МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Управление качеством программного обеспечения

Методические указания для практических занятий

Направление подготовки 09.03.02 Информационные

системы и технологии

Направленность (профиль) Информационные системы управления технологическими и сервисными процессами

(Электронный документ)

Невинномысск 2026

Содержание

ТЕМА 1. ОСНОВНЫЕ ПОНЯТИЯ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №1 ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ: "РАЗРАБОТКА ЧЕРЕЗ ТЕСТИРОВАНИЕ"

Количество аудиторных часов - 2

Цель работы

Знакомство с технологией "разработка через тестирование". Изучение инструментов позволяющих применять данную технологию.

Общие сведения

Разработка через тестирование (англ. test-driven development, TDD) — техника разработки программного обеспечения, которая основывается на повторении очень коротких циклов разработки: сначала пишется тест, покрывающий желаемое изменение, затем пишется код, который позволит пройти тест, и под конец проводится рефакторинг нового кода к соответствующим стандартам.

TDD требует от разработчика создания автоматизированных модульных тестов, определяющих требования к коду непосредственно перед написанием самого кода. Тест содержит проверки условий, которые могут либо выполняться, либо нет. Когда они выполняются, говорят, что тест пройден. Прохождение теста подтверждает поведение, предполагаемое программистом. Разработчики часто пользуются библиотеками для тестирования (англ. testing frameworks) для создания и автоматизации запуска наборов тестов. На практике модульные тесты покрывают критические и нетривиальные участки кода. Это может быть код, который подвержен частым изменениям, код, от работы которого зависит работоспособность большого количества другого кода, или код с большим количеством зависимостей.

Среда разработки должна быстро реагировать на небольшие модификации кода. Архитектура программы должна базироваться на использовании множества сильно связанных компонентов, которые слабо сцеплены друг с другом, благодаря чему тестирование кода упрощается.

TDD не только предполагает проверку корректности, но и влияет на дизайн программы. Опираясь на тесты, разработчики могут быстрее представить, какая функциональность необходима пользователю. Таким образом, детали интерфейса появляются задолго до окончательной реализации решения

Задание

Изучить библиотеки для тестирования. Рассмотреть применение NUnit, ReSharper.

ТЕМА 1. ОСНОВНЫЕ ПОНЯТИЯ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №2 МОДУЛЬНОЕ ТЕСТИРОВАНИЕ

Количество аудиторных часов - 6

Цель работы

Овладение навыками модульного тестирования.

Общие сведения

Модульное тестирование - это тестирование программы на уровне отдельно взятых модулей, функций или классов. Цель модульного тестирования состоит в выявлении локализованных в модуле ошибок в реализации алгоритмов, а также в определении степени готовности системы к переходу на следующий уровень разработки и тестирования. Модульное тестирование проводится по принципу "белого ящика", то есть основывается на знании внутренней структуры программы, и часто включает те или иные методы анализа покрытия кода.

Модульное тестирование обычно подразумевает создание вокруг каждого модуля определенной среды, включающей заглушки для всех интерфейсов тестируемого модуля. Некоторые из них могут использоваться для подачи входных значений, другие для анализа результатов, присутствие третьих может быть продиктовано требованиями, накладываемыми компилятором и сборщиком.

На уровне модульного тестирования проще всего обнаружить дефекты, связанные с алгоритмическими ошибками и ошибками кодирования алгоритмов, типа работы с условиями и счетчиками циклов, а также с использованием локальных переменных и ресурсов. Ошибки, связанные с неверной трактовкой данных, некорректной реализацией интерфейсов, совместимостью, производительностью и т.п. обычно пропускаются на уровне модульного тестирования и выявляются на более поздних стадиях тестирования.

Задание

"Мозговая атака": Этапы:

Получить вопрос (задание) для обсуждения.

Задать вопросы относительно не понятных моментах в вопросе (задании).

Высказать свои мысли по данному вопросу (заданию).

Записать все прозвучавшие высказывания с уточнениями.

По окончанию прочитать все, что было записано.

Обсудить все варианты ответов.

Выяснить, как можно использовать полученные результаты при выполнении данной лабораторной работы.

"Работа в команде" Этапы:

Разбиться на команды.

Реализовать полученный вопрос (задание), согласно технологии TDD.

Представить результаты.

ТЕМА 2. РАЗНОВИДНОСТИ ТЕСТИРОВАНИЯ И ОЦЕНКИ КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №3. СОСТАВЛЕНИЕ НАБОРОВ ТЕСТОВЫХ ДАННЫХ

для функционального тестирования. Стратегия "черного ящика"

Основные теоретические положения

При функциональном тестировании программа рассматривается как «черный ящик», и целью тестирования является выяснение обстоятельств, в которых поведение программы не соответствует спецификации.

Исчерпывающее тестирование, т. е. тестирование на всех возможных наборах данных, в большинстве случаев невозможно (потребовались бы гигантские затраты времени и средств). А для программ, где исполнение команды зависит от предшествующих ей событий, необходимо также проверить и все возможные последовательности! Но, с другой стороны, необходимо обеспечить требуемое качество программного обеспечения. Возникает вопрос: как обнаружить максимальное количество ошибок, ограничив при этом подмножество возможных входных данных?

Существуют различные методы формирования тестовых наборов. В числе основных критериев «черного ящика» следующие:

покрытие функций. Для каждой из функций, реализуемых программой, требуется подобрать и выполнить хотя бы один тест;

покрытие классов входных данных. Критерий тестирования классов входных данных требует классифицировать входные данные, разделить их на классы таким образом, чтобы все данные из одного класса были равнозначны с точки зрения проверки правильности программы. Считается, что если программа работает правильно на одном наборе входных данных из этого класса, то она будет правильно работать на любом другом наборе данных из этого же класса. Критерий требует выполнения хотя бы одного теста для каждого класса входных данных;

покрытие классов выходных данных. Аналогичен предыдущему критерию, только проверяются не входные данные, а выходные;

покрытие области допустимых значений (тестирование границ класса). Для переменной, возможные значения которой перечислены (ноты, цвет, пол, диагноз и т.п.), следует убедиться, что на все указанные значения программа реагирует правильно и не принимает вместо них никаких иных значений. Если класс допустимых значений представляет собой числовой диапазон, то выделяются нормальные условия (в середине класса), граничные (экстремальные) условия и исключительные условия (выход за границу класса);

тестирование длины набора данных (можно считать частным случаем тестирования области допустимых значений). Определяется допустимое количество элементов в наборе. Если программа последовательно обрабатывает элементы некоторого набора данных, имеет смысл проверить следующие ситуации:

пустой набор (не содержит ни одного элемента);

единичный набор (состоит из одного-единственного элемента);

слишком короткий набор (если предусмотрена минимально допустимая длина);

набор минимально возможной длины (если такая предусмотрена); нормальный набор (состоит из нескольких элементов);

набор из нескольких частей (если такое возможно. Например, если программа читает литеры из текстового файла или печатает текст, то как она отнесется к переходу на следующую строку? На следующую страницу?);

набор максимально возможной длины (если такая предусмотрена);

слишком длинный набор (с длиной больше максимально допустимой);

тестирование упорядоченности набора данных. Важно для задач сортировки и поиска экстремумов. В этом случае имеет смысл проверить следующие ситуации (классы входных

данных): данные неупорядочены; данные упорядочены в прямом порядке; данные упорядочены в обратном порядке; в наборе имеются повторяющиеся значения; экстремальное значение находится в середине набора; экстремальное значение находится в начале набора; экстремальное значение находится в конце набора; в наборе несколько совпадающих экстремальных значений.

