Приложение 1

Федеральное государственное бюджетное образовательное учреждение
высшего образования

РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА и ГОСУДАРСТВЕННОЙ СЛУЖБЫ при ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

СЕВЕРО-ЗАПАДНЫЙ ИНСТИТУТ УПРАВЛЕНИЯ

ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

Утвержден решением цикловой (методической) комиссией по специальности 09.02.07 «Информационные системы и программирование»

Протокол № 1

от « 25 » декабря 2022 г

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

МДК.02.02 Инструментальные средства разработки программного обеспечения

по специальности 09.02.07 «Информационные системы и программирование

Квалификация – специалист по информационным системам

Форма обучения - очная

Год набора - 2022

Санкт-Петербург, 2022 год

Автор(ы)–составитель(и):

доцент кафедры бизнес-информатики, кандидат технических наук
Буров Сергей Александрович

Заведующий кафедрой:

доктор военных наук, профессор Наумов Владимир Николаевич

СОДЕРЖАНИЕ

1. Перечень планируемых результатов обучения по дисциплине- перечень компетенций с указанием компонентов компетенций дисциплины, как отдельного элемента ОП

Код компетенции	Наименование компетенции	Код компонента компетенции	Наименование компонента компетенции
ОК-1	Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам	ОК-1.1	уметь обосновывать постановку цели, выбор и применения методов и способов решения профессиональных задач, связанных с разработкой программного обеспечения
ОК-2	Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.	ОК-2.1 ОК-2.2 ОК-2.3	знать технологии сбора, накопления, обработки, передачи и распространения информации уметь обрабатывать текстовую и числовую информацию уметь применять мультимедийные технологии обработки и представления информации
ОК-3	Планировать и реализовывать собственное профессиональное и личностное развитие.	ОК-3.1	уметь обосновывать самоанализ и корректировать результаты собственной работы
ОК-4	Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами	ОК-4.1 ОК-4.2	уметь взаимодействовать с обучающимися, преподавателями в ходе обучения уметь обосновывать анализ работы членов команды;
ОК-5	ОК 5 Осуществлять устную и письменную коммуникация на государственном языке с учетом особенностей социального и культурного контекста	ОК-5.1	уметь ясно формулировать и излагать мысли
ОК-6	Проявлять гражданско-патриотическую позицию, демонстрировать осознание поведения на основе традиционных общечеловеческих ценностей	ОК-6.1	соблюдать нормы поведения во время учебных занятий

Код компетенции	Наименование компетенции	Код компонента компетенции	Наименование компонента компетенции
ОК-7	Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях.	ОК-7.1 ОК-7.2	знать основы применения ресурсосберегающих технологий в профессиональной деятельности выполнять правила техники безопасности во время учебных занятий
ОК-8	Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности	ОК-8.1	эффективно использовать средств физической культуры для сохранения и укрепления здоровья при выполнении профессиональной деятельности
ОК-9	Использовать информационные технологии в профессиональной деятельности.	ОК-9.1 ОК-9.2	знать состав, структуру, принципы реализации и функционирования информационных технологий знать инструментальные средства информационных технологий
ОК-10	Пользоваться профессиональной документацией на государственном и иностранном языке	ОК-10.1	уметь использовать необходимую техническую документацию по проектированию программного обеспечения, в том числе на английском языке
ОК-11	Планировать предпринимательскую деятельность в профессиональной сфере	ОК-11.1	уметь обосновывать применение методов и способов решения задач по проектированию программного обеспечения при планировании предпринимательской деятельности

Код компетенции	Наименование компетенции	Код компонента компетенции	Наименование компонента компетенции
ПК-2.2	Выполнять интеграцию модулей в программное обеспечение	ПК-2.2.1	знать методы и принципы интеграции модулей в программное обеспечение
		ПК-2.2.2	уметь выполнять интеграцию модуля в программное обеспечение в инструментальной среде программирования
ПК-2.3	Выполнять отладку программного модуля с использованием специализированных программных средств	ПК-2.3.1	знать основные методы отладки программного обеспечения
		ПК-2.3.2	уметь выполнять отладку программного модуля с использованием системы контроля версий
ПК-2.5	Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования	ПК-2.5.1	знать основные стандарты кодирования языка программирования
		ПК-2.5.2	уметь находить в коде программного продукта несоответствие стандартам языка программирования

2. Оценочные средства- представление полного комплекта контрольных заданий и иных материалов, необходимых для оценки планируемых результатов обучения по дисциплине (модулю) для проведения текущей и промежуточной аттестации. 2.1 Оценочные средства по дисциплине для текущего контроля

2.1 Оценочные средства по дисциплине для текущего контроля

2.1.1 Компетентностно-ориентированные задания

Задание № 1 «Разработка структуры проекта»

Цель работы: Формирование навыков постановки задачи и разработки технического задания на программный продукт.

Задание:

1) Выбрать вариант задания на проектирование и разработку учебной программы.

2) В соответствии с вариантом выполнить разработку технического задания, которое должно включать:

- введение;
- основание для разработки;
- назначение;
- требования к программе и программному продукту;
- требования к программной документации.

3. Оформить отчет.

Варианты заданий:

1) Ввести вещественную матрицу размерности n * m построчно, а вывести по столбцам.

2) Выяснить сколько положительных элементов содержит матрица размерности n * m , если a ij = sin(i+j/2).

3) Дана квадратная вещественная матрица размерности n. Является ли матрица симметричной относительно главной диагонали.

4) Дана квадратная вещественная матрица размерности n. Транспонировать матрицу.

5) Дана квадратная вещественная матрица размерности n. Сравнить сумму элементов матрицы на главной и побочной диагоналях.

6) Дана квадратная вещественная матрица размерности n. Найти количество нулевых элементов, стоящих:

выше главной диагонали; ниже главной диагонали; выше и ниже побочной.

7) Дана вещественная матрица размерности n * m. По матрице получить логический вектор, присвоив его k-ому элементу значение True , если выполнено указанное условие и значение False иначе:

все элементы k столбца нулевые;

элементы k строки матрицы упорядочены по убыванию; k строка массива симметрична.

8) Дана вещественная матрица размерности n * m. Сформировать вектор b, в котором элементы вычисляются как:

произведение элементов соответствующих строк; среднее арифметическое соответствующих столбцов;

разность наибольших и наименьших элементов соответствующих строк; значения первых отрицательных элементов в столбце.

9) Дана вещественная матрица размерности n * m. Вывести номера столбцов, содержащих только отрицательные элементы.

10) Дана вещественная матрица размерности n * m. Вывести номера строк, содержащих больше положительных элементов, чем отрицательных.

11) Дана вещественная матрица размерности n * m. Найти общую сумму элементов только тех столбцов, которые имеют хотя бы один нулевой элемент.

12) Дана вещественная матрица размерности n * m. Поменять местами строки с максимальным и минимальным элементами.

13) Дана вещественная матрица размерности n * m. Удалить k столбец матрицы.

14) Дана вещественная квадратная матрица размерности n. Поменять местами элементы главной и побочной диагоналей матрицы:

по строкам; по столбцам.

15) Дана вещественная матрица размерности m * n. Упорядочить элементы каждой четной строки по возрастанию.

16) Дана вещественная матрица размерности m * n. Расположить все элементы матрицы по убыванию. Обход матрицы осуществлять по строкам.

17) Дана вещественная матрица размерности m * n. Определить индексы первого нулевого элемента матрицы. Обход матрицы осуществлять по столбцам.

18) Известно положение двух ферзей на шахматной доске. Бьют ли они друг друга?

Содержание отчета

1) Титульный лист.

2) Наименование и цель работы.

3) Краткое теоретическое описание.

4) Задание.

5) Листинг программы.

6) Результаты выполнения программы.

Задание № 2 «Разработка модульной структуры проекта (диаграммы модулей)»

Разработка эскизного проекта

Эскизный проект возникает как результат анализа требований, предъявленных к программному продукту. В нем в общем виде формулируются указания по созданию программного продукта. Здесь ставится задача для каждого разработчика, описываются алгоритм решения задачи, способы взаимодействия создаваемого продукта с другими программами и устройствами ввода- вывода, выбираются структуры данных, определяются способы хранения данных на диске или в базе данных.

Эскизный проект не может быть слишком большим. Он должен быть обозримым, схематичным, четко показывающим основные этапы создания программного продукта. Обычно эскизный проект содержит не больше 5— 6 страниц текста. К нему прилагаются диаграммы, рисунки и чертежи, а также календарный план выполнения проекта.

После того как эскизный проект создан, он раздается всем участникам разработки для изучения и обсуждения. Каждый разработчик обдумывает свой участок проекта, вносит свои предложения и дополнения, конкретизирует план выполнения проекта.

