Учреждение профессионального образования
«Колледж Казанского инновационного университета»
Альметьевский филиал

.
УТВЕРЖДЕН
в составе Основной
образовательной программы –
программы подготовки специалистов среднего звена
протокол № 6 от «28» августа 2024 г.

Фонд оценочных средств
по профессиональному модулю

ПМ.02 Осуществление интеграции программных модулей

программы подготовки специалистов среднего звена

по специальности

09.02.07 Информационные системы и программирование
(на базе основного общего образования)

Срок получения СПО по ППССЗ – 3 г.10 мес.

Форма обучения - очная

Присваиваемая квалификация
Программист

Альметьевск 2024

Фонд оценочных средств по профессиональному модулю «Осуществление интеграции программных модулей» программы подготовки специалистов среднего звена по специальности 09.02.07 Информационные системы и программирование разработан на основе программы профессионального модуля и требований работодателя.

1.Общие положения

Фонд оценочных средства (ФОС) предназначены для контроля и оценки образовательных достижений обучающихся, освоивших программу дисциплины «Осуществление интеграции программных модулей».

ФОС включают контрольные материалы для проведения текущего и итогового контроля.

ФОС разработаны на основании положений:

- программы подготовки специалистов среднего звена по специальности 09.02.07 Информационные системы и программирование
- программы учебной дисциплины «Осуществление интеграции программных модулей»

2. Показатели оценки результатов освоения дисциплин, формы и методы контроля и оценки

В результате контроля и оценки по дисциплине осуществляется комплексная проверка следующих умений (У) и знаний (З):

Результаты обучения (освоенные умения, усвоенные знания, практический опыт)	Основные показатели оценки результатов
В результате освоения дисциплины обучающийся должен знать: - актуальный профессиональный и социальный контекст, в котором приходится работать и жить; основные источники информации и ресурсы для решения задач и проблем в профессиональном и/или социальном контексте; алгоритмы выполнения работ в профессиональной и смежных областях; методы работы в профессиональной и смежных сферах; структуру плана для решения задач; порядок оценки результатов решения задач профессиональной деятельности; - современные средства и устройства информатизации; порядок их применения и программное обеспечение в профессиональной деятельности; номенклатура информационных источников, применяемых в профессиональной деятельности; приемы структурирования информации; формат оформления результатов поиска информации; - основы предпринимательской деятельности; основы финансовой грамотности; правила разработки бизнес-планов; порядок выстраивания презентации; кредитные банковские продукты; - психологические основы деятельности коллектива, психологические особенности личности; основы проектной деятельности; - особенности социального и культурного контекста; правила оформления документов и построения устных сообщений; - сущность гражданско-патриотической позиции, духовно-нравственных ценностей; значимость профессиональной деятельности по	- знает модели процесса разработки программного обеспечения; - знает основные принципы процесса разработки программного обеспечения виды и варианты интеграционных решений; - знает основные протоколы доступа к данным; - методы отладочных классов; - знает стандарты качества программной документации; - знает графические средства проектирования архитектуры программных продуктов; - знает основные подходы к интегрированию программных модулей; - знает основы верификации программного обеспечения; - знает современные технологии и инструменты интеграции; - знает методы и способы идентификации сбоев и ошибок при интеграции приложений; основы верификации и аттестации программного обеспечения; - знает основные методы отладки; - знает методы и схемы обработки исключительных ситуаций; - знает приемы работы с инструментальными средствами тестирования и отладки; - знает основы организации инспектирования и верификации; - знает встроенные и основные специализированные инструменты анализа качества программных продуктов;

специальности; - правила экологической безопасности при ведении профессиональной деятельности; основные ресурсы, задействованные в профессиональной деятельности; пути обеспечения ресурсосбережения; - роль физической культуры в общекультурном, профессиональном и социальном развитии человека; основы здорового образа жизни; условия профессиональной деятельности и зоны риска физического здоровья для специальности; средства профилактики перенапряжения; - правила построения простых и сложных предложений на профессиональные темы; основные общеупотребительные глаголы (бытовая и профессиональная лексика); лексический минимум, относящийся к описанию предметов, средств и процессов профессиональной деятельности; особенности произношения; правила чтения текстов профессиональной направленности; - модели процесса разработки программного обеспечения; основные принципы процесса разработки программного обеспечения; основные подходы к интегрированию программных модулей; виды и варианты интеграционных решений; современные технологии и инструменты интеграции; основные протоколы доступа к данным; методы и способы идентификации сбоев и ошибок при интеграции приложений; методы отладочных классов; стандарты качества программной документации; основы организации инспектирования и верификации; встроенные и основные специализированные инструменты анализа качества программных продуктов; графические средства проектирования архитектуры программных продуктов; методы организации работы в команде разработчиков; - модели процесса разработки программного обеспечения; основные принципы процесса разработки программного обеспечения; основные подходы к интегрированию программных модулей; основы верификации программного обеспечения; современные технологии и инструменты интеграции; основные протоколы доступа к данным; методы и способы идентификации сбоев и ошибок при интеграции приложений; основные методы отладки; методы и схемы обработки исключительных ситуаций; основные методы и виды тестирования программных продуктов; стандарты качества программной документации; основы организации	- знает методы организации работы в команде разработчиков; - основы верификации и аттестации программного обеспечения; - методы и способы идентификации сбоев и ошибок при интеграции приложений; - методы и схемы обработки исключительных ситуаций; - основные методы и виды тестирования программных продуктов; - стандарты качества программной документации; - модели процесса разработки программного обеспечения; - основные принципы процесса разработки программного обеспечении; - основные подходы к интегрированию программных модулей; - основы организации инспектирования и верификации; - встроенные и основные специализированные инструменты анализа качества программных продуктов; - методы организации работы в команде разработчиков

инспектирования и верификации; приемы работы с инструментальными средствами тестирования и отладки; методы организации работы в команде разработчиков; - модели процесса разработки программного обеспечения; основные принципы процесса разработки программного обеспечения; основные подходы к интегрированию программных модулей; основы верификации и аттестации программного обеспечения; методы и способы идентификации сбоев и ошибок при интеграции приложений; основные методы отладки; методы и схемы обработки исключительных ситуаций; приемы работы с инструментальными средствами тестирования и отладки; стандарты качества программной документации; основы организации инспектирования и верификации; встроенные и основные специализированные инструменты анализа качества программных продуктов; методы организации работы в команде разработчиков; - модели процесса разработки программного обеспечения; основные принципы процесса разработки программного обеспечения; основные подходы к интегрированию программных модулей; основы верификации и аттестации программного обеспечения; методы и способы идентификации сбоев и ошибок при интеграции приложений; методы и схемы обработки исключительных ситуаций; основные методы и виды тестирования программных продуктов; приемы работы с инструментальными средствами тестирования и отладки; стандарты качества программной документации; основы организации инспектирования и верификации; встроенные и основные специализированные инструменты анализа качества программных продуктов; методы организации работы в команде разработчиков; - модели процесса разработки программного обеспечения; основные принципы процесса разработки программного обеспечения; основные подходы к интегрированию программных модулей; основы верификации и аттестации программного обеспечения; стандарты качества программной документации; основы организации инспектирования и верификации; встроенные и основные специализированные инструменты анализа качества программных продуктов; методы организации работы в команде разработчиков	

В результате освоения дисциплины обучающийся должен уметь: - распознавать задачу и/или проблему в профессиональном и/или социальном контексте; анализировать задачу и/или проблему и выделять её составные части; определять этапы решения задачи; выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы; составить план действия; определить необходимые ресурсы; владеть актуальными методами работы в профессиональной и смежных сферах; реализовать составленный план; оценивать результат и последствия своих действий (самостоятельно или с помощью наставника); - применять средства информационных технологий для решения профессиональных задач; использовать современное программное обеспечение; определять задачи для поиска информации; определять необходимые источники информации; планировать процесс поиска; структурировать получаемую информацию; выделять наиболее значимое в перечне информации; оценивать практическую значимость результатов поиска; оформлять результаты поиска; - выявлять достоинства и недостатки коммерческой идеи; презентовать идеи открытия собственного дела в профессиональной деятельности; оформлять бизнес-план; рассчитывать размеры выплат по процентным ставкам кредитования; определять инвестиционную привлекательность коммерческих идей в рамках профессиональной деятельности; презентовать бизнес-идею; определять источники финансирования; - организовывать работу коллектива и команды; взаимодействовать с коллегами, руководством, клиентами в ходе профессиональной деятельности; - грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке, проявлять толерантность в рабочем коллективе; - описывать значимость своей специальности; - соблюдать нормы экологической безопасности; определять направления ресурсосбережения в рамках профессиональной деятельности по специальности; - использовать физкультурно-оздоровительную деятельность для укрепления здоровья, достижения жизненных и профессиональных целей; применять рациональные приемы	- умеет анализировать проектную и техническую документацию; - умеет использовать специализированные графические средства построения и анализа архитектуры программных продуктов; - умеет организовывать заданную интеграцию модулей в программные средства на базе имеющейся архитектуры и автоматизации бизнес-процессов; - умеет определять источники и приемники данных; - умеет проводить сравнительный анализ; - выполнять отладку, используя методы и инструменты условной компиляции (классы Debug и Trace); - умеет оценивать размер минимального набора тестов; - умеет разрабатывать тестовые пакеты и тестовые сценарии; - умеет использовать методы для получения кода с заданной функциональностью и степенью качества; - умеет использовать различные транспортные протоколы и стандарты форматирования сообщений; - умеет создавать классы- исключения на основе базовых классов; - умеет выполнять ручное и автоматизированное тестирование программного модуля; использовать выбранную систему контроля версий; - умеет использовать инструментальные средства отладки программных продуктов; - умеет выполнять тестирование интеграции; - умеет организовывать постобработку данных; - использовать приемы работы в системах контроля версии; - умеет выполнять отладку, используя методы и инструменты условной компиляции; - умеет выявлять ошибки в системных компонентах на основе спецификаций; выполнять тестирование интеграции; - организовывать постобработку данных; - оценивать размер минимального набора тестов; - разрабатывать тестовые пакеты и тестовые сценарии; - выполнять ручное и автоматизированное тестирование

двигательных функций в профессиональной деятельности; пользоваться средствами профилактики перенапряжения характерными для данной специальности; - понимать общий смысл четко произнесенных высказываний на известные темы (профессиональные и бытовые), понимать тексты на базовые профессиональные темы; участвовать в диалогах на знакомые общие и профессиональные темы; строить простые высказывания о себе и о своей профессиональной деятельности; кратко обосновывать и объяснить свои действия (текущие и планируемые); писать простые связные сообщения на знакомые или интересующие профессиональные темы; - анализировать проектную и техническую документацию; использовать специализированные графические средства построения и анализа архитектуры программных продуктов; организовывать заданную интеграцию модулей в программные средства на базе имеющейся архитектуры и автоматизации бизнес-процессов; определять источники и приемники данных; проводить сравнительный анализ; выполнять отладку, используя методы и инструменты условной компиляции (классы Debug и Trace); оценивать размер минимального набора тестов; - использовать выбранную систему контроля версий; использовать методы для получения кода с заданной функциональностью и степенью качества; организовывать заданную интеграцию модулей в программные средства на базе имеющейся архитектуры и автоматизации бизнес-процессов; использовать различные транспортные протоколы и стандарты форматирования сообщений; выполнять тестирование интеграции; организовывать постобработку данных; создавать классы-исключения на основе базовых классов; выполнять ручное и автоматизированное тестирование программного модуля; выявлять ошибки в системных компонентах на основе спецификаций; использовать приемы работы в системах контроля версий; - использовать выбранную систему контроля версий; использовать методы для получения кода с заданной функциональностью и степенью качества; анализировать проектную и техническую документацию; использовать инструментальные средства отладки программных продуктов; определять источники и приемники данных; выполнять тестирование	программного модуля; - использовать выбранную систему контроля версий; - использовать методы для получения кода с заданной функциональностью и степенью качества; - анализировать проектную и техническую документацию; - использовать приемы работы в системах контроля версий; - выявлять ошибки в системных компонентах на основе спецификаций

интеграции; организовывать постобработку данных; использовать приемы работы в системах контроля версий; выполнять отладку, используя методы и инструменты условной компиляции; выявлять ошибки в системных компонентах на основе спецификаций; - использовать выбранную систему контроля версий; анализировать проектную и техническую документацию; выполнять тестирование интеграции; организовывать постобработку данных; использовать приемы работы в системах контроля версий; оценивать размер минимального набора тестов; разрабатывать тестовые пакеты и тестовые сценарии; выполнять ручное и автоматизированное тестирование программного модуля; выявлять ошибки в системных компонентах на основе спецификаций; - использовать выбранную систему контроля версий; использовать методы для получения кода с заданной функциональностью и степенью качества; анализировать проектную и техническую документацию; организовывать постобработку данных; приемы работы в системах контроля версий; выявлять ошибки в системных компонентах на основе спецификации	
В результате освоения дисциплины обучающийся должен получить практический опыт: - разрабатывать и оформлять требования к программным модуля по предложенной документации; разрабатывать тестовые наборы (пакеты) для программного модуля; разрабатывать тестовые сценарии программного средства; инспектировать разработанные программные модули на предмет соответствия стандартам кодирования; - интегрировать модули в программное обеспечение; отлаживать программные модули; инспектировать разработанные программные модули на предмет соответствия стандартам кодирования; - отлаживать программные модули; инспектировать разработанные программные модули на предмет соответствия стандартам кодирования; - разрабатывать тестовые наборы (пакеты) для программного модуля; разрабатывать тестовые сценарии программного средства; инспектировать разработанные программные модули на предмет соответствия стандартам кодирования; - инспектировать разработанные программные	- разрабатывает и оформляет требования к программным модулям по предложенной документации; - разрабатывает тестовые наборы (пакеты) для программного модуля; - разрабатывает тестовые сценарии программного средства; - интегрирует модули в программное обеспечение; - инспектирует разработанные программные модули на предмет соответствия стандартам кодирования; - отлаживать программные модули; - разрабатывать тестовые наборы (пакеты) для программного модуля; - разрабатывать тестовые сценарии программного средства; - инспектировать разработанные программные модули на предмет соответствия стандартам кодирования

модули на предмет соответствия стандартам кодирования.	

В рамках программы учебной дисциплины обучающиеся получают первоначальный практический опыт (ПО), продолжают развивать общие компетенции (ОК), приступают к освоению элементов профессиональных компетенций (ПК):

Коды ОК, ПК	Содержание общих компетенций и осваиваемые элементы профессиональных компетенций
ОК 01	Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02	Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 03	Планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по правовой и финансовой грамотности в различных жизненных ситуациях
ОК 04	Эффективно взаимодействовать и работать в коллективе и команде
ОК 05	Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста.
ОК 06	Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных российских духовно-нравственных ценностей, в том числе с учетом гармонизации межнациональных и межрелигиозных отношений, применять стандарты антикоррупционного поведения.
ОК 07	Содействовать сохранению окружающей среды, ресурсосбережению, применять знания об изменении климата, принципы бережливого производства, эффективно действовать в чрезвычайных ситуациях
ОК08	Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности
ОК 09	Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1.	Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент

ПК 2.2.	Выполнять интеграцию модулей в программное обеспечение
ПК 2.3	Выполнять отладку программного модуля с использованием специализированных программных средств
ПК 2.4.	Осуществлять разработку текстовых наборов и текстовых сценариев для программного обеспечения
ПК 2.5.	Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования

3. Распределение типов контрольных заданий по элементам знаний и умений

Основной целью оценки освоения дисциплины является оценка умений и знаний. Оценка освоения умений и знаний и опыта практической деятельности осуществляется с использованием следующих форм и методов контроля: выполнение практических работ, лабораторных работ, тестирования

Распределение оценивания результатов обучения

Результаты освоения учебной дисциплины	**Результаты освоения профессионального модуля направлены на формирование общих компетенций (ОК)**	**Формы и методы оценки**
ПК2.1. Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент ***Знания:*** модели процесса разработки программного обеспечения; основные принципы процесса разработки программного обеспечения; основные подходы к интегрированию программных модулей; виды и варианты интеграционных решений; современные технологии и инструменты интеграции; основные протоколы доступа к данным; методы и способы идентификации сбоев и ошибок при интеграции приложений; методы отладочных классов; стандарты качества программной документации; основы организации инспектирования и верификации; встроенные и основные специализированные инструменты анализа качества программных продуктов; графические средства проектирования архитектуры программных продуктов; методы организации работы в команде разработчиков; ***Умения:*** - анализировать проектную и техническую документацию; использовать специализированные графические средства построения и анализа архитектуры программных продуктов; организовывать заданную интеграцию модулей в программные средства на базе имеющейся архитектуры и автоматизации бизнес-процессов; определять источники и приемники данных; проводить сравнительный анализ; выполнять отладку, используя методы и инструменты условной компиляции (классы Debug и	ОК 01 ОК 02 ОК 03 ОК 04 ОК 05 ОК06 ОК07 ОК 08 ОК 09	- Практические работы - Лабораторные работы - Тестирование - Оценка правильности заполнения документации

Trace); оценивать размер минимального набора тестов; ***Практический опыт:*** - разрабатывать и оформлять требования к программным модуля по предложенной документации; разрабатывать тестовые наборы (пакеты) для программного модуля; разрабатывать тестовые сценарии программного средства; инспектировать разработанные программные модули на предмет соответствия стандартам кодирования;		
ПК2.2. Выполнять интеграцию модулей в программное обеспечение ***Знания:*** - модели процесса разработки программного обеспечения; основные принципы процесса разработки программного обеспечения; основные подходы к интегрированию программных модулей; основы верификации программного обеспечения; современные технологии и инструменты интеграции; основные протоколы доступа к данным; методы и способы идентификации сбоев и ошибок при интеграции приложений; основные методы отладки; методы и схемы обработки исключительных ситуаций; основные методы и виды тестирования программных продуктов; стандарты качества программной документации; основы организации инспектирования и верификации; приемы работы с инструментальными средствами тестирования и отладки; методы организации работы в команде разработчиков; ***Умения:*** - использовать выбранную систему контроля версий; использовать методы для получения кода с заданной функциональностью и степенью качества; организовывать заданную интеграцию модулей в программные средства на базе имеющейся архитектуры и автоматизации бизнес-процессов; использовать различные транспортные протоколы и стандарты форматирования сообщений; выполнять тестирование интеграции; организовывать постобработку данных; создавать классы- исключения на основе базовых классов; выполнять ручное и автоматизированное тестирование программного модуля; выявлять ошибки в	ОК 01 ОК 02 ОК 03 ОК 04 ОК 05 ОК06 ОК07 ОК 08 ОК 09	- Практические работы - Лабораторные работы -Тестирование - Оценка правильности заполнения документации

системных компонентах на основе спецификаций; использовать приемы работы в системах контроля версий; ***Практический опыт:*** - интегрировать модули в программное обеспечение; отлаживать программные модули; инспектировать разработанные программные модули на предмет соответствия стандартам кодирования;		
ПК 2.3. Выполнять отладку программного модуля с использованием специализированных программных средств ***Знания:*** - модели процесса разработки программного обеспечения; основные принципы процесса разработки программного обеспечения; основные подходы к интегрированию программных модулей; основы верификации и аттестации программного обеспечения; методы и способы идентификации сбоев и ошибок при интеграции приложений; основные методы отладки; методы и схемы обработки исключительных ситуаций; приемы работы с инструментальными средствами тестирования и отладки; стандарты качества программной документации; основы организации инспектирования и верификации; встроенные и основные специализированные инструменты анализа качества программных продуктов; методы организации работы в команде разработчиков; ***Умения:*** - использовать выбранную систему контроля версий; использовать методы для получения кода с заданной функциональностью и степенью качества; анализировать проектную и техническую документацию; использовать инструментальные средства отладки программных продуктов; определять источники и приемники данных; выполнять тестирование интеграции; организовывать постобработку данных; использовать приемы работы в системах контроля версий; выполнять отладку, используя методы и инструменты условной компиляции; выявлять ошибки в системных компонентах на основе спецификаций;	ОК 01 ОК 02 ОК 03 ОК 04 ОК 05 ОК06 ОК07 ОК 08 ОК 09	- Практические работы - Лабораторные работы - Тестирование - Оценка правильности заполнения документации

Практический опыт: - отлаживать программные модули; инспектировать разработанные программные модули на предмет соответствия стандартам кодирования;		
ПК 2.4. Осуществлять разработку текстовых наборов и текстовых сценариев для программного обеспечения ***Знания:*** - модели процесса разработки программного обеспечения; основные принципы процесса разработки программного обеспечения; основные подходы к интегрированию программных модулей; основы верификации и аттестации программного обеспечения; методы и способы идентификации сбоев и ошибок при интеграции приложений; методы и схемы обработки исключительных ситуаций; основные методы и виды тестирования программных продуктов; приемы работы с инструментальными средствами тестирования и отладки; стандарты качества программной документации; основы организации инспектирования и верификации; встроенные и основные специализированные инструменты анализа качества программных продуктов; методы организации работы в команде разработчиков; ***Умения:*** - использовать выбранную систему контроля версий; анализировать проектную и техническую документацию; выполнять тестирование интеграции; организовывать постобработку данных; использовать приемы работы в системах контроля версий; оценивать размер минимального набора тестов; разрабатывать тестовые пакеты и тестовые сценарии; выполнять ручное и автоматизированное тестирование программного модуля; выявлять ошибки в системных компонентах на основе спецификаций; ***Практический опыт:*** - разрабатывать тестовые наборы (пакеты) для программного модуля; разрабатывать тестовые сценарии программного средства; инспектировать разработанные программные модули на предмет	ОК 01 ОК 02 ОК 03 ОК 04 ОК 05 ОК06 ОК07 ОК 08 ОК 09	- Практические работы - Лабораторные работы - Тестирование - Оценка правильности заполнения документации

соответствия стандартам кодирования;		
ПК2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования ***Знания:*** - модели процесса разработки программного обеспечения; основные принципы процесса разработки программного обеспечения; основные подходы к интегрированию программных модулей; основы верификации и аттестации программного обеспечения; стандарты качества программной документации; основы организации инспектирования и верификации; встроенные и основные специализированные инструменты анализа качества программных продуктов; методы организации работы в команде разработчиков ***Умения:*** - использовать выбранную систему контроля версий; использовать методы для получения кода с заданной функциональностью и степенью качества; анализировать проектную и техническую документацию; организовывать постобработку данных; приемы работы в системах контроля версий; выявлять ошибки в системных компонентах на основе спецификации ***Практический опыт:*** - инспектировать разработанные программные модули на предмет соответствия стандартам кодирования.		

2. Формы промежуточной аттестации по профессиональному модулю

Элемент модуля	Формы промежуточной аттестации
МДК. 02.01 Технология разработки программного обеспечения	Дифференцированный зачет (ДЗ)
МДК. 02.02 Инструментальные средства разработки программного обеспечения	Дифференцированный зачет (ДЗ)
МДК. 02.03 Математическое моделирование	Дифференцированный зачет (ДЗ)
УП. 02.01 Учебная практика	Дифференцированный зачет (ДЗ)
ПП.02.01 Производственная практика	Дифференцированный зачет (ДЗ)
ПМ.02 Осуществление интеграции программных модулей	Экзамен по модулю

3. Задания для оценки освоения дисциплины МДК.02.01 Технология разработки программного обеспечения

Тема 2.1.1. Основные понятия и стандартизация требований к программному обеспечению

Задание 2.1.1.1. Практическая работа № 1 «Анализ предметной области»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.2., ПК2.3., ПК2.4., ПК2.5.*

Цель: описать и проанализировать ИС, определить необходимые элементы КТС ИС и системного ПО ИС.

Теоретические сведения

Проблемы управления программными проектами впервые появились в 60-х– начале70-х годов прошлого века, когда провалились многие большие проекты по разработке программных продуктов. Были зафиксированы задержки в создании ПО, программное обеспечение было ненадежным, затраты на разработку в несколько раз превосходили первоначальные оценки и т.д.

Провалы этих проектов обуславливались не только некомпетентностью руководителей и программистов. Напротив, в этих больших поисковых проектах принимали участие люди, уровень квалификации которых был явно выше среднего. Причины провалов коренились в тех подходах, которые использовались в управлении проектами. Применяемая методика была основана на опыте управления техническими проектами и оказалась неэффективной при разработке программных проектов.

Руководители программных проектов выполняют такую же работу, что и руководители технических проектов. Вместе с тем процесс разработки ПО существенно отличается от процессов реализации технических проектов. Ниже приведен небольшой список этих отличий.

1. Программный продукт нематериален. Менеджер судостроительного проекта или проекта постройки здания видит результат выполнения своего проекта. Если реализация проекта отстает от графика, то это видно по незавершенности конструкции. В противоположность этому процент незавершенности программного проекта нельзя увидеть или потрогать. Менеджер программного проекта может полагаться только на документацию, которая фиксирует процесс разработки программного продукта.

2. Не существует стандартных процессов разработки программного обеспечения. На сегодняшний день не существует четкой зависимости между процессом создания ПО и типом создаваемого программного продукта. Другие технические дисциплины имеют длительную историю, процессы разработки технических изделий многократно опробованы и проверены.

Изучением же процессов создания ПО специалисты занимаются только последние несколько лет.

Поэтому прока нельзя точно предсказать, на каком этапе процесса разработки ПО могут возникнуть проблемы, угрожающие всему проекту.

3. Большие программные проекты – это часто одноразовые проекты. Большие программные проекты, как правило, значительно отличаются от проектов, реализованных ранее.

Поэтому, чтобы уменьшить неопределенность в планировании проекта, руководители проектов должны обладать очень большим практическим опытом. Но постоянные технологические изменения в компьютерной технике обесценивают предыдущий опыт. Перечисленные особенности могут привести к тому, что реализация проекта выйдет за рамки временного графика или бюджетных ассигнований. Об этом всегда нужно помнить.

Процессы управления программными проектами

Невозможно описать и стандартизировать все работы, выполняемые менеджером проекта по созданию ПО, но в большинстве случаев к ним относятся.

− Написание предложений по созданию ПО.

− Планирование и составление графика работ проекта.
− Оценивание стоимости проекта.
− Контроль процессов выполнения работ.
− Подбор персонала.
− Написание отчетов и представлений.

Время выполнения больших программных проектов может занимать несколько лет. В течение этого времени цели и намерения организации, оказавшей программный проект, могут существенно измениться. Может оказаться, что разрабатываемый программный продукт стал уже ненужным либо исходные требования к ПО устарели и их нужно кардинально менять. В такой ситуации руководство организации-разработчика может принять решение о прекращении разработки ПО или об изменении проекта в целом.

Планирование проекта

Эффективное управление проектами напрямую зависит от правильного планирования работ.

План, разработанный на начальном этапе проекта, рассматривается всеми его участниками как руководящий документ, выполнение которого должно привести к успешному завершению проекта. Этот первоначальный план должен максимально подробно описывать все этапы реализации проекта.

План проекта должен показать ресурсы, необходимые для реализации проекта, разделение работ на этапы и временной график выполнения этих этапов. Детализация планов проектов очень разнится в зависимости от типа разрабатываемого программного продукта и организации Разработчика. Но в любом случае большинство планов содержит следующие разделы.

1. Введение. Краткое описание целей проекта и проектных ограничений (бюджетных, временных и т.д.).

2. Организация выполнения проекта. Описание способа подбора команды разработчиков и распределение обязанностей между членами команды.

3. Анализ рисков. Описание возможных проектных рисков, вероятность их проявления и стратегий, направленных на их уменьшение.

4. Аппаратные и программные ресурсы для реализации проекта. Перечень аппаратных средств и программного обеспечения, необходимого для разработки программного продукта.

5. Разбиение работ на этапы. Проект разбивается на отдельные процессы, определяются этапы выполнения проекта, приводится описание результатов каждого этапа и контрольные отметки.

6. График работ. В графике работ отображаются зависимости между отдельными этапами разработки по, оценки времени их выполнения и распределение членов команды проекта по отдельным этапам.

7. Механизмы контроля и мониторинга за ходом выполнения проекта. Описываются механизмы и сроки предоставления отчетов о ходе работ, а также механизмы мониторинга всего проекта.

При планировании проекта разработки ПО определяются контрольные точки – вехи, отмечающие окончание определенного этапа работ. Для каждой вехи создается отчет, который предоставляется руководству проекта.

При определении контрольных точек весь процесс создания ПО должен быть разбит на отдельные этапы с указанным «выходом» (результатом) каждого этапа.

Информационный процесс - это осуществление всей совокупности следующих элементарных информационных актов: прием или создание информации, ее хранение, передача и использование. Информационная система - это совокупность механизмов, обеспечивающих полное осуществление информационного процесса.

Вне ИС информация может лишь сохраняться в виде записей на тех или иных физических носителях, но не может быть ни принятой, ни переданной, ни использованной.

Информационная система - организационно-техническая система, которая предназначена для выполнения информационно-вычислительных работ или предоставления

информационновычислительных работ, или предоставления информационно-вычислительных услуг, удовлетворяющих потребности системы управления и ее пользователей – управленческого персонала, внешних пользователей путем использования и/или создания информационных продуктов. Информационные системы существуют в рамках системы управления и полностью подчинены целям функционирования этих систем.

Информационно-вычислительная работа - деятельность, связанная с использованием информационных продуктов. Типичным примером информационной работы является поддержка информационных технологий управления.

Информационно-вычислительная услуга - это разовая информационно-вычислительная работа. Под информационным продуктом понимается вещественный или нематериальный результат интеллектуального человеческого труда, обычно материализованный на определенном носителе, например разнообразных программных продуктов, выходной информации в виде документов управления, баз данных, хранилищ данных, баз знаний, проектов ИС и ИТ.

Методологическую основу изучения ИС составляет системный подход, в соответствии с которым любая система представляет собой совокупность взаимосвязанных объектов, функционирующих совместно для достижения общей цели.

Информационная система представляет собой совокупность функциональной структуры, информационного, математического, технического, организационного и кадрового обеспечения, которые объединены в единую системы в целях сбора, хранения, обработки и выдачи необходимой информации для выполнения функций управления. Она обеспечивает информационные потоки:

I-1 - информационный поток из внешней среды в систему управления, который, с одной стороны, представляет собой поток нормативной информации, создаваемый государственными учреждениями в части законодательства, а с другой стороны - поток информации о конъюнктуре рынка, создаваемый конкурентами, потребителями, поставщиками;

I-2 - информационный поток из системы управления во внешнюю среду (отчетная информация, прежде всего финансовая в государственные органы, инвесторам, кредиторам, потребителям; маркетинговая информация потенциальным потребителям);

I-3 - информационный поток из системы управления на объект, представляет собой совокупность плановой, нормативной и распорядительной информации для осуществления хозяйственных процессов;

I-4 - информационный поток от объекта в систему управления, который отражает учетную информацию о состоянии объекта управления экономической системой (сырья, материалов, денежных, энергетических, трудовых ресурсов, готовой продукции и выполненных услугах) в результате выполнения хозяйственных процессов.

Задачи информационных систем

Корпоративные системы позволяют решить следующие задачи:

- гарантировать требуемое качество управления предприятием;
- повысить оперативность и эффективность взаимодействия между подразделениями;
- обеспечить управляемость качеством выпускаемой продукции;
- увеличить экономическую эффективность деятельности предприятия;
- создать систему статистического учета на предприятии;
- осуществлять прогноз развития предприятия;
- создать систему стратегического и оперативного планирования, систему прогнозирования.

Задания для практической работы

1. Выберите предметную область
2. Выберите название ИС в рамках предметной области.
3. Определите цель ИС
4. Проведите анализ осуществимости ИС

4.1.Что произойдет с организацией, если система не будет введена в эксплуатацию?

4.2.Какие текущие проблемы существуют в организации и как новая система поможет их решить?

4.3.Каким образом (и будет ли) ИС способствовать целям бизнеса?

4.4.Требует ли разработка ИС технологии, которая до этого раньше не использовалась в организации?

5. Где будет размещена ИС? Кто является пользователем ИС?

6. Комплекс технических средств ИТ

6.1.Какие средства компьютерной техники необходимы для ИС?

6.2.Какие средства коммуникационной техники необходимы для ИС?

6.3.Какие средства организационной техники необходимы для ИС?

6.4.Какие средства оперативной полиграфии необходимы для ИС?

7. Опишите системное ПО ИТ.

Таблица 1. Варианты предметных областей

Предметная область	**Сущность задачи**
Страховая медицинская компания	Страховая медицинская компания (СМК) заключает договоры добровольного медицинского страхования с населением и договоры с лечебными учреждениями на лечение застрахованных клиентов. При возникновении страхового случая клиент подает заявку на оказание медицинских услуг по условиям договора инспектору, который работает с данным клиентом. Инспектор направляет данного клиента в лечебное учреждение. Отчеты о своей деятельности инспектор предоставляет в бухгалтерию. Бухгалтерия проверяет оплату договоров, перечисляет денежные средства за оказанные услуги лечебным учреждениям, производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. СМК не только оплачивает лечение застрахованного лица при возникновении с ним страхового случая, но и, при возникновении каких-либо осложнений после лечения, оплачивает лечение этих осложнений
Агентство недвижимости	Агентство недвижимости занимается покупкой, продажей, сдачей в аренду объектов недвижимости по договорам с их собственниками. Агентство управляет объектами недвижимости как физических, так и юридических лиц. Собственник может иметь несколько объектов. В случае покупки или аренды клиент может произвести осмотр объекта. В качестве одной из услуг, предлагаемых агентством, является проведение инспектирования текущего состояния объекта для адекватного определения его рыночной цены. По результатам своей деятельности агентство производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики
Кадровое агентство	Кадровое агентство способствует трудоустройству безработных граждан. Агентство ведет учет и классификацию данных о безработных на основании резюме от них. От предприятий города поступают данные о свободных вакансиях, на основании которых агентство предлагает различные варианты трудоустройства соискателям. В случае положительного исхода поиска вакансия считается заполненной, а безработный становится трудоустроенным. По результатам своей деятельности кадровое агентство производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики
Компания по разработке	Компания заключает договор с клиентом на разработку программного продукта согласно техническому заданию. После

программных продуктов	утверждения технического задания определяется состав и объем работ, составляется предварительная смета. На каждый проект назначается ответственный за его выполнение – куратор проекта, который распределяет нагрузку между программистами и следит за выполнением технического задания. Когда программный продукт готов, то его внедряют, производят обучение клиента и осуществляют дальнейшее сопровождение. По результатам своей деятельности компания производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики

Задание 2.1.1.2. Практическая работа № 2 «Разработка и оформление технического задания»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.2., ПК2.3., ПК2.4., ПК2.5.*

Цель: приобретение навыков разработки технического задания на программный продукт, ознакомиться с правилами написания технического задания

Теоретические сведения

Техническое задание (ТЗ, техзадание) - исходный документ для проектирования сооружения или промышленного комплекса, конструирования технического устройства (прибора, машины, системы управления и т. д.), разработки информационных систем, стандартов либо проведения научно-исследовательских работ (НИР).

ТЗ содержит основные технические требования, предъявляемые к сооружению, изделию или услуге и исходные данные для разработки. В ТЗ указываются назначение объекта, область его применения, стадии разработки конструкторской (проектной, технологической, программной и т.п.) документации, ее состав, сроки исполнения и т. д., а также особые требования, обусловленные спецификой самого объекта либо условиями его эксплуатации. Как правило, ТЗ составляют на основе анализа результатов предварительных исследований, расчетов и моделирования. Типовые требования к составу и содержанию технического задания приведены в таблице 1.

Таблица 1. Состав и содержание технического задания (ГОСТ 34.602- 89)

№	Раздел	Содержание
1	Общие сведения	- полное наименование системы и ее условное обозначение - шифр темы или шифр (номер) договора; - наименование предприятий разработчика и заказчика системы, их реквизиты - перечень документов, на основании которых создается ИС - плановые сроки начала и окончания работ - сведения об источниках и порядке финансирования работ - порядок оформления и предъявления заказчику результатов работ по созданию системы, ее частей и отдельных средств
2	Назначение и цели создания (развития) системы	- вид автоматизируемой деятельности - перечень объектов, на которых предполагается использование системы - наименования и требуемые значения технических, технологических, производственно-экономических и др. показателей объекта, которые должны быть достигнуты при внедрении ИС

3	Характеристика объектов автоматизации	- краткие сведения об объекте автоматизации - сведения об условиях эксплуатации и характеристиках окружающей среды
4	Требования к системе	Требования к системе в целом: - требования к структуре и функционированию системы (перечень подсистем, уровни иерархии, степень централизации, способы информационного обмена, режимы функционирования, взаимодействие со смежными системами, перспективы развития системы) - требования к персоналу (численность пользователей, квалификация, режим работы, порядок подготовки) - показатели назначения (степень приспособляемости системы к изменениям процессов управления и значений параметров) - требования к надежности, безопасности, эргономике, транспортабельности, эксплуатации, техническому обслуживанию и ремонту, защите и сохранности информации, защите от внешних воздействий, к патентной чистоте, по стандартизации и унификации Требования к функциям (по подсистемам): - перечень подлежащих автоматизации задач - временной регламент реализации каждой функции - требования к качеству реализации каждой функции, к форме представления выходной информации, характеристики точности, достоверности выдачи результатов - перечень и критерии отказов Требования к видам обеспечения: - математическому (состав и область применения мат. моделей и методов, типовых и разрабатываемых алгоритмов) - информационному (состав, структура и организация данных, обмен данными между компонентами системы, информационная совместимость со смежными системами, используемые классификаторы, СУБД, контроль данных и ведение информационных массивов, процедуры придания юридической силы выходным документам) - лингвистическому (языки программирования, языки взаимодействия пользователей с системой, системы кодирования, языки ввода- вывода) - программному (независимость программных средств от платформы, качество программных средств и способы его контроля, использование фондов алгоритмов и программ) - техническому - метрологическому - организационному (структура и функции эксплуатирующих подразделений, защита от ошибочных действий персонала) - методическому (состав нормативно- технической документации
5	Состав и содержание работ по созданию системы	- перечень стадий и этапов работ - сроки исполнения - состав организаций — исполнителей работ - вид и порядок экспертизы технической документации - программа обеспечения надежности

		- программа метрологического обеспечения
6	Порядок контроля и приемки системы	- виды, состав, объем и методы испытаний системы - общие требования к приемке работ по стадиям - статус приемной комиссии
7	Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие	- преобразование входной информации к машиночитаемому виду - изменения в объекте автоматизации - сроки и порядок комплектования и обучения персонала
8	Требования к документированию	- перечень подлежащих разработке документов - перечень документов на машинных носителях
8	Источники разработки	- документы и информационные материалы, на основании которых разрабатывается ТЗ и система

Порядок разработки технического задания

Разработка технического задания выполняется в следующей последовательности. Прежде всего, устанавливают набор выполняемых функций, а также перечень и характеристики исходных данных.

Затем определяют перечень результатов, их характеристики и способы представления.

Далее уточняют среду функционирования программною обеспечения: конкретную комплектацию и параметры технических средств, версию используемой операционной системы и, возможно, версии и параметры другого установленного программного обеспечения, с которым предстоит взаимодействовать будущему программному продукту.

В случаях, когда разрабатываемое программное обеспеченно собирает и хранит некоторую информацию или включается в управление каким-либо техническим процессом, необходимо также четко регламентировать действия программы в случае сбоев оборудования и энергоснабжения.

1. Общие положения

1.1 Техническое задание оформляют в соответствии с ГОСТ 19.106-78 на листах формата А4 и А3 по ГОСТ 2.301-68, как правило, без заполнения полей листа. Номера листов (страниц) проставляют в верхней части листа над текстом.

1.2 Лист утверждения и титульный лист оформляют в соответствии с ГОСТ 19.104-78.

Информационную часть (аннотацию и содержание), лист регистрации изменений допускается и в документ не включать.

1.3 Для внесения изменений и дополнений в техническое задание на последующих стадиях разработки программы или программного изделия выпускают дополнение к нему. Согласование и утверждение дополнения к техническому заданию проводят в том же порядке, который установлен для технического задания.

1.4. Техническое задание должно содержать следующие разделы:

- введение;
- наименование и область применения;
- основание для разработки;
- назначение разработки;
- технические требования к программе или программному изделию;
- технико-экономические показатели;
- стадии и этапы разработки;
- порядок контроля и приемки;
- приложения.

В зависимости от особенностей программы или программного изделия допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них. При необходимости допускается в техническое задание включать приложения.

2. Содержание разделов

2.1 Введение должно включать краткую характеристику области применения программы или программного продукта, а также объекта (например, системы), в котором предполагается их использовать. Основное назначение введения - продемонстрировать актуальность данной разработки и показать, какое место эта разработка занимает в ряду подобных.

2.2 В разделе «Наименование и область применения» указывают наименование, краткую характеристику области применения программы или программного изделия и объекта, в котором используют программу или программное изделие

2.3 В разделе «Основание для разработки» должны быть указаны:

- документ (документы), на основании которых ведется разработка. Таким документом может служить план, приказ, договор и т. п.;
- организация, утвердившая этот документ, и дата его утверждения;
- наименование и (или) условное обозначение темы разработки.

2.4 В разделе «Назначение разработки» должно быть указано функциональное и эксплуатационное назначение программы или программного изделия.

2.5 Раздел «Технические требования к программе или программному изделию» должен содержать следующие подразделы:

- требования к функциональным характеристикам;
- требования к надежности;
- условия эксплуатации;
- требования к составу и параметрам технических средств;
- требования к информационной и программной совместимости;
- требования к маркировке и упаковке;
- требования к транспортированию и хранению;
- специальные требования.

В подразделе «Требования к функциональным характеристикам» должны быть указаны требования к составу выполняемых функций, организации входных и выходных данных, временным характеристикам и т. п.

2.5.2 В подразделе «Требования к надежности» должны быть указаны требования к обеспечению надежного функционирования (обеспечение устойчивого функционирования, контроль входной и выходной информации, время восстановления после отказа и т. п.).

2.5.3 В подразделе «Условия эксплуатации» должны быть указаны условия эксплуатации (температура окружающего воздуха, относительная влажность и т. п. для выбранных типов носителей данных), при которых должны обеспечиваться заданные характеристики, а также вид обслуживания, необходимое количество и квалификация персонала.

2.5.4 В подразделе «Требования к составу и параметрам технических средств» указывают необходимый состав технических средств с указанием их технических характеристик.

2.5.5 В подразделе «Требования к информационной и программной совместимости» должны быть указаны требования к информационным структурам на входе и выходе и методам решения, исходным кодам, языкам программирования. При необходимости должна обеспечиваться защита информации и программ.

2.5.6 В подразделе «Требования к маркировке и упаковке» в общем случае указывают требования к маркировке программного изделия, варианты и способы упаковки.

2.5.7 В подразделе «Требования к транспортированию и хранению» должны быть указаны для программного изделия условия транспортирования, места хранения, условия хранения, условия складирования, сроки хранения в различных условиях.

2.5.8 В разделе «Технико-экономические показатели» должны быть указаны: ориентировочная экономическая эффективность, предполагаемая годовая потребность, экономические преимущества разработки по сравнению с лучшими отечественными и зарубежными образцами или аналогами.

2.6 В разделе «Стадии и этапы разработки» устанавливают необходимые стадии разработки, этапы и содержание работ (перечень программных документов, которые должны

быть разработаны, согласованы и утверждены), а также, как правило, сроки разработки и определяют исполнителей.

2.7 В разделе «Порядок контроля и приемки» должны быть указаны виды испытаний и общие требования к приемке работы.

2.8 В приложениях к техническому заданию при необходимости приводят:

- перечень научно-исследовательских и других работ, обосновывающих разработку;
- схемы алгоритмов, таблицы, описания, обоснования, расчеты и другие документы, которые могут быть использованы при разработке;
- другие источники разработки.

В случаях, если какие-либо требования, предусмотренные техническим заданием, заказчик не предъявляет, следует в соответствующем месте указать «Требования не предъявляются».

Задания для практической работы

1. Разработать техническое задание по варианту выбранному в практической работе №1.

Задание 2.1.1.3. Практическая работа № 3 «Построение архитектуры программного средства»

Проверяемые результаты обучения: *ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.2., ПК2.3., ПК2.4., ПК2.5.*

Цель: приобретение навыков создания формальных моделей и на их основе определение спецификаций разрабатываемого программного обеспечения, приобретение навыков проектирования программного обеспечения

Теоретические сведения

Эскизный проект

Эскизный проект предусматривает разработку предварительных проектных решений по системе и ее частям.

Выполнение стадии эскизного проектирования не является строго обязательной. Если основные проектные решения определены ранее или достаточно очевидны для конкретной ИС и объекта автоматизации, то эта стадия может быть исключена из общей последовательности работ.

Содержание эскизного проекта задается в ТЗ на систему. Как правило, на этапе эскизного проектирования определяются:

- функции ИС;
- функции подсистем, их цели и ожидаемый эффект от внедрения;
- состав комплексов задач и отдельных задач;
- концепция информационной базы и ее укрупненная структура;
- функции системы управления базой данных;
- состав вычислительной системы и других технических средств;
- функции и параметры основных программных средств.

По результатам проделанной работы оформляется, согласовывается и утверждается документация в объеме, необходимом для описания полной совокупности принятых проектных решений и достаточном для дальнейшего выполнения работ по созданию системы.

На основе технического задания (и эскизного проекта) разрабатывается технический проект.

Основная задача эскизного проекта – создать прообраз будущей автоматизированной системы. При разработке эскизного проекта разработчик определяет основные контуры будущей системы, а заказчик в свою очередь получает представление об основных чертах будущего объекта автоматизации и анализирует их возможную применимость в последующей работе.

При разработке эскизного проекта составляются:

- Ведомость эскизного проекта. Общая информация по проекту.

- Пояснительная записка к эскизному проекту. Вводная информация, позволяющая ее потребителю быстро освоить данные по конкретному проекту.

- Схема организационной структуры. Описание организационной структуры организации, которая будет использовать создаваемую автоматизированную систему в практической работе.

- Структурная схема комплекса технических средств. Техническая составляющая автоматизированной системы, включающая в себя набор серверов, рабочих станций, схему локальной вычислительной сети и структурированной кабельной системы.

- Схема функциональной структуры. Описание задач, которые будут использоваться в работе подсистем. Видение участков информационной системы и порядок и их взаимодействия.

- Схема автоматизации. Логический процесс создания автоматизированной системы от начала до конца.

- Согласно ГОСТ 34.201-89, дополнительно в эскизный проект по необходимости может быть включено техническое задание на разработку новых технических средств.

Эскизный проект чаще всего не разделяют, он выполняется в рамках общего (первоначального) этапа всего проекта. Перечень работ, составляющих эскизный проект, может варьироваться в зависимости от конкретного технического задания заказчика (его пожеланий) и сложности проектируемого проекта. Соответственно варьируется и цена этого этапа.

Эскизный проект не всегда создается под конкретного заказчика. Нередко разработчики с помощью эскизного проекта стремятся показать свой творческий потенциал и найти потенциальных заказчиков. Не случайно на различные конкурсы представляются именно эскизные проекты.

Разработка спецификаций

Разработка программного обеспечения начинается с анализа требований к нему. В

результате анализа получают спецификации разрабатываемого программного обеспечения, строят общую модель его взаимодействия с пользователем или другими программами и конкретизируют его основные функции.

При структурном подходе к программированию на этапе анализа и определения спецификаций разрабатывают три типа моделей: модели функций, модели данных и модели потоков данных. Поскольку разные модели описывают проектируемое программное обеспечение с разных сторон, рекомендуется использовать сразу несколько моделей, разрабатываемых в виде диаграмм, и пояснять их текстовыми описаниями, словарями и т. п.

Структурный анализ предполагает использование следующих видов моделей:

- диаграмм потоков данных (DFD - Data Flow Diagrams), описывающих взаимодействие источников и потребителей информации через процессы, которые должны быть реализованы в системе;

- диаграмм «сущность-связь» (ERD Entity-Relationship Diagrams), описывающих базы данных разрабатываемой системы;

- диаграмм переходов состояний (STD - State Transition Diagrams), характеризующих поведение системы во времени;

- функциональных диаграмм (методика SADT);

- спецификаций процессов;

- словаря терминов.

Спецификации процессов

Спецификации процессов обычно представляют в виде краткого текстового описания, схем алгоритмов, псевдокодов, Flow-форм или диаграмм Насси - Шнейдермана.

Словарь терминов

Словарь терминов представляет собой краткое описание основных понятий, используемых при составлении спецификации. Он должен включать определение основных понятий предметной области, описание структур элементов данных, их типом и форматов, а также всех сокращений и условных обозначении.

Диаграммы переходов состояний

С помощью диаграмм переходов состояний можно моделировать последующее функционирование системы на основе ее предыдущего и текущего функционирования.

Моделируемая система в любой заданный момент времени находится точно в одном из конечного множества состояний. С течением времени она может изменить свое состояние, при этом переходы между состояниями должны быть точно определены.

Функциональные диаграммы

Функциональные диаграммы отражают взаимосвязи функций разрабатываемого программного обеспечения.

Они создаются на ранних этапах проектирования систем, для того чтобы помочь проектировщику выявить основные функции и составные части проектируемой системы и, по возможности, обнаружить и устранить существенные ошибки. Для создания функциональных диаграмм предлагается использовать методологию SADT.

Диаграммы потоков данных

Для описания потоков информации в системе применяются диаграммы потоков данных (DFD – Data Flow Diagrams). DFD позволяет описать требуемое поведение системы в виде совокупности процессов, взаимодействующих посредством связывающих их потоков данных. DFD показывает, как каждый из процессов преобразует свои входные потоки данных в выходные потоки данных и как процессы взаимодействуют между собой.

Диаграммы «сущность - связь»

Диаграмма сущность-связь - инструмент разработки моделей данных, обеспечивающий стандартный способ определения данных и отношений между ними. Она включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Такая диаграмма не слишком детализирована, в нее включаются основные сущности и связи между ними, которые удовлетворяют требованиям, предъявляемым к ИС.

Разработка документации. Стадия «Технический проект».

Проект технический - образ намеченного к созданию объекта, представленный в виде его описания, схем, чертежей, расчетов, обоснований, числовых показателей.

Цель технического проекта - определение основных методов, используемых при создании информационной системы, и окончательное определение ее сметной стоимости.

Техническое проектирование подсистем осуществляется в соответствии с утвержденным техническим заданием.

Технический проект программной системы подробно описывает:

- выполняемые функции и варианты их использования;
- соответствующие им документы;
- структуры обрабатываемых баз данных;
- взаимосвязи данных;
- алгоритмы их обработки.

Технический проект должен включать данные об объемах и интенсивности потоков обрабатываемой информации, количестве пользователей программной системы, характеристиках оборудования и программного обеспечения, взаимодействующего с проектируемым программным продуктом.

При разработке технического проекта оформляются:

- ведомость технического проекта. Общая информация по проекту;
- пояснительная записка к техническому проекту. Вводная информация, позволяющая ее потребителю быстро освоить данные по конкретному проекту;
- описание систем классификации и кодирования;
- перечень входных данных (документов). Перечень информации, которая используется как входящий поток и служит источником накопления;
- перечень выходных данных (документов). Перечень информации, которая используется для анализа накопленных данных;
- описание используемого программного обеспечения. Перечень программного обеспечения и СУБД, которые планируется использовать для создания информационной системы;

- описание используемых технических средств. Перечень аппаратных средств, на которых планируется работа проектируемого программного продукта;

- проектная оценка надежности системы. Экспертная оценки надежности с выявлением наиболее благополучных участков программной системы и ее узких мест;

- ведомость оборудования и материалов. Перечень оборудования и материалов, которые потребуются в ходе реализации проекта.

Структурная схема

Структурной называют схему, отражающую состав и взаимодействие по управлению частями разрабатываемого программного обеспечения. Структурная схема определяется архитектурой разрабатываемого ПО.

Функциональная схема

Функциональная схема - это схема взаимодействия компонентов программного обеспечения с описанием информационных потоков, состава данных в потоках и указанием используемых файлов и устройств.

Разработка алгоритмов

Метод пошаговой детализации реализует нисходящий подход к программированию и предполагает пошаговую разработку алгоритма.

Структурные карты

Методика структурных карт используется на этапе проектирования ПО для того, чтобы продемонстрировать, каким образом программный продукт выполняет системные требования.

Структурные карты Константайна предназначены для описания отношений между модулями.

Техника структурных карт Джексона основана на методе структурного программирования Джексона, который выявляет соответствие между структурой потоков данных и структурой программы. Основное внимание в методе сконцентрировано на соответствии входных и выходных потоков данных.

Задания для практической работы

Задание №1

1. На основе технического задания из практической работы №2 выполнить анализ функциональных и эксплуатационных требований к программному продукту.

2. Определить основные технические решения (выбор языка программирования, структура программного продукта, состав функций ПП, режимы функционирования) и занести результаты в документ, называемый «Эскизным проектом».

3. Определить диаграммы потоков данных для решаемой задачи.

4. Определить диаграммы «сущность-связь», если программный продукт содержит базу данных.

5. Добавить словарь терминов.

6. Оформить результаты, используя MS Office или MS Visio в виде эскизного проекта.

Задание №2

1. Разработать функциональную схему программного продукта.

2. Представить структурную схему в виде структурных карт Константайна.

3. Представить структурную схему в виде структурных карт Джексона.

4. Оформить результаты, используя MS Office или MS Visio в виде технического проекта.

Задание 2.1.1.4. Практическая работа № 4 «Изучение работы в системе контроля версий»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.2., ПК2.3., ПК2.4., ПК2.5.*

Цель: применение методов объектно-ориентированного проектирования

Теоретические сведения

Сущность объектно-ориентированного подхода к программированию заключается в том, что основные идеи объектно-ориентированного подхода опираются на следующие положения:

- программа представляет собой модель некоторого реального процесса, части реального мира;
- модель реального мира или его части может быть описана как совокупность взаимодействующих между собой объектов;
- объект описывается набором параметров, значения которых определяют состояние объекта, и набором операций (действий), которые может выполнять объект;
- взаимодействие между объектами осуществляется посылкой специальных сообщений от одного объекта к другому. Сообщение, полученное объектом, может потребовать выполнения определенных действий, например, изменения состояния объекта;
- объекты, описанные одним и тем же набором параметров и способные выполнять один и тот же набор действий представляют собой класс однотипных объектов.

С точки зрения языка программирования класс объектов можно рассматривать как тип данного, а отдельный объект - как данное этого типа. Определение программистом собственных классов объектов для конкретного набора задач должно позволить описывать отдельные задачи в терминах самого класса задач (при соответствующем выборе имен типов и имен объектов, их параметров и выполняемых действий).

Таким образом, объектно-ориентированный подход предполагает, что при разработке программы должны быть определены классы используемых в программе объектов и построены их описания, затем созданы экземпляры необходимых объектов и определено взаимодействие между ними.

Классы объектов часто удобно строить так, чтобы они образовывали иерархическую структуру. Например, класс «Студент», описывающий абстрактного студента, может служить основой для построения классов «Студент 1 курса», «Студент 2 курса» и т.д., которые обладают всеми свойствами студента вообще и некоторыми дополнительными свойствами, характеризующими студента конкретного курса. При разработке интерфейса с пользователем программы могут использовать объекты общего класса «Окно» и объекты классов специальных окон, например, окон информационных сообщений, окон ввода данных и т.п. В таких иерархических структурах один класс может рассматриваться как базовый для других, производных от него классов. Объект производного класса обладает всеми свойствами базового класса и некоторыми собственными свойствами, он может реагировать на те же типы сообщений от других объектов, что и объект базового класса и на сообщения, имеющие смысл только для производного класса. Обычно говорят, что объект производного класса наследует все свойства своего базового класса.