Критерии покрытия функций, классов входных и выходных данных хорошо согласуются друг с другом и обычно каждой из функций ПП соответствуют свои классы входных и выходных данных. Например, программа учета кадров предприятия имеет функции: принять на работу, уволить с работы, перевести с одной должности на другую, выдать кадровую сводку. Соответственно классы входных данных

– приказы о приеме, об увольнении, о переводе, заявка на кадровую сводку, а выходных – записи о приеме, об увольнении, о переводе, кадровая сводка.

В данной лабораторной работе предлагается рассмотреть два метода формирования тестовых наборов: 1) на основе классов эквивалентности и 2) на основе граничных значений классов эквивалентности. Оба метода основаны на исследовании входных данных. Они не позволяют проверять результаты, получаемые при различных сочетаниях данных. Для построения тестов, проверяющих сочетания данных, применяют методы, использующие булеву алгебру.

Эквивалентное разбиение. Метод эквивалентного разбиения заключается в следующем. Область всех возможных наборов входных данных программы по каждому параметру разбивают на конечное число групп классов эквивалентности. Наборы данных такого класса объединяют по принципу обнаружения одних и тех же ошибок: если набор какого-либо класса обнаруживает некоторую ошибку, то предполагается, что все другие тесты этого класса эквивалентности тоже обнаружат эту ошибку и наоборот.

Разработку тестов методом эквивалентного разбиения осуществляют в два этапа: на первом выделяют классы эквивалентности, а на втором формируют тесты.

Формирование классов эквивалентности является эвристическим процессом, однако целесообразным считают выделять в отдельные классы эквивалентности наборы, содержащие допустимые и недопустимые значения некоторого параметра. При этом существует ряд правил:

если некоторый параметр x может принимать значения в интервале [1, 999], то выделяют один правильный класс $1 < x < 999$ и два неправильных: $x < 1$ и $x > 999$;

если входное условие определяет диапазон значений порядкового типа, например, «в автомобиле могут ехать от одного до шести человек», то определяется один правильный класс эквивалентности и два неправильных: ни одного и более шести человек;

если входное условие описывает множество входных значений и есть основания полагать, что каждое значение программист трактует особо, например, «типы графических файлов: bmp, jpeg, vsd», то определяют правильный класс эквивалентности для каждого значения и один неправильный класс, например, txt;

если входное условие описывает ситуацию «должно быть», например,

«первым символом идентификатора должна быть буква», то определяется один правильный класс эквивалентности (первый символ буква) и один неправильный (первый символ не буква);

если есть основание считать, что различные элементы класса эквивалентности трактуются программой неодинаково, то данный класс разбивается на меньшие классы эквивалентности.

Таким образом, классы эквивалентности выделяют, перебирая ограничения, установленные для каждого входного значения в техническом задании или при уточнении спецификации. Каждое ограничение разбивают на две или более групп. При этом используют специальные бланки таблицы классов эквивалентности:

Ограничение на значение параметра	**Правильные классы эквивалентности**	**Неправильные классы эквивалентности**

Правильные классы включают правильные данные, неправильные классы неправильные данные. Для правильных и неправильных классов тесты проектируют отдельно. При построении тестов правильных классов учитывают, что каждый тест должен проверять по возможности максимальное количество различных входных условий. Такой подход позволяет минимизировать общее число необходимых тестов. Для каждого неправильного класса эквивалентности формируют свой тест. Последнее обусловлено тем, что определенные проверки с ошибочными входами скрывают или заменяют другие проверки с ошибочными входами.

Анализ граничных значений. Граничные значения — это значения на границах классов эквивалентности входных значений или около них. Анализ показывает, что в этих местах резко увеличивается возможность обнаружения ошибок. Например, если в программе анализа вида треугольника было записано A + B □ □C вместо A + B > C, то задание граничных значений приведет к ошибке: линия будет отнесена к одному из видов треугольника.

Применение метода анализа граничных значений требует определенной степени творчества и специализации в рассматриваемой проблеме. Тем не менее, существует несколько общих правил для применения этого метода:

если входное условие описывает область значений, то следует построить тесты для границ области и тесты с неправильными входными данными для ситуаций незначительного выхода за границы области, например, если описана область [-1.0, +1.0], то должны быть сгенерированы тесты: -1.0, +1.0,-1.001 и +1.001;

если входное условие удовлетворяет дискретному ряду значений, то следует построить тесты для минимального и максимального значений и тесты, содержащие значения большие и меньшие этих двух значений, например, если входной файл может содержать от 1 до 255 записей, то следует проверить О, 1, 255 и 256 записей;

если существуют ограничения выходных значений, то целесообразно аналогично тестировать и их: конечно, не всегда можно получить результат вне выходной области, но тем не менее стоит рассмотреть эту возможность;

если некоторое входное или выходное значение программы является упорядоченным множеством, например, это последовательный файл, линейный список или таблица, то следует сосредоточить внимание на первом и последнем элементах этого множества.

Помимо указанных граничных значений, целесообразно поискать другие. Однако следует помнить, что граничные значения могут быть едва уловимы и определение их связано с большими трудностями, что является недостатком этого метода.

Пример. Пусть необходимо выполнить тестирование программы, определяющей точку пересечения двух прямых на плоскости. При этом она должна определять параллельность прямой одной из осей координат.

В основе программы лежит решение системы линейных уравнений:

Ax + By = C, Dx + Ey = F.

По методу эквивалентных разбиений формируем для каждого коэффициента один правильный класс эквивалентности (коэффициент –вещественное число) и один неправильный (коэффициент – не вещественное число). Откуда генерируем 7 тестов:

1) все коэффициенты вещественные числа (1 тест);

2-7) поочередно каждый из коэффициентов не вещественное число (6 тестов).

По методу граничных значений можно считать, что для исходных данных граничные значения отсутствуют, т. е. коэффициенты «любые» вещественные числа.

Для результатов получаем, что возможны варианты: единственное решение, прямые сливаются множество решений, прямые параллельны отсутствие решений. Следовательно, целесообразно предложить тесты с результатами внутри областей возможных значений результатов:

результат единственное решение (δ ≠ 0 (определитель δ=AE-BD));

результат множество решений (δ = 0 и δ X = δУ = 0, (δ X=CE-BF, δ У=AF-CD));

результат отсутствие решений (δ = 0, но δ X ≠ 0 или δУ ≠ 0);
и с результатами на границе: 11) δ = 0,01;
12) δ = -0,01;
13) δ = 0, δ X = 0,01, δ =0;
14) δ = 0, δ У =-0,01, δ X = 0.