Разработка технического проекта

После изучения эскизного проекта всеми заинтересованными лицами наступает время создания технического проекта. В его обсуждении принимает участие вся команда разработчиков под руководством менеджера проекта. Каждый разработчик вносит свои предложения по реализации и улучшению проекта, уточняет и детализирует относящиеся к нему положения проекта, согласует интерфейсы с другими разработчиками.

Технический проект будет рабочим документом на все время реализации проекта, поэтому он должен быть понятен и приемлем для всех программистов. В нем не должно быть недомолвок, двусмысленностей, не должно оставаться пробелов и недоговоренностей.

При разработке технического проекта окончательно определяется конфигурация технических средств, и вся дальнейшая работа ведется с учетом этой конфигурации. Уточняется операционная среда, в которой будет функционировать программный продукт, и системное программное обеспечение. Например, Web-приложение работает в браузере. Браузеры по-разному интерпретируют языки HTML и JavaScript, поэтому надо сразу решить, будет ли программный продукт рассчитан на определенный браузер или он должен работать в любом. В первом случае разработчики могут включить в продукт дополнительные возможности языков HTML и JavaScript, интерпретируемые данным браузером, во втором — должны использовать только стандартные конструкции, что может значительно затруднить разработку.

техническом проекте уточняются типы и структуры исходных и промежуточных данных, полностью детализируется алгоритм решения задачи. Задача разбивается на модули, которые распределяются среди программистов.

При объектно-ориентированном проектировании в техническом проекте определяются все объекты, необходимые для осуществления проекта и выявляются связи между ними. Полностью выписывается строение каждого объекта, его поля и методы. Объекты записываются в виде интерфейсов или абстрактных классов, дальнейшая разработка которых поручается конкретным программистам.

После проработки технического проекта каждым участником разработки собираются и обобщаются их уточнения и замечания. Окончательная версия проекта обсуждается командой разработчиков. Менеджер проекта выносит технический проект на утверждение руководством фирмы-разработчика и заказчиком программного продукта. После этого технический проект стано- вится рабочим проектом для группы разработчиков.

Рабочий проект

После утверждения технического проекта он становится основным рабочим документом для команды разработчиков программного продукта. Рабочий проект — это большой, подробный документ, наиболее полно описывающий будущий программный продукт и план его создания. В нем содержатся детальные указания каждому разработчику и команде в целом, определена структура базы данных и других хранилищ данных, которой будут руководствоваться все разработчики. Короче говоря, в рабочем проекте должны содержаться все сведения, нужные каждому разработчику и команде в целом. В частности, в нем должны быть записаны этапы и сроки разработки, чтобы каждый программист твердо знал их.

При объектно-ориентированном проектировании в рабочем проекте должны быть полностью описаны все классы и связи между ними. Это описание можно сделать в виде абстрактных классов или интерфейсов, на языке разработки или на языке описания. Важно, чтобы все участники проекта правильно понимали эту запись и одинаково интерпретировали ее.

Каждому участнику проекта выдается экземпляр рабочего проекта. При всяком изменении рабочего проекта участники получают его новую версию. В настоящее время с развитием Web- технологии, как правило, создается собственный сайт для каждого проекта. Все рабочие документы публикуются на этом сайте, а при каждом их изменении участники проекта получают уведомление по электронной почте.

Упражнения

1. Разработайте проект автоматизации библиотечного каталога.
2. Проведите анализ работы деканата и разработайте проект его автоматизации.
3. Проанализируйте информационные потоки вашего факультета и спроектируйте компьютерную систему их обработки.

Задание № 3 «Разработка перечня артефактов и протоколов проекта»

Системный анализ и пути решения задачи

При разработке ПС человек имеет дело с системами. Под системой будем понимать совокупность взаимодействующих (находящихся в отношениях) друг с другом элементов. ПС можно рассматривать как пример системы. Логически связанный набор программ является другим примером системы. Любая отдельная программа также является системой. Понять систему — значит осмысленно перебрать все пути взаимодействия между ее элементами.

Целью системного анализа в наиболее общем виде является описание и исследование систем, определение путей и методов разработки ПО. Система характеризуется структурой и поведением. Применительно к разработке ПО системный анализ представляет собой анализ существующей структуры отношений в рамках конкретной предметной области, выявление роли и места будущей программной системы, ее основных функций и свойств. В этой связи системный анализ также можно назвать внешним проектированием.

Этап системного анализа состоит из следующих трех стадий:

1) обоснование необходимости разработки программы;

2) научно-исследовательские работы (НИР);

3) разработка и утверждение технического задания.

На первой стадии выполняются постановка задачи, сбор исходных материалов, Выбор и обоснование критериев эффективности и качества разрабатываемой программы, обоснование необходимости проведения научно-исследовательских работ.

На стадии научно-исследовательских работ решаются следующие задачи: определяется структура входных и выходных данных, осуществляется предварительный выбор методов решения задач, обосновывается целесообразность применения ранее разработанных программ, определяются требования к техническим средствам, обосновывается принципиальная возможность решения поставленной задачи.

На стадии разработки и утверждения технического задания определяются требования к программе, разрабатываются технико-экономического обоснования разработки программ, определяются стадии, этапы и сроки разработки программы и документации на нее, согласовывается и утверждаетсятехническое задание.

Результат системного анализа — спецификация (техническое задание) как самостоятельный документ имеет очень важное значение. Этот документ является формальным соглашением между заказчиком продукта и его разработчиками.

В настоящее время можно выделить пять основных подходов к организации процесса создания и использования программного обеспечения.

1) Водопадный подход. При таком подходе разработка ПС состоит из цепочки этапов. На каждом этапе создаются документы, используемые на последующем этапе. В исходном документе фиксируются требования к ПС. В конце этой цепочки создаются программы, включаемые в ПС.

2) Исследовательское программирование. Этот подход предполагает быструю (насколько это возможно) реализацию рабочих версий программ ПС, выполняющих лишь в первом приближении требуемые функции. После экспериментального применения реализованных программ производится их модификация с целью сделать их более полезными для пользователей. Этот процесс повторяется до тех пор, пока ПС не будет достаточно приемлемо для пользователей. Такой подход применялся на ранних этапах развития программирования, когда технологии программирования не придавали большого значения (использовалась интуитивная технология). В настоящее время этот подход применяется для разработки таких ПС, для которых пользователи не могут точно сформулировать требования (например, для разработки систем искусственного интеллекта).

3) Прототипирование. Этот подход моделирует начальную фазу исследовательского программирования вплоть до создания рабочих версий программ, предназначенных для проведения экспериментов с целью установить требования к ПС. В дальнейшем должна последовать разработка ПС по установленным требованиям в рамках какого-либо другого подхода (например, водопадного).

4) Формальные преобразования. Этот подход включает разработку формальных спецификаций ПС и превращение их в программы путем корректных преобразований.

На этом подходе базируется компьютерная технология (CASE-технология) разработки ПС.

5) Сборочное программирование. Этот подход предполагает, что ПС конструируется, главным образом, из компонентов, которые уже существуют. Должно быть некоторое хранилище (библиотека) таких компонентов, каждая из которых может многократно использоваться в разных ПС. Такие компоненты называются повторно используемыми(reusable). Процесс разработки ПС при данном подходе состоит скорее из сборки программ из компонентов, чем из их программирования.

Задание:

1) В соответствии с подготовленным техническим заданием выполнить разработку спецификаций на программный продукт, которые должны включать:

- спецификации процессов;
- словарь терминов;
- диаграммы переходов состояний;
- диаграммы потоков с детализацией.

2) Оформить отчет.

Задание № 4. «Настройка работы системы контроля версий (типов импортируемых файлов, путей, фильтров и др. параметров импорта в репозиторий)»

Система управления/контроля версиями (от англ. Version Control System или Revision Control System) — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости, возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение и многое другое.

Такие системы наиболее широко применяются при разработке программного обеспечения, для хранения исходных кодов разрабатываемой программы. Однако, они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов, в частности, они всё чаще применяются в САПР, обычно, в составе систем управления данными об изделии (PDM). Управление версиями используется в инструментах конфигурационного управления (Software Configuration Management Tools).

Распространённые системы управления версиями

−Subversion

−Darcs

−Microsoft Visual SourceSafe

−Bazaar

−Rational ClearCase

−Perforce

−BitKeeper

−Mercurial

−Git

−GNU Arch

−CVS — устаревшая. Потомок: Subversion

−RCS — устаревшая. Потомок: CVS Основные понятия

Репозиторий (repository) – центральное хранилище, которое содержит версии файлов. Очень часто репозиторий организуется средствами какой-нибудь СУБД.

Версия файла (revision) – состояние файла в определенный момент времени. Репозиторий предоставляет возможность хранить неограниченное число версий одного и того же файла.

