

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Саратовский государственный технический университет
имени Гагарина Ю.А.»

Филиал федерального государственного бюджетного образовательного
учреждения высшего образования
«Саратовский государственный технический университет
имени Гагарина Ю.А.» в г. Петровске



УТВЕРЖДАЮ
Директор филиала СГТУ
имени Гагарина Ю.А. в г.Петровске
Е.А.Бесшапошникова
«30» июня 2025 г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

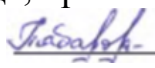
по дисциплине

МДК.02.01 «Технология разработки программного обеспечения»

направление подготовки

09.02.07 «Информационные системы и программирование»

Методические указания рассмотрены
на заседании предметной (цикловой) комиссии
общепрофессиональных дисциплин и
профессиональных модулей
«16» июня 2025 года, протокол №13

Председатель ПЦК  /Ю.А.Табарова/

Петровск 2025

Пояснительная записка

Методические указания по выполнению практических работ подготовлены на основе рабочей программы учебной дисциплины МДК.02.01 «Технология разработки программного обеспечения», разработанной на основе ФГОС СПО по специальности 09.02.07 «Информационные системы и программирование» и соответствующих общих (ОК) и профессиональных (ПК) 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 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 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 тестовых наборов и тестовых сценариев для программного обеспечения

ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования

При выполнении практических работ студент должен *знать*:

- 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;
- основы верификации и аттестации программного обеспечения;
- основные методы отладки;
- методы и схемы обработки исключительных ситуаций;
- основные методы и виды тестирования программных продуктов;
- приемы работы с инструментальными средствами тестирования и отладки;

- методы организации работы в команде разработчиков

При выполнении практических работ студент должен *уметь*:

- 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;
- оценивать размер минимального набора тестов;
- разрабатывать тестовые пакеты и тестовые сценарии;
- выявлять ошибки в системных компонентах на основе спецификаций;
- 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;
- использовать приемы работы в системах контроля версий;
- анализировать проектную и техническую документацию;
- использовать инструментальные средства отладки программных продуктов;
- выполнять отладку, используя методы и инструменты условной компиляции.

Содержание практических занятий определено рабочей программой и тематическим планированием, соответствует теоретическому материалу изучаемых разделов учебной дисциплины.

Объем практических занятий по дисциплине определяется учебным планом по данной специальности.

Продолжительность практического занятия – 2 академических часа. Перед проведением практического занятия преподавателем организуется инструктаж, а по ее окончании – обсуждение итогов.

Комплект методических указаний по выполнению практических работ по дисциплине МДК.02.01 «Технология разработки программного обеспечения» содержит 42 практических занятия.

**Перечень практических работ
по дисциплине МДК.02.01 «Технология разработки программного
обеспечения»**

ПРАКТИЧЕСКАЯ РАБОТА № 1

Тема: Анализ предметной области

ПРАКТИЧЕСКАЯ РАБОТА № 2

Тема: Анализ предметной области

ПРАКТИЧЕСКАЯ РАБОТА № 3

Тема: Анализ предметной области

ПРАКТИЧЕСКАЯ РАБОТА № 4

Тема: Анализ предметной области

ПРАКТИЧЕСКАЯ РАБОТА № 5

Тема: Разработка и оформление технического задания

ПРАКТИЧЕСКАЯ РАБОТА № 6

Тема: Разработка и оформление технического задания

ПРАКТИЧЕСКАЯ РАБОТА № 7

Тема: Разработка и оформление технического задания

ПРАКТИЧЕСКАЯ РАБОТА № 8

Тема: Разработка и оформление технического задания

ПРАКТИЧЕСКАЯ РАБОТА № 9

Тема: Работа в системе контроля версий

ПРАКТИЧЕСКАЯ РАБОТА № 10

Тема: Анализ и оценка цифровой безопасности и цифровых рисков

ПРАКТИЧЕСКАЯ РАБОТА № 11

Тема: Обзор, характеристики, особенности и преимущества использования планшетов/смартфонов

ПРАКТИЧЕСКАЯ РАБОТА № 12

Тема: Построение диаграммы вариантов использования

ПРАКТИЧЕСКАЯ РАБОТА № 13

Тема: Построение диаграммы вариантов использования

ПРАКТИЧЕСКАЯ РАБОТА № 14

Тема: Построение диаграммы последовательности

ПРАКТИЧЕСКАЯ РАБОТА № 15

Тема: Построение диаграммы последовательности

ПРАКТИЧЕСКАЯ РАБОТА № 16

Тема: Построение диаграммы кооперации

ПРАКТИЧЕСКАЯ РАБОТА № 17

Тема: Построение диаграммы кооперации

ПРАКТИЧЕСКАЯ РАБОТА № 18

Тема: Построение диаграммы развертывания

ПРАКТИЧЕСКАЯ РАБОТА № 19

Тема: Построение диаграммы развертывания

ПРАКТИЧЕСКАЯ РАБОТА № 20

Тема: Построение диаграммы деятельности

ПРАКТИЧЕСКАЯ РАБОТА № 21

Тема: Построение диаграммы деятельности

ПРАКТИЧЕСКАЯ РАБОТА № 22

Тема: Построение диаграммы состояний

ПРАКТИЧЕСКАЯ РАБОТА № 23

Тема: Построение диаграммы состояний

ПРАКТИЧЕСКАЯ РАБОТА № 24

Тема: Построение диаграммы классов

ПРАКТИЧЕСКАЯ РАБОТА № 25

Тема: Построение диаграммы классов

ПРАКТИЧЕСКАЯ РАБОТА № 26

Тема: Построение диаграммы компонентов

ПРАКТИЧЕСКАЯ РАБОТА № 27

Тема: Построение диаграммы компонентов

ПРАКТИЧЕСКАЯ РАБОТА № 28

Тема: Построение диаграмм потоков данных

ПРАКТИЧЕСКАЯ РАБОТА № 29

Тема: Построение диаграмм потоков данных

ПРАКТИЧЕСКАЯ РАБОТА № 30

Тема: Оценка необходимого количества тестов

ПРАКТИЧЕСКАЯ РАБОТА № 31

Тема: Оценка необходимого количества тестов

ПРАКТИЧЕСКАЯ РАБОТА № 32

Тема: Разработка тестового сценария.

ПРАКТИЧЕСКАЯ РАБОТА № 33

Тема: Разработка тестового сценария

ПРАКТИЧЕСКАЯ РАБОТА № 34

Тема: Разработка тестового сценария

ПРАКТИЧЕСКАЯ РАБОТА № 35

Тема: Разработка тестовых пакетов

ПРАКТИЧЕСКАЯ РАБОТА № 36

Тема: Разработка тестовых пакетов

ПРАКТИЧЕСКАЯ РАБОТА № 37

Тема: Разработка тестовых пакетов

ПРАКТИЧЕСКАЯ РАБОТА № 38

Тема: Оценка программных средств с помощью метрик

ПРАКТИЧЕСКАЯ РАБОТА № 39

Тема: Оценка программных средств с помощью метрик

ПРАКТИЧЕСКАЯ РАБОТА № 40

Тема: Оценка программных средств с помощью метрик

ПРАКТИЧЕСКАЯ РАБОТА № 41

Тема: Инспекция программного кода на предмет соответствия стандартам кодирования

ПРАКТИЧЕСКАЯ РАБОТА № 42

Тема: Инспекция программного кода на предмет соответствия стандартам кодирования

**ИНСТРУКЦИИ ДЛЯ ОБУЧАЮЩИХСЯ
ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ**

Прежде чем приступить к выполнению заданий, внимательно прочитайте данные рекомендации. Практические работы включают в себя задания следующих видов.

В ходе выполнения практических работ студент должен:

- выполнять требования по охране труда
- соблюдать инструкцию по правилам и мерам безопасности в кабинете информационных технологий
- строго выполнять весь объем работы, указанный в задании
- соблюдать требования эксплуатации компьютерной техники (правила включения и выключения)
- предоставить отчет о проделанной работе по окончании выполненной работы, который должен содержать:

1. Название работы.
2. Цель работы.
3. Задание и его решение.
4. Вывод о проделанной работе.

Текст отчета по практической работе должен быть набран на компьютере шрифтом Times New Roman размером 14 пт. (при оформлении текста используется текстовый редактор MS Word). Шрифт, используемый в иллюстративном материале (таблицы и рисунки), рекомендуется уменьшить до 12 пт. Межстрочный интервал в основном тексте - полуторный. В иллюстративном материале межстрочный интервал рекомендуется сделать одинарным. Поля страницы должны быть: левое поле - 30 мм; правое поле - 15 мм; верхнее и нижнее поле - 20 мм.

Каждый абзац должен начинаться с красной строки. Отступ абзаца - 1,25 см от левой границы текста.

Студент должен выполнить практическую работу самостоятельно (или в группе, если это предусмотрено заданием). Практическая работа выполняется согласно заданию и методическим рекомендациям. После выполнения практической работы обучающийся самостоятельно себя контролирует путем ответов на вопросы. Результат работы представляется преподавателю в виде файла (файлов) в личном каталоге, защищается обучающимися.

По ходу выполнения работы при возникновении вопросов обучающийся может получить консультацию у преподавателя или самостоятельно воспользоваться лекционным материалом, рекомендуемой литературой.

1. Оформление текстовых документов

Текст работы печатается на одной стороне листа формата А4, должен быть только чёрного цвета и иметь поля (верхнее, нижнее - 2 см, левое - 3 см, правое - 1,5 см). Шрифт Times New Roman размером 14, межстрочный интервал 1,5, абзацный отступ 1,25.

Правила оформления таблиц, рисунков, графиков

Все таблицы и рисунки должны иметь названия и порядковую нумерацию (например, Таблица 1, Рисунок 3). Нумерация таблиц и рисунков должна быть сквозной для всего текста до приложений. Таблицы, рисунки каждого приложения обозначают отдельной нумерацией арабскими цифрами с добавлением перед цифрой обозначения приложения (напр., Таблица В.1).

Оформление таблицы.

Название таблицы помещается слева над таблицей без абзацного отступа, в одной строке с ее номером через тире (14 шрифтом).