Задачи

Задача 3.1. Есть программа, которая интерпретирует три целых числа, вводимых с клавиатуры, как длины сторон треугольника и выводит сообщение, о том, какой это треугольник: равносторонний, равнобедренный или неравносторонний. Напишите на листе бумаги тесты (последовательности входных данных и ожидаемые результаты), которые, как вам кажется, будут адекватно проверять эту программу. Рекомендуется запись тестов в виде таблицы:

Критерий формирования теста	№ теста	Исходные данные			Ожидаемый результат
		a	b	c	

Задача 3.2. В программе «Деканат» переменная, обозначающая количество студентов в группе, может принимать значения от 1 до 30. Составьте тестовый набор для этой переменной.

Задача 3.3. В компьютерной обучающей системе тестовые задания для контроля знаний берутся из файла типа .txt, где каждое задание занимает одну строку. Для формирования теста указывается имя файла, количество заданий в тесте и количество вариантов теста (не более 10). Варианты должны различаться не менее чем тремя заданиями. Если заданий в файле недостаточно для реализации этого требования, выдается сообщение «Недостаточно заданий», если файл не найден «файл отсутствует». Увидеть составленный вариант теста для контроля знаний можно, нажав на соответствующую кнопку «Вариант №».

Составьте тестовые наборы для проверки перечисленных функций. Рекомендуется оформить их в виде таблицы, подобно задаче 3.1.

Задача 3.4. Используя методы эквивалентного разбиения и граничных значений, составьте тестовые наборы для проверки перечисленных ниже функций программы "Геометрические фигуры". Протестируйте указанную программу.

В программе "Геометрические фигуры" (файл geometry.exe) после ее запуска пользователю предоставляется выбрать вариант геометрической программы. Для выбора пользователь должен нажать на кнопку с названием нужного варианта. Предусмотрены следующие варианты:

Вариант 1. Программа определяет тип треугольника. Возможные результаты: прямоугольный, остроугольный, тупоугольный, равнобедренный, правильный (равносторонний)

Вариант 2. Четырехугольник задается координатами вершин. Программа проверяет, является ли он квадратом

Вариант 3. Четырехугольник задается координатами вершин. Программа проверяет, является ли он ромбом

Вариант 4. Определение взаимного положения прямой и окружности. Прямая описывается уравнением Y=kX+b. Окружность с центром в начале координат задается радиусом R. Результат – линии не пересекаются, пересекаются в двух точках, прямая линия является касательной к окружности.

Вариант 5. Определение взаимного положения двух окружностей. Окружности задаются координатами центра X,Y и радиусом R. Результат – линии не пересекатся, пересекаются в двух точках, касаются в одной точке, совпадают.

Контрольные вопросы

Почему стратегия функционального тестирования называется также стратегией "черного ящика"?

По каким критериям осуществляется функциональное тестирование?

В чем заключается метод эквивалентного разбиения?

По какому принципу формируют классы эквивалентности?

Почему анализ граничных значений считается одним из наиболее полезных методов проектирования тестов?

ТЕМА 2. РАЗНОВИДНОСТИ ТЕСТИРОВАНИЯ И ОЦЕНКИ КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №4. СОСТАВЛЕНИЕ НАБОРОВ ТЕСТОВЫХ ДАННЫХ

для структурного тестирования. Стратегия "белого (прозрачного) ящика"

Основные теоретические положения

При использовании стратегии «белого ящика» тестовые наборы формируют путем анализа маршрутов, предусмотренных алгоритмом. Под маршрутами при этом понимают последовательности операторов программы, которые выполняются при конкретном варианте исходных данных. Тестирование, проводимое по составленным таким образом тестам, называют структурным или тестированием по маршрутам.

В основе структурного тестирования лежит концепция максимально полного тестирования всех маршрутов программы. Так, если алгоритм программы включает ветвление, то при одном наборе исходных данных может быть выполнена последовательность операторов, реализующая действия, которые предусматривает одна ветвь, а при втором □□другая. Соответственно, для программы будут существовать маршруты, различающиеся выбранным при ветвлении вариантом.

Считают, что программа проверена полностью, если с помощью тестов удается осуществить выполнение программы по всем возможным маршрутам передач управления. Однако нетрудно видеть, что даже в программе среднего уровня сложности число неповторяющихся маршрутов может быть очень велико, и, следовательно, полное или исчерпывающее тестирование маршрутов, как правило, невозможно.

Последовательность составления тестов следующая:

На основе алгоритма (текста) программы формируется потоковый граф (или граф-схема). Узлы (вершины) потокового графа соответствуют линейным участкам программы, включают один или несколько операторов программы. Дуги (ориентированные ребра) потокового графа отображают поток управления в программе (передачи управления между операторами). Различают операторные и предикатные узлы. Из операторного узла выходит одна дуга, а из предикатного □□две дуги. Предикатные узлы соответствуют простым условиям в программе. Составное условие программы (условие, в котором используется одна или несколько булевых операций (OR, AND)) отображается в несколько предикатных узлов. На рис. 4.1 показан пример такого преобразования для фрагмента программы: if a OR b then x else y end if.

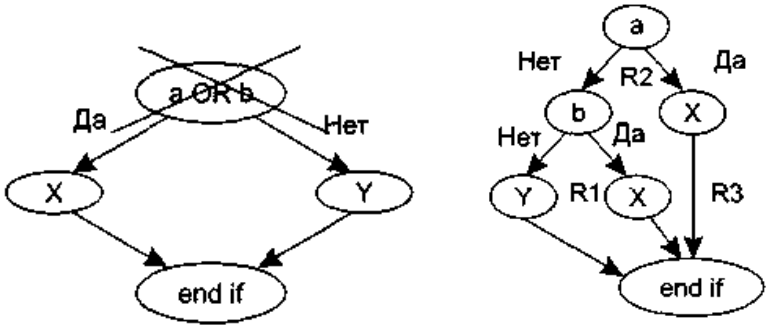

Рис. 4.1. Прямое отображение фрагмента программы в потоковый граф и преобразованный потоковый граф

Выбирается критерий тестирования. Формирование тестовых наборов для тестирования маршрутов может осуществляться по нескольким критериям: покрытие маршрутов, покрытие операторов, покрытие решений (переходов), покрытие условий, покрытие решений/условий, комбинаторное покрытие условий, покрытие потоков данных, покрытие циклов.

Подготавливаются тестовые варианты, инициирующие выполнение каждого пути. Каждый

тестовый вариант формируется в следующем виде:

Исходные данные (ИД):

Ожидаемые результаты (ОЖ.РЕЗ.):

Тесты, составленные по критерию покрытие маршрутов и обеспечивающие проверку базового множества, гарантируют однократное выполнение каждого оператора и выполнение каждого условия по True-ветви и по False-ветви. Базовое множество образуют все независимые пути графа. Независимым называется любой путь от начального до конечного узла графа, который вводит новый оператор обработки или новое условие. В терминах потокового графа независимый путь должен содержать дугу, не входящую в ранее определенные пути. Независимые пути формируются в порядке от самого короткого к самому длинному.