Актуальная версия файла – обычно это самая последняя версия файла, размещенного в репозитории.

Рабочая версия файла (working copy) – версия файла, с которой в текущий момент ведется работа, и которая не загружена в репозиторий.

Загрузка (Upload) – размещение файла в репозитории. В процессе загрузки в репозиторий помещается рабочая версия файла.

Выгрузка (Checkout) – получение файла из репозитория. В процессе выгрузки осуществляется получение из репозитория необходимой версии файла.

Синхронизация (update, sync) – приведение в соответствие рабочих версий файлов с актуальными версиями в репозитории. В процессе синхронизации в репозиторий загружаются те файлы, рабочие копии которых являются более "свежими" (т.е. имеют более поздние версии), по сравнению с файлами в репозитории, и выгружаются те файлы, рабочие копии которых устарели по сравнению с копиями в репозитории.

Borland StarTeam

Borland StarTeam – очень мощный и функциональный кросс-платформенный продукт, разрабатываемый в прошлом фирмой StarBase, которую Borland приобрела в конце 2002 г. Заметное преимущество данного решения состоит в том, что версия 2005 выступает центральным элементом стратегии управления жизненным циклом приложений (Application Lifecycle Management, ALM) компании Borland и обладает расширенными возможностями интеграции со всеми ее ключевыми пакетами, используемыми при разработке программного обеспечения.

MS SourceSafe

Microsoft Visual SourceSafe (Visual SourceSafe, VSS) — программный продукт компании Майкрософт, файл-серверная система управления версиями, предназначенная для небольших команд разработчиков. VSS позволяет хранить в общем хранилище файлы, разделяемые несколькими пользователями, для каждого файла хранится история версий. VSS входит в состав пакета Microsoft Visual Studio и интегрирован с продуктами этого пакета. Доступен только для

платформы Windows. Версию для Unix поддерживает компания MainSoft. В ноябре 2005 года вышла обновлённая версия продукта — Visual SourceSafe 2005, обещающая повышенную стабильность и производительность, улучшенный механизм слияния для XML-файлов и файлов в Юникоде, а также работу через HTTP. Visual SourceSafe нацелен на индивидуальных разработчиков либо небольшие команды разработчиков. Там где VSS недостаточно, ему на замену предлагается новый продукт Майкрософт — Team Foundation Server, входящий в состав Visual Studio Team System.

Rational Clear Case

ClearCase поддерживает следующие возможности, разительно отличающие его в лучшую сторону от других средств контроля:

–Общий контроль версий не только файлов, но и директорий/поддиректорий;

–Бесконечное число ответвлений от определенной версии;

–Автоматическая компрессия файлов и их кеширование (CC позволяет хранить большое количество данных, при всем при этом база данных остается компактной и быстрой);

–Позволяет легко конвертировать базы данных других средств контроля, например: PVCS, SourceSafe, RCS, CVS и SCCS;

–Поддерживает параллельную разработку и мультикомандные подразделения, расположенные в географически удаленных друг от друга местах;

–Мультиплатформенность (способен объединить единой средой участников, работающих на разных операционных системах);

–Имеет интеграцию со средствами разработки;

–Имеет Web-интерфейс для удаленного контроля.

CVS

CVS (Concurrent Versions System, "Система Конкурирующих Версий"). Хранит историю изменений определённого набора файлов, как правило исходного кода программного обеспечения, и облегчает совместную работу группы людей (часто — программистов) над одним проектом. CVS популярна в мире открытого ПО. Система распространяется на условиях лицензии GNU GPL.

Subversion

Subversion — централизованная система (в отличие от распределённых систем, таких, как Git или Mercurial), то есть данные хранятся в едином хранилище. Хранилище может располагаться на локальном диске или на сетевом сервере. Работа в Subversion мало отличается от работы в других централизованных системах управления версиями. Для совместной работы над файлами в Subversion преимущественно используется модель Копирование-Изменение-Слияние. Кроме того, для файлов, не допускающих слияние (различные бинарные форматы файлов), можно использовать модель Блокирование-Изменение-Разблокирование.

Задание:

1) Настроить подключение к репозиторию
2) Скачать проект
3) Добавить свой класс к проекту
4) Внести изменения в класс
5) Обновить класс в репозитории
6) Удалить все локальные файлы и скачать проект из репозитория
7) Добавить "лишний" файл в репозиторий и затем удалить его из репозитория.
8) Изучить журнал изменений файлов, посмотреть какие изменения внесены другими разработчиками.

Примечание: опробовать Git, Subversion, Mercurial (локально) Ссылки на учебные репозитории:

• Git
https://github.com/irgups/project_2015_01.git
https://github.com/irgups/project_2015_02.git
• Subversion
https://github.com/irgups/project_2015_01
https://github.com/irgups/project_2015_02

Задание № 5 «Разработка и интеграция модулей проекта (командная работа)»

Цель работы. Освоить процесс проектирования модулей программного обеспечения.

Задание:
1) Описать этапы проектирования модулей программы.
2) Составить в виде блок-схемы алгоритм решения задачи.
3) Составить отчет по практической работе.

Отчет по практической работе должен включать:
1) Алгоритм решения задачи.
2) Набор тестов для отладки программы.

Задача. Составить алгоритм решения задачи, приведенной ниже, с использованием структурных единиц: процедур и/или функций.

Варианты индивидуальных заданий.

1) Даны два двумерных массива вещественных элементов. Размер исходных массивов не превосходит 10x10 элементов. Для каждого из массивов указать номера столбцов, содержащих только положительные элементы. Если таковых столбцов в массиве нет, то вывести соответствующее сообщение. Проверку столбца на положительность элементов оформить в виде процедуры с передачей в нее всех элементов текущего столбца.

2) Даны два двумерных массива натуральных элементов. Размер исходных массивов не превосходит 10x10 элементов. Для каждого из массивов указать номера столбцов, содержащих только кратные 5 или 7 элементы. Если таких столбцов в массиве нет, то вывести

соответствующее сообщение. Проверку столбца на наличие указанных элементов оформить в виде процедуры с передачей в нее всех элементов текущего столбца.

3) Даны пять одномерных массива вещественных элементов. Размер каждого массива не превосходит 100 элементов. Для каждого из массивов определить, составляют ли его элементы знакочередующуюся последовательность. Если да, то указать порядковый номер такого массива, в противном случае вывести отрицательный ответ. Проверку массива на выполнение условия оформить в виде процедуры с передачей в нее всех элементов рассматриваемого массива.

4) Даны два двумерных массива символьных (буквы русского алфавита) элементов. Размер исходных массивов не превосходит 10x10 элементов. Для каждого из массивов указать номера строк, содержащих элементы только строчных букв, если таких строк нет ни для какого массива, то вывести соответствующее сообщение. Проверку строки на наличие указанных элементов оформить в виде процедуры с передачей в нее всех элементов текущей строки.

5) Даны два двумерных массива вещественных элементов. Размер исходных массивов не превосходит 10x10 элементов. Для каждого из массивов указать количество столбцов, содержащих только не положительные элементы. Если таких столбцов нет ни для одного из массивов, то вывести соответствующее сообщение. Проверку столбца на наличие указанных элементов оформить в виде процедуры с передачей в нее всех элементов текущего столбца.

6) Даны пять одномерных массива вещественных элементов. Размер каждого массива не превосходит 100 элементов. Для каждого из массивов определить, составляют ли его элементы одного знака. Если да, то указать порядковый номер такого массива, в противном случае вывести отрицательный ответ. Проверку массива на выполнение условия оформить в виде процедуры с передачей в нее всех элементов рассматриваемого массива.

7) Даны два двумерных массива целочисленных элементов. Размер исходных массивов не превосходит 10x10 элементов. Для каждого из массивов указать количество строк, содержащих элементы, четность которых чередуется, а вторым в четных строках является нечетный элемент. Если таких строк нет ни для одного из массивов, то вывести соответствующее сообщение. Проверку строки на наличие указанных элементов оформить в виде процедуры с передачей в нее всех элементов текущего столбца.

8) Даны пять одномерных массива символьных (только латинские буквы) элементов. Размер каждого массива не превосходит 100 элементов. Для каждого из массивов определить, чередуются ли в нем буквы строчные и прописные. Если да, то указать порядковый номер такого массива, в противном случае вывести отрицательный ответ. Проверку массива на выполнение условия оформить в виде процедуры с передачей в нее всех элементов рассматриваемого массива.

9) Даны два двумерных массива целочисленных элементов. Размер исходных массивов не превосходит 10x10 элементов. Для каждого из массивов указать количество строк, для которых сумма элементов, стоящих на нечетных местах в строке, является положительным числом. Если таких строк нет ни для одного из массивов, то вывести соответствующее сообщение. Проверку строки на выполнение условия и расчет оформить в виде процедуры с передачей в нее всех элементов текущей строки.