Некоторые параметры объекта могут быть локализованы внутри объекта и недоступны для прямого воздействия извне объекта. Например, во время движения объекта-автомобиля объектводитель может воздействовать только на ограниченный набор органов управления (рулевое колесо, педали газа, сцепления и тормоза, рычаг переключения передач) и ему недоступен целый ряд параметров, характеризующих состояние двигателя и автомобиля в целом

Очевидно, для того, чтобы продуктивно применять объектный подход для разработки программ, необходимы языки программирования, поддерживающие этот подход, т.е. позволяющие строить описание классов объектов, образовывать данные объектных типов, выполнять операции над объектами. Одним из первых таких языков стал язык SmallTalk в котором все данные являются объектами некоторых классов, а общая система классов строится как иерархическая структура на основе предопределенных базовых классов.

Опыт программирования показывает, что любой методический подход в технологии программирования не должен применяться слепо с игнорированием других подходов. Это относится и к объектно-ориентированному подходу. Существует ряд типовых проблем, для которых его полезность наиболее очевидна, к таким проблемам относятся, в частности, задачи имитационного моделирования, программирование диалогов с пользователем. Существуют и задачи, в которых применение объектного подхода ни к чему, кроме излишних затрат труда, не

приведет. В связи с этим наибольшее распространение получили объектно-ориентированные языки программирования, позволяющие сочетать объектный подход с другими методологиями. В некоторых языках и системах программирования применение объектного подхода ограничивается средствами интерфейса с пользователем (например, Visual FoxPro ранних версий).

Наиболее используемыми в настоящее время объектно-ориентированными языками являются Паскаль с объектами и Си++, причем наиболее развитые средства для работы с объектами содержатся в Си++.

Объектно-базирующееся программирование - это методология разработки программ, основанная на использовании совокупности объектов, каждый из которых является реализацией определенного класса. Программный код и данные структурируются так, чтобы имитировалось поведение фактически существующих объектов. Содержимое объекта защищено от внешнего мира посредством инкапсуляции. Благодаря наследованию уже запрограммированные функциональные возможности можно использовать и для других объектов. Объекты являются программным представлением физических и/или логических сущностей реального мира. Они необходимы для моделирования поведения физических или логических объектов, которые они представляют. Для изменения поведения и состояния элементов управления используются их свойства, методы, поля и события. Классы задают структуру объектов. При программировании создаются объекты - представители классов. С другой стороны, классы составляют группы одноименных объектов.

Внутренняя структура класса в Visual Basic передается объекту с использованием модуля класса. С использованием команды Project Add Class Module модуль класса можно добавить в проект.

После добавления модуля класса выводится окно кода, в котором можно реализовать компоненты (свойства, поля, методы, события) класса.

Задания для практической работы

Пример использования методики объектно–ориентированного программирования

Создать в предметной области «Автомобили» класс с требуемой функциональностью (использовать компоненты класса: методы, поля и т.д.). Создать объект - экземпляр класса. Создать пример использования объектом компонентов класса.

Реализация задания

Приводится проект, дающий справку желающим приобрести автомобиль. Создан класс Class1, содержащий компоненты, определяющие название фирмы-изготовителя, модель автомобиля, его стоимость, изображение автомобиля и следующие технические характеристики:

- тип двигателя (бензин/дизель),
- число цилиндров/рабочий объѐм,
- система питания (карбюратор/впрыскивание),
- мощность (л.с),
- максимальная скорость (км/час),
- разгон 0 - 100 (км/час)/сек,
- привод (передний/задний/4х4).

Далее создаѐтся экземпляр класса: Dim av As New Class1, использующий компоненты класса.

Пользователю предлагается решить вопрос о необходимости покупки, выбрать фирмуизготовителя, ответить на вопрос о выводе изображения покупаемого автомобиля, либо его технических характеристик, либо обеих категорий одновременно (используются процедуры Property Get и Property Let, созданные в классе Class1), после чего программа адекватно реагирует: либо выводятся вышеперечисленные данные, либо выводится некоторое сообщение.

Для реализации проекта нужно выполнить следующую последовательность действий:

1. добавить в стандартный проект модуль класса (Project Add Class Module Class Module Открыть),

2. создать:

- методы класса. Четыре метода создаются в процедурах: Public Function Met1(), Public Function Met2(), Public Function Met3(), Public Function Met4() (Tools Add Procedure ввести имя { Met1, Met2, Met3, Met4} выбрать Function выбрать Public OK),

- свойства класса. Свойства задаются с использованием процедур Property Get и Property Let (Tools Add Procedure ?ввести имя (здесь - varian) выбрать Property выбрать Public OK),

- поля класса - avto, firma, model, stoim, pict, var, см. ниже.

3. создать на форме:

- два элемента управления ComboBox с именами Combo1 и Combo2,

- два элемента управления CommandButton с именами Command1 и Command2; значению

свойства Caption объекта Command1 присвоить значение "OK", Command2 - "Exit",

- элементы управления Label1 - Label4, значениям свойств Caption присвоить: Label1 - "Хотите ли Вы купить машину?", Label2 - "Выберите фирму-изготовитель", Label3 - "Хотите ли Вы увидеть изображение выбранного автомобиля или его технические характеристики ?", Label4- "", свойству Visible объекта Label4 присвоить False,

- массив элементов управления OptionButton (присвоить значения свойствам - Option1(0).Caption= "да", Option1(1).Caption= "нет"),

- массивы элементов управления PictureBox: Picture1(0) - Picture1(12) и Picture2(0) - Picture(12). Свойству Visible всех элементов управления присвоить значение False.

Свойству Picture каждого элемента управления присвоить значение изображения соответствующего автомобиля и списка технических характеристик (эти списки создаются в приложении Excel, далее таблицы передаются в приложение Paint и сохраняются как рисунки).

4. ввести код в область класса (см. ниже "область проекта Class1"),

5. ввести код, данный ниже, в области:

- General Declarations формы,
- Combo1, событие Click,
- Combo2, событие Click,
- Command1, событие Click,
- Command, событие Click,
- Form, событие Load,
- Form, событие Unload,

6. стартовать проект, получить справку о предполагаемой покупке.

```
//////////////////////////////область проекта Class1////////////////////////////////
Public avto As Boolean
Public firma As String
Public model As String
Public stoim As String
Public pict As String
Dim var As String
Private Sub Class_Initialize() ' инициализация полей класса
avto = False: firma = "": model = "": stoim = "": var = ""
End Sub
Public Function Met1()
' Если пользователь нажал кнопку (OptionButton) - Да, то выполнить процедуры
' Met2, Met3, Met4, результатом выполнения которых является вывод данных:
' марка, стоимость, изображение и технические характеристики, иначе
' Met1 = False и выводится сообщение "Приносим свои извинения, мы даем
' информацию для желающих купить автомобиль"
If avto = True Then
model = Met2()
stoim = Met3()
pict = Met4() ' поле pict определяет номера элементов массивов
```

```
' PictureBox, см. Met4
Met1 = True
Else
Met1 = False
End If
End Function
' после щелчка на кнопках Да/Нет (два переключателя OptionButton) и выбора '
фирмы из списка ComboBox с именем Combo1 определить марку автомобиля
Public Function Met2()
Select Case firma
Case "AUDI": Met2 = "A6"
Case "CITROEN": Met2 = "C5"
Case "FORD": Met2 = "Focus"
Case "HONDA": Met2 = "Accord"
Case "HYUNDAI": Met2 = "Elanta"
Case "JEEP": Met2 = "Grand Cherokee LTD"
Case "LAND ROVER": Met2 = "Land Rover Discovery"
Case "LEXSUS": Met2 = "RX330"
Case "MITSUBISHI": Met2 = "Pajero III"
Case "NISSAN": Met2 = "Primera(1.8)"
Case "PEUGEOT": Met2 = "307 XR"
Case "PORSCHE": Met2 = "Cayenne Turbo"
Case "RENAULT": Met2 = "Laguna II"
End Select
End Function
' определить стоимость автомобиля в долларах США
Public Function Met3()
Select Case firma
Case "AUDI": Met3 = "41500"
Case "CITROEN": Met3 = "20100"
Case "FORD": Met3 = "12430"
Case "HONDA": Met3 = "33900"
Case "HYUNDAI": Met3 = "13790"
Case "JEEP": Met3 = "41690"
Case "LAND ROVER": Met3 = "40850"
Case "LEXSUS": Met3 = "65500"
Case "MITSUBISHI": Met3 = "56640"
Case "NISSAN": Met3 = "25100"
Case "PEUGEOT": Met3 = "13808"
Case "PORSCHE": Met3 = "140500"
Case "RENAULT": Met3 = "22900"
End Select
End Function
Public Function Met4()
' при выборе данных из списка ComboBox с именем Combo2
' (после щелчка на кнопке "OK" ) определяется номер элемента массива
' PictureBox, соответствующий выбранной фирме-изготовителю и
' на экран позднее выводится соответствующая фотография
' и/или технические характеристики автомобиля
Select Case firma
Case "AUDI": Met4 = "0"
Case "CITROEN": Met4 = "1"
Case "FORD": Met4 = "2"
```

```
Case "HONDA": Met4 = "3"
Case "HYUNDAI": Met4 = "4"
Case "JEEP": Met4 = "5"
Case "LAND ROVER": Met4 = "6"
Case "LEXSUS": Met4 = "7"
Case "MITSUBISHI": Met4 = "8"
Case "NISSAN": Met4 = "9"
Case "PEUGEOT": Met4 = "10"
Case "PORSCHE": Met4 = "11"
Case "RENAULT": Met4 = "12"
End Select
End Function
' процедура Property Let используется для задания значения свойства,
' Property Get - для считывания значения свойства
Public Property Get varian() As String
Select Case var
Case Is = 0: varian = "pict"
Case Is = 1: varian = "texn"
Case Is = 2: varian = "all"
End Select
End Property
Public Property Let varian(ByVal vNewValue As String)
Select Case vNewValue
Case "изображение": var = 0
Case "технические параметры": var = 1
Case Else: var = 2
End Select
End Property
////////////////////////////////область проекта Form1//////////////////////////
Dim av As Class1 ' av - экземпляр класса
Dim v As String
Dim i As Integer, j As Integer
Private Sub Combo1_Click()
' сделать невидимыми элементы управления Label и Picture
' (формирующие фотографии, технические характеристики, фирму,
' марку и стоимость), для того, чтобы впоследствии на форму
' выводились только те из них, которые определяет своими
' действиями покупатель
Label5.Visible = False
For i = 0 To 12
Picture1(i).Visible = False
Picture2(i).Visible = False
Next
Dim ot As String ' переменная для хранения сообщения программы
av.firma = Combo1.Text ' значение поля firma объекта av взять из
' списка ComboBox с именем Combo1
av.avto = Option1(0).Value ' значение поля avto объекта av взять
' из поля массива OptionButton
If av.Met1 = True Then
ot = " " & CStr(av.firma) & vbCrLf: ot = ot & " " & vbCrLf
ot = ot & " модель " & CStr(av.model) & vbCrLf: ot = ot & " " & vbCrLf
ot = ot & " цена в $ " & CStr(av.stoim) & vbCrLf: ot = ot & " " & vbCrLf
ot = ot & "Для получения более полной информации обращайтесь_
```

```
по телефону 7077888"
MsgBox Title:="Мы можем предложить", Prompt:=ot
Else
Label5.Visible = False
Picture1(Val(av.pict)).Visible = False ' аргумент Picture1: (av.pict)
' определяет индекс элемента массива PictureBox
ot = "Приносим свои извинения, мы даем информацию для желающих_
купить автомобиль"
MsgBox Title:="Автосалон START", Prompt:=ot
End If
End Sub
Private Sub Combo2_Click()
av.varian = Combo2.Text ' см. процедуру Property Let. Присваиваем
' свойству varian значение выбранные из списка ComboBox с именем Combo2
End Sub
Private Sub Command1_Click()
Label5.Visible = False
Label5.Caption = ""
For i = 0 To 12
Picture1(i).Visible = False
Picture2(i).Visible = False
Next
v = av.varian ' см. процедуру Property Get. Переменной v присваиваем
' значение свойства varian объекта av
av.avto = Option1(0).Value
If av.Met1 = True Then
Select Case v
Case "pict"
Picture1(Val(av.pict)).Visible = True
Case "texn"
Picture2(Val(av.pict)).Visible = True ' технические характеристики
' хранятся как изображения в соответствующих элементах
' массива PictureBox2
Case "all"
Picture1(Val(av.pict)).Visible = True
Picture2(Val(av.pict)).Visible = True
Label5.Visible = True
Label5.Caption = CStr(av.firma) & " " & CStr(av.model) & _
vbCrLf & "цена в $ " & CStr(av.stoim)
End Select
Else
Picture1(Val(av.pict)).Visible = False
Picture2(Val(av.pict)).Visible = False
MsgBox Title:="Автосалон START", Prompt:="Приносим свои_
извинения, мы даем информацию для желающих купить автомобиль"
End If
End Sub
Private Sub Command2_Click()
MsgBox Title:="Автосалон START", Prompt: = "Мы всегда рады помочь!_
Будем рады новой встрече!"
End ' выход из программы после сообщения MsgBox.
End Sub
Private Sub Form_Load()
```

```
Set av = New Class1 ' описание объектной переменной дано выше
' заполнение списка ComboBox с именем Combo1 названиями фирм
Combo1.AddItem "AUDI"
Combo1.AddItem "CITROEN"
Combo1.AddItem "FORD"
Combo1.AddItem "HONDA"
Combo1.AddItem "HYUNDAI"
Combo1.AddItem "JEEP"
Combo1.AddItem "LAND ROVER"
Combo1.AddItem "LEXSUS"
Combo1.AddItem "MITSUBISHI"
Combo1.AddItem "NISSAN"
Combo1.AddItem "PEUGEOT"
Combo1.AddItem "PORSCHE"
' заполнение списка ComboBox с именем Combo2 предложениями для
' выбора данных в процедурах Property Get и Property Let
Combo2.AddItem "изображение"
Combo2.AddItem "технические параметры"
Combo2.AddItem "все данные"
End Sub
Private Sub Form_Unload(Cancel As Integer)
Set av = Nothing ' удалить объект из памяти
End Sub
```

Инструкция пользователя

После старта проекта при отрицательном ответе на вопрос «Хотите ли Вы купить машину?» выводится сообщение: «Приносим свои извинения, мы даем информацию для желающих купить автомобиль».

При положительном ответе на вопрос и выборе фирмы-изготовителя из списка на экран выводится сообщение о фирме, марке и стоимости автомобиля.

Далее покупатель может просмотреть или внешний вид, или технические характеристики, или одновременно обе категории, выбрав соответствующую строку во втором списке и сделав щелчок на кнопке ОК (используются процедуры Property Get и Property Let), где дан результат работы программы при выборе строки «все данные».

При щелчке на кнопке Exit выводится сообщение: «Мы всегда рады помочь! Будем рады новой встрече!» и проводится выход из программы.

Заключение (выводы)

Созданный программный продукт позволяет клиенту получить справочные данные при покупке автомобиля. Представленная программа является лишь небольшим примером использования классов, в реальности же сфера применения свойств объектно-базирующегося программирования гораздо шире.

Задания для выполнения:

1. Для предметной области (выбранной в практической работе №1) выполнить объектноориентированное проектирование программного продукта

Критерии оценки:

Оценка «отлично» – практические задачи выполнены в полном объеме, студент отвечает на все поставленные вопросы. Все задания выполнены правильно. Работа оформлена аккуратно и в соответствии с требованиями. Продемонстрировано глубокое понимание материала.

Оценка «хорошо» – студент допускает незначительные неточности, правильно применены теоретические знания. Задания выполнены с небольшими ошибками. Оформление

работы имеет несущественные недочеты. Продемонстрировано хорошее понимание материала

Оценка «удовлетворительно» – отсутствие полного объема работ; низкое качество выполнения работ, часть заданий выполнена с ошибками; Оформление работы имеет существенные недочеты; продемонстрировано базовое понимание материала

Оценка «неудовлетворительно» – отсутствие полного объема работ; в работе допущены серьёзные ошибки и нарушение всех перечисленных выше требований; отсутствует понимание материала.

Тема 2.1.2. Описание и анализ требований. Диаграммы IDEF

Задание 2.1.2.1. Практическая работа № 5 «Построение организационных диаграмм с помощью MS Visio»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.2., ПК2.3., ПК2.4., ПК2.5.*

Цель: изучение основ создания организационных диаграмм в ручном и автоматическом режимах в MS OFFICE VISIO.

Основные понятия

Организационная диаграмма – это схематическое представление об иерархии внутри компании, а также распределении полномочий и ответственности между сотрудниками и отделами.

Использование организационных диаграмм позволяет решить сразу несколько задач:

1.проанализировать состояние дел в компании на текущий момент,

2.вовремя обнаружить какую-то проблему в организации и системе управления,

3.избежать появления новых проблем и ошибок.

Говоря об организационной структуре, имеют в виду концептуальную схему, вокруг которой организуется группа людей, основу, на которой держатся все функции. Организационная структура предприятия - это, по сути, руководство для пользования, которое объясняет, как организация выстроена и как она работает. Если говорить конкретнее, то организационная структура описывает, как в компании принимаются решения и кто является ее лидером.

Элементы организационной структуры

Организационная структура любой организации будет зависеть от того, кто является ее участниками, какие задачи она решает и как далеко организация зашла в своем развитии.

Независимо от того, какую организационную структуру вы выбираете, три элемента всегда будут присутствовать в ней.

Управление - конкретный человек или группа людей, которые принимают решения в организации.

Правила, по которым работает организация. Многие из этих правил могут быть заявлены явно, в то время как другие могут быть скрытыми, но при этом не менее обязательными для исполнения.

Распределение труда. Распределение труда может быть формальным или неформальным, временным или постоянным, но в каждой организации непременно будет определенный тип распределения труда.

Организационные диаграммы описывают связи при помощи трех типов графических элементов:

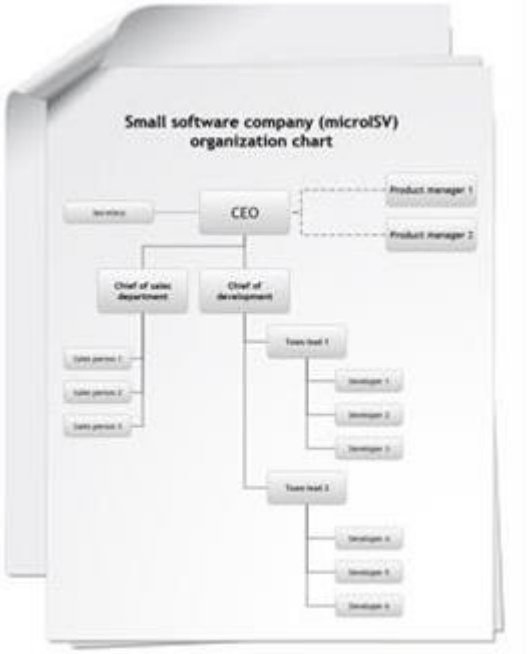
1. Линия: указывает на прямые отношения между руководителями и подчиненными. Для описания взаимосвязей между различными иерархическими уровнями организации линии рисуют слева или справа от диаграммы.

2. Ступенька: служит для иллюстрации отношений между помощниками менеджера, а также связей между сферами деятельности, в которых помощники могут давать советы руководителю, но при этом их мнение не является авторитетным.

3. Функционал: показывает связи между должностями специалистов и сферами деятельности, в которых мнение специалистов является авторитетным для руководителя.

Типы организационных структур

Линейная модель: каждый руководитель обеспечивает руководство нижестоящими подразделениями по всем видам деятельности. Решения спускаются сверху вниз. Этот вид структуры подходит для маленьких организаций вроде небольших бухгалтерских фирм и адвокатских контор. Линейная структура позволяет легко принимать решения.



Типы организационных структур

1. линейная модель: каждый руководитель обеспечивает руководство нижестоящими подразделениями по всем видам деятельности;
2. функциональная модель: «одно подразделение = одна функция»;
3. линейно-функциональная модель: ступенчатая иерархическая;
4. процессная модель: «одно подразделение = один процесс»;
5. матричная модель: «один процесс или один проект = группа сотрудников из разных функциональных подразделений»;
6. множественная (смешанная).

В линейной структуре управления каждый руководитель обеспечивает руководство нижестоящими подразделениями по всем видам деятельности. Достоинство – простота, экономичность, предельное единоначалие. Основной недостаток – высокие требования к квалификации руководителей. Сейчас практически не используется.

Функциональная организационная структура – связь административного управления с осуществлением функционального управления.

Линейно-функциональная структура – линейные руководители являются единоначальниками, а им оказывают помощь функциональные органы. Линейные руководители низших ступеней административно не подчинены функциональным руководителям высших ступеней управления. Она применялась наиболее широко.

Преимущества:

– четкая система взаимных связей внутри функций и в соответствующих им подразделениях;

– четкая система единоначалия – один руководитель сосредотачивает в своих руках руководство всей совокупностью функций, составляющих деятельность;

– ясно выраженная ответственность;

– быстрая реакция исполнительных функциональных подразделений на прямые указания вышестоящих.

Недостатки:

– в работе руководителей практически всех уровней оперативные проблемы («текучка») доминируют над стратегическими;

– слабые горизонтальные связи между функциональными подразделениями порождают волокиту и перекладывание ответственности при решении проблем, требующих участия нескольких подразделений;

– малая гибкость и приспособляемость к изменению ситуации;

– критерии эффективности и качества работы подразделений и организации в целом разные и часто взаимоисключающие;

– большое число «этажей» или уровней управления между работниками, выпускающими продукцию, и лицом, принимающим решение;

– перегрузка управленцев верхнего уровня;

– повышенная зависимость результатов работы организации от квалификации, личных и деловых качеств высших управленцев.

Процессная модель. Истоки концепции управления процессами ведут к теориям управления, разработанным еще в XIX веке. В 80-х годах XIX-го века Фредерик Тейлор предложил менеджерам использовать методы процессного управления для наилучшей организации деятельности. В начале 1900-х годов Анри Файоль разработал концепцию реинжиниринга – осуществление деятельности в соответствии с поставленными задачами путем получения оптимального преимущества из всех доступных ресурсов.

Преимущества процессных структур:

– четкая система взаимных связей внутри процессов и в соответствующих им подразделениях;

– четкая система единоначалия – один руководитель сосредотачивает в своих руках руководство всей совокупностью операций и действий, направленных на достижение поставленной цели и получение заданного результата;

– наделение сотрудников большими полномочиями и увеличение роли каждого из них в работе компании приводит к значительному повышению их отдачи;

– быстрая реакция исполнительных процессных подразделений на изменение внешних условий;

– в работе руководителей стратегические проблемы доминируют над оперативными;

– критерии эффективности и качества работы подразделений и организации в целом согласованы и сонаправлены.

Недостатки процессной структуры:

– повышенная зависимость результатов работы организации от квалификации, личных и деловых качеств рядовых работников и исполнителей.

– управление смешанными в функциональном смысле рабочими командами – более сложная задача, нежели управление функциональными подразделениями;

– наличие в команде нескольких человек различной функциональной квалификации неизбежно приводит к некоторым задержкам и ошибкам, возникающим при передаче работы между членами команды. Однако потери здесь значительно меньше, чем при традиционной организации работ, когда исполнители подчиняются различным подразделениям компании.

Матричная модель. Матричные структуры совмещают принципы построения функциональных и процессных систем. В этих структурах существуют жестко регламентированные процессы, находящиеся под управлением менеджера процесса. При этом деятельность осуществляется работниками, находящимися в оперативном подчинении менеджера процесса и в административном подчинении руководителя в функциональном «колодце».

Преимущества

– Комплексный подход к реализации проекта, решению проблемы;

– Концентрация усилий на решении одной задачи, на выполнении одного конкретного проекта;

– Большая гибкость структуры;

– Активизация деятельности руководителей проектов и исполнителей в результате

формирования проектных групп;

– Усиление личной ответственности конкретного руководителя как за проект в целом, так и за его элементы.

Недостатки

– При наличии нескольких организационных проектов или программ проектные структуры приводят к дроблению ресурсов и заметно усложняют поддержание и развитие производственного и научно-технического потенциала компании как единого целого;

– От руководителя проекта требуется не только управление всеми стадиями жизненного цикла проекта, но и учет места проекта в сети проектов данной компании;

– Формирование проектных групп, не являющихся устойчивыми образованиями, лишает работников осознания своего места в компании;

– При использовании проектной структуры возникают трудности с перспективным использованием специалистов в данной компании;

– Наблюдается частичное дублирование функций.

Смешанные структуры. Если применять различные модели организации деятельности в пределах отдельных бизнес-процессов, то можно использовать преимущества той или иной организационной модели. При этом для организации в целом будет применяться процессная организация основных структурных блоков, а в рамках отдельных блоков могут применяться различные модели.

4. Методика выполнения лабораторной работы

В качестве примера рассматривается процесс создания организационной диаграммы ВУЗа двумя способами:

1.вручную;

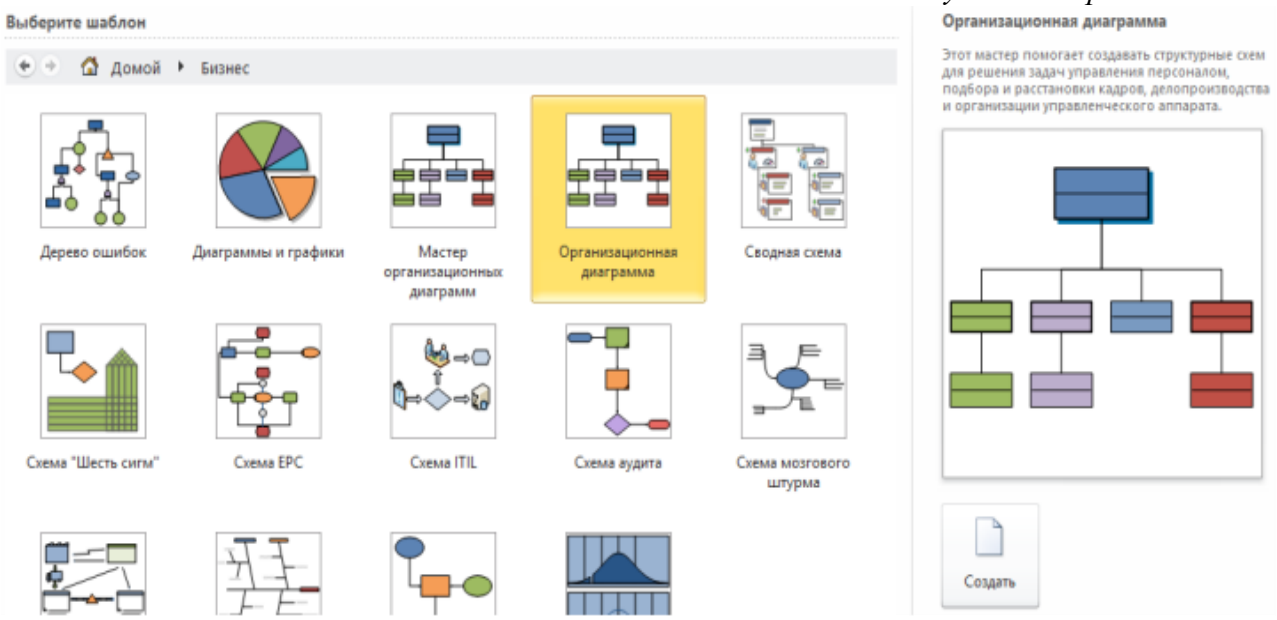
2.с помощью мастера.

Создание организационной диаграммы вручную

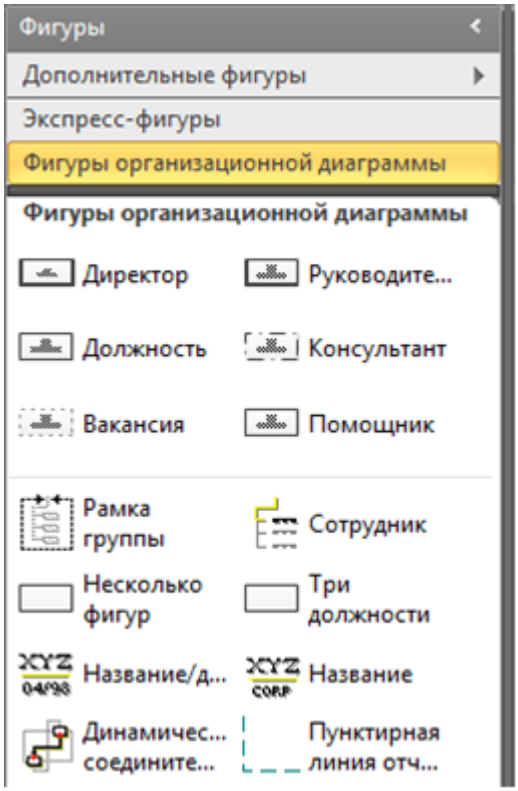
Для построения организационной диаграммы вручную необходимо проделать следующие действия:

1. Запустить MS Visio.
2. В разделе Выберите шаблон выбрать категорию Бизнес, а затем Организационная диаграмма и нажать кнопку Создать.

Рисунок Выбор шаблона



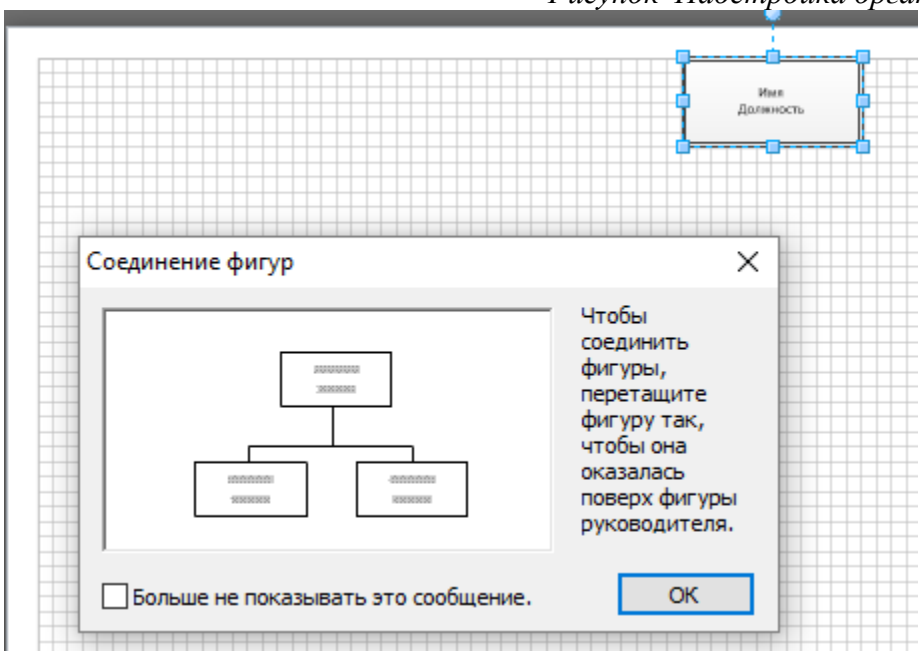
В разделе фигур можно увидеть основные элементы, необходимые для построения организационной диаграммы .



3. Перетащите фигуру *Директор* из набора фигур *Фигуры организационной диаграммы* в центр вверху страницы документа.

4. Надстройка организационных диаграмм отображает анимированное диалоговое окно , показывающее, как нужно добавлять в схему дополнительные фигуры.

Рисунок Надстройка организационных диаграмм



5. Не снимая выделения с фигуры, при необходимости введите *ФИО*, затем нажмите клавишу ENTER и во второй строке введите *Должность*.

При создании организационной диаграммы ВУЗа можно ограничиться только указанием должностей.

Рисунок Фигура Директор

5. Добавьте еще две фигуры *Директор* и расположите по обе стороны от фигуры *Ректор*. Соедините их используя *Соединительную линию.* В результате должна получиться схема, показанная на рисунке 6.

7. Перетащите фигуру *Руководитель* на фигуру *Директор*. Затем, при необходимости, введите ФИО, нажмите клавишу ENTER и введите *Должность*.

Надстройка автоматически размещает новую фигуру под фигурой *Директор*.

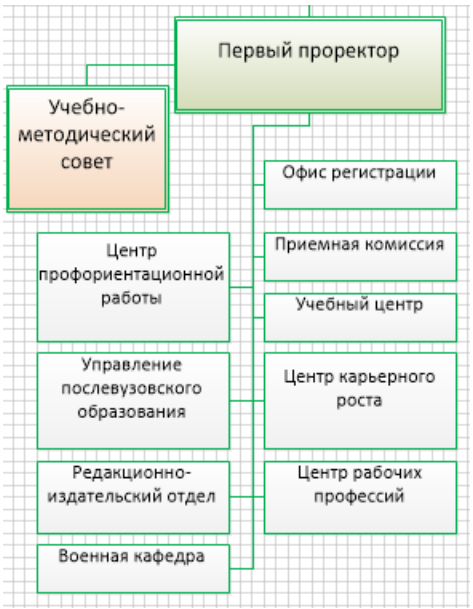
8. Повторите предыдущий шаг и обратите внимание, что надстройка разместила вторую фигуру руководителя

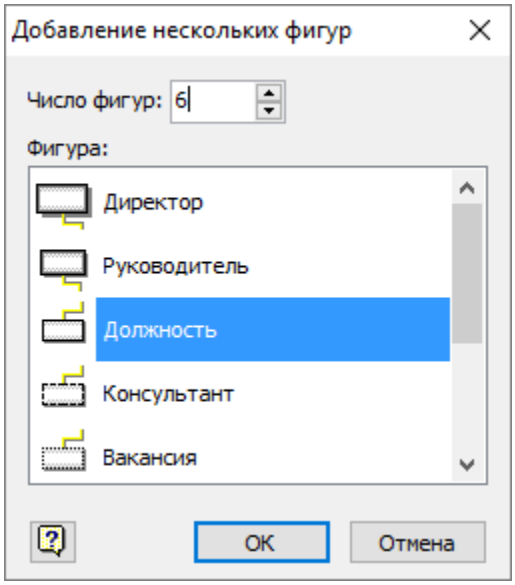
сбоку от первой. Разместите все необходимые фигуры *Руководитель* в соответствии с рисунком 7.

9. Используя фигуры *Руководитель* и *Несколько фигур* добавьте необходимые *Должности,* относящиеся к руководителю **Первый проректор.**

При использовании элемента *Несколько фигур* необходимо выполнить следующие действия: – перетащить элемент Несколько фигур на необходимую фигуру;

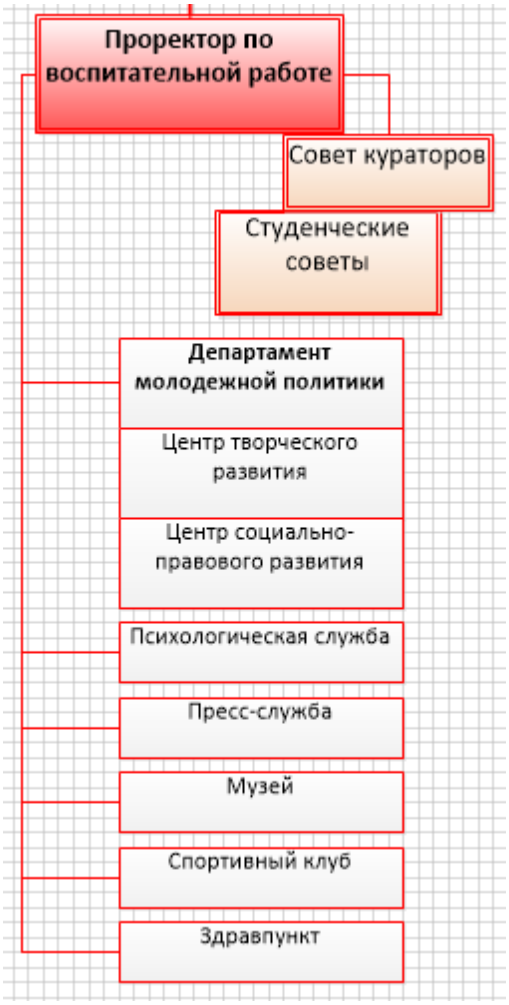
– затем, в появившемся окне задать необходимые параметры, например, как на рисунке 9; – нажать ОК.



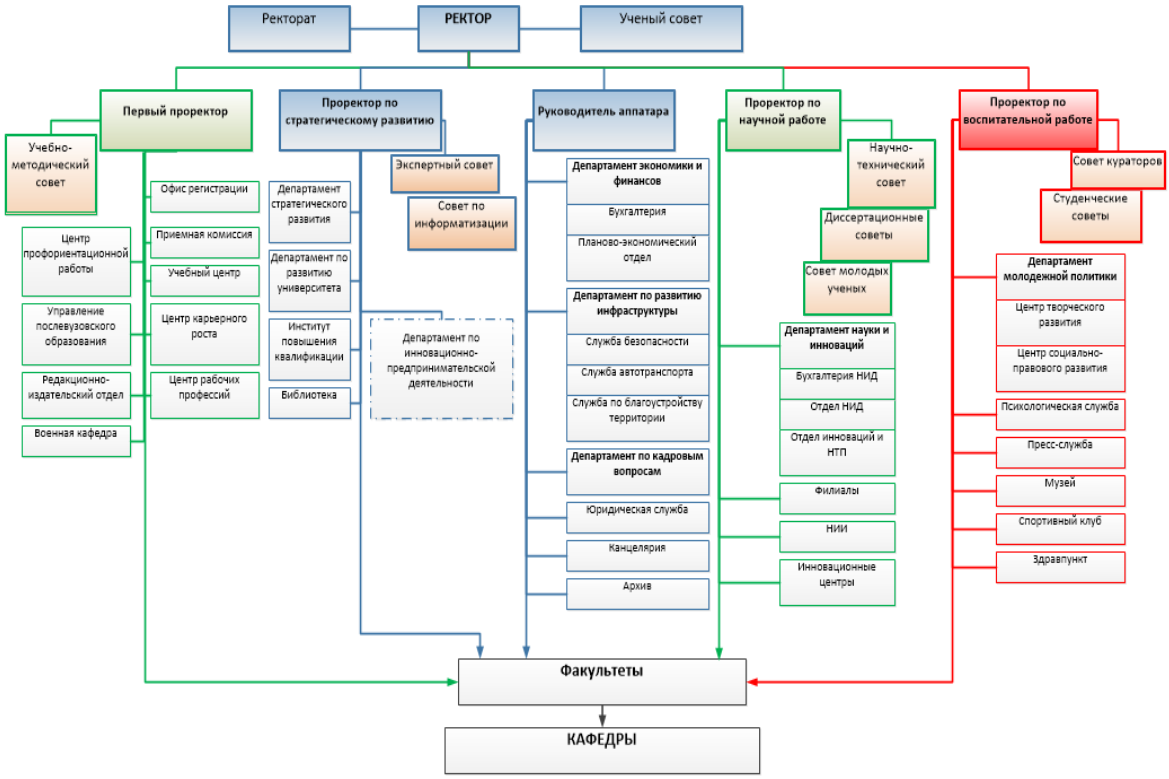


Повторите предыдущий шаг и создайте полную организационную диаграмму


Руководитель аппатара
Департамент экономики и финансов
Бухгалтерия
Планово-экономический отдел
Департамент по развитию инфраструктуры
Служба безопасности
Служба автотранспорта
Служба по благоустройству территории
Департамент по кадровым вопросам
Юридическая служба
Канцелярия
Архив
Проректор по научной работе
Научно-технический совет
Диссертационные советы
Совет молодых ученых
Департамент науки и инноваций
Бухгалтерия НИД
Отдел НИД
Отдел инноваций и НТП
Филиалы
НИИ
Инновационные центры



11. В результате выполнения всех описанных шагов организационная диаграмма ВУЗа должна принять вид

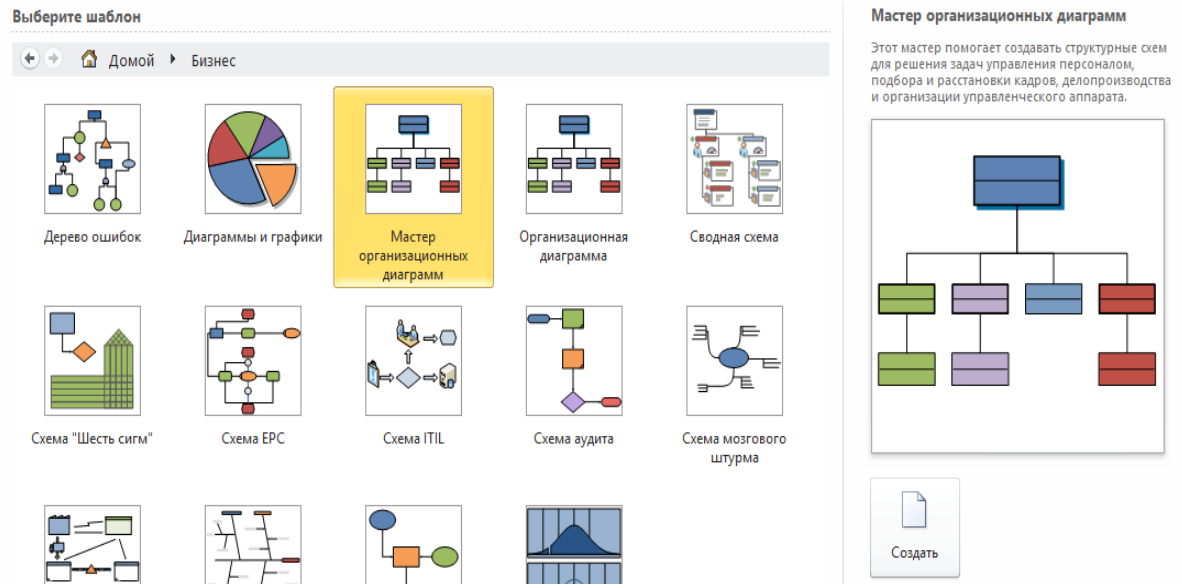


4.2 Создание организационной диаграммы с помощью мастера диаграмм на основе данных

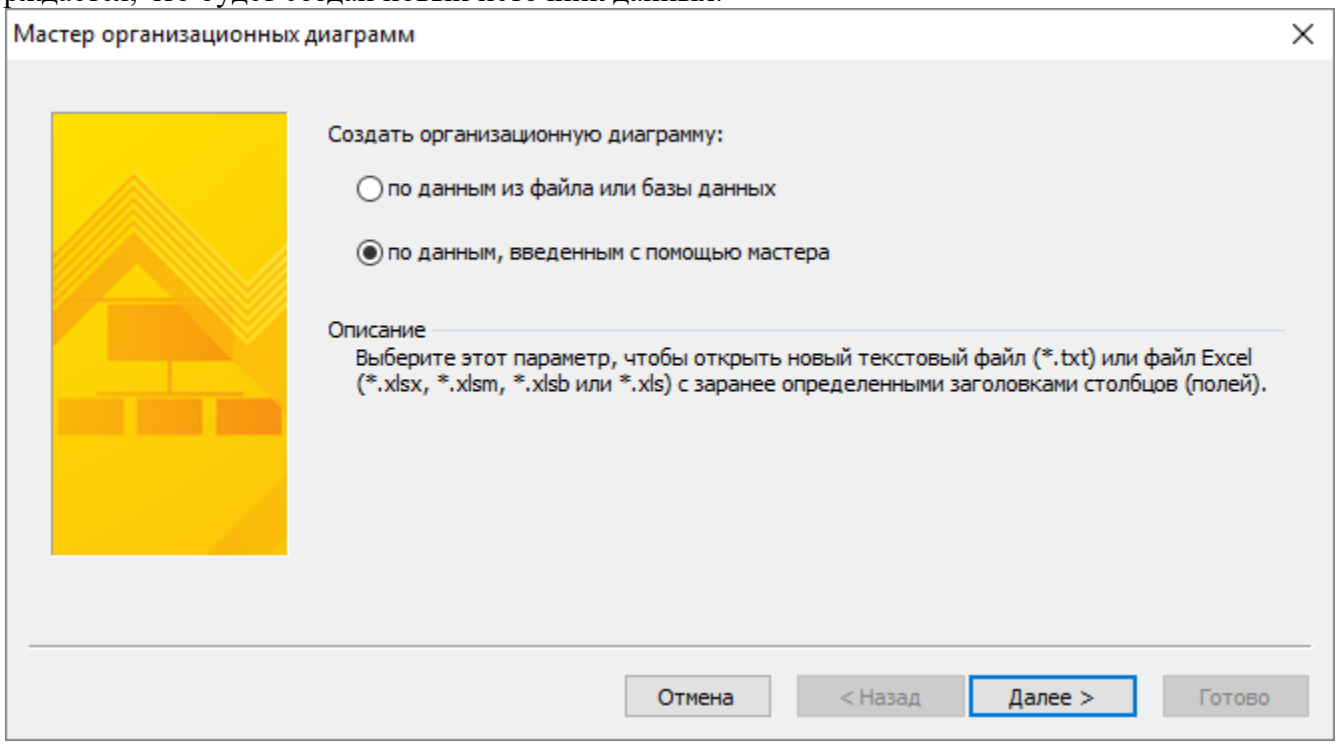
Построение организационной диаграммы с помощью мастера рассмотрим также на примере ВУЗа, однако немного упростим структуру.

Для построения организационной диаграммы с помощью мастера диаграмм необходимо проделать следующие действия:

1. Запустить *MS Visio*.
2. В разделе *Выберите шаблон* выбрать категорию *Бизнес*, а затем *Мастер организационных диаграмм* и нажать кнопку *Создать*.

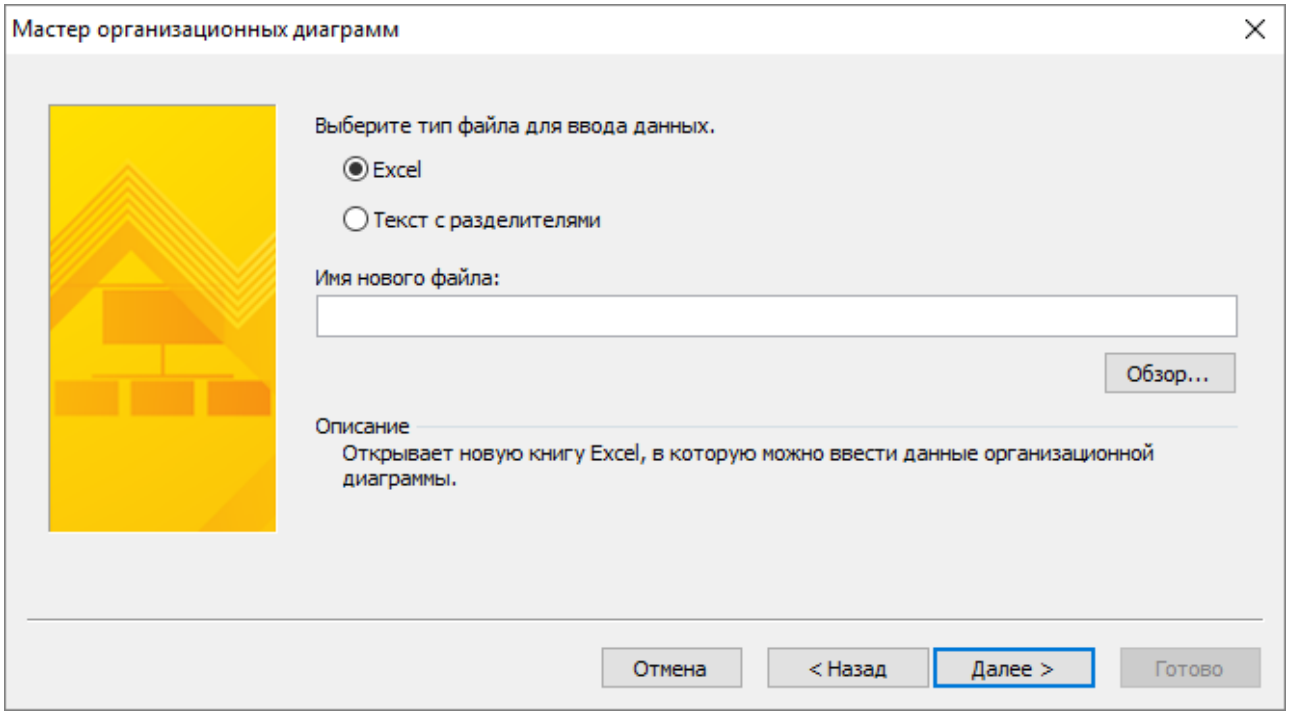


3. На первой странице мастера организационных диаграмм выберем переключатель *По данным, введенным с помощью мастера*. Обратите внимание – в описании этого переключателя подтверждается, что будет создан новый источник данных.

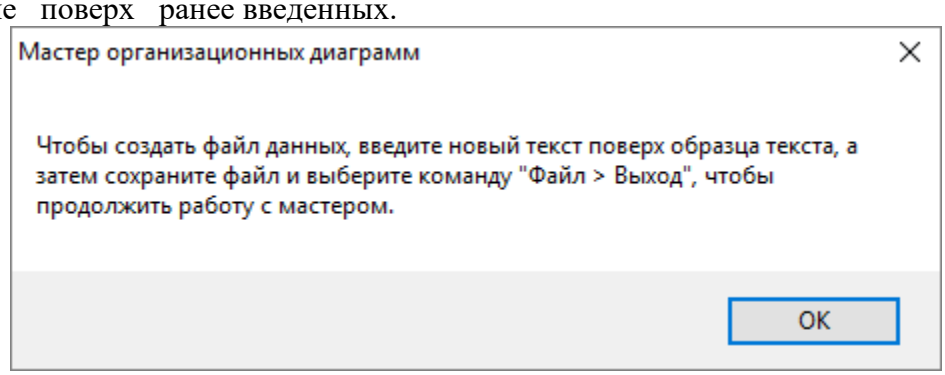


4. Далее необходимо нажать кнопку *Далее* и выбрать тип файла.

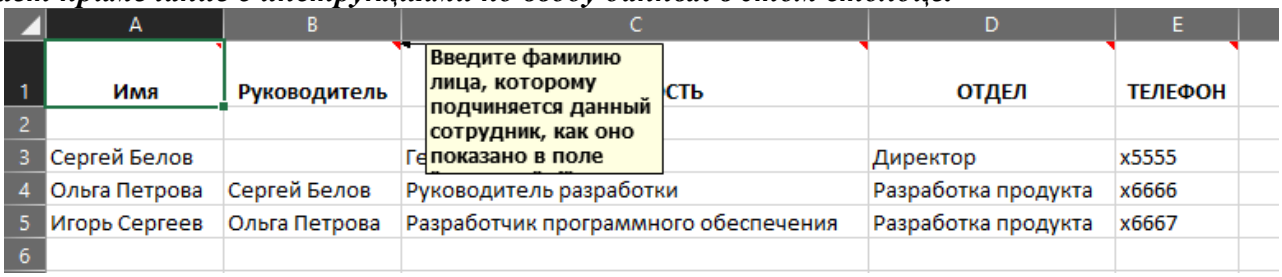
На странице выбора типа файла выберем переключатель *Excel* и щелкнем на кнопке *Обзор*. В открывшемся диалоговом окне выберем папку, в которую нужно сохранить файл, в поле *Имя файла* введем *Данные оргдиаграммы* и щелкнем на кнопке Сохранить. Имя файла отображается в поле *Имя нового файла*.



5. Щелкнем на кнопке *Далее*. MS Visio показывает на необходимость ввести свои данные поверх ранее введенных.



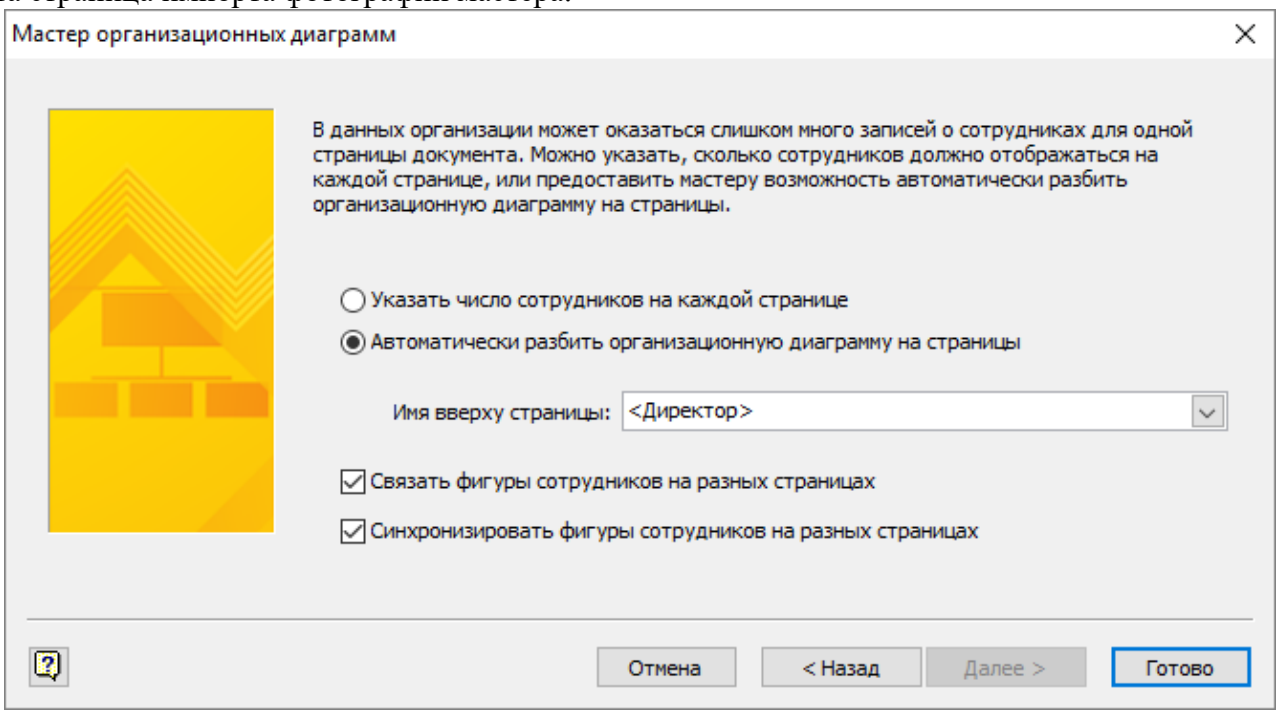
6. Щелкнем на кнопке *Ок*. В Excel отображается форматированная книга. ***На рисунке 19 показана книга Excel для ввода данных диаграммы, при этом заголовок каждого столбца включает примечание с инструкциями по вводу данных в этом столбце.***

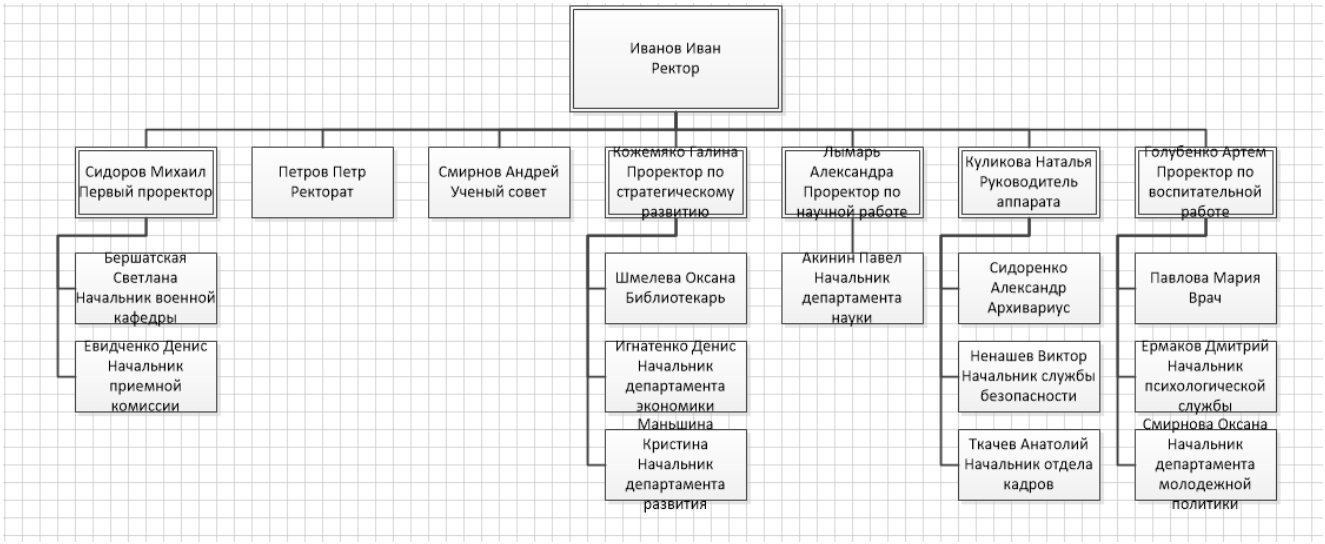


Далее вносим свои данные

	A	B	C	D	E
1	**Имя**	**Руководитель**	**ДОЛЖНОСТЬ**	**ОТДЕЛ**	**ТЕЛЕФОН**
2					
3	Иванов Иван		Ректор	Ректорат	
4	Петров Петр	Иванов Иван	Ректорат	Ректорат	
5	Смирнов Андрей	Иванов Иван	Ученый совет	Ректорат	
6	Сидоров Михаил	Иванов Иван	Первый проректор	Ректорат	
7	Кожемяко Галина	Иванов Иван	Проректор по стратегическому развитию	Ректорат	
8	Куликова Наталья	Иванов Иван	Руководитель аппарата	Ректорат	
9	Лымарь Александра	Иванов Иван	Проректор по научной работе	Ректорат	
10	Голубенко Артем	Иванов Иван	Проректор по воспитательной работе	Ректорат	
11	Евидченко Денис	Сидоров Михаил	Начальник приемной комиссии	Приемная комиссия	
12	Бершатская Светлана	Сидоров Михаил	Начальник военной кафедры	Военная кафедра	
13	Маньшина Кристина	Кожемяко Галина	Начальник департамента развития	Департамент стратегического развития	
14	Шмелева Оксана	Кожемяко Галина	Библиотекарь	Библиотека	
15	Игнатенко Денис	Кожемяко Галина	Начальник департамента экономики	Департамент экономики и финансов	
16	Ненашев Виктор	Куликова Наталья	Начальник службы безопасности	Департамент по развитию инфраструктуры	
17	Ткачев Анатолий	Куликова Наталья	Начальник отдела кадров	Департамент по кадровым вопросам	
18	Сидоренко Александр	Куликова Наталья	Архивариус	Архив	
19	Акинин Павел	Лымарь Александра	Начальник департамента науки	Департамент науки и инноваций	
20	Смирнова Оксана	Голубенко Артем	Начальник департамента молодежной политики	Департамент молодежной политики	
21	Ермаков Дмитрий	Голубенко Артем	Начальник психологической службы	Психологическая служба	
22	Павлова Мария	Голубенко Артем	Врач	Здравпункт	

8. Закрываем Excel. После закрытия Excel происходит возврат в программу MS Visio, где открыта страница импорта фотографий мастера.





