МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к выполнению лабораторных работ по дисциплине
"Технологии тестирования программных средств"
и "Технологии тестирования информационных
систем".

Воронеж 2018

Составитель: канд. техн. наук Э.И. Воробьев
УДК 681.3.
Методические указания по выполнению лабораторных работ по курсу "Технологии тестирования программных средств" для студентов направления 09.03.01 "Информатика и вычислительная техника" и "Технологии тестирования информационных систем" для студентов направления 09.03.02 "Информационные системы и технологии" дневной формы обучения / Воронеж. гос. техн. ун-т.; Сост. Э. И. Воробьев. Воронеж, 2018.

Настоящие методические указания посвящены рассмотрению основных методов к разработке тестовых вариантов для программных систем и порядка проведения тестирования. Предназначено для студентов 3-4 курса

Рецензент
Ответственный за выпуск зав. кафедрой Я. Е. Львович

Печатается по решению редакционно-издательского совета Воронежского государственного технического университета.

ЛАБОРАТОРНАЯ РАБОТА №1

Виды тестирования. Планирование тестирования

Цель работы: изучить классификацию видов тестирования, практически закрепить эти знания путем генерации тестов различных видов, научиться планировать тестовые активности в зависимости от специфики поставляемой на тестирование функциональности.

Теоретические сведения

Тестирование – процесс, направленный на оценку корректности, полноты и качества разработанного программного обеспечения.

Тестирование можно классифицировать по очень большому количеству признаков. Далее приведен обобщенный список видов тестирования по различным основаниям.

Типы тестов по покрытию (по глубине)

Smoke test **–** тестирование системы для определения корректной работы базовых функций программы в целом, без углубления в детали. При проведении теста определяется пригодность сборки для дальнейшего тестирования.

Minimal Acceptance Test (MAT, Positive test*):* тестирование системы или ее части только на валидных данных (валидные данные – это данные, которые необходимо использовать для корректной работы модуля/функции). При тестировании проверяется правильной работы всех функций и модулей с валидными данными.

Для крупных и сложных приложений используется ограниченный набор сценариев и функций.

Acceptance Test (AT): полное тестирование системы или ее части как на корректных, так и на некорректных данных/сценариях. Вид теста, направленный на подтверждение того, что приложение может использоваться по назначению при любых условиях.

Тест на этом уровне покрывает все возможные сценарии тестирования: проверку работоспособности модулей при вводе корректных значений; проверку при вводе некорректных значений; использование форматов данных отличных от тех, которые указаны в требованиях; проверку исключительных ситуаций, сообщений об ошибках; тестирование на различных комбинациях входных параметров; проверку всех классов эквивалентности; тестирование граничных значений интервалов; сценарии не предусмотренные спецификацией и т.д.

Тестовые активности (типы тестов по покрытию (по ширине)):

Defect Validation **–** проверка результата исправления дефектов. Включает в себя проверку на воспроизводимость дефектов, которые были

исправлены в новой сборке продукта, а также проверку того, что исправление не повлияло на ранее работавшую функциональность

New Feature Test (NFT, AT of NF) – определение качества поставленной на тестирование новой функциональности, которая ранее не тестировалась. Данный тип тестирования включает в себя: проведение полного теста (АТ) непосредственно новой функциональности; тестирование новой функциональности на соответствие документации; проверку всевозможных взаимодействий ранее реализованной функциональности с новыми модулями и функциями.

Regression testing (регрессионное тестирование) **–** проводится с целью оценки качества ранее реализованной функциональности. Включает в себя проверку стабильности ранее реализованной функциональности после внесения изменений, например добавления новой функциональности, исправление дефектов, оптимизация кода, разворачивание приложения на новом окружении. Регрессионное тестирование может быть проведено на уровне Smoke, MAT или AT.

Типы тестов по знанию коду

Черный ящик – тестирование системы, функциональное или нефункциональное, без знания внутренней структуры и компонентов системы. У тестировщика нет доступа к внутренней структуре и коду приложения либо в процессе тестирования он не обращается к ним.

Белый ящик – тестирование основанное на анализе внутренней структуры компонентов или системы. У тестировщика есть доступ к внутренней структуре и коду приложения.

Серый ящик – комбинация методов белого и черного ящика, состоящая в том, что к части кода архитектуры у тестировщика есть, а к части кода – нет.

Типы тестов по степени автоматизации

Ручное – тестирование, в котором тест-кейсы выполняются тестировщиком вручную без использования средств автоматизации.

Автоматизированное – набор техник, подходов и инструментальных средств, позволяющий исключить человека из выполнения некоторых задач в процессе тестирования. Тест-кейсы частично или полностью выполняет специальное инструментальное средство.

Типы тестов по изолированности компонентов

Unit/component (модульное) – тестирование отдельных компонентов (модулей) программного обеспечения.

Integration (интеграционное) – тестируется взаимодействие между интегрированными компонентами или системами.

System (системное) – тестируется работоспособность системы в целом с целью проверки того, что она соответствуе установленным требованиям.

Типы тестов по подготовленности.

Интуитивное тестирование выполняется без подготовки к тестам, без определения ожидаемых результатов, проектирования тестовых сценариев.

Исследовательское тестирование – метод проектирования тестовых сценариев во время выполнения этих сценариев. Тестировщик совершает проверки, продумывает их, придумывает новые проверки, часто использует для этого полученную информацию.

Тестирование по документации – тестирование по подготовленным тестовым сценариям, руководству по осуществлению тестов.

Титы тестов по месту и времени проведения

User Acceptance Testing (UAT) (приемочное тестирование) – формальное тестирование по отношению к потребностям, требованиям и бизнес процессам пользователя, проводимое с целью определения соответствия системы критериям приёмки и дать возможность пользователям, заказчикам или иным авторизованным лицам определить, принимать систему.

Alpha Testing (альфа-тестирование) **–** моделируемое или действительное функциональное тестирование, выполняется в организации, разрабатывающей продукт, но не проектной командой (это может быть независимая команда тестировщиков, потенциальные пользователи, заказчики). Альфа тестирование часто применяется к коробочному программному обеспечению в качестве внутреннего приемочного тестирования.

Beta Testing (бета-тестирование) **–** эксплуатационное тестирование потенциальными или существующими клиентами/заказчиками на внешней стороне (в среде, где продукт будет использоваться) никак связанными с разработчиками, с целью определения действительно ли компонент или система удовлетворяет требованиям клиента/заказчика и вписывается в бизнес-процессы. Бета-тестирование часто проводится как форма внешнего приемочного тестирования готового программного обеспечения для того, чтобы получить отзывы рынка.

Типы тестов по объекту тестирования

Functional testing (функциональное тестирование) – это тестирование, основанное на анализе спецификации, функциональности компонента или системы. Функциональным можно назвать любой вид тестирования, который согласно требованиям проверяет правильную работу.

Safety testing (тестирование безопасности) – тестирование программного продукта с целью определить его безопасность (безопасность – способность программного продукта при использовании оговоренным образом оставаться в рамках приемлемого риска причинения вреда здоровью, бизнесу, программам, собственности или окружающей среде.

***Security testing (тестирование защищенности*)** – это тестирование с целью оценить защищенность программного продукта. Тестирование защищенности проверяет фактическую реакцию защитных механизмов, встроенных в систему, на проникновение.