10) Даны два двумерных массива вещественных элементов. Размер исходных массивов не превосходит 10x10 элементов. Для каждого из массивов указать номера столбцов, произведение отрицательных элементов которых является положительным числом. Если таких столбцов нет ни для одного из массивов, то вывести соответствующее сообщение. Проверку столбца на выполнение условия и расчет оформить в виде процедуры с передачей в нее всех элементов текущего столбца.

11) Даны пять одномерных массива символьных (только латинские буквы) элементов. Размер каждого массива не превосходит 100 элементов. Для каждого из массивов определить, расположены ли в нем строчные буквы в алфавитном порядке. Если да, то указать порядковый номер такого массива, в противном случае вывести отрицательный ответ. Проверку массива на

выполнение условия оформить в виде процедуры с передачей в нее всех элементов рассматриваемого массива.

12) Даны два двумерных массива целочисленных элементов. Размер исходных массивов не превосходит 10x10 элементов. Для каждого из массивов проверить выполнение условия: все четные строки массива таковы, что суммы их элементов образуют возрастающую последовательность. Вывести соответствующее сообщение. Вычисление суммы элементов массива и проверку последовательности чисел на выполнение условия оформить в виде процедуры с передачей в нее всех необходимых элементов.

13) Даны два двумерных массива вещественных элементов. Размер исходных массивов не превосходит 10x10 элементов. Преобразовать все нечетные строки каждого массива так, чтобы элементы составляли возрастающую по абсолютной величине последовательность. Вывести преобразованные массивы. Упорядочивание элементов оформить в виде процедуры с передачей в нее всех необходимых элементов.

14) Даны два двумерных массива целочисленных элементов. Размер исходных массивов не превосходит 10x10 элементов. Для каждого столбца массивов вычислить суммы и количества элементов, значения которых находятся в заданном диапазоне. Если чисел, удовлетворяющих этому условию нет, то вывести соответствующее сообщение. Вычисление для элементов столбца массива оформить в виде процедуры с передачей в нее всех необходимых элементов.

15) Даны пять одномерных массива символьных (только латинские буквы) элементов. Размер каждого массива не превосходит 100 элементов. Преобразовать все массивы так, чтобы все строчные буквы были расположены по алфавиту. При этом переставлять только строчные буквы, оставив прописные буквы на своих местах. Преобразование каждого массива оформить в виде процедуры с передачей в нее всех необходимых элементов. Если перестановка элементов не потребовалась, то есть исходные массивы удовлетворяют требуемому условию, то вывести соответствующее сообщение.

Задание:

1) Разработать модули программы, спроектированные во время практического занятия

2) Отладить программу с использованием тестов, составленных во время практического занятия

3) Составить отчет.

Задание № 6 «Отладка отдельных модулей программного проекта»

Цель работы. Изучить основные подходы к проектированию тестов.

Рассмотрим два основных подхода к проектированию тестов.

Первый подход ориентируется только на стратегию тестирования, называемую стратегией "черного ящика", тестированием с управлением по данным или тестированием с управлением по входу-выходу. При использовании этой стратегии программа рассматривается как черный ящик. Тестовые данные используются только в соответствии со спецификацией программы (т. е. без учета знаний о ее внутренней структуре). Недостижимый идеал сторонника первого подхода — проверить все возможные комбинации и значения на входе. Обычно их слишком много даже для простейших алгоритмов. Так, для программы расчета среднего арифметического четырех чисел надо готовить 107 тестовых данных.

При первом подходе обнаружение всех ошибок в программе является критерием исчерпывающего входного тестирования. Последнее может быть достигнуто, если в качестве тестовых наборов использовать все возможные наборы входных данных. Следовательно, приходим к выводу, что для исчерпывающего тестирования программы требуется бесконечное число тестов, а значит, построение исчерпывающего входного теста невозможно. Это подтверждается двумя аргументами: во-первых, нельзя создать тест, гарантирующий

отсутствие ошибок; во-вторых, разработка таких тестов противоречит экономическим требованиям. Поскольку исчерпывающее тестирование исключается, нашей целью должна стать максимизация результативности капиталовложений в тестирование (максимизация числа ошибок, обнаруживаемых одним тестом). Для этого необходимо рассматривать внутреннюю структуру программы и делать некоторые разумные, но, конечно, не обладающие полной гарантией достоверности предположения.

Второй подход использует стратегию "белого ящика", или стратегию тестирования, управляемую логикой программы, которая позволяет исследовать внутреннюю структуру программы. В этом случае тестировщик получает тестовые данные путем анализа только логики программы; стремится, чтобы каждая команда была выполнена хотя бы один раз. При достаточной квалификации добивается, чтобы каждая команда условного перехода выполнялась бы в каждом направлении хотя бы один раз. Цикл должен выполняться один раз, ни разу, максимальное число раз. Цель тестирования всех путей извне также недостижима. В программе из двух последовательных циклов внутри каждого из них включено ветвление на десять путей, имеется 1018 путей расчета. Причем выполнение всех путей расчета не гарантирует выполнения всех спецификаций.

Сравним способ построения тестов при данной стратегии с исчерпывающим входным тестированием стратегии "черного ящика". Неверно предположение, что достаточно построить такой набор тестов, в котором каждый оператор исполняется хотя бы один раз. Исчерпывающему входному тестированию может быть поставлено в соответствие исчерпывающее тестирование маршрутов. Подразумевается, что программа проверена полностью, если с помощью тестов удается осуществить выполнение этой программы по всем возможным маршрутам ее потока (графа) передач управления.

Последнее утверждение имеет два слабых пункта: во-первых, число не повторяющих друг друга маршрутов — астрономическое; во-вторых, даже если каждый маршрут может быть проверен, сама программа может содержать ошибки (например, некоторые маршруты пропущены). Свойство пути выполняться правильно для одних данных и неправильно для других — называемое чувствительностью к данным, наиболее часто проявляется за счет численных погрешностей и погрешностей усечения методов. Тестирование каждого из всех маршрутов одним тестом не гарантирует выявление чувствительности к данным.

В результате всех изложенных выше замечаний отметим, что ни исчерпывающее входное тестирование, ни исчерпывающее тестирование маршрутов не могут стать полезными стратегиями, потому что оба они нереализуемы. Поэтому реальным путем, который позволит создать хорошую, но, конечно, не абсолютную стратегию, является сочетание тестирования программы несколькими методами.

Рассмотрим пример тестирования оператора:

if A and B then… при использовании разных критериев полноты тестирования.

При критерии покрытия условий требовались бы два теста: A = true, B = false и A = false, B = true. Но в этом случае не выполняется then-предложение оператора if.

Существует еще один критерий, названный покрытием решений/условий. Он требует такого достаточного набора тестов, чтобы все возможные результаты каждого условия в решении выполнялись, по крайней мере, один раз; все результаты каждого решения выполнялись тоже один раз и каждой точке входа передавалось управление, по крайней мере, один раз.

Недостатком критерия покрытия решений/условий является невозможность его применения для выполнения всех результатов всех условий. Часто подобное выполнение имеет место вследствие того, что определенные условия скрыты другими условиями. Например, если условие AND есть ложь, то никакое из последующих условий в выражении не будет выполнено. Аналогично, если условие OR есть истина, то никакое из последующих условий не будет выполнено. Следовательно, критерии покрытия условий и покрытия решений/условий недостаточно чувствительны к ошибкам в логических выражениях.

Критерием, который решает эти и некоторые другие проблемы, является комбинаторное покрытие условий. Он требует создания такого числа тестов, чтобы все возможные комбинации результатов условия в каждом решении и все точки входа выполнялись, по крайней мере, один раз.

В случае циклов число тестов для удовлетворения критерию комбинаторного покрытия условий обычно больше, чем число путей.

Легко видеть, что набор тестов, удовлетворяющий критерию комбинаторного покрытия условий, удовлетворяет также и критериям покрытия решений, покрытия условий и покрытия решений/условий.

Таким образом, для программ, содержащих только одно условие на каждое решение, минимальным является критерий, набор тестов которого вызывает выполнение всех результатов каждого решения, по крайней мере, один раз; передает управление каждой точке входа (например, оператор CASE).

Для программ, содержащих решения, каждое из которых имеет более одного условия, минимальный критерий состоит из набора тестов, вызывающих все возможные комбинации результатов условий в каждом решении и передающих управление каждой точке входа программы, по крайней мере, один раз.

Деление алгоритма на типовые стандартные структуры позволяет минимизировать усилия программиста, затрачиваемые им на тестирование. Запрет на вложенные структуры как раз и объясняется излишними затратами на тестирование. Использование цепочки простых альтернатив с одним действием или структуры ВЫБОР вместо вложенных простых АЛЬТЕРНАТИВ значительно сокращает число тестов!