10. На последнем шаге необходимо произвести форматирование диаграммы. В результате организационная диаграмма должна принять вид

Задание

Построить организационную диаграмму в соответствии в вариантом.

Варианты

1. «Отдел кадров»;
2. «Агентство аренды»;
3. «Аптека»;
4. «Ателье»;
5. «Аэропорт»;
6. «Библиотека»;
7. «Кинотеатр»;
8. «Поликлиника»;
9. «Автосалон»;
10. «Таксопарк».

Контрольные вопросы

1. Что такое организационная диаграмма?
2. Способы построения оргдиаграмм в MS Visio?
3. Каковы принципы создания организационных диаграмм в MS Visio?
4. Какие существуют типы организационных структур? Перечислите их преимущества и

недостатки.

Задание 2.1.2.2. Практическая работа № 6 «Построение диаграмм прецедентов на языке UML с помощью MS Visio»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.2., ПК2.3., ПК2.4., ПК2.5.*

Цель: изучение основ создания диаграмм прецедентов (вариантов использования) на языке UML.

Краткие теоретические сведения

Общие сведения о языке UML

Язык UML представляет собой общецелевой язык визуального моделирования, который разработан для спецификации, визуализации, проектирования и документирования компонентов программного обеспечения, бизнес-процессов и других систем.

Язык UML одновременно является простым и мощным средством моделирования, который может быть эффективно использован для построения концептуальных, логических и графических моделей сложных систем самого различного целевого назначения.

В языке UML используется четыре основных вида графических конструкций:

Значки или пиктограммы. Значок представляет собой графическую фигуру фиксированного размера и формы. Примерами значков могут служить окончания связей элементов диаграмм или некоторые другие дополнительные обозначения.

Графические символы на плоскости. Такие двумерные символы изображаются с помощью некоторых геометрических фигур и могут иметь различную высоту и ширину с целью размещения внутри этих фигур других конструкций языка UML.

Наиболее часто внутри таких символов помещаются строки текста, которые уточняют семантику или фиксируют отдельные свойства соответствующих элементов языка UML. Информация, содержащаяся внутри фигур, имеет важное значение для конкретной модели проектируемой системы, поскольку регламентирует реализацию соответствующих элементов в программном коде.

Пути, которые представляют собой последовательности из отрезков линий, соединяющих отдельные графические символы. При этом концевые точки отрезков линий должны обязательно соприкасаться с геометрическими фигурами, служащими для обозначения вершин диаграмм, как принято в теории графов. С концептуальной точки зрения путям в языке UML придается особое значение, поскольку они являются простыми топологическими сущностями.

Строки текста. Служат для представления различных видов информации в некоторой грамматической форме. Предполагается, что каждое использование строки текста должно соответствовать синтаксису в нотации языка UML, посредством которого может быть реализован грамматический разбор этой строки.

При графическом изображении диаграмм следует придерживаться следующих основных рекомендаций:

Каждая диаграмма должна служить законченным представлением соответствующего фрагмента моделируемой предметной области.

Все сущности на диаграмме модели должны быть одного концептуального уровня. Здесь имеется в виду согласованность не только имен одинаковых элементов, но и возможность вложения отдельных диаграмм друг в друга для достижения полноты представлений.

Вся информация о сущностях должна быть явно представлена на диаграммах. Речь идет о том, что, хотя в языке UML при отсутствии некоторых символов на диаграмме могут быть использованы их значения по умолчанию (например, в случае неявного указания видимости атрибутов и операций классов), необходимо стремиться к явному указанию свойств всех элементов диаграмм.

Диаграммы не должны содержать противоречивой информации. Противоречивость модели может служить причиной серьезнейших проблем при ее реализации и последующем

использовании на практике. Например, наличие замкнутых путей при изображении отношений агрегирования или композиции приводит к ошибкам в программном коде, который будет реализовывать соответствующие классы. Наличие элементов с одинаковыми именами и различными атрибутами свойств в одном пространстве имен также приводит к неоднозначной интерпретации и может служить источником проблем.

Диаграммы не следует перегружать текстовой информацией. Принято считать, что визуализация модели является наиболее эффективной, если она содержит минимум пояснительного текста.

Каждая диаграмма должна быть самодостаточной для правильной интерпретации всех ее элементов и понимания семантики всех используемых графических символов.

Количество типов диаграмм для конкретной модели приложения не является строго фиксированным.

Разработка диаграммы вариантов использования преследует цели:

- Определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы.
- Сформулировать общие требования к функциональному поведению проектируемой системы.
- Разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей.
- Подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью так называемых вариантов использования.

При этом актером (actor) или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик.

В свою очередь, вариант использования (usecase) служит для описания сервисов, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актером. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие актеров с системой.

В самом общем случае, диаграмма вариантов использования представляет собой граф специального вида,

Внимание!

Вариант использования(usecase)–конструкция или стандартный элемент языка UML, который применяется для спецификации общих особенностей поведения системы или любой другой сущности предметной области без рассмотрения внутренней структуры этой сущности. Каждый вариант использования определяет последовательность действий, которые должны быть выполнены проектируемой системой при взаимодействии ее с соответствующим актером.

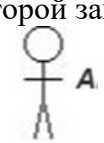
Диаграмма вариантов может дополняться пояснительным текстом, который раскрывает смысл или семантику составляющих ее компонентов.

Такой пояснительный текст получил название примечанияилисценария.

Отдельный вариант использования обозначается на диаграмме эллипсом, внутри которого содержится его краткое название или имя в форме глагола с пояснительными словами.

Актер(actor)представляет собой любую внешнюю по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные

возможности для достижения определенных целей или решения частных задач. При этом актеры служат для обозначения согласованного множества ролей, которые могут играть пользователи в процессе взаимодействия с проектируемой системой. Каждый актер может рассматриваться как некая отдельная роль относительно конкретного варианта использования. Стандартным графическим обозначением актера на диаграммах является фигурка «человечка», под которой записывается конкретное имя актера.



Интерфейс(interface)служит для спецификации параметров модели, которые видимы извне без указания их внутренней структуры. В языке UML интерфейс является классификатором и характеризует только ограниченную часть поведения моделируемой сущности. Применительно к диаграммам вариантов использования, интерфейсы определяют совокупность операций, которые обеспечивают необходимый набор сервисов или функциональности для актеров. Интерфейсы не могут содержать ни атрибутов, ни состояний, ни направленных ассоциаций. Они содержат только операции без указания особенностей их реализации.

На диаграмме вариантов использования интерфейс изображается в виде маленького круга, рядом с которым записывается его имя.

В качестве имени может быть существительное, которое характеризует соответствующую информацию или сервис (например, «датчик», «сирена», «видеокамера»), но чаще строка текста (например, «запрос к базе данных», «форма ввода», «устройство подачи звукового сигнала»).

Если имя записывается на английском, то оно должно начинаться с заглавной буквы I, например, ISecurelnformation, ISensor.

Графический символ отдельного интерфейса может соединяться на диаграмме сплошной линией с тем вариантом использования, который его поддерживает. Сплошная линия в этом случае указывает на тот факт, что связанный с интерфейсом вариант использования должен реализовывать все операции, необходимые для данного интерфейса, а возможно и больше

Кроме этого, интерфейсы могут соединяться с вариантами использования пунктирной линией со стрелкой, означающей, что вариант использования предназначен для спецификации только того сервиса, который необходим для реализации данного интерфейса.

Графически примечания обозначаются прямоугольником с «загнутым» верхним правым уголком (рис. 5). Внутри прямоугольника содержится текст примечания. Примечание может относиться к любому элементу

Примечания(notes)в языке UML предназначены для включения в модель произвольной текстовой информации, имеющей непосредственное отношение к контексту разрабатываемого проекта. В качестве такой информации могут быть комментарии разработчика (например, дата и версия разработки диаграммы или ее отдельных компонентов), ограничения (например, на значения отдельных связей или экземпляры сущностей) и помеченные значения.

Применительно к диаграммам вариантов использования примечание может носить самую общую информацию, относящуюся к общему контексту системы.

Графически примечания обозначаются прямоугольником с «загнутым» верхним правым уголком. Внутри прямоугольника содержится текст примечания. Примечание может относиться к любому элементу

Между компонентами диаграммы вариантов использования могут существовать различные отношения, которые описывают взаимодействие экземпляров одних актеров и вариантов использования с экземплярами других актеров и вариантов.

Один актер может взаимодействовать с несколькими вариантами использования. В этом случае этот актер обращается к нескольким сервисам данной системы. В свою очередь один вариант использования может взаимодействовать с несколькими актерами, предоставляя для всех них свой сервис. Следует заметить, что два варианта использования, определенные для одной и той же сущности, не могут взаимодействовать друг с другом, поскольку каждый из них самостоятельно описывает законченный вариант использования этой сущности. Более того, варианты использования всегда предусматривают некоторые сигналы или сообщения, когда взаимодействуют с актерами за пределами системы. В то же время могут быть определены другие способы для взаимодействия с элементами внутри системы.

В языке UML имеется несколько стандартных видов отношений между актерами и вариантами использования:

Отношение ассоциации (association relationship)

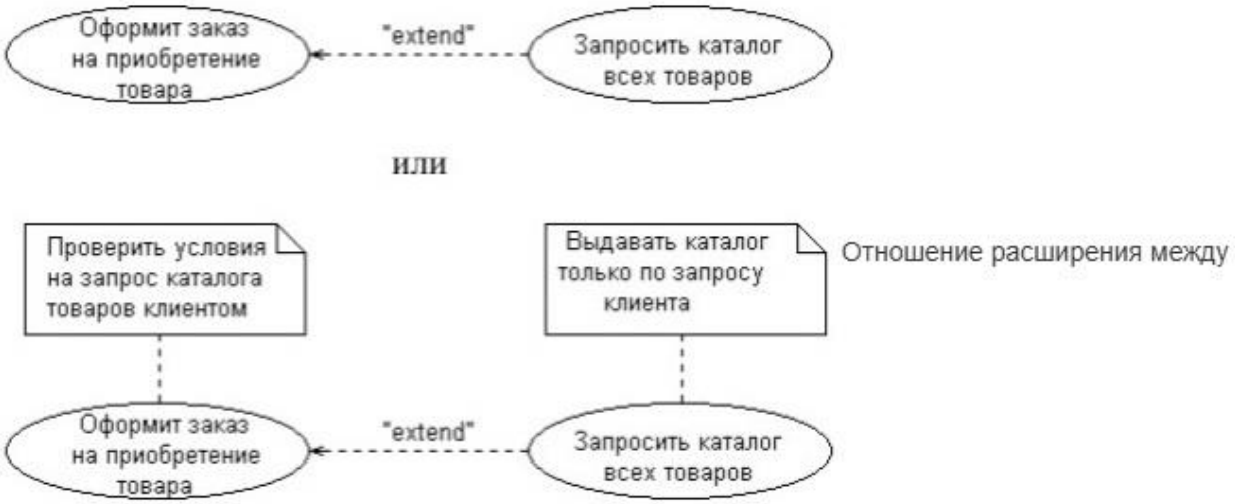
Отношение ассоциации является одним из фундаментальных понятий в языке UML и в той или иной степени используется при построении всех графических моделей систем в форме канонических диаграмм.

Применительно к диаграммам вариантов использования оно служит для обозначения специфической роли актера в отдельном варианте использования. Другими словами, ассоциация специфицирует семантические особенности взаимодействия актеров и вариантов использования в графической модели системы. Таким образом, это отношение устанавливает, какую конкретную роль играет актер при взаимодействии с экземпляром варианта использования. На диаграмме вариантов использования, так же, как и на других диаграммах, отношение ассоциации обозначается сплошной линией между актером и вариантом использования. Эта линия может иметь дополнительные условные обозначения, такие, например, как имя и кратность

Отношение расширения (extend relationship)

Отношение расширения определяет взаимосвязь экземпляров отдельного варианта использования с более общим вариантом, свойства которого определяются на основе способа совместного объединения данных экземпляров.

Так, если имеет место отношение расширения от варианта использования А к варианту использования В, то это означает, что свойства экземпляра варианта использования В могут быть дополнены благодаря наличию свойств у расширенного варианта использования А.



Вариантами использования обозначается пунктирной линией со стрелкой (вариант отношения зависимости), направленной от того варианта использования, который является расширением для исходного варианта использования. Данная линия со стрелкой помечается ключевым словом «extend» («расширяет»)

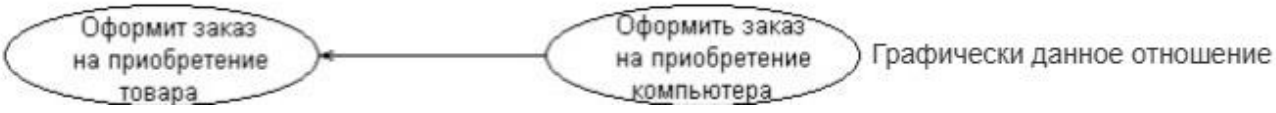
Отношение расширения отмечает тот факт, что один из вариантов использования может присоединять к своему поведению некоторое дополнительное поведение, определенное для другого варианта использования.

Один из вариантов использования может быть расширением для нескольких базовых вариантов, а также иметь в качестве собственных расширений несколько других вариантов. Базовый вариант использования может дополнительно никак не зависеть от своих расширений.

Отношение обобщения (generalization relationship)

Отношение обобщения служит для указания того факта, что некоторый вариант использования А может быть обобщен до варианта использования В.

В этом случае вариант А будет являться специализацией варианта В. При этом В называется предком или родителем по отношению А, а вариант А – потомком по отношению к варианту использования В. Следует подчеркнуть, что потомок наследует все свойства и поведение своего родителя, а также может быть дополнен новыми свойствами и особенностями поведения.



Обозначается сплошной линией со стрелкой в форме незакрашенного треугольника, которая указывает на родительский вариант использования. Эта линия со стрелкой имеет

специальное название – стрелка «обобщение».

Отношение обобщения между вариантами использования применяется в том случае, когда необходимо отметить, что дочерние варианты использования обладают всеми атрибутами и особенностями поведения родительских вариантов. При этом дочерние варианты использования участвуют во всех отношениях родительских вариантов. В свою очередь, дочерние варианты могут наделяться новыми свойствами поведения, которые отсутствуют у родительских вариантов использования, а также уточнять или модифицировать наследуемые от них свойства поведения.

Между отдельными актерами также может существовать отношение обобщения. Данное отношение является направленным и указывает на факт специализации одних актеров относительно других. Например, отношение обобщения от актера А к актеру В отмечает тот факт, что каждый экземпляр актера А является одновременно экземпляром актера В и обладает всеми его свойствами. В этом случае актер В является родителем по отношению к актеру А, а актер А, соответственно, потомком актера В. При этом актер А обладает способностью играть такое же множество ролей, что и актер В.

Графически данное отношение также обозначается стрелкой обобщения, т. е. сплошной линией со стрелкой в форме незакрашенного треугольника, которая указывает на родительского актера.

Отношение включения (include relationship)

Отношение включения между двумя вариантами использования указывает, что некоторое заданное поведение для одного варианта использования включается в качестве составного компонента последовательность поведения другого варианта использования. Данное отношение является направленным бинарным отношением в том смысле, что пара экземпляров вариантов использования всегда упорядочена в отношении включения.

Отношение включения, направленное от варианта использования А к варианту использования В, указывает, что каждый экземпляр варианта А включает в себя функциональные свойства, заданные для варианта В. Эти свойства специализируют поведение соответствующего варианта А на данной диаграмме. Графически данное отношение обозначается пунктирной линией со стрелкой (вариант отношения зависимости), направленной от базового варианта использования к включаемому. При этом данная линия со стрелкой помечается ключевым словом «include» («включает»)

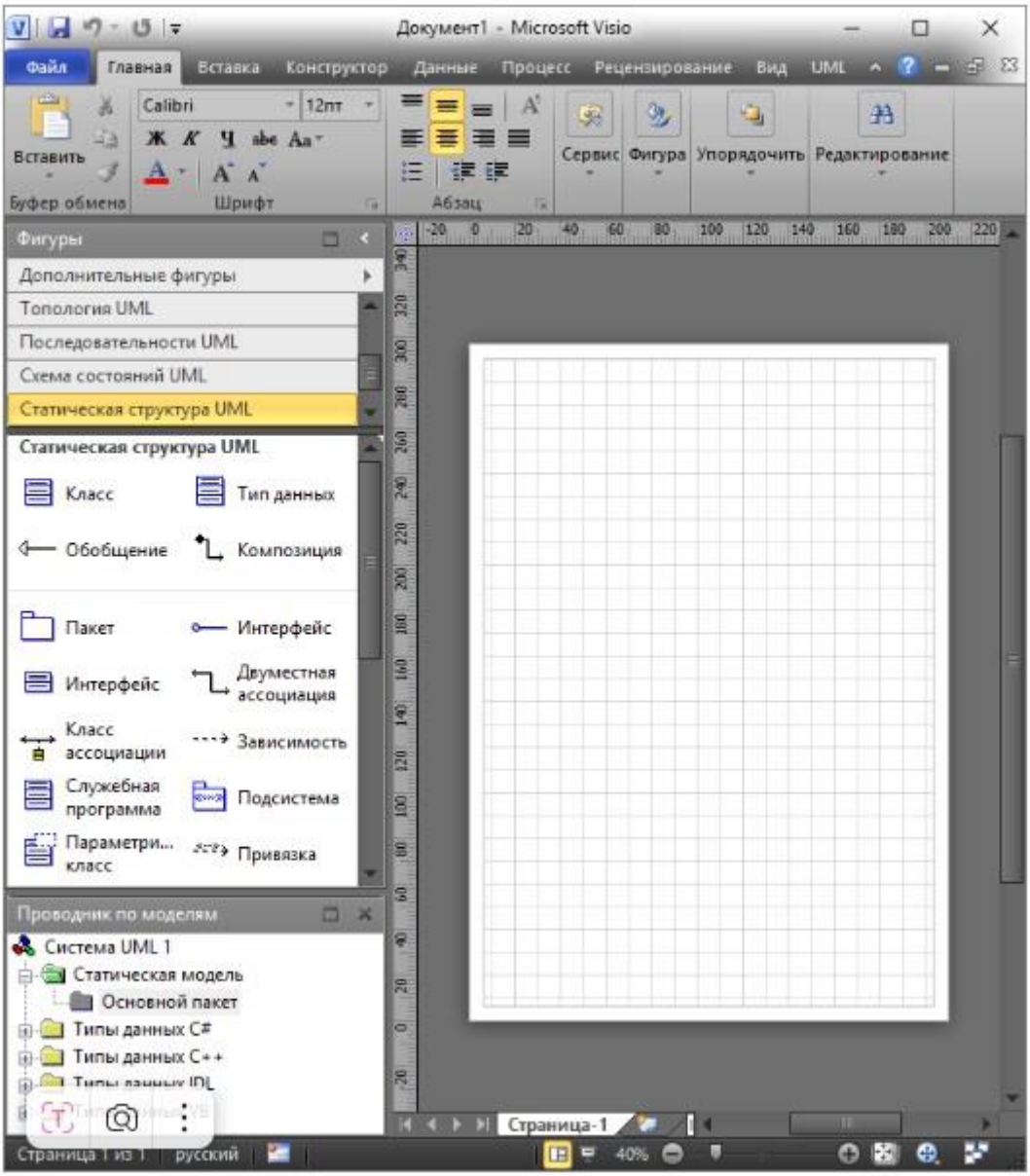
Методика выполнения

В качестве примера рассматривается моделирование системы продажи товаров по каталогу.

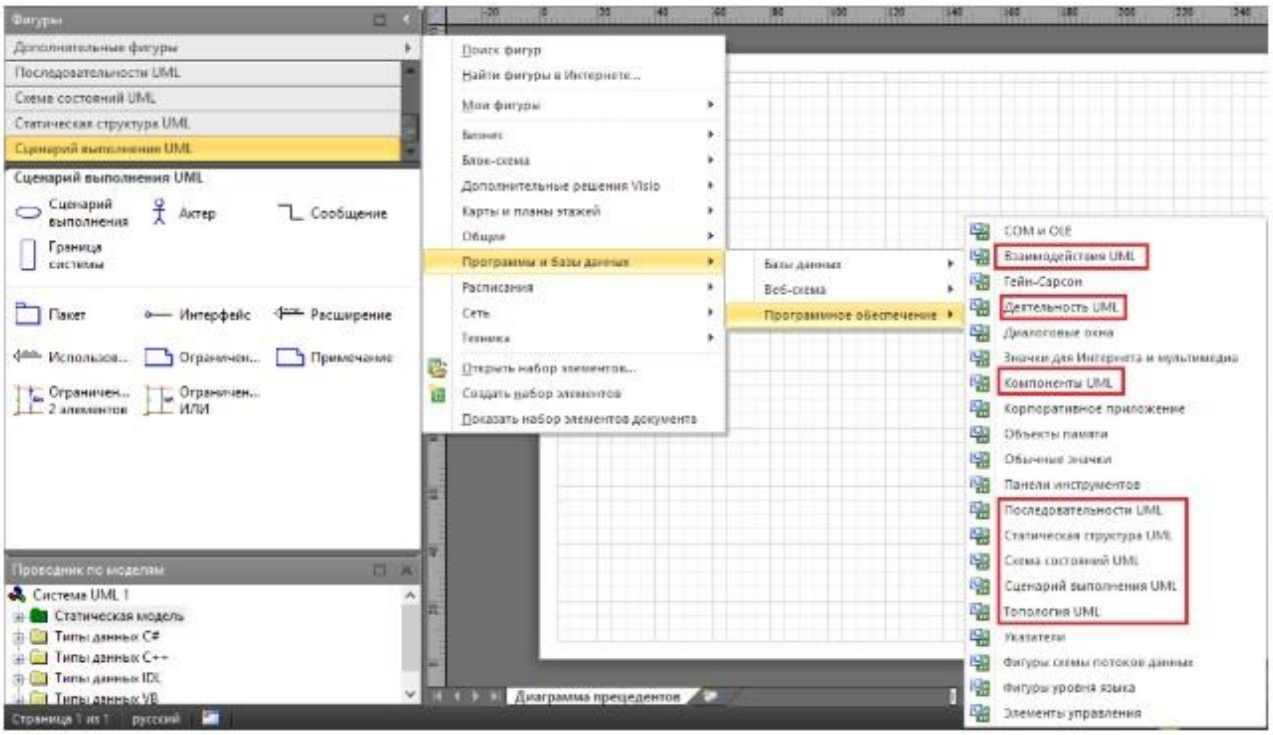
Запустите MS Visio.

На экране выбора шаблона выберите категорию ПрограммыиБДи в ней элемент СхемамоделиUML. Нажмите кнопку Создать в правой части экрана.

Окно программы примет вид



Далее необходимо открыть все фигуры, необходимые для построения UML-диаграмм. Для этого в левой части экрана необходимо нажать кнопку Дополнительныефигуры.В открывшемся вспомогательном меню выбрать Программы и БД -> Программное обеспечение и выбрать все доступные фигуры для построения UML



После этого необходимо провести следующие этапы моделирования.

Выбор актеров.

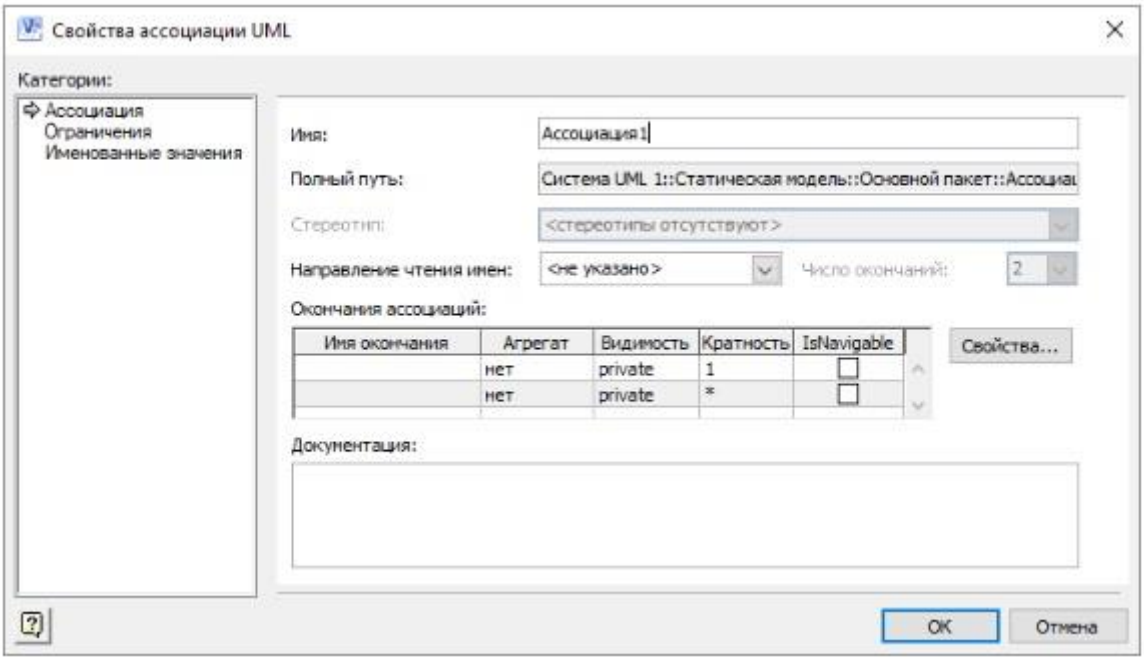
В качестве актеров данной системы могут выступать два субъекта, один из которых является продавцом, а другой – покупателем. Каждый из этих актеров взаимодействует с рассматриваемой системой продажи товаров по каталогу и является ее пользователем, т. е. они оба обращаются к соответствующему сервису «Оформить заказ на покупку товара». Как следует из существа выдвигаемых к системе требований, этот сервис выступает в качестве варианта использования разрабатываемой диаграммы, первоначальная структура которой может включать в себя только двух указанных актеров и единственный вариант использования.

В группе фигур СценарийвыполненияUMLвыбрать блок Границасистемыи добавить его на лист.

Внутрь границы системы добавить блок Сценарийвыполненияи добавить к нему название, дважды щелкнув внутри блока.

Добавить два блока Актер–покупатель и продавец.

С помощью блока Сообщение установите связь актеров и варианта использования. Двойным щелчком правой кнопки мыши по блоку Сообщение откройте окно Свойств ассоциации UML, проведите настройки

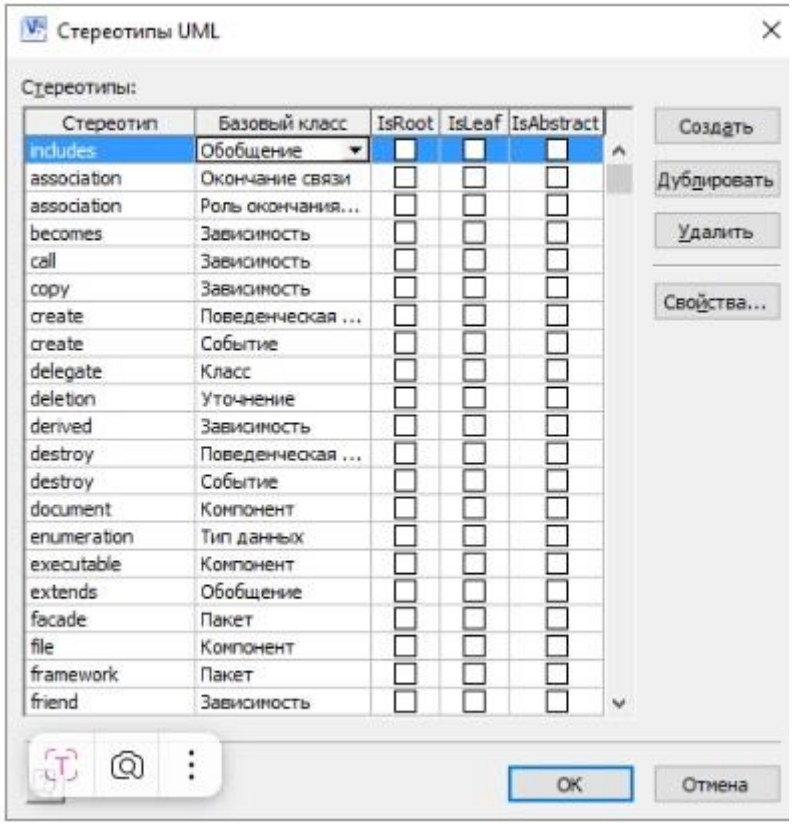


Выделение дополнительных вариантов использования.

Детализировать вариант использования «Оформить заказ на продажу товара» можно выделив следующие дополнительные варианты использования:

- обеспечить покупателя информацией – является отношением включения;
- согласовать условия оплаты – является отношением включения;
- заказать товар со склада – является отношением включения;
- запросить каталог товаров – является отношением расширения.

Так как в MS Visio отсутствует отношение включения, его необходимо добавить самостоятельно. Для этого перейти на вкладку UML -> в группе Модель выбрать пункт Стереотипы. В открывшемся окне нажать кнопку Создать и настроить стереотип



Далее на новом листе необходимо добавить границу системы и все варианты использования. После чего соединить варианты использования с помощью блока Расширение.

Для того, чтобы изменить тип отношения дважды щелкните по стрелке и окне свойств задайте необходимые параметры.

Дополненная диаграмма вариантов использования примет вид

Написание описательной спецификации для каждого варианта использования.

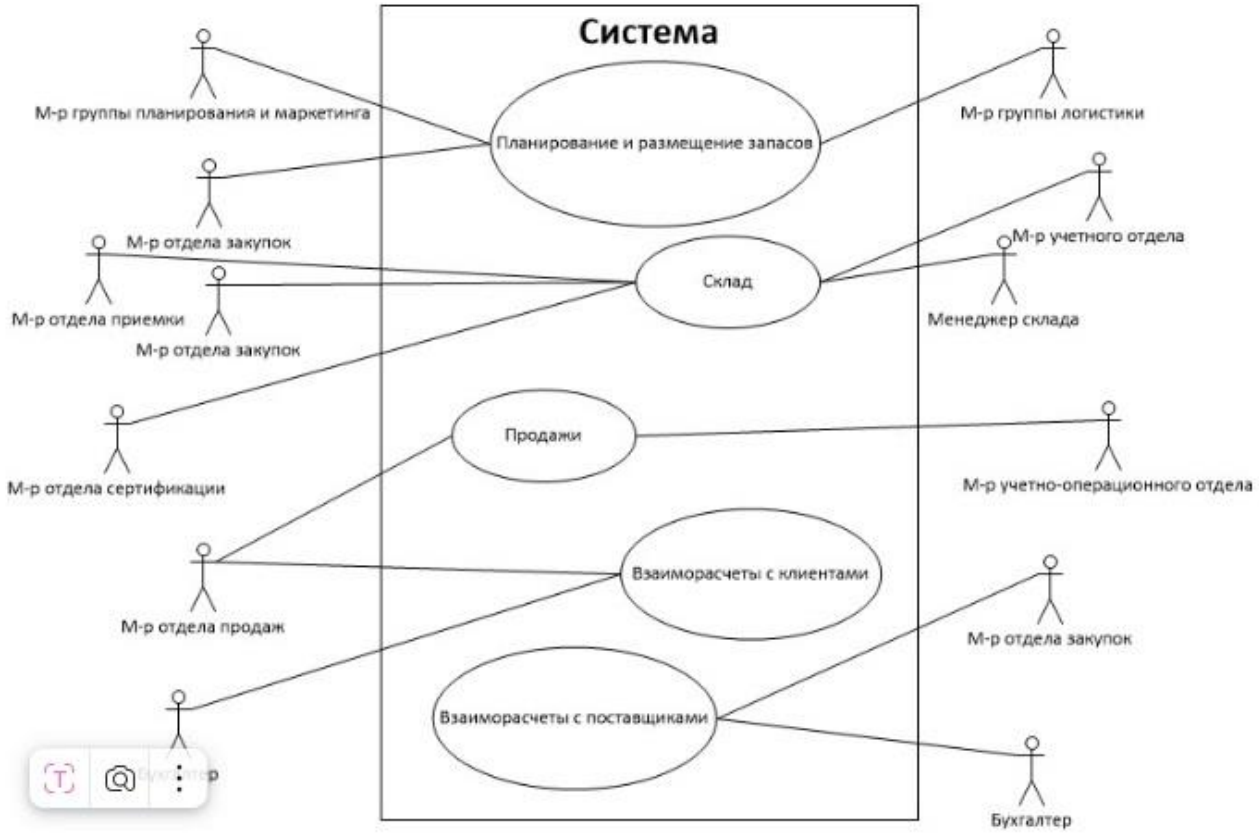
Спецификация для варианта использования «Оформить заказ на покупку компьютера»

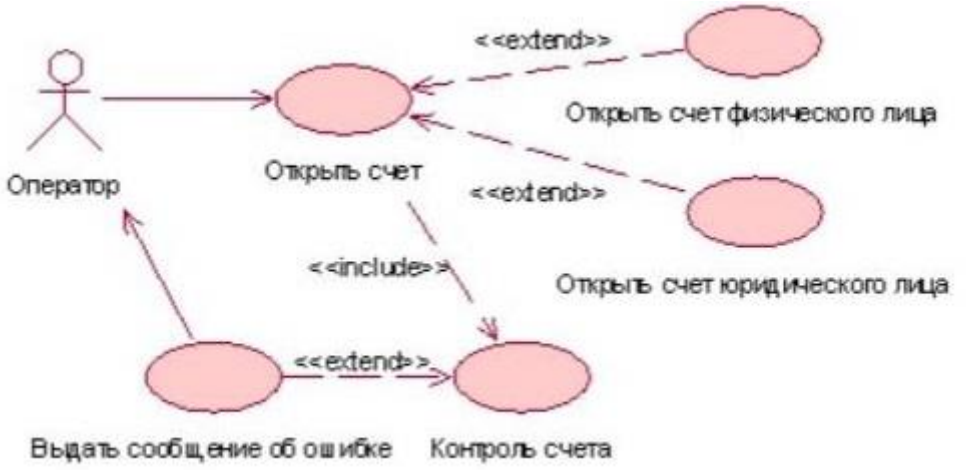
приведена в таблице

Раздел	Описание
Краткое описание	Покупатель желает оформить заказ на покупку компьютера, который он выбрал в каталоге товаров. При условии, что клиент зарегистрирован и выбранный компьютер есть в наличии оформляется заказ. Если клиент не зарегистрирован, то предлагается ему пройти регистрацию, и после этого заказать выбранный компьютер. Если компьютера нет в наличии, то предлагается заказать товар со склада в течении заданного срока поставки.
Субъекты	Продавец, Покупатель
Предусловия	В каталоге товаров имеются компьютеры, которые можно заказать. У покупателей есть доступ к системе для регистрации. Продавцы умеют пользоваться рассматриваемой системой продажи. У покупателя есть бонусы.
Основной поток	Зарегистрированный покупатель имеет возможность заказать любой компьютер из каталога товаров. В случае наличия выбранного компьютера оформляется заказ с присвоением ему уникального номера. После этого покупателю предлагается выбрать способ оплаты и способ получения компьютера. В случае отсутствия компьютера в наличии предлагается оформить заказ со склада и ожидания его поставки в рамках указанного срока или выбрать другой компьютер.
Альтернативный поток	Покупатель не зарегистрирован. В этом случае, прежде чем оформить заказ на компьютер, ему предлагается пройти регистрацию. Попытка заказать товар, который отсутствует на складе Начисление бонусов
Постусловия	Заказ оформлен и определен срок поставки компьютера и место его получения

На рисунках ниже приведены примеры диаграмм вариантов использования для различных систем.


Пишет пресс-релиз
<<include>>
Проверка правописания
<<include>>
Маркетолог
Выбор получателей
<<include>>
Рассылает пресс-релиз
<<extend>>
Шифрование текста
Отправка внешнего email
Отправка факса
Отправка через IM
Отправка внутреннего email
Отправка "бумажным" письмом
Телефонный шлюз


Система
М-р группы планирования и маркетинга
Планирование и размещение запасов
М-р группы логистики
М-р отдела закупок
М-р учетного отдела
М-р отдела приемки
Склад
Менеджер склада
М-р отдела закупок
М-р отдела сертификации
Продажи
М-р учетно-операционного отдела
Взаиморасчеты с клиентами
М-р отдела продаж
М-р отдела закупок
Взаиморасчеты с поставщиками
Бухгалтер



Задание

Построить диаграмму прецедентов (вариантов использования) в соответствии с вариантом. Составить спецификцию.

Варианты

«Отдел кадров»;
«Агентство аренды»;
«Аптека»;
«Ателье»;
«Аэропорт»;
«Библиотека»;
«Кинотеатр»;
«Поликлиника»;
«Автосалон»; 10.«Таксопарк».

Контрольные вопросы

Для чего используется язык UML?
Назначение диаграммы вариантов использования?
Что такое «актер»?
Что такое «вариант использования»?
Что такое «интерфейс»?
Что такое «примечание»?
Перечислить виды отношений между актерами и вариантами использования, охарактеризовать каждое из них?

Задание 2.1.2.3. Практическая работа № 7 «Построение диаграмм классов на языке UML с помощью MS Visio»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.2., ПК2.3., ПК2.4., ПК2.5.*

Цель: изучение основ создания диаграмм классов на языке UML, получение навыков построения диаграмм классов, применение приобретенных навыков для построения объектноориентированных моделей определенной предметной области.

Задание 1. Создание физической диаграммы в MS Visio:

1. Запустите MS Visio. (Кнопка «Пуск»/ «Программы» / MS Visio).

2. Установите следующие параметры страницы: Ориентация – Альбомная, Автоподбор размера – выключен, Имя страницы – Диаграмма классов для системы продажи товаров по каталогу.

3. Перейдите в категорию Статическая структура UML, ознакомьтесь с содержимым этой категории и найдите элементы: Класс, Пакет, Подсистема, Интерфейс, Метакласс,

Двусторонняя ассоциация, Обобщение, Композиция, Примечание, Ограничение и др.

4. Создайте поэтапно статическую структуру классов UML, с помощью которой может быть сформирована некоторая функциональная часть системы, например, Система продажи товаров по каталогу.

Для чего:

–Выберите структурные элементы (идентифицируйте классы), участвующие в организации продаж, например, Продавец, Товар, Заказ, Заказ_Оплата, Клиент, Корпоративный_Клиент, Частный_Клиент и создайте предварительный вариант совокупности классов с указанием имен

– Установите для каждого класса атрибуты в соответствии с перечнем и содержательным описанием бизнес- процессов: например, для класса Продавец в качестве атрибутов могут выступать данные: фамилия, имя, отчество, телефон. В данном случае все атрибуты видимы, принадлежат основному пакету Продавец.

- для класса Товар в качестве атрибутов могут выступать данные: тип, марка, артикул

Рисунок 119 – Формирование статической структуры объектов диаграммы

Рисунок 120 – Описание атрибутов класса Продавец

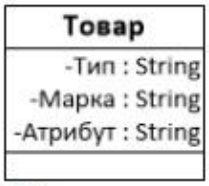
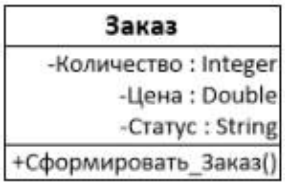


Рисунок 121 – Описание атрибутов класса Товар

для класса *Заказ* в качестве атрибутов могут выступать данные: количество, цена, статус, а в качестве операций – сформировать заказ.



для класса *Заказ_Оплата* в качестве атрибутов могут выступать данные: дата получения, проплачен, номер, цена, а в качестве операций – отправить, закрыть.

Рисунок 123 – Описание атрибутов и операций класса Заказ_Оплата

для класса *Клиент* в качестве атрибутов могут выступать данные: имя, адрес, а в качестве операций – кредитный рейтинг.

Рисунок 124 – Описание атрибутов и операций класса Клиент

для класса *Корпоративный_Клиент* в качестве атрибутов могут выступать данные: контактное имя, кредитный рейтинг, кредитный лимит, а в качестве операций – сделать, напоминание, счет за месяц.

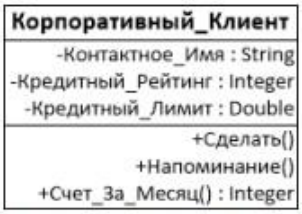


Рисунок 125 – Описание атрибутов и операций класса Корпоративный_Клиент

для класса *Частный_Клиент* в качестве атрибутов могут выступать данные: номер кредитной карты.

Рисунок 126 – Описание атрибутов класса Частный_Клиент

для класса *Вариант_Оплаты* в качестве атрибутов могут выступать данные: тип оплаты, а в качестве операций

– выбор варианта оплаты.

Рисунок 128 – Описание атрибутов и операций класса Каталог_товаров

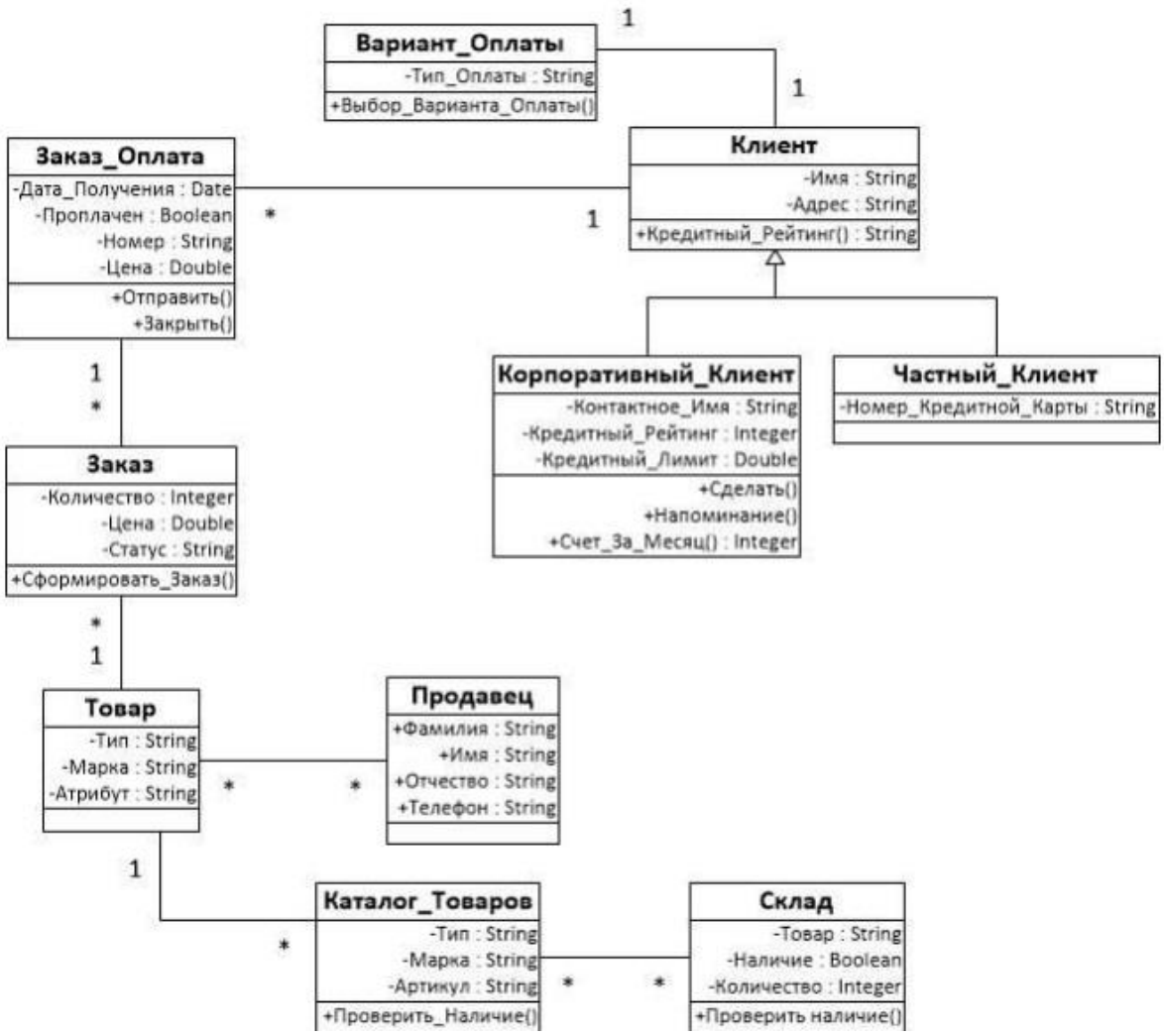
для класса *Склад* в качестве атрибутов могут выступать данные: товар, наличие, количество, а в качестве операций – Проверить наличие.

– Убедитесь, что все элементы наполнены адекватным содержанием и расположите все структурные элементы диаграммы наиболее оптимально на странице для установления отношений между ними.

В качестве примера на рис. показан набор классов, описывающих реализацию системы продаж товаров по каталогу. Акцент сделан на классе Клиент, с которым связан класс Заказ_Оплата посредством двусторонней ассоциации «один-ко-многим», Вариант_Оплаты –

двусторонней ассоциацией «один-кодному» и классы Корпоративный_Клиент и Частный_Клиент посредством отношения обобщения. Классы Заказ_Оплата и Товар связаны с классом Заказ посредством двусторонней ассоциации «один-ко-многим».

Класс Товар связан с классом Продавец двусторонней ассоциацией «многие-ко-многим» и классом Каталог_Товаров двусторонней ассоциацией «один-ко-многим». Класс Каталог_Товаров связан посредством двусторонней ассоциации «многие-комногим» с классом Склад.

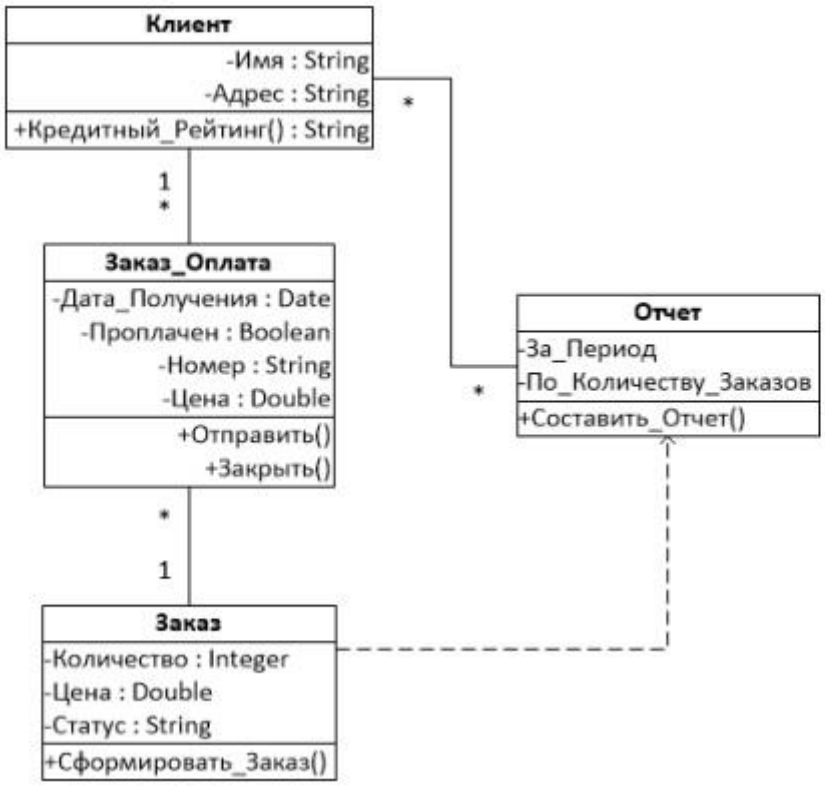


Задание 2.

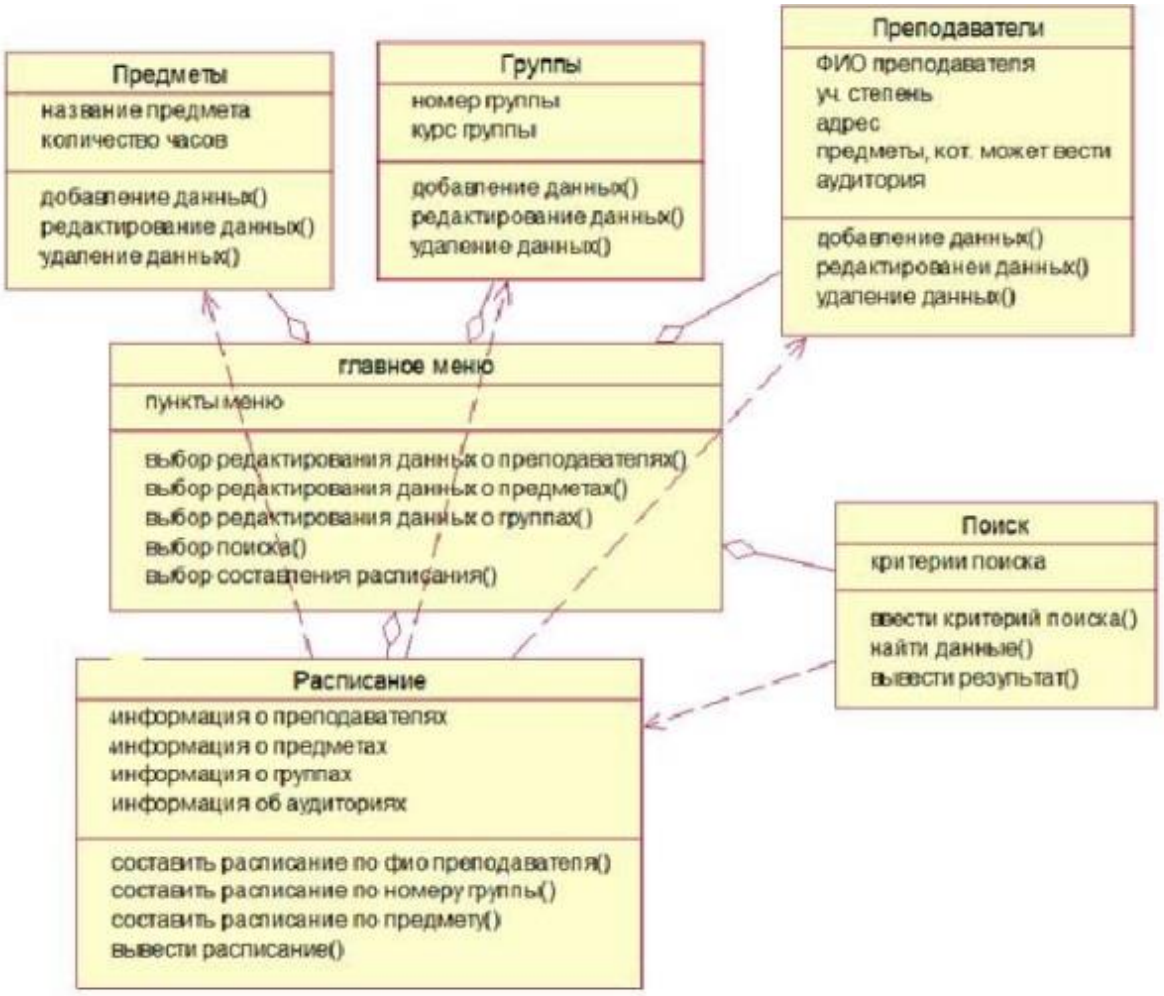
Создайте новую страницу с именем Диаграмма классов учета клиентов, и установите следующие опции:

Ориентация – Альбомная, Автоподбор размера – выключен.

1. Идентифицируйте классы учета клиентов, осуществляющих заказы и создайте диаграмму классов с указанием их имен, атрибутов, операций, например, так как показано на рис.

