпуска программы?
6. Что такое IEEE Std. 829?
7. Какие разделы включает тест-план?
8. Какие поля включает тест-кейс?
9. Как приоритизировать тест-кейсы?
10. Как вести базу дефектов?

### Список литературы, рекомендуемый к использованию по данной теме:

1. IEEE Std. 829 - Standard for Software and System Test Documentation.
2. Антамошкин, О.А. Программная инженерия. Теория и практика: учебник. – Красноярск: Сибирский федеральный университет, 2012.

## ЛАБОРАТОРНАЯ РАБОТА № 13 ТЕСТИРОВАНИЕ ВЕРСТКИ И ИНТЕРФЕЙСА САЙТА. ПРОВЕРКА ФОРМ

**Цель работы:** Освоение методов тестирования веб-интерфейсов и пользовательских форм.

### Теоретическая часть

#### 1. Тестирование пользовательского интерфейса (GUI)

Графический интерфейс пользователя (Graphical user interface, GUI) – разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т. п.), представленные пользователю на дисплее, исполнены в виде графических изображений.

#### 2. Общие проверки

1. Вид элементов при уменьшении окна браузера + появление скрола.
2. Правильность написания текста + текст должен быть выровнен.
3. Правильность перемещения фокуса в окне (Tab/Tab+Shift).
4. Выбранные элементы выделяются.
5. Унификация дизайна (цвет, шрифт, размер).
3. Наиболее распространенные баги
  1. Курсор не переходит в поинтер при наведении на активный элемент.
  2. Орфографические и грамматические ошибки.
  3. Не ровное расположение полей ввода в формах.
  4. Неправильное отображение элементов при смене размера окна браузера.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, браузеры Chrome, Firefox, Edge, а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

### **Порядок выполнения работы:**

1. Изучить теоретические сведения о тестировании GUI.
2. Открыть тестируемый веб-сайт.
3. Проверить элементы интерфейса по чек-листу.
4. Проверить формы ввода данных.
5. Зафиксировать найденные дефекты.
6. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:
  - титульный лист с темой лабораторной работы;
  - цель работы;
  - чек-лист проверок;
  - список найденных дефектов;
  - скриншоты дефектов;
  - вывод.

Таблица 1

Чек-лист проверок

№	Элемент	Проверка	Результат (OK/FAIL)
1	Кнопки		
2	Поля ввода		
3	Меню		
4	Формы		
5	Текст		

Таблица 2

Найденные дефекты

№ дефекта	Описание	Приоритет	Скриншот
1			
2			
3			
...			

### **Контрольные вопросы**

1. Что такое GUI-тестирование?
2. Какие элементы интерфейса проверяются?
3. Что такое чек-лист тестирования?
4. Какие баги интерфейса наиболее распространены?
5. Как проверить адаптивность интерфейса?
6. Что такое унификация дизайна?
7. Как проверить перемещение фокуса?
8. Какие браузеры нужно проверять?
9. Что такое тултипы?
10. Как проверить орфографию на сайте?

### **Список литературы, рекомендуемый к использованию по данной теме:**

1. Руководство по обеспечению доступности веб-контента (WCAG).

2. Антамошкин, О.А. Программная инженерия. Теория и практика: учебник. – Красноярск: Сибирский федеральный университет, 2012.

## **ЛАБОРАТОРНАЯ РАБОТА № 14**

### **КРОССБРАУЗЕРНОЕ ТЕСТИРОВАНИЕ. ПРОВЕРКА СОВМЕСТИМОСТИ**

**Цель работы:** Освоение методов тестирования совместимости веб-приложений в различных браузерах.

#### **Теоретическая часть**

##### **1. Тестирование совместимости**

Тестирование совместимости – процесс оценки поведения приложения в различных браузерах, операционных системах, на устройствах с разным разрешением экрана.