Число независимых линейных путей в базовом множестве определяется цикломатической сложностью алгоритма, которая вычисляется одним из трех способов:

цикломатическая сложность V(G) равна количеству регионов потокового графа; Регион замкнутая область, образованная дугами и узлами. Окружающая граф среда рассматривается как дополнительный регион. Например, показанный на рис.

4.1 граф имеет три региона □□Rl, R2, R3;

цикломатическая сложность определяется по формуле

V(G) = E-N+2,

где E □□количество дуг, N □□количество узлов потокового графа;

цикломатическая сложность V(G) = p +1, где p – количество предикатных узлов G.

Пример 4.1.

Цикломатическая сложность алгоритма на рис.4.1.:

V(G) = 3 региона;

2) V(G) = 7 дуг 6 узлов + 2 = 3;

3) V(G) = 2 предикатных узлов + 1 = 3. Независимые пути:

Путь 1: a x – end if. Путь 2: a b x – end if. Путь 3: a b y – end if.

Критерий покрытия операторов подразумевает такой подбор тестов, чтобы каждый оператор программы выполнялся, по крайней мере, один раз. Это необходимое, но недостаточное условие для приемлемого тестирования.

Для реализации критерия покрытие решений (переходов) необходимо такое количество и состав тестов, чтобы результат проверки каждого условия (т.е. решение) принимал значения «истина» или «ложь», по крайней мере, один раз. Нетрудно видеть, что критерий покрытия решений удовлетворяет критерию покрытия операторов, но является более «сильным».

Критерий покрытия условий является еще более «сильным» по сравнению с предыдущими. В этом случае формируют некоторое количество тестов, достаточное для того, чтобы все возможные результаты каждого условия в решении были выполнены, по крайней мере, один раз. Однако, как и в случае покрытия решений, этот критерий не всегда приводит к выполнению каждого оператора, по крайней мере, один раз. К критерию требуется дополнение, заключающееся в том, что каждой точке входа управление должно быть передано, по крайней мере, один раз.

Согласно покрытию решений/условий тесты должны составляться так, чтобы, по крайней мере, один раз выполнились все возможные результаты каждого условия и все результаты каждого решения, и каждому оператору управление передавалось, по крайней мере, один раз.

Критерий комбинаторное покрытие условий требует создания такого множества тестов, чтобы все возможные комбинации результатов условий в каждом решении и все операторы выполнялись, по крайней мере, один раз.

Пример 4.2.

Требуется выполнить структурное тестирование текста программы, которая определяет значение x в зависимости от значений параметров процедуры.

Procedure т (a, b:real; var x:real); begin

if(a>1) and (b=0) then x:=x /a; if(a=2) or (x>l) then x:=x+1;

end;

Для формирования тестов программу представляют в виде графа, вершины которого соответствуют операторам программы, а дуги представляют возможные варианты передачи управления (рис.4.2).

Покрытие операторов будет реализовано при a = 2, b = 0, x = 3.

Однако, хотя исходные данные заданы так, чтобы все операторы программы были выполнены хотя бы один раз, для проверки программы этого явно недостаточно. Например, из второго условия следует, что переменная x может принимать любое значение, и в некоторых версиях языка Pascal это значение проверяться не будет. Кроме того, если при написании программы в первом условии указано, что (a > 1) or (b = 0), или, если во втором условии вместо x > 1 записано x > 0, то эти ошибки обнаружены не будут. Также существует путь 1-2-4-6, в котором x вообще не меняется и, если здесь есть ошибка, она не будет обнаружена.

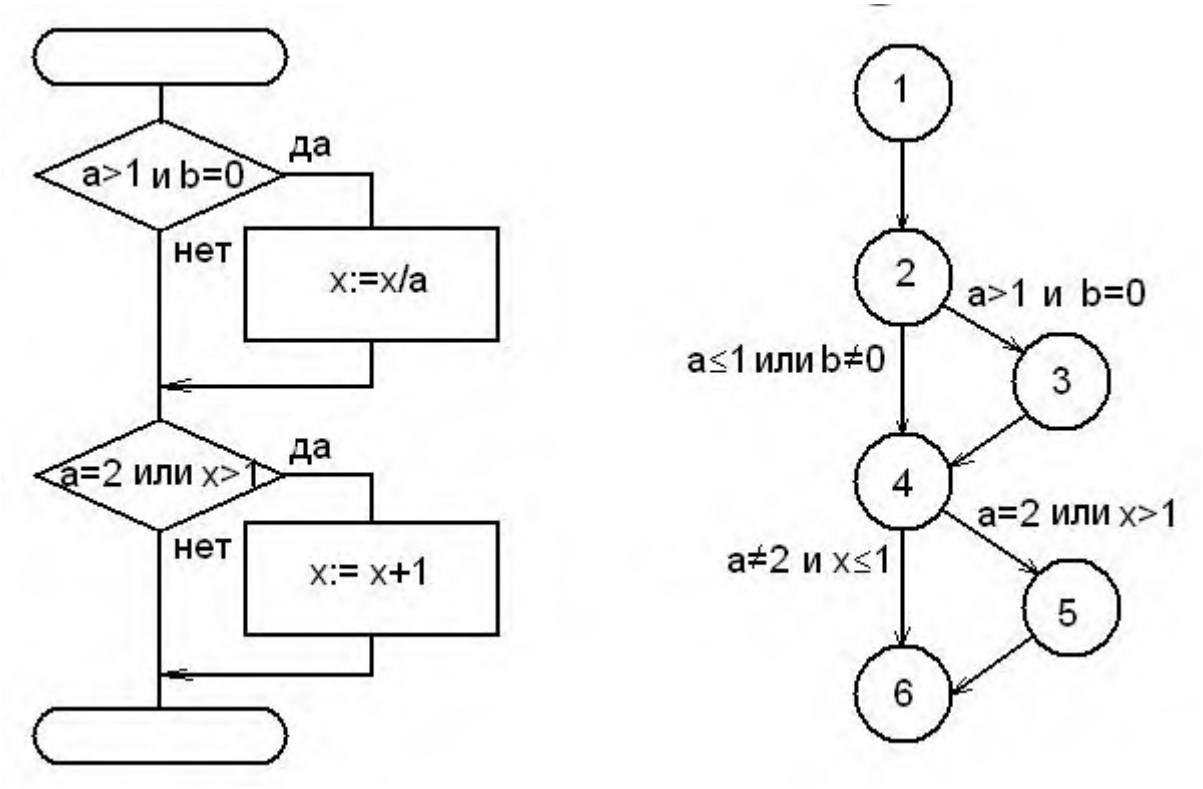

Рис. 4.2. Схема алгоритма процедуры (слева) и ее граф передач управления

По методу покрытия решений (переходов) рассматриваемую программу можно протестировать двумя тестами, покрывающими либо пути: 1-2-4-6, 1-2-3-4-5-6, либо пути: 1-2-3-4-6, 1-2-4-5-6, например:

a = 3, b = 0, x = 3 — путь 1-2-3-4-5-6;

a = 2, b = 1 , x = 1 — путь 1-2-4-5-6

Однако путь, где x не меняется, будет проверен с вероятностью 50 %: если во втором условии вместо x > 1 записано x < 1, то этими двумя тестами ошибка обнаружена не будет.