Задание:

1) Оформить внешнюю спецификацию.

2) Составить в виде блок-схемы алгоритм решения задачи.

3) Создать программу решения задачи на любом алгоритмическом языке программирования.

4) Составить набор тестов и провести тестирование созданной программы с помощью методов «белого ящика» (покрытия операторов, покрытия решений, покрытия условий, комбинаторного покрытия условий).

5) Оформить отчет.

Отчет должен включать:

1) Внешнюю спецификацию.

2) Алгоритм решения задачи.

3) Текст программы на языке программирования.

4) Набор тестов для отладки программы, соответствующий конкретным методам «белого ящика».

Задача. «Нахождение характерных точек функции». Составить алгоритм и написать программу последовательного вычисления значений заданной функции Y(X) до тех пор, пока не будет пройдена некоторая характерная точка графика функции. Значения аргумента X составляют возрастающую последовательность с шагом h. Начальное значение X0 и шаг изменения аргумента h задаются пользователем.

Варианты индивидуальных заданий:

1	Локальный минимум функции $x^2+0.5-\sin(3*x)$	9	Точка (точки), в которой функция $\left\|\frac{2x^3-3}{x^2+1}\right\|$ равна 10.
2	Точка (точки), для которой $\sqrt{x^2+5}-1=5$.	10	Локальный минимум функции $3(x+4)^2-\sin(x+15)$
3	Пересечение графиков функций $\sqrt{x^2+1}(3+x)$ и $\frac{5}{x^2-7}$.	11	Точка (точки), в которой функция $\frac{2x^2}{\sin(x-1)}-1$ равна -500.
4	Все нули функции $x^2-\sin(3x)$.	12	Локальный максимум функции $\frac{2x+15}{3+x^2}$
5	Локальный минимум функции $\frac{5}{3x-x^2-3}$	13	Пересечение графиков функций $(x^2+1)\sin(3+x)$ и $\frac{x+5}{x^2+1}$.
6	Точка (точки), в которой функция x^2-e^x равна -10.	14	Локальный минимум функции $x^2-3x+15$
7	Пересечение графиков функций $100\sin(1+x)$ и x^3+5.	15	Все нули функции $\sqrt{x^2+0.5}-\sin(3x)$.
8	Локальный минимум функции $\sqrt{e^x}-x$	16	Локальный максимум функции $200x-e^x$

Задание № 7 «Отладка проекта»

Цель работы. Получение практических навыков тестирования и отладки программы.

Тестирование – процесс выполнения программы на наборе тестов с целью выявления ошибок.

Локализацией называют процесс определения оператора программы, выполнение которого вызвало нарушение нормального вычислительного процесса. Для исправления ошибки необходимо определить ее причину, т.е. определить оператор или фрагмент, содержащие ошибку. Причины ошибок могут быть как очевидны, так и очень глубоко скрыты. В целом сложность отладки обусловлена следующими причинами:

– требует от программиста глубоких знаний специфики управления используемыми техническими средствами, операционной системы, среды и языка программирования, реализуемых процессов, природы и специфики различных ошибок, методик отладки и соответствующих программных средств;

– психологически дискомфортна, так как необходимо искать собственные ошибки и, как правило, в условиях ограниченного времени;

– возможно взаимовлияние ошибок в разных частях программы, например, за счет затирания области памяти одного модуля другим из-за ошибок адресации;

– отсутствуют четко сформулированные методики отладки.

Отладка программы в любом случае предполагает обдумывание и логическое осмысление всей имеющейся информации об ошибке. Большинство ошибок можно обнаружить по косвенным признакам посредством тщательного анализа текстов программ и результатов тестирования без получения дополнительной информации. При этом используют различные методы:

– ручного тестирования;

– индукции;
– дедукции;
– обратного прослеживания.

Метод ручного тестирования

Это - самый простой и естественный способ данной группы. При обнаружении ошибки необходимо выполнить тестируемую программу вручную, используя тестовый набор, при работе с которыми была обнаружена ошибка. Метод очень эффективен, но не применим для больших программ, программ со сложными вычислениями и в тех случаях, когда ошибка связана с неверным представлением программиста о выполнении некоторых операций. Данный метод часто используют как составную часть других методов отладки.

Метод индукции

Метод основан на тщательном анализе симптомов ошибки, которые могут проявляться как неверные результаты вычислений или как сообщение об ошибке. Если компьютер просто "зависает", то фрагмент проявления ошибки вычисляют, исходя из последних полученных результатов и действий пользователя. Полученную таким образом информацию организуют и тщательно изучают, просматривая соответствующий фрагмент программы. В результате этих действий выдвигают гипотезы об ошибках, каждую из которых проверяют. Если гипотеза верна, то детализируют информацию об ошибке, иначе - выдвигают другую гипотезу. Последовательность выполнения отладки методом индукции показана на рисунке в виде схемы алгоритма.

Самый ответственный этап - выявление симптомов ошибки. Организуя данные об ошибке, целесообразно записать все, что известно о её проявлениях, причем фиксируют, как ситуации, в которых фрагмент с ошибкой выполняется нормально, так и ситуации, в которых ошибка проявляется. Если в результате изучения данных никаких гипотез не появляется, то необходима дополнительная информация об ошибке. Дополнительную информацию можно получить, например, в результате выполнения схожих тестов. В процессе доказательства пытаются выяснить, все ли проявления ошибки объясняет данная гипотеза, если не все, то либо гипотеза не верна, либо ошибок несколько.

Метод дедукции

По методу дедукции вначале формируют множество причин, которые могли бы вызвать данное проявление ошибки. Затем анализируя причины, исключают те, которые противоречат имеющимся данным. Если все причины исключены, то следует выполнить дополнительное тестирование исследуемого фрагмента. В противном случае наиболее вероятную гипотезу пытаются доказать. Если гипотеза объясняет полученные признаки ошибки, то ошибка найдена, иначе - проверяют следующую причину.

Метод обратного прослеживания

Для небольших программ эффективно применение метода обратного прослеживания. Начинают с точки вывода неправильного результата. Для этой точки строится гипотеза о значениях основных переменных, которые могли бы привести к получению имеющегося результата. Далее, исходя из этой гипотезы, делают предложения о значениях переменных в предыдущей точке. Процесс продолжают, пока не обнаружат причину ошибки.

Задание:
1) Составить в виде блок-схемы алгоритм решения задачи.

2) Создать программу решения задачи на любом алгоритмическом языке программирования.

3) Отладить программу.

4) Составить отчет.

Отчет должен включать:

1) Алгоритм решения задачи.

2) Текст программы на языке программирования.

3) Набор тестов для отладки программы.

Задача: составить список учебной группы, включающей 25 человек. Для каждого учащегося указать дату рождения, год поступления, курс, группу, оценки каждого года обучения.

Назначение задачи: получить значение определённого критерия и упорядочить список студентов по нему.

Достигаемая цель: упорядочить список студентов по среднему баллу и получить его.

Задание № 8 «Инспекция кода модулей проекта»

Цель работы получить практические навыки разработки модулей программной системы и интеграции этих модулей.

Задание:

1) Оформить внешнюю спецификацию.

2) Составить в виде блок-схемы алгоритм решения задачи.

3) Спроектировать и разработать модули программы для решения задачи на любом алгоритмическом языке программирования.

4) Выполнить отладку и тестирование модулей программы.

5) Выполнить инкрементную интеграцию модулей с использованием одного из подходов.

6) Выполнить системное тестирование программы.

7) Оформить отчет.

Отчет должен включать:

1) Внешнюю спецификацию.

2) Алгоритм решения задачи.

3) Текст программы на языке программирования.

4) Набор тестов для отладки модулей программы.

5) Описание процесса интеграции модулей.

Задача. Задан двумерный массив размерности n x m. Отсортировать элементы строк массива по возрастанию значений, а затем отсортировать строки массива по возрастанию среднего арифметического элементов строк.

Реализовать сортировку разными способами и сравнить эффективность этих способов для разных исходных данных.

Задание № 9 «Тестирование интерфейса пользователя средствами инструментальной среды разработки»

Цель работы. Получение практических навыков автоматической генерации тестов на основе формального описания.

Задание:
1) Сформировать диаграмму вариантов использования для задачи № 1.
2) Сгенерировать набор тестов.
3) Составить отчет.

Отчет должен включать:
1. Диаграмму вариантов использования. Файл с тестовым набором.

Задание № 10 «Разработка тестовых модулей проекта для тестирования отдельных модулей»

Цель работы. Получение практических навыков использования средств автоматизации тестирования.