##### **2. Проверяемые браузеры**

1. Chrome
2. Firefox
3. MS Edge
4. Safari

##### **3. Проверяемые ОС**

1. Windows 7, 8, 10
2. macOS
3. Linux

##### **4. Тестирование совместимости**

Проверка совместимости продукта со всеми последними версиями браузеров Chrome, Firefox, MS Edge, Safari и ОС Windows 7, 8 и 10 является примером данного вида тестирования.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, браузеры Chrome, Firefox, Edge, а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

#### **Порядок выполнения работы:**

1. Изучить теоретические сведения о тестировании совместимости.
2. Открыть тестируемый веб-сайт в разных браузерах.
3. Проверить функциональность в каждом браузере.
4. Зафиксировать различия и дефекты.
5. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:
  - титульный лист с темой лабораторной работы;
  - цель работы;
  - список проверяемых браузеров;
  - таблица результатов тестирования;
  - список найденных дефектов;
  - вывод.

Таблица 1

Результаты тестирования

Браузер	Версия	Функциональность	Визуальное отображение	Статус
Chrome				
Firefox				

Edge				
Safari				

Таблица 2  
Найденные дефекты

№ дефекта	Браузер	Описание дефекта	Приоритет
1			
2			
3			
...			

### Контрольные вопросы

1. Что такое тестирование совместимости?
2. Какие браузеры нужно проверять?
3. Почему сайт может по-разному работать в браузерах?
4. Какие инструменты для кроссбраузерного тестирования существуют?
5. Что такое адаптивный дизайн?
6. Как проверить совместимость с мобильными устройствами?
7. Что такое разрешение экрана?
8. Как эмулировать разные устройства?
9. Какие CSS-свойства могут вызывать проблемы?
10. Как проверить JavaScript в разных браузерах?

### Список литературы, рекомендуемый к использованию по данной теме:

1. Руководство по обеспечению доступности веб-контента (WCAG).
2. Антамошкин, О.А. Программная инженерия. Теория и практика: учебник. – Красноярск: Сибирский федеральный университет, 2012.

## ЛАБОРАТОРНАЯ РАБОТА № 15 ЗАМЕР СКОРОСТИ ЗАГРУЗКИ И ОТКЛИКА СИСТЕМЫ

**Цель работы:** Освоение методов тестирования производительности веб-приложений.

### Теоретическая часть

#### 1. Тестирование производительности

Тестирование производительности – комплекс проверок, направленный на определение лимитов производительности приложения.

#### 2. Инструменты

1. Google PageSpeed Insights
2. Яндекс.Вебмастер
3. Pingdom
4. Sitespeed.ru
5. GTmetrix
6. YSlow
7. WebPagetest

#### 3. Нормативы

Хорошая скорость загрузки страницы – 0.35–0.38 секунд. Такие результаты показывают сайты в топе выдачи.

Рейтинг PageSpeed определяется по 100-балльной шкале: чем больше баллов, тем лучше. Если ваш сайт получил более 85, значит, все хорошо.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, доступ к сети Интернет, а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

### **Порядок выполнения работы:**

1. Изучить теоретические сведения о тестировании производительности.
2. Выбрать 3-4 инструмента для тестирования.
3. Измерить скорость загрузки сайта.
4. Проанализировать отчеты.
5. Сформулировать рекомендации по оптимизации.
6. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:
  - титульный лист с темой лабораторной работы;
  - цель работы;
  - список использованных инструментов;
  - результаты замеров скорости;
  - рекомендации по оптимизации;
  - вывод.

Таблица 1

Результаты замеров

Инструмент	Время загрузки	Рейтинг	Рекомендации
Google PageSpeed			
Pingdom			
GTmetrix			
Sitespeed.ru			

### **Контрольные вопросы**

1. Что такое тестирование производительности?
2. Какие инструменты для замера скорости существуют?
3. Какая скорость загрузки считается хорошей?
4. Что влияет на скорость загрузки сайта?
5. Как оптимизировать скорость загрузки?
6. Что такое время ответа сервера?
7. Как измерить скорость на мобильных устройствах?
8. Что такое кэширование?
9. Как оптимизировать изображения?
10. Что такое CDN?