В каждой таблице следует указывать единицы измерения показателей. Если единица измерения в таблице является общей для всех числовых табличных данных, то ее приводят в заголовке таблицы после ее названия.

При переносе: слово “Таблица” указывают один раз слева над первой частью таблицы, над другими частями пишут слова “Продолжение таблицы” или “Окончание таблицы” справа, с указанием номера (обозначения) таблицы. Если в конце страницы таблица прерывается и ее продолжение будет на следующей странице, то в первой части таблицы нижнюю горизонтальную черту, ограничивающую таблицу, не проводят.

Таблица 1 – Распределение ответов респондентов на вопросы анкеты по возрастным группам (в процентах)

Варианты ответов	Возрастные группы				Всего по выборке
	18-24 года	25-29 лет	30-45 лет	старше 45 лет	

Не допускается прямое копирование в текст Диплома выходных таблиц отчета компьютерной программы STATISTICA. Таблицы должны быть построены заново.

Оформление рисунка.

Все иллюстративные материалы (рисунки, диаграммы, графики) в Дипломе имеют название «Рисунок». На графический материал должна быть дана ссылка в тексте документа.

Иллюстрации могут быть в компьютерном исполнении, в том числе и цветные.

Порядковый номер рисунка и – через тире – его название проставляются под рисунком по центру строки (см. Рисунок 1).



Рисунок 1 – Пятиконечная звезда

2. Составление программы на языке программирования

Правила оформления кода:

1. Используйте разумные имена для переменных и функций

Программа должна быть хорошо понятна человеку при чтении. Если при чтении программы приходится понимать назначение переменных и функций по тому, как они используются, то читать код становится гораздо сложнее. Неудачно выбранные имена могут привести к тому, что смысл программы может быть неправильно понят. Выбирайте такие имена, которые

бы объясняли смысл переменных и функций, тогда код станет гораздо понятнее, и не потребуется писать множество комментариев.

При написании составных слов, например в именах переменных, пишите их слитно без пробелов, при этом каждое новое слово пишется с большой буквы.

2. Не дублируйте код

Если в программе есть одинаковые выражения или фрагмента кода, вынесите этот код в отдельную функцию. Верным признаком необходимости создания новой функции является желание скопировать фрагмент кода из одного места программы в другое. В таком случае сразу перенесите этот фрагмент в отдельную функцию.

Если фрагменты кода похожи, но не идентичны, подумайте, не получится ли и их вынести в одну функцию, возможно добавив параметры или условия.

3. Не используйте «магические константы»

Использование неименованных «магических» констант в коде нежелательно:

- при чтении кода может быть не понятно, что это за число, и почему оно именно такое;
- чаще всего одно и то же число потребуется написать в нескольких местах кода. Если его придётся изменять, можно пропустить одно из использований, что приведёт к ошибке.

Если в коде нужно использовать константу, дайте ей имя, используя `const`.

4. Расставляйте пробелы вокруг бинарных операторов. Это улучшает читаемость формул.

5. Всегда выделяйте блоки условных операторов и циклов скобками. В любой блок условия или цикла может захотеться добавить новое выражение. При этом можно забыть добавить скобки. Лучше сразу добавить скобки, чтобы потом не было с этим проблем.

6. Расставляйте скобки одинаково. Выберите и используйте для себя один из стилей расстановки скобок. Это улучшает читаемость структуры программы.

7. Не делайте строки слишком длинными. Строка программы должна помещаться на экране. Обычно рекомендуют ограничить максимальную ширину строки в 80 символов. Если определение или вызов функции получается слишком широким поместите по одному параметру на каждой строке. Длинные математические выражения разбивайте на несколько строк, разбивая по границам логических блоков выражения.

8. Объявляйте переменные непосредственно перед использованием. Обязательно указывайте начальное значение для переменных. Значение неинициализированной переменной может быть любым. Использование (чтение) такого значения приведёт к недетерминированной работе программы, а в некоторых случаях является неопределённым поведением. Чтобы избежать проблем всегда инициализируйте переменные прямо в момент их создания.

9.Единый стиль оформления кода во всем проекте;

10. Визуальное выделение наиболее значимых частей — используя *вертикальное форматирование*, мы выделяем объявление переменных, цикл заполнения массива случайными числами и цикл обработки по формуле. Если ваша функция выполняет несколько действий — то разумно разделить соответствующие блоки кода пустыми строками.

Цель работы: описать и проанализировать ИС, определить необходимые элементы комплекса технических средств ИС и системного программного обеспечения ИС.

Оборудование: ПК, программное обеспечение – MS Word, инструкции по выполнению работы.

Справочный материал:

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. Не существует четкой зависимости между процессом создания ПО и типом создаваемого программного продукта. 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 значительно отличаются от проектов, реализованных ранее. Поэтому, чтобы уменьшить неопределенность в планировании проекта, руководители проектов должны обладать большим практическим опытом. Но постоянные технологические изменения в компьютерной технике оценивают предыдущий опыт. Перечисленные особенности могут привести к тому, что реализация проекта выйдет за рамки временного графика или бюджета.

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, заказавшей программный проект, могут существенно измениться. Может оказаться, что разрабатываемый программный продукт стал уже ненужным или 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 работа 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, отмечающие окончание определенного этапа работ. Для каждой точки создается отчет, который предоставляется руководству проекта.

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

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.

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.

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.

Содержание работы:

Задание 1.

Предметная область «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

1. Выберите название ИС в рамках предметной области

2. Определите цель ИС

3. Проведите анализ осуществимости ИС.

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

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

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

3.4. Требуется ли разработка ИС технологии, которая до этого раньше не использовалась в организации?

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

5. Комплекс технических средств ИТ

5.1. Какие средства компьютерной техники необходимы для ИС?

5.2. Какие средства коммуникационной техники необходимы для ИС?

- 5.3.Какие средства организационной техники необходимы для ИС?
- 5.4.Какие средства оперативной полиграфии необходимы для ИС?
- 6.Опишите системное ПО ИТ.

ПРАКТИЧЕСКАЯ РАБОТА № 2
Тема: Анализ предметной области

Цель работы: описать и проанализировать ИС, определить необходимые элементы комплекса технических средств ИС и системного программного обеспечения ИС.

Оборудование: ПК, программное обеспечение – MS Word, инструкции по выполнению работы.

Содержание работы:

Задание 1.

Предметная область «Туроператор»

Туроператор предоставляет возможность своим клиентам осуществить туристическую или деловую поездку в различные города России и мира. При разработке нового тура сначала анализируется текущая ситуация на рынке туризма и выбирается направление тура. После этого определяется статус тура, бронируются места в гостиницах и билеты на переезд к месту тура, разрабатывается культурная/ деловая/ развлекательная программа, утверждаются сроки тура. На каждый тур назначается ответственное лицо от туроператора, которое будет вести данный тур для улаживания проблем в случае возникновения каких-нибудь чрезвычайных или форс-мажорных ситуаций. Клиент приходит в офис туроператора, где вместе с менеджером выбирает уже разработанный тур и оформляет путевку. После возвращения из тура клиент может высказать свои замечания или пожелания, которые будут учтены при доработке существующих туров или при разработке новых. Также, для дальнейшего улучшения тура, туроператор проводит анализ отчетов от посредников (гостиница, гиды и т.д.). По результатам своей деятельности туроператор производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики

1. Выберите название ИС в рамках предметной области

2. Определите цель ИС

3. Проведите анализ осуществимости ИС.

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

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

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

3.4. Требуется ли разработка ИС технологии, которая до этого раньше не использовалась в организации?

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

5. Комплекс технических средств ИТ

5.1. Какие средства компьютерной техники необходимы для ИС?

5.2. Какие средства коммуникационной техники необходимы для ИС?

5.3. Какие средства организационной техники необходимы для ИС?

5.4. Какие средства оперативной полиграфии необходимы для ИС?

6. Опишите системное ПО ИТ.

ПРАКТИЧЕСКАЯ РАБОТА № 3

Тема: Анализ предметной области

Цель работы: описать и проанализировать ИС, определить необходимые элементы комплекса технических средств ИС и системного программного обеспечения ИС.

Оборудование: ПК, программное обеспечение – MS Word, инструкции по выполнению работы.

Содержание работы:

Задание 1.

Предметная область «Компания по разработке программных продуктов»

Компания заключает договор с клиентом на разработку программного продукта согласно техническому заданию. После 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.

1. Выберите название ИС в рамках предметной области

2. Определите цель ИС

3. Проведите анализ осуществимости ИС.

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

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

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

3.4. Требуется ли разработка ИС технологии, которая до этого раньше не использовалась в организации?

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

5. Комплекс технических средств ИТ

5.1. Какие средства компьютерной техники необходимы для ИС?

5.2. Какие средства коммуникационной техники необходимы для ИС?

5.3. Какие средства организационной техники необходимы для ИС?

5.4. Какие средства оперативной полиграфии необходимы для ИС?

6. Опишите системное ПО ИТ.

ПРАКТИЧЕСКАЯ РАБОТА № 4

Тема: Анализ предметной области

Цель работы: описать и проанализировать ИС, определить необходимые элементы комплекса технических средств ИС и системного программного обеспечения ИС.

Оборудование: ПК, программное обеспечение – MS Word, инструкции по выполнению работы.

Содержание работы:

Задание 1.

Предметная область «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

1. Выберите название ИС в рамках предметной области
2. Определите цель ИС
3. Проведите анализ осуществимости ИС.
 - 3.1. Что произойдет с организацией, если система не будет введена в эксплуатацию?
 - 3.2. Какие текущие проблемы существуют в организации и как новая система поможет их решить?
 - 3.3. Каким образом (и будет ли) ИС способствовать целям бизнеса?
 - 3.4. Требуется ли разработка ИС технологии, которая до этого раньше не использовалась в организации?
4. Где будет размещена ИС? Кто является пользователем ИС?
5. Комплекс технических средств ИТ
 - 5.1. Какие средства компьютерной техники необходимы для ИС?
 - 5.2. Какие средства коммуникационной техники необходимы для ИС?
 - 5.3. Какие средства организационной техники необходимы для ИС?
 - 5.4. Какие средства оперативной полиграфии необходимы для ИС?
6. Опишите системное ПО ИТ.