Для того чтобы продолжать тестирование, когда один тест не прошёл, в генератор тестов встроена возможность выбора – генерировать один большой тест или набор атомарных тестов. Атомарный тест – тот, который не требует приведения системы в состояние, отличное от начального состояния.

В связи с наличием ограничения инструмента прогона тестов на тестовую длину, в тесты после каждой законченной инструкции вставляется строка разреза. Во время прогона по этим строкам осуществляется разрез теста в случае, если его длина превышает допустимое ограничение. После прохождения части теста до строки разреза продолжается выполнение теста с первой инструкции, следующей за строкой разреза. Нарезку и сам прогон тестов осуществляет прогонщик тестов.

В качестве прогонщика тестов мы используем Rational Robot, который выполняет сгенерированные наборы инструкций. В случае удачного выполнения всех инструкций выносится вердикт – тест прошёл. В противном случае, если на каком-то этапе выполнения теста, поведение системы не соответствует требованиям, Robot прекращает его выполнение, вынося соответствующий вердикт – тест не прошёл.

Задание:
1) Выполнить тестовый набор
2) Проанализировать отчёт о прохождении тестов.
3) Составить отчет

Отчет должен включать:
1) Отчёт о прохождении тестов.
2) Анализ отчёта о прохождении тестов.

Задание № 11 «Выполнение функционального тестирования»

Цель работы. Получить практические навыки разработки тестов на основе внешней спецификации программы.

Программа в случае тестирования с управлением по данным рассматривается как "черный ящик", и целью тестирования является выяснение обстоятельств, в которых поведение программы не соответствует спецификации. Различают следующие методы формирования тестовых наборов:

- эквивалентное разбиение;
- анализ граничных значений;
- анализ причинно-следственных связей;
- предположение об ошибке.

Эквивалентное разбиение

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. Для составления классов эквивалентности нужно перебрать ограничения, установленные для каждого входного значения в техническом задании или при уточнении спецификации. Каждое ограничение разбивают на две или более групп.

Граничные значения

Граничные значения - это значения на границах классов эквивалентности входных значений или около них.

Анализ причинно-следственных связей

Метод анализа причинно-следственных связей позволяет системно выбирать тесты, используя алгебру логики. Причиной называют отдельное входное условие или класс эквивалентности. Следствием - выходное условие или преобразование системы. Идея заключается в отнесении всех следствий к причинам, то есть в уточнении причинно-следственных связей.

Предположение об ошибке

Метод основан на интуиции программиста с большим опытом работы. Составляется список, в котором перечисляются возможные ошибки или ситуации, в которых они могут появиться, а затем на основе списка составляются тесты.

Задание:
1) На основе внешней спецификации задачи составить набор тестов на основе подхода «чёрного ящика».
2) Провести тестирование программы.
3) Оформить отчет.

Отчет должен включать:
1) Внешнюю спецификацию.
2) Алгоритм решения задачи.
3) Текст программы на языке программирования.
4) Набор тестов на основе подхода «чёрного ящика» для отладки программы.

Задание № 12 «Тестирование интеграции»

Цель работы. Получить практические навыки отладки программ с помощью отладчика среды программирования.

Теоретические основы.

Отладка — это процесс определения и устранения причин ошибок. Этим она отличается от тестирования, направленного на обнаружение ошибок. В некоторых проектах отладка занимает до 50% общего времени разработки. Многие программисты считают отладку самым трудным аспектом программирования.

Для сокращения времени отладки необходимо пользоваться научным подходом. Классический научный подход включает следующие этапы:

1) Сбор данных при помощи повторяющихся экспериментов.

2) Формулирование гипотезы, объясняющей релевантные данные.

3) Разработка эксперимента, призванного подтвердить или опровергнуть гипотезу.

4) Подтверждение или опровержение гипотезы.

5) Повторение процесса в случае надобности.

Эффективный метод поиска дефектов при отладке с использованием научного подхода может быть описан следующими шагами:

1) Стабилизация ошибки.

2) Определение источника ошибки.

a) Сбор данных, приводящих к дефекту.

b) Анализ собранных данных и формулирование гипотезы, объясняющей дефект.

c) Определение способа подтверждения или опровержения гипотезы, основанного или на тестировании программы, или на изучении кода.

d) Подтверждение или опровержение гипотезы при помощи процедуры, определенной в п. 2(c).

3) Исправление дефекта.

4) Тестирование исправления.

5) Поиск похожих ошибок.

Способ подтверждения или опровержения гипотезы может быть одним из следующего списка:

1) сокращение подозрительной области кода;

2) проверка классов и методов, в которых дефекты обнаруживались ранее;

3) проверка кода, который изменялся недавно.

Отладка — это тот этап разработки программы, от которого зависит возможность се выпуска. Конечно, лучше всего вообще избегать ошибок. Однако потратить время на улучшение навыков отладки все же стоит, потому что эффективность отладки, выполняемой лучшими и худшими программистами, различается минимум в 10 раз.

Систематичный подход к поиску и исправлению ошибок — непременное условие успешности отладки. Организуйте отладку так, чтобы каждый тест приближал вас к цели. Используйте Научный Метод Отладки.

Прежде чем приступать к исправлению программы, поймите суть проблемы. Случайные предположения о причинах ошибок и случайные исправления только ухудшат программу.

Установите в настройках компилятора самый строгий уровень диагностики и устраняйте причины всех ошибок и предупреждений.

Инструменты отладки значительно облегчают разработку ПО. Найдите их и используйте. Большинство современных сред программирования (Delphi, C++ Builder, Visual Studio и т.д.) включают средства отладки, которые обеспечивают максимально эффективную отладку. Они позволяют:

−выполнять программу по шагам, причем как с заходом в подпрограммы, так и выполняя их целиком;

−предусматривать точки останова;

−выполнять программу до оператора, указанного курсором;

−отображать содержимое любых переменных при пошаговом выполнении;

−отслеживать поток сообщений и т.п.

Задание:

1) Составить в виде блок-схемы алгоритм решения задачи.

2) Создать программу решения задачи на любом алгоритмическом языке программирования.

3) Отладить программу с использованием инструментальных средств.

4) Составить отчет.

Отчет должен включать:

1) Алгоритм решения задачи.

2) Текст программы на языке программирования.

3) Набор тестов для отладки программы.

Задача: Имеется матрица размера N*M. Определить в какой строке количество положительных элементов наибольшее.

Задание № 13 «Документирование результатов тестирования»

Цель работы. Получение практических навыков оформления протоколов тестирования и отладки программы.

Тестирование – процесс выполнения программы на наборе тестов с целью выявления ошибок.

Обеспечить повторяемость процесса тестирования недостаточно – вы должны оценивать и проект, чтобы можно было точно сказать, улучшается он в результате изменений или ухудшается. Вот некоторые категории данных, которые можно собирать с целью оценки проекта:

• административное описание дефекта (дата обнаружения, сотрудник, сообщивший о дефекте, номер сборки программы, дата исправления);

• полное описание проблемы;

• действия, предпринятые для воспроизведения проблемы;

• предложенные способы решения проблемы;

• родственные дефекты;

• тяжесть проблемы (например, критическая проблема, «неприятная» или косметическая);

• источник дефекта: выработка требований, проектирование, кодирование или тестирование;

• вид дефекта кодирования: ошибка занижения или завышения на 1, ошибка присваивания, недопустимый индекс массива, неправильный вызов метода и т. д.;

• классы и методы, измененные при исправлении дефекта;

• число строк, затронутых дефектом;

• время, ушедшее на нахождение дефекта;

• время, ушедшее на исправление дефекта.

Собирая эти данные, вы сможете подсчитывать некоторые показатели, позволяющие сделать вывод об изменении качества проекта:

• число дефектов в каждом классе; все числа целесообразно отсортировать в порядке от худшего класса к лучшему и, возможно, нормализовать по размеру класса;

• число дефектов в каждом методе, все числа целесообразно отсортировать в порядке от худшего метода к лучшему и, возможно, нормализовать по размеру метода;

• среднее время тестирования в расчете на один обнаруженный дефект;

• среднее число обнаруженных дефектов в расчете на один тест;

• среднее время программирования в расчете на один исправленный дефект;

• процент кода, покрытого тестами;

• число дефектов, относящихся к каждой категории тяжести.

Кроме протоколов тестирования уровня проекта, вы можете хранить и личные протоколы тестирования. Можете включать в них контрольные списки ошибок, которые вы допускаете чаще всего, и указывать время, затрачиваемое вами на написание кода, его тестирование и исправление ошибок.

Задание:
1) Выполнить тестирование программы.
2) Оформить протоколы тестирования.
3) Оформить отчет.
Отчет должен включать:
1) Внешнюю спецификацию.
2) Набор тестов.
3) Текст программы на языке программирования.
4) Протоколы тестирования программы.