### **Список литературы, рекомендуемый к использованию по данной теме:**

1. Google PageSpeed Insights Documentation.
2. Антамошкин, О.А. Программная инженерия. Теория и практика: учебник. – Красноярск: Сибирский федеральный университет, 2012.

## **ЛАБОРАТОРНАЯ РАБОТА № 16 СТРЕСС-ТЕСТИРОВАНИЕ. МОДЕЛИРОВАНИЕ НАГРУЗКИ**

**Цель работы:** Освоение методов нагрузочного тестирования с использованием специализированных инструментов.

## Теоретическая часть

### 1. Нагрузочное тестирование

Нагрузочное тестирование – проверка работоспособности программного продукта при предельной входной нагрузке.

### 2. Инструменты

1. Apache JMeter

2. LoadRunner

3. Gatling

### 3. Типы нагрузочного тестирования

1. Исследование масштабируемости.

2. Консалтинг в области нагрузочного тестирования.

3. Инфраструктурная оптимизация.

### 4. Масштабируемость

Масштабируемость – это способность системы адаптироваться к расширению предъявляемых требований и росту объемов решаемых задач, наращивать производительность при увеличении нагрузки и добавлении аппаратных ресурсов.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, Apache JMeter, а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

### Порядок выполнения работы:

1. Изучить теоретические сведения о нагрузочном тестировании.

2. Установить и настроить Apache JMeter.

3. Создать тестовый план.

4. Настроить количество пользователей и длительность теста.

5. Запустить тест.

6. Проанализировать результаты.

7. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:

- титульный лист с темой лабораторной работы;
- цель работы;
- конфигурация тестового плана JMeter;
- результаты нагрузочного тестирования;
- график зависимости производительности от нагрузки;
- вывод.

Таблица 1

Параметры теста

Параметр	Значение
Количество пользователей	
Длительность теста	
Целевой URL	
Тип запроса	

Таблица 2

Результаты теста

Метрика	Значение
Среднее время отклика	
90-й перцентиль	

Количество ошибок	
Пропускная способность	

### **Контрольные вопросы**

1. Что такое нагрузочное тестирование?
2. Какие инструменты для нагрузочного тестирования существуют?
3. Что такое стресс-тестирование?
4. Как определить максимальное количество пользователей?
5. Что такое масштабируемость системы?
6. Что такое Apache JMeter?
7. Как создать тестовый план в JMeter?
8. Что такое пропускная способность?
9. Как интерпретировать результаты теста?
10. Что такое перцентиль?

### **Список литературы, рекомендуемый к использованию по данной теме:**

1. Apache JMeter Documentation.
2. Антамошкин, О.А. Программная инженерия. Теория и практика: учебник. – Красноярск: Сибирский федеральный университет, 2012.

## **ЛАБОРАТОРНАЯ РАБОТА № 17 ПОИСК УЯЗВИМОСТЕЙ В ТЕСТОВОЙ СРЕДЕ**

**Цель работы:** Освоение методов тестирования безопасности информационных систем.

### **Теоретическая часть**

#### 1. Тестирование безопасности

Тестирование безопасности – вид тестирования, используемый для проверки безопасности системы и анализа рисков.

#### 2. Принципы безопасности (CIA)

1. Конфиденциальность.

2. Целостность.

3. Доступность.

#### 3. Типы уязвимостей

1. SQL-инъекции.

2. XSS (Cross-Site Scripting).

3. CSRF (Cross-Site Request Forgery).

#### 4. Общая стратегия безопасности

Общая стратегия безопасности основывается на трех основных принципах: Конфиденциальность, Целостность, Доступность.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, доступ к тестовой среде, а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

### **Порядок выполнения работы:**

1. Изучить теоретические сведения о тестировании безопасности.
2. Получить доступ к тестовой среде.