Compatibility testing (тестирование совместимости) – процесс тестирования для определения возможности взаимодействия программного продукта, проверка работоспособности приложения в различных средах (браузеры и их версии, операционные системы, их типа, версии и разрядность)

Виды тестов:

 кросс-браузерное тестирование (различные браузеры или версии браузеров)

 кросс-платформенное тестирование (различные операционные системы или версии операционных систем)

Нефункциональное тестирование – это проверка характеристик программы. Иначе говоря, когда проверяется не именно правильность работы, а какие-либо свойства (внешний вид и удобство пользования, скорость работы и т.п.).

1. Тестирование пользовательского интерфейса (GUI) – тестирование, выполняемое путем взаимодействия с системой через графический интерфейс пользователя.

- навигация
- цвета, графика, оформление
- содержание выводимой информации
- поведение курсора и горячие клавиши
- отображение различного количества данных (нет данных, минимальное и максимальное количество)
- изменение размеров окна или разрешения экрана

2. Тестирование удобства использования (Usability Testing) – тестирование с целью определения степени понятности, легкости в изучении и использовании, привлекательности программного продукта для пользователя при условии использования в заданных условиях эксплуатации.

- визуальное оформление
- навигация
- логичность

3. Тестирование доступности (Accessibility testing) – тестирование, которое определяет степень легкости, с которой пользователи с ограниченными способностями могут использовать систему или ее компоненты.

4. Тестирование интернационализации – тестирование способности продукта работать в локализованных средах (способность изменять элементы интерфейса в зависимости от длины и направления текста, менять сортировки/форматы под различные локали и т.д.). (Максим Черняк).

Интернационализация – это процесс, упрощающий дальнейшую

адаптацию продукта к языковым и культурным особенностям региона, отличного от того, в котором разрабатывался продукт. Это адаптация продукта для потенциального использования практически в любом месте, Интернационализация производится на начальных этапах разработки, в то время как локализация — для каждого целевого языка.

5. Тестирование локализации (Localization testing) – тестирование, проводимое с целью проверить качество перевода продукта с одного языка на другой.

6. Тестирование производительности или нагрузочное тестирование – процесс тестирования с целью определения производительности программного продукта.

Виды тестов:

- нагрузочное тестирование (Performance and Load testing) – вид тестирования производительности, проводимый с целью оценки поведения компонента или системы при возрастающей нагрузке, например количестве параллельных пользователей и/или операций, а также определения какую нагрузку может выдержать компонент или система;
- объемное тестирование (Volume testing) – позволяет получить оценку производительности при увеличении объемов данных в базе данных приложения;
- тестирование стабильности и надежности (Stability / Reliability testing) – позволяет проверять работоспособность приложения при длительном (многочасовом) тестировании со средним уровнем нагрузки.
- стрессовое тестирование (Stress testing) – вид тестирования производительности, оценивающий систему или компонент на граничных значениях рабочих нагрузок или за их пределами, или же в состоянии ограниченных ресурсов, таких как память или доступ к серверу.

7. Тестирование требований (Requirements testing) – проверка требований на соответствие основным характеристикам качества.

8. Тестирование прототипа (Prototyte testing) – метод выявления структурных, логических ошибок и ошибок проектирования на ранней стадии развития продукта до начала фактической разработки.

9. Тестирование установки (Installability testing) и лицензирования – процесс тестирования устанавливаемости программного продукта.

Виды тестов:

- формальный тест программы установки приложения (проверка пользовательского интерфейса, навигации, удобства пользования, соответствия общепринятым стандартам оформления);
- функциональный тест программы установки;
- тестирование механизма лицензирования и функций защиты от пиратства;
- проверка стабильности приложения после установки.

10. Тестирование на отказ и восстановление (Failover and Recovery Testing) – тестирование при помощи эмуляции отказов системы или реально вызываемых отказов в управляемом окружении.

Тестирование программного продукта включает следующие этапы:

1. Изучение и анализ предмета тестирования.
2. Планирование тестирования.
3. Исполнение тестирования.

Изучение и анализ предмета тестирования начинается еще до утверждения спецификации и продолжается на стадии разработки (кодирования) программного обеспечения. Конечной целью этапа изучение и анализ предмета тестирования является получение ответов на два вопроса:

- какие функциональности предстоит протестировать,
- как эти функциональности работают.

Планирование тестирования происходит на стадии разработки (кодирования) программного обеспечения. На стадии планирования тестирования перед тестировщиком стоит задача поиска компромисса между объемом тестирования, который возможен в теории, и объемом тестирования, который возможен на практике. На данной стадии необходимо ответить на вопрос: как будем тестировать? Результатом планирования тестирования является тестовая документация.

Выполнение тестирования происходит на стадии тестирования и представляет собой практический поиск дефектов с использованием тестовой документации, составленной ранее.

Для всех программных продуктов выполняют следующие типы тестов и их композиции.

Для первого билда рекомендуется проводить Smoke+AT готовой функциональности: поверхностное тестирование (Smoke Test) выполняется для определения пригодности сборки для дальнейшего тестирования; полное тестирование системы или ее части как на корректных, так и на некорректных данных/сценариях (Acceptance Test, AT) позволяет обнаружить дефекты и внести запись о них в багтрэкинговую систему.

Для последующих билдов композиции тестов могут быть следующими:

- Если не была добавлена новая функциональность, то: DV+MAT. Т.е., выполняется проверка исправления дефектов программистом (Defect Validation, DV), а также проверка работоспособности остальной функциональности после исправления дефектов на позитивных сценариях (Minimal Acceptance Test, MAT).

- Если была добавлена новая функциональность, то: Smoke+DV+NFT+Regression Test. В частности, выполняется поверхностное тестирование (Smoke Test), проверка исправления дефектов программистом (Defect Validation, DV), тестирование новых функциональностей (New Feature Testing, NFT), проверка старых функциональностей, т.е. регрессионное тестирование (Regression Test).

- Если была добавлена новая функциональность, то возможен также вариант: DV+NFT+Resression test, т.е. без выполнения Smoke Test.

В зависимости от типа и специфики приложения (web, desktop, mobile) выполняют специализированные тесты (например, кроссбраузерное или кроссплатформенное тестирование, тестирование локализации и интернационализации и др.).

Порядок выполнения работы

1. Получить задание у преподавателя.

2. Выполнить генерацию тестов различных видов для конкретного объекта реального мира (пример приведен на рисунке 1).

3. Спланировать тестовые активности для следующих задач:

3.1 Поставлен на тестирование модуль 1, модуль 2, модуль 3.

3.2 Проведены исправления (fix) для заведенных дефектов, доставлена новая функциональность – модуль 4.

3.3 Заказчик решил расширять рынки сбыта и просит осуществить поддержку для Великобритании (кроме уже существующей Беларуси).

3.4 Заказчик хочет убедиться, что ПО держит нагрузку в 2000 пользователей.

4. Оформить отчет и защитить лабораторную работу.

Пример выполнения лабораторной работы

Необходимо составить тестовый план для объекта «Карандаш».

Пример тестового плана для объекта карандаш представлен на рисунке 1.1.

Рисунок 1.1 – Пример генерации тестов различных видов для объекта «Карандаш»

Содержание отчета

1. Цель работы.
2. Краткие теоретические сведения.
3. Сгенерированные тесты различных видов для выбранного объекта реального мира.
4. Тестовые активности для сформулированных задач.
5. Выводы по работе.