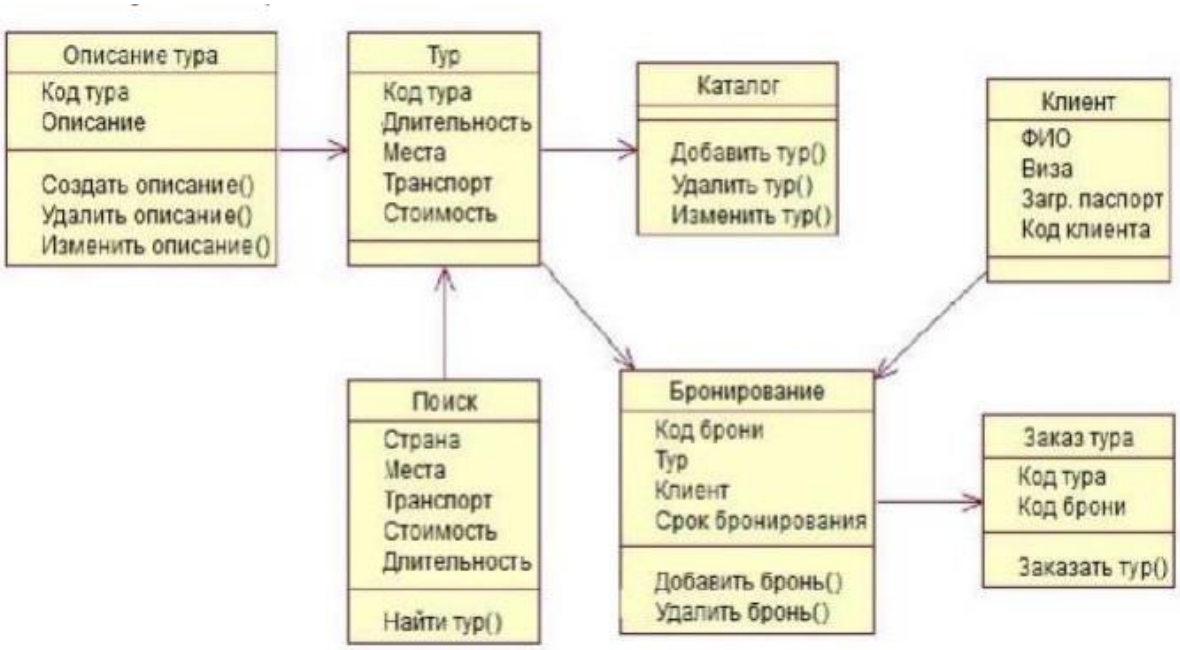


Задание 3. Аналогично заданию 1, создайте диаграмму классов БД образовательного учреждения на новой странице документа:

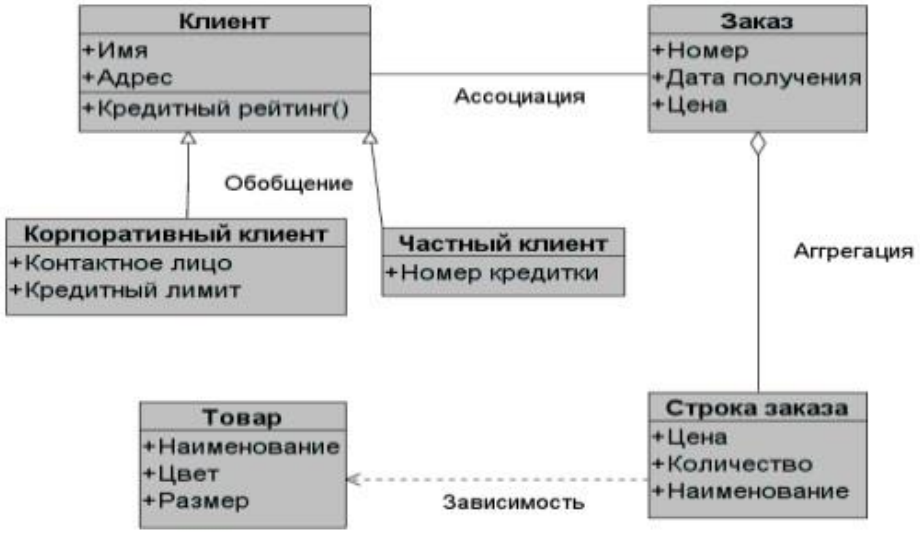


Задание 4.

Создайте диаграмму классов Информационной системы туристического агентства на новой странице документа:



Задание 5. Выполните построение диаграммы классов компании по прокату авто .



Критерии оценивания:

Оценка «отлично» – полный объем выполненных работ, выполнены все задания лабораторной работы, продемонстрировано глубокое понимание материала самостоятельность выполнения работы, аккуратность и законченность работы.

Оценка «хорошо» –все задания лабораторной работы выполнены, работа оформлена с незначительными отклонениями от требований, продемонстрировано хорошее понимание материала

Оценка «удовлетворительно» - задания лабораторной работы выполнены с замечаниями, работа имеет существенные отклонения в оформлении, продемонстрировано базовое понимание материала.

Оценка «неудовлетворительно» – отсутствие полного объема работ; в работе допущены серьёзные ошибки и нарушение всех перечисленных выше требований.

Задание 2.1.2.4. Лабораторная работа № 1 «Построение диаграммы Вариантов использования и диаграммы. Последовательности. Кооперации и диаграммы.

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.2., ПК2.3., ПК2.4., ПК2.5.*

Цель: Научиться строить диаграммы Вариантов использования и диаграммы Последовательности.

Теоретическая справка:

Визуальное моделирование в UML можно представить, как некоторый процесс поуровневого спуска от наиболее обшей и абстрактной концептуальной модели исходной системы к логической, а затем и к физической модели соответствующей программной системы.

Для достижения этих целей вначале строится модель в форме, так называемой диаграммы вариантов использования (use case diagram), которая описывает функциональное назначение системы или, другими словами, то, что система будет делать в процессе своего функционирования. Диаграмма вариантов использования является исходным концептуальным представлением или концептуальной моделью системы в процессе ее проектирования и разработки.

Разработка диаграммы вариантов использования преследует цели:

- определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;

- сформулировать общие требования к функциональному поведению проектируемой системы;

- разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;

- подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых вариантов использования. При этом актером (actor) или действующим лицом называется любая сущность, взаимодействующая с системой извне. Это может быть человек, техническое устройство, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик. В свою очередь, вариант использования (use case) служит для описания сервисов, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актером. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие актеров с системой.

Состав диаграммы Use Case

Диаграмма вариантов использования состоит из актеров, для которых система производит действие, и собственно действие Use Case, которое описывает то, что актер хочет получить от системы. Актер обозначается значком человечка, а Use Case - овалом. Дополнительно в диаграммы могут быть добавлены комментарии.

Виды взаимодействий

Между актерами и вариантами использования могут быть различные виды взаимодействия. Основные виды взаимодействия следующие:

- Простая ассоциация - отражается линией между актером и вариантом использования (без стрелки). Отражает связь актера и варианта использования. На рисунке между актером администратор и вариантом использования просматривать заказ Направленная ассоциация - то же что и простая ассоциация, но показывает, что вариант использования инициализируется актером.

Обозначается стрелкой.

- Наследование - показывает, что потомок наследует атрибуты и поведение своего прямого предка. Может применяться как для актеров, так для вариантов использования.

- Расширение (extend) - показывает, что вариант использования расширяет базовую последовательность действий и вставляет собственную последовательность. При этом в отличие от типа отношений «включение» расширенная последовательность может осуществляться в зависимости от определенных условий.

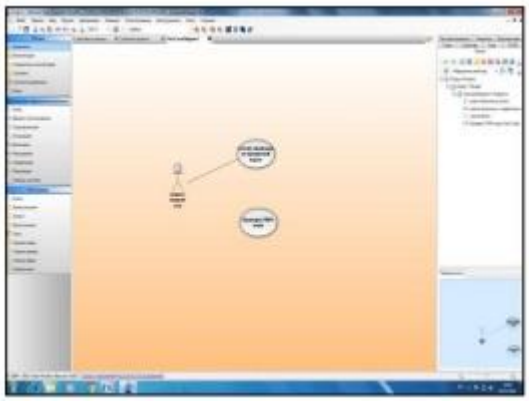
- Включение (include) - показывает, что вариант использования включается в базовую последовательность и выполняется всегда.

Существуют и другие виды взаимодействия, но они менее важны и реже применяются.

Задание 1. Построить диаграмму вариантов использования модели работы банкомата.

Выполните следующие действия:

1. Добавить актера с именем Клиент банкомата.
2. Добавить вариант использования Снятие наличных по кредитной **карте.**
3. Добавить направленную ассоциацию от бизнес-актера Клиент Банкомата к варианту использования Снятие наличных по кредитной карте
4. Добавить вариант использования Проверка ПИН-кода, как изображено на рисунке



5. Добавить актера с именем Банк.

6. Добавить вариант использования Получение справки о состоянии счета.

7. Добавить вариант использования Блокирование кредитной карточки.

8. Добавить направленную ассоциацию от бизнес-актера Клиент Банкомата к варианту использования Получение справки о состоянии счета.

9. Добавить направленную ассоциацию от варианта использования Снятие наличных по кредитной карточке к сервису Банк.

10. Добавить направленную ассоциацию от варианта использования Получение справки о состоянии счета к сервису Банк.

11. Добавить отношение зависимости со стереотипом «include», направленное от варианта использования Снятие наличных по кредитной карте к варианту использования Проверка Пин-кода.

12. Добавить отношение зависимости со стереотипом «include», направленное от варианта использования Получение справки о состоянии счета к варианту использования Проверка Пин-кода.

13. Добавить отношение зависимости со стереотипом «extend», направленное от варианта использования Блокирование кредитной карточки к варианту использования Проверка Пин-кода.

Задание 2. Построить диаграмму вариантов использования.

Имеются следующие данные:

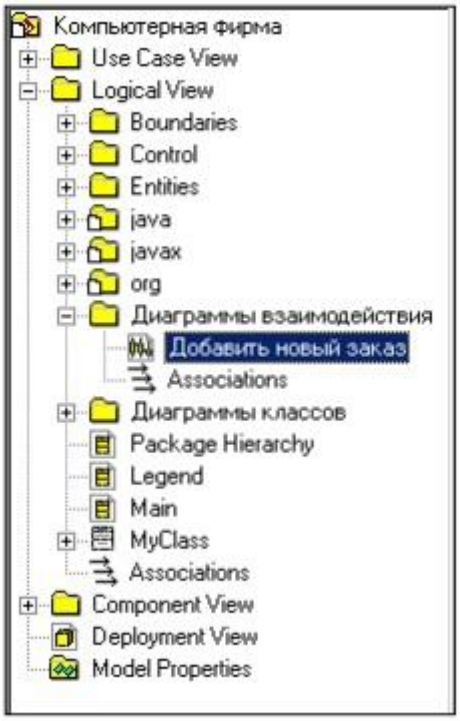
- четыре действующих лица: Клиента банка, Банк, Кассира и Оператора,

- пять вариантов использования: Снять наличные, Перевести деньги со счета, Положить деньги на счет, Пополнить запас денег и Подтвердить пользователя,

- три зависимости, и отношения между действующими лицами и вариантами использования. Варианты использования: Снять наличные, Перевести деньги со счета, Положить деньги на счет - требуют включения идентификации клиента в системе.

Это поведение может быть выделено в новый вариант использования включения, называемый Подтвердить пользователя. Базовые варианты использования не зависимы от метода, используемого для идентификации. Поэтому он инкапсулируется (скрывается) в варианте использования включения. С точки зрения базовых вариантов использования не имеет значение производится ли идентификация с помощью магнитной карты или сканированием сетчатки глаза. Они только зависят от результата выполнения варианта использования Подтвердить клиента.

Для создания диаграммы Последовательности необходимо щелкнуть правой кнопкой мыши по пакету Диаграммы взаимодействия и в появившемся меню выбрать пункт New > Sequence Diagram, ввести ее имя, после чего дважды щелкнуть по ней в браузере, как показано на рисунке



Построение диаграммы последовательности начинается с размещения на ней объектов, которые будут обмениваться сообщениями. Сначала необходимо разместить объекты, которые посылают сообщения, а потом объекты, получающие их. Инициатором взаимодействия выступает актер Менеджер по работе с клиентами. Поэтому на диаграмме он будет находится в левом углу, как показано на рисунке.

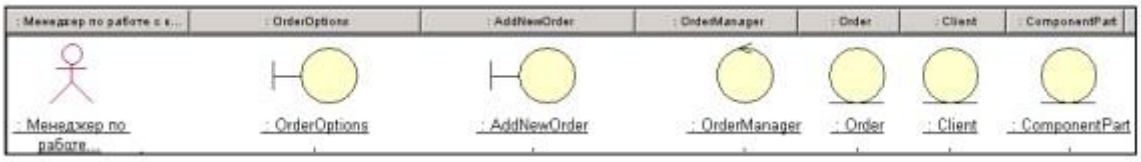
- объект класса OrderOptions (Параметры работы с заказом), отвечающий за выбор возможного действия с заказом в рассматриваемом прецеденте;

- объект класса AddNewOrder (Добавление нового заказа), отвечающий за добавление заказа;

- объект класса OrderManager (Менеджер по работе с заказами), отвечающий за обработку потока событий рассматриваемого прецедента;

- объект класса Order (Заказ);

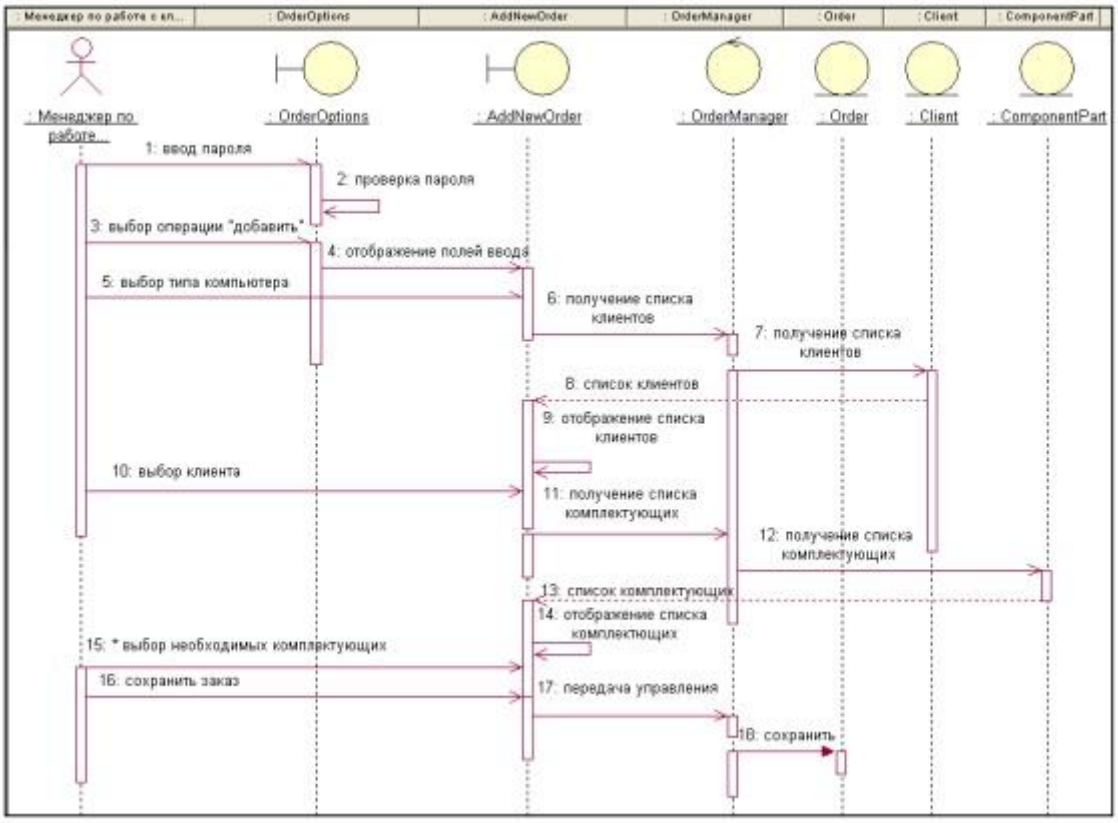
- объект класса Client (Клиент); - объект класса ComponentPart (Комплектующее изделие).



Теперь на диаграмме следует разместить сообщения, которыми будут обмениваться объекты, представленные в таблице

№ сообщения	Объект - отправитель сообщения	Объект - получатель сообщения	Название
1	Менеджер по работе с клиентами	OrderOptions	ввод пароля
2	OrderOptions	OrderOptions	проверка пароля
3	Менеджер по работе с клиентами	OrderOptions	выбор операции "добавить"
4	OrderOptions	AddNewOrder	отображение полей ввода
5	Менеджер по работе с клиентами	AddNewOrder	выбор типа компьютера
6	AddNewOrder	OrderManager	получение списка клиентов
7	OrderManager	Client	получение списка клиентов
8	Client	AddNewOrder	список клиентов
9	AddNewOrder	AddNewOrder	отображение списка клиентов
10	Менеджер по работе с клиентами	AddNewOrder	выбор клиента
11	AddNewOrder	OrderManager	получение списка комплектующих
12	OrderManager	ComponentPart	получение списка комплектующих
13	ComponentPart	AddNewOrder	список комплектующих
14	AddNewOrder	AddNewOrder	отображение списка комплектующих
15	Менеджер по работе с клиентами	AddNewOrder	* выбор необходимых комплектующих
16	Менеджер по работе с клиентами	AddNewOrder	сохранить заказ
17	AddNewOrder	OrderManager	передача управления
18	OrderManager	Order	сохранить

В итоге получается диаграмма последовательности, показанная на рисунке



Задание 2.1.2.4. Лабораторная работа № 2 «Построение диаграммы Деятельности, диаграммы Состояний и диаграммы Классов. Построение диаграммы компонентов.

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.2., ПК2.3., ПК2.4., ПК2.5.*

Цель: Научиться строить диаграмму Деятельности.

Теоретические сведения:

При моделировании поведения системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке UML используются так называемые диаграммы деятельности. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние срабатывает только при завершении этой операции в предыдущем состоянии.

Графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия, а дугами - переходы от одного состояния действия к другому. На диаграмме деятельности отображается логика или последовательность перехода от одной деятельности к другой, при этом внимание фиксируется на результате деятельности. Компонентами диаграммы деятельности являются:

-состояния действия,

-переходы,

-дорожки,

-объекты

Состояние действия.

Состояние действия (action state) является специальным случаем состояния с некоторым входным действием и, по крайней мере, одним выходящим из состояния переходом. Этот переход неявно предполагает, что входное действие уже завершилось. Состояние действия не может иметь внутренних переходов, поскольку оно является элементарным. Обычное использование состояния действия заключается в моделировании одного шага выполнения

алгоритма (процедуры) или потока управления.

Внутри фигуры записывается выражение действия (action-expression), которое должно быть уникальным в пределах одной диаграммы деятельности.

Переходы.

При построении диаграммы деятельности используются только нетриггерные переходы, т. е. такие, которые срабатывают сразу после завершения деятельности или выполнения соответствующего действия. На диаграмме такой переход изображается сплошной линией со стрелкой.

Если из состояния действия выходит не единственный переход, то сработать может только один из них. Тогда для каждого из таких переходов должно быть явно записано сторожевое условие в форме булевского выражения в прямых скобках.

Дорожки.

Диаграммы деятельности могут быть т.к. использованы для моделирования бизнес-процессов. Применительно к бизнес-процессам желательно выполнение каждого действия ассоциировать с конкретным подразделением компании. В этом случае подразделение несет ответственность за реализацию отдельных действий, а сам бизнес-процесс представляется в виде переходов действий из одного подразделения к другому.

Для моделирования этих особенностей в языке UML используется специальная конструкция, получившее название дорожки (swimlanes). При этом все состояния действия на диаграмме деятельности делятся на отдельные группы, которые отделяются друг от друга вертикальными линиями. Две соседние линии и образуют дорожку, а группа состояний между этими линиями выполняется отдельным подразделением

Названия подразделений явно указываются в верхней части дорожки.

Пересекать линию дорожки могут только переходы, которые в этом случае обозначают выход или вход потока управления в соответствующее подразделение компании.

Объекты. В общем случае действия на диаграмме деятельности выполняются над теми или иными объектами. Эти объекты либо инициируют выполнение действий, либо определяют некоторый результат этих действий.

Для графического представления объектов используются прямоугольник класса, с тем отличием, что имя объекта подчеркивается.

Далее после имени может указываться характеристика состояния объекта в прямых скобках. Такие прямоугольники объектов присоединяются к состояниям действия отношением зависимости пунктирной линией со стрелкой.

Задание: построить диаграмму Деятельности.

Содержание работы:

1. Изучить теоретические сведения по теме “Диаграмма деятельности».
2. Разработать диаграмму деятельности для произвольной системы индивидуального задания
3. Оформить отчет, включив в него описание всех компонентов диаграммы деятельности согласно индивидуальному варианту задания

Критерии оценивания:

Оценка **«отлично»** – полный объем выполненных работ, выполнены все задания лабораторной работы, продемонстрировано глубокое понимание материала самостоятельность выполнения работы, аккуратность и законченность работы.

Оценка **«хорошо»** –все задания лабораторной работы выполнены, работа оформлена с незначительными отклонениями от требований, продемонстрировано хорошее понимание материала

Оценка **«удовлетворительно»** - задания лабораторной работы выполнены с замечаниями, работа имеет существенные отклонения в оформлении, продемонстрировано базовое понимание материала.

Оценка **«неудовлетворительно»** – отсутствие полного объема работ; в работе допущены серьёзные ошибки и нарушение всех перечисленных выше требований.

Тема 2.1.3. Оценка качества программных средств

Задание 2.1.3.1. Лабораторная работа № 3. «Разработка тестового сценария и тестовых пакетов»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.2., ПК2.3., ПК2.4., ПК2.5.*

Цели: усвоить знание о видах тестирования; освоить способы обнаружения и фиксирования ошибок.

Теоретические сведения

Общепринятая практика состоит в том, что после завершения продукта и до передачи его заказчику независимой группой тестировщиков проводится тестирование ПО.

Уровни тестирования:

Модульное тестирование. Тестируется минимально возможный для тестирования компонент, например отдельный класс или функция;

Интеграционное тестирование. Проверяется, есть ли какие-либо проблемы в интерфейсах и взаимодействии между интегрируемыми компонентами, например, не передается информация, передается некорректная информация;

Системное тестирование. Тестируется интегрированная система на ее соответствие исходным требованиям.

Виды программных ошибок	Способы их обнаружения
Ошибки выполнения, выявляемые автоматически: а) переполнение, защита памяти; б) несоответствие типов; в) зацикливание	Динамический контроль: аппаратурой процессора; run-time системы программирования; операционной системой – по превышению лимита времени

Тест – это набор контрольных входных данных совместно с ожидаемыми результатами.

Тесты должны обладать определенными свойствами.

Детективность: тест должен с большой вероятностью обнаруживать возможные ошибки.

Покрывающая способность: один тест должен выявлять как можно больше ошибок.

Воспроизводимость: ошибка должна выявляться независимо от изменяющихся условий.

С помощью тестирования разных видов обнаруживаются ошибки в разрабатываемом программном обеспечении. После обнаружения ошибок проводится их устранение.

Задание 1

Создать приложение Простой калькулятор, в котором реализовать выполнение простых операций с вводимыми двумя операндами. Выполнить тестирование приложения на различных данных, отличающихся по типу и значению.

Программа работы

1. Разработать интерфейс приложения и написать программные коды для событий кнопок.

2. Сохранить проект в отдельной папке, скопировать исполняемый файл на рабочий стол.

3. Составить тесты для проверки работы приложения.

4. Провести тестирование исполняемого файла

Задание 2

В Древней Греции (II в. до н.э.) был известен шифр, называемый "квадрат Полибия".

Шифровальная таблица представляла собой квадрат с пятью столбцами и пятью

строками, которые нумеровались цифрами от 1 до 5. В каждую клетку такого квадрата записывалась одна буква. В результате каждой букве соответствовала пара чисел, и шифрование сводилось к замене буквы
парой чисел. Для латинского алфавита квадрат Полибия имеет вид:

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I, J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Пользуясь изложенным способом создать программу, которая:
а) зашифрует введенный текст и сохранит его в файл;
б) считает зашифрованный текст из файла и расшифрует данный текст.

Задание 3

Спроектировать тесты по принципу «белого ящика» для программы, разработанной в задании № 2. Выбрать несколько алгоритмов для тестирования и обозначить буквами или цифрами ветви этих алгоритмов. Выписать пути алгоритма, которые должны быть проверены тестами для выбранного метода тестирования. Записать тесты, которые позволят пройти по путям алгоритма.

Протестировать разработанную вами программу. Результаты оформить в виде таблиц:

Тест	Ожидаемый результат	Фактический результат	Результат тестирования
…	…	…	…

Задание 4

Проверить все виды тестов и сделать выводы об их эффективности

Задание 2.1.3.2. Лабораторная работа № 4. «Оценка программных средств с помощью метрик»

Проверяемые результаты обучения: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК4.1

Цель: знакомство с ГОСТ 28.195-89 «Оценка качества программных средств. Общие положения»; определить способы получения информации о ПС; формирование информационно -правовых компетенции обучающихся.

Теоретические сведения

Необходимая документация: ГОСТ 28.195-89

Одной из важнейших проблем обеспечения качества программных средств является формализация характеристик качества и методология их оценки. Для определения адекватности качества функционирования, наличия технических возможностей программных средств к взаимодействию, совершенствованию и развитию необходимо использовать стандарты в области оценки характеристик их качества.

Показатели качества программного обеспечения устанавливают ГОСТ 28.195-89 «Оценка качества программных средств. Общие положения» и ГОСТ Р ИСО/МЭК 9126 «Информационная технология. Оценка программной продукции. Характеристика качества и руководства по их применению». Одновременное существование двух действующих стандартов, нормирующих одни и те же показатели, ставит вопрос об их гармонизации. Ниже рассмотрим каждый из перечисленных стандартов. ГОСТ 28.195-89 «Оценка качества программных средств. Общие положения» устанавливает общие положения по оценке качества программных средств, номенклатуру и применяемость показателей качества.

Оценка качества ПС представляет собой совокупность операций, включающих выбор номенклатуры показателей качества оцениваемого ПС, определение значений этих показателей и сравнение их с базовыми значениями.

Методы определения показателей качества ПС различаются: по способам получения информации о ПС – измерительный, регистрационный, органолептический, расчетный; по источникам получения информации – экспертный, социологический.

Измерительный метод основан на получении информации о свойствах и характеристиках ПС с использованием инструментальных средств. Например, с использованием этого метода определяется объем ПС - число строк исходного текста программ и число строк - комментариев, число операторов и операндов, число исполненных операторов, число ветвей в программе, число точек входа (выхода), время выполнения ветви программы, время реакции и другие показатели.

Регистрационный метод основан на получении информации во время испытаний или функционирования ПС, когда регистрируются и подсчитываются определенные события, например, время и число сбоев и отказов, время передачи управления другим модулям, время начала и окончания работы.

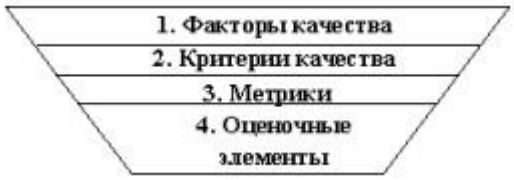
Органолептический метод основан на использовании информации, получаемой в результате анализа восприятия органов чувств (зрения, слуха), и применяется для определения таких показателей как удобство применения, эффективность и т. п.

Расчетный метод основан на использовании теоретических и эмпирических зависимостей (на ранних этапах разработки), статистических данных, накапливаемых при испытаниях, эксплуатации и сопровождении ПС. При помощи расчетного метода определяются длительность и точность вычислений, время реакции, необходимые ресурсы.

Определение значений показателей качества ПС экспертным методом осуществляется группой экспертов-специалистов, компетентных в решении данной задачи, на базе их опыта и интуиции. Экспертный метод применяется в случаях, когда задача не может быть решена никаким другим из существующих способов или другие способы являются значительно более трудоемкими.

Экспертный метод рекомендуется применять при определении показателей наглядности, полноты и доступности программной документации, легкости освоения, структурности.

Социологические методы основаны на обработке специальных анкет-вопросников.



Показатели качества объединены в систему из четырех уровней. Каждый вышестоящий уровень содержит в качестве составляющих показатели нижестоящих уровней.

Стандарт ИСО 9126 (ГОСТ Р ИСО/МЭК 9126) «Информационная технология. Оценка программной продукции. Характеристика качества и руководства по их применению».

Определенные настоящим стандартом характеристики дополнены рядом требований по выбору метрик и их измерению для различных проектов ПС. Они применимы к любому типу ПС, включая компьютерные программы и данные, содержащиеся в программируемом оборудовании.

Эти характеристики обеспечивают согласованную терминологию для анализа качества ПС. Кроме того, они определяют схему для выбора и специфицирования требований к качеству ПС, а также для сопоставления возможностей различных программных продуктов, таких как функциональные возможности, надежность, практичность и эффективность.

Все множество атрибутов качества ПС может быть классифицировано в структуру иерархического дерева характеристик и субхарактеристик. Самый высший уровень этой структуры состоит из характеристик качества, а самый нижний уровень – из их атрибутов. Эта

иерархия не строгая, поскольку некоторые атрибуты могут быть связаны с более чем одной субхарактеристикой.

Таким же образом, внешние свойства (такие, как пригодность, корректность, устойчивость к ошибкам или временная эффективность) влияют на наблюдаемое качество. Недостаток качества в использовании (например, пользователь не может закончить задачу) может быть прослежен к внешнему качеству (например, функциональная пригодность или простота использования) и связанным с ним внутренним атрибутам, которые необходимо изменить.

Внутренние метрики могут применяться в ходе проектирования и программирования к неисполняемым компонентам ПС (таким, как спецификация или исходный программный текст).

При разработке ПС промежуточные продукты следует оценивать с использованием внутренних метрик, которые измеряют свойства программ, и могут быть выведены из моделируемого поведения. Основная цель внутренних метрик – обеспечивать, чтобы было достигнуто требуемое внешнее качество. Внутренние метрики дают возможность пользователям, испытателям и разработчикам оценивать качество ЖЦ программ и заниматься вопросами технологического обеспечения качества задолго до того, как ПС становится готовым исполняемым продуктом.

Внутренние метрики позволяют измерять внутренние атрибуты или формировать признаки внешних атрибутов путем анализа статических свойств промежуточных или поставляемых программных компонентов. Измерения внутренних метрик используют категории, числа или характеристики элементов из состава ПС, которые, например, имеются в процедурах исходного программного текста, в графе потока управления, в потоке данных и в представлениях изменения состояний памяти. Документация также может оцениваться с использованием внутренних метрик.

Внешние метрики используют меры ПС, выведенные из поведения системы, частью которых они являются, путем испытаний, эксплуатации или наблюдения исполняемого ПС или системы. Перед приобретением или использованием ПС его следует оценить с использованием метрик, основанных на деловых и профессиональных целях, связанных с использованием, эксплуатацией и управлением продуктом в определенной организационной и технической среде.

Внешние метрики обеспечивают заказчикам, пользователям, испытателям и разработчикам возможность определять качество ПС в ходе испытаний или эксплуатации.

Когда требования к качеству ПС определены, в них должны быть перечислены характеристики и субхарактеристики, которые составляют полный набор показателей качества.

Затем определяются подходящие внешние метрики и их приемлемые диапазоны значений, устанавливающие количественные и качественные критерии, которые подтверждают, что ПС удовлетворяет потребностям заказчика и пользователя. Далее определяются и специфицируются внутренние атрибуты качества, чтобы спланировать удовлетворение требуемых внешних характеристик качества в конечном продукте и обеспечивать их в промежуточных продуктах в ходе разработки. Подходящие внутренние метрики и приемлемые диапазоны специфицируются для получения числовых значений или категорий внутренних характеристик качества, чтобы их можно было использовать для проверки того, что промежуточные продукты в процессе разработки удовлетворяют внутренним спецификациям качества. Рекомендуется использовать внутренние метрики, которые имеют наиболее сильные связи с целевыми внешними метриками, чтобы они могли помогать при прогнозировании значений внешних метрик.

Метрики качества в использовании измеряют, в какой степени продукт удовлетворяет потребности конкретных пользователей в достижении заданных целей с результативностью, продуктивностью и удовлетворением в заданном контексте использования. При этом результативность подразумевает точность и полноту достижения определенных целей пользователями при применении ПС; продуктивность соответствует соотношению израсходованных ресурсов и результативности при эксплуатации ПС, а удовлетворенность –

психологическое отношение к качеству использования продукта. Эта метрика не входит в число шести базовых характеристик ПС, регламентируемых стандартом ИСО 9126, однако рекомендуется для интегральной оценки результатов функционирования комплексов программ.

Оценивание качества в использовании должно подтверждать его для определенных сценариев и задач, оно составляет полный объединенный эффект характеристик качества ПС для пользователя. Качество в использовании – это восприятие пользователем качества системы, содержащей ПС, и оно измеряется скорее в терминах результатов использования комплекса программ, чем собственных внутренних свойств ПС. Связь качества в использовании с другими характеристиками качества ПС зависит от типа пользователя, так, например, для конечного пользователя качество в использовании обусловливают, в основном, характеристики функциональных возможностей, надежности, практичности и эффективности, а для персонала сопровождения ПС качество в использовании определяет сопровождаемость. На качество в использовании могут влиять любые характеристики качества, и это понятие шире, чем практичность, которая связана с простотой использования и привлекательностью. Качество в использовании, в той или иной степени, характеризуется сложностью применения комплекса программ, которую можно описать трудоемкостью использования с требуемой результативностью.

Многие характеристики и субхарактеристики ПС обобщенно отражаются неявными технико-экономическими показателями, которые поддерживают функциональную пригодность конкретного ПС. Однако их измерение и оценка влияния на показатели качества, представляет сложную проблему.

Задание №1. Провести сравнение понятий «качество» государственным и международным стандартами. Выписать документы, в которых даны данные определения.

Задание №2. Опишите методы получения информации о ПС по ГОСТу. Для каждого метода выделите источник информации.

Задание №3. Выберите стандарты для оценки качества ПС. Перечислите критерии надежности ПС по ГОСТу.

Задание №4. Методика оценки качественных показателей ПП основана на составлении метрики ПП. В лабораторной работе необходимо выполнить следующее:

1. Выбрать показатели качества (не менее 5) и сформулировать их сущность. Каждый показатель должен быть существенным, т. е. должны быть ясны потенциальные выгоды его использования. Показатели представить в виде таблицы

Показатели качества	Сущность показателя	Экспертная оценка (вес) wi	Оценка, установленная экспериментом ri

2. Установить веса показателей wi (?wi =1).

3. Для каждого показателя установить конкретную численную оценку ri от 0 до 1, исходя из следующего:

§ 0 – свойство в ПП присутствует, но качество его неприемлемо;

§ 0.5 — 1 – свойство в ПП присутствует и обладает приемлемым качеством;

§ 1 – свойство в ПП присутствует и обладает очень высоким качеством.

§ Возможно, присвоение промежуточных значений в соответствии с мнением оценивающего лица относительно полезности того или иного свойства ПП.

$$K = \frac{\sum w_i \cdot r_i}{общее\_количество\_показателей}$$

Результатом выполнения данной работы является отчет

Задание 2.1.3.3. Лабораторная работа № 5. «Инспекция программного кода на предмет соответствия стандартам кодирования»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.2., ПК2.3., ПК2.4., ПК2.5.*

Цель: научиться выполнять реорганизацию программного кода на основании шаблонов рефакторинга.

Задание №1

Выполнить анализ программного кода для разрабатываемого ПО и модульных тестов с целью плохо организованного кода.

Задание №2

Используя шаблоны рефакторинга, выполнить реорганизацию программного кода разрабатываемого ПО.

Задание №3

Выполнить описание произведенных операций рефакторинга (было-стало-шаблон рефакторинга).

Задание №4

В случае необходимости скорректировать проектную документацию.

Задание №5

Сделать выводы по результатам выполнения работ.

Класс Main

Было:

```
пакетная игра;
импорт игры. Персонажи. *;
импорт игры. Персонажи. Персонажи;
импорт игры. Энергетика. Энергетика;
импорт игры. Энергетика. Освещение;
импорт игры. Уровни. Блок;
импорт игры. Уровни. Уровень;
импортировать game.Levels.Level_data;
импорт игры. Weapon.Bullet;
импорт игры. Оружие. Оружие;
import javafx.animation.AnimationTimer;
импорт javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.scene.input.KeyCode;
import javafx.scene.layout.Pane;
import javafx.stage.Stage;
import java.io.DataInputStream;
import java.io.FileInputStream;
импорт java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
публичный класс Main расширяет приложение {
public static ArrayList <Block> blocks = new ArrayList <> ();
public static ArrayList <Bullet> bullets = new ArrayList <> ();
public static ArrayList <Bullet> врагаBullets = новый ArrayList <> ();
public static ArrayList <EnemyBase> враги = новый ArrayList <> ();
```

```
static HashMap <KeyCode, Boolean> keys = new HashMap <> ();
публичная статическая сцена этапа;
публичная статическая сцена;
public static Pane gameRoot = new Pane ();
public static Pane appRoot = new Pane ();
публичное статическое меню;
публичный статический Персонаж букера;
публичная статическая HUD HUD;
статическое оружие общего назначения;
публичная статика елизавета елизавета;
статический VendingMachine vendingMachine;
статическое учебное пособие;
частные статические CutScenes cutScene;
публичная статика Энергетика энергетика;
публичная статическая молния молнии;
public static int levelNumber;
уровень статического уровня;
public static AnimationTimer timer = new AnimationTimer () {
@Override
public void handle (давно) {
Обновить();
}
};
приватное статическое void update () {
для (EnemyBase враг: враги) {
enemy.update ();
if (врага.getDelete ()) {
enemies.remove (враг);
перемена;
}
}
Bullet.update ();
Controller.update ();
booker.update ();
if (! energetic.getName (). equals (""))
energetic.update ();
if (levelNumber> 0)
elizabeth.update ();
если (молния! = ноль) {
lightning.update ();
if (lightning.getDelete ())
молния = ноль;
}
menu.update ();
hud.update ();
weapon.update ();
if (booker.getTranslateX ()> Level_data.BLOCK_SIZE * 295)
cutScene = новые CutScenes (levelNumber);
}
@Override
public void start (Stage primaryStage) выдает исключение {
stage = primaryStage;
сцена = новая сцена (appRoot, 1280, 720);
```

```
. AppRoot.getChildren () добавить (gameRoot);
уровень = новый уровень ();
try (DataInputStream dataInputStream = new DataInputStream (новый FileInputStream ("C:
/DeadShock/saves/data.dat"))) {
levelNumber = dataInputStream.readInt ();
level.createLevels (levelNumber);
level.changeImageView (levelNumber);
vendingMachine = new VendingMachine ();
букер = новый персонаж ();
booker.setMoney (dataInputStream.readInt ());
booker.setSalt (dataInputStream.readByte ());
booker.setCountLives (2);
оружие = новое оружие ();
weapon.setWeaponClip (dataInputStream.readInt ());
weapon.setBullets (dataInputStream.readInt ());
hud = новый HUD ();
Елизавета = новая Елизавета ();
энергичный = новый Энергетический ();
} catch (IOException e) {
levelNumber = 0;
level.createLevels (levelNumber);
vendingMachine = new VendingMachine ();
букер = новый персонаж ();
оружие = новое оружие ();
hud = новый HUD ();
энергичный = новый Энергетический ();
tutorial = new Tutorial (levelNumber);
}
switch (levelNumber) {
случай 0:
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 117, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 127, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 148, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 161, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 171, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 185, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 204, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 215, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 228, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 233, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 243, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 252, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 262, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 277, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 280, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 286, 200));
перемена;
Дело 1:
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 57, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 67, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 74, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 87, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 104, 150));
```

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 117, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 133, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 156, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 177, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 193, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 201, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 216, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 224, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 246, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 260, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 277, 100));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 34,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 36,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 60,
Level_data.BLOCK_SIZE * 9));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 61,
Level_data.BLOCK_SIZE * 9));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 106,
Level_data.BLOCK_SIZE * 7));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 107,
Level_data.BLOCK_SIZE * 7));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 168,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 170,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 196,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 197,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 232,
Level_data.BLOCK_SIZE * 8));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 233,
Level_data.BLOCK_SIZE * 8));
перемена;
}
меню = новое меню ();
appRoot.getChildren () добавить (menu.menuBox).
booker.translateXProperty (). addListener (((наблюдаемый, oldValue, newValue) -> {
int offset = newValue.intValue ();
if (offset> 600 && offset <gameRoot.getWidth () - 680) {
gameRoot.setLayoutX (- (смещение - 600));
level.getBackground (). setLayoutX ((смещение - 600) / 1,5);
}
если (смещение <= 100)
level.getBackground () setLayoutX (0).
}));
vendingMachine.createButtons ();
stage.getIcons (). add (новое изображение ("файл: / C: /DeadShock/images/icon.jpg"));
stage.setTitle ("DeadShock");
stage.setResizable (ложь);
stage.setWidth (scene.getWidth ());

```
stage.setHeight (scene.getHeight ());
stage.setScene (сцены);
stage.show ();
timer.start ();
}
public static void main (String [] args) {
запуск (арг);
}
}
```
Стало:
```
пакетная игра;
импорт игры. Персонажи. *;
импорт игры. Персонажи. Персонажи;
импорт игры. Энергетика. Энергетика;
импорт игры. Энергетика. Освещение;
импорт игры. Уровни. Блок;
импорт игры. Уровни. Уровень;
импортировать game.Levels.Level_data;
импорт игры. Weapon.Bullet;
импорт игры. Оружие. Оружие;
import javafx.animation.AnimationTimer;
импорт javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.scene.input.KeyCode;
import javafx.scene.layout.Pane;
import javafx.stage.Stage;
import java.io.DataInputStream;
import java.io.FileInputStream;
импорт java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
публичный класс Main расширяет приложение {
public static ArrayList <Block> blocks = new ArrayList <> ();
public static ArrayList <Bullet> bullets = new ArrayList <> ();
public static ArrayList <Bullet> врагаBullets = новый ArrayList <> ();
public static ArrayList <EnemyBase> враги = новый ArrayList <> ();
static HashMap <KeyCode, Boolean> keys = new HashMap <> ();
публичная статическая сцена этапа;
публичная статическая сцена;
public static Pane gameRoot = new Pane ();
public static Pane appRoot = new Pane ();
публичное статическое меню;
публичный статический Персонаж букера;
публичная статическая HUD HUD;
статическое оружие общего назначения;
публичная статика елизавета елизавета;
статический VendingMachine vendingMachine;
статическое учебное пособие;
частные статические CutScenes cutScene;
публичная статика Энергетика энергетика;
публичная статическая молния молнии;
public static int levelNumber;
```

```
уровень статического уровня;
public static AnimationTimer timer = new AnimationTimer () {
@Override
public void handle (давно) {
Обновить();
}
};
private void initContent () {
. AppRoot.getChildren () добавить (gameRoot);
уровень = новый уровень ();
try (DataInputStream dataInputStream = new DataInputStream (новый FileInputStream ("C:
/DeadShock/saves/data.dat"))) {
levelNumber = dataInputStream.readInt ();
level.createLevels (levelNumber);
level.changeImageView (levelNumber);
vendingMachine = new VendingMachine ();
букер = новый персонаж ();
booker.setMoney (dataInputStream.readInt ());
booker.setSalt (dataInputStream.readByte ());
booker.setCountLives (2);
оружие = новое оружие ();
weapon.setWeaponClip (dataInputStream.readInt ());
weapon.setBullets (dataInputStream.readInt ());
hud = новый HUD ();
Елизавета = новая Елизавета ();
энергичный = новый Энергетический ();
} catch (IOException e) {
levelNumber = 0;
level.createLevels (levelNumber);
vendingMachine = new VendingMachine ();
букер = новый персонаж ();
оружие = новое оружие ();
hud = новый HUD ();
энергичный = новый Энергетический ();
tutorial = new Tutorial (levelNumber);
}
createEnemies ();
меню = новое меню ();
appRoot.getChildren () добавить (menu.menuBox).
booker.translateXProperty (). addListener (((наблюдаемый, oldValue, newValue) -> {
int offset = newValue.intValue ();
if (offset> 600 && offset <gameRoot.getWidth () - 680) {
gameRoot.setLayoutX (- (смещение - 600));
level.getBackground (). setLayoutX ((смещение - 600) / 1,5);
}
если (смещение <= 100)
level.getBackground () setLayoutX (0).
}));
vendingMachine.createButtons ();
}
public static void createEnemies () {
switch (levelNumber) {
случай 0:
```

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 117, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 127, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 148, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 161, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 171, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 185, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 204, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 215, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 228, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 233, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 243, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 252, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 262, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 277, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 280, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 286, 200));
перемена;
Дело 1:
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 57, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 67, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 74, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 87, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 104, 150));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 117, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 133, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 156, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 177, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 193, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 201, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 216, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 224, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 246, 200));
враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 260, 200));
враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 277, 100));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 34,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 36,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 60,
Level_data.BLOCK_SIZE * 9));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 61,
Level_data.BLOCK_SIZE * 9));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 106,
Level_data.BLOCK_SIZE * 7));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 107,
Level_data.BLOCK_SIZE * 7));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 168,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 170,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 196,
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 197,

```
Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 232,
Level_data.BLOCK_SIZE * 8));
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 233,
Level_data.BLOCK_SIZE * 8));
перемена;
}
}
приватное статическое void update () {
для (EnemyBase враг: враги) {
enemy.update ();
If (enemy.GetDelete()) {
enemies.remove(enemy);
break;
}
}
Bullet.update();
Controller.update();
booker.update();
If (!energetic.GetName().Equals(""))
energetic.update();
if (levelNumber > 0)
elizabeth.update();
if (lightning != null) {
lightning.update();
If (lightning.GetDelete())
lightning = null;
}
menu.update();
hud.update();
weapon.update();
if (booker.getTranslateX() > Level_data.BLOCK_SIZE * 295)
cutScene = new CutScenes(levelNumber);
}
@Override
public void start(Stage primaryStage) throws Exception {
stage = primaryStage;
scene = new Scene(appRoot, 1280, 720);
initContent();
stage.getIcons().add(new Image("file:/C:/DeadShock/images/icon.jpg"));
stage.setTitle("DeadShock");
stage.setResizable(false);
stage.setWidth(scene.getWidth());
stage.setHeight(scene.getHeight());
stage.setScene(scene);
stage.show();
timer.start();
}
public static void main(String[] args) {
launch(args);
}
}
```

Шаблон рефакторинга: Выделение метода(Extract Method)

Критерии оценивания:

Оценка **«отлично»** – полный объем выполненных работ, выполнены все задания лабораторной работы, продемонстрировано глубокое понимание материала самостоятельность выполнения работы, аккуратность и законченность работы.

Оценка **«хорошо»** –все задания лабораторной работы выполнены, работа оформлена с незначительными отклонениями от требований, продемонстрировано хорошее понимание материала

Оценка **«удовлетворительно»** - задания лабораторной работы выполнены с замечаниями, работа имеет существенные отклонения в оформлении, продемонстрировано базовое понимание материала.

Оценка **«неудовлетворительно»** – отсутствие полного объема работ; в работе допущены серьёзные ошибки и нарушение всех перечисленных выше требований.

3. Тестовые задания для текущего контроля
ОК01, ОК02, ОК03, ОК04, ОК05, ОК09, ПК2.1, ПК2.2, ПК2.3, ПК2.4, ПК2.5.

Цель тестового задания - контроль знаний освоения дисциплины, получение ответа от испытуемого, на основе которого может быть сделан вывод о его знаниях, представлениях из определенной области содержания дисциплины.

Задание: перечень вопросов, соответствующих содержанию дисциплины.

Инструкция: выберите один/несколько правильных ответов из предложенных

1. Прочитайте текст, выберите все правильные ответы

Какому типу тестирования принадлежат такие методы формирования тестовых наборов, как эквивалентное разбиение, анализ граничных значений, анализ причинно-следственных связей и предположение об ошибке?

А) функциональному;

Б) модульному;

В) системному;

Г) интеграционному.

Ответ:

2. Прочитайте текст, выберите все правильные ответы

На каком этапе разработки ПО руководитель и команда проекта определяют масштаб и цели проекта?

А) анализ;

Б) планирование;

В) проектирование;

Г) разработка.

Ответ:

3. Прочитайте текст и установите соответствие

Сопоставьте названия и краткое содержание требований к программным модулям.

К каждой позиции, данной в левом столбце, подберите соответствующую позицию из правого столбца:

А	результат работы данного фрагмента программы не зависит от работы других модулей	1	Один вход и один выход
Б	на входе программный модуль получает	2	Логическая независимость

	определенный набор исходных данных, выполняет их обработку и возвращает один набор выходных данных		
В	обмен информацией между отдельными модулями должен быть минимален	3	Функциональная завершенность
Г	модуль выполняет набор определенных операций для реализации каждой отдельной функции, достаточных для завершения начатой обработки данных	4	Слабые информационные связи с другими программными модулями

Запишите выбранные цифры под соответствующими буквами:

А	Б	В	Г

4. Прочитайте текст и установите последовательность

Установите правильную последовательность этапов создания документации на проект ПО:

А) сбор и документирование архитектурной информации в виде группы представлений и специального блока с общей информацией для всех представлений;

Б) определение потребностей заинтересованных сторон;

В) подготовка архитектурной документации в вид, пригодный для той или иной заинтересованной стороны;

Г) проверка того, что созданная документация удовлетворяет требованиям заинтересованных сторон.

Запишите соответствующую последовательность цифр слева направо

5. Как называется период времени, который начинается с момента принятия решения о необходимости создания программного продукта и заканчивается в момент его полного изъятия из эксплуатации?

6. Какой тип прототипа программы показывает только внешнюю сторону продукта и демонстрирует наличие каких-либо функциональных возможностей без их реализации?

7. Какой командой можно подключить к проекту программы на языке C# пространство имен, в котором находятся отладочные классы Debug и Trace?

8. Дополните предложение: « … тестирование называют тестированием по «маршрутам», так как в этом случае тестовые наборы формируют путем анализа маршрутов (последовательностей операторов программы, которые выполняются при конкретном варианте исходных данных), предусмотренных алгоритмом».

9. Дополните предложение: «Объектно-ориентированный подход предполагает, что при разработке программы должны быть определены …., используемых в программе, и построены их описания, затем созданы экземпляры необходимых объектов и определено взаимодействие между ними».

10. Прочитайте текст, выберите все правильные ответы

Какая качественная характеристика ПО определяется частотой отказов, вызванных наличием ошибок в программном обеспечении?

а) стабильность;

б) гибкость;

в) мобильность;

г) модифицируемость.

Ответ:

11. Прочитайте текст, выберите все правильные ответы

В какой строке предложенного фрагмента программного кода на C# возникнет исключение типа FormatException?

```
try
{
Console.WriteLine("Введите число: ");
string str = Console.ReadLine();
int number = Convert.ToInt32(str);
number*=10;
}
catch (FormatException)
{
 Console.WriteLine("Ошибка! Вы ввели не число");
}
Console.WriteLine();
Console.WriteLine("Результат вычислений: "+ number);
Console.ReadKey(true);
```

а) number*=10;

б) string str = Console.ReadLine();

в) int number = Convert.ToInt32(str);

г) catch (FormatException).

Ответ:

12. Прочитайте текст и установите соответствие

Сопоставьте характеристики и названия групп операций интеграции данных

К каждой позиции, данной в левом столбце, подберите соответствующую позицию из правого столбца:

А	комбинация различных представлений одной и той же сущности реального мира в единое представление	1	Трансформация данных
Б	преобразование из исходной схемы (схемы коллекции - источника данных) в целевую (единую интегрированную схему)	2	Разрешение сущностей
В	выделение и связывание информации об одной и той же сущности реального мира из разных коллекций данных.	3	Слияние сущностей

Запишите выбранные цифры под соответствующими буквами:

А	Б	В

12. Прочитайте текст и установите соответствие

Сопоставьте операторы обработки исключительных ситуаций в программном коде и названия языков программирования, к которым они относятся.

К каждой позиции, данной в левом столбце, подберите соответствующую позицию из правого столбца:

А	точное и, по возможности, полное описание назначения программы (программной системы), её функций, требований на входные данные, её результатов, особенностей функционирования и нефункциональных свойств, состава, структуры программы и её интерфейсов.	1	Верификация
Б	это процесс проверки, что продукт соответствует определённым требованиям и спецификациям на каждом этапе разработки	2	Валидация
В	это процесс проверки, что продукт соответствует ожиданиям и потребностям пользователей	3	Спецификация программы

Запишите выбранные цифры под соответствующими буквами:

А	Б	В

14. Прочитайте текст и установите последовательность

Установите правильную последовательность основных шагов процедуры регрессионного тестирования ПО.

А) Установите критерии входа и точку входа (первой команды программы);

Б) Распознайте изменения исходного программного кода;

В) Выберите точку выхода (последней команды программы);

Г) Установите приоритет этих изменений и требований к программному продукту;

Д) Составьте план тестов

15. Прочитайте текст, выберите все правильные ответы

Как называется фиксированное состояние реализации программного продукта на конкретную дату, выполненной в соответствии с техническим заданием заказчика, которому присваивается символическое обозначение в виде номера?

А) версия;

Б) релиз;

В) сборка;

Г) вариант.

Ответ:

16. Прочитайте текст и установите соответствие

Сопоставьте операторы обработки исключительных ситуаций в программном коде и названия языков программирования, к которым они относятся.

К каждой позиции, данной в левом столбце, подберите соответствующую позицию из правого столбца:

А	C#;	1	try:
			//операторы
			except:
			//операторы

Б	1C;	2	try
			{
			//операторы
			}
			catch
			{
			//операторы
			}

В	Python
	.
		3	Попытка
			//операторы попытки
			Исключение
			//операторы исключения.
			КонецПопытки

Запишите выбранные цифры под соответствующими буквами:

А	Б	В

17. Прочитайте текст и установите соответствие

Сопоставьте названия и формулировки типов программных ошибок

К каждой позиции, данной в левом столбце, подберите соответствующую позицию из правого столбца:

А	наблюдаемое нарушение требований, проявляющееся при каком-то реальном сценарии работы ПО	1	defect
Б	ошибка в коде программы, вызывающая нарушения требований при работе, то место, которое надо исправить	2	failure
В	самое общее нарушение каких-либо требований или ожиданий, не обязательно проявляющееся вовне (например, нарушения стандартов кодирования, недостаточная гибкость системы и пр.)	3	fault

Запишите выбранные цифры под соответствующими буквами:

А	Б	В

18. Прочитайте текст и установите последовательность

Установите правильную последовательность характеристик номера версии программного обеспечения:

А) минорная версия (minor subversion, промежуточная версия) программного обеспечения;

Б) мажорная версия (major version) программного обеспечения;
В) сборка (build) программного обеспечения;
Г) релиз (release) программного обеспечения.

19.. Какое ключевое слово должно присутствовать в сигнатуре метода на языке C# для того, чтобы функция принадлежала всему классу, а не конкретным объектам класса?