3. Провести сканирование на уязвимости.
4. Попытаться эксплуатировать найденные уязвимости.
5. Зафиксировать результаты.
6. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:
  - титульный лист с темой лабораторной работы;
  - цель работы;
  - описание тестируемой системы;
  - список найденных уязвимостей;
  - рекомендации по устранению;
  - вывод.

Таблица 1

Найденные уязвимости

№	Тип уязвимости	Описание	Уровень риска	Рекомендации
1				
2				
3				
...				

### Контрольные вопросы

1. Что такое тестирование безопасности?
2. Какие принципы безопасности существуют?
3. Что такое SQL-инъекция?
4. Что такое XSS?
5. Как защитить приложение от уязвимостей?
6. Что такое CSRF?
7. Что такое конфиденциальность?
8. Что такое целостность?
9. Что такое доступность?
10. Как провести сканирование на уязвимости?

### Список литературы, рекомендуемый к использованию по данной теме:

1. OWASP Top 10 Documentation.
2. Антамошкин, О.А. Программная инженерия. Теория и практика: учебник. – Красноярск: Сибирский федеральный университет, 2012.

## ЛАБОРАТОРНАЯ РАБОТА № 18 ТЕСТИРОВАНИЕ АВТОРИЗАЦИИ И РАЗГРАНИЧЕНИЯ ПРАВ ДОСТУПА

**Цель работы:** Освоение методов тестирования системы авторизации и управления доступом.

### Теоретическая часть

1. Тестирование авторизации  
Тестирование авторизации – проверка корректности работы системы разграничения прав доступа.
2. Проверяемые аспекты
  1. Регистрация пользователей.
  2. Вход в систему.
  3. Восстановление пароля.

4. Разграничение ролей.
5. Доступ к защищенным ресурсам.
3. Принципы безопасности

Конфиденциальность – это сокрытие определенных ресурсов или информации. Под конфиденциальностью можно понимать ограничение доступа к ресурсу некоторой категории пользователей.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, доступ к тестируемому приложению, а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

### **Порядок выполнения работы:**

1. Изучить теоретические сведения об авторизации.
2. Получить доступ к тестируемому приложению.
3. Создать пользователей с разными ролями.
4. Протестировать сценарии доступа.
5. Зафиксировать нарушения.
6. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:
  - титульный лист с темой лабораторной работы;
  - цель работы;
  - описание ролей пользователей;
  - тестовые сценарии;
  - результаты тестирования;
  - вывод.

Таблица 1  
Роли пользователей

Роль	Права доступа	Описание
Администратор		
Пользователь		
Гость		

Таблица 2  
Результаты тестирования

Сценарий	Роль	Ожидаемый доступ	Фактический доступ	Результат

### **Контрольные вопросы**

1. Что такое авторизация?
2. Чем авторизация отличается от аутентификации?
3. Какие роли пользователей могут существовать?
4. Как тестировать разграничение прав?
5. Что такое несанкционированный доступ?
6. Что такое восстановление пароля?
7. Как проверить защиту ресурсов?
8. Что такое сессия пользователя?
9. Как тестировать выход из системы?
10. Что такое многофакторная аутентификация?

**Список литературы, рекомендуемый к использованию по данной теме:**

1. OWASP Authentication Cheat Sheet.
2. Антамошкин, О.А. Программная инженерия. Теория и практика: учебник. – Красноярск: Сибирский федеральный университет, 2012.

## **ЛАБОРАТОРНАЯ РАБОТА № 19**

### **НАПИСАНИЕ ПРОСТОГО АВТОТЕСТА (СКРИПТА) В SELENIUM**

**Цель работы:** Освоение основ автоматизации тестирования веб-приложений с использованием Selenium.

#### **Теоретическая часть**

##### **1. Selenium**

Selenium – популярный и востребованный бесплатный инструмент от компании OpenQA.org для тестирования web-приложений. Selenium поддерживается мобильными и десктопными браузерами, позволяет разрабатывать скрипты практически на любом языке программирования.