Контрольные вопросы

1. Что такое тестирование?
2. Какие существуют типы тестов по покрытию? Дайте характеристику каждому.
3. Какие существуют тестовые активности? Дайте характеристику каждому.
4. Какие существуют типы тестов знанию кода? Дайте характеристику каждому.
5. Какие существуют типы тестов по степени автоматизации? Дайте характеристику каждому.
6. Какие существуют типы тестов по изолированности компонентов? Дайте характеристику каждому.
7. Какие существуют типы тестов по подготовленности? Дайте характеристику каждому.
8. Какие существуют типы тестов по месту и времени проведения? Дайте характеристику каждому.
9. Какие существуют типы тестов по объекту тестирования? Дайте характеристику каждому.
10. Какие существуют типы функциональных тестов? Дайте характеристику каждому.
11. Какие существуют типы нефункциональных тестов? Дайте характеристику каждому.
12. Какие этапы составляют процесс тестирования?
13. Что происходит на этапе изучения и анализа предмета тестирования?
14. Что происходит на этапе планирования тестирования?
15. Что происходит на этапе исполнения тестирования?
16. Какие типы тестов выполняют для первой поставки программного продукта?
17. Какие типы тестов выполняют для последующих поставок программного продукта?

Лабораторная работа № 2.
Составление плана тестирования

1. Цель работы: изучить элементы плана тестирования; приобрести навыки разработки плана тестирования.

1.1 Краткие теоретические сведения

Тест-план, или план тестирования (test plan), – часть проектной документации, описывающая и регламентирующая процесс тестирования.

1.1.1 Ключевые секции тест-плана.

1.1.1.1 Перечень работ.

Эта секция включает перечень функциональных областей приложений, которые будут подвергаться тестированию. Здесь же может быть перечень компонентов или функциональности, которые не будут тестироваться по тем или иным причинам.

1.1.1.2 Критерии качества.

Одна из наиболее важных секций, содержащая перечень критериев, по которым оценивается текущее и финальное качество продукта.

1.1.1.3 Оценка рисков.

Риск (risk) – сочетание вероятности наступления события и последствий, вызванных этим событием. В секции приводится перечень рисков, которые могут (вполне вероятно) возникнуть в процессе работы с проектом. По каждому риску даётся оценка представляемой им угрозы и приводятся варианты выхода из ситуации.

1.1.1.4 Документация.

Здесь приводится полный перечень используемой документации, а также указывается, кто и когда должен её готовить, кому передавать и т.п.

1.1.1.5 Стратегия тестирования.

Стратегия тестирования подразумевает описание процесса тестирования с точки зрения применяемых методов, подходов, инструментальных средств и т. п.

1.1.1.6 Ресурсы.

В данной секции перечисляются необходимые для успешного проведения тестирования ресурсы:

– программные;
– аппаратные;
– человеческие;
– временные;
– финансовые.

1.1.7 Метрики.

Секция, в которой приводятся числовые характеристики показателей качества, способы их оценки, формулы и т. п.

1.1.1.8 Расписание и ключевые точки.

Самая простая для понимания секция. Фактически это календарь, в котором указано, что и к какому моменту должно быть сделано.

Задание

1. Приведите по пять примеров тестов для графического редактора Photoshop, которые относились бы:

а) к уровню Smoke Test;

б) к уровню Critical Path Test;

в) к уровню Extended Test.

2. Приведите пример риска, который может быть отмечен при планировании тестовых испытаний. Дайте рекомендации по недопущению такой ситуации и выходу из неё в случае возникновения.

3. Перечислите основные секции тестового плана и дайте краткое пояснение того, что размещается в каждой из них.

Баллы суммируются по всем пунктам (максимальный балл равен 5) в соответствии с таблицей 1.1.

Таблица 1.1 Критерий оценки выполнения заданий

Что оценивается	Количество правильных ответов	Количество баллов
1а. Приведено правильных примеров	2 и менее	0
	3 и более	1
1б. Приведено правильных примеров	2 и менее	0
	3 и более	1
1в. Приведено правильных примеров	2 и менее	0
	3 и более	1
2 Пример риска	Не приведён	0
	Приведён, но без рекомендаций	0,5
	Приведён с рекомендациями	1
3 Перечислено секций тестового плана	3 и менее	0
	4 и более без пояснений	0,5
	4 и более с пояснениями	1

Контрольные вопросы

1 Что подлежит тестированию?

2 Что такое тест-план?

3 Как определить критерии качества?

4 Что такое «риски» и как они оцениваются?

Лабораторная работа № 3. Проектирование тест-кейсов

Цель работы: приобрести практические навыки создания тестов и тест-кейсов; научиться создавать тест-кейсы для приложений.

Задание:

На основе представленного ниже набора требований сформируйте для разрабатываемых приложений (на любом языке программирования):

– смоук-тест (оформить в виде таблицы excel);

– чек-лист для теста критического пути (оформить в виде таблицы excel);

– тест-кейс (оформить по приведенному примеру).

1. Краткие теоретические сведения.

1.1 **Smoke testing** (встречаются названия intake test , build verification test) — тестирование, проводимое на начальном этапе (например после нового билда) и в первую очередь направленное на проверку готовности разработанного продукта к проведению более расширенного тестирования, определения общего состояния качества продукта.

Это короткий цикл тестов, подтверждающий (отрицающий) факт того, что приложение стартует и выполняет свои основные функции. Данный тип тестирования позволяет на начальном этапе выявить основные быстро находимые критические дефекты. Исходя из того, что данные проверки практически всегда одинаковы и редко претерпевают изменениям, целесообразно будет их автоматизировать.

Стоит понимать, что данный тип тестирования является видом тестирования продукта по глубине, а не просто видом тестовых испытаний. Как говорилось выше, данный тип тестирования определяет, пригоден ли продукт для дальнейшего, более полного тестирования.

В случае, если он не проходит smoke testing — продукт необходимо отправить на доработку. Обязательно необходимо записывать результаты прохождения теста. Это необходимо для того, чтобы сохранить записи того, что работает, а что нет. Можно разделить результаты на пройдено и провалено (достаточно таблицы excel).

1.2 **Чек-лист для критического пути (**critical path test) — основной тип тестовых испытаний, во время которого значимые элементы и функции приложения проверяются на предмет правильности работы при стандартном их использовании.

Чаще всего на практике на данном уровне тестирования проверяется основная масса требований к продукту. Пример: возможность набора текста, вставки картинок, возможность войти на сайт, создать запись, и т.д.

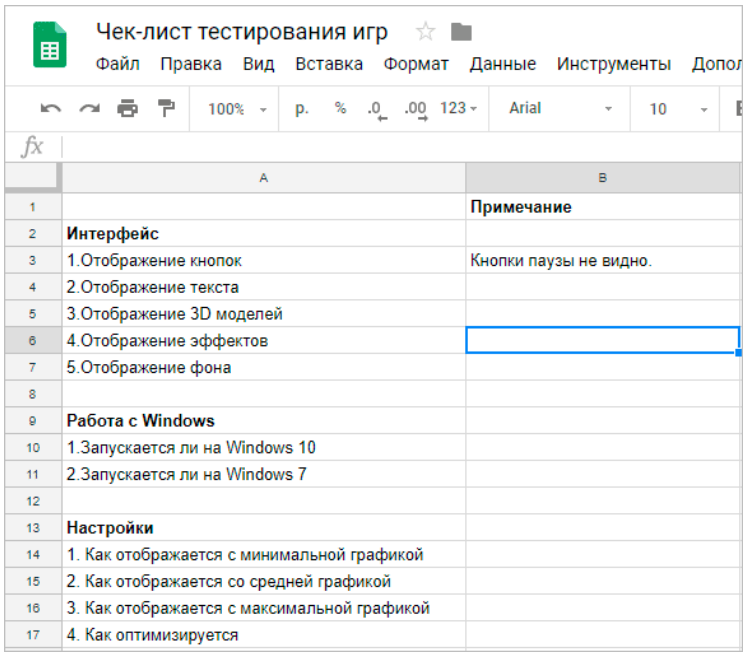
Для данного вида тестирования пишутся наиболее подробные и глубокие тест-кейсы, чтобы покрыть всю возможную функциональность приложения.