ПРАКТИЧЕСКАЯ РАБОТА № 5

Тема: Разработка и оформление технического задания

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

Оборудование: ПК, программное обеспечение – MS Word, инструкции по выполнению работы.

Справочный материал:

Техническое задание (ТЗ) – исходный документ для разработки информационных систем.

ТЗ содержит основные технические требования, предъявляемые к программному продукту, и исходные данные для разработки. В ТЗ указываются назначение программного продукта, область его применения, стадии разработки конструкторской, проектной, программной документации, ее состав, сроки исполнения, а также особые требования, обусловленные спецификой продукта либо условиями его эксплуатации. Как правило, ТЗ составляется на основе анализа результатов предварительных требований, исследований. Типовые требования к составу и содержанию ТЗ приведены в таблице.

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

№ п/п	Раздел	Содержание
1	Общие сведения	<ul style="list-style-type: none">- полное наименование системы и ее условное обозначение- шифр темы или шифр (номер) договора;- наименование предприятия 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	<ul style="list-style-type: none">- 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	<ul style="list-style-type: none">- краткие сведения об объекте автоматизации- сведения об условиях эксплуатации и характеристиках окружающей среды
4	Требования к	Требования к системе в целом:

	системе	<ul style="list-style-type: none"> - 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 <p>Требования к функциям (по подсистемам):</p> <ul style="list-style-type: none"> - 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 <p>Требования к видам обеспечения:</p> <ul style="list-style-type: none"> - математическому (состав и область применения математических 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
--	---------	---

		<p>средств и способы его контроля, использование фондов алгоритмов и программ)</p> <ul style="list-style-type: none"> - 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	<ul style="list-style-type: none"> - 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	<ul style="list-style-type: none"> - 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	<ul style="list-style-type: none"> - преобразование входной информации к машиночитаемому виду - изменения в объекте автоматизации - сроки и порядок комплектования и обучения персонала
8	Требования к документированию	<ul style="list-style-type: none"> - перечень подлежащих разработке документов - перечень документов на машинных носителях
9	Источники разработки	<ul style="list-style-type: none"> - документы и информационные материалы, на основании которых разрабатывается ТЗ и система

Содержание работы:

Разработка ТЗ 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 ПО: конкретную комплектацию и параметры технических средств, версию используемой ОС и, возможно, версии и параметры другого установленного ПО, с которым предстоит взаимодействовать будущему программному продукту.

В случаях, когда разрабатываемое ПО 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 1.

1. Разработать техническое задание по информационной системе из практической работе №1 «Страховая медицинская компания»
2. Оформить отчет

Порядок выполнения отчета по практической работе

1. Разработать техническое задание на программный продукт
2. Оформить работу в соответствии с ГОСТ 19.106-78. При оформлении использовать MS Office.
3. Сдать и защитить работу

Защита отчета по практической работе заключается в предъявлении преподавателю полученных результатов (на экране монитора и печатном виде), демонстрации полученных навыков и ответах на вопросы преподавателя.

ПРАКТИЧЕСКАЯ РАБОТА № 6

Тема: Разработка и оформление технического задания

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

Оборудование: ПК, программное обеспечение – MS Word, инструкции по выполнению работы

Содержание работы:

1. Разработать техническое задание по информационной системе из практической работе №2 «Туроператор»
2. Оформить отчет

Порядок выполнения отчета по практической работе

1. Разработать техническое задание на программный продукт
2. Оформить работу в соответствии с ГОСТ 19.106-78. При оформлении использовать MS Office.
3. Сдать и защитить работу

Защита отчета по практической работе заключается в предъявлении преподавателю полученных результатов (на экране монитора и печатном виде), демонстрации полученных навыков и ответах на вопросы преподавателя.

ПРАКТИЧЕСКАЯ РАБОТА № 7

Тема: Разработка и оформление технического задания

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

Оборудование: ПК, программное обеспечение – MS Word, инструкции по выполнению работы

Содержание работы:

1. Разработать техническое задание по информационной системе из практической работе №3 «Компания по разработке программных продуктов»
2. Оформить отчет

Порядок выполнения отчета по практической работе

1. Разработать техническое задание на программный продукт
2. Оформить работу в соответствии с ГОСТ 19.106-78. При оформлении использовать MS Office.
3. Сдать и защитить работу

Защита отчета по практической работе заключается в предъявлении преподавателю полученных результатов (на экране монитора и печатном виде), демонстрации полученных навыков и ответах на вопросы преподавателя.

ПРАКТИЧЕСКАЯ РАБОТА № 8

Тема: Разработка и оформление технического задания

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

Оборудование: ПК, программное обеспечение – MS Word, инструкции по выполнению работы

Содержание работы:

1. Разработать техническое задание по информационной системе из практической работе №4 «Агентство недвижимости»
2. Оформить отчет

Порядок выполнения отчета по практической работе

1. Разработать техническое задание на программный продукт
2. Оформить работу в соответствии с ГОСТ 19.106-78. При оформлении использовать MS Office.
3. Сдать и защитить работу

Защита отчета по практической работе заключается в предъявлении преподавателю полученных результатов (на экране монитора и печатном виде), демонстрации полученных навыков и ответах на вопросы преподавателя.

Цель работы: научиться использовать систему контроля версий git при работе с программным кодом

Оборудование: ПК, программное обеспечение – GitHub, MS Word, инструкции по выполнению работы

Справочный материал:

Git — это набор консольных утилит, которые отслеживают и фиксируют изменения в файлах (чаще всего речь идет об исходном коде программ, но можно использовать его для любых файлов).

С его помощью можно откатиться на более старую версию вашего проекта, сравнивать, анализировать, сливать изменения и многое другое, т.е. проводить контроль версий (или управление версиями). Существуют различные системы для контроля версий, например SVN, Mercurial, Perforce, CVS, Bitkeeper и др.

Git является распределенным, т.е. не зависит от одного центрального сервера, на котором хранятся файлы. Вместо этого он работает полностью локально, сохраняя данные в папках на жестком диске, которые называются репозиторием. Тем не менее, вы можете хранить копию репозитория онлайн, это облегчает работу над одним проектом для нескольких людей. Для этого используются различные сайты, например github и bitbucket.

Содержание работы:

1. Создать аккаунт на <https://github.com>
 - a. логин = первая буква имени и фамилия в латинице. Например, логин для "Василий Колесников" будет vkolesnikov
 - b. пароль произвольный
2. Создать репозиторий на github.com НЕ ДОБАВЛЯЯ В НЕГО README.MD
3. Инициализировать новый репозиторий на компьютере (создать отдельный каталог).
4. Все необходимые команды написаны в репозитории после создания, но надо понимать, что они делают и когда их выполнять:
 - a. Команда git init создает в ТЕКУЩЕМ каталоге новый локальный репозиторий (выполняется один раз):git init
 - b. Прежде чем добавлять файлы в репозиторий нужно создать в корне репозитория файл .gitignore и записать туда игнорируемые файлы и каталоги.Для C# там надо написать:# в каталогах bin и obj записываются скомпилированные файлы */bin/ */obj/ # в каталоге .vs хранятся локальные настройки VisualStudio .vs/ # в каталоге packages хранятся зависимости packages/
 - c. Для записи коммитов в репозиторий пользователь должен быть идентифицирован. Для этого выполните команды (один раз для репозитория):git config user.name "Ваше имя" git config user.email "Ваша почта" Если вы работаете дома, то можете один раз выполнить эти команды с флагом --global

d. В колледже используется прокси, из-за которого git может ругаться на не верный сертификат. В этом случае отключите проверку сертификата: `git config --global http.sslVerify false`

e. Команда `git add <имя файла>` помечает файл как отслеживаемый системой контроля версий. Вместо конкретного имени можно указать "." (символ точки), чтобы добавить в отслеживаемые все изменившиеся файлы. Выполняется перед каждым коммитом. `git add .`

f. Команда `git commit` сохраняет отслеживаемые файлы в локальный репозиторий. Выполняется при завершении какого-то законченного атомарного действия (создали класс, написали функцию...). `git commit -m "текст комментария"`

g. Команда `git remote add <алиас> <url>` добавляет в настройки локального репозитория ссылку на внешний репозиторий. Выполняется один раз для каждого внешнего репозитория (их может быть более одного). `git remote add origin <ссылку на проект копируйте из внешнего репозитория>`

h. Команда `git push [-u] <алиас внешнего репозитория> <название ветки>` отправляет содержимое локального репозитория в указанный внешний репозиторий. Ветка по-умолчанию в Git-е: *master*. Ключ -u задает комбинацию алиас + внешний репозиторий по-умолчанию, т.е. в следующий раз можно использовать просто команду `git push`. `git push origin master`

i. если на компьютере установлен credential manager, то он может запоминать последнего пользователя и не разрешить запись в другой репозиторий. В этом случае удаляем его из системы: `git config --system --unset credential.helper`

j. для GOGS (что-то с буфером передачи) `git config --global http.postBuffer 157286400`

5. Исследовать команды, описанные в лекции. Команды и возвращаемый результат записать в `readme.md`, используя формат Markdown.

a. куски кода или консольный ввод/вывод в Markdown-е обозначаются тройными обратными кавычками: `````
Здесь кусок кода `````

6. Скопировать из лекции фрагмент оформления `readme`, отредактировать его и вставить в начало своего `readme.md`

7. Результаты опубликовать в репозитории, созданном в п.2 и скинуть ссылку преподавателю.

Контрольные вопросы

1. Что такое VCS? Что такое Git? Почему его используют?
2. Как создать репозиторий, подключить внешний репозиторий?
3. Как загрузить удаленный репозиторий?
4. Что такое коммит? Как посмотреть историю коммитов?
5. Что такое ветка в Git?
6. Как отправить свои изменения на удаленный репозиторий?
7. Как добавить изменения в уже созданный коммит? изменить название такого коммита?
8. Разница между git и svn (если есть)?

ПРАКТИЧЕСКАЯ РАБОТА № 10

Тема: Анализ и оценка цифровой безопасности и цифровых рисков

Цель работы: ознакомиться с алгоритмами оценки риска информационной безопасности; изучить методы оценивания информационных рисков; научиться анализировать и оценивать цифровую безопасность и цифровые риски.