20. Какие существуют общие виды систем контроля версий;

Ключ к тестовым заданиям

№ задания	Верный ответ	Критерии
1	А	1б – полное правильное соответствие 0 б – остальные случаи
2	Б	1б – полное правильное соответствие 0 б – остальные случаи
3	Б1А2Г3В4	1б – полное правильное соответствие 0 б – остальные случаи
4	БАГВ	1б – полное правильное соответствие 0 б – остальные случаи
5	Жизненный цикл программного обеспечения	1б – полное правильное соответствие 0 б – остальные случаи
6	горизонтальный	1б – полное правильное соответствие 0 б – остальные случаи
7	Using Diagnostics	1б – полное правильное соответствие 0 б – остальные случаи
8	структурное	1б – полное правильное соответствие 0 б – остальные случаи
9	Классы объектов	1б – полное правильное соответствие 0 б – остальные случаи
10	А	1б – полное правильное соответствие 0 б – остальные случаи
11	В	1б – полное правильное соответствие 0 б – остальные случаи
12	Б1В2А3	1б – полное правильное соответствие 0 б – остальные случаи
13	Б1В2А3	1б – полное правильное соответствие 0 б – остальные случаи
14	БГАВД	1б – полное правильное соответствие 0 б – остальные случаи
15	А	1б – полное правильное соответствие 0 б – остальные случаи
16	В1А2Б3	1б – полное правильное соответствие 0 б – остальные случаи
17	В1А2Б3	1б – полное правильное соответствие 0 б – остальные случаи
18	БАГВ	1б – полное правильное соответствие 0 б – остальные случаи
19	static	1б – полное правильное соответствие 0 б – остальные случаи
20	Централизованные и	1б – полное правильное соответствие

	распределенные	0 б – остальные случаи

Критерии оценки:

соответствие ответов обучающихся ключу теста

Оценка «**отлично**» - если обучающийся правильно выполнил от 80% до 100% тестовых заданий в отведенное время

Оценка «**хорошо**» - если обучающийся правильно выполнил от 60% до 80% тестовых заданий в отведенное время

Оценка «**удовлетворительно**» - если обучающийся правильно выполнил от 40% до 60% тестовых заданий в отведенное время

Оценка «**неудовлетворительно**» ставится в случае выполнения менее 40% тестовых заданий

Время выполнения: 35-40 минут

4. Задания для оценки освоения дисциплины МДК.2.2 Инструментальные средства разработки программного обеспечения

Тема 2.2.1. Инструментальные средства разработки программного обеспечения

Задание 2.1.2.1. Практическая работа № 10 «Разработка структуры проекта»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.4., ПК2.5.*

Цель: 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.

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?

Контрольные вопросы

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

Задание 2.1.2.2. Практическая работа № 11 «Разработка модульной структуры проекта»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.4., ПК2.5.*

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 становится 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. Проанализируйте информационные потоки вашего факультета и спроектируйт компьютерную систему их обработки.

Задание 2.1.2.3. Практическая работа № 12 «Разработка перечня артефактов и протоколов проекта»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.4.,*

ПК2.5.

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 утверждается техническое 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. Содержание отчета:

− тема лабораторной работы;
− цель лабораторной работы;
− ответы на контрольные вопросы;
− задание на лабораторную работу;
− разработанные спецификации процессов;
− словарь терминов;
− диаграммы переходов состояний;
− диаграммы потоков с детализацией;
− выводы по проделанной работе.

Контрольные вопросы

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?

Критерии оценивания:

Оценка «**отлично**» – полный объем выполненных работ, выполнены все задания лабораторной работы, продемонстрировано глубокое понимание материала самостоятельность выполнения работы, аккуратность и законченность работы.

Оценка «**хорошо**» –все задания лабораторной работы выполнены, работа оформлена с незначительными отклонениями от требований, продемонстрировано хорошее понимание материала

Оценка «удовлетворительно» - задания лабораторной работы выполнены с замечаниями, работа имеет существенные отклонения в оформлении, продемонстрировано базовое понимание материала.

Оценка «неудовлетворительно» – отсутствие полного объема работ; в работе допущены серьёзные ошибки и нарушение всех перечисленных выше требований.

Задание 2.1.2.4. Лабораторная работа № 6 «Настройка работы системы контроля

версий (типов импортируемых файлов, путей, фильтров и др. параметров импорта в репозиторий)».

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.4., ПК2.5.*

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 ведѐтся работа с большим количеством непрерывно изменяющихся электронных документов, в частности, они всѐ чаще применяются в САПР, обычно, в составе систем управления данными об изделии (PDM). Управление версиями используется в инструментах конфигурационного управления (Software Configuration Management Tools).

Распространѐнные 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 обновлѐнная 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 определѐнного 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 распределѐнных 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.

1. Настроить подключение к репозиторию
2. Скачать проект
3. Добавить свой класс к проекту
4. Внести изменения к 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 2.1.2.5. Лабораторная работа № 7 «Разработка и интеграция модулей проекта».

Проверяемые результаты обучения: *ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.4., ПК2.5.*

Цель: 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

Задание 2.1.2.6. Лабораторная работа № 8 «Отладка отдельных модулей программного проекта. Организация обработки исключений. Применение отладочных классов в проекте»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.4., ПК2.5.*

Цель: Изучить основные подходы к проектированию тестов.

Теоретическая часть.

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).

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.

	Вид функции Y(X),		Вид функции Y(X),

№ п/п	характерные точки	№ п/п	характерные точки
	Локальный минимум функции $x^2 + 0.5 - \sin(3^* x)$	6	Точка (точки), в которой функция $\frac{2x^3 - 3}{x^2 + 1}$ равна 10.
	Точка (точки), для которой $x^2 + 5 - 1 = 5$.	7	Локальный минимум функции $3(x+4)^2 - \sin(x+15)$
	Пересечение графиков функций $\frac{}{x^2 + 1(3+x)}$ и $\frac{5}{x^2 - 7}$.	8	Точка (точки), в которой функция $\frac{2x^2}{\sin(x-1)} - 1$ равна -500.
	Все нули функции $x_2 - \sin(3x)$.	9	Локальный максимум функции $\frac{2x + 15}{3 + x^2}$
	Локальный минимум функции $\frac{5}{3x - x^2 - 3}$	0	Пересечение графиков функций $(x^2+1)\sin(3+x)$ и $\frac{x+5}{x^2+1}$.
	Точка (точки), в которой функция $x_2 - e_x$ равна -10.	1	Локальный минимум функции $x^2 - 3x + 15$
	Пересечение графиков функций $100\sin(1+x)$ и $x^3 + 5$.	2	Все нули функции $x^2 + 0.5 - \sin(3x)$.
	Локальный минимум функции $e^x - x$	3	Локальный максимум функции $200x - e^x$
	Все нули функции $\frac{x^2 - 15}{\ln x^2 + 3x - 7}$.	4	Все нули функции $x - e^x \cos(x)$.

0	Локальный максимум функции $\frac{2x^2+1}{3+x^2}$	**5**	Точка (точки), в которой функция $\frac{x^2+5}{x_2+7}$ равна 15.
1	Пересечение графиков функций $10\cos(x)$ и $x^3/3+5$.	**6**	Все нули функции $\frac{x^2-15}{2x_2+3x-7}$.
2	Локальный максимум функции $-(x-2)^2+\cos(x)$	**7**	Пересечение графиков функций $3x^2-15$ и $10\cos(x)+7$.
3	Точка (точки), в которой функция $3x^2-\frac{12x-5}{x_2+1}$ равна 5.	**8**	Локальный минимум функции $-100x+e^x$
4	Все нули функции $\frac{x+5}{x_2-7}$.	**9**	Локальный максимум функции $-\frac{x}{x^2+3}$

5	Локальный максимум функции $-e^{-x}-100x$	**0**	Пересечение графиков функций $4(x-5)^2-15$ и $\frac{100-7}{x^2+2}$.

Критерии оценивания:

Оценка **«отлично»** – полный объем выполненных работ, выполнены все задания лабораторной работы, продемонстрировано глубокое понимание материала самостоятельность выполнения работы, аккуратность и законченность работы.

Оценка **«хорошо»** –все задания лабораторной работы выполнены, работа оформлена с незначительными отклонениями от требований, продемонстрировано хорошее понимание материала

Оценка **«удовлетворительно»** - задания лабораторной работы выполнены с замечаниями, работа имеет существенные отклонения в оформлении, продемонстрировано базовое понимание материала.

Оценка **«неудовлетворительно»** – отсутствие полного объема работ; в работе допущены серьёзные ошибки и нарушение всех перечисленных выше требований.

Тема 2.2.2. Инструментарий тестирования и анализа качества программных средств

Задание 2.2.2.1. Практическая работа № 13 «Отладка проекта»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.4., ПК2.5.*

Цель: Получение практических навыков тестирования и отладки программы.

Теоретические основы.

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 ее 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.

Задание 2.2.2.2. Практическая работа № 14 «Инспекция кода модулей проекта»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.4., ПК2.5.*

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

Теоретические сведения

Термин «интеграция» относится к такой операции в процессе разработки ПО, при которой вы объединяете отдельные программные компоненты в единую систему. В небольших проектах интеграция может занять одно утро и заключаться в объединении горстки классов. В больших — могут потребоваться недели или месяцы, чтобы связать воедино весь набор программ. Независимо от размера задач в них применяются одни и те же принципы.

Тема интеграции тесно переплетается с вопросом последовательности конструирования. Порядок, в котором вы создаете классы или компоненты, влияет на порядок их интеграции: вы не можете интегрировать то, что еще не было создано. Последовательности интеграции и конструирования имеют большое значение.

Поскольку интеграция выполняется после того, как разработчик завершил модульное тестирование, и одновременно с системным тестированием, ее иногда считают операцией, относящейся к тестированию. Однако она достаточно сложна, и поэтому ее следует рассматривать как независимый вид деятельности.

Аккуратная интеграция обеспечивает:

- упрощенную диагностику дефектов;
- меньшее число ошибок;
- меньшее количество «лесов»;
- раннее создание первой работающей версии продукта;
- уменьшение общего времени разработки;
- лучшие отношения с заказчиком;
- улучшение морального климата;
- увеличение шансов завершения проекта;
- более надежные оценки графика проекта;
- более аккуратные отчеты о состоянии;
- лучшее качество кода;
- меньшее количество документации.

Интеграция программ выполняется посредством поэтапного или инкрементного подхода.

Поэтапная интеграция состоит из этапов, перечисленных ниже:

1. «Модульная разработка»: проектирование, кодирование, тестирование и отладка каждого класса.
2. «Системная интеграция»: объединение классов в одну огромную систему.

3. «Системная дезинтеграция»: тестирование и отладка всей системы. Проблема поэтапной интеграции в том, что, когда классы в системе впервые соединяются вместе, неизбежно возникают новые проблемы и их причины могут быть в чем угодно. Поскольку у вас масса классов, которые никогда раньше не работали вместе, виновником может быть плохо протестированный класс, ошибка в интерфейсе между двумя классами или ошибка, вызванная взаимодействием двух классов. Все классы находятся под подозрением.

Неопределенность местонахождения любой из проблем сочетается с тем фактом, что все эти проблемы вдруг проявляют себя одновременно. Это заставляет вас иметь дело не только с проблемами, вызванными взаимодействием классов, но и другими ошибками, которые трудно диагностировать, так как они взаимодействуют.

Поэтому поэтапную интеграцию называют еще «интеграцией большого взрыва»

Поэтапную интеграцию нельзя начинать до начала последних стадий проекта, когда будут разработаны и протестированы все классы. Когда классы, наконец, будут объединены и проявится большое число ошибок, программисты тут же ударятся в паническую отладку вместо методического определения и исправления ошибок.

Для небольших программ - нет, а для крошечных - поэтапная интеграция может быть наилучшим подходом. Если программа состоит из двух-трех классов, поэтапная интеграция может сэкономить ваше время, если вам повезет. Но в большинстве случаев инкрементный подход будет лучше.

При инкрементной интеграции вы пишете и тестируете маленькие участки программы, а затем комбинируете эти кусочки друг с другом по одному. При таком подходе - по одному элементу за раз - вы выполняете перечисленные далее действия:

1. Разрабатываете небольшую, функциональную часть системы. Это может быть наименьшая функциональная часть, самая сложная часть, основная часть или их комбинация. Тщательно тестируете и отлаживаете ее. Она послужит скелетом, на котором будут наращиваться мускулы, нервы и кожа, составляющие остальные части системы.

2. Проектируете, кодируете, тестируете и отлаживаете класс.

3. Прикрепляете новый класс к скелету. Тестируете и отлаживаете соединение скелета и нового класса. Убеждаетесь, что эта комбинация работает, прежде чем переходить к добавлению нового класса. Если дело сделано, повторяете процесс, начиная с п. 2.

Инкрементный подход имеет массу преимуществ перед традиционным поэтапным подходом независимо от того, какую инкрементную стратегию вы используете:

Ошибки можно легко обнаружить Когда во время инкрементной интеграции возникает новая проблема, то очевидно, что к этому причастен новый класс. Либо его интерфейс с остальной частью программы неправилен, либо его взаимодействие с ранее интегрированными классами приводит к ошибке. В любом случае вы точно знаете, где искать проблему.

В таком проекте система раньше становится работоспособной Когда код интегрирован и способен выполняться, даже если система еще не пригодна к использованию, это выглядит так, будто это скоро произойдет. При инкрементной интеграции программисты раньше видят результаты своей работы, поэтому их моральное состояние лучше, чем в том случае, когда они подозревают, что их проект может никогда не сделать первый вдох.

Вы получаете улучшенный мониторинг состояния При частой интеграции реализованная и нереализованная функциональность видна с первого взгляда. Менеджеры будут иметь лучшее представление о состоянии проекта, видя, что 50% системы уже работает, а не слыша, что кодирование «завершено на 99%».

Вы улучшите отношения с заказчиком Если частая интеграция влияет на моральное состояние разработчиков, то она также оказывает влияние и на моральное состояние заказчика. Клиенты любят видеть признаки прогресса, а инкрементная интеграция предоставляет им такую возможность достаточно часто.

Системные модули тестируются гораздо полнее Интеграция начинается на ранних стадиях проекта. Вы интегрируете каждый класс по мере его готовности, а не ожидая одного внушительного мероприятия по интеграции в конце разработки. Программист тестирует классы в обоих случаях, но в качестве элемента общей системы они используются гораздо чаще при

инкрементной, чем при поэтапной интеграции.

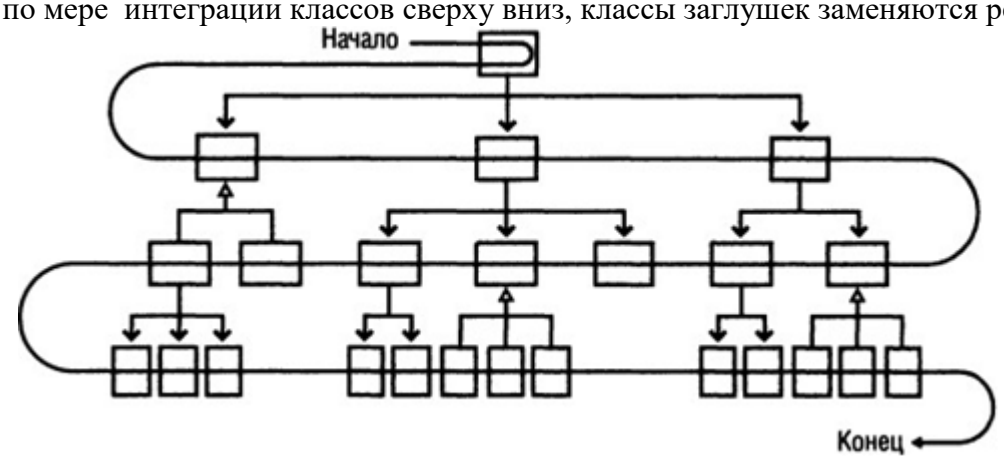
Вы можете создать систему за более короткое время Если интеграция тщательно спланирована, вы можете проектировать одну часть системы в то время, когда другая часть уже кодируется. Это не уменьшает общее число человеко-часов, требуемых для полного проектирования и кодирования, но позволяет выполнять часть работ параллельно, что является преимуществом в тех случаях, когда время имеет критическое значение.

При поэтапной интеграции вам не нужно планировать порядок создания компонентов проекта. Все компоненты интегрируются одновременно, поэтому вы можете разрабатывать их в любом порядке - главное, чтобы они все были готовы к часу X.

При инкрементной интеграции вы должны планировать более аккуратно. Большинство систем требует интеграции некоторых компонентов перед интеграцией других. Так что планирование интеграции влияет на планирование конструирования — порядок, в котором конструируются компоненты, должен обеспечивать порядок, в котором они будут интегрироваться.

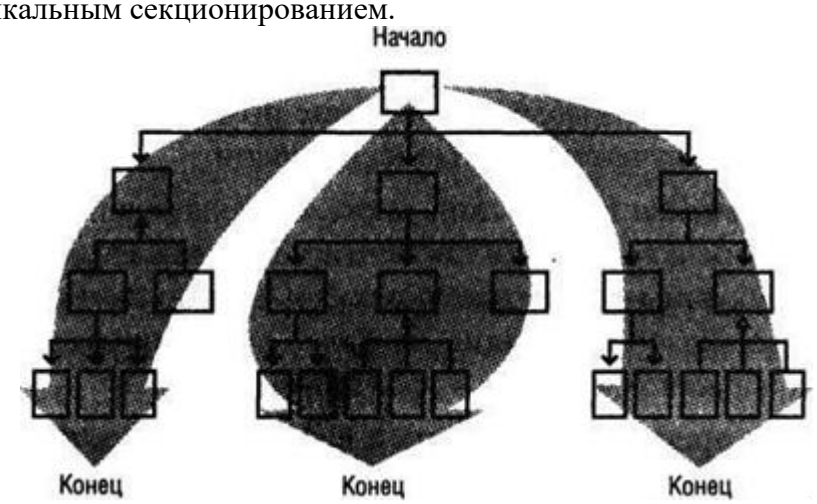
Нисходящая интеграция

При нисходящей интеграции класс на вершине иерархии пишется и интегрируется первым. Вершина иерархии — это главное окно, управляющий цикл приложения, объект, содержащий метод main() в программе на Java, функция WinMain() в программировании для Microsoft Windows или аналогичные. Для работы этого верхнего класса пишутся заглушки. Затем, по мере интеграции классов сверху вниз, классы заглушек заменяются реальными.

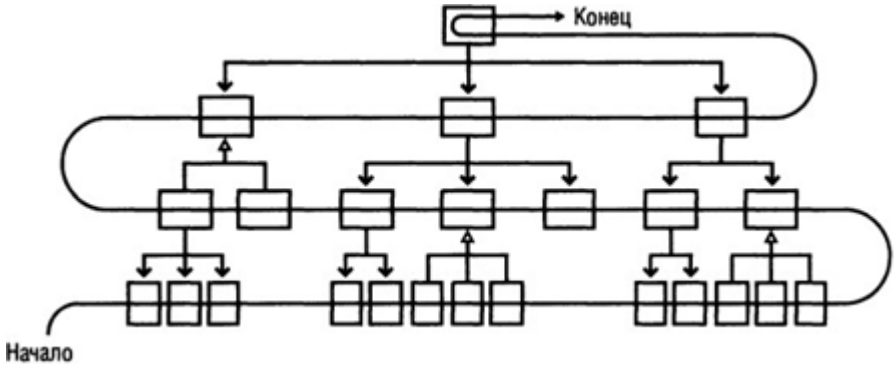


При нисходящей интеграции вы создаете те классы, которые находятся на вершине иерархии, первыми, а те, что внизу, — последними.

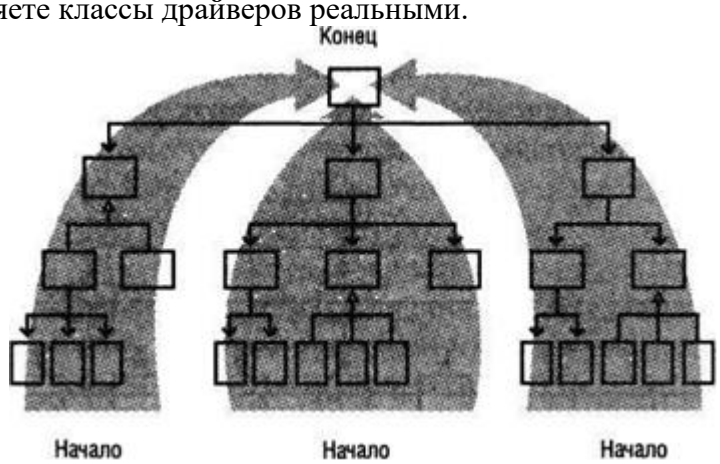
Хорошей альтернативой нисходящей интеграции в чистом виде может стать подход с вертикальным секционированием.



При этом систему реализуют сверху вниз по частям, возможно, по очереди выделяя функциональные области и переходя от одной к другой.

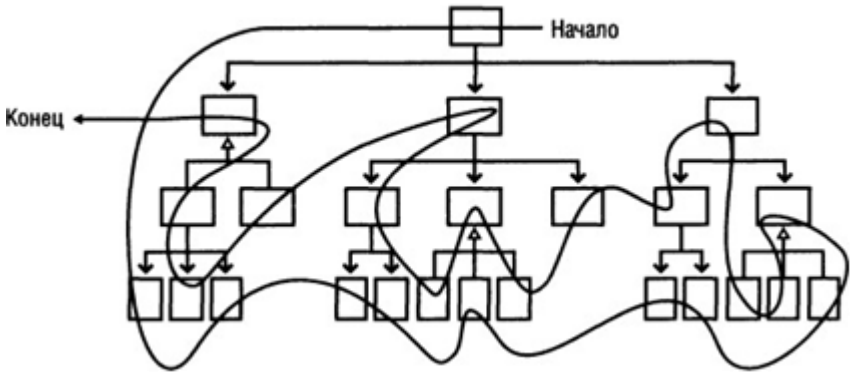


При восходящей интеграции вы пишете и интегрируете сначала классы, находящиеся в низу иерархии. Добавление низкоуровневых классов по одному, а не всех одновременно — вот что делает восходящую интеграцию инкрементной стратегией. Сначала вы пишете тестовые драйверы для выполнения низкоуровневых классов, а затем добавляете эти классы к тестовым драйверам, пристраивая их по мере готовности. Добавляя класс более высокого уровня, вы заменяете классы драйверов реальными.



Как и нисходящую, восходящую интеграцию в чистом виде используют редко — вместо нее можно применять гибридный подход, реализующий секционную интеграцию. Сэндвич-интеграция

Проблемы с нисходящей и восходящей интеграциями в чистом виде привели к тому, что некоторые эксперты стали рекомендовать сэндвич-подход.



Сначала вы объединяете высокоуровневые классы бизнес-объектов на вершине иерархии. Затем добавляете классы, взаимодействующие с аппаратной частью, и широко используемые вспомогательные классы в низу иерархии.

Напоследок вы оставляете классы среднего уровня.

Риск-ориентированная интеграция

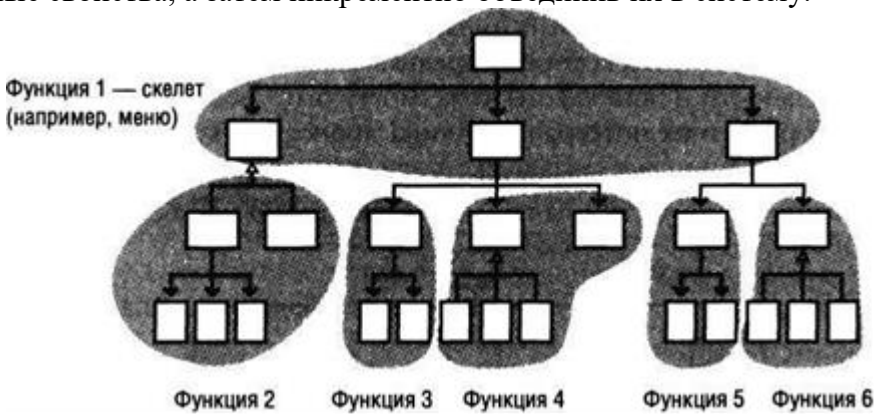
Риск-ориентированную интеграцию, которую также называют «интеграцией, начиная с самых сложных частей» (hard part first integration), похожа на сэндвич-интеграцию тем, что пытается избежать проблем, присущих нисходящей или восходящей интеграциям в чистом виде. Кроме того, в ней также есть тенденция к объединению классов верхнего и нижнего уровней в первую очередь, оставляя классы среднего уровня напоследок. Однако суть в другом.

При риск-ориентированной интеграции вы определяете степень риска, связанную с каждым классом. Вы решаете, какие части системы будут самыми трудными, и реализуете их первыми.

Функционально-ориентированная интеграция

Еще один поход - интеграция одной функции в каждый момент времени. Под «функцией» понимается не нечто расплывчатое, а какое-нибудь поддающееся определению свойство системы, в которой выполняется интеграция.

Когда интегрируемая функция превышает по размерам отдельный класс, то «единица приращения» инкрементной интеграции становится больше отдельного класса. Это немного снижает преимущество инкрементного подхода в том плане, что уменьшает вашу уверенность об источнике новых ошибок. Однако если вы тщательно тестировали классы, реализующие эту функцию, перед интеграцией, то это лишь небольшой недостаток. Вы можете использовать стратегии инкрементной интеграции рекурсивно, сформировав сначала из небольших кусков отдельные свойства, а затем инкрементно объединив их в систему.



Обычно процесс начинается с формирования скелета, поскольку он способен поддерживать остальную функциональность. В интерактивной системе такой изначальной опцией может стать система интерактивного меню. Вы можете прикреплять остальную функциональность к той опции, которую интегрировали первой.

Т-образная интеграция

Последний подход, который часто упоминается в связи с проблемами нисходящей и восходящей методик, называется «Т-образной интеграцией». При таком подходе выбирается некоторый вертикальный слой, который разрабатывается и интегрируется раньше других. Этот слой должен проходить сквозь всю систему от начала до конца и позволять выявлять основные проблемы в допущениях, сделанных при проектировании системы. Реализовав этот вертикальный участок (и устранив все связанные с этим проблемы), можно разрабатывать основную канву системы (например, системное меню для настольного приложения). Этот подход часто комбинируют с риск-ориентированной и функционально-ориентированной интеграциями.

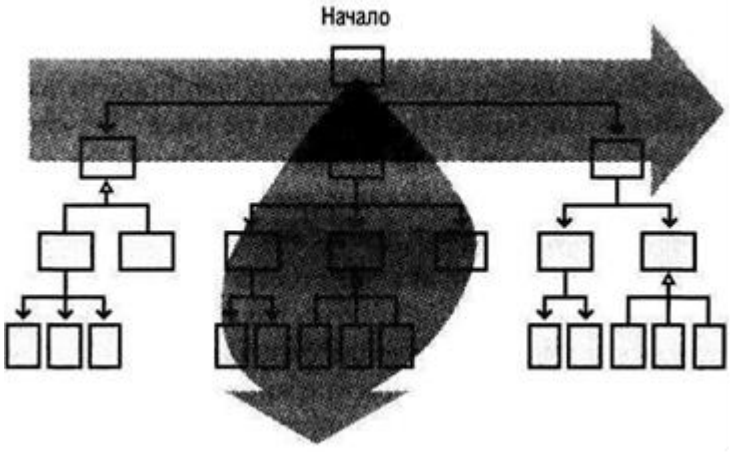


Рис. 9 Т-образная интеграция

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.

Задача. Задан двумерный массив размерности n□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.

Контрольные вопросы.

1. Значение фазы интеграции программных модулей.
2. Подходы к интегрированию программных модулей. Эффективность и оптимизация программ.

Задание 2.2.2.3. Практическая работа № 15 «Тестирование интерфейса пользователя средствами инструментальной среды разработки»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.4., ПК2.5.*

Цель: получение практических навыков автоматической генерации тестов на основе формального описания. Теоретическая часть.

Практически все программные системы предусматривают интерфейс с оператором. Практически всегда этот интерфейс – графический (GUI – Graphical User‘s Interface). Соответственно, актуальна и задача тестирования создаваемого графического интерфейса.

Вообще говоря, задача тестирования создаваемых программ возникла практически одновременно с самими программами. Известно, что эта задача очень трудоѐмка как в смысле усилий по созданию достаточного количества тестов (отвечающих заданному критерию тестового покрытия), так и в смысле времени прогона всех этих тестов. Поэтому решение этой задачи стараются автоматизировать (в обоих смыслах).

Для решения задачи тестирования программ с программным интерфейсом (API – Application Program Interface: вызовы методов или процедур, пересылки сообщений) известны подходы – методы и инструменты – хорошо зарекомендовавшие себя в индустрии создания программного обеспечения. Основа этих подходов следующая: создается формальная спецификация программы, и по этой спецификации генерируются как сами тесты, так и тестовые оракулы – программы, проверяющие правильность поведения тестируемой программы. Спецификации, как набор требований к создаваемой программе, существовали всегда, Ключевым словом здесь является формальная спецификация. Формальная спецификация – это спецификация в форме, допускающей еѐ формальные же преобразования и обработку компьютером. Это позволяет анализировать набор требований с точки зрения их полноты, непротиворечивости и т.п. Для задачи автоматизации тестирования эта формальная запись должна также обеспечивать возможность описания формальной связи между понятиями, используемыми в спецификации, и сущностями языка реализации программы.

Правильность функционирования системы определяется соответствием реального поведения системы эталонному поведению. Для того чтобы качественно определять это соответствие, нужно уметь формализовать эталонное поведение системы. Распространѐнным способом описания поведения системы является описание с помощью диаграмм UML (Unified

Modeling Language). Стандарт UML предлагает использование трех видов диаграмм для описания графического интерфейса системы:

- Диаграммы сценариев использования (Use Case).
- Диаграммы конечных автоматов (State Chart).
- Диаграммы действий (Activity).

С помощью UML/Use Case diagram можно описывать на высоком уровне наборы сценариев использования, поддерживаемых системой. Данный подход имеет ряд преимуществ и недостатков по отношению к другим подходам, но существенным с точки зрения автоматизации генерации тестов является недостаточная формальная строгость описания.

Широко распространено использование UML/State Chart diagram для спецификации поведения системы, и такой подход очень удобен с точки зрения генерации тестов. Но составление спецификации поведения современных систем с помощью конечных автоматов есть очень трудоѐмкое занятие, так как число состояний системы очень велико. С ростом функциональности системы спецификация становится всѐ менее и менее наглядной.

Перечисленные недостатки этих подходов к описанию поведения системы преодолеваются с помощью диаграмм действий (Activity). С одной стороны нотация UML/Activity diagram является более строгой, чем у сценариев использования, а с другой стороны предоставляет более широкие возможности по сравнению с диаграммами конечных автоматов.

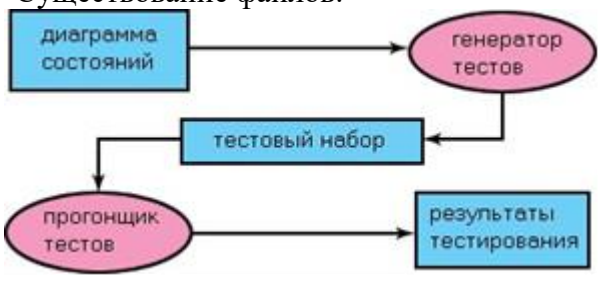
Для создания прототипа работающей версии данного подхода используется инструмент Rational Rose. В первую очередь для спецификации графического интерфейса пользователя при помощи диаграмм действий UML.

Для прогона сгенерированных по диаграмме состояний тестов используется инструмент Rational Robot. Из возможностей инструмента в работе мы использовали следующие:

1. Возможность выполнять тестовые воздействия, соответствующие переходам между состояниями в спецификации.

2. Возможность проверять соответствие свойств объектов реальной системы и эталонных свойств, содержащихся в спецификации. Из тех возможностей, которые доступны с помощью этого инструмента, используется проверка следующих свойств объектов:

- Наличие и состояние окон (заголовок, активность, доступность, статус).
- Наличие и состояние таких объектов, как PushButton, CheckBox, RadioButton, List, Tree и др. (текст, доступность, размер).
- Значение буфера обмена.
- Наличие в оперативной памяти запущенных процессов.
- Существование файлов.



Генератор строит по диаграмме состояний набор тестов. Условием окончания работы генератора является выполнение тестового покрытия. В данном случае в качестве покрытия было выбрано условие прохода по всем рѐбрам графа состояний, то есть выполнения каждого доступного тестового воздействия из каждого достижимого состояния.

Автоматическая генерация тестов по диаграммам действий имеет следующие преимущества перед остальными подходами к тестированию графического интерфейса:

Генератор тестов есть программа (script), написанная на языке ‗Rational Rose Scripting language‘ (расширение к языку Summit BasicScriptLanguage). В Rational Rose есть встроенный интерпретатор инструкций, написанных на этом языке, посредством которого можно

обращаться ко всем объектам модели (диаграммы состояний).

- Спецификация автоматически интерпретируется (тем самым она проверяется и компилируется в набор тестов).

- Если какая-то функциональность системы изменилась, то диаграмму состояний достаточно изменить в соответствующем месте, и затем сгенерировать новый тестовый набор. Фактически, это снимает большую часть проблем, возникающих при организации регрессионного тестирования.

- Гарантия тестового покрытия. Эта гарантия даѐтся соответствующим алгоритмом обхода графа состояний.

В процессе построения обхода, генератор тестов компилирует набор тестов - инструкции на языке SQABasic. Эти инструкции есть чередование тестовых воздействий и оракула свойств объектов, соответствующих данному состоянию.

Задание.

1. Сформировать диаграмму вариантов использования для задачи практической работы «Разработка структуры проекта».
2. Сгенерировать набор тестов.

Критерии оценивания:

Оценка «отлично» – полный объем выполненных работ, выполнены все задания лабораторной работы, продемонстрировано глубокое понимание материала самостоятельность выполнения работы, аккуратность и законченность работы.

Оценка «хорошо» –все задания лабораторной работы выполнены, работа оформлена с незначительными отклонениями от требований, продемонстрировано хорошее понимание материала

Оценка «удовлетворительно» - задания лабораторной работы выполнены с замечаниями, работа имеет существенные отклонения в оформлении, продемонстрировано базовое понимание материала.

Оценка «неудовлетворительно» – отсутствие полного объема работ; в работе допущены серьёзные ошибки и нарушение всех перечисленных выше требований.

Задание 2.2.2.4. Лабораторная работа № 9 «Разработка тестовых модулей проекта для тестирования отдельных модулей. Выполнение функционального тестирования. Тестирование интеграции. Документирование результатов тестирования».

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК2.1, ПК2.4., ПК2.5.*

Цель: получение практических навыков использования средств автоматизации тестирования.

Теоретическая часть.

Для того чтобы продолжать тестирование, когда один тест не прошѐл, 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 прошѐл. В противном случае, если на каком-то этапе выполнения

теста, поведение системы не соответствует требованиям, Robot прекращает его выполнение, вынося соответствующий вердикт – тест не прошѐл.

Задание.

1. Выполнить тестовый набор лабораторной работы № 2.
2. Проанализировать отчет о прохождении тестов.

Критерии оценивания:

Оценка **«отлично»** – полный объем выполненных работ, выполнены все задания лабораторной работы, продемонстрировано глубокое понимание материала самостоятельность выполнения работы, аккуратность и законченность работы.

Оценка **«хорошо»** –все задания лабораторной работы выполнены, работа оформлена с незначительными отклонениями от требований, продемонстрировано хорошее понимание материала

Оценка **«удовлетворительно»** - задания лабораторной работы выполнены с замечаниями, работа имеет существенные отклонения в оформлении, продемонстрировано базовое понимание материала.

Оценка **«неудовлетворительно»** – отсутствие полного объема работ; в работе допущены серьёзные ошибки и нарушение всех перечисленных выше требований.

3. Тестовые задания для текущего контроля
ОК01, ОК02, ОК03, ОК04, ОК05, ОК09, ПК2.1, ПК2.4, ПК2.5.

1. Прочитайте текст, выберите все правильные ответы

Как называется численное значение критерия качества, определяющее степень, в которой программе присуще определенное критерием свойство?

а) метрика;
б) оценочный элемент;
в) показатель качества;
г) фактор качества.

Ответ:

2. Прочитайте текст, выберите все правильные ответы

Как называется процесс анализа или эксплуатации программного обеспечения с целью выявления дефектов?

а) тестирование;
б) испытание;
в) исследование;
г) валидация.

Ответ:

3. Прочитайте текст, выберите все правильные ответы

Для какого процесса проверки и анализа качества программного средства создается представительная комиссия из экспертов, представителей заказчика и представителей разработчика?

а) верификации;
б) аттестации;
в) валидации;
г) эксплуатации.

Ответ:

4. Прочитайте текст, выберите все правильные ответы

Как называется событие при выполнении программы, которое приводит к ее ненормальному или неправильному поведению?

а) исключение;
б) дефект;
в) семантическая ошибка;
г) логическая ошибка.

Ответ:

4. Прочитайте текст, выберите все правильные ответы

К какому виду исключений относится попытка деления на 0?

а) программным;
б) аппаратным;
в) эксплуатационным;
г) алгоритмическим.

Ответ:

5. Прочитайте текст, выберите все правильные ответы

С какого ключевого слова начинается оператор обработки исключительной ситуации в языке программирования 1С?

а) Пока;
б) except;
в) Попытка;
г) try;

Ответ:

6. Прочитайте текст и установите соответствие

Сопоставьте названия и назначение блоков оператора обработки исключений в Python

К каждой позиции, данной в левом столбце, подберите соответствующую позицию из правого столбца:

А	запуск кода всегда	1	try
Б	запуск кода, если не было исключений	2	except
В	запуск кода	3	else
Г	запуск кода, если возникло исключение	4	finally

Запишите выбранные цифры под соответствующими буквами:

А	Б	В	Г

7. Прочитайте текст и установите соответствие

Сопоставьте названия исключений и фрагменты программного кода с исключениями.

Название исключения Программный код с исключением

К каждой позиции, данной в левом столбце, подберите соответствующую позицию из правого столбца:

А	for i in range(10): print('Привет Мир!')	1	SyntaxError
Б	a = 2 b = 0	2	TypeError

	c = a/b		
В	a = 2 b = 3 c = a b	3	IndentationError
Г	a = 2 b = 'PythonRu' c = a + b	4	ZeroDivisionError

Запишите выбранные цифры под соответствующими буквами:

А	Б	В	Г

8. Прочитайте текст и установите последовательность

Установите правильную последовательность этапов процесса компиляции

А) оптимизация;

Б) лексический анализ;

В) генерация кода;

Г) семантический анализ;

Д) синтаксический анализ;

Запишите соответствующую последовательность цифр слева направо

9. Прочитайте текст и установите последовательность

Установите правильную последовательность процесса разработки тестового сценария.

А) Определение цели;

Б) Описание ожидаемых результатов;

В) Документирование;

Г) Описание шагов;

Д) Понимание требований.

Запишите соответствующую последовательность цифр слева направо

10. Какая отладочная функция Python проверяет условие на истинность, и если условие истинно, то код работает дальше, а если ложно, то возбуждает исключение типа AssertionError?

11. К какой группе исключений в Python относятся исключения ZeroDivisionError (Ошибка деления на ноль) FloatingPointError (Ошибка плавающей точки) и OverflowError (Ошибка переполнения)?

12. В какой блок нужно заключить программный код на языке Python, если он может привести к исключению при выполнении?

13. Какое сообщение увидит пользователь при выполнении данного фрагмента программы?

```
Попытка
Попытка
a = 1/0;
Исключение
Сообщить("Конкретная ошибка");
```

КонецПопытки;
Исключение
Сообщить("Общая ошибка");
КонецПопытки;

14. Какой текст ошибки платформа 1С выдаст, если в имени переменной допущена опечатка?

15. Какой параметр сложности интеграции определяется как несовместимость форматов обмена данными, протоколов взаимодействия и интерфейсов программ?

16. При какой технике тестирования у тестировщика есть доступ к разрабатываемому коду?

17. Какой вид тестирования следует применить в первую очередь после выхода новой версии программы?

18. Допишите название определения: «…. - это документ, описывающий весь объем работ по тестированию, начиная с описания объекта, стратегии, расписания, критериев начала и окончания тестирования, до необходимого в процессе работы оборудования, специальных знаний, а также оценки рисков с вариантами их разрешения».

19. Допишите название определения: «…. – это автономный элемент программного обеспечения, предназначенный для многократного использования, который может распространяться для использования в других программах в виде скомпилированного кода».

20. Прочитайте текст, выберите все правильные ответы

Как называется характеристика программного обеспечения, демонстрирующая то, что продукт удовлетворяет установленным требованиям или превышает их?

А) полнота;
Б) качество;
В) стандарт;
Г) алгоритмическая сложность.

Ответ:

21. Прочитайте текст, выберите все правильные ответы

Как называется единый стандарт кодирования, который применим к объектно-ориентированному языку программирования Python?

А) PEP 8;
Б) UTF-8;
В) Latin-1;
Г) ISO-8.

Ответ:

22. Прочитайте текст, выберите все правильные ответы

На каком этапе компиляции программы выполняется удаление излишних конструкций и упрощение кода с сохранением его смысла.

А) лексического анализа;
Б) оптимизации;
В) генерации кода;
Г) семантического анализа.

Ответ:

23. Прочитайте текст, выберите все правильные ответы

Какое качество задается требованиями заказчика в спецификациях и отражается в характеристиках конечного программного продукта?

А) внутреннее;

Б) при использовании в процессе эксплуатации;

В) внешнее;

Г) эксплуатационное.

Ответ:

24. Прочитайте текст, выберите все правильные ответы

Как называется инструмент, который используется для отслеживания, внесения и управления изменениями в программном коде?

А) схема контроля версий;

Б) схема компоновки данных;

В) система компоновки данных;

Г) система контроля версий.

Ответ:

Ключ к тестовым заданиям

№ задания	Верный ответ	Критерии
1	В	1б – полное правильное соответствие 0 б – остальные случаи
2	А	1б – полное правильное соответствие 0 б – остальные случаи
3	Б	1б – полное правильное соответствие 0 б – остальные случаи
4	А	1б – полное правильное соответствие 0 б – остальные случаи
5	Б	1б – полное правильное соответствие 0 б – остальные случаи
6	В	1б – полное правильное соответствие 0 б – остальные случаи
7	В1Г2Б3А4	1б – полное правильное соответствие 0 б – остальные случаи
8	В1Г2А3Б4	1б – полное правильное соответствие 0 б – остальные случаи
9	БДГАВ	1б – полное правильное соответствие 0 б – остальные случаи
10	ДАГБВ	1б – полное правильное соответствие 0 б – остальные случаи
11	Assert	1б – полное правильное соответствие 0 б – остальные случаи
12	ArithmeticError (арифметические ошибки)	1б – полное правильное соответствие 0 б – остальные случаи
13	try	1б – полное правильное соответствие 0 б – остальные случаи
14	Конкретная ошибка	1б – полное правильное соответствие

		0 б – остальные случаи
15	Переменная не определена	1б – полное правильное соответствие 0 б – остальные случаи
16	технологическая разница	1б – полное правильное соответствие 0 б – остальные случаи
17	метод белого ящика	1б – полное правильное соответствие 0 б – остальные случаи
18	дымовое тестирование	1б – полное правильное соответствие 0 б – остальные случаи
19	тест-план	1б – полное правильное соответствие 0 б – остальные случаи
20	программный компонент	1б – полное правильное соответствие 0 б – остальные случаи
21	Б	1б – полное правильное соответствие 0 б – остальные случаи
22	А	1б – полное правильное соответствие 0 б – остальные случаи
23	Б	1б – полное правильное соответствие 0 б – остальные случаи
24	В	1б – полное правильное соответствие 0 б – остальные случаи

Критерии оценки:

соответствие ответов обучающихся ключу теста

Оценка «**отлично**» - если обучающийся правильно выполнил от 80% до 100% тестовых заданий в отведенное время

Оценка «**хорошо**» - если обучающийся правильно выполнил от 60% до 80% тестовых заданий в отведенное время

Оценка «**удовлетворительно**» - если обучающийся правильно выполнил от 40% до 60% тестовых заданий в отведенное время

Оценка «**неудовлетворительно**» ставится в случае выполнения менее 40% тестовых заданий

Время выполнения: 35-40 минут

5. Задания для оценки освоения дисциплины МДК.2.3 Математическое моделирование

Тема 2.3.1. Основы моделирования. Детерминированные задачи

Задание 2.3.1.1. Практическая работа № 16 «Сведения произвольной задачи линейного программирования к основной задаче линейного программирования. Решение задач линейного программирования симплекс-методом»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ПК2.1, ПК2.4., ПК2.5.*

Цель: Научиться сводить произвольную задачу линейного программирования к основной задаче линейного программирования. Решить задачу линейного программирования симплекс-методом.

Оборудование и материалы: Методические рекомендации; рабочая тетрадь.

Краткие теоретические основания выполнения задания

Задачи оптимального планирования, связанные с отысканием оптимума заданной целевой функции (линейной формы) при наличии ограничений в виде линейных уравнений или линейных неравенств относятся к задачам линейного программирования.

Линейное программирование – это направление математического программирования, изучающее методы решения экстремальных задач, которые характеризуются линейной зависимостью между переменными и линейным критерием.

Сущность линейного программирования состоит в нахождении точек наибольшего или наименьшего значения некоторой функции при определенном наборе ограничений, налагаемых на аргументы и образующих систему ограничений, которая имеет, как правило, бесконечное множество решений. Каждая совокупность значений переменных (аргументов функции F), которые удовлетворяют системе ограничений, называется допустимым планом задачи линейного программирования. Функция F, максимум или минимум которой определяется, называется целевой функцией задачи. Допустимый план, на котором достигается максимум или минимум функции F, называется оптимальным планом задачи.

Система ограничений, определяющая множество планов, диктуется условиями производства. Задачей линейного программирования (ЗЛП) является выбор из множества допустимых планов наиболее выгодного (оптимального).

Общая форма задачи линейного программирования формулируют следующим образом:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \{\leq, \geq, =\} b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \{\leq, \geq, =\} b_2, \\ \ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \{\leq, \geq, =\} b_m. \end{cases}$$

$$x_1 \geq 0,\ x_2 \geq 0,\ \ldots,\ x_n \geq 0$$

$$F = c_1x_1 + c_2x_2 + \ldots + c_nx_n \rightarrow \max(\min)$$

Коэффициенты ai,j, bi, cj, j = 1, 2, ... , n, i =1, 2, ... , m – любые действительные числа (возможно 0).

Итак, решения, удовлетворяющие системе ограничений (1) условий задачи и требованиям неотрицательности (2), называются допустимыми, а решения, удовлетворяющие одновременно и требованиям минимизации (максимализации) (3) целевой функции, - оптимальными.

Выше описанная задача линейного программирования (ЗЛП) представлена в общей форме, но одна и та же (ЗЛП) может быть сформулирована в различных эквивалентных формах. Наиболее важными формами задачи линейного программирования являются каноническая и стандартная.

В канонической форме задача является задачей на максимум (минимум) некоторой линейной функции F, ее система ограничений состоит только из равенств (уравнений). При этом переменные задачи x1, x2, ..., xn являются неотрицательными:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2, \\ \ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m. \end{cases}$$

$$x_1 \geq 0,\ x_2 \geq 0,\ \ldots,\ x_n \geq 0$$

$$F = c_1x_1 + c_2x_2 + \ldots + c_nx_n \rightarrow \max(\min)$$

$$F = c_1x_1 + c_2x_2 + \ldots + c_nx_n \rightarrow \max(\min)$$

К канонической форме можно преобразовать любую задачу линейного программирования. Правило приведения ЗЛП к каноническому виду:

1. Если в исходной задаче некоторое ограничение (например, первое) было неравенством, то оно преобразуется в равенство, введением в левую часть некоторой неотрицательной переменной, при чем в неравенства «≤» вводится дополнительная неотрицательная переменная со знаком «+»; в случаи неравенства «≥» - со знаком «-»

Вводим переменную .

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1 \qquad (7)$$

$$x_{n+1} = b_1 - a_{11}x_1 - a_{12}x_2 + \ldots + a_{1n}x_n$$

Тогда неравенство (7) запишется в виде:

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + x_{n+1} \qquad (8)$$

В каждое из неравенств вводится своя “уравнивающая” переменная, после чего система ограничений становится системой уравнений.

2. Если в исходной задаче некоторая переменная не подчинена условию неотрицательности, то ее заменяют (в целевой функции и во всех ограничениях) разностью

$$x_k = x_k - x_l,$$
$$x_k \geq 0,\ x_l \geq 0$$

неотрицательных переменных , свободный индекс

3. Если в ограничениях правая часть отрицательна, то следует умножить это ограничение на (-1)

4. Наконец, если исходная задача была задачей на минимум, то введением новой целевой функции F1 = -F мы преобразуем нашу задачу на минимум функции F в задачу на максимум функции F1.

Таким образом, всякую задачу линейного программирования можно сформулировать в канонической форме.

В стандартной форме задача линейного программирования является задачей на максимум (минимум) линейной целевой функции. Система ограничений ее состоит из одних линейных неравенств типа « <= » или « >= ». Все переменные задачи неотрицательны.

$$F = c_1x_1 + c_2x_2 + \ldots + c_nx_n \to \max(\min)$$

$$F = c_1x_1 + c_2x_2 + \ldots + c_nx_n \to \max(\min)$$

$$F = c_1x_1 + c_2x_2 + \ldots + c_nx_n \to \max(\min)$$

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1,$$
$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leq b_2,$$
$$\ldots\ldots$$
$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq b_m.$$

$$F = c_1x_1 + c_2x_2 + \ldots + c_nx_n \to \max(\min)$$

$$x_1 \geq 0,\ x_2 \geq 0,\ \ldots,\ x_n \geq 0$$

Всякую задачу линейного программирования можно сформулировать в стандартной форме. Преобразование задачи на минимум в задачу на максимум, а также обеспечение не отрицательности переменных производится так же, как и раньше. Всякое равенство в системе ограничений равносильно системе взаимопротивоположных неравенств:

Существует и другие способы преобразования системы равенств в систему неравенств, т.е. всякую задачу линейного программирования можно сформулировать в стандартной форме.

Решение задач линейного программирования симплекс-методом.

Идея симплекс-метода заключается в последовательном улучшении первоначального плана путем упорядоченного перехода от одного опорного плана к другому и завершается нахождением оптимального плана. Симплексметодом решаются только канонические задачи линейного программирования. Решение канонической задачи симплекс-методом существенно облегчается применением так называемых симплексных таблиц. Всякую каноническую задачу можно записать условно в виде таблицы. Таблица заполняется следующим образом: первые т строк содержат в условной форме уравнения системы ограничений, разрешенные относительно базисных переменных. В последней строке записана целевая функция, эта строка называется F - строкой. В столбцах записаны свободные переменные и свободные члены.

Условие оптимальности плана: если ЗЛП на максимум, то в F-строке не должно быть отрицательных элементов; если ЗЛП на минимум, то в F-строке не должно быть положительных элементов. Алгоритм решения:

1. Исходную задачу линейного программирования приводим к каноническому виду путем введения базисных переменных.

2. Базисные переменные выражаем через свободные переменные.

3. Строим начальный план, полагая свободные переменные равными нулю, тогда базисные переменные будут равны свободным членам.

4. Строим первую симплекс-таблицу.

5. Проверяем план на оптимальность. Если план не оптимален, то его улучшаем.

6. Улучшение плана.

а) выбор разрешающего столбца: для этого в F- строке выбираем максимальный по абсолютной величине из отрицательных элементов, если задача на максимум, или, максимальный из положительных элементов, если задача на минимум. Пусть это будет столбец с номером s;

б) выбор разрешающей строки: выбираем строку с минимальным симплексным отношением. Симплексные отношения - это отношение свободных членов к положительным элементам разрешающего столбца. Пусть это будет строка с номером r.

в) выбор разрешающего элемента: элемент, стоящий на пересечении разрешающих строки и столбца. Пусть это будет элемент ars .

г) переменную xs вводим в базис вместо переменной xr .

д) элементы новой симплекс-таблицы bij пересчитываем по следующим формулам:

1

разрешающий элемент brs □ ,

ars

элементы разрешающего столбца bis □□ ais ,i □ r ,

ars

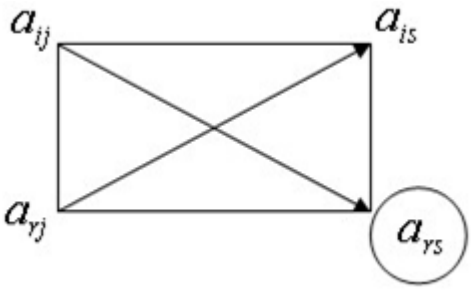
arj элементы разрешающей строки brj □ , j □ s ,

ars

остальные элементы симплекс-таблицы по правилу прямоугольника:

bij □ aij □ars □arj □ais

ars



7. Вновь полученный план проверяем на оптимальность.

Порядок выполнения заданий

Задание 1.

а) Привести к канонической форме задачу линейного программирования.

□2x1 □x2 □ 3x3 □ 5

□

□ x1 □ 2x3 □ 8 □□□x1 □ 2x2 □1

x1 □ 0,x2 □ 0,x3 □ 0

F □ x1 □x2 □ 3x3 □ min

б) Напишите задачу в стандартной форме.

Решение:

а) Введем дополнительные переменные x4 , x5 . Причем в первое неравенство введем переменную x4 со знаком плюс, а в третье – неотрицательную переменную, x5 со знаком минус запишем задачу в виде:

□2x1 □x2 □ 3x3 □x4 □ 5

□

□□□□xx11□□22xx32□□8x5 □1

x1 □ 0,x2 □ 0,x3 □ 0,x4 □ 0,x5 □ 0

Переведем min на max, домножив целевую функцию на (-1) F □ □x1 □ x2 □3x3 □ 0□ x4 □ 0□ x5 □ max что и дает эквивалентную задачу в канонической форме.

б) Всякую задачу линейного программирования можно сформулировать в стандартной форме. Преобразование задачи на минимум в задачу на максимум, а также обеспечение не отрицательности переменных производится так же, как и раньше. Всякое равенство в системе ограничений равносильно системе взаимопротивоположных неравенств, тогда получим:

□2x1 □x2 □ 3x3 □ 5

□□ x1 □ 2x3 □ 8

□

□□x1 □ 2x3 □ 8

□□□x1 □ 2x2 □1

x1 □ 0,x2 □ 0,x3 □ 0

F □□x1 □x2 □3x3 □ max

Задание 2. Для производства двух видов, изделии P1 и P2 используется, три вида сырья S1,S2,S3, запасы которого соответственно равны 100, 60, 180 единиц. Для производства одной единицы продукции P1 используется 2 единицы сырья S1 и по 1 единице сырья S2иS3. Для производства одной единицы продукции P2 используется по 1 единице сырья S1иS2 и 4 единицы сырья S3.

Прибыль от реализации 1 единицы каждой продукции P1 и P2 соответственно равна 30 и 20 единиц. Необходимо составить симплекс-методом такой план выпуска продукции P1 и P2, при котором суммарная прибыль будет наибольшей.

Решение.

1. Составим математическую модель задачи: Пусть x1 – единица готовой продукции

вида Р1, x2 - единица готовой продукции вида Р2,

Цель фабрики получить максимальную прибыль от реализации всей продукции видов Р1 и Р2, тогда:

F □ 30□ x1 □ 20□ x2 □ max

Система ограничений:

□2x1 □x2 □100

□

□ x1 □x2 □ 60

□□x1 □ 4x2 □180 x1 □ 0, x2 □ 0 условие неотрицательности

2. Задачу приводим к каноническому виду: F □ 30□ x1 □ 20□ x2 □ max

□2x1 □x2 □x3 □100

□

□ x1 □x2 □x4 □ 60

□□x1 □ 4x2 □x5 □180

x1 □ 0, x2 □ 0 x3 □ 0, x4 □ 0, x5 □ 0

3. Базисные переменные выражаем через свободные:

□x3 □100□2x1 □x2

□□ x4 □60□x1 □x2 □□x5 □180□x1 □4x2

4. Записываем начальный план: $X_0 = (0;\ 0;\ 100;\ 60;\ 180)$.

5. Строим первую симплекс-таблицу:

Таблица 1. Первая симплекс-таблица

Своб. перем. / Базис. перем.	$-x_1$ ▼	$-x_2$	Свободные члены	Симплексные отношения
► x_3	(2)	1	100	$\frac{100}{2} = 50 \min$
x_4	1	1	60	$\frac{60}{1} = 60$
x_5	1	4	180	$\frac{180}{1} = 180$
F-строка	-30	-20	0	

6. Начальный план не оптимален, так как в F-строке есть отрицательные элементы.

7. Улучшение плана. Строим вторую симплекс-таблицу, элементы которой пересчитываем по соответствующим формулам.