##### **2. Продукты Selenium**

1. Selenium IDE – дополнение к браузеру Firefox для записи, редактирования и отладки тестов.
2. Selenium WebDriver – набор библиотек для различных языков программирования.
3. Selenium Server – принимает команды удаленно и выполняет их в браузере.
4. Selenium Grid – масштабирующая сеть серверов.

##### **3. Поддерживаемые языки**

HTML, Java, C, Perl, PHP, Python, Ruby.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, Selenium WebDriver, браузер Chrome, а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

#### **Порядок выполнения работы:**

1. Изучить теоретические сведения о Selenium.
2. Установить Selenium WebDriver.
3. Написать тестовый скрипт.
4. Запустить тест.
5. Проанализировать результаты.
6. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:
  - титульный лист с темой лабораторной работы;
  - цель работы;
  - листинг тестового скрипта;
  - результаты выполнения теста;
  - скриншоты;
  - вывод.

Таблица 1

Параметры теста

Параметр	Значение
URL	
Браузер	
Язык скрипта	
Количество шагов	

### **Контрольные вопросы**

1. Что такое Selenium?
2. Какие компоненты Selenium существуют?
3. Какие языки программирования поддерживает Selenium?
4. В чем преимущества автоматизации тестирования?
5. Когда не стоит автоматизировать тесты?
6. Что такое Selenium WebDriver?
7. Что такое Selenium IDE?
8. Как установить Selenium?
9. Как запустить тест в Selenium?
10. Как интерпретировать результаты теста?

### **Список литературы, рекомендуемый к использованию по данной теме:**

1. Selenium Documentation. - OpenQA.org.
2. Зубкова, Т.М. Технология разработки программного обеспечения: учебное пособие. – Оренбург: ОГУ, 2017.

## **ЛАБОРАТОРНАЯ РАБОТА № 20 СОЗДАНИЕ UNIT-ТЕСТОВ ДЛЯ КЛАССА/МЕТОДА**

**Цель работы:** Закрепление навыков модульного тестирования с использованием фреймворков.

### **Теоретическая часть**

#### **1. Unit-тест**

Unit-тест – это программный сценарий, который вызывает те или иные функции тестируемой программы и анализирует результаты их работы.

#### **2. Фреймворки**

1. MSTest (Visual Studio).
2. NUnit.
3. xUnit.

#### **3. Структура теста**

1. Arrange (подготовка).
2. Act (действие).
3. Assert (проверка).

#### **4. Достоинства модульного тестирования**

1. Модульные тесты значительно оптимизируют процесс тестирования.
2. Более высокий уровень контроля качества по сравнению с ручным тестированием.
3. Решается проблема с описанием тестовых случаев.

**Оборудование и материалы:** для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, Microsoft Visual Studio, MSTest/NUnit, а также программой MS Word для подготовки отчета.

**Указания по технике безопасности:** к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

### **Порядок выполнения работы:**

1. Изучить теоретические сведения о Unit-тестах.
2. Создать проект модульных тестов.
3. Добавить ссылку на тестируемый проект.
4. Написать тестовые методы (минимум 5).
5. Запустить тесты.

6. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:

- титульный лист с темой лабораторной работы;
- цель работы;
- листинг тестируемого класса;
- листинг тестовых методов;
- результаты выполнения тестов;
- вывод.

Таблица 1

Результаты тестирования

№ теста	Имя метода	Ожидаемый результат	Фактический результат	Статус
1				
2				
3				
...				

### Контрольные вопросы

1. Что такое Unit-тест?
2. Какие фреймворки для модульного тестирования существуют?
3. Что такое паттерн AAA (Arrange-Act-Assert)?
4. Какое покрытие кода тестами считается оптимальным?
5. В чем недостатки модульного тестирования?
6. Что такое тестовый метод?
7. Что такое атрибут [TestMethod]?
8. Как добавить ссылку на проект?
9. Как запустить все тесты?
10. Как интерпретировать результаты?