Оборудование: ПК, программное обеспечение – MS Word, инструкции по выполнению работы

Справочный материал:

Оценка рисков нарушения информационной безопасности компьютерных систем является одной из важнейших составляющих процесса управления информационной безопасностью.

Риск информационной безопасности – потенциальная возможность использования определенной угрозой уязвимостей актива или группы активов для причинения вреда организации.

Уязвимость – слабость в системе защиты, делающая возможной реализацию угрозы. Угроза информационной безопасности – совокупность условий и факторов, которые могут стать причиной нарушений целостности, доступности, конфиденциальности информации.

Информационный актив – это материальный или нематериальный объект, который:

- является информацией или содержит информацию,
- служит для обработки, хранения или передачи информации,
- имеет ценность для организации.

Анализ рисков представляет собой процедуры выявления факторов рисков и оценки их значимости, по сути, анализ вероятности того, что произойдут определенные нежелательные события и отрицательно повлияют на достижение целей проекта. Анализ рисков включает оценку рисков и методы снижения рисков или уменьшения связанных с ним неблагоприятных последствий, т.е. идентификацию и вычисление уровней (мер) рисков на основе оценок, присвоенных ресурсам, угрозам и уязвимостям ресурсов.

Назначение анализа рисков – дать потенциальным партнерам необходимые данные для принятия решений о целесообразности участия в проекте и выработки мер по защите от возможных финансовых потерь.

Анализ рисков можно подразделить на два взаимно дополняющих друг друга вида: качественный и количественный. *Качественный анализ* имеет целью определить (идентифицировать) факторы, области и виды рисков. *Количественный анализ* рисков должен дать возможность численно определить размеры отдельных рисков и риска проекта в целом.

Контроль рисков состоит в идентификации и выборе контрмер, позволяющих снизить риски до приемлемого уровня.



Целью **оценки рисков** является выявление следующих положений:

- приемлемость существующих рисков;
- определение неприемлемых рисков, которые в первую очередь нуждаются в уменьшении;
- определение защитных средств, экономически целесообразных для уменьшения неприемлемых рисков.

Содержание работы:

Задание 1. Работа со стандартом

1.Загрузите ГОСТ Р ИСО/МЭК 27005—2010 «Информационная технология. Методы и средства обеспечения безопасности. Менеджмент риска информационной безопасности»

2.Ознакомьтесь с Приложениями С, D и E ГОСТа.

3.Выберите три различных информационных актива организации (см. вариант).

Номер варианта	Организация	Метод оценки риска (см. Приложение E ГОСТа)
1	Отделение коммерческого банка	1
2	Поликлиника	2
3	Колледж	3
4	Офис страховой компании	4
5	Интернет-магазин	2
6	Центр оказания государственных услуг	3
7	Отделение полиции	4
8	Аудиторская компания	1
9	Дизайнерская фирма	2
10	Офис адвоката	4
11	Компания по разработке сторонних организаций	1
12	Агентство недвижимости	2
13	Офис благотворительного фонда	4
14	Туристическое агентство	3
15	Издательство	1
16	Отделение налоговой службы	4
17	Бюро перевода документов	2
18	Рекламное агентство	3
19	Офис нотариуса	1
20	Гостиница	2
21	Научно-проектное предприятие	3
22	Городской архив	4
23	Железнодорожная касса	2
24	Праздничное агентство	3
25	Редакция газеты	1

4.Из Приложения D ГОСТа подберите три конкретных уязвимости системы защиты указанных информационных активов.

5.Пользуясь Приложением С ГОСТа напишите три угрозы, реализация которых возможна пока в системе не устранены названные в пункте 4 уязвимости.

6.Пользуясь одним из методов (см. вариант) предложенных в Приложении Е ГОСТа произведите оценку рисков информационной безопасности.

7.Оценку ценности информационного актива производить на основании возможных потерь для организации в случае реализации угрозы.

Содержание отчета

- 1.Титульный лист
- 2.Содержание
- 3.Задание
- 4.Обоснование выбора информационных активов организации
- 5.Оценка ценности информационных активов
- 6.Уязвимости системы защиты информации
- 7.Угрозы информационной безопасности
- 8.Оценка рисков
- 9.Выводы

Задание 2. Оценить риски информационной системы

1. Проанализировать систему информационной безопасности конкретного предприятия (рассмотреть имеющиеся средства и методы защиты информации).
2. Произвести оценку рисков существующей системы безопасности, результаты свести в таблицу

Наименование угрозы	Вероятность наступления	Ущерб от реализации	Риск
Стихийные бедствия, аварии, пожары и т.п.			
Непреднамеренные ошибки пользователей			
Перебои электропитания			
Вредоносное ПО			
Халатность пользователей			
Другое			
Сумма рисков			

Проставить в таблице коэффициент вероятности наступления и коэффициент ущерба от реализации угрозы по 3х бальной шкале. Произведение этих составляющих позволят определить риск. Рассчитать

общую сумму рисков. На основе полученных данных выделить три типа риска: низкий (1,2), средний (3,4), высокий (6,9). Построить диаграмму оценки рисков по категориям. Сделать выводы.

3. Предложить средства улучшения системы информационной безопасности предприятия, охарактеризовать каждое из них. Провести сравнительный анализ этих средств с их аналогами (допускается использование данных социологических опросов; характеристики технических средств должны соответствовать реальным).

4. Провести оценку рисков разработанной системы безопасности и свести данные в таблицу 2 – Оценка рисков разработанной системы безопасности, аналогичную таблице 1. Построить диаграмму количества рисков после принятия разработанной политики безопасности. Сделать выводы.

5. Сделать общий вывод (можно ли считать разработанную систему безопасности эффективной, ответ аргументировать).

ПРАКТИЧЕСКАЯ РАБОТА № 11

Тема: Обзор, характеристики, особенности и преимущества использования планшетов/смартфонов

Цель работы: ознакомиться с различными типами и видами мобильных устройств, их особенностями и преимуществами использования.

Оборудование: ПК, программное обеспечение – MS Word, инструкции по выполнению работы

Содержание работы:

Задание 1. Рассмотреть виды планшетов, заполнить следующую таблицу:

Модель планшета	Экран	Производительность	Аккумулятор	Камера	Возможности

Сделать вывод

Задание 2. Рассмотреть виды смартфонов, заполнить следующую таблицу:

Модель смартфона	Экран	Производительность	Аккумулятор	Камера	Возможности

Сделать вывод

Задание 3. По итогам выполнения заданий 2 и 3 сделать общий вывод по использованию планшетов и смартфонов при работе с различными видами программного обеспечения.

ПРАКТИЧЕСКАЯ РАБОТА № 12

Тема: Построение диаграммы вариантов использования

Цель работы: исследование процесса построения диаграмм вариантов использования в заданной предметной области; научиться строить диаграммы вариантов использования с применением ПО

Оборудование: ПК, программное обеспечение – MS Word, Visual Paradigm Online, инструкции по выполнению работы.

Справочный материал:

Диаграмма вариантов использования (use case diagram) — это графическое представление взаимодействия между внешними сущностями (актёрами) и функциями системы, которые необходимы этим актёрам. Цель диаграммы — описать функциональное назначение системы, то, что она должна делать в процессе своего функционирования.

Элементы

Актёры — представляют роли, взаимодействующие с системой. Могут быть первичными (инициируют взаимодействие) или вторичными (предоставляют услуги системе).

Варианты использования (use cases) — описывают значимые действия системы. Изображаются в виде овалов с подписями.

Отношения — соединяют актёров с вариантами использования, показывают взаимодействие и зависимости. Некоторые типы связей:

- *Ассоциация* — базовая связь актёра с вариантом использования.
- *Обобщение* — наследование между актёрами или вариантами использования
- *Включение* — обязательная часть варианта использования.
- *Расширение* — опциональное поведение.

Границы системы — прямоугольная рамка, охватывающая все варианты использования.

Правила построения

- 1.Использовать описательные имена для актёров и вариантов использования, чтобы обеспечить ясность и понимание.
- 2.Формулировать варианты использования с точки зрения ценности для пользователя — например, «Оформить заказ» вместо «Нажать кнопку Купить».
- 3.Избегать избыточного использования отношений — особенно «include» и «extend».
- 4.Смешивать уровни детализации — на одной диаграмме одни варианты использования слишком общие, другие чрезмерно детализированы.

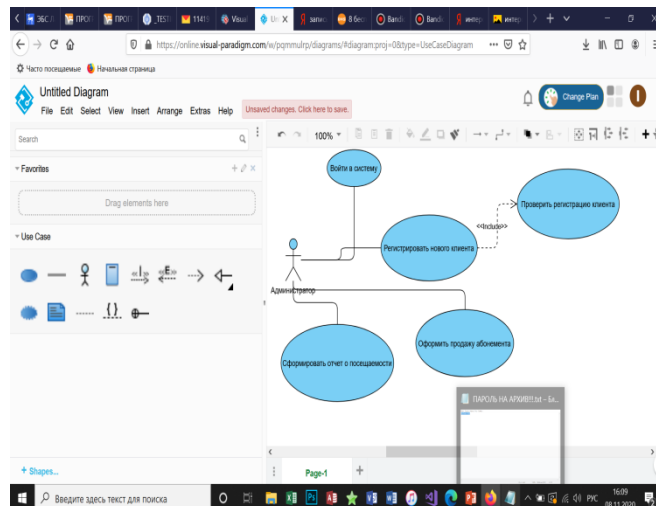
Содержание работы:

Задание 1. Создать диаграмму вариантов использования Моделирование системы продажи товаров по каталогу.