8. План, соответствующий таблице 2, X1 □ □50; 0; 0; 10; 130□. не оптимален, так как в F-строке есть отрицательные элементы. Улучшаем его.

9. Улучшение плана. Строим третью симплекс-таблицу, элементы которой пересчитываем по соответствующим формулам.

Таблица 3. Третья симплекс-таблица

Своб. перем. / Базис. перем.	$-x_3$	$-x_4$	Свободные члены	Симплексные отношения
x_1	1	-1	40	
x_2	-1	2	20	
x_5	3	-7	60	
F-строка	10	10	1600	

11.

12.План, соответствующий таблице 3, X2 □ □40; 20; 0; 0; 60□. оптимален, так как в F строке нет отрицательных элементов.

Ответ: если предприятие будет выпускать продукцию вида P1 и P2 в количестве 40 и 20 единиц соответственно, то получит максимальную прибыль в размере 1600 единиц, при этом сырье S1иS2 будет израсходовано полностью, а сырье S3 останется в количестве 60 единиц.

Задание 2.3.1.2. Практическая работа № 17 «Нахождение начального решения транспортной задачи. Решение транспортной задачи методом потенциалов»

Проверяемые результаты обучения: *ОК1, ОК2, ОК03, ПК2.1, ПК2.4., ПК2.5.*

Цель работы: Найти начальное решение транспортной задачи двумя методами: методом северо-западного угла и методом наименьшей стоимости. Найти оптимальное решение транспортной задачи методом потенциалов.

Краткая теория

Симплексный метод для решения задач линейного программирования является универсальным, он позволяет решить любую задачу, но решение иных задач связано с трудоемкими расчетами. Можно выделить класс задач, которые решаются более простыми специальными методами. К числу таких задач относятся так называемые **транспортные задачи.**

Классическая транспортная задача - о наиболее экономном плане перевозок однородного продукта или взаимозаменяемых продуктов из пунктов отправления в пункты назначения.

Классическая транспортная задача (сокращенно ТЗ) формулируется следующим образом.

В пунктах отправления $A^1, A^2, ..., A^m$, которые будем называть также поставщиками, $a^1, a^2, ..., a^m$ соответственно. В пункты сосредоточены запасы однородного груза в количествах $B^1, B^2, ..., B^n$, именуемые потребителями, надлежит назначения доставить соответственно $b^1, b^2, ..., b^n$ единиц груза.

Известен c^{ij} - стоимость перевозки единицы Ai в транспортный тариф груза из пункта $i = 1,2,...,m$, $j = 1,2,...,n$ пункт B_j, .

Требуется составить такой план перевозок груза, при котором общая стоимость F всех перевозок была бы наименьшей, при этом все заявки были бы выполнены.

В термин "транспортный тариф" вкладывается условное понимание стоимости единицы груза - это может быть себестоимость, расстояние, тариф, время, расход топлива или электроэнергии и др.

Пусть суммарные запасы грузов у поставщиков равны суммарным потребностям потребителей:

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$$

Это условие называется условием баланса. Если для ТЗ условие баланса выполняется, то модель ТЗ называется закрытой, если условие баланса не выполнено, то модель ТЗ - открытая. Составим математическую модель ТЗ.

x^{ij} - количество груза, которое A_i отправляет потребителю B^j (

Пусть поставщик

i = 1,2,...,m, j = 1,2,...,n) со стоимостью c^{ij}. Данные задачи можно представить в перевозок виде таблицы 1. Таблица 1.

Поставщики	Потребители				Запасы
	B_1	B_2	…	B_n	
$A1$	c_{11} x_{11}	c_{12} x_{12}	…	c_{1n} x_{1n}	$a1$
$A2$	c_{21} x_{21}	c_{22} x_{22}	…	c_{2n} x_{2n}	a_2
…	…	…	…	…	…
Am	c_{m1} x_{m1}	c_{m2} x_{m2}	…	c_{mn} x_{mn}	am
Потребности	b_1	b_2	…	b_n	$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$

$x_{ij} \geq 0$

По смыслу своему и должны удовлетворять величины следующим ограничениям:

A^i все запасы должны быть вывезены (ограничения по ресурсам): ● Из пункта

$$\sum_{j=1}^{n} x_{ij} = a_i \ (i = 1,2,..., m)$$

j =1 ; должны быть выполнены (ограничения по Заявки потребителей B^j

●

$$\sum_{i=1}^{m} x_{ij} = b_j \ (j = 1,2,..., n)$$ потребностям):

.

x^{ij} единиц груза из пункта поставки в пункт потребления B^j Затраты на перевозку A_i

$c \cdot x \quad x$

составляют $ij \quad ij$ рублей; общая же ij равна сумме всех стоимость всех таких перевозок

$m \quad n$

$F = \sum\sum cij \cdot xij \rightarrow \min$

затрат: $i=1\ j=1$

Математическая постановка ТЗ состоит в следующем:

x^{ij}, удовлетворяющих системе ограничении: составить план перевозок

$$\begin{cases} \sum_{i=1}^{m} xij = b_j, (j = 1,2,..., n) \\ \sum_{j=1}^{n} xij = a_i, i = 1,2,..., m \end{cases}$$

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j,$$

условию $xij \geq 0$, $i = 1,2,...,m, j = 1,2,...,n$, при неотрицательности: функция

достигает своего минимума: котором целевая

$m \quad n$

$F = \sum\sum cij \cdot xij \rightarrow \min$

$i=1\ j=1$

Из математической модели видно, что ТЗ является частным случаем общей задачи линейного программирования. В общей теории линейного программирования доказаны следующие теоремы:

Теорема 1. Транспортная задача при выполнении условия баланса всегда имеет решение.

Теорема 2. Система ограничений транспортной задачи содержит m+n-1 линейно-независимых уравнений.

При решении задач практический смысл теоремы 2 заключается в следующем: число назначенных перевозок равно $m+n-1$.

Процедура решения ТЗ будет состоять в последовательном улучшении опорных планов и проверки их на оптимальность.

Методы построения начального плана.

Существует несколько методов построения первоначального опорного плана ТЗ (опорный план - план, удовлетворяющий системе ограничений и условию неотрицательности). Рассмотрим только два из них: метод северо-западного угла и метод наименьшей стоимости.

Как уже отмечалось, в опорном плане x^{ij}, отличных не более $\boldsymbol{r = m + n - 1}$ переменных

от нуля. Если таких переменных равно $\boldsymbol{r}$, то такой план называют невырожденным, в противном случае - вырожденным.

Метод северо-западного угла. Назначение перевозок начинаем с левой верхней клетки (северо-западный угол). Сравнивая ресурсы поставщика и потребности потребителя, назначаем максимально возможную перевозку. Если ресурсов поставщика недостаточно, то переходим к следующему поставщику. Если ресурсов у поставщика достаточно, то назначив нужную перевозку первому потребителю, переходим к следующему потребителю. При назначении перевозок для удобства записываем остаток ресурсов (потребностей); если ресурсы

закончились или потребности удовлетворены, то ставим букву "к" (конец). Если при назначении перевозки одновременно закончились запасы ресурсов у поставщика и удовлетворены потребности потребителя, то из "игры" выводим только одного участника, другому оставляем нуль запасов или нуль потребностей.

Метод наименьшей стоимости. Выбираем клетку с наименьшей тарифной ставкой и назначаем максимально возможную перевозку. Если запасы закончились или потребности удовлетворены, то поставщика или потребителя исключаем. Среди оставшихся клеток снова выбираем клетку с наименьшей стоимостью и назначаем максимально возможную перевозку. Если в результате назначения перевозки закончились запасы поставщика или удовлетворены потребности потребителя, то его исключаем из дальнейшего рассмотрения.

Метод потенциалов построения оптимального плана.

Наиболее простым методом решения ТЗ является метод потенциалов. Потенциалами называются условные u^i, v^j приписанные определенным числа поставщику и образом каждому потребителю.

Теорема 3(условие оптимальности плана). Сумма потенциалов поставщика и потребителя равна тарифной ставке для занятых клеток; сумма потенциалов поставщика и потребителя не превышает тарифную ставку для свободных клеток:

$$\begin{cases} u_i + v_j = c_{ij}, & x_{ij} \geq 0, \\ ui + v\,j \leq cij, & xij = 0. \end{cases}$$

Замечание. Опорный план должен быть невырожденным. ***Алгоритм решения транспортной задачи:***

1. Строим начальные планы методом северо-западного угла и наименьшей стоимости, из них выбираем лучший.
2. Находим потенциалы поставщиков и потребителей, используя первое условие $u^i + v^j = c^{ij}$

оптимальности плана:

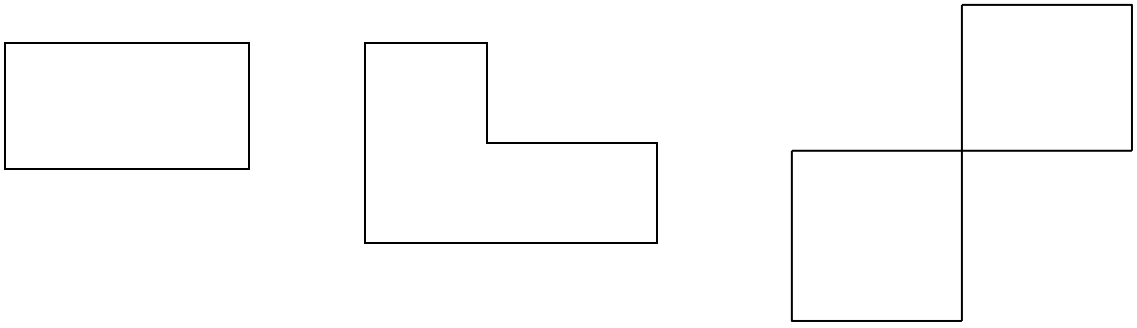
3. Проверяем второе условие оптимальности $ui + v\,j \leq cij$. Если плана для свободных клеток оно выполнено, то план оптимален. Если не выполнено, то улучшаем план.
4. Улучшение плана.

a) при невыполнении второго условия оптимальности плана в клетку заносим нарушение $u^i + v^j - c^{ij}$ со знаком "+". Такие клетки называются потенциальными;

b) среди всех потенциальных клеток выбираем клетку с наибольшим нарушением;

c) строим для выбранной клетки замкнутый контур, состоящий из вертикальных и горизонтальных отрезков прямой, причем вершины контура лежат в занятых клетках, за исключением той клетки, для которой строится контур. Виды контуров приведены на рисунке

1;



вершины контура поочередно помечаем, знаками "+","-", начиная с клетки, для которой построен контур;

e) среди клеток, помеченных знаком "-", выбираем наименьшую перевозку. На эту величину увеличиваем перевозки в клетках, помеченных знаком "+", и уменьшаем в

клетках, помеченных знаком "-". В результате переназначения перевозок освобождается одна клѐтка.

5. Вновь полученный план проверяем на оптимальность.

Порядок выполнения заданий

Задание. Имеются три пункта поставки однородного груза **A1, A2, A3** и пять пунктов **B1, B2, B3, B4, B5** потребления этого груза. На пунктах **A1, A2 и A3** находится груз соответственно в количестве ***a1, a2 и a3*** тонн. В пункты **B1, B2, B3, B4, B5** требуется доставить соответственно ***b1, b2, b3, b4, b5*** тонн груза. Расстояние между пунктами поставки и пунктами потребления приведено в таблице:

Пункты поставки	**Пункты потребления**				
	B1	**B2**	**B3**	**B4**	**B5**
A1	D11	D12	D13	D14	D15
A2	D21	D22	D23	D24	D25
A3	D31	D32	D33	D34	D35

Найти такой план закрепления потребителей за поставщиками однородного груза, чтобы общие затраты по перевозкам были минимальными.

a1=200, a2=250, a3=200, b1=190, 100, 120, 110, b5=130.

$$D = \begin{pmatrix} 28 & 27 & 18 & 27 \\ 26 & 27 & 24 & 32 \\ 27 & 33 & 21 & 3 \\ 23 & 1 & 34 & \end{pmatrix}$$

b2= b3= b4=

Решение.

1. Построим начальный план двумя методами: методом северо-западного угла и методом наименьшей стоимости, и выберем тот план, который будет наилучшим, то есть получим минимальные затраты за перевозку однородного груза.

А) Строим начальный план методом северо-западного угла. Составим таблицу значений:

Потребители Поставщики	**B1**	**B2**	**B3**	**B4**	**B5**	Запасы
A1	28 190	27 10	18	27	24	200, 10, к
A2	18	26 90	27 120	32 40	21	250, 160, 40, к
A3	27	33	23	31 70	34 130	200, 130, к
Потребности	190, к	100, 90, к	120, к	110, 70, к	130, к	650 =650

Число назначенных перевозок ***m+n-1=3+5-1=7***, то есть начальный план невырожденный

$x_{12} = 10,\ x_{22} = 90,\ x_{23} = 120,\ x_{24} = 40,\ x_{34} = 70,$

[1] й.

1
=
1
9
0
, Начальный план:
$x_{35} = 130$ $x\ x_{13} = 120$, $x_{15} = 70$, $x_{21} = 190$, x_{25} = $x_{32} = 90$, x_{34} =
При 1 60, 110
таком плане 2
суммарные =
транспортные 1
издержки 0
равны: ,
F = 28 . При таком плане транспортные
190 + 27 . 10 + издержки
26 . 90 + 27 .
120 + 32 . 40 +
31. 70 + 34 .
130 = 5320 + 270 + 2340 + 3240
+ 1280 + 2170 + 4420 = 19040(*единиц*)

Б) Строим начальный план методом наименьшей стоимости. Составим таблицу значений:

Потребители Поставщики	**B1**	**B2**	**B3**	**B4**	**B5**	Запасы
A1	28	27 10	18 120	27	24 70	200, 80, 10, к
A2	18	26	27	32	21	250, 60, к
	190				60	
A3	27	33 90	23	31 110	34	200, 90, к
Потребности	190, к	100, 90, к	120, к	110, к	130, 70, к	650= 650

1680 + 3420 +
+ 1260 + 2970 + 3410 = 15170(*единиц*)

Сравнивая транспортные издержки, видим, что план, построенный методом наименьшей стоимости, лучший.

2. $F = 27 \cdot 10 + 18 \cdot 120 + 24 \cdot 70 + 18 \cdot 190 + 21 \cdot 60 + 33 \cdot 90 + 31 \cdot 110 = 270 + 2160$

Выбираем + лучший план и находим потенциалы поставщиков и потребителей, используя первое условие оптимальности = c_{ij} плана: $u_i + v_j$

Потребители, v_j / Поставщики, u_i		**21**	**27**	**18**	**25**	**24**
		B1	**B2**	**B3**	**B4**	**B5**
0	**A1**	28	27 10	18 120	27	24 70
-3	**A2**	18 190	26	27	32	21 60
6	**A3**	27	33 90	23 **+1**	31 110	34

Используя первое условие оптимальности плана, составим систему линейных уравнений для определения потенциалов:

$u_1 + v_2 = 27$

$u + v = 18$

$u + v = 24$

$u + v = 21$

$u_2 + v_1 = 18$

$u_3 + v_2 = 33$

$u_3 + v_4 = 31$

Система линейных уравнений содержит 7 уравнений и 8 неизвестных, т.е. она имеет множество решений. Так как нужно одно решение, то любой из неизвестных задаем значение и вычисляем остальные неизвестные.

$u = 0$

Пусть u_3, тогда

$u_3 = 6$, $v_1 = 21$, $v_2 = 27$, $v_3 = 18$, $v_4 = 25$, $v_5 = 24$

2 = − 3 ,

3. Проверяем второе условие оптимальности $ui + v\ j \leq cij$. Если плана для свободных клеток есть нарушения, то заносим их со знаком «+». В результате проверки получили одну потенциальную клетку. Таким образом, начальный план не оптимален.

4. Улучшение плана. Выбираем клетку с максимальным нарушением и для нее строим замкнутый контур.

Потребители, v_j Поставщики, u_i						
		B1	**B2**	**B3**	**B4**	**B5**
	A1	28	27 + 10	18 - 120	27	24 70
	A2	18	44	27	32	21
		190				60
	A3	27	3 ---------- 0	23 **+1**	31 110	34

Среди клеток, помеченных знаком «-», выбираем наименьшую перевозку: q = min(90,120) = 90

На эту величину увеличиваем перевозки в клетках, помеченных знаком «+», и уменьшаем в клетках, помеченных знаком «-». В результате переназначения перевозок имеем план:

Потребители, v_j Поставщики, u_i		**21**	**27**	**18**	**26**	**24**
		B1	**B2**	**B3**	**B4**	**B5**
0	**A1**	28	27 100	18 30	27	24 70
-3	**A2**	18 190	26	27	32	21 60
5	**A3**	27	33	23 90	31 110	34

Используя первое условие оптимальности плана, составим систему линейных уравнений для определения потенциалов:

$$\begin{cases} u_1 + v_2 = 27 \\ u \quad \| \quad 1\ 3 \quad = 18 \\ u_1 + v_5 = 24 \\ u_2 + v_1 = 18 \\ u \quad | \quad = 21 \\ | \quad 2 \quad 5 \\ u3 + v3 = 23 \\ u3 + v4 = 31 \end{cases}$$

Система линейных уравнений содержит 7 уравнений и 8 неизвестных, т.е. она имеет множество решений. Так как нужно одно решение, то любой из неизвестных задаем значение и вычисляем остальные неизвестные. $u = 0$

Пусть 1 , тогда $u2 = -3$, $u3 = 5$, $v1 = 21$, $v2 = 27$, $v3 = 18$, $v4 = 26$, $v5 = 24$ Проверяем второе условие $ui + v\,j \le cij$. оптимальности плана для свободных клеток

Условие оптимальности выполнены, следовательно, план, соответствующий таблице, оптимален.

x $x13 = 30$, $x15 = 70$, $x21 = 190$, $x25 = 60$, $x33 = 90$, $x34 = 110$

1
2
=
1
0

0 , $F = 27 . 100 + 18. 30 + 24 . 70 + 18. 190 + 21. 60 + 23. 90 + 31. 110 = 2700 + 540 + 1680 + 3420 + +1260 + 2070 + 3410 = 15080$(*единиц*)

Ответ: Сравнивая три метода нахождения оптимального плана, делаем вывод, что метод потенциалов находит оптимальный план решения транспортной задачи, так как получили минимальные транспортные издержки равные 15080 единиц.

Задание 2.3.1.3. Практическая работа № 18 «Применение метода стрельбы для решения линейной краевой задачи. Задача о распределении средств между предприятиями. Задача о замене оборудования»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ПК2.1, ПК2.4., ПК2.5.*

Цель: научиться применять метод стрельбы для решения линейной краевой задачи

Краткая теория

Пример краевой задачи

Примером двухточечной краевой задачи является задача:

$$y'' = f(x, y, y'), \qquad 0 < x < 1,$$
$$y(0) = Y_0, \qquad y(1) = Y_1. \qquad (8.1)$$

с граничными условиями на обоих $0 \le x \le 1$,отрезка на котором концах надо найти

$y = y(x)$.

решение На этом примере мы схематически изложим некоторые способы численного решения краевых задач.

$f(x, y, y')$ y',

Если функция в (8.1) линейна по аргументам y и то мы имеем линейную краевую задачу, иначе — нелинейную краевую задачу.

Линейная краевая задача

Рассмотрим частную, но довольно распространенную краевую задачу следующего вида:

$$Ly = y'' - p(x)\, y = f(x), \qquad 0 < x < 1,$$
$$y(0) = Y_0, \qquad y(1) = Y_1. \qquad (8.2)$$

Для этой задачи проиллюстрируем два способа решения: один основан на идее численного построения общего решения линейного дифференциального уравнения, другой (конечно- разностный) сводит исходную дифференциальную краевую задачу к системе линейных алгебраических уравнений, решение которой находится методом прогонки.

Метод численного построения общего решения

Для нахождения решения краевой задачи можно численно построить решение дифференциального уравнения, представимое в виде

(8.2) $y(x) = C_1 y_1(x) + C_2 y_2(x) + y_0(x),$

$y_0(x)$где

$y'' - p(x)\, y = f(x),$

$y_1(x)$ $y_2(x)$

$y'' - p(x)\, y = 0.$ а и

однородного C_1 C_2

$y_0(x),\ y_1(x),\ y_2(x)$

решения условий — какое-либо решение неоднородного уравнения

— два любые линейно независимые уравнения Постоянные и находятся из граничных задачи (8.2).

Так как решения произвольны, то их можно построить различными способами. Например, можно задать какие-то начальные условия и решить одну задачу Коши для неоднородного и две задачи Коши для однородного уравнений. Эти условия, в частности, могут быть такими:

$y_0(0) = 0, \quad y_0'(0) = 0$ — для неоднородного уравнения;

$y_1(0) = 1, \quad y_1'(0) = 0;$

$y_2(0) = 0, \quad y_2'(0) = 1$ — для однородного уравнения.

$p(x) >> 1$

Однако при реализации этого способа, например, в случае для рассматриваемого уравнения могут возникнуть трудности, связанные с неустойчивостью задачи Коши. В этом

случае можно попытаться $y_0(x), \ y_1(x), \ y_2(x)$ построить с помощью решения одной краевой задачи для неоднородного уравнения и двух краевых задач для однородного уравнения. Краевые условия для этих задач могут быть, например, следующими:

$y_0(0) = 0, \quad y_0'(1) = 0$ — для неоднородного уравнения;

$y_1(0) = 1, \quad y_1'(1) = 0;$

$y_2(0) = 0, \quad y_2'(1) = 1$ — для однородного уравнения.

Эти задачи могут быть решены методом прогонки. Условия устойчивости метода прогонки при как легко проверить, выполнены. Этот подход может оказаться полезным, если $p(x) > 0$, краевые условия таковы, что для исходной задачи (8.2) метод прогонки применен быть не может.

Отметим, что с учетом специфики краевых условий исходной задачи можно строить общее решение вида

$$y(x) = y_0(x) + C y_1(x),$$

где $y_0(x)$ — некоторое решение неоднородного уравнения, а — некоторое решение однородного уравнения.

Конечно-разностный метод (метод прогонки)

При нахождении решения линейной краевой задачи:

$$y'' - p(x)\, y = f(x), \qquad 0 < x < 1,$$

$$y(0) = Y_0, \qquad y(1) = Y_1.$$

$p(x) >> 1$ для методом построения общего решения, если оно находится с помощью решения задач Коши, могут возникнуть трудности, связанные с вычислительной неустойчивостью задачи Коши.

Для решения поставленной задачи можно воспользоваться разностной схемой:

$$\frac{y_{m+1} - 2y_m + y_{m-1}}{h^2} - p(x_m)\, y_m = f(x_m),$$

$0 < m < M, \qquad Mh = 1, \qquad y_0 = Y_0, \qquad y_M = Y_1$

и решить разностную задачу методом прогонки. Условия применимости метода прогонки при как легко проверить, выполнены. Подробнее о методе прогонки см. в [1–4, 17,

$p(x) > 0,$

31]. В

[17] рассмотрены различные варианты метода прогонки.

Нелинейная краевая задача

Краевая задача

$$y'' = f(x, y, y'), \qquad 0 < x < 1,$$
$$y(0) = Y_0, \qquad y(1) = Y_1. \tag{8.3}$$

является нелинейной краевой задачей, если $f(x, y, y')$ функция нелинейна хотя бы по y', одному из аргументов y или

В настоящей работе реализованы два способа решения нелинейных краевых задач:

метод стрельбы и *метод линеаризации* (метод Ньютона), который сводит решение нелинейной краевой задачи к решению серии линейных краевых задач.

2.6. Метод стрельбы

Метод стрельбы для решения краевой задачи $y'' = f(x, y, y'), \quad 0 < x < 1,$ (8.3) базируется на том, что имеются удобные способы численного решения задачи Коши, т. е. задачи $(0, Y_0)$, следующего вида

$$y(0) = Y_0, \qquad (8.4)$$
$$y'(0) = \operatorname{tg}\alpha,$$

где Y_0 — ордината точки, из которой выходит интегральная кривая; α — угол наклона интегральной кривой к оси x при $x = 1$ выходе из точки $(0, Y_0)$ (рис. 8). При фиксированном α решение задачи (8.4) имеет вид $y = y(x, \alpha)$. При решении $y(x, \alpha)$ $y(x, \alpha)\big|_{x=1} = y(1, \alpha)$. зависит только от α:

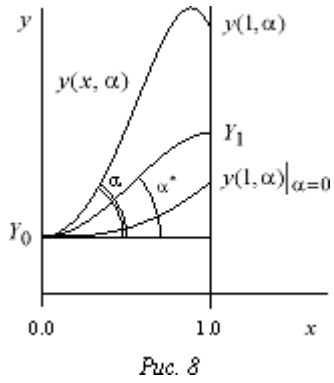


Рис. 8

Используя указанное замечание о решении задачи Коши (8.4), можно задачу (8.3) переформулировать следующим образом:

найти такой угол $\alpha = \alpha^*$, при котором интегральная кривая, выходящая из точки $(0, Y_0)$ под углом α, попадет в точку $(1, Y_1)$ оси абсцисс, то есть

$$y(1, \alpha) = Y_1. \qquad (8.5)$$

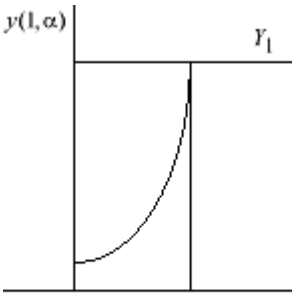


Рис. 9

Решение задачи (8.4) при этом совпадает с решением задачи (8.3). Таким образом, дело сводится к решению искомым уравнения (8.5) (рис. 9). Уравнение (8.5) — это уравнение вида

$$F(\alpha) = 0,$$

где $F(\alpha) = y(1, \alpha) - Y_1$.

Оно отличается от привычных уравнений лишь тем, что функция $F(\alpha)$ задана не аналитическим выражением, а с помощью алгоритма численного решения задачи (8.4).

Для решения уравнения (8.5) можно использовать любой метод, пригодный для уточнения корней нелинейного уравнения, например, метод деления отрезка пополам, метод Ньютона (касательных) и др. Метод Ньютона здесь предпочтительнее (если имеется достаточно хорошее начальное приближение) из-за высокой стоимости вычисления одного значения функции $F(\alpha)$ (нужно решить задачу Коши (8.4) с данным α).

Метод стрельбы, сводящий решение краевой задачи (8.3) к вычислению решений задачи Коши (8.4), хорошо работает в том случае, если решение $y(x, \alpha)$ «не слишком сильно» зависит от α. В противном случае он становится вычислительно неустойчивым, даже если решение задачи (8.3) зависит от входных данных «умеренно».

При решении уравнений $F(\alpha) = 0,$ методом деления отрезка α_0 α_1 пополам, мы задаем $y(1, \alpha_0) - Y_1$ $y(1, \alpha_1) - Y_1$ и так, чтобы разности и имели

$\alpha_2 = \frac{\alpha_0 + \alpha_1}{2}$.

разные знаки. Затем полагаем

$y(1, \alpha_2)$.

Вычисляем Затем вычисляем по одной из формул:

$\alpha_3 = \frac{\alpha_1 + \alpha_2}{2}$ $\frac{\alpha_0 + \alpha_2}{2}$ или

в зависимости от того, имеют ли $y(1, \alpha_2) - Y_1$ $y(1, \alpha_1) - Y_1$ разности и $y(1, \alpha_3)$.

соответственно разные или одинаковые знаки.

Затем $|y(1, \alpha_n) - Y_1| < \varepsilon$ вычисляем Процесс продолжаем до тех пор, пока не будет достигнута требуемая точность $F(\alpha) = 0$ α_0,

В случае использования для решения

затем последующие α_n уравнения метода Ньютона задаем а вычисляем по рекуррентной формуле

$$\alpha_{n+1} = \alpha_n - \frac{F(\alpha_n)}{F'(\alpha_n)},$$

$F'(\alpha_n)$ $n = 0, 1, \ldots$

Производная может быть вычислена по одной из формул численного дифференцирования, например, первого порядка аппроксимации:

$F'(\alpha_n) \approx \frac{F(\alpha_n + h) - F(\alpha_n)}{h}$.

Вычислительная неустойчивость задачи Коши

Поясним причину возникновения вычислительной неустойчивости на примере следующей линейной краевой задачи:

$$y'' - p^2 y = 0, \quad 0 < x < 1, \quad y(0) = Y_0, \quad y(1) = Y_1. \tag{8.6}$$

при p^2 постоянном Выпишем решение этой задачи:

с $y(x) = \frac{e^{-px} - e^{-p(2-x)}}{1 - e^{-2p}} Y_0 + \frac{e^{-p(1-x)} - e^{-p(1+x)}}{1 - e^{-2p}} Y_1$. ростом p остаются ограниченными на отрезке функциями; при всех они

не Коэффициенты Y_0 Y_1 при и превосходят единицу. Поэтому небольшие ошибки при задании и ведут к столь же $0 \le x \le 1$ небольшим погрешностям в решении, т. е. краевая задача является Y_0 Y_1 $p > 0$ «хорошей».

Рассмотрим теперь задачу Коши:

$$y'' - p^2 y = 0, \quad 0 < x < 1, \quad y(0) = Y_0, \quad y'(0) = \operatorname{tg}\alpha. \tag{8.7}$$

Ее решение имеет вид:

$$y(x) = \frac{pY_0 + \operatorname{tg}\alpha}{2p} e^{px} + \frac{pY_0 - \operatorname{tg}\alpha}{2p} e^{-px}.$$

$\operatorname{tg}\alpha$ □, то значение решения при $x = 1$ задании

допущена

погрешность получит приращение

$$\Delta y(1) = \frac{\varepsilon}{2p} e^{p} - \frac{\varepsilon}{2p} e^{-p}. \tag{8.8}$$

При больших p вычитаемое в равенстве (8.8) пренебрежимо мало, но коэффициент в первом $e^{p}/2p$ слагаемом становится большим. Поэтому метод стрельбы при решении задачи (8.6), будучи формально приемлемой процедурой, при больших p становится практически непригодным. Подробнее о возникновении неустойчивостей см. [1, 2].

Метод линеаризации (метод Ньютона)

Метод Ньютона сводит решение нелинейной краевой задачи к решению серии линейных краевых задач и состоит в следующем.

Пусть для нелинейной краевой задачи (8.3) известна $y_0(x)$, функция $y(x)$.

$y(x) = y_0(x) + y(x)$, удовлетворяющая граничным условиям и грубо приближенно равная искомому Положим

(8.9)

$v(x)$ где — поправка к нулевому приближению Подставим (8.9) в уравнение (8.8) и

y'' $f(x, y_0 + v, y_0' + v') = f(x, y_0, y_0') + \frac{\partial f(x, y_0, y_0')}{\partial y} v + \frac{\partial f(x, y_0, y_0')}{\partial y'} v' + O(v^2 + |v'|^2).$ $O(v^2 + |v'|^2),$ линеаризуем задачу, используя следующие равенства:

Отбрасывая $\tilde{v}(x)$: остаточный член получим линейную краевую задачу для нахождения поправки

$$\tilde{v}'' = p(x)\tilde{v}' + g(x)\tilde{v} + r(x), \tilde{v}(0) = 0, \quad \tilde{v}(1) = 0, \quad (8.10)$$

где

$$p(x) = \frac{\partial f(x, y_0, y_0')}{\partial y'}, \qquad q(x) = \frac{\partial f(x, y_0, y_0')}{\partial y},$$

$$r(x) = f(x, y_0, y_0') - y_0''.$$

$\tilde{v}$

Решая линейную краевую задачу (8.10) каким-либо численным методом, найдем поправку и примем за первое приближение

$$y_1(x) \equiv y_0(x) + \tilde{v}.$$

$y_1(x),$ $y(x) = y_1(x) + \tilde{v}_1$ Аналогично, зная приближение положим и найдем

$\max |\tilde{v}(x)| \le \varepsilon, \quad x \in [0, 1],$

следующее приближение. Продолжая процесс до тех пор, пока не будут выполнены неравенства задачи. где — требуемая точность, найдем приближенное решение исходной нелинейной

Задание

1. Начните выполнение работы с темы «*Линейная краевая задача*». Выбрав с помощью меню один из методов решения линейной краевой задачи, перейдите к пункту меню «*Параметры*». Наберите следующую краевую задачу:

$y'' - py = -p, \quad 0 < x < 1,$

$y(0) = 1, \quad y(1) = 1.$

$p = \text{const} > 0.$ $y \equiv 1.$ для Решением этой задачи является функция Установите значение шага сетки h = 0,05.

2. Найдите решение этой задачи методом построения общего решения и методом прогонки для разных p, начиная с умеренных значений и увеличивая их до величины порядка 1200. Сравните получаемые решения с точным и объясните наблюдаемые эффекты. Попытайтесь найти решение этой же задачи методом стрельбы. Проанализируйте, как влияет при разных p точность задания недостающего начального условия на левом конце интервала на успешное решение задачи методом стрельбы.

3. Объясните полученные результаты. Замените левое краевое условие $y'(0) = 0)$ (положите, например, и посмотрите, как изменится характер решения.

4. Выполните п. 1.1, 1.2 для задачи:

$y'' + py = p, \quad 0 < x < 1,$

$y(1) = 1, \quad y'(0) = 0.$

Ее точное $y \equiv 1$ решение Объясните полученные результаты. Найдите условие устойчивости метода прогонки для данной задачи.

5. Получите численное решение следующих нелинейных краевых задач:

2.1. $y'' + px\cos y = 0, \quad 0 < x < 1,$

$y'(0) = 0, \quad y(1) = 0,$ $p =$

$$y'' + \frac{0{,}5}{1 - 0{,}5y} y'^2 = 0, \qquad 0 < x < 1,$$

$y(0) = y_0, \quad y(1) = 0,$

$y_0 = 0{,}25;\ 0{,}5;\ 1;\ 1{,}5;\ 1{,}8;\ 1{,}9;\ 1{,}95;$

$$y'' + \sin y = 0, \qquad 0 < x < x_k,$$

$y(0) = 0, \quad y(x_k) = \pi,$

$x_k = 0{,}5;\ 1;\ 2;\ 4;\ 6.$

6. Рассмотрите следующие краевые задачи:

3.1. $y'' = e^y, \quad 0 < x < 1,$

$y(0) = 1, \quad y(1) = a;$

3.2. $y'' = -e^y, \quad 0 < x \le 1,$

$y(0) = 1, \quad y(1) = a.$

Параметр a меняется от 0 до 2. Что при этом происходит с решением задач? Почему в задаче 3.2 при значениях $a > 1{,}4999\ldots$ не работает метод линеаризации?

Критерии оценивания:

Оценка «**отлично**» – полный объем выполненных работ, выполнены все задания лабораторной работы, продемонстрировано глубокое понимание материала самостоятельность выполнения работы, аккуратность и законченность работы.

Оценка «**хорошо**» –все задания лабораторной работы выполнены, работа оформлена с незначительными отклонениями от требований, продемонстрировано хорошее понимание материала

Оценка «удовлетворительно» - задания лабораторной работы выполнены с замечаниями, работа имеет существенные отклонения в оформлении, продемонстрировано базовое понимание материала.

Оценка «неудовлетворительно» – отсутствие полного объема работ; в работе допущены серьёзные ошибки и нарушение всех перечисленных выше требований.

Задание 2.3.1.4. Лабораторная работа «Построение простейших математических моделей. Построение простейших статистических моделей»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ПК2.1, ПК2.4., ПК2.5.*

Цель работы: закрепить практические навыки по построению простейших математических и простейших статистических моделей.

Краткая теория

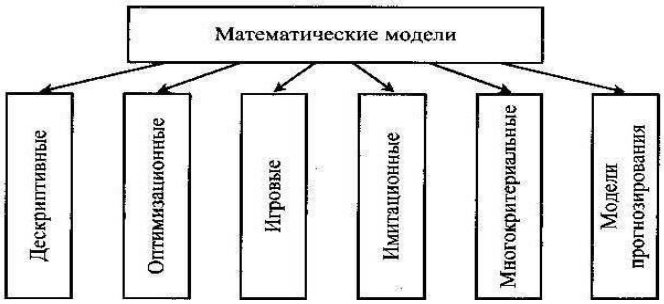
Построение математической модели процесса, явления или объекта начинается с построения упрощенного варианта модели, в котором учитываются только основные черты. В результате прослеживаются основные связи между входными параметрами, ограничениями и показателем эффективности. Общего подхода к построению модели нет. В каждом конкретном случае при построении математической модели учитывается большое количество факторов: цель построения модели, круг решаемых задач, точность описания модели и точность выполнения вычислений. Математическая модель должна отражать все существенные факторы, определяющие ее поведение, и при этом быть простой и удобной для восприятия результатов. Каждая математическая модель процесса, явления или объекта в своей основе имеет математический количественный метод.

Применение математических количественных методов для обоснования выбора того или иного управляющего решения во всех областях человеческой деятельности называется *исследованием операций.* Целью исследования операций является нахождение с

использованием специального математического аппарата решения, удовлетворяющего заданным условиям. На самом деле при решении практически любой задачи имеется неограниченное количество решений. Множество решений, удовлетворяющих заданным условиям (ограничениям), называется допустимым множеством решением. Выбор из множества допустимых решений одного решения, наилучшего в каком-либо смысле, называемого *оптимальным* решением, и есть задача исследования операций.

Модель — это материальный или идеальный объект, заменяющий оригинал, наделенный основными характеристиками (чертами) оригинала и предназначенный для проведения некоторых действий над ним с целью получения новых сведений об оригинале.

Рис. 1. Классификация моделей



При построении математической модели необходимо обеспечить *достаточную* точность вычислений (точность решения) и *необходимую* подробность модели. Любая математическая модель включает в себя описание основных, т. е. *необходимых* для исследования свойств и законов функционирования исследуемого объекта, процесса или явления. В своей основе каждая математическая модель имеет целевую функцию, которая описывает функционирование реального объекта, процесса или явления. В зависимости от исследуемого (моделируемого) объекта, явления или процесса *целевая функция* может быть представлена одной функциональной зависимостью, системой уравнений (линейных, нелинейных, дифференциальных и т. д.), набором статистических данных и т. д. При работе с целевой функцией исследователь воздействует на нее через ***набор входных параметров***

Входной параметр 1		Выходной параметр 1
Входной параметр 2		Выходной параметр *2*
Входной параметр 3	Модель системы	Выходной параметр 3
Входной параметр *n*- 1	(объекта или процесса)	Выходной параметр *m* - 1
Входной параметр *n*		Выходной параметр *m*

По способу реализации математические модели можно разделить следующим образом.

1. Линейное программирование.

Математическая модель целиком (целевая функция и ограничения) описывается уравнениями первого порядка. Линейное программирование включает в себя несколько методов решения (задач):

- симплексный;
- графический;
- транспортная задача;
- целочисленное программирование.

2. Нелинейное программирование.

Целевая функция и ограничения, составляющие математическую модель, содержат хотя бы одно нелинейное уравнение (уравнение второго порядка и выше). Нелинейное программирование содержит несколько методов решения (задач):

- графический;
- регулярного симплекса;
- деформируемого многогранника (Нелдера - Мида);
- градиентный.

3. Динамическое программирование.

Ориентировано на решение задач прокладки магистралей кратчайшим путем и перераспределения различных видов ресурсов.

4. Сетевое планирование.

Решает проблему построения графика выполнения работ, распределения производственных, финансовых и людских ресурсов.

5. Принятие решений и элементы планирования.

В этом случае и качестве целевой функции выступает набор статистических данных или некоторые данные прогноза. Решением задачи являются рекомендации о способах поведения (стратегии). Решение носит рекомендательный характер (приблизительное решение). Выбор стратегии целиком остается за человеком — ответственным лицом, принимающим решение.

Для принятия решения разработаны следующие теории:

- теория игр;
- системы массового обслуживания.

Порядок выполнения заданий

Задание 1. Составить математическую модель следующей задачи. На складе имеется 300 кг сырья. Надо изготовить два вида продукции. На изготовление первого изделия требуется 2 кг сырья, а на изготовление второго изделия — 5 кг. Определить план выпуска двух изделий.

Решение.

Обозначим, x1 – единица первого изделия, x2 – единица второго изделия. Тогда составим математическая модель: 2x1+5x2=300.

Задание 2. Составить математическую модель следующей задачи. Предположим, что для производства продукции вида A и B можно использовать материал 3-х сортов. При этом на изготовление единицы изделия вида A расходуется 14 кг первого сорта, 12 кг второго сорта и 8 кг третьего сорта. На изготовление продукции вида B расходуется 8 кг первого сорта, 4 кг второго сорта, 2 кг третьего сорта. На складе фабрики имеется всего материала первого сорта 624 кг, второго сорта 541 кг, третьего сорта 376 кг. От реализации единицы готовой продукции вида A фабрика имеет прибыль вида 7 руб., а от реализации единицы готовой продукции вида B фабрика имеет прибыль вида 3 руб. Определить максимальную прибыль от реализации всей продукции видов A и B.

Решение

Составим математическую модель задачи:

Пусть x_1 – единица готовой продукции вида А,

x_2 - единица готовой продукции вида В,

Цель фабрики получить максимальную прибыль от реализации всей продукции видов А и В, тогда:

$$F = 7 \cdot x_1 + 3 \cdot x_2 \to \max$$

Система ограничений:

$$\begin{cases} 14x_1 + 8x_2 \le 624 \\ 12x_1 + 4x_2 \le 541 \\ 8x_1 + 2x_2 \le 376 \end{cases}$$

$x_1 \ge 0, \quad x_2 \ge 0$ *условие неотрицательности*

Задание 3. Составить математическую модель следующей задачи. Имеются три пункта поставки однородного груза А1, А2, А3 и пять пунктов В1, В2, В3, В4, В5 потребления этого груза. На пунктах А1, А2 и А3 находится груз соответственно в количестве 200, 450, 250 тонн. В пункты В1, В2, В3, В4, В5 требуется доставить соответственно 100, 125, 325, 250, 100 тонн груза. Расстояние между пунктами поставки и пунктами потребления приведено в таблице:

Пункты поставки	**Пункты потребления**				
	B1	**B2**	**B3**	**B4**	**B5**
A1	5	8	7	10	3
A2	4	2	2	5	6
A3	7	3	5	9	2

Решение:

1. Проверка сбалансированности модели задачи. Модель является сбалансированной, т.к. суммарный объем запасов сырья равен суммарному объему потребности в ней:
   200+450+250=100+125+325+250+100.
2. Построение математической модели – неизвестными в этой задачи является объем перевозок. Пусть x_{ij} - объем перевозок с *i*-го предприятия в *j*-го пункт потребления.

Суммарные транспортные расходы - это функционал качества (критерий цели):

$$F = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij} x_{ij},$$

Где c_{ij} - стоимость перевозки единицы продукции с *i*-го предприятия в *j*-й пунктах потребления.

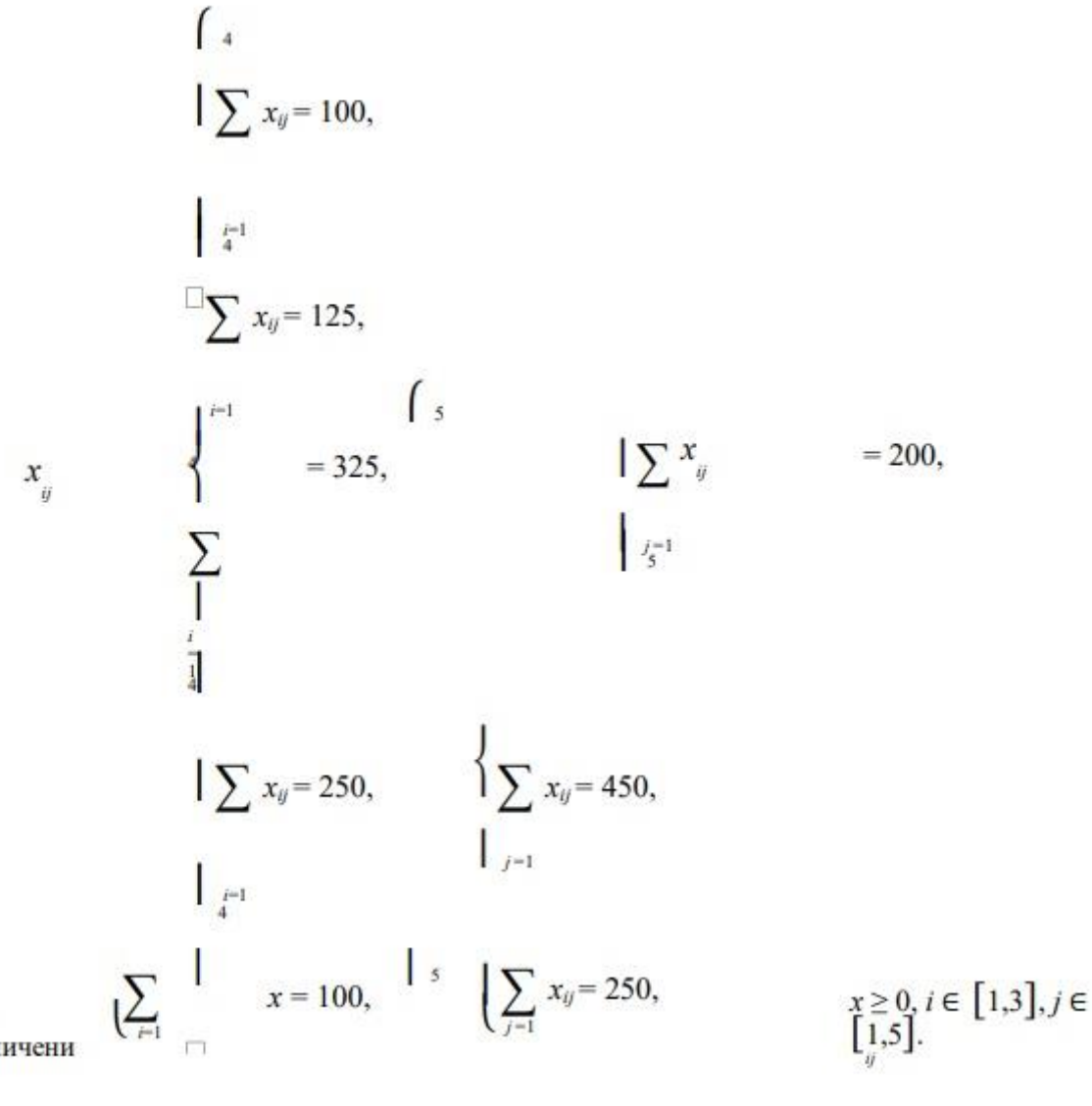
Неизвестные в этой задачи должны удовлетворять следующим ограничениям:

- Объем перевозок не могут быть отрицательными;
- Поскольку модель сбалансирована, то вся продукция должна быть вывезена с предприятия, а потребность всех пунктов потребления должна быть полностью удовлетворены.

Итак, имеем следующую задачу:

- Найти минимум функционала:

$$F = \sum_{i=1}^{4}\sum_{j=1}^{5} c_{ij} x_{ij} \to \min,$$



Задание 2.2.2.5. Лабораторная работа «Решение простейших однокритериальных задач»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ПК2.1, ПК2.4., ПК2.5.*

Цель: определить оптимальное решение однокритериальных и многокритериальных задач в простейших случаях.

Краткая теория

В зависимости от вида показателя эффективности различают задачи принятия решений по скалярному показателю (однокритериальные задачи) и задачи принятия решений по векторному показателю (многокритериальные задачи). Задачами математического программирования называют однокритериальные задачи оптимизации. Методы их решения оперируют с детерминированными математическими моделями. В этих моделях отражены разнообразные проблемы распределения ограниченных ресурсов в экономике, военном деле, создании новой техники и т.д. Пути решения этих проблем, так или иначе, связаны с планированием целенаправленной деятельности, т.е. с разработкой определенных установок на будущее

Задача математического программирования формулируется следующим образом: найти значения $x_1, x_2, \ldots, x_n$ переменных, доставляющие максимум (минимум) заданной $y = f(x_1, x_2, \ldots, x_n)$ целевой функции при условиях:

$$g_j(x_1, x_2, \ldots, x_n) \leq (\geq, =) b_j, (j = \overline{1, m}).$$

Различают два вида задач математического программирования:

1. Задачи линейного программирования.
2. Задачи нелинейного программирования.

В первых задачах функция и ограничения линейны относительно переменных .

Во вторых задачах целевая функция и (или) условия имеют разного рода нелинейности.

Графоаналитический метод решения задач оптимизации

Этим методом вручную решаются простые задачи оптимизации. Математические модели в этих задачах не должны быть сложными, т.к. в противном случае требуется много времени для их решения. Для начала рассмотрим однопараметрическую однокритериальную задачу оптимизации.

Постановка задачи: Дан один критерий . Объект (процесс) описан уравнением (уравнениями), включающими один искомый параметр . Имеется система ограничений

1) $x \geq a_1$;

2) $a_2 \leq x \leq b_1$;

и т.д.

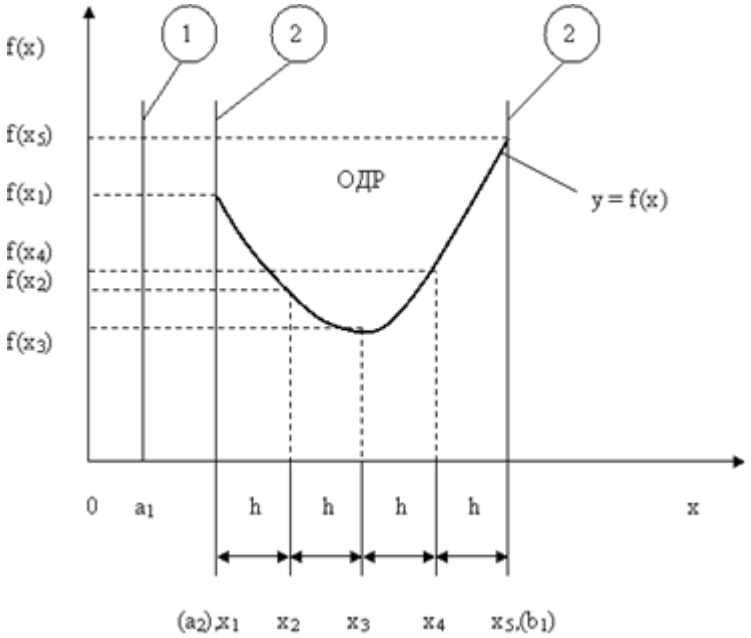
Необходимо найти оптимальное значение $x - x_{ОПТ}$ параметра , обращающее целевую функцию $f(x)$ в максимум или минимум.

Задача решается в два этапа:

1. Построение области допустимых решений (ОДР).
2. Нахождение в пределах ОДР оптимального решения.

При построении ОДР на первом этапе рассматривается система ограничений. Все ограничения должны быть выполнены. Выполнение первого ограничения в приведенной выше постановке задачи оптимизации означает, что искомое значение параметра x должно находиться правее a_1 причем, в a_1 разрешенный интервал входит (рис.1). Выполнение второго ограничения означает, что искомое значение параметра x должно находиться в интервале (на $[a_2 b_1]$ отрезке)

следует иметь в виду, что границы интервала в интервал входят.

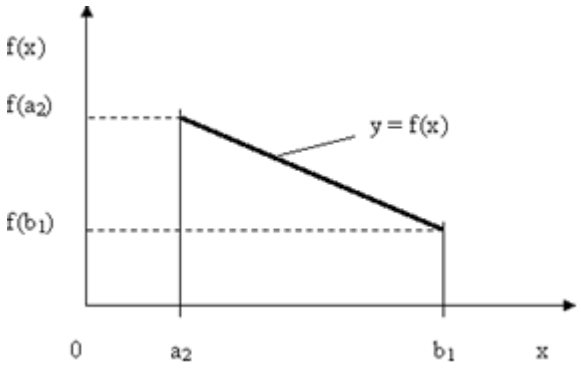


Когда однопараметрическая однокритериальная задача оптимизации решается с применением графоаналитического метода вручную, то на втором этапе применяют метод перебора. Суть его заключается в следующем. В пределах ОДР через определенный интервал h выбирается ряд значений параметра xВ рассматриваемом нами случае ОДР разбита на четыре отрезка, и выбрано пять значений параметра xДля этих значений параметраx рассчитываются соответствующие значения целевой функции. Среди них находят минимальное (максимальное) значение. Значение параметра ,x_iобращающее целевую функцию в минимум (максимум), является оптимальным. Если в рассматриваемом нами$f(x)$случае стремится к минимуму, то

$x_{опт} = x_3$, если к максимуму,$x_{опт} = x_5$то .

При решении практических задач оптимизации всегда следует иметь в виду, какова целевая функция. Это значительно упрощает работу как при решении задач оптимизации вручную с применением графоаналитического метода, так и при решении таких задач с использованием компьютерных программ. Причем, это относится и к случаю использования готовых программ, и, что особенно важно, к разработке собственных программ.

Рассмотрим, например, следующий частный случай, когда целевая функция линейная



В данном случае на втором этапе вычисляют значения целевой функции только на границах ОДР. Эти значения сравнивают и выбирают наименьшее или наибольшее. Для примера, приведенного на рис. 2, если , то , если , то .

В задачах, как правило, присутствует не один, а несколько признаков предпочтения (критериев). Такие задачи называются многокритериальными. Критерии могут оказаться противоречивыми, т.е. решение, лучшее по определенному признаку, может оказаться худшим по другому признаку.

Например, минимизация стоимости и максимизация качества товара почти всегда противоречивы. В этом случае задача отыскания решения, предпочтительного по всем признакам, будет некорректной, т.е. не будет иметь ни одного решения.

В случае противоречивых критериев имеются следующие подходы к отысканию подходящего решения.

1) Замена некоторых критериев ограничениями вида ≤ или ≥ . Например, минимизация стоимости f (x) → min, может быть заменена ограничением вида f (x) ≤ A, где A некоторая верхняя оценка стоимости, т.е. максимально допустимая стоимость.

2) Свертка критериев. Создается один глобальный скалярный критерий, целевая функция которого является некоторой функцией от исходных целевых функций. Наиболее употребимыми являются линейные свертки вида αf(x) + βg(x) (в случае двух критериев). Нетривиальной является задача отыскания адекватных значений коэффициентов α и β , отражающих относительную важность целевых функций f (x) и g (x).

3) Ранжирование критериев. Критерии ранжируются по степени важности.

4) Отыскание решений, лучших хотя бы по одному критерию.

Подходы 1) и 2) приводят к однокритериальной задаче.

Подход 3) приводит к задаче с упорядоченными критериями.

Подход 4) приводит к задаче с независимыми критериями. В задаче с упорядоченными критериями критерии упорядочиваются по важности, и требуется найти оптимальное решение для наименее важного критерия на множестве решений, оптимальных для более важного критерия

ому критерию, далее вложено множество оптимальных решений по второму по важности критерию, и т.д.

В задаче с независимыми критериями требуется найти множество недоминируемых (эффективных) решений. Недоминируемое решение лучше любого другого допустимого решения хотя бы по одному критерию либо не хуже по всем критериям.

Множество недоминируемых решений также называется множеством Парето.

Задание 1. Решить графическим способом задачу. Для производства двух видов, изделии P_1 и P_2 используется, три вида сырья S_1, S_2, S_3, запасы которого соответственно равны 100, 60, 180 единиц. Для производства одной единицы продукции P_1 используется 2 единицы сырья S_1 и по 1 единице сырья $S_2 u S_3$. Для производства одной единицы продукции P_2 используется по 1 единице сырья $S_1 u S_2$ и 4 единицы сырья S_3. Прибыль от реализации 1 единицы каждой продукции P_1 и P_2 соответственно равна 30 и 20 единиц. Необходимо составить такой план выпуска продукции P_1 и P_2, при котором суммарная прибыль будет наибольшей.