Покрытие условий проверяет четыре условия:

l) a>l; 2) b = 0; 3) a = 2; 4) x>1.

Необходимо реализовать все возможные ситуации: Тесты, удовлетворяющие этому условию:

a= 2, b = 0, x = 4 — путь 1-2-3-4-5-6, условия: 1 да, 2 да, 3-да, 4-да

a = 1, b = 1, x = 1 — путь 1-2-4-6, условия: 1 нет, 2 нет, 3 – нет, 4-нет.

Критерий покрытия условий часто удовлетворяет критерию покрытия решений, но не всегда. Тесты критерия покрытия условий для ранее рассмотренных примеров покрывают результаты всех

решений, но это случайное совпадение. Например, тесты:

a=1,Ь = 0, x = 3 — путь 1-2-3-6, условия: 1 нет, 2 да, 3 нет, 4 да;

a = 2, b = 1, x = 1 — путь 1-2-3-4-5-6, условия: 1 да, 2 нет, 3 да, 4 – нет покрывают результаты всех условий, но только два из четырех результатов решений: не выполняется результат «истина» первого решения и результат «ложь» второго.

Основной недостаток метода – недостаточная чувствительность к ошибкам в логических выражениях.

Покрытие решений/условий

Анализ, проведенный выше, показывает, что этому критерию удовлетворяют

тесты:

a = 2, Ь=0, x=4 – путь 1-2-3-4-5-6, условия: 1 да, 2 да, 3 да, 4 да;

a=1, b=1, x=1 – путь 1-2-4-6, условия: 1 нет, 2 нет, 3 нет, 4 нет.

Комбинаторное покрытие условий требует покрыть тестами восемь комбинаций:

1) a>1,b = 0; 5) a = 2,x>1;

2) a>1, b ≠ 0; 6) a = 2, x<1;

3) a<1, b = 0; 7) a ≠ 2, x>1;

4) a<1, b ≠ 0 8) a ≠ 2, x<1.

Эти комбинации можно проверить четырьмя тестами: a = 2, b = 0, x = 4 — проверяет комбинации (1), (5);

a = 2, b = 1, x = 1 — проверяет комбинации (2), (6);

a= 1, b = 0, x = 2 — проверяет комбинации (3), (7);

a=1,Ь=1,x=1— проверяет комбинации (4), (8).

В данном случае то, что четырем тестам соответствуют четыре пути, является совпадением. Представленные тесты не покрывают всех путей, например, acd. Поэтому иногда необходима реализация восьми тестов.

Таким образом, для программ, содержащих только одно условие на каждое решение, минимальным является набор тестов, который проверяет все результаты каждого решения и передает управление каждому оператору, по крайней мере, один раз.

Для программ, содержащих вычисления, каждое из которых требует проверки более чем одного условия, минимальный набор тестов должен:

генерировать все возможные комбинации результатов проверок условий для каждого вычисления;

передавать управление каждому оператору, по крайней мере, один раз.

Термин «возможных» употреблен здесь потому, что некоторые комбинации условий могут быть нереализуемы. Например, для комбинации k<0 и k>40 задать k невозможно.

Тестовое покрытие циклов

Цикл — наиболее распространенная конструкция алгоритмов, реализуемых в программном обеспечении. При проверке циклов основное внимание обращается на правильность конструкций циклов. Количество наборов тестов для проверки циклов по принципу «белого ящика» зависит от типа цикла. Различают 4 типа циклов: простые, вложенные, объединенные, неструктурированные.

Для проверки простых циклов с количеством повторений п может использоваться один из следующих наборов тестов:

прогон всего цикла;

только один проход цикла;

два прохода цикла;

т проходов цикла, где т<п;

п 1, п, п + 1 проходов цикла.

С увеличением уровня вложенности циклов ко астает. Это приводит к нереализуемому количеству тестов. Д

ства тестов применяется специальная мето-

дика, использующая понятия объемлющего и в

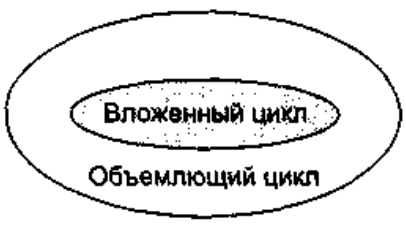

Рис. 4.3. Объемлющий

и вложенный циклы

Порядок тестирования вложенных циклов следующий:

Выбирается самый внутренний цикл. Устанавливаются минимальные значения параметров всех остальных циклов.

Для внутреннего цикла проводятся тесты простого цикла. Добавляются тесты для исключенных значений и значений, выходящих за пределы рабочего диапазона.

Переходят в следующий по порядку объемлющий цикл. Выполняют его тестирование. При этом сохраняются минимальные значения параметров для всех внешних (объемлющих) циклов и типовые значения для всех вложенных циклов.

Работа продолжается до тех пор, пока не будут протестированы все циклы.

Если каждый из циклов независим от других (объединенные циклы), то используется техника тестирования простых циклов. При наличии зависимости (например, конечное значение счетчика первого цикла используется как начальное значение счетчика второго цикла) используется методика для вложенных циклов.

Неструктурированные циклы тестированию не подлежат. Этот тип циклов должен быть переделан с помощью структурированных программных конструкций.

Задачи

Задача 5.1. Определить цикломатическую сложность потоковых графов, представленных на рис.4.4.

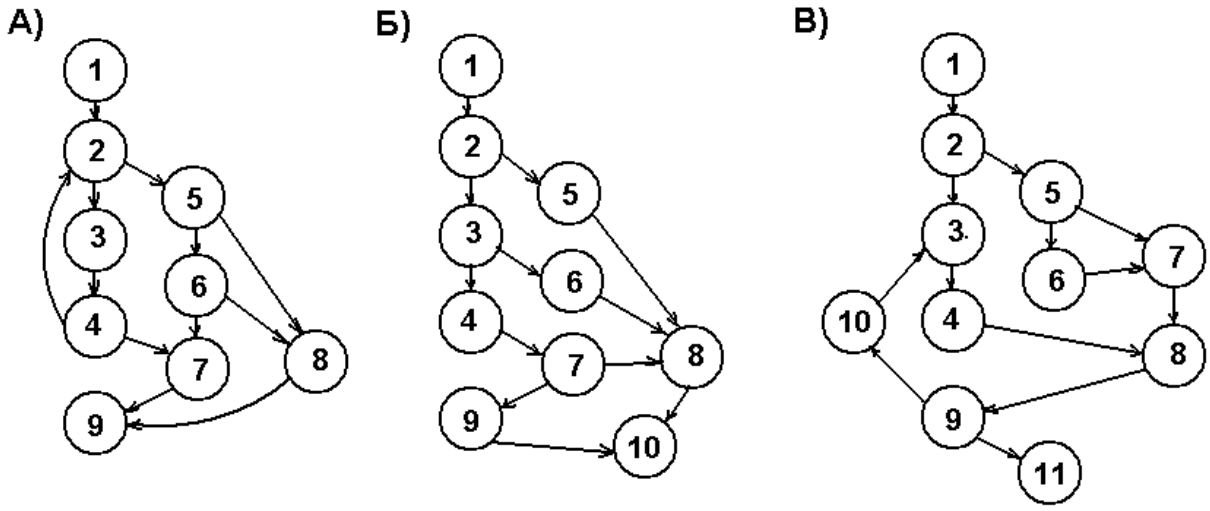

Рис. 4.4. Потоковые графы

Задача 5.2. В соответствии с концепцией максимально полного тестирования всех маршрутов программы определить независимые пути потоковых графов, представленных на рис.4.4.