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 следующие варианты использования

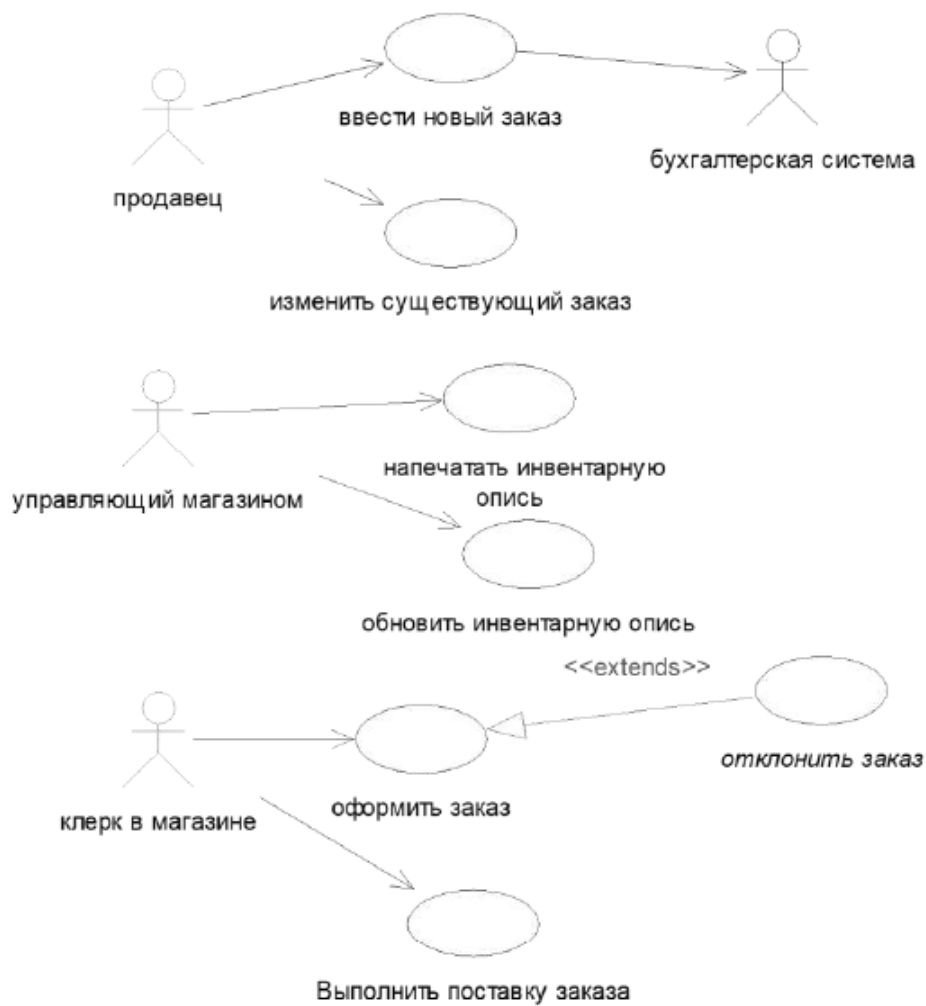
1. Запустите сервис Visual Paradigm Online по ссылке <https://online.visual-paradigm.com/>. Перейдите на страницу «Программное Обеспечение для онлайн-Диаграмм» и нажмите кнопку «Начните работать бесплатно»
2. На появившейся странице в правом верхнем углу нажмите кнопку «Регистрация». Зарегистрируйтесь в системе, для использования бесплатной версии.
3. Нажмите кнопку «создать новый» и выберите **use case**. Должно появиться окно с инструментами для создания диаграмм вариантов использования:



Выполните данную диаграмму в среде Visual Paradigm Online



Задание 2. Создание диаграммы вариантов использования и действующих лиц по образцу и с описанием действием



Задание 3. Построить диаграмму вариантов использования.

Имеются следующие данные:

- 1.четыре действующих лица: Клиента банка, Банк, Кассира и Оператора,
- 2.пять вариантов использования: Снять наличные, Перевести деньги со счета, Положить деньги на счет, Пополнить запас денег и Подтвердить пользователя,
- 3.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 зависят 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.

ПРАКТИЧЕСКАЯ РАБОТА № 13

Тема: Построение диаграммы вариантов использования

Цель работы: исследование процесса построения диаграмм вариантов использования в заданной предметной области; научиться строить диаграммы вариантов использования с применением ПО

Оборудование: ПК, программное обеспечение – MS Word, Smart Draw или UML Diagram, инструкции по выполнению работы.

Содержание работы:

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
- 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.

Задания для самостоятельной работы:

- 1.Построить диаграмму вариантов использования «Заказ товара в интернет-магазине»
- 2.Построить диаграмму вариантов использования «Продажа товаров по каталогу».

ПРАКТИЧЕСКАЯ РАБОТА № 14

Тема: Построение диаграммы последовательности

Цель работы: исследование процесса построения диаграмм последовательности в заданной предметной области; научиться строить диаграммы последовательности с применением ПО

Оборудование: ПК, программное обеспечение – MS Word, Smart Draw или UML Diagram, инструкции по выполнению работы.

Справочный материал:

Диаграмма последовательности (sequence diagram) — это наглядное представление совокупности разных элементов модели системы, изображение того, как и в каком порядке они взаимодействуют. Такие диаграммы показывают временной порядок или хронологию: то, когда, как и в какой очереди передаются сообщения.

Диаграммы последовательности ориентированы на время и визуально показывают порядок взаимодействия, используя вертикальную ось диаграммы для представления времени

Элементы

Объекты — сущности, которые взаимодействуют друг с другом. Представлены прямоугольниками с именами внутри, помимо названия объекта указывают его класс (разделяются двоеточием).

Линия жизни (lifeline) — вертикальная линия, которая представляет объект или участника взаимодействия и связывает его с сообщениями во времени. Начинается с появления объекта на диаграмме и продолжается до его удаления или окончания взаимодействия.

Сообщения — стрелки, направленные от одной линии жизни к другой. Показывают обмен информацией между объектами, например, запрос или ответ, вызов хранимой процедуры или отправку сообщения.

Фрагмент выполнения (активация) — узкий прямоугольник на линии жизни, который показывает начало и завершение действия с участием объекта, его активизации во времени.

Правила построения:

1. Изображать исключительно те объекты, которые непосредственно участвуют во взаимодействии, не показывать возможные статические ассоциации с другими объектами.
2. Учитывать порядок инициализации сообщений — сообщения, расположенные на диаграмме выше, иницируются раньше тех, которые расположены ниже.
3. Для сложных сценариев (ветвления, параллельные процессы) — визуализировать каждый поток управления на отдельной диаграмме последовательности.

Содержание работы:

Задание 1. Построить диаграмму последовательности для системы продажи товаров по каталогу для каждого варианта использования и для всей системы в целом

1. Запустите сервис SmartDraw <https://app.smartdraw.com/>

2.Нажмите кнопку Запустить приложение

3.Выберите диаграмму последовательности UML Sequence Diagram

4.Построить диаграмму последовательности для системы продажи товаров по каталогу для каждого варианта использования и для всей системы в целом:

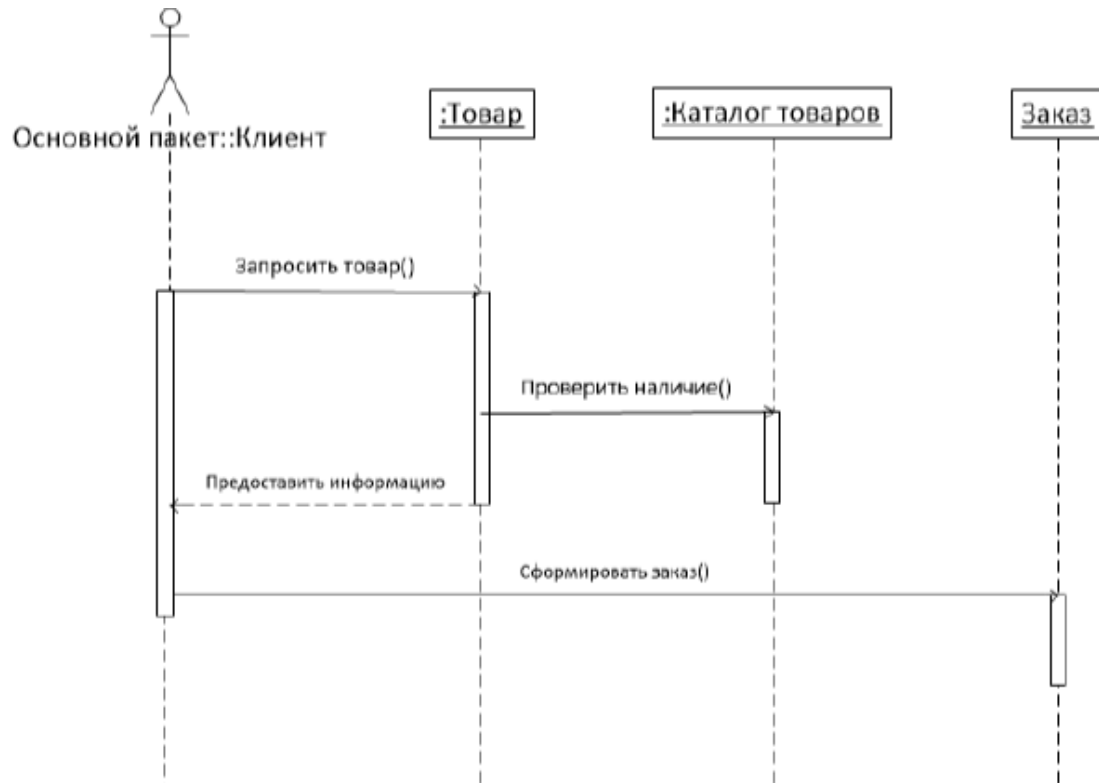


Диаграмма последовательности для варианта использования «Обеспечить покупателя информацией»