Задание 2.2.2.6. Лабораторная работа «Задача Коши для уравнения теплопроводности»

Проверяемые результаты обучения: ОК1, ОК2, ОК03, ОК 04, ОК05, ПК4.1

Цель: определить оптимальное решение однокритериальных и многокритериальных задач в простейших случаях.

Краткая теория

Многие задачи науки и техники сводятся к решению обыкновенных дифференциальных уравнений (ОДУ). ОДУ называются такие уравнения, которые содержат одну или несколько производных от искомой функции. В общем виде ОДУ можно записать следующим образом:

$F(x, y, y', y'', \ldots, y^{(n)}) = 0$ $y^{(i)}$

, где x – независимая

переменная, - i-ая производная от

искомой функции. n - порядок уравнения. Общее решение ОДУ n– $y = \varphi(x, c_1, \ldots, c_n)$го порядка

содержит n

$c_1, \ldots, c_n$

произвольных постоянных , т.е. общее решение имеет вид .

Для выделения единственного решения необходимо задать n дополнительных условий. В зависимости от способа задания дополнительных условий существуют два различных типа задач: задача Коши и краевая задача. Если дополнительные условия задаются в одной точке, то такая задача называется задачей Коши. Дополнительные условия в задаче Коши называются начальными условиями. Если же дополнительные условия задаются в более чем одной точке, т.е. при различных значениях независимой переменной, то такая задача называется краевой. Сами дополнительные условия называются краевыми или граничными.

Ясно, что при n=1 можно говорить только о задачи Коши.

Примеры постановки задачи Коши:

$$\frac{dy}{dx} = x^2 y^3, \quad y(1) = 1$$

$$\frac{d^2y}{dx^2} = \frac{dy}{dx} + xy^2, \quad y(1) = 1, \; y'(1) = 0$$

Примеры краевых задач:

$$\frac{d^2y}{dx^2} + 2\frac{dy}{dx} - y = \sin(x), \quad y(0) = 1, \; y(1) = 0$$

$$\frac{d^3y}{dx^3} = x + x\frac{d^2y}{dx^2} - \frac{dy}{dx}, \quad y(1) = 0, \; y'(1) = 0, \; y(3) = 2$$

Решить такие задачи аналитически удается лишь для некоторых специальных типов уравнений.

Единственность решения задачи Коши для уравнения теплопроводности в классе ограниченных функций.

Теорема 1.

Пусть *D* - ограниченная область в R*n*. Если решение *u*(*x*,*t*) смешанной задачи для уравнения теплопроводности

ut(*x*,*t*)−*a*2(*x*,*t*)Δ*xu*(*x*,*t*)=*f*(*x*,*t*), (*x*,*t*)∈*G*=*D*×(0;*T*)

с начальными условиями *u*(*x*,0)=*u*0(*x*), *u*(0)∈*C*(*D*¯) с граничными условиями первого рода *u*(*x*,*t*)|*x*∈∂*D*=*v*(*x*,*t*), *v*(*x*,*t*)∈*C*(∂*D*×[0;*T*])

существует в классе функций *C*2,1*x*,*t*(*G*)∩*C*(*G*¯), то оно единственно в этом классе и

непрерывно зависит от начальных и граничных данных (в равномерной метрике).

Доказательство теоремы 1. Единственность. Пусть *u*~ и *u*^ - решение задачи. Тогда их разность *u*=*u*~−*u*^ удовлетворяет однородному уравнению теплопроводности с однородными начальными и граничными условиями:

ut(*x*,*t*)=*a*2(*x*,*t*)Δ*xux*(*x*,*t*), (*x*,*t*)∈*G*, *u*(*x*,0)=0, *u*(*x*,*t*)|*x*∈∂*D*=0.

Согласно принципу максимума в ограниченной области выполняются неравенства 0⩽*u*(*x*,*t*)⩽0 (*x*,*t*)∈*G*¯

Следовательно *u*~=*u*^ в *G*¯.

Непрерывная зависимость. Пусть теперь *u*~ и *u*^ - решение задачи Коши отвечающие различным начально-краевым данным: *u*0~,*u*0^ и *v*~,*v*^ соответственно.

Тогда разность $u=u\sim-u\wedge$ является решением смешанной задачи для однородного уравнения теплопроводности

$ut(x,t)=a2(x,t)\Delta xu(x,t)$, $(x,t)\in G$ с начальными условиями $u(x,0)=u0\wedge-u0\sim$ и граничными условиями $u(x,t)|x\in\partial D=v\sim(x,t)-v\wedge(x,t)$.

Воспользуемся для функции u принципом максимума, получаем оценку $|u\sim(x,t)-u\wedge(x,t)|\leqslant\max\{\sup D^{-}|u0\sim-u0\wedge,\sup\partial D\times[0;T]|v\sim-v\wedge|\}$, $(x,t)\in G^{-}$,

что означает непрерывную зависимость решения от начальных и краевых данных в равномерной метрике. Из принципа единственности максимума в неограниченной области вытекает следующее

Теорема 2.

Если решение задачи Коши

$ut(x,t)-a2(x,t)\Delta xu(x,t)=f(x,t)$, $(x,t)\in G=\mathrm{R}n\times(0;T)$, $u(x,0)=u0(x)$, $u0\in C(\mathrm{R}n)$

с ограниченными начальными данными $u0$ существует в классе функций $C2,1x,t(G)\cap C(G^{-})\cap B(G^{-})$, то оно единственно в нем и непрерывно зависит от начальных данных.

Задания для самостоятельной работы

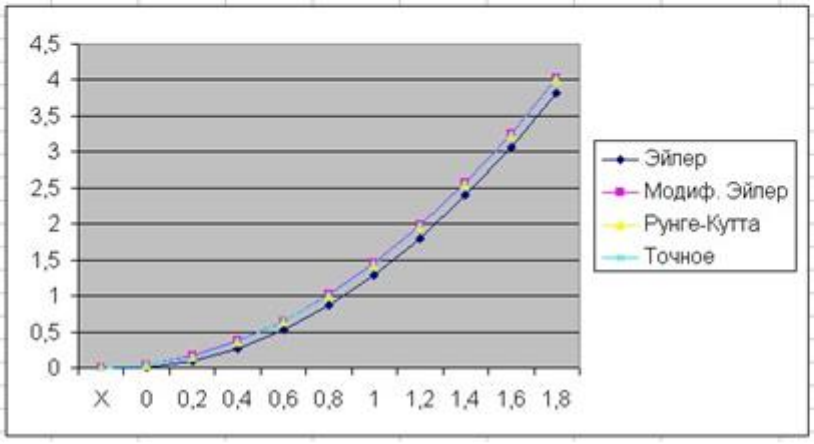
Решение обыкновенных дифференциальных уравнений (ОДУ)

В приведѐнном примере решается задача Коши, то есть, ищется решение дифференциального уравнения первого порядка вида dy/dx = f(x,y) на интервале $x \in [x_0,x_n]$ при условии $y(x_0)=y_0$ и равномерном шаге сетки по x.

Решение выполняется методами Эйлера, "предиктор-корректор" (он же модифицированный метод Эйлера) и методом Рунге-Кутта 4 порядка точности. Пример может служить образцом для Ваших решений, правда, функцию придѐтся перепрограммировать несколько раз при различных значениях аргумента - поскольку без применения макросов на VBA Excel не позволяет создать полноценную функцию, которую было бы удобно вызывать с разными значениями аргументов.

Здесь решается уравнение $dy/dx = 2x-y+x^2$ на интервале [0,2], начальное значение y(0)=0, для оценки точности задано также точное решение в виде функции $u(x)=x^2$. Оценка погрешности делается в норме L_1, как и принято в данном случае.

	A	B	C	D	E	F	G	H	I	J	K	L
1	X	Эйлер	y~	Модиф. Эйлер	k1	k2	k3	k4	Рунге-Кутта	Точное		
2	0	0	0	0					0	0	A=	0
3	0,2	0	0	0,044	0	0,21	0,189	0,4022	0,040006667	0,04	B=	2
4	0,4	0,088	0,1232	0,16728	0,399993	0,609994	0,588994	0,802195	0,160012125	0,16	N=	10
5	0,6	0,2624	0,325824	0,36997	0,799988	1,009989	0,988989	1,20219	0,360016594	0,36	H=	0,2
6	0,8	0,52192	0,607976	0,652175	1,199983	1,409985	1,388985	1,602186	0,640020252	0,64		
7	1	0,865536	0,96974	1,013984	1,59998	1,809982	1,788982	2,002183	1,000023248	1		
8	1,2	1,292429	1,411187	1,455467	1,999977	2,209979	2,188979	2,402181	1,440025701	1,44		
9	1,4	1,801943	1,932373	1,976683	2,399974	2,609977	2,588977	2,802179	1,960027709	1,96		
10	1,6	2,393554	2,533346	2,57768	2,799972	3,009975	2,988975	3,202177	2,560029353	2,56		
11	1,8	3,066844	3,214144	3,258497	3,199971	3,409974	3,388973	3,602176	3,240030699	3,24		
12	2	3,821475	3,974798	4,019168	3,599969	3,809972	3,788972	4,002175	4,000031801	4		
13	Погрешн.	0,178525		0,019168					3,18007E-05			



Критерии оценивания:

Оценка **«отлично»** – полный объем выполненных работ, выполнены все задания лабораторной работы, продемонстрировано глубокое понимание материала самостоятельность выполнения работы, аккуратность и законченность работы.

Оценка **«хорошо»** –все задания лабораторной работы выполнены, работа оформлена с незначительными отклонениями от требований, продемонстрировано хорошее понимание материала

Оценка **«удовлетворительно»** - задания лабораторной работы выполнены с замечаниями, работа имеет существенные отклонения в оформлении, продемонстрировано базовое понимание материала.

Оценка **«неудовлетворительно»** – отсутствие полного объема работ; в работе допущены серьёзные ошибки и нарушение всех перечисленных выше требований.

Тема 2.3.2. Задачи в условиях неопределенности

Задание 2.3.2.1. Практическая работа № 19 «Нахождение кратчайших путей в графе. Решение задачи о максимальном потоке»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ПК2.1, ПК2.4., ПК2.5.*

Цель: решить задачу о нахождении кратчайших путей в графе. Решить задачу о нахождении максимального потока.

Краткая теория

Граф это множество точек или вершин и множество линий или ребер, соединяющих между собой все или часть этих точек. Вершины, прилегающие к одному и тому же ребру, называются смежными. Если ребра ориентированы, что обычно показывают стрелками, то они называются дугами, и граф с такими ребрами называется ориентированным графом. Если ребра не имеют ориентации, граф называется неориентированным.

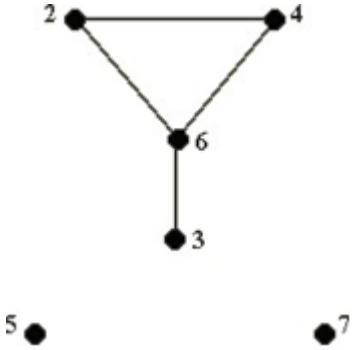
Графы обычно изображаются в виде геометрических фигур, так что вершины графа изображаются точками, а ребра - линиями, соединяющими точки (рис. 1).

Петля это дуга, начальная и конечная вершина которой совпадают.

Простой граф - граф без кратных ребер и петель.

Степень вершины это удвоенное количество петель, находящихся у этой вершины плюс количество остальных прилегающих к ней ребер.

Пустым называется граф без ребер. Полным называется граф, в котором каждые две вершины смежные.



Путь в ориентированном графе — это последовательность дуг, в которой конечная вершина всякой дуги, отличной от последней, является начальной вершиной следующей.

Маршрут в графе путь, ориентацией дуг которого можно пренебречь.

Цепь маршрут, в котором все ребра попарно различны.

Цикл замкнутый маршрут, являющийся цепью.

Маршрут, в котором все вершины попарно различны, называют простой цепью. Цикл, в котором все вершины, кроме первой и последней, попарно различны, называются простым циклом.

Подграф графа это граф, являющийся подмоделью исходного графа, т.е. подграф содержит некоторые вершины исходного графа и некоторые ребра (только те, оба конца которых входят в подграф).

Подграф называется остовным подграфом, если множество его вершин совпадает с множеством вершин самого графа.

Граф называется связным, если любая пара его вершин связана. Связными компонентами графа называются подграфы данного графа, вершины которых связаны.

Дерево — это связный граф без циклов. Деревья особенно часто возникают на практике при изображении различных иерархий. Например, популярны генеалогические деревья.

Граф без цикла называется лесом. Вершины степени 1 в дереве называются листьями. Деревья - очень удобный инструмент представления информации самого разного вида. Деревья отличаются от простых графов тем, что при обходе дерева невозможны циклы. Это делает графы очень удобной формой организации данных для различных алгоритмов.

Очевидно, что графический способ представления графов непригоден для ПК. Поэтому существуют другие способы представления графов. В теории графов применяются

1 .Матрица инцидентности. Это матрица A с n строками, соответствующими вершинам, и m столбцами, соответствующего рѐбрам. Для ориентированного графа столбец, соответствующий дуге (x,y) содержит - 1 в строке, соответствующей вершине x и 1, в строке, соответствующей вершине y. Во всех остальных 0. Петлю, т.е. дугу (x,x) можно представлять иным значением в строке x, например, 2. Если граф неориентированный, то столбец, соответствующий ребру (x,y) содержит 1, соответствующие x и y и нули во всех остальных строках.

2. Матрица смежности. Это матрица n×n где n - число вершин, где bij = 1, если существует ребро, идущее из вершины x в вершину y и bij = 0 в противном случае.

Нахождение минимального остова в графе

Алгоритм решения

1.Упорядочить ребра графа по возрастанию весов;

2.Выбрать ребро с минимальным весом, не образующее цикл с ранее выбранными ребрами. Занести выбранное ребро в список ребер строящегося остова;

3.Проверить, все ли вершины графа вошли в построенный остов. Если нет, то выполнить пункт 2.

Нахождение кратчайшего пути в графе

Пусть дан граф, дугам которого приписаны веса. Задача о нахождении кратчайшего пути состоит в нахождении кратчайшего пути от заданной начальной вершины до заданной конечной вершины, при условии, что такой путь существует.

Данная задача может быть разбита на две:

1.для начальной заданной вершины найти все кратчайшие пути от этой вершины к другим;

2.найти кратчайшие пути между всеми парами вершин.

Рассмотрим алгоритм решения для задачи первого типа:

Необходимо найти путь от s - начальной вершины до t - конечной вершины. Каждой вершине присваиваем пометки I(Xi).

1.I(s) = 0, I(Xi) равно бесконечности для всех Xi не равных s и считать эти пометки временными. Положить p = s.

2.Для всех Xi, принадлежащих Г(p) и пометки которых временны, изменить пометки по следующему правилу:

I(Xi) = min[I(Xi), I(p) + c(p, Xi)]

3.среди всех вершин с временными пометками найти такую, для которой I(Xi*) = min[I(Xi)]

4.считать пометку вершины Xi* постоянной и положить p = Xi*.

5.если p = t, то I(p) является длинной кратчайшего пути, если нет, перейти к шагу 2. Как только все пометки расставлены, кратчайшие пути получают, используя соотношение I(Xi') + c(Xi',Xi) = I(Xi) (1).

Для решения задачи второго типа можно применять данный алгоритм для каждой вершины.

Порядок выполнения заданий

Задача 1. Составить матрицы инцидентности и смежности для графа:

Матрица инцидентности

	u	***v***	***w***
a	1	0	0
b	0	0	1
c	1	1	1
d	0	1	0

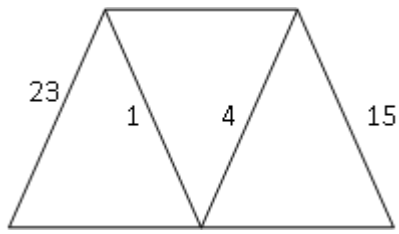
Матрица смежности

	a	***b***	***c***	***d***
a	0	0	1	0
b	0	0	1	0
c	1	1	0	1
d	0	0	1	0

Где ***u, v, w*** – ребра данного графика

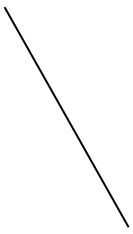
Задача 2. На представленном графе найдите: а) минимальный остов дерева, б) найдите кратчайший путь от начальной точки X1 до всех остальных точек.

X2 20 X3



X4

36 X5 9



Из оставшихся элементов выбираем минимальный - ***(X3,X5) = 4***. Элемент обводим кружком. Чтобы выполнялось условие **2** пункты **X2** и **X3** не должны соединяться, поэтому элемент ***(X2,X3)*** зачѐркивается. И т.д.

Задание 2.3.2.2. Практическая работа № 20 «Составление систем уравнений Колмогорова. Нахождение финальных вероятностей. Нахождение характеристик простейших систем массового обслуживания. Решение задач массового обслуживания методами имитационного моделирования»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ПК2.1, ПК2.4., ПК2.5.*

Цель: отработать и закрепить умения составлять системы уравнений Колмогорова, отработать и закрепить умения находить финальные вероятности, отработать и закрепить умения определите основные показатели СМО.

Краткая теория

Марковский случайный процесс

Построение математических моделей в условиях неопределенности - очень сложная или невыполнимая задача. Лишь для некоторых упрощенных случаев можно построить математическую модель.

Следует различать два вида неопределенности:

- вероятностные характеристики либо известны, либо могут быть получены в результате эксперимента. Такая неопределенность называется стохастической, и для большинства объектов, содержащих такую неопределенность, можно построить математическую модель, например выход из строя оборудования, приход нового клиента и т. д.

- вероятностные характеристики определить невозможно. В этом случае задачу можно попытаться решить с помощью экспертных оценок, но результат будет весьма приблизительным, например, каковы будут модели женской одежды через пять лет?

Строгую математическую модель с аналитическим вычислением всех интересующих величин можно построить только в том случае, если случайный процесс носит марковский характер.

Случайный процесс будет марковским, если вероятностные характеристики процесса в момент времени t зависят только от текущего (настоящего) состояния процесса в этот момент времени t и не зависят от того, как (каким способом и когда) рассматриваемый процесс перешел в текущее состояние.

Из всего многообразия марковских процессов хорошо изучены и представляют большой практический интерес марковские случайные процессы с дискретными состояниями и непрерывным временем.

Под дискретным состоянием будем понимать, что процесс переходит из одного состояния в другое скачкообразно за очень короткое время (практически мгновенно), и количество этих состояний известно (фиксировано).

Под непрерывным временем будем понимать такое, при котором переход из одного допустимого состояния в другое допустимое состояние происходит в произвольные моменты времени, т. е. заранее не определенные.

Потоки событий. Однородные события, следующие друг за другом в произвольные моменты времени (случайно), называются потоком событий (или входным потоком заявок). Примерами потоков событий могут быть: поток пассажиров в авиакассе, поток посетителей парикмахерской, поток отказов технического устройства и т.д. Здесь под событием понимается факт поступления заявок на обработку (приход покупателя, наличие отказа технического средства, поступление телефонного вызова и т.д.), а не результат его обработки (как это рассматривается в теории вероятностей). Поэтому в системах массового обслуживания вероятностными характеристиками будет обладать не отдельное событие, а интервал времени. λ Интенсивностью потока событий называется среднее число событий за единицу времени. λ

Интенсивность может быть как числом постоянным (константой), так и величиной, зависящей от времени t. Например, количество пассажиров в городском транспорте в «часы пик» резко увеличивается по сравнению с другим временем суток.

Финальные вероятности состояний

Будем рассматривать марковские процессы с дискретными состояниями и непрерывным временем.

Пример 1 : Техническое устройство состоит из трѐх узлов и в любой момент времени может находиться в одном из восьми состояний

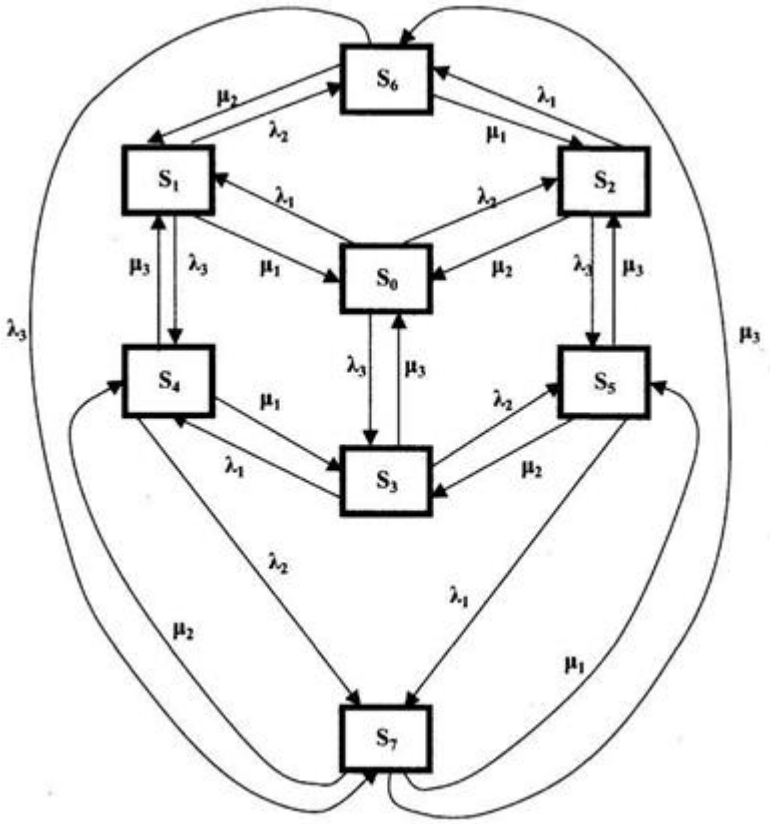


Рис. Состояния технического устройства

Возможные состояния устройства таковы:

S0 — все три узла исправны;

S1— первый узел неисправен, второй и третий исправны;

S2 — второй узел неисправен, первый и третий исправны;

S3 — третий узел неисправен, первый и второй исправны; S4 — первый и третий узлы неисправны, второй исправен; S5 — второй и третий узлы неисправны, первый исправен;

S6 — первый и второй узлы неисправны, третий исправен; S7 — все три узла неисправны.

Размеченным графом будем считать такой граф, у которого стрелками указаны переходы из одного состояния в другое, а рядом со стрелкой указана интенсивность перехода. Будем различать

λ μ

две интенсивности — прямую , и обратную .

Тогда λ1 , λ2 и 3 интенсивности потоков

—

1

третьего узлов, а узлов. μ , μ и отказов соответственно первого, второго и

3 соответственно

—

интенсивности потоков возвратов

(ремонтов)

Если для ремонта каждого узла имеется отдельный специалист, то среднее время ремонта каждого узла есть величина постоянная и не имеет значения, один или несколько узлов вышли из строя.

На основе построенного размеченного графа (см. рис. 1) создадим математическую модель.

Наше техническое устройство в соответствии с построенным графом в любой момент времени будет находиться в одном из восьми возможных состояний. Обозначим вероятность каждого i-го состояния как pi(t), тогда

$$\sum_{i=1}^{n} p_i(t) = 1.$$

Для определения вероятности каждого состояния технического устройства составим соответствующие дифференциальные уравнения:

$$\begin{cases} \frac{dp_0(t)}{dt} = -(\mu_{10} + \mu_{20} + \mu_{30} + \lambda_{12} + \lambda_{13}) p_0 + \mu_{11} p_1 + \mu_{22} p_2 + \mu_{33} p_3; \\ \frac{dp_1(t)}{dt} = \lambda_{12} p_0 - (\mu_{11} + \lambda_{21} + \lambda_{23}) p_1; \\ \frac{dp_2(t)}{dt} = \lambda_{13} p_0 - (\mu_{22} + \lambda_{31} + \lambda_{32}) p_2; \\ \frac{dp_3(t)}{dt} = \lambda_{23} p_1 + \lambda_{32} p_2 - (\mu_{33} + \lambda_{34}) p_3; \\ \frac{dp_4(t)}{dt} = \lambda_{34} p_3 - (\mu_{44} + \lambda_{45}) p_4; \\ \frac{dp_5(t)}{dt} = \lambda_{45} p_4 - (\mu_{55} + \lambda_{56}) p_5; \\ \frac{dp_6(t)}{dt} = \lambda_{56} p_5 - (\mu_{66} + \lambda_{67}) p_6; \\ \frac{dp_7(t)}{dt} = \lambda_{67} p_6 - (\mu_{77} + \lambda_{78}) p_7; \\ \frac{dp_8(t)}{dt} = \lambda_{78} p_7 - (\mu_{88} + \mu_{89}) p_8; \\ \frac{dp_9(t)}{dt} = \mu_{89} p_8 - (\mu_{99} + \mu_{90}) p_9. \end{cases}$$

Эта система дифференциальных уравнений называется системой уравнений Колмогорова.

Имеем систему из восьми линейных дифференциальных уравнений с восемью неизвестными.

Известно, что сумма всех вероятностей равна единице, т. е.

p0 +pl +p2 +p3 +p4 +p5+p6 +p7 =1

Таким образом, любое из уравнений, входящее в систему уравнений, можно записать, используя последнее уравнение, и найти значения вероятностей для каждого события.

Для облегчения процесса составления дифференциальных уравнений можно применить следующее правило:

В левой части каждого уравнения следует записать производную вероятности г-го состояния устройства.

В правой части сумма произведений потока событий, входящих в текущее состояние, умноженная на вероятность состояния, из которого исходит поток, минус суммарная интенсивность исходящих потоков событий из текущего состояния, умноженная на вероятность текущего состояния.

Если финальные вероятности существуют: $\lim_{t \to \infty} p_i(t) = p_i$ при $i = 1, 2, 3, ..., n$,

то их сумма будет равна единице: $\sum_{i=1}^{n} p_i = 1.$

Финальные вероятности показывают, какое среднее время устройство будет находиться в каждом состоянии. Финальные вероятности находятся из системы дифференциальных уравнений, если их правые части приравнять нулю.

Задания:

Вариант 1

1. Техническое устройство состоит из трѐх узлов и в любой момент времени может находиться в одном из восьми состояний (рис. 1).Численные значения интенсивности потоков событий: λ1=2; λ2=2; λ3=1; μ1=4; μ2=4; μ3=2. Найдите финальные вероятности состояний устройства.

2. Интенсивность потока автомобилей, поступающих на моечную станцию (одноканальная СМО) – 5 автомобиля в час, а интенсивность обслуживания – 6 автомобилей в час.

Предполагая, что станция работает в стационарном режиме, найти среднее число автомобилей, находящихся на станции, среднюю длину очереди и среднее время ожидания обслуживания.

3. Какое оптимальное число линий обслуживания должна иметь СМО, если λ = 3, μ = 2, c1 = 4, c2 = 2.

4. Определить число взлетно-посадочных полос для самолѐтов с учетом требования, что вероятность ожидания P(w > 0) должна быть меньше, чем 0,06. Интенсивность потока равна 28 требований в сутки и интенсивность линий обслуживания – 32 самолѐтов в сутки.

Вариант 2

1. Техническое устройство состоит из трѐх узлов и в любой момент времени может находиться в одном из восьми состояний (рис. 1).Численные значения интенсивности потоков событий: λ1=2; λ2=1; λ3=1; μ1=4; μ2=2; μ3=2. Найдите финальные вероятности состояний устройства.

2. Интенсивность потока автомобилей, поступающих на моечную станцию (одноканальная СМО) – 6 автомобиля в час, а интенсивность обслуживания – 7 автомобилей в час.

Предполагая, что станция работает в стационарном режиме, найти среднее число автомобилей, находящихся на станции, среднюю длину очереди и среднее время ожидания обслуживания.

3. Какое оптимальное число линий обслуживания должна иметь СМО, если $\lambda = 4$, $\mu = 2$, $c1 = 5$, $c2 = 2$.

4. Определить число взлетно-посадочных полос для самолѐтов с учетом требования, что вероятность ожидания $P(w > 0)$ должна быть меньше, чем 0,06. Интенсивность потока равна 30 требований в сутки и интенсивность линий обслуживания – 34 самолѐтов в сутки.

Вариант 3

1. Техническое устройство состоит из трѐх узлов и в любой момент времени может находиться в одном из восьми состояний (рис. 1).Численные значения интенсивности потоков событий: $\lambda1=1$; $\lambda2=2$; $\lambda3=2$; $\mu1=4$; $\mu2=4$; $\mu3=4$. Найдите финальные вероятности состояний устройства.

2. Интенсивность потока автомобилей, поступающих на моечную станцию (одноканальная СМО) – 4 автомобиля в час, а интенсивность обслуживания – 5 автомобилей в час.

Предполагая, что станция работает в стационарном режиме, найти среднее число автомобилей, находящихся на станции, среднюю длину очереди и среднее время ожидания обслуживания.

3. Какое оптимальное число линий обслуживания должна иметь СМО, если $\lambda = 2$, $\mu = 1$, $c1 = 3$, $c2 = 2$.

4. Определить число взлетно-посадочных полос для самолѐтов с учетом требования, что вероятность ожидания $P(w > 0)$ должна быть меньше, чем 0,08. Интенсивность потока равна 28 требований в сутки и интенсивность линий обслуживания – 32 самолѐтов в сутки.

Вариант 4

1. Техническое устройство состоит из трѐх узлов и в любой момент времени может находиться в одном из восьми состояний (рис. 1).Численные значения интенсивности потоков событий: $\lambda1=2$; $\lambda2=2$; $\lambda3=2$; $\mu1=2$; $\mu2=2$; $\mu3=4$. Найдите финальные вероятности состояний устройства.

2. Интенсивность потока автомобилей, поступающих на моечную станцию (одноканальная СМО) – 8 автомобиля в час, а интенсивность обслуживания – 9 автомобилей в час.

Предполагая, что станция работает в стационарном режиме, найти среднее число автомобилей, находящихся на станции, среднюю длину очереди и среднее время ожидания обслуживания.

3. Какое оптимальное число линий обслуживания должна иметь СМО, если $\lambda = 7$, $\mu = 8$, $c1 = 4$, $c2 = 2$.

4. Определить число взлетно-посадочных полос для самолѐтов с учетом требования, что вероятность ожидания $P(w > 0)$ должна быть меньше, чем 0,06. Интенсивность потока равна 18 требований в сутки и интенсивность линий обслуживания – 22 самолѐтов в сутки.

Контрольные вопросы:

1. Дайте определение марковскому процессу.
2. Какие типы неопределенностей встречаются.
3. Дайте определение потоку событий.
4. Как составить уравнения Колмогорова.
5. Какие виды СМО Вы знаете?
6. При каких предположениях изучаются одноканальные СМО с отказами?
7. Почему стационарный режим в одноканальных СМО с ожиданием существует только при условии $\varphi > 0$?
8. Какие средние характеристики можно рассчитать в одноканальных СМО с ожиданием?

Задание 2.3.2.3. Практическая работа № 21 «Построение прогнозов. Решение матричной игры методом итераций. Игры с природой»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ПК2.1, ПК2.4., ПК2.5.*

Цель: научиться выбирать оптимальную стратегию игры

Краткая теория

Два предприятия производят продукцию и поставляют еѐ на рынок региона. Они являются единственными поставщиками продукции в регион, поэтому полностью определяют рынок данной продукции в регионе.

Каждое из предприятий имеет возможность производить продукцию с применением одной из трѐх различных технологий. В зависимости от качества продукции, произведѐнной по каждой технологии, предприятия могут установить цену единицы продукции на уровне 12, 8 и 4 денежных единиц соответственно. При этом предприятия имеют различные затраты на производство единицы продукции.

Таблица

Затраты на единицу продукции, произведенной на предприятиях региона (д.е.).

Технология	Цена реализации единицы продукции, д.е.	Полная себестоимость единицы продукции, д.е.	
		Предприятие А	Предприятие В
1	12	8	10
2	8	5	4
3	4	2	1

В результате маркетингового исследования рынка продукции региона была определена функция спроса на продукцию: **Y = 10 – 0,6*X,**

где Y – количество продукции, которое приобретѐт население региона (тыс. ед.), а X – средняя цена продукции предприятий, д.е.

Значения долей продукции предприятия А, приобретенной населением, зависят от соотношения цен на продукцию предприятия А и предприятия В. В результате маркетингового исследования эта зависимость установлена и значения вычислены (табл. 2).

Таблица 2.Доля продукции предприятия А, приобретаемой населением в зависимости от соотношения цен на продукцию

Цена реализации 1 ед. продукции, д.е.		Доля продукции А, предприятия купленной населением
Предприятие А	Предприятие В	
12	12	0,31
12	8	0,33
12	4	0,18
8	12	0,7

8	8	0,3
8	4	0,2
4	12	0,92
4	8	0,85
4	4	0,72

В задаче необходимо определить:

1. Существует ли в данной задаче ситуация равновесия при выборе технологий производства продукции обоими предприятиями?
2. Существуют ли технологии, которые предприятия заведомо не будут выбирать вследствие невыгодности?
3. Сколько продукции будет реализовано в ситуации равновесия? Какое предприятие окажется в выигрышном положении?

Критерии оценивания:

Оценка «отлично» – полный объем выполненных работ, выполнены все задания лабораторной работы, продемонстрировано глубокое понимание материала самостоятельность выполнения работы, аккуратность и законченность работы.

Оценка «хорошо» –все задания лабораторной работы выполнены, работа оформлена с незначительными отклонениями от требований, продемонстрировано хорошее понимание материала

Оценка «удовлетворительно» - задания лабораторной работы выполнены с замечаниями, работа имеет существенные отклонения в оформлении, продемонстрировано базовое понимание материала.

Оценка «неудовлетворительно» – отсутствие полного объема работ; в работе допущены серьёзные ошибки и нарушение всех перечисленных выше требований.

Задание 2.3.2.4. Лабораторная работа «Моделирование прогноза»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ПК2.1, ПК2.4., ПК2.5.*

Цель: изучение возможностей и формирование умения использования универсальной компьютерной технологии для решения задач выявления тенденций и прогнозирования развития процесса на основе моделирования рядов динамики (с помощью табличного процессора Excel)

Краткая теория

Тренд – это функция заданного вида, с помощью которой можно аппроксимировать построенный по данным таблицы график. Тренд служит для выявления тенденций развития процесса, представленного в виде диаграммы, и обеспечивает прогноз на заданный период.

В MS Excel предусмотрено несколько стандартных типов тренда: линейный, логарифмический, степенной, экспоненциальный, полиномиальный, скользящее среднее. Необходимые условия для построения тренда:

− период времени, за который изучается исследуемый процесс, должен быть достаточным для выявления закономерности;

− тренд в анализируемый период должен развиваться эволюционно;

− процесс, представленный диаграммой, должен обладать определенной инертностью.

− Тренд можно строить для диаграмм типа:

− линейчатый график,

− гистограмма,

− диаграмма с областями,

− XY-точеная диаграмма.

При установлении наиболее подходящего типа регрессионной зависимости для описания процесса изменения показателей какой-либо величины используют показатель достоверности описания функции. Тип регрессионной линии считается установленным, если величина достоверности аппроксимации R2=1. Однако, если аппроксимации R2 <0,6 уместно говорить о том, что тип зависимости для описания процесса изменения показателя не подходит.

Если ни в одном из вариантов исследуемых типов регрессионных линий (трендов) величина достоверности аппроксимации не равна единице, то выбирают тот тип, для которого величина достоверности аппроксимации максимальна.

Задание

На основании приведенных данных построить тренды и проанализировать, как описывают процесс динамики продаж линейная, логарифмическая, полиномиальная, степенная и экспоненциальная зависимости. Рассчитать прогноз на основе аппроксимирующих зависимостей, а также с помощью функций ПРЕДСКАЗ, РОСТ и ТЕНДЕНЦИЯ. Провести анализ с целью определения, какой из примененных методов дает более точный результат. Постановка задачи.

Имеются две наблюдаемые величины x и y, например, объем реализации фирмы, торгующей кондитерскими изделиями, за ряд лет ее работы. Необходимо выяснить какая из наиболее распространенных функциональных зависимостей подходит для описания процесса реализации товара, и какого результата по объемам продаж можно ожидать в последующие годы работы фирмы. Для того чтобы построить прогноз развития какой-либо ситуации на практике зачастую необходимо знать закономерность изменения исследуемой величины или объекта.

Для выявления тенденций развития процесса продаж необходимо построить тренды и осуществить их анализ. Построим и проанализируем, как описывают процесс динамики продаж линейная, логарифмическая, полиномиальная, степенная и экспоненциальная зависимости.

Технология работы

1.В MS Excel создайте рабочую книгу с листами: Прогнозирование, Линейная, Логарифмическая, Полиномиальная, Степенная, Экспоненциальная и оформите лист Прогнозирование как показано на рисунке

	A	B	C	D	E	F	G	H
1			**Прогнозирование объема продаж предприятия**					
2								
3				Метки трендов на диаграммах				
4								
5								
6			Объем продаж					
7			Статистические данные	Теоретические данные				
8		Год	Объем продаж	линейная аппроксимация	логарифмическая аппроксимация	полиномиальная аппроксимация 2 степени	степенная аппроксимация	экспоненциальная аппроксимация
9	1	1996	149					
10	2	1997	145					
11	3	1998	168					
12	4	1999	146					
13	5	2000	177					
14	6	2001	176					
15	7	2002	190					
16	8	2003	186					
17	9	2004	176					
18	10	2005	211					
19	11	2006	170					
20	***Контрольные суммы***							
21	**ПРОГНОЗ на 2007 год**							
22	12	2007						
23	12	2007	ПРЕДСКАЗ					
24	12	2007	РОСТ					
25	12	2007	ТЕНДЕНЦИЯ					
26								
27								
28								
29								
30								

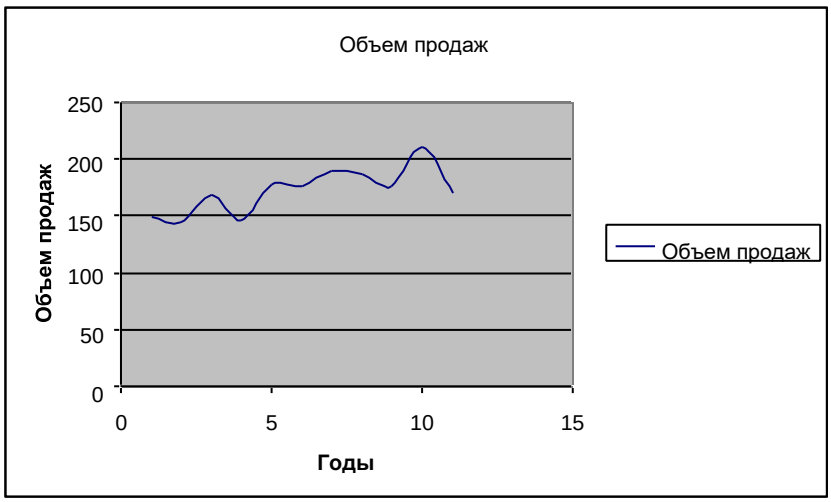
Прогнозирование | Линейная | Логарифмическая | Полиномиальная | Степенная

Для правильности последующих вычислений в Excel необходимо, чтобы значения периодов были представлены их номерами, начиная с 1 (ячейки A9:A19).

2.Исходным пунктом моделирования трендов является построение диаграммы.

На основе исходных данных, представленных в таблице, постройте точечную диаграмму со значениями, соединенными сглаживающими линиями без маркеров.

Для построения использовать Мастер диаграмм. Выберите подтип диаграммы «Точечную диаграмму со значениями, соединенными сглаживающими линиями без маркеров». Если в левом нижнем углу диалогового окна Мастер диаграмм нажать и удерживать кнопку «Просмотр результата», то справа вместо галереи видов вы увидите образец будущей диаграммы. В качестве диапазонов значений для построения диаграммы взять несмежные диапазоны ячеек A8:A19 и C8:C19. В третьем шаге Мастера диаграмм на вкладке Заголовки обозначьте ось X заголовком «Годы», а ось У – заголовком «Объем продаж». На этом же шаге расположите легенду внизу. На четвертом шаге поместите диаграмму на имеющемся листе.



3.Для свободного размещения на графике текстовых меток тренда, содержащих вид уравнения и коэффициент детерминации (величина достоверности аппроксимации R2), предварительно занесите график в буфер обмена и скопируйте его в начало других пяти листов (Линейная, Логарифмическая, Полиномиальная, Степенная, Экспоненциальная). Если у вас в книге недостает листов, выполните их вставку.

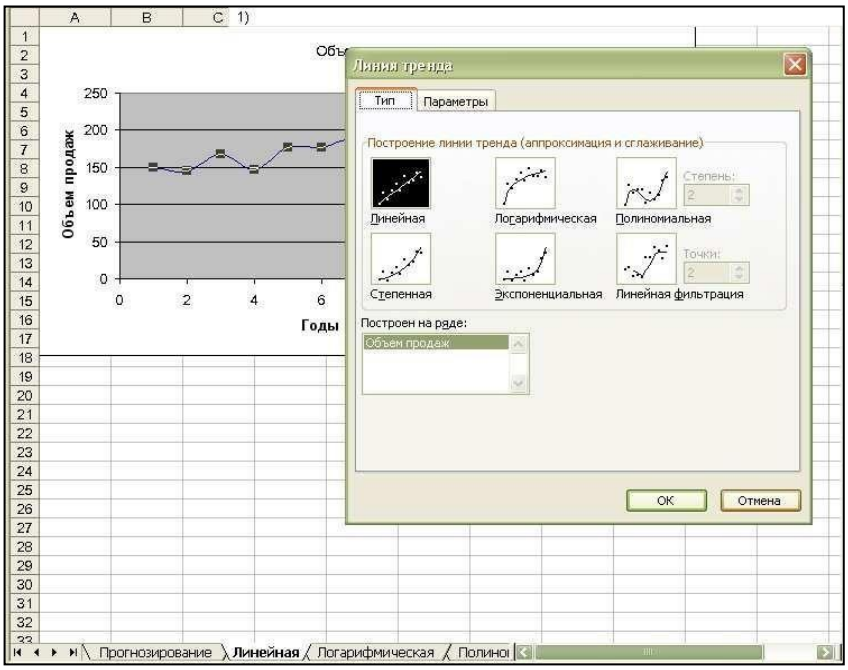
4.Построить линейный тренд для диаграммы. Для этого необходимо:

− установить указатель мыши на линии диаграммы и щелкнуть левой кнопкой мыши так, чтобы на линии появились черные метки

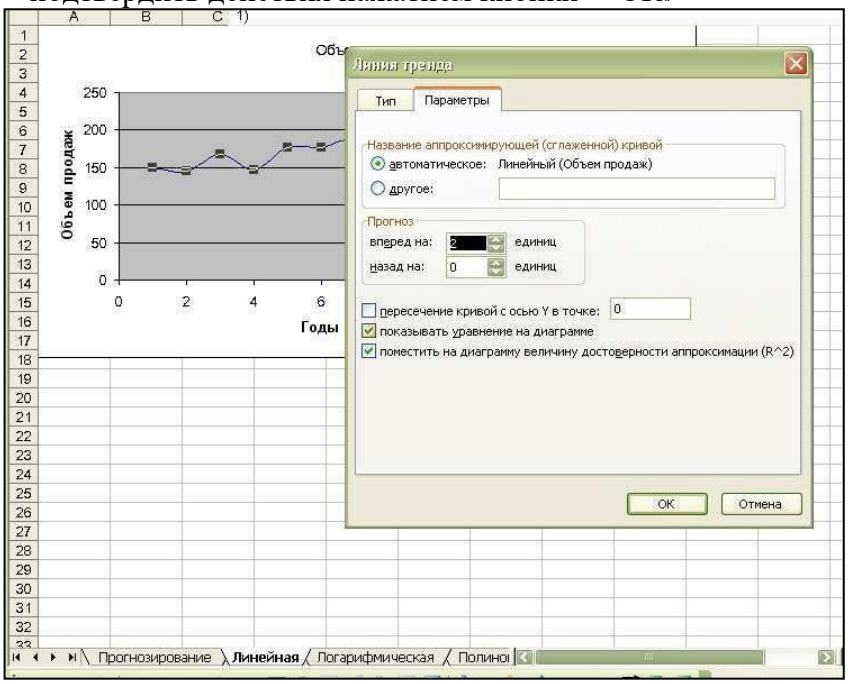
− для выделенной диаграммы вызвать контекстное меню, щелкнув правой кнопкой мыши;

− выполнить команду Добавить линию тренда.

− в диалоговом окне Линия тренда на вкладке Тип выбрать окно Линейная

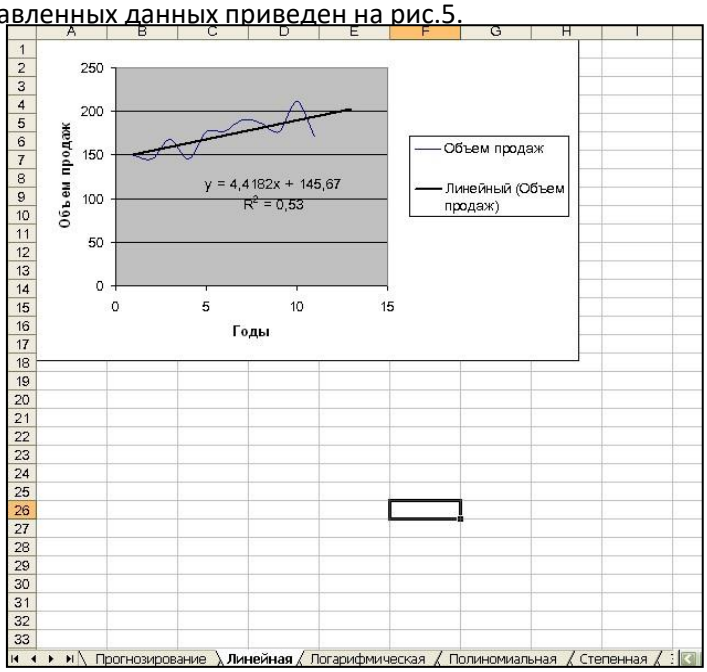


Параметры установить следующие параметры
− название аппроксимирующей кривой: автоматическое
− прогноз: вперед на 2 периода;
− показывать уравнение на диаграмме: установите флажок;
− поместить на диаграмму величину достоверности аппроксимации: установите флажок.
− подтвердить действия нажатием кнопки ―ОК‖

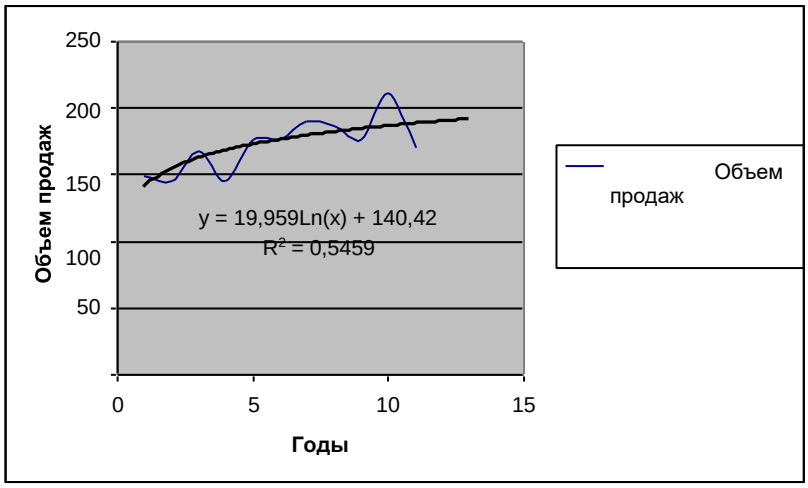


Произвести настройку оформления вида полученного тренда в области рабочего листа ―Тренды‖, отведенной для представления диаграмм. Итог оформления графически

представленных данных приведен на рис.5.



5.Перейдите на лист Логарифмическая. Постройте аналогичным образом логарифмический тренд для диаграммы.



Постройте полиномиальный тренд для диаграммы. Для этого необходимо:

− установить указатель мыши на линии диаграммы и щелкнуть левой кнопкой мыши так, чтобы на линии появились черные метки

− для выделенной диаграммы вызвать контекстное меню, щелкнув правой кнопкой мыши

− выполнить команду Добавить линию тренда.

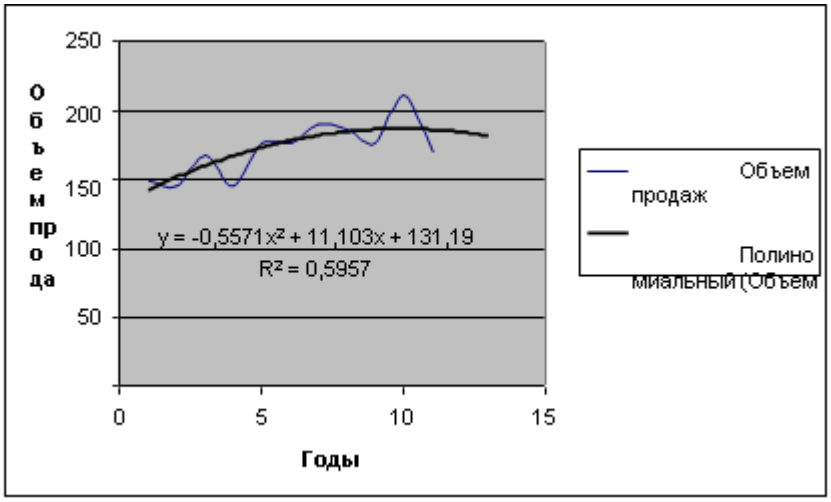
− в диалоговом окне Линия тренда на вкладке Тип выбрать окно полиномиальная, установите для полинома степень 2 на вкладке Параметры установить следующие параметры:

− название аппроксимирующей кривой: автоматическое

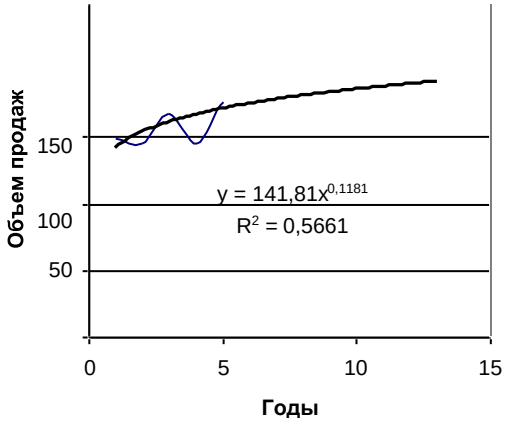
− прогноз: вперед на 2 периода;

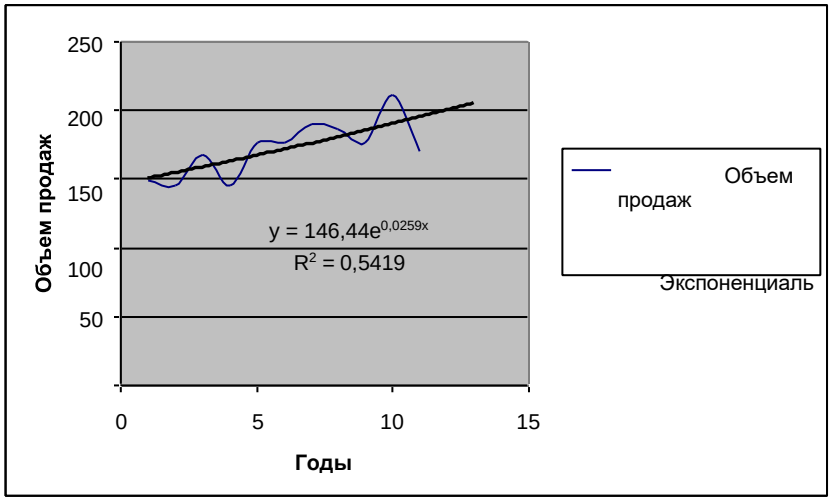
− показывать уравнение на диаграмме: установите флажок;

− поместить на диаграмму величину достоверности аппроксимации: установите флажок.



7.Аналогичным образом построить степенной и экспоненциальный тренды для диаграммы на соответствующих листах книги Excel.

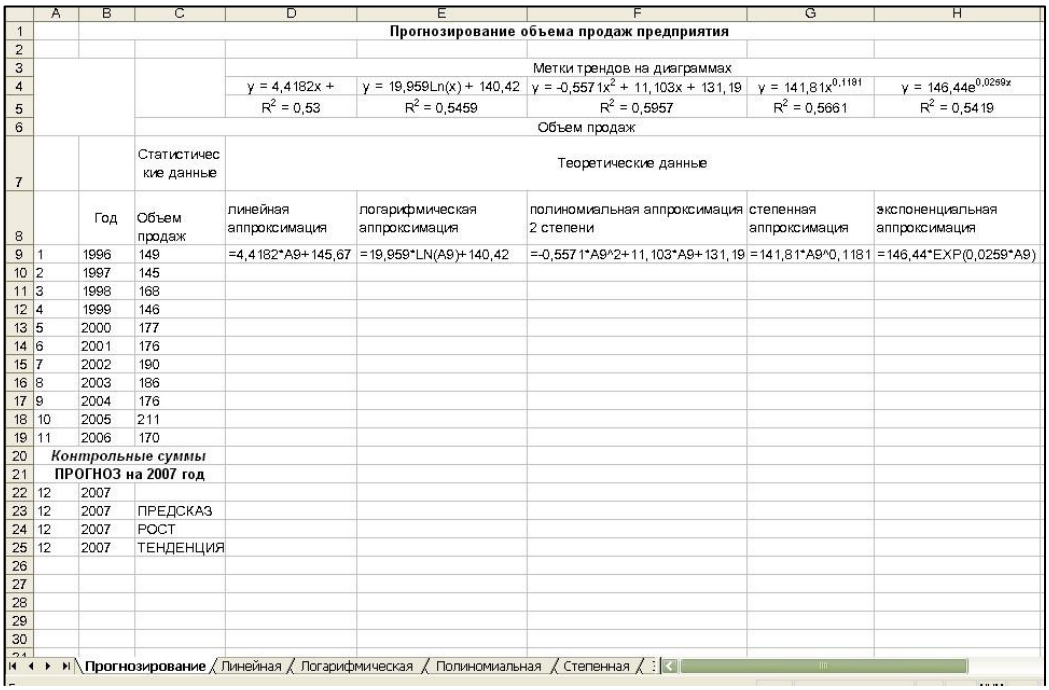




Конечный результат моделирования должен оцениваться пользователем с точки зрения здравого смысла на основе неформального комплекса знаний об условиях развития процесса, о допустимых предельных значениях показателя и т.п. В Excel для анализа трендов автоматически выводится только коэффициент детерминации (R2). Статистики-практики применяют метод сверки контрольных сумм теоретического (сглаженного по тренду) ряда признака с суммой значений исходного ряда. Однако для подсчета этих сумм сначала необходимо построить ряды теоретических значений показателя по найденным уравнениям трендов.

8. Перейдите на лист Прогнозирование.

Скопируйте метки трендов с диаграмм и вставьте их в соответствующие ячейки как показано на рис.