Задача 5.3. Для заданной процедуры (варианты 1-6) составить потоковый граф, определить цикломатическую сложность потокового графа по каждой из трех формул и составить тестовые наборы по критерию покрытия маршрутов.

Задача 5.4. Для заданной процедуры (варианты 1-6) составить граф-схему и тестовые наборы для тестирования маршрутов по критериям:

покрытия операторов;

покрытия решений (переходов);

покрытия условий;

покрытия решений/условий;

комбинаторного покрытия условий.

Проанализируйте целесообразность каждого из критериев для своей программы, укажите их недостатки, достоинства и преимущества над другими критериями.

Задача 5.5. Определить типы циклов в потоковых графах, представленных на рис.4.4, 4.5.

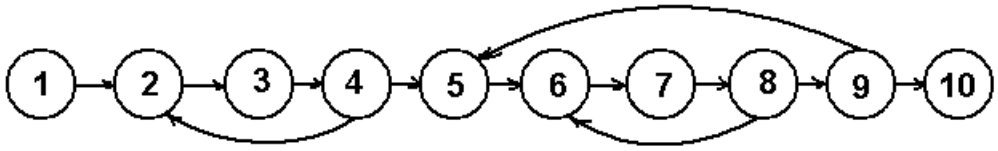

Рис. 4.5. Потоковый граф

Задача 5.5. Сколько наборов тестов необходимо для тестирования программы, потоковый граф которой представлен на рис. 4.5.

Вариант 1.
Варианты заданий для структурного тестирования

```
procedure m(a,b: real; var x: real) begin
if (a>0)and(b<0) then x:=x+1;
if ((a=2)or(x>3))and(b>-10) then x:=x-1;
end;
```

Вариант 2.

```
procedure m(a,b,c: real; var x: real) begin
if (a>0)and(b<0)and(x>6) then x:=x+1; if (a=4)or(c<0) then x:=x+11;
end;
```

Вариант 3.

```
procedure m(a,b: real; var x: real) begin
if (a<=6)and(b<0) then x:=x+1; if (a=7)        then x:=x-1
else if (x>3) then x=x*2;
end;
```

Вариант 4.

```
procedure m(a,b: real; var x: real) begin
if (a>0) then
if(b<0) then x:=x+1
else x=x*2;
if (a>2)or(x=0) then x:=x+1;
end;
```

Вариант 5.

```
procedure m(a,b: real; var x: real) begin
if (a>0)and(b<0)  then x:=x+1
else if ((a=2)or(x>3))and(b>-10) then x:=x-1;
end;
```

Вариант 6.

```
procedure m(a,b,c: real; var x: real) begin
if (a>0)and(b<0)and(x>6)
then x:=x+1
else if (a=4)or(c<0)
then x:=x+11;
end;
```

Контрольные вопросы

Почему стратегия структурного тестирования называется также стратегией "белого ящика"?
Что показывает цикломатическая сложность алгоритма?

В чем отличие критериев покрытия условий и покрытия решений/условий
Какой из критериев "белого ящика" считается самым сильным и почему?
Приведите порядок тестирования вложенных циклов.

ТЕМА 2. РАЗНОВИДНОСТИ ТЕСТИРОВАНИЯ И ОЦЕНКИ КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №5 ИНТЕГРАЦИОННОЕ ТЕСТИРОВАНИЕ

Количество аудиторных часов - 2

Цель работы

Овладение навыками интеграционного тестирования.

Общие сведения

Интеграционное тестирование называют еще тестированием архитектуры системы. С одной стороны, это название обусловлено тем, что интеграционные тесты включают в себя проверки всех возможных видов взаимодействий между программными модулями и элементами, которые определяются в архитектуре системы - таким образом, интеграционные тесты проверяют полноту взаимодействий в тестируемой реализации системы. С другой стороны, результаты выполнения интеграционных тестов - один из основных источников информации для процесса улучшения и уточнения архитектуры системы, межмодульных и межкомпонентных интерфейсов. Т.е., с этой точки зрения, интеграционные тесты проверяют корректность взаимодействия компонент системы.

В результате проведения интеграционного тестирования и устранения всех выявленных дефектов получается согласованная и

целостная архитектура программной системы, т.е. можно считать, что интеграционное тестирование - это тестирование архитектуры и низкоуровневых функциональных требований.

Интеграционное тестирование, как правило, представляет собой итеративный процесс, при котором проверяется совокупность модулей, возрастающая от итерации к итерации. В интеграционном тестирование выделяют три методы выполнения: восходящее тестирование; монолитное тестирование; нисходящее тестирование.

Задание

Согласно варианту провести один из методов интеграционного тестирования.

ТЕМА 2. РАЗНОВИДНОСТИ ТЕСТИРОВАНИЯ И ОЦЕНКИ КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №6 СИСТЕМНОЕ ТЕСТИРОВАНИЕ

Количество аудиторных часов - 4

Цель работы

Овладение навыками системного тестирования.

Общие сведения

Системное тестирование - один из самых сложных видов тестирования. На этапе системного тестирования проводится не только функциональное тестирование, но и оценка характеристик качества системы - ее устойчивости, надежности, безопасности и производительности. На этом этапе выявляются многие проблемы внешних интерфейсов системы, связанные с неверным взаимодействием с другими системами, аппаратным обеспечением, неверным распределением памяти, отсутствием корректного освобождения ресурсов и т.п.

После завершения системного тестирования разработка переходит в фазу приемо-сдаточных испытаний (для программных систем, разрабатываемых на заказ) или в фазу альфа- и бета-тестирования (для программных систем общего применения).

Системное тестирование проводится в несколько фаз, на каждой из которых проверяется один из аспектов поведения системы, т.е. проводится один из типов системного тестирования. Все эти фазы могут протекать одновременно или последовательно. Следующий раздел посвящен рассмотрению особенностей каждого из типов системного тестирования на каждой фазе.

Виды системного тестирования:

)функциональное тестирование;

)тестирование производительности;

)нагрузочное или стрессовое тестирование;

)тестирование конфигурации;

)тестирование безопасности;

)тестирование надежности и восстановления после сбоев;

)тестирование удобства использования.

Задание

Согласно варианту провести несколько видов системного тестирования.

ТЕМА 2. РАЗНОВИДНОСТИ ТЕСТИРОВАНИЯ И ОЦЕНКИ КАЧЕСТВА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ
ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №7 РУЧНОЕ ТЕСТИРОВАНИЕ, ГЕНЕРАЦИЯ ТЕСТОВ

Количество аудиторных часов - 2

Цель работы

Овладение навыками ручного тестирования и составление тестовых случаев.