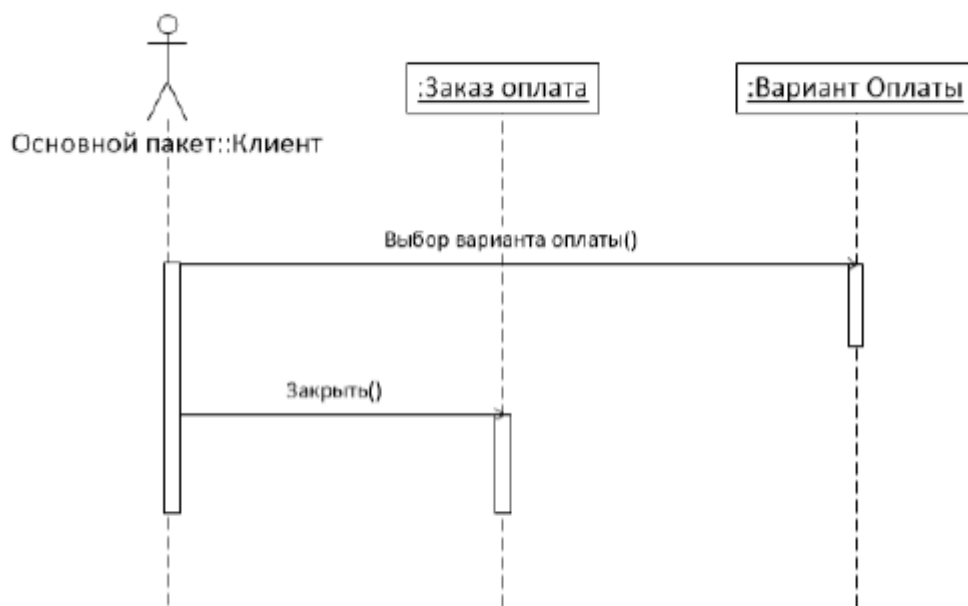


Диаграмма последовательности для варианта использования «Согласовать условия оплаты»

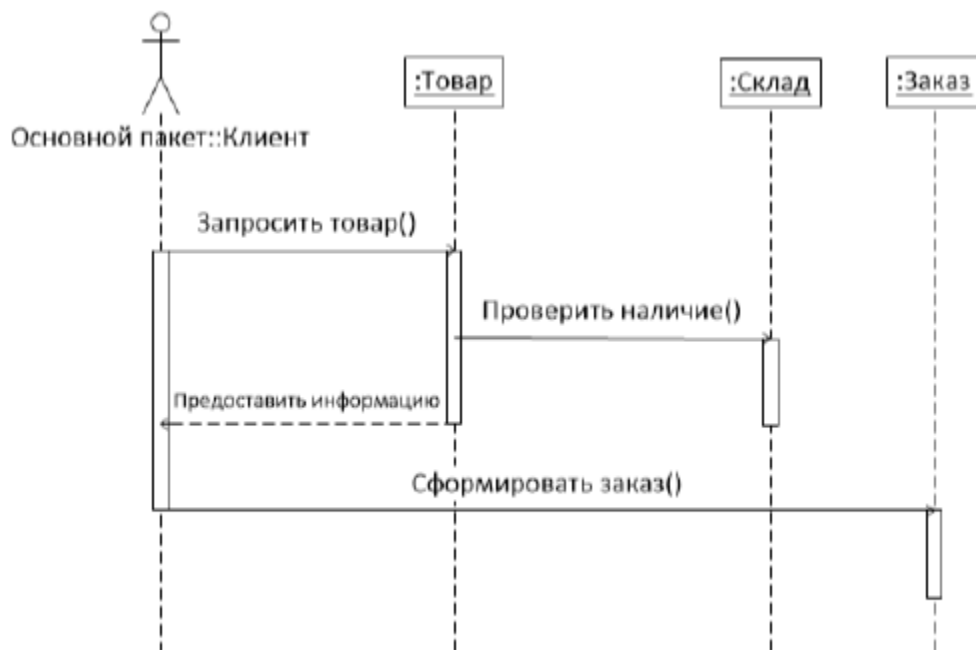
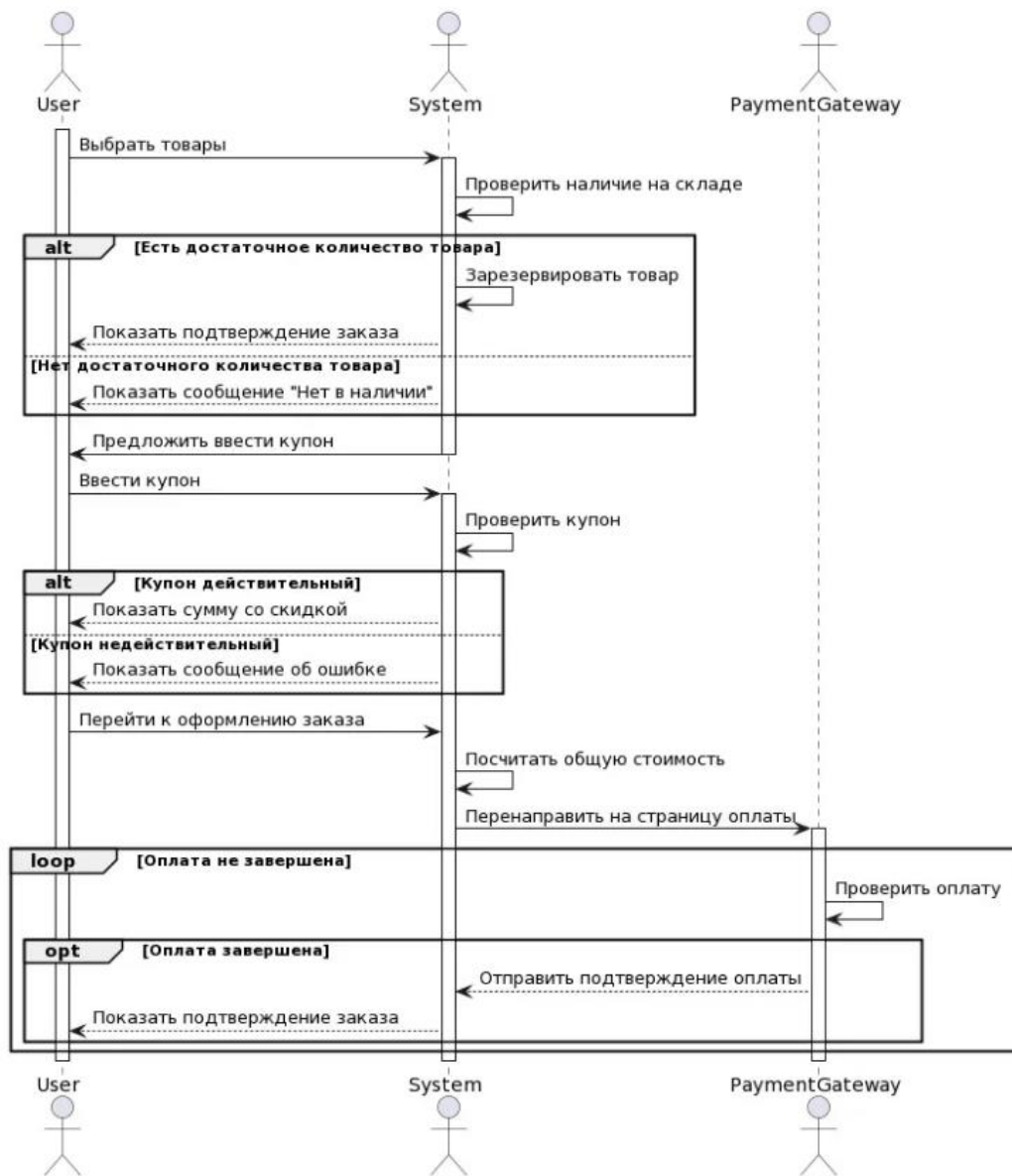


Диаграмма последовательности для варианта использования «Заказать товар со склада»



5.Сохранить.

Задание 2. Построить диаграмму последовательности для процесса оформления заказа на Web-сайте. Описать объекты, сообщения



ПРАКТИЧЕСКАЯ РАБОТА № 15

Тема: Построение диаграммы последовательности

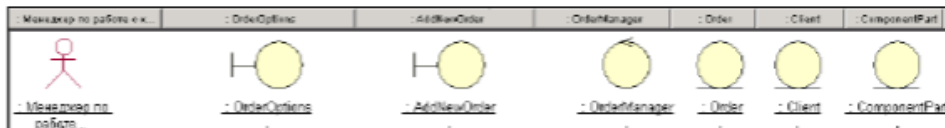
Цель работы: исследование процесса построения диаграмм последовательности в заданной предметной области; научиться строить диаграммы последовательности с применением ПО

Оборудование: ПК, программное обеспечение – MS Word, Smart Draw или UML Diagram, инструкции по выполнению работы.

Содержание работы:

Задание 1. Построить диаграмму последовательности

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 находиться в левом углу, как показано на рисунке