### Список литературы, рекомендуемый к использованию по данной теме:

1. Microsoft Visual Studio Documentation. - Microsoft, 2024.
2. Зубкова, Т.М. Технология разработки программного обеспечения: учебное пособие. – Оренбург: ОГУ, 2017.

## ЛАБОРАТОРНАЯ РАБОТА № 21

### ПРОВЕДЕНИЕ РЕГРЕССИОННОГО ТЕСТИРОВАНИЯ ПОСЛЕ ИЗМЕНЕНИЙ В КОДЕ

**Цель работы:** Освоение методов регрессионного тестирования для проверки изменений в ПО.

#### Теоретическая часть

##### 1. Регрессионное тестирование

Регрессионное тестирование – проверка новой версии ПО на корректность функциональности, протестированной в предыдущей версии. Выполняется с регулярной частотой.

##### 2. Цели регрессионного тестирования

1. Проверить исправление ошибки.
2. Воспроизвести ошибку другим способом.
3. Протестировать последствия исправлений.
3. Главная проблема

Главная проблема регрессионного тестирования – выбор между полным и частичным перетестированием и пополнением тестовых наборов.

##### 4. Наведенная ошибка

В случае наведенной ошибки исправление в одном месте привело к ошибке в другом, что демонстрирует необходимость проведения полного перетестирования.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, Microsoft Visual Studio, а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

### **Порядок выполнения работы:**

1. Изучить теоретические сведения о регрессионном тестировании.
2. Получить версию ПО с изменениями.
3. Выполнить ранее созданные тесты.
4. Зафиксировать новые дефекты.
5. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:
  - титульный лист с темой лабораторной работы;
  - цель работы;
  - список изменений в ПО;
  - результаты регрессионного тестирования;
  - список новых дефектов;
  - вывод.

Таблица 1  
Изменения в ПО

№	Описание изменения	Дата	Автор
1			
2			
3			

Таблица 2  
Результаты тестирования

№ теста	Название теста	Результат до изменений	Результат после изменений	Статус
1				
2				
3				
...				

### **Контрольные вопросы**

1. Что такое регрессионное тестирование?
2. Когда выполняется регрессионное тестирование?
3. В чем главная проблема регрессионного тестирования?
4. Что такое наведенная ошибка?
5. Как выбрать тесты для регресса?
6. Чем регресс отличается от повторного тестирования?
7. Что такое тестовый набор?
8. Как документировать результаты регресса?
9. Как часто нужно проводить регрессионное тестирование?
10. Что такое автоматизация регресса?

Список литературы, рекомендуемый к использованию по данной теме:

1. Антамошкин, О.А. Программная инженерия. Теория и практика: учебник. – Красноярск: Сибирский федеральный университет, 2012.
2. Зубкова, Т.М. Технология разработки программного обеспечения: учебное пособие. – Оренбург: ОГУ, 2017.

## **ЛАБОРАТОРНАЯ РАБОТА № 22**

### **ВЫБОР ТЕСТОВ ДЛЯ РЕГРЕССА. АНАЛИЗ ВЛИЯНИЯ ИЗМЕНЕНИЙ**

**Цель работы:** Освоение методов анализа влияния изменений на систему и выбора тестов для регрессионного тестирования.

#### **Теоретическая часть**

##### **1. Анализ влияния изменений**

Анализ влияния изменений – определение компонентов системы, которые могут быть затронуты изменениями.

##### **2. Критерии выбора тестов**

1. Тесты для измененных компонентов.
2. Тесты для связанных компонентов.
3. Тесты для критической функциональности.

##### **3. Регрессионное тестирование**

Регрессионное тестирование обеспечивает организацию повторного тестирования с целью сокращения ресурсов, необходимых для обеспечения заданного уровня качества продукта.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, Microsoft Visual Studio, а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

#### **Порядок выполнения работы:**

1. Изучить теоретические сведения об анализе влияния.
2. Получить описание изменений.
3. Определить затронутые компоненты.
4. Выбрать тесты для регрессионного тестирования.
5. Обосновать выбор.
6. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:

- титульный лист с темой лабораторной работы;
- цель работы;
- описание изменений;
- анализ влияния изменений;
- список выбранных тестов;
- обоснование выбора;
- вывод.