Тест критического пути может быть, как позитивным, так и негативным:

- Позитивный тест критического пути — это проверка работоспособности функций программного продукта, с которыми пользователь сталкивается ежедневно.
- Негативный тест критического пути — это проверка всевозможных вариантов нестандартного использования функциональности, используемой пользователем каждый день.

Тест критического пути является одним из самых распространенных видов функционального тестирования. А также позволяет выявить самые быстро находимые дефекты и исправить приложение в более сжатые сроки.

Возможный пример оформления приведен ниже (рисунок 1).



	A	B
1		**Примечание**
2	**Интерфейс**	
3	1.Отображение кнопок	Кнопки паузы не видно.
4	2.Отображение текста	
5	3.Отображение 3D моделей	
6	4.Отображение эффектов	
7	5.Отображение фона	
8		
9	**Работа с Windows**	
10	1.Запускается ли на Windows 10	
11	2.Запускается ли на Windows 7	
12		
13	**Настройки**	
14	1. Как отображается с минимальной графикой	
15	2. Как отображается со средней графикой	
16	3. Как отображается с максимальной графикой	
17	4. Как оптимизируется	

Рисунок 1 – Оформление чек-листа

2. Оформление тест-кейсов.

Тест-кейс – это профессиональная документация тестировщика, последовательность действий направленная на проверку какого-либо функционала, описывающая как прийти к фактическому результату.

Набор тест-кейсов называют тест-комплектом. Иногда тест-набор путают с тест-планом. Тест-план описывает какие работы, как и когда должны быть проведены в рамках тестирования продукта, а так же что необходимо для их выполнения.

Любой тест-кейс обязательно включает в себя:

- Уникальный идентификатор тест-кейса – необходим для удобной организации хранения и навигации по нашим тест-наборам.

- Название – основная тема, или идея тест-кейса. Кратное описание его сути.

- Предусловия – описание условий, которые не имеют прямого отношения к проверяемому функционалу, но должны быть выполнены. Например, оставить комментарий на вашем портале может только зарегистрированный пользователь. Значит для тест-кейса «Создание комментария» будет необходимо выполнение предусловия «пользователь зарегистрирован», и «пользователь авторизован»

- Шаги – описание последовательности действий, которая должна привести нас к ожидаемому результату

- Ожидаемый результат – результат: что мы ожидаем увидеть после выполнения шагов.

Чего не должно быть в тест-кейсе:

1.Зависимостей от других тест-кейсов;

2. Нечеткой формулировки шагов или ожидаемого результата;

3. Отсутствия необходимой для прохождения тест-кейса информации;

4. Излишней детализации.

Возможный план по разработке тест-кейсов приведены на рисунке 2.

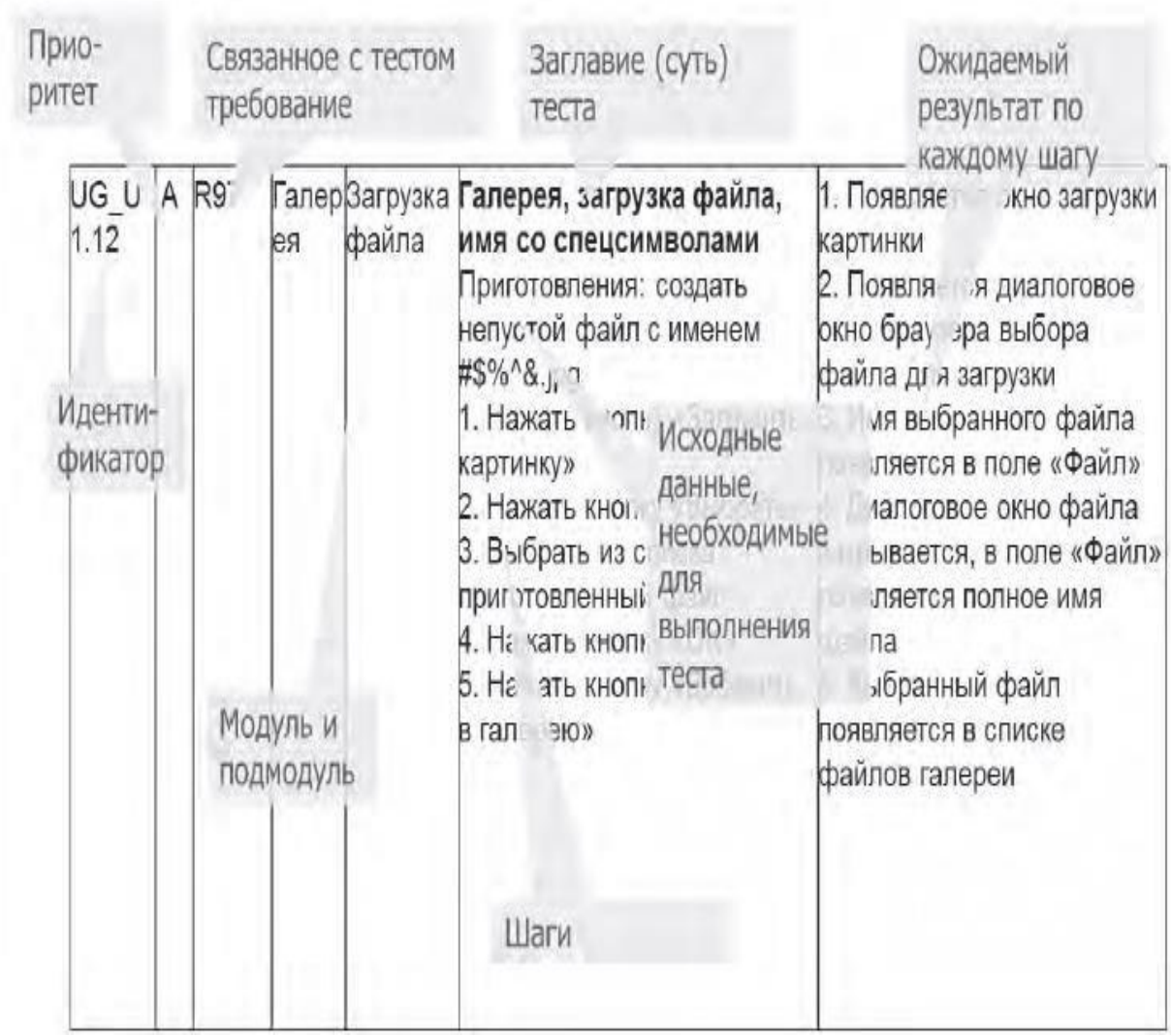


Рисунок 2 – Элементы тест-кейсов

Требования к разрабатываемому приложению 1.

Приложение должно выполнять математические вычисления.

1 Приложение должно работать под всеми версиями ОС Windows.

2 Приложение должно быть максимально похоже на стандартный калькулятор Windows (рисунок 3) за исключением некоторых особенностей.

3 Несколько приложений должны иметь возможность работать одновременно.

4 При запуске приложения должно отображаться окно со стандартными для калькулятора кнопками и полем ввода и отображения данных.

5 Для начала вычислений пользователь должен нажать кнопку «Начать».

6 Приложение должно позволять легко сохранять вычисления в выбранном пользователем формате.

7 Опционально предусматривается поддержка нескольких языков.