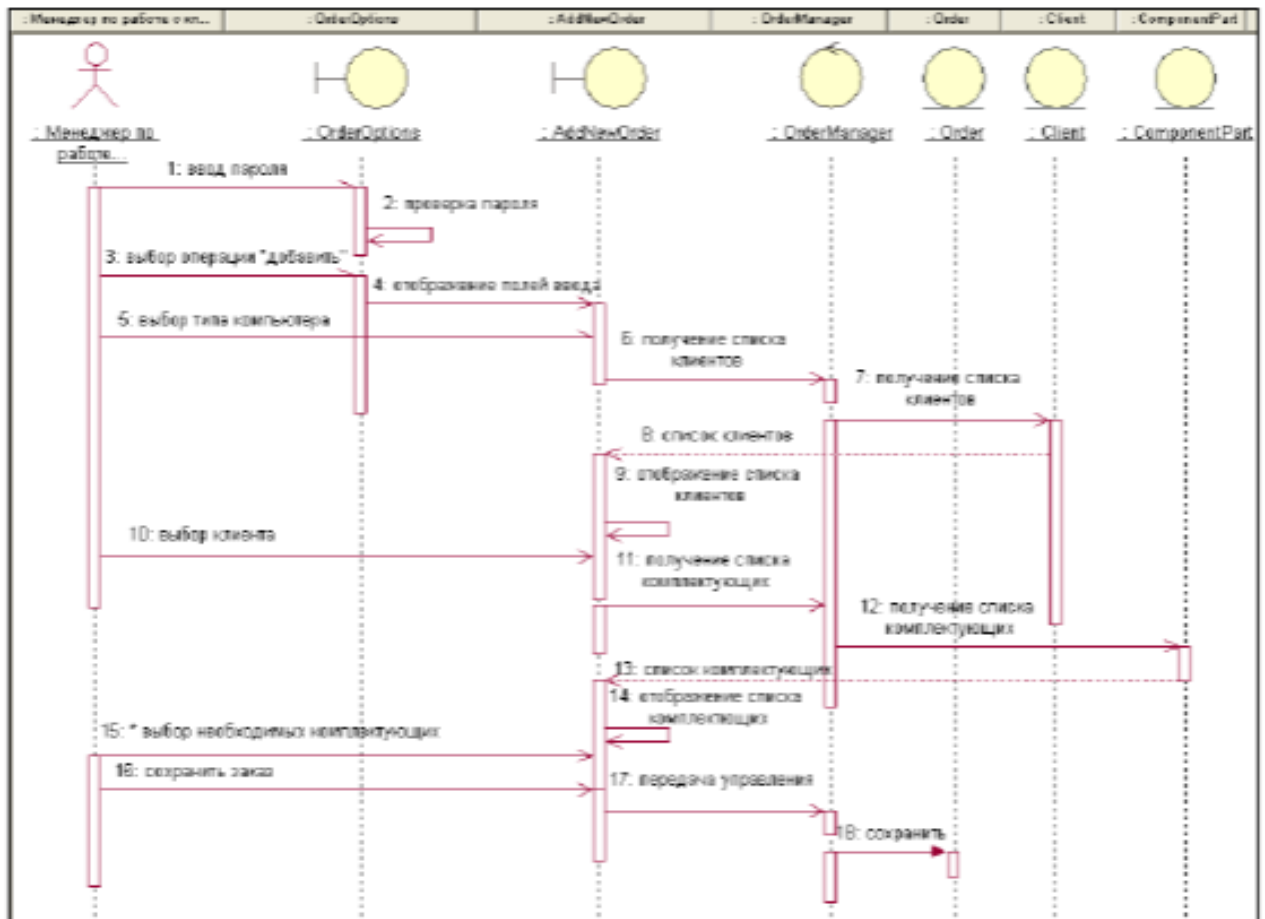


- объект класса OrderOptions (Параметры работы с заказом), отвечающий за выбор возможного действия с заказом в рассматриваемом прецеденте;
- объект класса AddNewOrder (Добавление нового заказа), отвечающий за добавление заказа;
- объект класса OrderManager (Менеджер по работе с заказами), отвечающий за обработку потока событий рассматриваемого прецедента;
- объект класса Order (Заказ);
- объект класса Client (Клиент);
- объект класса ComponentPart (Комплектуемое 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



Задание 2. Построить диаграмму последовательности для каждого варианта использования и для всей системы в целом в соответствии с вариантом.

Варианты:

- 1 «Отдел кадров»;
- 2 «Агентство аренды»;
- 3 «Аптека»;
- 4 «Ателье»;
- 5 «Аэропорт»;
- 6 «Библиотека»;
- 7 «Кинотеатр»;
- 8 «Поликлиника»;
- 9 «Автосалон»;
- 10 «Таксопарк».
- 11 «Издательство»;
- 12 «Прокат велосипедов»;
- 13 «Спортивный клуб».

Отчет по практическому занятию выполняется в формате MS Word, который содержит пошаговое описание процесса построения диаграммы, а также скриншоты результатов согласно заданию.

ПРАКТИЧЕСКАЯ РАБОТА № 16

Тема: Построение диаграммы кооперации

Цель работы: исследование процесса построения диаграмм кооперации в заданной предметной области; научиться строить диаграммы кооперации с применением ПО

Оборудование: ПК, программное обеспечение – MS Word, Smart Draw или UML Diagram или Visual Paradigm Online, инструкции по выполнению работы.

Справочный материал:

Диаграмма кооперации (collaboration diagram) в языке UML — это графическое представление структурных отношений между объектами, участвующими во взаимодействии. Цель диаграммы — описать поведение системы на уровне отдельных объектов, которые обмениваются сообщениями, чтобы достичь определённой цели или реализовать некоторый вариант использования.

Особенность диаграммы — возможность графически представить не только последовательность взаимодействия, но и все структурные отношения между объектами

Структура

Объекты — изображаются в виде прямоугольников, содержат имя объекта, его класс и, возможно, значения атрибутов.

Ассоциации между объектами — указываются в виде различных соединительных линий, можно явно указать имена ассоциации и роли, которые играют объекты в этой ассоциации.

Динамические связи — потоки сообщений — представляются в виде соединительных линий между объектами, над которыми располагается стрелка с указанием направления, имени сообщения и порядкового номера в общей последовательности инициализации сообщений

Правила построения

1. Определить объекты — это могут быть классы, модули, актёры или другие важные сущности.
2. Определить взаимодействия — определить, как объекты работают вместе для выполнения задач или сценариев в системе, выявить сообщения, которыми они обмениваются во время взаимодействий.
3. Добавить сообщения — нарисовать стрелки между жизненными линиями объектов, указать имя сообщения и любые соответствующие параметры или данные, которые отправляются.
4. Указать отношения — если между объектами есть связи или зависимости, показать их с помощью правильных обозначений, например, пунктирных линий или стрелок.
5. Документировать диаграмму — добавить необходимые объяснения или заметки.

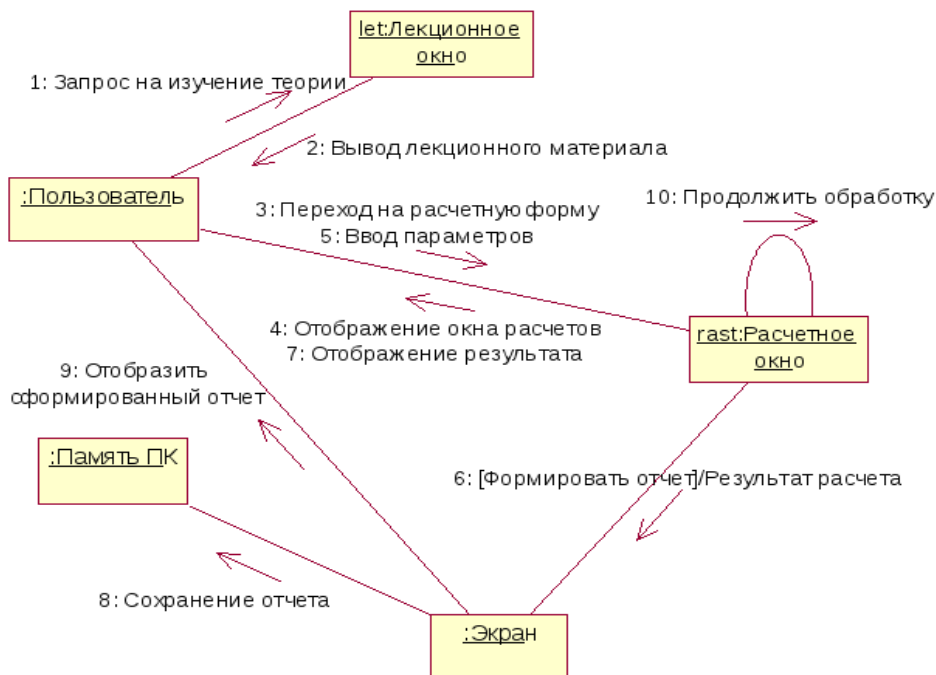
Однако диаграммы кооперации стоит применять только для сложных операций, требующих кооперативного взаимодействия нескольких объектов.

Более простые операции — ограниченные одним или двумя классами — лучше моделировать с помощью диаграмм видов деятельности

Содержание работы:

Задание 1. Программное средство представляет среду для формирования отчетов по лекционному материалу. Пользователь может вводить свою информацию в отчеты, изменять параметры. После оформления отчетов пользователю предоставлена возможность сохранения отчета

1. Запустите сервис для создания диаграмм
2. Выберите Collaboration diagram.
3. Создайте следующую диаграмму.

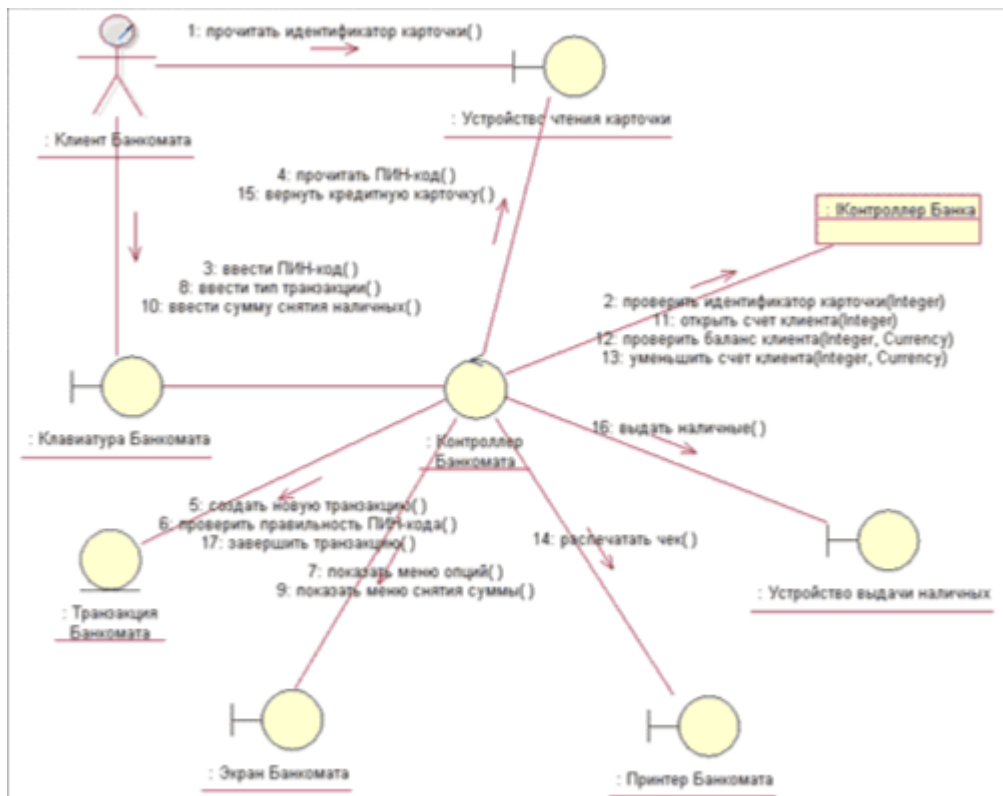


Задание 2. Построить диаграмму кооперации для модели банкомат

1. Добавить *объекты* классов с именами: Контроллер Банкомата, Транзакция Банкомата, Клавиатура Банкомата, Экран Банкомата, Принтер Банкомата, Устройство выдачи наличных и ИИнтерфейс Банка.
2. Добавить *связи*, соединяющие *объекты* классов с именами: Контроллер Банкомата с Устройством чтения карточки, Контроллер Банкомата с Транзакцией Банкомата, Контроллер Банкомата с Клавиатурой Банкомата, Контроллер Банкомата с Экраном Банкомата, Контроллер Банкомата с Принтером Банкомата, Контроллер Банкомата с Устройством выдачи наличных и Контроллер Банкомата с ИИнтерфейсом Банка.
3. Добавить *сообщение*: проверить идентификатор карточки (Integer) , направленное от *объекта* класса Контроллер Банкомата к *объекту* класса ИИнтерфейс Банка.
4. Добавить *сообщение*: ввести ПИН-код(), направленное от *объекта* класса-актера Клиент Банкомата к *объекту* класса Клавиатура Банкомата.