2.1.2 Тесты

Тестовое задание 1.

1. Сколько систем контроля версий (VCS) существует:
а**) много;**
б) только GIT;
в) GIT и SVN;
г) GIT и CVS;
2. Что такое GitHub:
а**) Веб-сервис для хостинга IT-проектов и их совместной разработки, основанный на Git;**
б) программа для работы с Git;
в) драйвер для Git;
г) UI для работы с локальной версией Git;
3. Что такое репозиторий Git:
а**) репозиторий Git представляет собой каталог файловой системы, в котором находятся файлы конфигурации репозитория, файлы журналов, хранящие операции, выполняемые над репозиторием, индекс, описывающий расположение файлов, и хранилище, содержащее собственно файлы;**
б) любая директория/папка в моей ОС;
в) любая папка, находящаяся внутри Git;
г) папка .git/ и все входящие в нее;
4. Что делает команда git status:
а**) показывает состояние проекта: кол-во untracked, deleted, new и прочих файлов, количество коммитов, на которое отличается локальная версия репозитория от удаленного и так далее;**
б) показывает имя и email нашего пользователя, а также является ли он авторизованным в системе GitHub или нет;
в) показывает место, занимаемое репозиторием на жестком диске и кол-во выделенного под репозиторий месте;
г) такой команды нет, есть только команда git show
5. Что делает команда git add:
а**) создает файл с указанным именем и сразу добавляет его в Git;**
б) добавляет локальный файл в удаленный репозиторий так, чтобы другие участники проекта могли его видеть;
в) это алиас/синоним команды git commit;
г) начинает отслеживать указанный файл или файлы;
6. Что означает статус файла untracked в выводе команды git status?:
а**) что система Git не отслеживает этот файл;**
б) что файл был удален из Git;

в) что файл находится вне репозитория Git;

г) что файл добавлен в .gitignore;

7. Что означает статус файла new в выводе команды git status?:

а) что файл только что был создан и еще не отслеживается системой Git;

б) **что файл только начал отслеживаться Git и пока не имеет истории;**

в) что файл удаляли из Git и потом восстановили командой git return;

г) такого статуса нет, есть только статус deleted file;

8. Что означает статус файла modified в выводе команды git status?:

а**) что файл имеет историю в системе Git и был изменен относительно его последнего состояния;**

б) такого статуса нет, есть только статусы new и deleted;

в) этот статус виден только командой git ignore и означает, что файл перестал отслеживаться системой Git;

г) статус означает, что файл добавлен в коммит;

9. Что такое коммит (commit)?:

а**) единица состояния проекта Git;**

б) это результат вывода команды git diff;

в) это обобщающее название одного из статусов файла в выводе git status: untracked, new, deleted или modified;

г) это слово ничего не означает, его ввели только для того, чтобы путать новичков;

10. Как сделать коммит (commit) в Git:

а**) сделать изменения, собрать эти изменения командой "git add" или "git commit -a" и указать коммит-сообщение после ключа "-m";**

б) всего лишь набрать команду git commit в любой момент времени;

в) сделать изменения в файлах и перечислить их после git commit. Например так: git commit a.file, b.file;

г) Нельзя сделать коммит, ведь такого понятия в Git не существует;

11. В какой ситуации необходимо делать Git status:

а**) в случае необходимости узнать, в каком статусе находится репозиторий, а так эта команда не является обязательной для любой манипуляции**

б) чем чаще, тем лучше;

в) всегда при создании коммита;

г) всегда после команды git pull;

12. Что такое ветка в репозитории Git:

а**) это разные пути развития проекта, по сути разные последовательности коммитов**

б) то же самое, что коммит (commit);

в) это минимум два коммита с одинаковым коммит-сообщением;

г) это механизм изменения конкретного файла;

13. Чем отличается master и origin master:

а) ветки origin master в Git не существует;

б) это просто два разных названия одной ветки;

в) master принадлежит локальному репозиторию, а origin master - удаленному;

г) Это две разные ветки локального репозитория;

14. Чем отличаются команды "git push" и "git pull"?:

а**) команда "git pull" нужна, чтобы стянуть изменения из удаленного репозитория, а команда "git push" нужна, чтобы выложить изменения в удаленный репозиторий";**

б) команды "git pull" не существует, а команда "git push" нужна, чтобы выложить изменения в удаленный репозиторий;

в) это алиасы;

г) команды "git push" не существует, а команда "git pull" нужна, чтобы стянуть изменения из удаленного репозитория;
15. Что означает команда Git log:
а**) показывает историю коммитов;**
б) пишет указанный после файл в лог;
в) такой команды нет, есть только команда git look;
г) удаляет файл из репозитория;

2. Этап, занимающий наибольшее время в жизненном цикле программы:

2.1.3 Контрольный опрос

Контрольный опрос 1:
1) Перечислите этапы разработки программных продуктов.
2) Для чего необходимо техническое задание?
3) Кто занимается разработкой технического задания?
4) Какие пункты включает техническое задание?

Контрольный опрос №2:
1) Для чего разрабатываются спецификации на программный продукт?
2) Что должны включать спецификации на программный продукт?
3) Что должна содержать спецификация процессов
4) Что такое словарь терминов и для чего он используется?
5) Что такое диаграмма переходов состояний и для чего ее используют?
6) Что такое диаграмма потоков и для чего ее используют?

Контрольный опрос №3:
1) Значение фазы интеграции программных модулей.
2.) Подходы к интегрированию программных модулей. Эффективность и оптимизация программ.

Контрольный опрос №4:
1) Что такое тестирование программы?
2) Что такое отладка программы?
3) Какие стадии тестирования выделяют при разработке программного обеспечения?
4) Какие различают подходы в формировании тестовых наборов?
5) В чем суть тестирования методом “покрытия операторов”?
6) В чем суть тестирования методом “покрытия решений”?
7) В чем суть тестирования методом “покрытия условий”?
8) В чем суть тестирования методом “комбинаторного покрытия условий”?
9) В чём суть метода эквивалентных разбиений?
10) В чём суть метода анализа граничных значений?
11) В чём суть метода анализа причинно-следственных связей?

2.2 Оценочные средства по дисциплине для промежуточной аттестации

2.2.1 Перечень вопросов для подготовки к дифференцированному зачёту:

1) Понятие репозитория проекта, структура проекта.
2) Виды, цели и уровни интеграции программных модулей.
3) Автоматизация бизнес-процессов.
4) Выбор источников и приемников данных, сопоставление объектов данных.
5) Транспортные протоколы.

6) Стандарты форматирования сообщений.
7) Организация работы команды в системе контроля версий.
8) Отладка программных продуктов.
9) Инструменты отладки.
10) Отладочные классы.
11) Ручное и автоматизированное тестирование.
12) Методы и средства организации тестирования.
13) Инструментарии анализа качества программных продуктов в среде разработке.
14) Обработка исключительных ситуаций.
15) Методы и способы идентификации.
16) Выявление ошибок системных компонентов.

2.2.2 Практические задания на дифференцированный зачёт:

1) Выполнить сценарий события, когда текст заголовка становится красным, когда пользователь наводит на него курсор и цвет возвращается, когда курсор отводится. Для этого нужно воспользоваться CSS и JavaScript. Преобразуйте страницу с заголовком «Добро пожаловать на нашу страницу!» - и текстом «Здесь много интересной информации. Здесь много интересной информации. Здесь много интересной информации. Здесь много интересной информации. Здесь много интересной информации.». Где такой сценарий можно еще использовать на практике, при каких ситуациях?

2) На Web-странице применить анимацию. Выполнить сценарий ситуации, когда текст «Текст, шагом марш!» должен перемещаться слева напра1.во. Для этого нужно воспользоваться тэгом, ограничивающем текст, идентификатором id, CSS, функцией moveTxtO,оператором if., атрибутом CSS pixelLeft., атрибутомpixelLeft, метода setTimeout, событием onLoad. Условие: чтобы запустить сценарий на выполнение, если текст находится менее чем в 500 пикселах от левой границы экрана. Каждый раз текст будет перемещаться вправо на два пиксела. Установить интервал до повторного запуска функции moveTxt(), равным 50 мс.

3) Выполнить сценарий ситуации проверки, содержится ли на странице элемент h1 - Первый заголовок? Можно воспользоваться страницей с одним элементом h1. Где такой сценарий можно еще использовать на практике, при каких ситуациях?

4) Способность отыскивать новые тэги позволяет сделать активным любой, даже самый незначительный элемент Web-страницы. Выполнить сценарий ситуации, когда имя тэга выясняется с помощью window.event.srcElement.tagName и указывается в строке состояния. Объект SrcElement обращается к исходному элементу, то есть к элементу, генерируемому событием. Подобный элемент можно легко обнаружить. Где такой сценарий можно еще использовать на практике, при каких ситуациях?