Таблица 1  
Анализ влияния

Компонент	Изменен (да/нет)	Затронут (да/нет)	Приоритет тестирования
'''			
'''			
'''			

Таблица 2  
Выбранные тесты

№ теста	Название теста	Компонент	Обоснование выбора
1			
2			
3			
...			

### Контрольные вопросы

1. Что такое анализ влияния изменений?
2. Какие критерии выбора тестов для регресса существуют?
3. Почему полное перетестирование не всегда возможно?
4. Что такое критическая функциональность?
5. Как документировать выбор тестов?
6. Что такое связанный компонент?
7. Как определить затронутые компоненты?
8. Что такое приоритет тестирования?
9. Как оценить риск изменений?
10. Что такое матрица трассировки?

### Список литературы, рекомендуемый к использованию по данной теме:

1. Антамошкин, О.А. Программная инженерия. Теория и практика: учебник. – Красноярск: Сибирский федеральный университет, 2012.
2. Зубкова, Т.М. Технология разработки программного обеспечения: учебное пособие. – Оренбург: ОГУ, 2017.

## ЛАБОРАТОРНАЯ РАБОТА № 23 РАСЧЁТ МЕТРИК КАЧЕСТВА ПРОЕКТА. ПОКРЫТИЕ КОДА ТЕСТАМИ

**Цель работы:** Освоение методов оценки качества ПО с использованием метрик.

### Теоретическая часть

#### 1. Метрики качества ПО

1. Покрытие кода тестами (%).
2. Плотность дефектов (дефектов на 1000 строк кода).
3. Процент пройденных тестов.
4. Время обнаружения дефекта.

#### 2. Покрытие кода

1. Покрытие операторов.
2. Покрытие условий.
3. Покрытие путей.

3. Покрытие – это часть структуры программы, которая была охвачена тестированием, выраженная в процентах.

Существует несколько различных способов измерения покрытия:

- покрытие операторов – каждая ли строка исходного кода была выполнена и протестирована;
- покрытие условий – каждая ли точка решения была выполнена и протестирована;
- покрытие путей – все ли возможные пути через заданную часть кода были выполнены.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, Microsoft Visual Studio, а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

### **Порядок выполнения работы:**

1. Изучить теоретические сведения о метриках качества.
2. Получить проект для анализа.
3. Рассчитать метрики (используя инструменты).
4. Проанализировать результаты.
5. Сформулировать рекомендации.
6. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:
  - титульный лист с темой лабораторной работы;
  - цель работы;
  - описание проекта;
  - таблица метрик качества;
  - анализ результатов;
  - рекомендации;
  - вывод.

Таблица 1  
Метрики качества

Метрика	Значение	Норма	Статус
Покрытие кода		80%	
Плотность дефектов		< 5 на 1000 строк	
Процент пройденных тестов		95%	
Время обнаружения дефекта		< 24 часа	

### **Контрольные вопросы**

1. Что такое метрики качества ПО?
2. Как рассчитывается покрытие кода?
3. Что такое плотность дефектов?
4. Какие метрики наиболее важны?
5. Как использовать метрики для улучшения качества?
6. Что такое покрытие операторов?
7. Что такое покрытие условий?
8. Что такое покрытие путей?
9. Как интерпретировать результаты метрик?
10. Какие инструменты для расчета метрик существуют?

### **Список литературы, рекомендуемый к использованию по данной теме:**

1. Антамошкин, О.А. Программная инженерия. Теория и практика: учебник. – Красноярск: Сибирский федеральный университет, 2012.
2. Зубкова, Т.М. Технология разработки программного обеспечения: учебное пособие. – Оренбург: ОГУ, 2017.