8 Приложение должно позволять выполнять вычисления сразу же после запуска.

9 Скорость вычислений должна быть максимально высокой.

10 Приложение должно позволять выполнять следующие операции: сложение, умножение, вычитание и деление чисел.

11 Приложение должно позволять строит графики простых функций.

12 Приложение должно запрашивать подтверждение («Результат не сохранён. Выйти?») в случае, если пользователь не сохранил результаты работы.

Требования к разрабатываемому приложению 2.

1 Приложение должно работать под версиями ОС Windows7 и Windows 8.

2 Несколько приложений должны иметь возможность работать одновременно, т. е. можно открыть несколько калькуляторов и вести в них невзаимо-связанные вычисления.

3 При запуске приложения должно отображаться окно с кнопками калькулятора (рисунок 3) и полем отображения данных.

4 Данные в приложение могут вводиться как с помощью кнопок приложения, так и с помощью клавиатуры.

5 Приложение должно позволять сохранять вычисления во внешний файл с расширением, задаваемым пользователем.

6 Должна быть предусмотрена поддержка английского и русского языков. Отображается тот язык, который выбран в ОС по умолчанию.

7 Вычисления должны производиться со скоростью не более 1 с.

8 Приложение должно позволять выполнять следующие операции: сложение, умножение, вычитание и деление чисел, взятие квадратного корня, возведение в степень, вычисление процентов, ввод отрицательного числа.

Рисунок 3 – Стандартный калькулятор Windows

Контрольные вопросы

1 В какой последовательности рекомендуется разрабатывать тесты?

2 Что такое смоук-тест? Приведите пример.

3 Что такое чек-лист?

4 Перечислите элементы тест-кейса.

5 Обязательно ли описывать ожидаемый результат в тест-кейсе и в чек-листе?

Лабораторная работа №4.
Автоматизация тестирования.

Цель работы: приобрести практические навыки проведения автоматизированного тестирования и использования программ для автоматизированного тестирования на примере программы Selenium IDE.

Теоретические сведения

Автоматизация тестирования (test automation) – набор техник, подходов и инструментальных средств, позволяющий исключить человека из выполнения некоторых задач в процессе тестирования.

Преимущества автоматизации тестирования.

1) Повторяемость – все написанные тесты всегда будут выполняться однообразно, т.е. исключен «человеческий фактор». Тестировщик не пропустит тест по неосторожности и ничего не напутает в результатах.

2) Быстрое выполнения – Быстрое выполнение – автоматизированному скрипту не нужно сверяться с инструкциями и документациями, это сильно экономит время выполнения

3) Меньше затраты на поддержку – Меньшие затраты на поддержку – когда автоматические скрипты уже написаны, на их поддержку и анализ результатов требуется, как правило, меньшее время, чем на проведение того же объема тестирования вручную.

4) Отчёты – автоматически рассылаемые и сохраняемые отчеты о результатах тестирования.

5) Выполнение без вмешательства – во время выполнения тестов инженер тестировщик может заниматься другими полезными делами или тесты могут выполняться в нерабочее время (этот метод предпочтительнее, т. к. нагрузка на локальные сети ночью снижена)

Недостатки автоматизации тестирования.

1) Повторяемость – все написанные тесты всегда будут выполняться однообразно. Это одновременно является и недостатком, т. к. тестировщик, выполняя тест вручную, может обратить внимание на некоторые детали и, проведя несколько дополнительных операций, найти дефект. Скрипт этого сделать не может.

2) Затраты на поддержку – несмотря на то что в случае автоматизированных тестов они меньше, чем затраты на ручное тестирование того же функционала, они все же есть. Чем чаще изменяется приложение, тем они выше.

3) Большие затраты на разработку – разработка автоматизированных тестов это сложный процесс, т. к. фактически идет разработка приложения, которое тестирует другое приложение. В сложных автоматизированных тестах также есть фреймворки, утилиты, библиотеки и прочее. Все это нужно тестировать и отлаживать, а это требует времени.

4) Стоимость инструмента для автоматизации – в случае, если используется лицензионное ПО, его стоимость может быть достаточно высока. Свободно распространяемые инструменты, как правило, отличаются более скромным функционалом и меньшим удобством работы.

5) Пропуск мелких ошибок – автоматический скрипт может пропускать мелкие ошибки, на проверку которых он не запрограммирован. Это могут быть неточности в позиционировании окон, ошибки в надписях, которые не проверяются, ошибки контроллеров и форм, с которыми не осуществляется взаимодействие во время выполнения скрипта

Что автоматизировать при тестировании?

1) Труднодоступные места в системе (бэкенд–процессы, логирование файлов, запись в БД).

2) Часто используемая функциональность, риски от ошибок в которой достаточно высоки. Автоматизировав проверку критической функциональности, можно гарантировать быстрое нахождение ошибок, а значит и быстрое их решение.

3) Рутинные операции, такие как переборы данных (формы с большим количеством вводимых полей). Заполнение полей различными данными и их проверку после сохранения.

4) Валидационные сообщения. Заполнение полей некорректными данными и проверку на появление той или иной валидации.

5) Длинные end-to-end-сценарии.

6) Проверка данных, требующих точных математических расчетов.

7) Проверка правильности поиска данных.

Задание на лабораторную работу

Установить Selenium IDE (установить расширение для браузера Google Chrome или для браузеров на ядре Chromium).