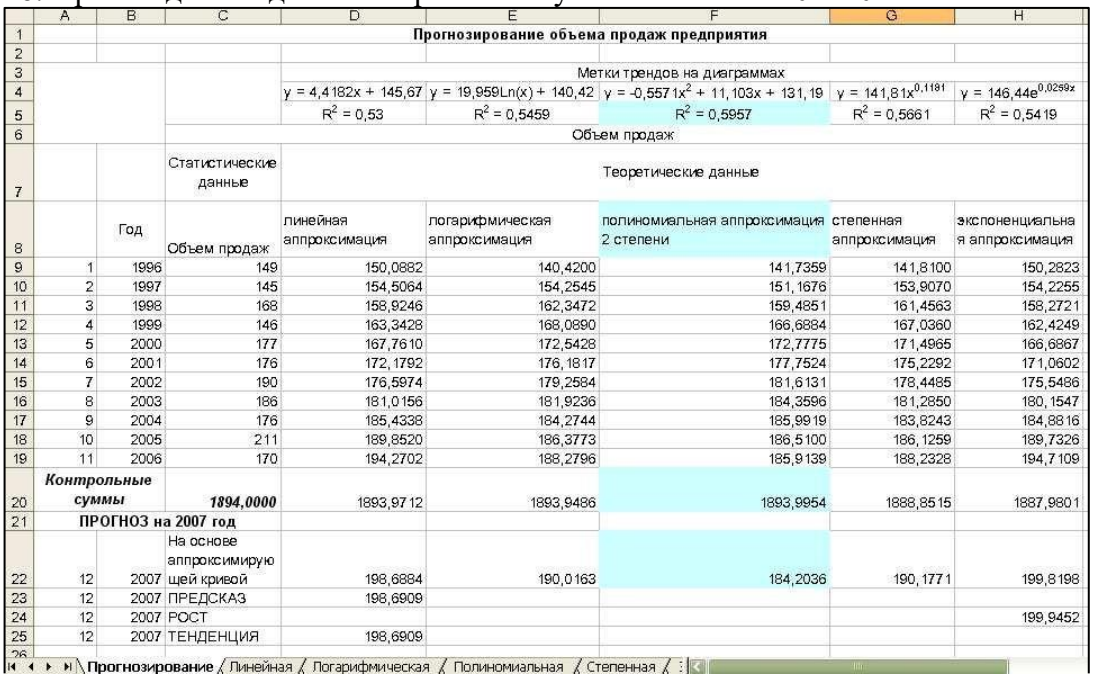
	A	B	C	D	E	F	G	H
1				Прогнозирование объема продаж предприятия				
2								
3				Метки трендов на диаграммах				
4				y = 4,4182x +	y = 19,959Ln(x) + 140,42	y = -0,5571x^2 + 11,103x + 131,19	y = 141,81x^{0,1181}	y = 146,44e^{0,0259x}
5				R^2 = 0,53	R^2 = 0,5459	R^2 = 0,5957	R^2 = 0,5661	R^2 = 0,5419
6				Объем продаж				
7			Статистические данные	Теоретические данные				
8		Год	Объем продаж	линейная аппроксимация	логарифмическая аппроксимация	полиномиальная аппроксимация 2 степени	степенная аппроксимация	экспоненциальная аппроксимация
9	1	1996	149	=4,4182*A9+145,67	=19,959*LN(A9)+140,42	=-0,5571*A9^2+11,103*A9+131,19	=141,81*A9^0,1181	=146,44*EXP(0,0259*A9)
10	2	1997	145					
11	3	1998	168					
12	4	1999	146					
13	5	2000	177					
14	6	2001	176					
15	7	2002	190					
16	8	2003	186					
17	9	2004	176					
18	10	2005	211					
19	11	2006	170					
20	*Контрольные суммы*							
21	ПРОГНОЗ на 2007 год							
22	12	2007						
23	12	2007	ПРЕДСКАЗ					
24	12	2007	РОСТ					
25	12	2007	ТЕНДЕНЦИЯ					
26								
27								
28								
29								
30								

Прогнозирование | Линейная | Логарифмическая | Полиномиальная | Степенная

9. Введите формулы для вычисления значений аппроксимирующих функций в соответствующие ячейки D9, E9, F9, G9, H9 (рис.10.).

Скопируйте формулы вниз по столбцам.

10. Произведите подсчет контрольных сумм в ячейках C20:H20



	A	B	C	D	E	F	G	H
1				Прогнозирование объема продаж предприятия				
2								
3				Метки трендов на диаграммах				
4				y = 4,4182x + 145,67	y = 19,959Ln(x) + 140,42	y = -0,5571x^2 + 11,103x + 131,19	y = 141,81x^{0,1181}	y = 146,44e^{0,0259x}
5				R^2 = 0,53	R^2 = 0,5459	R^2 = 0,5957	R^2 = 0,5661	R^2 = 0,5419
6				Объем продаж				
7			Статистические данные	Теоретические данные				
8		Год	Объем продаж	линейная аппроксимация	логарифмическая аппроксимация	полиномиальная аппроксимация 2 степени	степенная аппроксимация	экспоненциальная аппроксимация
9	1	1996	149	150,0882	140,4200	141,7359	141,8100	150,2823
10	2	1997	145	154,5064	154,2545	151,1676	153,9070	154,2255
11	3	1998	168	158,9246	162,3472	159,4851	161,4563	158,2721
12	4	1999	146	163,3428	168,0890	166,6884	167,0360	162,4249
13	5	2000	177	167,7610	172,5428	172,7775	171,4965	166,6867
14	6	2001	176	172,1792	176,1817	177,7524	175,2292	171,0602
15	7	2002	190	176,5974	179,2584	181,6131	178,4485	175,5486
16	8	2003	186	181,0156	181,9236	184,3596	181,2850	180,1547
17	9	2004	176	185,4338	184,2744	185,9919	183,8243	184,8816
18	10	2005	211	189,8520	186,3773	186,5100	186,1259	189,7326
19	11	2006	170	194,2702	188,2796	185,9139	188,2328	194,7109
20	*Контрольные суммы*		*1894,0000*	1893,9712	1893,9486	1893,9954	1888,8515	1887,9801
21	ПРОГНОЗ на 2007 год							
22	12	2007	На основе аппроксимирующей кривой	198,6884	190,0163	184,2036	190,1771	199,8198
23	12	2007	ПРЕДСКАЗ	198,6909				
24	12	2007	РОСТ					199,9452
25	12	2007	ТЕНДЕНЦИЯ	198,6909				
26								

Прогнозирование | Линейная | Логарифмическая | Полиномиальная | Степенная

В результате получили множество числовых рядов исходных данных, сглаженных по исследуемым трендам (D9:D19; E9:E19; F9:F19; G9:G19; H9:H19), множество вспомогательных контрольных сумм (D20:H20) для выявления наилучшего тренда путем сверки их с главной контрольной суммой (C20).

11. Поместите выводы из анализа полученных результатов исследования динамики продаж с помощью аппроксимации в этом же листе (рис.12.). Проанализировать построенные графики можно, например следующим образом:

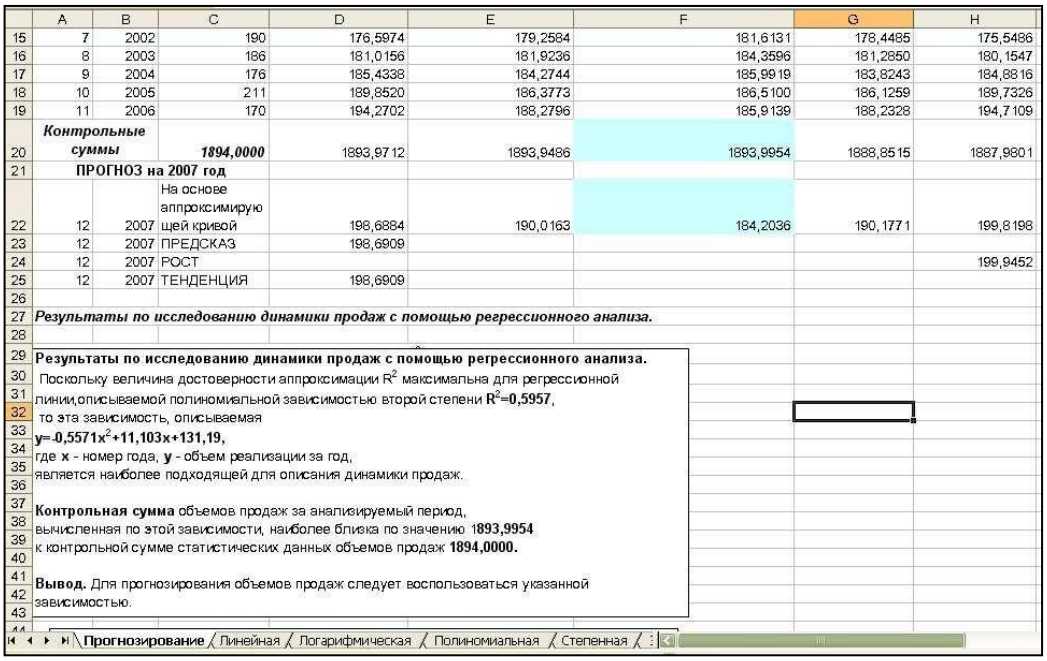
Результаты по исследованию динамики продаж с помощью регрессионного анализа.

Поскольку величина достоверности аппроксимации R2 максимальна для регрессионной линии, описываемой полиномиальной зависимостью второй степени R2=0,5957, то эта

зависимость, описываемая уравнением y=-0,5571x2+11,103x+131,19, где x - номер года, y - объем реализации за год, является наиболее подходящей для описания динамики продаж.

Контрольная сумма объемов продаж за анализируемый период, вычисленная по этой зависимости, наиболее близка по значению 1893,9954 к контрольной сумме статистических данных объемов продаж 1894,0000.

Вывод. Для прогнозирования объемов продаж следует воспользоваться полиномиальной зависимостью.



	A	B	C	D	E	F	G	H
15	7	2002	190	176,5974	179,2584	181,6131	178,4485	175,5486
16	8	2003	186	181,0156	181,9236	184,3596	181,2850	180,1547
17	9	2004	176	185,4338	184,2744	185,9919	183,8243	184,8816
18	10	2005	211	189,8520	186,3773	186,5100	186,1259	189,7326
19	11	2006	170	194,2702	188,2796	185,9139	188,2328	194,7109
20	***Контрольные суммы***		***1894,0000***	1893,9712	1893,9486	1893,9954	1888,8515	1887,9801
21	**ПРОГНОЗ на 2007 год**							
22	12	2007	На основе аппроксимирующей кривой	198,6884	190,0163	184,2036	190,1771	199,8198
23	12	2007	ПРЕДСКАЗ	198,6909				
24	12	2007	РОСТ					199,9452
25	12	2007	ТЕНДЕНЦИЯ	198,6909				
26								
27	***Результаты по исследованию динамики продаж с помощью регрессионного анализа.***							
28								

Результаты по исследованию динамики продаж с помощью регрессионного анализа.
Поскольку величина достоверности аппроксимации R^2 максимальна для регрессионной линии,описываемой полиномиальной зависимостью второй степени **R^2=0,5957**, то эта зависимость, описываемая
$y=-0,5571x^2+11,103x+131,19$,
где **x** - номер года, **y** - объем реализации за год,
является наиболее подходящей для описания динамики продаж.

Контрольная сумма объемов продаж за анализируемый период, вычисленная по этой зависимости, наиболее близка по значению **1893,9954** к контрольной сумме статистических данных объемов продаж **1894,0000.**

Вывод. Для прогнозирования объемов продаж следует воспользоваться указанной зависимостью.

Прогнозирование | Линейная | Логарифмическая | Полиномиальная | Степенная

12. Рассчитайте прогноз объема продаж на основе функций прогнозирования ПРЕДСКАЗ, РОСТ, ТЕНДЕНЦИЯ и расположите результаты вычислений прогноза с помощью функций в соответствующих столбцах. При этом следует учитывать следующее.

Функция ТЕНДЕНЦИЯ возвращает значения в соответствии с линейным трендом. Аппроксимирует прямой линией (по методу наименьших квадратов) массивы **известные_значения_y** и **известные_значения_x**. Возвращает значения **y**, в соответствии с этой прямой для заданного массива **новые_значения_x**. Синтаксис:

ТЕНДЕНЦИЯ(известные_значения_y;известные_значения_x;новые_значения_x;конст)
Известные_значения_y — множество значений y, которые уже известны для соотношения y = mx + b.

Функция РОСТ возвращает значения в соответствии с экспоненциальным трендом.

Рассчитывает прогнозируемый экспоненциальный рост на основании имеющихся данных. Функция РОСТ возвращает значения *y* для последовательности новых значений *x*, задаваемых с помощью существующих *x*- и *y*-значений. Функция рабочего листа РОСТ может применяться также для для аппроксимации существующих x- и y-значений экспоненциальной кривой. Синтаксис:
РОСТ(известные_значения_y;известные_значения_x;новые_значения_x;конст)

Известные_значения_y — это множество значений y, которые уже известны в соотношении y = b*m^x.

Функция ПРЕДСКАЗ возвращает значение линейного тренда. Вычисляет или предсказывает будущее значение по существующим значениям. Предсказываемое значение — это y- значение, соответствующее заданному x-значению. Известные значения — это x- и y-значения, а новое значение предсказывается с использованием линейной регрессии. Эту функцию можно использовать для предсказания будущих продаж, потребностей в оборудовании или тенденций потребления. Синтаксис:

− ПРЕДСКАЗ(x;известные_значения_y;известные_значения_x)

− x — это точка данных, для которой предсказывается значение.
− Известные_значения_y — это зависимый массив или интервал данных.
− Известные_значения_x — это независимый массив или интервал данных.
13. Сделайте сравнительный анализ используемых методов прогнозирования.
14. Сохраните результаты работы в файле.

Индивидуальное задание

Создайте новую рабочую книгу.

1. Выберите таблицу с данными согласно своему индивидуальному варианту.
2. Сохраните результат работы в файл.
3. В ячейку A1 введите – описание переменной x, в ячейку B1 – описание переменной y.
4. Осуществите ввод исследуемых данных в столбцы A и B ниже описанных переменных. 5. Оформите созданную расчетную таблицу
6. Сохраните результат работы в файл.
7. Установить курсор в ячейку C1 и постройте диаграмму ―Объем реализации продукции за наделю‖ по диапазону значений столбца B. 8. Произведите оформление построенной диаграммы
9. Сохраните результат работы в файл.
10. Выберите Зависимость 1 согласно индивидуальному варианту тип для первой линии тренда.
11. Постройте первый тренд для диаграммы.
12. Произведите настройку оформления вида полученного тренда
13. Выберите Зависимость 2 согласно индивидуальному варианту тип для второй линии тренда.
14. Постройте второй тренд для диаграммы.
15. Произведите настройку оформления вида построенных трендов
16. Произведите анализ полученных результатов.
17. Сохраните результат работы в файл.
18. Предъявите работу преподавателю. Заключительные действия
19. Закройте все открытые файлы электронной таблицы.
20. Закончите работу с MS Excel.

Вариант 1

День	1	2	3	4	5	6	7	8
Количество проданных ящиков деталей	13	19	29	30	37	44	49	55

Исследуемые зависимости: линейная, степенная.

Вариант 2

Неделя	1	2	3	4	5	6	7	8	9	10
Количество поступивших упаковок продукции	9	16	20	27	34	39	44	52	58	64

Исследуемые зависимости: экспоненциальная, логарифмическая.

Вариант 3

День	1	2	3	4	5	6	7	8	9
Количество отпущенных флаконов пеногерметика	7	17	19	28	35	42	41	52	57

Исследуемые зависимости: полиномиальная, экспоненциальная.

Вариант 4

День	1	2	3	4	5	6	7	8	9
Количество заказанных пачек медикамента С	12	21	30	36	44	54	61	70	78

Исследуемые зависимости: логарифмическая, линейная

Вариант 5

Месяц	1	2	3	4	5	6	7	8	9	10	11
Количество заказов на переплетные работы	12	17	23	32	35	40	48	54	59	65	72

Исследуемые зависимости: степенная, полиномиальная.

Вариант 6

Час	1	2	3	4	5	6	7	8	9
Количество проданных бутылок напитка К	10	18	22	28	34	39	46	51	54

Исследуемые зависимости: линейная, экспоненциальная.

Вариант 7

Неделя	1	2	3	4	5	6	7	8
Количество проданных подержанных машин	12	18	25	32	40	46	53	60

Исследуемые зависимости: экспоненциальная, линейная.

Вариант 8

День	1	2	3	4	5	6	7	8	9
Количество заказов на хлебобулочное изделие N	14	23	30	39	45	54	63	70	78

Исследуемые зависимости: полиномиальная, линейная.

Вариант 9

Месяц	1	2	3	4	5	6	7	8	9	10	11
Количество проданных сувениров А	15	22	26	33	40	45	52	58	63	69	78

Исследуемые зависимости: логарифмическая, экспоненциальная.

Вариант 10

Неделя	1	2	3	4	5	6	7	8
Количество заказов на установку	9	15	24	29	38	46	52	58

машинной сигнализации								

Исследуемые зависимости: степенная, логарифмическая.

Вариант 11

Неделя	1	2	3	4	5	6	7	8	9	10	11
Количество	9	12	17	23	30	36	40	48	54	65	76

заказов на ремонт стиральных машин											

Исследуемые зависимости: линейная, полиномиальная.

Вариант 12

День	1	2	3	4	5	6	7	8
Количество абитуриентов интересующихся специальностью Z	13	19	26	30	37	44	49	55

Исследуемые зависимости: экспоненциальная, линейная.

Вариант 13

Месяц	1	2	3	4	5	6	7	8
Количество заказов на литературу типа X	12	18	25	32	40	46	53	60

Исследуемые зависимости: полиномиальная, экспоненциальная.

Вариант 14

День	1	2	3	4	5	6	7	8	9
Количество проданных флаконов шампуня B	7	17	19	28	35	42	41	52	57

Исследуемые зависимости: логарифмическая, линейная.

Вариант 15

Неделя	1	2	3	4	5	6	7	8
Количество проданных ящиков кондитерской продукции типа Ш	9	15	24	29	38	46	52	58

Исследуемые зависимости: степенная, полиномиальная.

Контрольные вопросы

1. Что отражает величина достоверности аппроксимации?
2. Дайте определение тренда.
3. В каких случаях необходимо использовать построение трендов?

4. На основе каких данных выбирается наилучшая регрессионная линия?
5. Как изменить формат представления регрессионной линии?
6. Какие типы регрессионных зависимостей Вам известны?
7. Опишите действия необходимые для построения линии тренда по построенной диаграмме.
8. Возможен ли ретроспективный анализ данных с использованием линий тренда?
9. Возможно ли использование регрессионных зависимостей при решении задач по оптимизации ресурсов и запасов?
10. Опишите ситуации, в которых правомочно представление нескольких графиков в одной системе координат.

Задание 2.3.2.5. Лабораторная работа «Выбор оптимального решения с помощью дерева решений»

Проверяемые результаты обучения*: ОК1, ОК2, ОК03, ПК2.1, ПК2.4., ПК2.5.*

Цель: освоение основных методов и способов построения деревьев решений, приобретение практических навыков по использованию инструментария Deductor 4. Краткая теория

Деревья решений (decision trees) являются одним из самых мощных средств решения задачи отнесения какого-либо объекта (строчки набора данных) к одному из заранее известных классов. Дерево решений – это классификатор, полученный из обучающего множества, содержащего объекты и их характеристики, на основе обучения. Дерево состоит из узлов и листьев, указывающих на класс.

Результатом работы алгоритма является список иерархических правил образующих дерево. Каждое правило – это интуитивно-понятная конструкция вида «Если…то…» (if - then). Дерево может использоваться для классификации объектов, не вошедших в обучающее множество. Чтобы принять решение, к какому классу следует отнести некоторый объект или ситуацию, требуется ответить на вопросы, стоящие в узлах этого дерева, начиная с его корня. Вопросы имеют вид «значение параметра А больше В?». Если ответ положительный, осуществляется переход к правому узлу следующего уровня; затем снова следует вопрос, связанный с соответствующим узлом и т. д.

Настройка назначения полей

Необходимо определить, как будут использоваться поля исходного набора данных при обучении дерева и дальнейшей практической работе с ним.

В левой части окна представлен список всех полей исходного набора данных. Для настройки поля следует выделить его в списке, при этом в правой части окна будут отображены текущие параметры поля:

1. Имя поля – идентификатор поля, определенный для него в источнике данных. Изменить его здесь нельзя.
2. Тип данных – тип данных, содержащихся в поле (вещественный, строковый, дата). Он также задается в источнике данных и здесь изменен быть не может.
3. Назначение – здесь необходимо выбрать порядок использования данного поля при обучении и работе дерева решений. Выбор производится с помощью списка, открываемого кнопкой и содержащего следующие варианты:

− Входное – значения поля будут являться исходными данными для построения и дальнейшей практической работы дерева решений, на их основе будет производиться классификация.

− Выходное – будет содержать результаты классификации. Выходное поле может быть только одно и оно должно быть дискретным.

− Информационное – поле не будет использоваться при обучении дерева, но будет помещено в результирующий набор в исходном состоянии.

− Неиспользуемое – поле не будет использоваться при построении и работе дерева решений и будет исключено из результирующей выборки. В отличие от непригодного, такое поле может быть использовано, если в этом возникнет необходимость.

− Непригодное – поле не может быть использовано при построении и работе алгоритма, но будет помещено в результирующий набор в исходном состоянии.

4. Вид данных – указывает на характер данных, содержащихся в поле (непрерывный или дискретный). Изменить это свойство здесь нельзя.

Статус непригодного поля устанавливается только автоматически и в дальнейшем может быть изменен только на неиспользуемое или информационное. Поле будет запрещено к использованию если:

− поле является дискретным и содержит всего одно уникальное значение; − непрерывное поле с нулевой дисперсией; − поле содержит пропущенные значения.

В случае если текущее поле содержит непрерывные (числовые) данные, отображается секция «Статистика», где показываются максимальное и минимальное значения поля, его среднее значение и стандартное отклонение. Если выделенное поле содержит дискретные (строковые) данные, то для него открывается секция «Уникальные значения», в которой отображается общее число уникальных значений поля, а также список самих уникальных значений.

Нормализация полей

Целью нормализации значений полей является преобразование данных к виду, наиболее подходящему для обработки средствами пакета Deductor. Для дерева решений данные, поступающие на вход, должны иметь числовой тип. В этом случае нормализатор может преобразовать дискретные данные к набору уникальных индексов.

Окно настройки нормализации полей вызывается с помощью кнопки «Настройка нормализации». В окне слева приведен полный список входных и выходных полей. При этом каждое поле помечено значком, обозначающим вид нормализации поля:

− линейная – линейная нормализация исходных значений;

− уникальные значения – преобразование уникальных значений в их индексы.

Для числовых (непрерывных) полей с линейной нормализацией дополнительные параметры недоступны. В полях «Минимум» и «Максимум» секции «Диапазон значений» можно посмотреть минимальное и максимальное значения этого поля.

Для дискретных полей справа находится список уникальных значений поля, где для каждого значения указывается количество его повторений. Поле «Количество значений» показывает общее число уникальных значений, принимаемых полем.

Настройка обучающей выборки

Обучающая выборка может быть разбита на три множества – обучающее, тестовое и валидационное.

1. Обучающее множество – включает записи (примеры), которые будут использоваться в качестве входных данных, а также соответствующие желаемые выходные значения.

2. Тестовое множество – также включает записи, содержащие входные и желаемые выходные значения, но используемое не для обучения модели, а для проверки его результатов.

3. Валидационное множество – множество примеров, используемое как для оценки результатов обучения модели, так и определения ее параметров.

Для разбиения исходного множества на обучающее, тестовое и валидационное необходимо настроить несколько параметров:

1. Из списка «Способ разделения исходного множества» выбирается порядок отбора записей во все три множества. Если выбран вариант «по порядку», то порядок следования записей при их разделении не меняется. Множества последовательно формируются в соответствии с определенным для них числом записей. Если выбран вариант «случайно», то отбор записей происходит случайным образом.

2. Затем необходимо указать, какие множества будут использоваться. Для того чтобы множество было сформировано, нужно установить флажок слева от его названия. Если

флажок сброшен, то множество использовано не будет. Обучающее множество используется всегда, поэтому сбросить флажок для него нельзя.

3. Для каждого из используемых множеств необходимо задать его размер. Размер может быть задан непосредственно количеством записей или в процентах от объема исходной выборки. Для этого достаточно дважды щелкнуть мышью в соответствующей клетке и ввести нужное значение с клавиатуры. При этом размер, введенный в процентах, автоматически пересчитывается в количество строк и наоборот. В поле «Количество строк (всего)» отображается общее количество записей в исходной выборке данных, которое может быть задействовано для формирования множеств. Если суммарное число строк для всех используемых множеств меньше полного числа строк исходной выборки, то размеры множеств можно задавать произвольно. Можно, например, использовать не все записи, а только часть из них. Если же суммарное указанное число строк превышает максимальное для данной исходной выборки, то автоматически включается баланс множеств, т.е. при указании для одного из множеств размера, в результате которого будет превышено максимальное число записей в исходной выборке, размер остальных множеств будет автоматически уменьшен таким образом, чтобы суммарный размер множеств не превышал доступного числа записей. В строке

«Итого» указывается количество записей, задействованных во всех используемых множествах, а также процент от полного числа записей исходной выборки, который они составляют.

4. В столбце «Порядок сортировки» можно определить порядок следования записей внутри каждого множества. Для этого необходимо дважды щелкнуть мышью в столбце

«Порядок сортировки» для соответствующего множества и с помощью появившейся кнопки выбора открыть список, в котором выбрать один из возможных вариантов:

− по возрастанию – записи в данном множестве будут следовать в порядке возрастания; − по убыванию – записи в данном множестве будут следовать в порядке убывания; − случайно – записи в данном множестве будут следовать в случайном порядке.

Для того чтобы обучающее множество было репрезентативным необходимо, чтобы все уникальные значения всех дискретных столбцов содержались в данном наборе данных.

Настройка параметров обучения

Необходимо установить параметры, в соответствии с которыми будет проводиться обучение дерева:

1. «Минимальное количество примеров, при котором будет создан новый узел» – задается минимальное количество примеров, которое возможно в узле. Если примеров, которые попадают в данный узел, будет меньше заданного – узел считается листом (т.е. дальнейшее ветвление прекращается).

2. «Строить дерево с более достоверными правилами в ущерб сложности» – установка данного флажка включает специальный алгоритм, который, усложняя структуру дерева, увеличивает достоверность результатов классификации. Сброс данного флажка хотя и приводит к упрощению дерева, снижает достоверность результатов классификации.

3. «Уровень доверия, используемый при отсечении узлов дерева». Значение этого параметра задается в процентах и должно лежать в пределах от 0 до 100. Эти значения выбираются из списка. Чем больше уровень доверия, тем более ветвистым получается дерево, и, соответственно, чем меньше уровень доверия, тем больше узлов будет отсечено при его построении.

Задание

1. Разработать сценарии построения дерева решений и проведения анализа «что - если».

2. По таблице (например, продаж) создать таблицу транзакций с полями (например, Менеджер, Организация, Вид товара). Таблицу получить путем слияния соответствующих полей из разных таблиц и последующей группировки.

3. Разработать сценарии построения дерева решений с представлением правил, наиболее популярных наборов и анализа «что - если» с входными полями (например, Менеджер и Организация) и выходным полем (например, Вид товара).

4. Создать отчеты по всем разработанным сценариям.

5. Продемонстрировать проект преподавателю с использованием тестовых наборов данных и защитить работу.

Контрольные вопросы

1. С какой целью проводится нормализация значений полей?
2. Для чего используется обучающая выборка? Из каких множеств она состоит?
3. Какие критерии используются для выбора параметров обучения?
4. Какие требования предъявляются к исходным данным при построении дерева решений?
5. Поясните смысл расчетных полей при анализе «что - если».

3. Тестовые задания для текущего контроля

Проверяемые результаты обучения*:* ОК1, ОК2, ОК03, ПК2.1, ПК2.4., ПК2.5.

1. Прочитайте текст, выберите все правильные ответы и запишите аргументы, обосновывающие выбор ответа

Задание №1. Какова цель метода наименьших квадратов в контексте математического моделирования?

А) максимизировать точность численных методов;

Б) минимизировать количество уравнений в системе;

В) минимизировать сумму квадратов разностей между предсказанными и наблюдаемыми значениями;

Г) максимизировать количество параметров в модели.

Ответ:

2. Прочитайте текст, выберите все правильные ответы и запишите аргументы, обосновывающие выбор ответа

Задание №2. Какова основная задача калибровки в математическом моделировании?

А) улучшение визуальной привлекательности графиков модели;

Б) сокращение времени выполнения программы;

В) увеличение степени сложности математических выражений в модели;

Г) настройка параметров модели так, чтобы она точно отражала реальные данные.

Ответ:

3. Определите оптимальную точку для решения задачи целочисленного программирования, представленной в виде следующей математической модели.

4. Какой вид имеет целевая функция данной задачи: «Фабрика производит изделия двух видов. На изготовление одного изделия вида А необходимо израсходовать три кг сырья, на изготовление одного изделия вида В – пять кг. Всего имеется 93 кг сырья. Необходимо составить такой план производства, чтобы получить наибольшую выручку, если стоимость одного изделия вида А 6 у.е., вида В – 7 у.е., причем требуется изготовить изделий вида А не более 35, а вида В – не более 45».

5. Вставьте пропущенное слово в предложение: «… математические модели реализуются с помощью систем программирования, электронных таблиц, специализированных математических пакетов и программных средств».

6. Вставьте пропущенное слово в предложение: «После математической постановки задачи отвлекаются от её предметной сущности и оперируют с абстрактными математическими

понятиями, величинами, формулами для выбора … решения задачи.».

7. В какой форме задана задача линейного программирования, в которой требуется найти экстремум функции

n

Z(X) □□cjxj □ max, удовлетворяющая ограничениям:

j□1

n

□aij x j □ bi , i □1,m, xj □ 0, j □1,n ?

8. Прочитайте текст, выберите все правильные ответы и запишите аргументы, обосновывающие выбор ответа

Как называются модели, реализованные с помощью систем программирования, электронных таблиц, специализированных математических пакетов и программных средств для моделирования?

А) математические;

Б) компьютерные;

В) имитационные;

Г) программные.

Ответ:

9. Прочитайте текст, выберите все правильные ответы и запишите аргументы, обосновывающие выбор ответа

Задание №2. Какая модель является предметом формализации?

А) структурно-функциональная;

Б) физическая;

В) математическая;

Г) имитационная.

Ответ:

10. Прочитайте текст, выберите все правильные ответы и запишите аргументы, обосновывающие выбор ответа

Задание №3. Определите, что такое математическая модель объекта?

А) совокупность записанных на языке математики формул, отражающих те или иные свойства объекта-оригинала или его поведение;

Б) совокупность данных, содержащих информацию о количественных характеристиках объекта и его поведении в виде таблицы;

В) созданная из какого-либо материала модель, точно отражающая внешние признаки объекта-оригинала.

Ответ:

Ключ к тестовым заданиям

№ задания	Верный ответ	Критерии
1	В	1б – полное правильное соответствие 0 б – остальные случаи
2	Г	1б – полное правильное соответствие 0 б – остальные случаи
3	канонической	1б – полное правильное соответствие 0 б – остальные случаи
4	(2,0)	1б – полное правильное соответствие

		0 б – остальные случаи
5	F □x1, x2 □ □ 6x1 □ 7x2 max;	1б – полное правильное соответствие 0 б – остальные случаи
6	Компьютерные	1б – полное правильное соответствие 0 б – остальные случаи
7	метода	1б – полное правильное соответствие 0 б – остальные случаи
8	Б	1б – полное правильное соответствие 0 б – остальные случаи
9	В	1б – полное правильное соответствие 0 б – остальные случаи
10	А	1б – полное правильное соответствие 0 б – остальные случаи

Критерии оценки:

соответствие ответов обучающихся ключу теста

Оценка «**отлично**» - если обучающийся правильно выполнил от 80% до 100% тестовых заданий в отведенное время

Оценка «**хорошо**» - если обучающийся правильно выполнил от 60% до 80% тестовых заданий в отведенное время

Оценка «**удовлетворительно**» - если обучающийся правильно выполнил от 40% до 60% тестовых заданий в отведенное время

Оценка «**неудовлетворительно**» ставится в случае выполнения менее 40% тестовых заданий

Время выполнения: 35-40 минут

Теоретические вопросы к дифференцированному зачету
МДК 02.01 «Технология разработки программного обеспечения
Проверяемые результаты освоения формируемых компетенций: ОК 01, ОК 02, ОК 03, ОК 04, ОК 05, ОК 09, ПК 2.1, ПК 2.4, ПК 2.5.

1. Общие характеристики качества программного средства.
2. Критерии качества программ.
3. Метрические показатели программ.
4. Виды метрик программ.
5. Классификация видов сложности программных продуктов.
6. Измеримые свойства алгоритмов.
7. Длина программы.
8. Объём программы.
9. Уровень программы.
10. Определение интеллектуального содержания программ.
11. Уровни языков программирования.
12. Метрика числа ошибок в программе.
13. Расчет времени, необходимого для программирования.
14. Метрики структурной сложности программ.
15. Современные принципы и методы разработки программных приложений.
16. Основные подходы к интегрированию программных модулей.
17. Стандарты кодирования.
18. Цели и задачи и виды тестирования. Стандарты качества программной документации. Меры и метрики.
19. Тестовое покрытие.
20. Тестовый сценарий, тестовый пакет.
21. Анализ спецификаций. Верификация и аттестация программного обеспечения.
22. Понятие репозитория проекта, структура проекта.
23. Виды, цели и уровни интеграции программных модулей. Автоматизация бизнес-процессов.
24. Отладка программных продуктов. Инструменты отладки. Отладочные классы.
25. Ручное и автоматизированное тестирование. Методы и средства организации тестирования.

Практическое задание

Разработать и презентовать групповой проект информационной системы для конкретной задачи и отрасли (по билету):

1. Сформировать команду (от 3 до 5 человек). Разделить обязанности в группе (менеджер проекта, дизайнер интерфейса, проектировщик системы, оформитель материалов и пр.).
2. Выбрать предметную область и существующий в ней (реальный или воображаемый) бизнес-субъект (производственная компания, научно-исследовательское предприятие, муниципальное учреждение и т.д.). Кратко описать свою компанию.
3. Для выбранной организации обозначить проблему, которая может быть решена с помощью ИТ. На диаграмме Исикавы покажите степень влияния фактора ИТ на проблему.
4. Представить описание автоматизируемого процесса (в любой формальной нотации).
5. Составить дерево требований к ИС (включая требования информационной безопасности).
6. Перечислить документы (стандарты и другие регламенты), необходимые для реализации внедрения проектируемой ИС на каждом этапе ее ЖЦ.
7. Разработать общий паспорт-план проекта реализации и внедрения ИС с перечислением его основных параметров (участники, риски, вехи и т.д.).
8. Разработать объектную модель системы (UML-диаграмма классов);
9. Разработать архитектуру системы (UML-диаграмма компонентов и развертывания);
10. Представить примеры пользовательского интерфейса (пункты меню, формы и пр.);

Теоретические вопросы к дифференцированному зачету
МДК 02.02 «Инструментальные средства разработки программного обеспечения»
Проверяемые результаты освоения формируемых компетенций: ОК 01, ОК 02, ОК 03, ОК 04, ОК 05, ОК 09, ПК 2.2, ПК 2.3, ПК 2.5.

1. Понятие репозитория проекта, структура проекта.
2. Виды, цели и уровни интеграции программных модулей. Автоматизация бизнес-процессов.
3. Выбор источников и приемников данных, сопоставление объектов данных.
4. Транспортные протоколы. Стандарты форматирования сообщений.
5. Организация работы команды в системе контроля версий.
6. В том числе практических занятий и лабораторных работ.
7. Разработка структуры проекта.
8. Разработка модульной структуры проекта (диаграммы модулей).
9. Разработка перечня артефактов и протоколов проекта.
10. Настройка работы системы контроля версий (типов импортируемых файлов, путей, фильтров и др. параметров импорта в репозиторий).
11. Разработка и интеграция модулей проекта (командная работа).
12. Отладка отдельных модулей программного проекта.
13. Организация обработки исключений.
14. Отладка программных продуктов. Инструменты отладки. Отладочные классы.
15. Ручное и автоматизированное тестирование. Методы и средства организации тестирования.
16. Инструментарии анализа качества программных продуктов в среде разработке.
17. Обработка исключительных ситуаций. Методы и способы идентификации сбоев и ошибок.
18. Выявление ошибок системных компонентов.
19. Применение отладочных классов в проекте.
20. Отладка проекта.

Практические задания:

1. Провести инспекцию кода модулей предложенного проекта.
2. Протестировать предложенный интерфейс пользователя средствами инструментальной среды разработки.
3. Разработать тестовый модуль предложенного проекта.
4. Выполнить функциональное тестирование предложенного проекта.
5. Выполнить документирование результатов тестирования проекта.

Вопросы к дифференцированному зачету
МДК 02.03 «Математическое моделирование»
Проверяемые результаты освоения формируемых компетенций: ОК 01, ОК 02, ОК 03, ОК 04, ОК 05, ОК 09, ПК 2.1, ПК 2.4, ПК 2.5.

1. Понятие решения. Множество решений, оптимальное решение. Показатель эффективности решения
2. Математические модели, принципы их построения, виды моделей.
3. Задачи: классификация, методы решения, граничные условия.
4. Общий вид и основная задача линейного программирования. Симплекс - метод.
5. Транспортная задача. Методы нахождения начального решения транспортной задачи. Метод потенциалов.
6. Общий вид задач нелинейного программирования. Графический метод решения задач нелинейного программирования. Метод множителей Лагранжа.

7. Основные понятия динамического программирования: шаговое управление, управление операцией в целом, оптимальное управление, выигрыш на данном шаге, выигрыш за всю операцию, аддитивный критерий, мультипликативный критерий.

8. Простейшие задачи, решаемые методом динамического программирования.

9. Методы хранения графов в памяти ЭВМ. Задача о нахождении кратчайших путей в графе и методы ее решения.

10. Задача о максимальном потоке и алгоритм Форда-Фалкерсона.

11. В том числе практических занятий и лабораторных работ

12. Построение простейших математических моделей. Построение простейших статистических моделей.

13. Решение простейших однокритериальных задач.

14. Задача Коши для уравнения теплопроводности.

15. Задача о распределении средств между предприятиями.

16. Задача о замене оборудования.

17. Системы массового обслуживания: понятия, примеры, модели.

18. Основные понятия теории марковских процессов: случайный процесс, марковский процесс, граф состояний, поток событий, вероятность состояния, уравнения Колмогорова, финальные вероятности состояний.

19. Метод имитационного моделирования. Единичный жребий и формы его организации. Примеры задач

20. Понятие прогноза. Количественные методы прогнозирования: скользящие средние, экспоненциальное сглаживание, проектирование тренда. Качественные методы прогноза

21. Предмет и задачи теории игр. Основные понятия теории игр: игра, игроки, партия, выигрыш, проигрыш, ход, личные и случайные ходы, стратегические игры, стратегия, оптимальная стратегия.

22. Антагонистические матричные игры: чистые и смешанные стратегии.

23. Методы решения конечных игр: сведение игры mxn к задаче линейного программирования, численный метод - метод итераций.

24. Область применимости теории принятия решений. Принятие решений в условиях определенности, в условиях риска, в условиях неопределенности.

25. Критерии принятия решений в условиях неопределенности. Дерево решений.

Теоретические вопросы к экзамену по модулю ПМ.02 в 6 семестре
Проверяемые результаты освоения формируемых компетенций: ОК 01, ОК 02, ОК 03, ОК 04, ОК 05, ОК 06, ОК 07, ОК 08, ОК 09, ПК 2.1, ПК 2.2, ПК 2.3, ПК 2.4, ПК 2.5.

1. Современные принципы и методы разработки программных приложений.

2. Методы организации работы в команде разработчиков. Системы контроля версий

3. Основные подходы к интегрированию программных модулей.

4. Стандарты кодирования.

5. Цели и задачи и виды тестирования. Стандарты качества программной документации. Меры и метрики.

6. Тестовое покрытие.

7. Тестовый сценарий, тестовый пакет.

8. Анализ спецификаций. Верификация и аттестация программного обеспечения.

9. Понятие репозитория проекта, структура проекта.

10. Виды, цели и уровни интеграции программных модулей. Автоматизация бизнес-процессов.

11. Выбор источников и приемников данных, сопоставление объектов данных.

12. Транспортные протоколы. Стандарты форматирования сообщений.

13. Организация работы команды в системе контроля версий.

14. Отладка программных продуктов. Инструменты отладки. Отладочные классы.

15. Ручное и автоматизированное тестирование. Методы и средства организации тестирования.

16. Инструментарии анализа качества программных продуктов в среде разработке

17. Обработка исключительных ситуаций. Методы и способы идентификации сбоев и ошибок.

18. Выявление ошибок системных компонентов.

19. Понятие решения. Множество решений, оптимальное решение. Показатель эффективности решения

20. Математические модели, принципы их построения, виды моделей.

21. Задачи: классификация, методы решения, граничные условия.

22. Общий вид и основная задача линейного программирования. Симплекс - метод.

23. Транспортная задача. Методы нахождения начального решения транспортной задачи. Метод потенциалов.

24. Общий вид задач нелинейного программирования. Графический метод решения задач нелинейного программирования. Метод множителей Лагранжа.

25. Основные понятия динамического программирования: шаговое управление, управление операцией в целом, оптимальное управление, выигрыш на данном шаге, выигрыш за всю операцию, аддитивный критерий, мультипликативный критерий.

26. Простейшие задачи, решаемые методом динамического программирования.

27. Методы хранения графов в памяти ЭВМ. Задача о нахождении кратчайших путей в графе и методы ее решения.

28. Задача о максимальном потоке и алгоритм Форда-Фалкерсона.

29. Системы массового обслуживания: понятия, примеры, модели.

30. Основные понятия теории марковских процессов: случайный процесс, марковский процесс, граф состояний, поток событий, вероятность состояния, уравнения Колмогорова, финальные вероятности состояний.

31. Схема гибели и размножения.

32. Метод имитационного моделирования. Единичный жребий и формы его организации. Примеры задач

33. Понятие прогноза. Количественные методы прогнозирования: скользящие средние, экспоненциальное сглаживание, проектирование тренда. Качественные методы прогноза.

34. Предмет и задачи теории игр. Основные понятия теории игр: игра, игроки, партия, выигрыш, проигрыш, ход, личные и случайные ходы, стратегические игры, стратегия, оптимальная стратегия.

35. Антагонистические матричные игры: чистые и смешанные стратегии.

36. Методы решения конечных игр: сведение игры mxn к задаче линейного программирования, численный метод - метод итераций.

37. Область применимости теории принятия решений. Принятие решений в условиях определенности, в условиях риска, в условиях неопределенности.

38. Критерии принятия решений в условиях неопределенности. Дерево решений.

Критерии оценки

Оценка **«отлично»** выставляется обучающемуся, если он глубоко и прочно усвоил программный материал, исчерпывающе, последовательно, четко и логически стройно его излагает, умеет тесно увязывать теорию с практикой, свободно справляется с задачами, вопросами и другими видами применения знаний, причем не затрудняется с ответом при видоизменении заданий, использует в ответе материал литературы, правильно обосновывает принятое решение, владеет разносторонними навыками и приемами выполнения практических задач.

Оценка **«хорошо»** выставляется обучающемуся, если он твердо знает материал, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на

вопрос, правильно применяет теоретические положения при решении практических вопросов и задач, владеет необходимыми навыками и приемами их выполнения.

Оценка **«удовлетворительно»** выставляется обучающемуся, если он имеет знания только основного материала, но не усвоил его деталей, допускает неточности, недостаточно правильные формулировки, нарушения логической последовательности в изложении программного материала, испытывает затруднения при выполнении практических работ.

Оценка **«неудовлетворительно»** выставляется обучающемуся, который не знает значительной части программного материала, допускает существенные ошибки, неуверенно, с большими затруднениями выполняет практические работы. Как правило, оценка «неудовлетворительно» ставится студентам, которые не могут продолжить обучение без дополнительных занятий по соответствующей дисциплине.

6. Оценка учебной и производственной практики (по профилю специальности)

6.1. Общие положения

Целью оценки по учебной и производственной практике является оценка профессиональных и общих компетенций; практического опыта и умений.

Оценка по учебной практике выставляется на основании данных аттестационного листа с указанием видов работ, выполненных обучающимся во время практики, их объема, качества выполнения в соответствии с технологией и требованиями организаций, в которой проходила практика.

Оценка по производственной практике выставляется на основании аттестационного листа, производственной характеристики, дневника, отчета по практике.

6.2. Виды работ практики и проверки результатов обучения профессионального модуля

6.2.1. Учебная практика

Код	**Код наименование результата обучения**
ОК 01 ***ОК 02*** ***ОК 03*** ***ОК 04*** ***ОК 05*** ***ОК 06*** ***ОК 07*** ***ОК 08*** ***ОК 09*** ***ПК 2.1*** ***ПК 2.2.*** ***ПК 3.3.*** ***ПК 2.4.*** ***ПК2.5.***	1. Анализ предметной области - разработка и оформление требований к программным модулям по предложенной документации; - анализировать проектную и техническую документацию; 2. Определение требований проекта - разработка и оформление требований к программным модулям по предложенной документации; - анализировать проектную и техническую документацию; 3. Разработка и оформление документа «Техническое задание» - разработка и оформление требований к программным модулям по предложенной документации; - анализировать проектную и техническую документацию; 4. Разработка структуры проекта - использовать специализированные графические средства построения и анализа архитектуры программных продуктов; - анализировать проектную и техническую документацию; 5. Работы в системе контроля версий - использовать выбранную систему контроля версий; 6. Внешнее проектирование (разработка внешней спецификации) - анализировать проектную и техническую документацию; - использовать специализированные графические средства построения и анализа архитектуры программных продуктов; 7. Внутреннее проектирование (разработка схем и диаграмм проекта) - использовать специализированные графические средства построения и анализа архитектуры программных продуктов; - определять источники и приемники данных; - проводить сравнительный анализ; 8. Разработка модулей проекта и их элементов - определять источники и приемники данных; - проводить сравнительный анализ; - использовать методы для получения кода с заданной функциональностью и степенью качества; -разрабатывать элементы программного модуля в соответствии с требованиями; - использовать выбранную систему контроля версий;

	9. Интеграция модулей в программное обеспечение - интеграции модулей в программное обеспечение; - использовать выбранную систему контроля версий; - использовать методы для получения кода с заданной функциональностью и степенью качества; - организовывать заданную интеграцию модулей в программные средства на базе имеющейся архитектуры и автоматизации бизнес -процессов; 10. Модификация модулей проекта – инспектировании разработанных программных модулей на предмет соответствия стандартам кодирования; – модификация программных модулей; использовать выбранную систему контроля версий; проводить сравнительный анализ; 11. Отладка модулей – выполнять отладку, используя методы и программного проекта. Организация обработки исключений инструменты условной компиляции (классы debug и trace); – выявлять ошибки в системных компонентах на основе спецификаций; – инспектирование разработанных программных модулей на предмет соответствия стандартам кодирования; 12. Отладка проекта. Инспекция кода модулей проекта – инспектирование разработанных программных модулей на предмет соответствия стандартам кодирования; 13. Тестирование интерфейса пользователя средствами инструментальной среды разработки, выполнение функционального тестирования – оценивать размер минимального наборатестов; – разработка тестовых сценариев программного средства; – разработка тестовых наборов (пакеты) для программного модуля; – разработке тестовых сценариев программного средства.

4.2.2. Производственная практика

Код	**Код наименование результата обучения**
ОК 01 ***ОК 02*** ***ОК 03*** ***ОК 04*** ***ОК 05*** ***ОК 06*** ***ОК 07*** ***ОК 08*** ***ОК 09*** ***ПК 2.1*** ***ПК 2.2.*** ***ПК 3.3.*** ***ПК 2.4.*** ***ПК2.5.***	Знакомство с местом практики. Изучение инструкций и правил. Анализ предметной области. Определение требований проекта Ознакомление с ГОСТ по разработке технического задания. Разработка структуры проекта Внешнее проектирование (разработка внешней спецификации) Внутреннее проектирование (разработка схем и диаграмм проекта) Разработка модулей проекта и их элементов. Работа в системе контроля версий. Интеграция модулей в программное обеспечение Модификация модулей проекта Разработка тестов для контроля правильности работы. Проведение тестирования и отладки разрабатываемых приложений. Оформление отчета по результатам тестов Проведение оценки качества программных продуктов.

Форма аттестационного листа

Приложение 1

АТТЕСТАЦИОННЫЙ ЛИСТ

по учебной практике

Ф.И.О. обучающегося _____

№ группы: _____

Специальность 09.02.07 Информационные системы и программирование

Профессиональный модуль_____

Место прохождения практики

_____

Сроки прохождения практики с _____**г.** по _____ **г.**

Виды и качество выполнения работ

Код, ПК	**Виды работ, обеспечивающих формирование**	**Качество выполнения работ в соответствии с технологией и (или) требованиями организации (оценка)**

Руководитель практики от образовательной организации:

М.П. _____ /_____

Подпись ФИО

Руководитель практики от (предприятия, организации, учреждения):

_____ /_____

М.П. Подпись ФИО

печать организации

Приложение 2

АТТЕСТАЦИОННЫЙ ЛИСТ
по производственной практике

Ф.И.О. обучающегося _____
№ группы: _____
Специальность 09.02.07 Информационные системы и программирование
Профессиональный модуль_____
Место прохождения практики
_____
Сроки прохождения практики с _____**г.** по _____ **г.**

Виды и качество выполнения работ

Код, ПК	**Виды работ, обеспечивающих формирование**	**Качество выполнения работ в соответствии с технологией и (или) требованиями организации (оценка)**

Руководитель практики от образовательной организации:

М.П. _____ /_____
Подпись ФИО

Руководитель практики от (предприятия, организации, учреждения):

_____ /_____
М.П. Подпись ФИО
печать организации

Критерии оценки:

«Отлично» – посещаемость при прохождении практики 100%, аттестационный лист имеет только отличную оценку качества выполненных видов работ, положительную характеристику от организации, дневник по прохождению практики полностью заполнен и сдан в установленный срок, отчет по практике представлен в полном объеме, все задания выполнены без замечаний и ошибок, студент рассказал о предприятии и объяснил алгоритм выполнения заданий в соответствии с требованиями.

«Хорошо» – посещаемость при прохождении практики 100%, аттестационный лист имеет только положительную оценку качества выполненных видов работ, положительную характеристику от организации, дневник по прохождению практики полностью заполнен и сдан в установленный срок, отчет по практике представлен в полном объеме, задания выполнены с незначительными замечаниями и ошибок, студент рассказал о предприятии, неуверенно объяснил алгоритм выполнения заданий в соответствии с требованиями, не смог ответить на некоторые вопросы при сдаче отчета по учебной практике.

«Удовлетворительно» – посещаемость при прохождении практики 75%, аттестационный лист имеет удовлетворительную оценку качества выполненных видов работ, удовлетворительную характеристику от организации, дневник по прохождению практики заполнен и сдан в установленный срок, отчет по практике представлен с замечаниями , задания выполнены с незначительными ошибками, которые пояснить не смог, студент рассказал о предприятии, неуверенно объяснил алгоритм выполнения заданий в соответствии с требованиями, не смог ответить на некоторые вопросы при сдаче отчета по практике.

«Неудовлетворительно» - посещаемость при прохождении практики 50%, аттестационный лист имеет неудовлетворительную оценку качества выполненных видов работ, характеристика от организации отсутствует, дневник по прохождению практики не заполнен и не сдан в установленный срок, отчет по практике представлен со значительными ошибками и замечаниями, студент не рассказал о предприятии, не объяснил алгоритм выполнения заданий в соответствии с требованиями, не ответил на вопросы при сдаче отчета по практике.

5. ОЦЕНОЧНЫЕ СРЕДСТВА ДЛЯ ПРОВЕДЕНИЯ ЭКЗАМЕНА ПО ПРОФЕССИОНАЛЬНОМУ МОДУЛЮ ОК 01, ОК 02, ОК 03, ОК 04, ОК 05, ОК 06, ОК 07, ОК 08, ОК 09, ПК 2.1, ПК 2.2, ПК 2.3, ПК 2.4, ПК 2.5.

Формой аттестации по профессиональному модулю является экзамен.

Форма проведения экзамена – выполнение заданий.

Итогом экзамена является однозначное решение: «вид профессиональной деятельности освоен / не освоен».

При принятии решения об итоговой оценке по профессиональному модулю учитывается оценка показателей для выполнения вида профессиональной деятельности, освоение которого проверяется. При отрицательном заключении хотя бы по одному показателю оценки результата освоения профессиональных компетенций принимается решение «вид профессиональной деятельности не освоен». При наличии противоречивых оценок по одному тому же показателю при выполнении разных видов работ, решение принимается в пользу студента.

Инструкция

Внимательно прочитайте задание.

Вы можете воспользоваться учебно-методической и справочной литературой, имеющейся на специальном столе

Время выполнения заданий – 150 минут.

Задание 1. Выполнить интеграцию модуля в программное обеспечение.

В ходе выполнения задания студент должен выполнить:

- определить этапы разработки программного обеспечения;
- построить концептуальную, логическую и физическую модель программного обеспечения и отдельного модуля;
- выбрать технологию разработки исходного модуля исходя из его назначения;
- выбрать метод и средства разработки, выполнить модификацию программных модулей.

Задание 2. Выполнить отладку программного модуля с использованием специализированных программных средств.

В ходе выполнения задания студент должен выполнить:

- выявить ошибки в программном модуле;
- определить возможность увеличения быстродействия программного продукта;
- определить способы оптимизации;
- выбрать метод и специализированные средства отладки программных модулей и программного продукта;
- произвести отладку программного продукта.

Задание 3. Осуществить разработку тестовых наборов и тестовых сценариев.

В ходе выполнения задания студент должен выполнить:

- разработать тестовый набор и тестовый сценарий;
- устранить ошибки в программных модулях;
- использовать методы тестирования программного обеспечения;
- внести изменения в программные модули для обеспечения качества программного обеспечения;
- правильно использовать инструментальные средства тестирования программных модулей.

Задание 4. Разработать и презентовать групповой проект информационной системы для конкретной задачи и отрасли:

1. Сформировать команду (от 3 до 5 человек). Разделить обязанности в группе (менеджер проекта, дизайнер интерфейса, проектировщик системы, оформитель материалов и пр.).

2. Выбрать предметную область и существующий в ней (реальный или воображаемый) бизнес-субъект (производственная компания, научноисследовательское предприятие, муниципальное учреждение и т.д.).

Кратко опишите свою компанию

3. Для выбранной организации обозначить проблему, которая может быть решена с помощью ИТ. На диаграмме Исикавы покажите степень влияния фактора ИТ на проблему.

4. Представить описание автоматизируемого процесса (в любой формальной нотации).

5. Составить матрицу пользовательских авторизаций ИС и (UML-диаграмму вариантов использования).

6. Составить дерево требований к ИС (включая требования информационной безопасности).

7. Перечислить документы (стандарты и другие регламенты), необходимые для реализации внедрения проектируемой ИС на каждом этапе ее ЖЦ.

8. Разработать общий паспорт-план проекта реализации и внедрения ИС с перечислением его основных параметров (участники, риски, вехи и т.д.).

9. Разработать объектную модель системы (UML-диаграмма классов);

10. Разработать архитектуру системы (UML-диаграмма компонентов и развертывания);

11. Представить примеры пользовательского интерфейса (пункты меню, формы и пр.);

12. Презентовать решение своей команды.

Критерии оценивания выполнения заданий экзамена

Оценка «отлично» - краткая, ясная и четкая констатация факта или события в ситуации, выход из ситуации найден, верно, на высоком профессиональном уровне, с правильными пояснениями. Обоснованность ответа. Необходимо мотивировать выбранный курс действий, приводящих к разрешению ситуации, и объяснить причины и рациональность его выбора.

Оценка «хорошо» - выход из ситуации найден в целом верно, но с небольшими неточностями, имеются неточности в пояснении.

Оценка «удовлетворительно» - выход из ситуации найден, верно, но не доведен до конца, либо в нем имеются ошибки, которые, однако, не приводят к принципиально неверному решению.

Оценка «неудовлетворительно» - выход из ситуации не найден или найден неверно