Общие сведения

Ручное тестирование заключается в выполнении задокументированной процедуры, где описана методика выполнения тестов, задающая порядок тестов и для каждого теста - список значений параметров, который подается на вход, и список результатов, ожидаемых на выходе. Поскольку процедура предназначена для выполнения человеком, в ее описании для краткости могут использоваться некоторые значения по умолчанию, ориентированные на здравый смысл, или ссылки на информацию, хранящуюся в другом документе.

Описание тестов разрабатывается для облегчения анализа и поддержки тестового набора. Описание может быть реализовано в произвольной форме, но при этом должны выполнять следующие задачи:

.Анализировать степень покрытия продукта тестами на основании описания тестового набора.

.Для любой функции тестируемого продукта найти тесты, в которых функция используется.

.Для любого теста определить все функции и их сочетания, которые данный тест использует (затрагивает).

.Понять структуру и взаимосвязи тестовых файлов. 5.Понять принцип построения системы автоматизации

тестирования

Задание

Подготовить тестовый случай, выполнить и задокументировать результаты.

ТЕМА 3. ОСОБЕННОСТИ ПРОЦЕССА И ТЕХНОЛОГИИ ИНДУСТРИАЛЬНОГО ТЕСТИРОВАНИЯ ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №8 ДОКУМЕНТАЦИЯ

Количество аудиторных часов - 2

Цель работы

Овладение навыками документирования результатов тестирования.

Общие сведения

Каждый дефект, обнаруженный в процессе тестирования, должен быть задокументирован и отслежен. При обнаружении нового дефекта его заносят в базу дефектов. При занесении нового дефекта рекомендуется указывать, как минимум, следующую информацию:

Наименование подсистемы, в которой обнаружен дефект.

Версия продукта (номер build), на котором дефект был найден.

Описание дефекта.

Описание процедуры (шагов, необходимых для воспроизведения дефекта).

Номер теста, на котором дефект был обнаружен.

Уровень дефекта, то есть степень его серьезности с точки зрения критериев качества продукта или заказчика.

Тестовый отчет обновляется после каждого цикла тестирования и должен содержать следующую информацию для каждого цикла:

Перечень функциональности в соответствии с пунктами требований, запланированный для тестирования на данном цикле, и реальные данные по нему.

Количество выполненных тестов – запланированное и реально исполненное.

Время, затраченное на тестирование каждой функции, и общее время тестирования.

Количество найденных дефектов.

Количество повторно открытых дефектов.

Отклонения от запланированной последовательности действий, если таковые имели место.

Выводы о необходимых корректировках в системе тестов, которые должны быть сделаны до следующего тестового цикла.

Задание

Задокументировать результаты тестирования. Для выполнения работы использовать тестовые случае из лабораторной работы №5.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Управление качеством программного обеспечения

Методические указания к самостоятельным работам

Направление подготовки 09.03.02 Информационные системы и технологии

Направленность (профиль) Информационные системы управления технологическими и сервисными процессами

Квалификация выпускника – бакалавр

(Электронный документ)

Невинномысск 2026

Содержание

1 Подготовка к лекциям

Главное в период подготовки к лекционным занятиям – научиться методам самостоятельного умственного труда, сознательно развивать свои творческие способности и овладевать навыками творческой работы. Для этого необходимо строго соблюдать дисциплину учебы и поведения. Четкое планирование своего рабочего времени и отдыха является необходимым условием для успешной самостоятельной работы. В основу его нужно положить рабочие программы изучаемых в семестре дисциплин.

Каждому студенту следует составлять еженедельный и семестровый планы работы, а также план на каждый рабочий день. С вечера всегда надо распределять работу на завтрашний день. В конце каждого дня целесообразно подводить итог работы: тщательно проверить, все ли выполнено по намеченному плану, не было ли каких-либо отступлений, а если были, по какой причине это произошло. Нужно осуществлять самоконтроль, который является необходимым условием успешной учебы. Если что-то осталось невыполненным, необходимо изыскать время для завершения этой части работы, не уменьшая объема недельного плана.

Слушание и запись лекций – сложный вид вузовской аудиторной работы. Внимательное слушание и конспектирование лекций предполагает интенсивную умственную деятельность студента. Краткие записи лекций, их конспектирование помогает усвоить учебный материал. Конспект является полезным тогда, когда записано самое существенное, основное и сделано это самим студентом. Не надо стремиться записать дословно всю лекцию. Такое «конспектирование» приносит больше вреда, чем пользы. Запись лекций рекомендуется вести по возможности собственными формулировками. Желательно запись осуществлять на одной странице, а следующую оставлять для проработки учебного материала самостоятельно в домашних условиях.

Конспект лекций лучше подразделять на пункты, параграфы, соблюдая красную строку. Этому в большой степени будут способствовать пункты плана лекции, предложенные преподавателям. Принципиальные места,

определения, формулы и другое следует сопровождать замечаниями «важно», «особо важно», «хорошо запомнить» и т.п. Можно делать это и с помощью разноцветных маркеров или ручек. Лучше если они будут собственными, чтобы не приходилось присить их у однокурсников и тем самым не отвлекать их во время лекции. Целесообразно разработать собственную «маркографию» (значки, символы), сокращения слов. Не лишним будет и изучение основ стенографии. Работая над конспектом лекций, всегда необходимо использовать не только учебник, но и ту литературу, которую дополнительно рекомендовал лектор. Именно такая серьезная, кропотливая работа с лекционным материалом позволит глубоко овладеть знаниями.

2 Подготовка к практическим занятиям

Подготовку к каждому практическому занятию студент должен начать с ознакомления с методическими указаниями, которые включают содержание работы. Тщательное продумывание и изучение вопросов основывается на проработке текущего материала лекции, а затем изучения обязательной и дополнительной литературы, рекомендованную к данной теме. На основе индивидуальных предпочтений студенту необходимо самостоятельно выбрать тему доклада по проблеме и по возможности подготовить по нему презентацию.

Если программой дисциплины предусмотрено выполнение практического задания, то его необходимо выполнить с учетом предложенной инструкции (устно или письменно). Все новые понятия по изучаемой теме необходимо выучить наизусть и внести в глоссарий, который целесообразно вести с самого начала изучения курса. Результат такой работы должен проявиться в способности студента свободно ответить на теоретические вопросы семинара, его выступлении и участии в коллективном обсуждении вопросов изучаемой темы, правильном выполнении практических заданий и контрольных работ.

В зависимости от содержания и количества отведенного времени на изучение каждой темы практическое занятие может состоять из четырех-пяти частей:

1. Обсуждение теоретических вопросов, определенных программой дисциплины.

2. Доклад и/ или выступление с презентациями по выбранной проблеме.

3. Обсуждение выступлений по теме – дискуссия.

4. Выполнение практического задания с последующим разбором полученных результатов или обсуждение практического задания.

5. Подведение итогов занятия.