5) Выполнить сценарий ситуации, используя событие onContextmenu, когда пользователь щелкает по полю документа правой кнопкой мыши, чтобы открыть контекстное меню. Это событие должно позволить запустить сценарий до того, как меню возникнет на экране, или вовсе предотвратить появление контекстного меню. Последнее можно отменить, воспользовавшись свойством event.returnValue и указав значение false. Тем самым отменяется событие, которое должно произойти по умолчанию.

6) Создать в Visual Studio DLL (динамическую библиотеку) логина и пароля для идентификации пользователя при загрузке приложения. Где такое задание может быть использовано на практике, при каких ситуациях?

7) Составить тесты для программного продукта методом «черного ящика». Результаты оформить в таблице. Проанализировать полученный результат

8) Составить тесты для программного продукта методом «белого ящика». Результаты оформить в таблице. Проанализировать полученный результат

9) Выберете нужный вид тестирования программного продукта. Проанализировать свой выбор и доказать его приоритетность перед другими. Результаты оформить в таблице как в

образце.

3. Описание системы оценивания, шкала оценивания

3.1 Показатели и критерии оценивания для текущего контроля.

Перечень оценочных средств для текущего контроля	**Показатели и критерии оценивания (в баллах для бакалавриата и специалитета, в оценках для магистратуры и СПО)**
Задание 1 Задание 2 Задание 3 Задание 4 Задание 5 Задание 6 Задание 7 Задание 8 Задание 9 Задание 10 Задание 11 Задание 12 Задание 13	«отлично» - задание выполнено в полном объёме, с предоставлением отчёта. При выполнении задания соблюдена последовательность выполнения пунктов, в отчёте правильно и аккуратно выполнены все рисунки, таблицы, графики, блок-схемы. Студент владеет теоретическим материалом, при ответе на вопросы формулирует собственные самостоятельные, обоснованные и аргументированные решения, предоставляет полные и развёрнутые ответы на дополнительные вопросы. «хорошо» - выполнены требования к оценке «отлично», но было допущено два-три недочёта, не более одной негрубой ошибки и одного недочёта. Студент владеет теоретическим материалом, формулирует собственные, самостоятельные, обоснованные, аргументированные суждения, допуская незначительные ошибки при ответах на дополнительные вопросы. «удовлетворительно» - задание выполнено не полностью, но объём выполненной части таков, что включает в себя действия по теме задания. Задание выполнено полностью, студент владеет материалом на минимально допустимом уровне, испытывает затруднения в формулировке собственных обоснованных и аргументированных суждений, допуская ошибки в ответах на дополнительные вопросы «неудовлетворительно» - задание не выполнено, или объём выполненного задания не включает в себя действия по теме задания.
Тестовое задание 1	Шкала оценивания в зависимости от количества правильных ответов: «неудовлетворительно» - от 0% до 40% «удовлетворительно» - от 41% до 60% «хорошо»- от 61% до 80% «отлично» - от 81% до 100%
Контрольный опрос 1 Контрольный опрос 2 Контрольный опрос 3 Контрольный опрос 4	«отлично» оценивается ответ, который показывает прочные знания основных процессов изучаемой предметной области, отличается глубиной и полнотой раскрытия темы; владение терминологическим аппаратом; умение объяснять сущность, явлений, процессов, событий, делать выводы и обобщения, давать аргументированные ответы, приводить примеры; свободное владение монологической речью, логичность и последовательность ответа. «хорошо» оценивается ответ, обнаруживающий прочные знания основных процессов изучаемой предметной области, отличается глубиной и полнотой раскрытия темы; владение терминологическим аппаратом; умение объяснять сущность, явлений, процессов, событий, делать выводы и обобщения, давать аргументированные ответы, приводить примеры; свободное владение монологической речью, логичность и последовательность ответа. Однако допускается одна-две неточности в ответе.

Перечень оценочных средств для текущего контроля	Показатели и критерии оценивания (в баллах для бакалавриата и специалитета, в оценках для магистратуры и СПО)
	«удовлетворительно» оценивается ответ, свидетельствующий в основном о знании процессов изучаемой предметной области, отличающийся недостаточной глубиной и полнотой раскрытия темы; знанием основных вопросов теории; слабо сформированными навыками анализа явлений, процессов, недостаточным умением давать аргументированные ответы и приводить примеры; недостаточно свободным владением монологической речью, логичностью и последовательностью ответа. Допускается несколько ошибок в содержании ответа. «неудовлетворительно» оценивается ответ, обнаруживающий незнание процессов изучаемой предметной области, отличающийся неглубоким раскрытием темы; незнанием основных вопросов теории, несформированными навыками анализа явлений, процессов; неумением давать аргументированные ответы, слабым владением монологической речью, отсутствием логичности и последовательности. Допускаются серьезные ошибки в содержании ответа.

3.2 Показатели и критерии оценивания для промежуточного контроля

Компонент компетенции (с указанием кода)	Индикаторы достижения компетенций	Критерии оценивания (в баллах для бакалавриата и специалитета, в оценках для магистратуры и СПО)
ПК-2.2.1 ПК-2.2.2	практическое задание по обеспечению интеграции заданного модуля в предложенный программный проект	**Оценка «отлично»** - в системе контроля версий выбрана верная версия проекта, проанализирована его архитектура, архитектура доработана для интеграции нового модуля; выбраны способы форматирования данных и организована их постобработка, транспортные протоколы и форматы сообщений обновлены (при необходимости); протестирована интеграция модулей проекта и выполнена отладка проекта с применением инструментальных средств среды; выполнена доработка модуля и дополнительная обработка исключительных ситуаций в том числе с созданием классов-исключений (при необходимости); определены качественные показатели полученного проекта; результат интеграции сохранен в системе контроля версий. Оценка «**хорошо**» - в системе контроля версий выбрана верная версия проекта, его архитектура доработана для интеграции

Компонент компетенции (с указанием кода)	Индикаторы достижения компетенций	Критерии оценивания (в баллах для бакалавриата и специалитета, в оценках для магистратуры и СПО)
		нового модуля; выбраны способы форматирования данных и организована их постобработка, транспортные протоколы и форматы сообщений обновлены (при необходимости); выполнена отладка проекта с применением инструментальных средств среды; выполнена доработка модуля и дополнительная обработка исключительных ситуаций (при необходимости); определены качественные показатели полученного проекта; результат интеграции сохранен в системе контроля версий. Оценка «**удовлетворительно**» - в системе контроля версий выбрана верная версия проекта, его архитектура доработана для интеграции нового модуля; выбраны способы форматирования данных и организована их постобработка, форматы сообщений обновлены (при необходимости); выполнена отладка проекта с применением инструментальных средств среды; выполнена доработка модуля (при необходимости); результат интеграции сохранен в системе контроля версий.
ПК-2.3.1 ПК-2.3.2	практическое задание по выполнению отладки программного модуля	Оценка «**отлично**» - в системе контроля версий выбрана верная версия проекта; протестирована интеграция модулей проекта и выполнена отладка проекта с применением инструментальных средств среды; проанализирована и сохранена отладочная информация; выполнена условная компиляция проекта в среде разработки; определены качественные показатели полученного проекта в полном объеме; результаты отладки сохранены в системе контроля версий. Оценка «**хорошо**»- в системе контроля версий выбрана верная версия проекта; протестирована интеграция модулей проекта и выполнена отладка проекта с применением инструментальных средств среды; выполнена условная компиляция проекта в среде разработки; определены качественные показатели полученного проекта в достаточном объеме; результаты

Компонент компетенции (с указанием кода)	**Индикаторы достижения компетенций**	**Критерии оценивания (в баллах для бакалавриата и специалитета, в оценках для магистратуры и СПО)**
		отладки сохранены в системе контроля версий. Оценка «**удовлетворительно**» - в системе контроля версий выбрана верная версия проекта; выполнена отладка проекта с применением инструментальных средств среды; выполнена условная компиляция проекта в среде разработки; определены качественные показатели полученного проекта в достаточном объеме; результаты отладки сохранены в системе контроля версий.
ПК-2.5.1 ПК-2.5.2	практическое задание по инспектированию программного кода	Оценка «**отлично**» - продемонстрировано знание стандартов кодирования более чем одного языка программирования, выявлены все имеющиеся несоответствия стандартам в предложенном коде. Оценка «**хорошо**» - продемонстрировано знание стандартов кодирования более чем одного языка программирования, выявлены существенные имеющиеся несоответствия стандартам в предложенном коде. Оценка «**удовлетворительно**» - продемонстрировано знание стандартов кодирования языка программирования, выявлены некоторые несоответствия стандартам в предложенном коде.