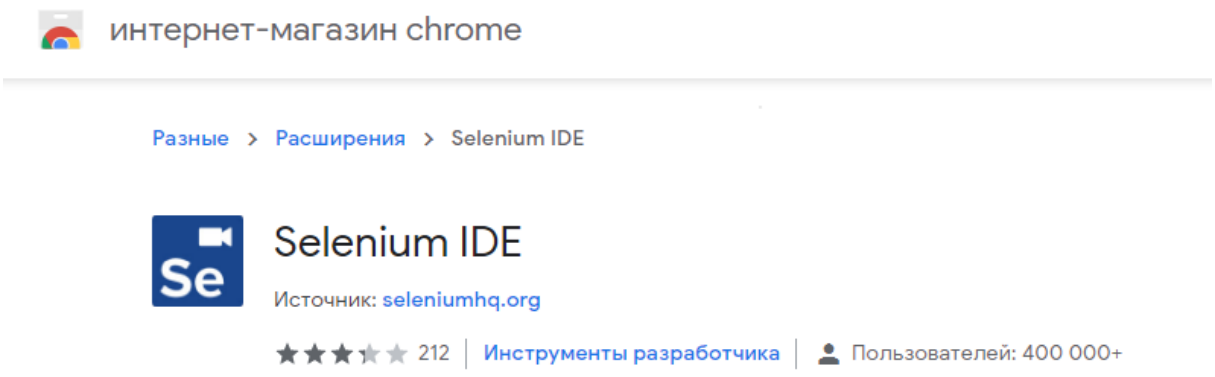


Рисунок 1 – Установка расширения

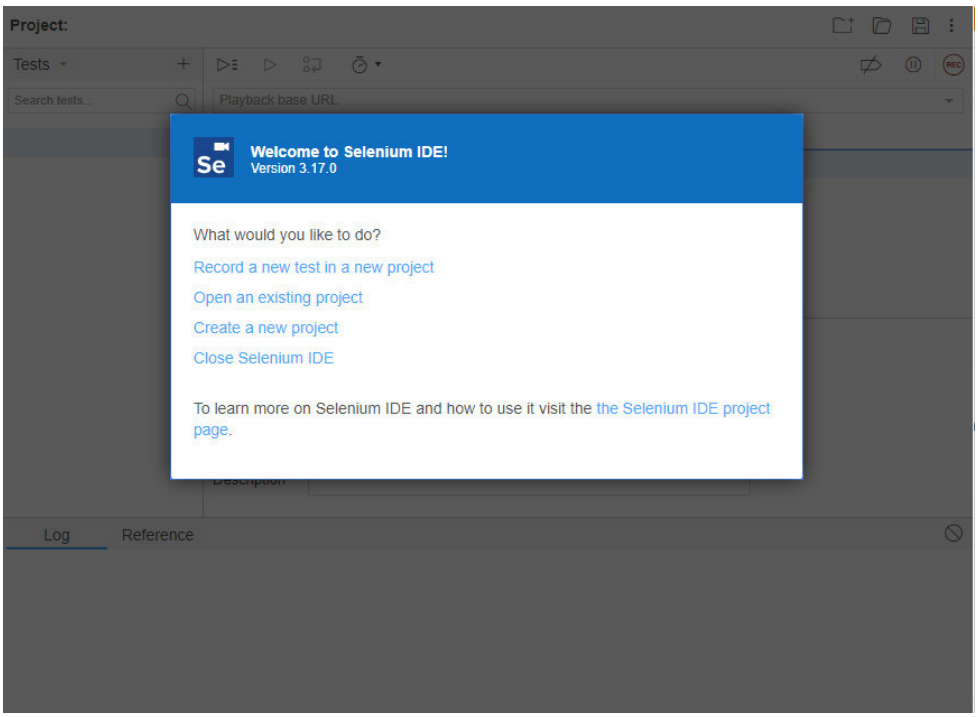


Рисунок 2 – Установленное расширение

Создать новый проект и указать ссылку с адресом на базу данных для тестирования.

Ознакомиться с интерфейсом программы и записать скрипт для тестирования посредством изучения открытых ресурсов в сети Интернет.

Убедиться, что присутствуют необходимые для синхронизации команды; что добавлены проверки, необходимые для тестирования того или иного функционала, а также несколько раз выполнить скрипт и убедиться в его работоспособности.

Требования к написанию скрипта:

1) Должен содержать действия с одним из сайтов, доступных в общем доступе (пример: https://vk.com, https://gmail.com, https://google.com и другие)

2) Количество команд в скрипте – не менее 3

3) Скрипт может быть открыт на другой ЭВМ и запуститься без ошибок.

4) Должно содержаться несколько проверок (assert/verify). В отчёте указать в чём отличия вышеуказанных проверок.

Требования к содержанию отчёта:

- Цель работы
- Краткие теоретические сведения
- Ход работы (пошаговая демонстрация выполнения задания на лабораторную работу)
- Выводы по работе
- Приложение

Контрольные вопросы:

1) Что такое автоматизация тестирования? Какие тесты называются автоматизированными?
2) В чем преимущества автоматизации тестирования?
3) В чем недостатки автоматизации тестирования?
4) Назовите области автоматизации тестирования.
5) Для чего необходимо проводить автоматизированное тестирование?

Лабораторная работа №5.
Тестирование методом «белого ящика».

Цель работы: приобрести практические навыки проведения тестирования программ методами «белого ящика».

Теоретические сведения

Особенности тестирования методом «белого ящика»:

- Известна внутренняя структура программы;
- Исследуются внутренние элементы программы и связи между ними;

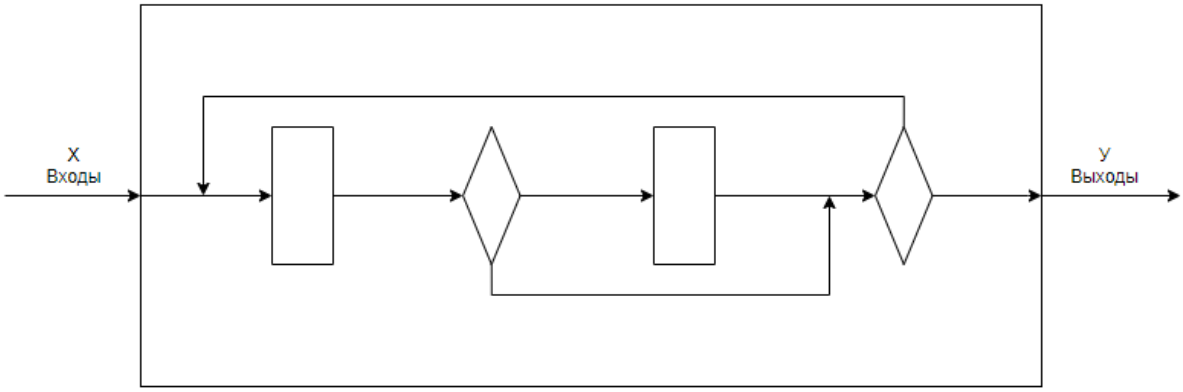


Рисунок 1 – Схема тестирования «белым ящиком»

Методы белого ящика

1. Метод покрытия операторов

Критерием этого метода является выполнение каждого оператора хотя бы один раз. Этот критерий является достаточно слабым, так как выполнение каждого оператора хотя бы один раз условие необходимое, но недостаточное для результирующего условия.

2. Метод покрытия решений

Согласно этому методу должно быть написано достаточное количество тестов, такое, что каждое решение на этих тестах выполняется по крайней мере один раз и при этом каждый оператор должен выполняться хотя бы один раз.

3. Метод покрытия условий

В этом случае записывают число тестов достаточное для того, чтобы всевозможные результаты каждого условия в решении выполнялись по крайней мере один раз.

4. Метод покрытия решений/условий

Этот метод требует достаточного набора тестов, чтобы возможные результаты каждого условия в решении выполнялись по крайней мере один раз и каждой точке входа передавалось управление по крайней мере один раз.

Тестирование программы с использованием потокового графа.

Для представления программы используется потоковый граф (рисунок 2). Особенности потокового графа:

1. Граф строится отображением управляющей структуры программы. В ходе отображения закрывающие скобки условных операторов и операторов циклов (end if; end loop) рассматриваются как отдельные (фиктивные) операторы.

2. Узлы (вершины) потокового графа соответствуют линейным участкам программы, включают один или несколько операторов программы.

3. Дуги потокового графа отображают поток управления в программе (передачи управления между операторами). Дуга – это ориентированное ребро.

4. Различают операторные и предикатные узлы. Из операторного узла выходит одна дуга, а из предикатного – две дуги.

5. Предикатные узлы соответствуют простым условиям в программе. Составное условие программы отображается в несколько предикатных узлов. Составным называют условие, в котором используется одна или несколько булевых операций (or, and).