Первая часть – обсуждение теоретических вопросов – проводится в виде фронтальной беседы со всей группой и включает выборочную проверку

преподавателем теоретических знаний студентов. Примерная продолжительность — до 15 минут. Вторая часть — выступление студентов с докладами, которые должны сопровождаться презентациями с целью усиления наглядности восприятия, по одному из вопросов практического занятия. Обязательный элемент доклада – представление и анализ статистических данных, обоснование социальных последствий любого экономического факта, явления или процесса. Примерная продолжительность — 20-25 минут. После докладов следует их обсуждение – дискуссия. В ходе этого этапа практического занятия могут быть заданы уточняющие вопросы к докладчикам. Примерная продолжительность – до 15-20 минут. Если программой предусмотрено выполнение практического задания в рамках конкретной темы, то преподавателями определяется его содержание и дается время на его выполнение, а замет идет обсуждение результатов. Подведением итогов заканчивается практическое занятие.

В процессе подготовки к практическим занятиям, студентам необходимо обратить особое внимание на самостоятельное изучение рекомендованной учебно-методической (а также научной и популярной) литературы. Самостоятельная работа с учебниками, учебными пособиями, научной, справочной и популярной литературой, материалами периодических изданий и Интернета, статистическими данными является наиболее эффективным методом получения знаний, позволяет значительно активизировать процесс овладения информацией, способствует более глубокому усвоению изучаемого материала, формирует у студентов свое отношение к конкретной проблеме. Более глубокому раскрытию вопросов способствует знакомство с дополнительной литературой, рекомендованной преподавателем по каждой теме семинарского или практического занятия, что позволяет студентам проявить свою индивидуальность в рамках выступления на данных занятиях, выявить широкий спектр мнений по изучаемой проблеме.

3 Самостоятельное изучение темы. Конспект

Конспект – наиболее совершенная и наиболее сложная форма записи. Слово «конспект» происходит от латинского «conspectus», что означает «обзор, изложение». В правильно составленном конспекте обычно выделено самое основное в изучаемом тексте, сосредоточено внимание на наиболее существенном, в кратких и четких формулировках обобщены важные теоретические положения.

Конспект представляет собой относительно подробное, последовательное изложение содержания прочитанного. На первых порах целесообразно в записях ближе держаться тексту, прибегая зачастую к прямому цитированию автора. В дальнейшем, по мере выработки навыков конспектирования, записи будут носить более свободный и сжатый характер.

Конспект книги обычно ведется в тетради. В самом начале конспекта указывается фамилия автора, полное название произведения, издательство, год и место издания. При цитировании обязательная ссылка на страницу книги. Если цитата взята из собрания сочинений, то необходимо указать соответствующий том. Следует помнить, что четкая ссылка на источник – непременное правило конспектирования. Если конспектируется статья, то указывается, где и когда она была напечатана.

Конспект подразделяется на части в соответсвтии с заранее продуманным планом. Пункты плана записываются в тексте или на полях конспекта. Писать его рекомендуется четко и разборчиво, так как небрежная запись с течением времени становиться малопонятной для ее автора. Существует правило: конспект, составленный для себя, должен быть по возможности написан так, чтобы его легко прочитал и кто-либо другой.

Формы конспекта могут быть разными и зависят от его целевого назначения (изучение материала в целом или под определенным углом зрения, подготовка к докладу, выступлению на занятии и т.д.), а также от характера произведения (монография, статья, документ и т.п.). Если речь идет просто об изложении содержания работы, текст конспекта может быть сплошным, с

выделением особо важных положений подчеркиванием или различными значками.

В случае, когда не ограничиваются переложением содержания, а фиксируют в конспекте и свои собственные суждения по данному вопросу или дополняют конспект соответствующими материалами их других источников, следует отводить место для такого рода записей. Рекомендуется разделить страницы тетради пополам по вертикали и в левой части вести конспект произведения, а в правой свои дополнительные записи, совмещая их по содержанию.

Конспектирование в большей мере, чем другие виды записей, помогает вырабатывать навыки правильного изложения в письменной форме важные теоретических и практических вопросов, умение четко их формулировать и ясно излагать своими словами.

Таким образом, составление конспекта требует вдумчивой работы, затраты времени и труда. Зато во время конспектирования приобретаются знания, создается фонд записей.

Конспект может быть текстуальным или тематическим. В текстуальном конспекте сохраняется логика и структура изучаемого произведения, а запись ведется в соответствии с расположением материала в книге. За основу тематического конспекта берется не план произведения, а содержание какой-либо темы или проблемы.

Текстуальный конспект желательно начинать после того, как вся книга прочитана и продумана, но это, к сожалению, не всегда возможно. В первую очередь необходимо составить план произведения письменно или мысленно, поскольку в соответствии с этим планом строится дальнейшая работа. Конспект включает в себя тезисы, которые составляют его основу. Но, в отличие от тезисов, конспект содержит краткую запись не только выводов, но и доказательств, вплоть до фактического материала. Иначе говоря, конспект – это расширенные тезисы, дополненные рассуждениями и доказательствами, мыслями и соображениями составителя записи.

Как правило, конспект включает в себя и выписки, но в него могут войти отдельные места, цитируемые дословно, а также факты, примеры, цифры, таблицы и схемы, взятые из книги. Следует помнить, что работа над конспектом только тогда будет творческой, когда она не ограничена текстом изучаемого произведения. Нужно дополнять конспект данными из другими источников.

В конспекте необходимо выделять отдельные места текста в зависимости от их значимости. Можно пользоваться различными способами: подчеркиваниями, вопросительными и восклицательными знаками, репликами, краткими оценками, писать на полях своих конспектов слова: «важно», «очень важно», «верно», «характерно».

В конспект могут помещаться диаграммы, схемы, таблицы, которые придадут ему наглядность.

Составлению тематического конспекта предшествует тщательное изучение всей литературы, подобранной для раскрытия данной темы. Бывает, что какая-либо тема рассматривается в нескольких главах или в разных местах книги. А в конспекте весь материал, относящийся к теме, будет сосредоточен в одном месте. В плане конспекта рекомендуется делать пометки, к каким источникам (вплоть до страницы) придется обратиться для раскрытия вопросов. Тематический конспект составляется обычно для того, чтобы глубже изучить определенный вопрос, подготовиться к докладу, лекции или выступлению на семинарском занятии. Такой конспект по содержанию приближается к реферату, докладу по избранной теме, особенно если включает и собственный вклад в изучение проблемы.

4 Методические рекомендации по подготовке и прохождению тестирования

Тесты – это вопросы или задания, предусматривающие конкретный, краткий, четкий ответ на имеющиеся ответы.

При самостоятельной подготовке к тестированию студенту необходимо:

а) проработать информационный материал по дисциплине, предварительно проконсультироваться с ведущим преподавателем по вопросам выбора учебной литературы;

б) выяснить условия тестирования: количество тестовых заданий, количество времени на выполнение тестов, система оценки результатов;

в) приступая к работе с тестами, внимательно и до конца прочтите вопрос и предлагаемые варианты ответов. Выберите правильные (их может быть несколько). На отдельном листке ответов выпишите цифру вопроса и буквы, соответствующие правильным ответам;

г) обязательно оставьте время для проверки ответов, чтобы избежать возможных ошибок.