# ИНТЕГРАЛЬНАЯ ОЦЕНКА ОТТЕСТИРОВАННОСТИ СИСТЕМЫ

**Цель работы:** Освоение методов интегральной оценки качества тестирования информационной системы.

## Теоретическая часть

### 1. Интегральная оценка

Интегральная оценка – комплексная оценка качества тестирования на основе множества метрик.

### 2. Компоненты оценки

1. Полнота тестирования.

2. Эффективность тестирования.

3. Покрытие требований.

4. Покрытие кода.

5. Качество тестовой документации.

### 3. Оценка оттестированности проекта

Оценка оттестированности проекта включает метрики и методику интегральной оценки.

Оборудование и материалы: для выполнения данной лабораторной работы необходим компьютер с установленной операционной системой Windows 7 или выше, Microsoft Visual Studio, а также программой MS Word для подготовки отчета.

Указания по технике безопасности: к выполнению лабораторных работ допускаются студенты, ознакомившиеся с правилами работы в лаборатории, прошедшие инструктаж безопасности.

## Порядок выполнения работы:

1. Изучить теоретические сведения об интегральной оценке.

2. Получить данные о тестировании.

3. Рассчитать интегральные показатели.

4. Сформировать итоговый отчет.

5. Оформить отчет. Отчет по лабораторной работе должен быть выполнен в редакторе MS Word и содержать:

- титульный лист с темой лабораторной работы;
- цель работы;
- исходные данные;
- расчет интегральных показателей;
- итоговый отчет о тестировании;
- вывод.

Таблица 1

Интегральные показатели

Показатель	Вес	Значение	Взвешенное значение
Полнота тестирования	0.3		
Эффективность тестирования	0.2		
Покрытие требований	0.2		
Покрытие кода	0.2		
Качество документации	0.1		
<b>**Итого**</b>	<b>**1.0**</b>		

## Контрольные вопросы

1. Что такое интегральная оценка тестирования?

2. Какие компоненты входят в оценку?

3. Как рассчитать полноту тестирования?

4. Что такое покрытие требований?

5. Как представить результаты оценки заказчику?

6. Что такое вес показателя?
7. Как рассчитать взвешенное значение?
8. Что такое эффективность тестирования?
9. Как оценить качество документации?
10. Как интерпретировать интегральный показатель?

**Список литературы, рекомендуемый к использованию по данной теме:**

1. Антамошкин, О.А. Программная инженерия. Теория и практика: учебник. – Красноярск: Сибирский федеральный университет, 2012.
2. Зубкова, Т.М. Технология разработки программного обеспечения: учебное пособие. – Оренбург: ОГУ, 2017.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Антамошкин, О.А. Программная инженерия. Теория и практика: учебник / О.А. Антамошкин. – Красноярск: Сибирский федеральный университет, 2012.
2. Зубкова, Т.М. Технология разработки программного обеспечения: учебное пособие / Т.М. Зубкова. – Оренбург: ОГУ, 2017.
3. Майерс, Г. Искусство тестирования программ. – М.: Финансы и статистика, 2001.
4. Луиза Тамре. Введение в тестирование программного обеспечения. - М.: Издательский дом «Вильямс», 2013.
5. Лекционные материалы курса «Тестирование информационных систем».
6. Стандарты IEEE Std. 829, IEEE Std. 1012, ISO/IEC 12207.
7. Руководство по обеспечению доступности веб-контента (WCAG).
8. Документация инструментов: Visual Studio, Selenium, JMeter.
9. Microsoft Visual Studio Documentation. - Microsoft, 2024.
10. OWASP Top 10 Documentation.
11. Apache JMeter Documentation.
12. Selenium Documentation. - OpenQA.org.
13. Google PageSpeed Insights Documentation.
14. 1С-Пабблишинг. 1С:Предприятие 8.3. Руководство разработчика [Электронный ресурс]. – Москва: 1С-Пабблишинг, 2024.