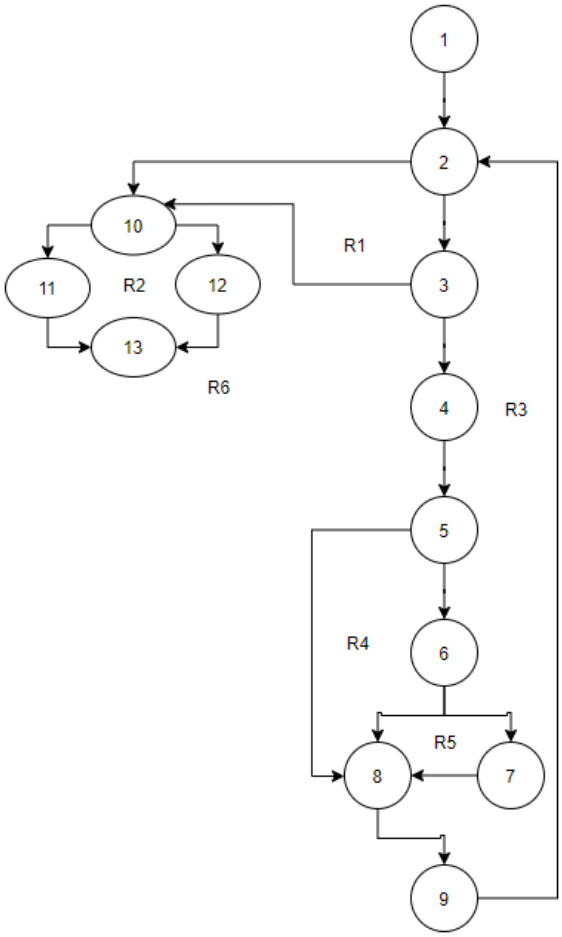


Рисунок 2 – Пример потокового графа

Цикломатическая сложность – метрика ПО, которая обеспечивает количественную оценку логической сложности программы. В способе тестирования базового пути цикломатическая сложность определяет:

- Количество независимых путей в базовом множестве программы;
- Верхнюю оценку количества тестов, которое гарантирует однократное выполнение всех операторов.

Независимым называется любой путь, который вводит новый оператор обработки или новое условие. В терминах потокового графа независимый путь должен содержать дугу, не входящую в ранее определенные пути.

Путь начинается в начальном узле, а заканчивается в конечном узле графа. Независимые пути формируются в порядке от самого короткого к самому длинному. Каждый новый путь включает новую дугу.

Все независимые пути графа образуют базовое множество.

Свойства базового множества:

1) тесты, обеспечивающие его проверку, гарантируют:

- однократное выполнение каждого оператора;

- выполнение каждого условия по True-ветви и по False-ветви;

2) мощность базового множества равна цикломатической сложности потокового графа.

Мощность базового множества дает априорную оценку количества независимых путей, которое имеет смысл искать в графе.

Цикломатическая сложность вычисляется одним из трех способов:

- цикломатическая сложность равныа количеству регионов потокового графа;
- цикломатическая сложность определяется по формуле: V(G)=E-N+2, где E – количество дуг, N – количество узлов потокового графа;
- цикломатическая сложность формируется по выражению V(G) = p + 1, где p – количество предикатных узлов в потоковом графе G.

Задание

1. построение потокового графа программы;
2. построение базового множества независимых линейных путей;
3. составление тестовых вариантов;
4. выполнение тестирования;
5. оформление результатов тестирования;

Варианты задания

1. Даны натуральное число N и одномерный массив A1, A2, …, AN, AN+1 вещественных чисел. Определить наибольшее из нечетных и количество четных чисел, входящих в этот массив.

2. Даны натуральное число N и одномерный массив A1, A2, …, AN целых чисел. Получить массив, который отличается от исходного тем, что все нечетные элементы удвоены, а четные получены сложением собственного значения с первоначальным значением последующего нечетного.

3. Даны натуральное число N (N>5) и одномерный массив A1, A2, …, AN символьных элементов. Определить три максимальных и два минимальных значения этого массива.

4. Даны натуральное число N и одномерный массив A1, A2, …, AN целых чисел. Определить наименьшее положительное среди A1, A2, …, AN.

5. Даны натуральное число N и одномерный массив A1, A2, …, AN вещественных чисел. В данном массиве определить число соседств двух положительных чисел.

6. Даны натуральное число N и одномерный массив A1, A2, …, AN целых чисел. В данном массиве определить число соседств двух чисел разного знака.

7. Даны натуральное число N и одномерный массив A1, A2, …, AN символьных элементов. Определить, является ли данный массив упорядоченным по убыванию.

8. Даны натуральное число N и одномерный массив A1, A2, …, AN натуральных чисел. Для каждого элемента определить число его вхождений в данный массив.

9. Даны натуральное число N и одномерный массив A1, A2, …, AN вещественных чисел. Получить все элементы, входящие в данный массив по одному разу.

10. Даны натуральное число N и одномерный массив A1, A2, …, AN символьных элементов. Получить все элементы, входящие в данный массив более одного раза.

Требования к содержанию отчёта:

- Цель работы
- Краткие теоретические сведения
- Ход работы (пошаговая демонстрация выполнения задания на лабораторную работу)
- Выводы по работе
- Приложение

Лабораторная работа №6.
Тестирование методом «черного ящика».

Цель работы: приобрести практические навыки проведения тестирования программ методами «черного ящика».

Теоретические сведения

Особенности тестирования методом «черного ящика»:

- Известны функции программы;
- Исследуется работа каждой функции на всей области определения;

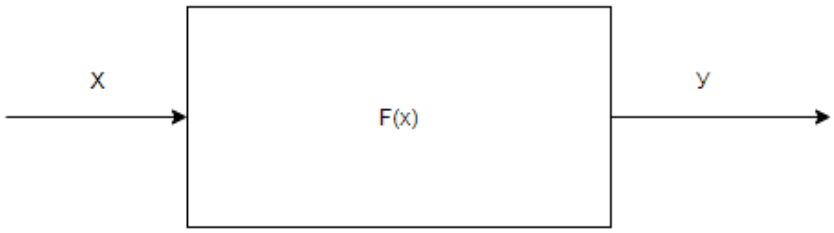


Рисунок 1 – Схема тестирования «черным ящиком»

Эти тесты демонстрируют:

- Как выполняются функции программ;
- Как принимаются исходные данные;
- Как вырабатываются результаты;
- Как сохраняется целостность внешней информации

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

Методы черного ящика

1. Метод эквивалентных разбиений

Его основу составляют 2 положения:

- Каждый тип должен включать столько различных входных и выходных условий, сколько необходимо для того, чтобы минимизировать общее число необходимых тестов;
- Необходимо разбивать входную область программы на конечное число классов эквивалентности.

Класс эквивалентности выделяется путем выбора каждого входного условия и разбиением его на две или более групп.

2. Анализ граничных решений (АГР)

Граничные условия – ситуация, возникающая непосредственно на границе, выше или ниже границ входных или выходных элементов класса эквивалентности (КЭ)

АГР предполагает:

- Выбор любого элемента в КЭ в качестве представительного при АГЗ осуществляется таким образом, чтобы проверить тестом каждую границу этого класса.
- При разработке тестов рассматриваются не только входные условия, но и выходные.

3. Метод функциональных диаграмм

Недостатком метода граничных решений и метода эквивалентных разбиений является то, что они не исследуют комбинации входных условий.

Метод функциональных диаграмм помогает создавать высоко результирующие тесты.

Функциональная диаграмма представляет собой формальный язык, на который транслируется спецификация, написанная на естественном языке.

Построение тестов осуществляется в несколько этапов:

- Спецификация разбивается на несколько участков
- Спецификация определяется причиной и следствием

Причины и следствия определяются путем последовательного чтения спецификации, каждой причине и следствию присваивается отдельный номер.

Графы причинно-следственных связей

Диаграммы причинно-следственных связей используются для проектирования тестовых вариантов и обеспечивают формальную запись логических условий и соответствующих действий. Данный способ является разновидностью тестирования «черного ящика». Используется автоматный подход к решению задачи.

На первом шаге способа тестирования, основанного на построении диаграмм причинно-следственных связей, для тестируемой программы (или отдельного тестируемого модуля) перечисляются причины (условия ввода или классы эквивалентности условий ввода) и следствия (действия или условия вывода). Каждой причине и следствию присваивается свой идентификатор.

На втором шаге данного способа тестирования разрабатывается граф причинно-следственных связей.

Введем нотацию базовых символов для записи графов причин и следствий. Причины будем обозначать символами c_i, а следствия – символами e_j. Каждый узел графа может находиться в состоянии 0 (состояние отсутствует) или 1 (состояние присутствует).

Функция «тождество» (рисунок 2) устанавливает, что если значение есть 1, то и значение есть 1. В противном случае значение есть 0.

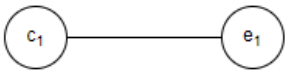


Рисунок 2 – Функция «тождество»

Функция «не» (рисунок 3) устанавливает, что если значение c_1 есть 1, то значение e_1 есть 0. В противном случае значение есть 1.

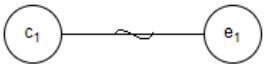


Рисунок 3 – Функция «не»

Функция «или» (рисунок 4) устанавливает, что если c_1 или c_2 есть 1, то e_1 есть 1. В противном случае e_1, есть 0.

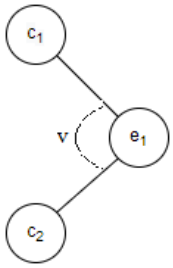


Рисунок 4 – Функция «или»

Функция «и» (рисунок 5) устанавливает, что если c_1 и c_2 есть 1, то e_1 есть 1. В противном случае есть 0.

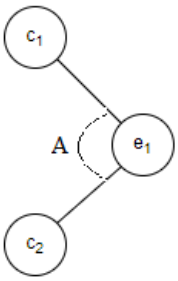


Рисунок 5 – Функция «и»

На третьем шаге рассматриваемого способа тестирования граф преобразуется в таблицу решений. Порядок генерации таблицы решений [1]:

1) Выбирается некоторое следствие, которое должно быть в состоянии «1».

2) Находятся все комбинации причин (с учетом ограничений), которые устанавливают это следствие в состояние «1». Для этого из следствия прокладывается обратная трасса через граф.

3) Для каждой комбинации причин, приводящих следствие в состояние «1», строится один столбец.

4) Для каждой комбинации причин доопределяются в состояния всех других следствий. Они помещаются в тот же столбец таблицы решений.

5) Действия 1-4 повторяются для всех следствий графа. На четвертом шаге данного способа тестирования столбцы таблицы решений преобразуются в тестовые варианты.

Практическое задание

Выполнение работы предусматривает следующую последовательность действий:

1. Определение причин (условий ввода) и следствий;
2. Построение графа причинно-следственных связей;
3. Создание таблиц решений;
4. Построение тестовых вариантов;
5. Оформление результатов тестирования.

Требования к содержанию отчёта:

- Цель работы
- Краткие теоретические сведения
- Ход работы (пошаговая демонстрация выполнения задания на лабораторную работу)
  1. Перечень причин и следствий;
  2. Граф причинно-следственных связей;
  3. Таблица решений;
  4. Тестовые варианты;
  5. Результаты тестирования.
- Выводы по работе

Варианты задания

Построить таблицу значений функции $y=f(x)$, x изменяется от x_{min} до x_{max} с шагом dx. Проконтролировать правильность ввода значений $x_{min/max}$, dx и корректность вычисляемого выражения.

1. $y=\frac{a+20b}{x^3}*\ln 2x-\frac{1}{(a-1)^2}$

2. $y=3\sqrt{\frac{5x-9}{7.5ab}+18}+e^{2x+\frac{0.5}{a}}$

3. $y=\frac{x^4-a^3-b^2}{\sqrt{19x-3.5}+\ln a}$

4. $y=\sin\frac{e^x-3a}{a^2+b^2}+\frac{10}{x^3}$

5. $y=\cos\frac{(x-a)^2}{x-2a}-\frac{3.5}{\sqrt{xb}}$

6. $y=\sin\frac{a+\cos^2 x}{\cos x^2-b}+2.5a\sqrt{b}$

7. $y=\frac{tg3a-20|b|-\sqrt{ab}}{x^2+b^3}$

8. $y=2arctg\frac{25a}{b}+3\cos^2\frac{9xb}{b-x}$

9. $y=|x^2-a^2|+\frac{9x^3}{(b-x)^3}*\sin\frac{x}{a}$

10. $y=|(2a-7.5x)^3|+e^{\frac{2b}{x}-\frac{a}{2b}}$

11. $y=\sin 3x+\cos^2\frac{x}{2a+b}-\frac{2x}{a}$

12. $y=\ln\frac{3x^3-2x^2+x}{(a^2+b)^2}+\frac{a}{x^3-4x^2-.}$

13. $y=e^{|\sin(3ax+b)|}*\frac{x}{(\sqrt{ax}+bx^2)^3}$

14. $y=\sin\frac{|x|}{2\sqrt{a}}+\cos^2\frac{x^3}{a+b}$

15. $y=2\operatorname{ctg}\frac{x^3-2x^2+|x|}{(a+\sqrt{b})^3}-\frac{1}{x}$

16. $y=\sin 5e^{\frac{x}{a}+\frac{b}{2}-\frac{3}{x}}+\cos^2 3ax$

Рисунок 6 – Варианты задания.

Лабораторная работа №7.
Unit-тестирование.

Цель работы: изучить основы разработки модульных тестов и получить навыки работы со средствами тестирования Unit Testing Framework от Microsoft, NUnit.

Теоретические сведения

Модульное тестирование, или unit-тестирование – процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы.

Цель модульного тестирование – изолировать отдельные части программы и показать, что по отдельности эти части работоспособны.

Будучи уверенным в корректной работоспособности модуля программы, удобно проводить рефакторинг.

Модульное тестирование помогает устранить сомнения по поводу отдельных модулей и может быть использовано для подхода к тестирования «снизу вверх». То есть, сначала тестируются отдельные части программы, а затем программа в целом.

Для проведения модульного тестирования будет использоваться набор средств от Microsoft – Unit Testing Framework.

Средства модульных тестов включают:

1. Обозреватель тестов.
2. Платформа Microsoft для модульного тестирования управляемого кода.
3. Платформа Microsoft для модульного тестирования на C++.
4. Средства покрытия кода.
5. Платформа изоляции Microsoft Fakes.

Также модно использовать компонент IntelliTest, чтобы изучить код .Net и создать тестовые данные и набор модульных тестов.

Практическое задание

В качестве справочных материалов предоставляются статьи с сайта Microsoft: https://docs.microsoft.com/ru-ru/visualstudio/test/unit-test-your-code?view=vs-2015&redirectedfrom=MSDN

1. Изучить средства тестирования, доступные в Visual Studio – Unit Testing Framework, NUnit.
2. Реализовать алгоритм в соответствии с номером варианта на любом языке, поддерживаемом платформой .NET. Использование других языков необходимо согласовать с преподавателем.
3. Разработать набор unit-тестов для алгоритма в соответствии с номером варианта.
4. Обеспечить максимально возможное покрытие кода тестами

Конечная цель:

- Работающая функциональность в соответствии с заданием
- Проходящие unit-тесты

Требования к содержанию отчёта:

- Цель работы
- Краткие теоретические сведения
- Ход работы (пошаговая демонстрация выполнения задания на лабораторную работу)
  1. Реализация алгоритма
  2. Проведение тестирования
- Выводы по работе

Варианты задания

Номер варианта	Алгоритм сортировки
1.	Методом быстрой сортировки
2.	Методом простых вставок
3.	Методом выбора
4.	Методом пузырька
5.	Пирамидальная сортировка
6.	Методом Шелла