5. Добавить *сообщение*: прочитать ПИН-код(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Устройство чтения карточки.
6. Добавить *сообщение*: создать новую транзакцию(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Транзакция Банкомата.
7. Добавить *сообщение*: проверить правильность ПИН-кода(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Транзакция Банкомата.
8. Добавить *сообщение*: показать меню опций(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Экран Банкомата.
9. Добавить *сообщение*: ввести тип транзакции(), направленное от *объекта* класса-актера Клиент Банкомата к *объекту* класса Клавиатура Банкомата.
10. Добавить *сообщение*: показать меню снятия суммы(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Экран Банкомата.
11. Добавить *сообщение*: ввести сумму снятия наличных(), направленное от *объекта* класса-актера Клиент Банкомата к *объекту* класса Клавиатура Банкомата.
12. Последовательно добавить 3 *сообщения*: открыть счет клиента (Integer) , проверить баланс клиента (Integer, Currency) и уменьшить счет клиента(Integer, Currency), направленные от *объекта* класса Контроллер Банкомата к *объекту* класса Интерфейс Банка.
13. Добавить *сообщение*: распечатать чек(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Принтер Банкомата.
14. Добавить *сообщение*: вернуть кредитную карточку(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Устройство чтения карточки.
15. Добавить *сообщение*: выдать наличные(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Устройство выдачи наличных.
16. Добавить *сообщение*: завершить транзакцию(), направленное от *объекта* класса Контроллер Банкомата к *объекту* класса Транзакция Банкомата.

Получим диаграмму кооперации, описывающая реализацию типичного хода событий варианта использования Снятие наличных по кредитной карточке для проекта системы управления банкоматом



Задание 3. Создать диаграмму, описывающую работу персонала библиотеки по обслуживанию клиентов: библиотекарь получает заказ от клиента, поручает сотруднику найти информацию по нужной клиенту книге, а после получения данных поручает еще одному сотруднику выдать книгу клиенту.

ПРАКТИЧЕСКАЯ РАБОТА № 17

Тема: Построение диаграммы кооперации

Цель работы: исследование процесса построения диаграмм кооперации в заданной предметной области; научиться строить диаграммы кооперации с применением ПО

Оборудование: ПК, программное обеспечение – MS Word, Smart Draw или UML Diagram или Visual Paradigm Online, инструкции по выполнению работы.

Содержание работы:

Задание 1. Создать диаграмму кооперации работы мобильного телефона

Задание 2. Разработать диаграмму кооперации, описывающую процесс управления учебными курсами в учебном центре

Задание 3. Разработать диаграмму кооперации: учитель выбирает в главном меню пункт «добавить ученика»; система показывает учителю окно добавления ученика, содержащее поля для ввода логина и пароля, а также кнопки «далее» и «назад»; учитель вводит желаемый логин и пароль ученика, нажимает кнопку «далее»; система добавляет ученика; учителю открывается главное меню и в течение 5 секунд выводится уведомление о том, что ученик был добавлен успешно.

ПРАКТИЧЕСКАЯ РАБОТА № 18

Тема: Построение диаграммы развертывания

Цель работы: исследование процесса построения диаграмм развертывания в заданной предметной области; научиться строить диаграммы развертывания с применением ПО

Оборудование: ПК, программное обеспечение – MS Word, Smart Draw или UML Diagram или Visual Paradigm Online, инструкции по выполнению работы.

Справочный материал:

Диаграмма развёртывания (англ. Deployment diagram) в Unified Modeling Language (UML) — это тип диаграммы, которая описывает физическое развёртывание компонентов системы на аппаратных компонентах (узлах). Цель — визуализировать аппаратные компоненты, пути связи между ними и расположение программных файлов, которые будут выполняться на этом оборудовании.

Например, чтобы описать веб-сайт, диаграмма развёртывания показывает, какие аппаратные компоненты существуют (например, веб-сервер, сервер базы данных, сервер приложения), какие программные компоненты работают на каждом узле (например, веб-приложение, база данных) и как различные части этого комплекса соединяются друг с другом (например, JDBC, REST, RMI)

Элементы

Узлы — представляют физические или виртуальные ресурсы, такие как серверы или устройства. Бывают двух типов:

Узел контекста выполнения — для выполнения программных компонентов.

Узлы машины — для представления физического оборудования.

Артефакты — представляют программные элементы, такие как файлы или библиотеки.

Соединения — показывают, как объекты развёртываются на узлах.

Пути связи — линии, идущие от одного узла к другому, — обозначают связи.

Правила создания

1. Определить компоненты — перечислить все программные части и аппаратные устройства, которые будут представлены на диаграмме.
2. Понять взаимосвязь — выяснить, как эти части соединяются и работают вместе.
3. Собрать требования — собрать подробную информацию об оборудовании, сетевых настройках и любых специальных правилах развёртывания.
4. Нарисовать узлы и компоненты — начать с рисования аппаратных устройств (узлов) и программных частей (компонентов) с использованием стандартных символов.
5. Подключить узлы и компоненты — использовать линии или стрелки, чтобы показать, как узлы и компоненты связаны.

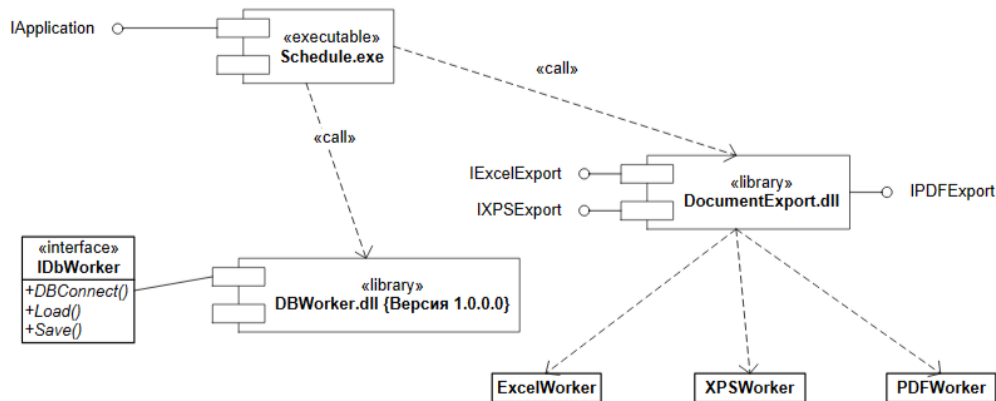
Содержание работы:

Задание 1. Постройте диаграмму размещения для системы генерации. Сделайте краткое описание диаграммы

а. Начните с идентификации всех аппаратных, механических и других типов устройств, которые необходимы для выполнения системой всех своих функций.

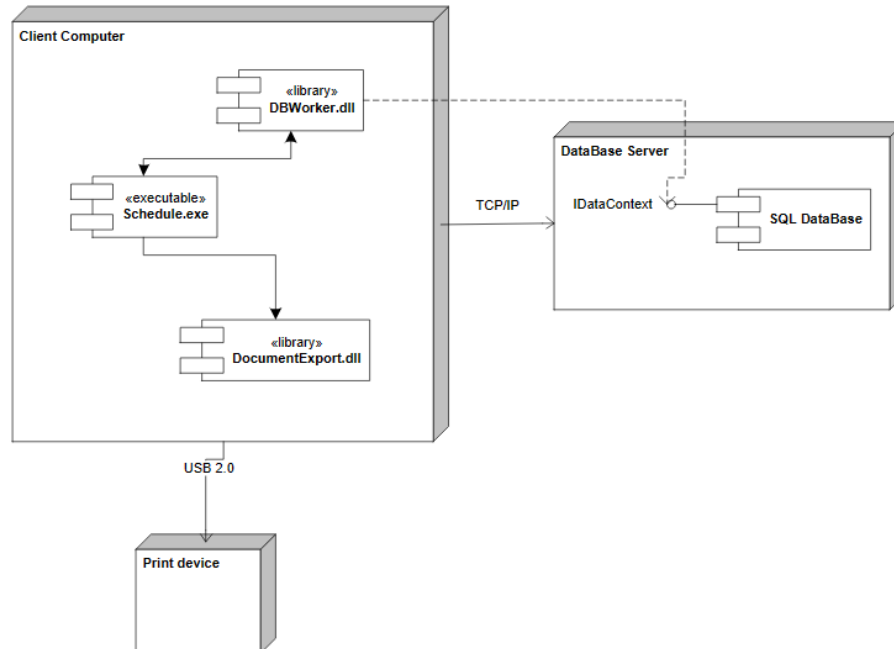
б. Дальнейшее построение диаграммы развертывания связано с размещением всех исполняемых компонентов диаграммы по узлам системы.

1. Для начала нужно определить компоненты системы. Возьмем пример диаграммы компонентов для системы генерации отчетов



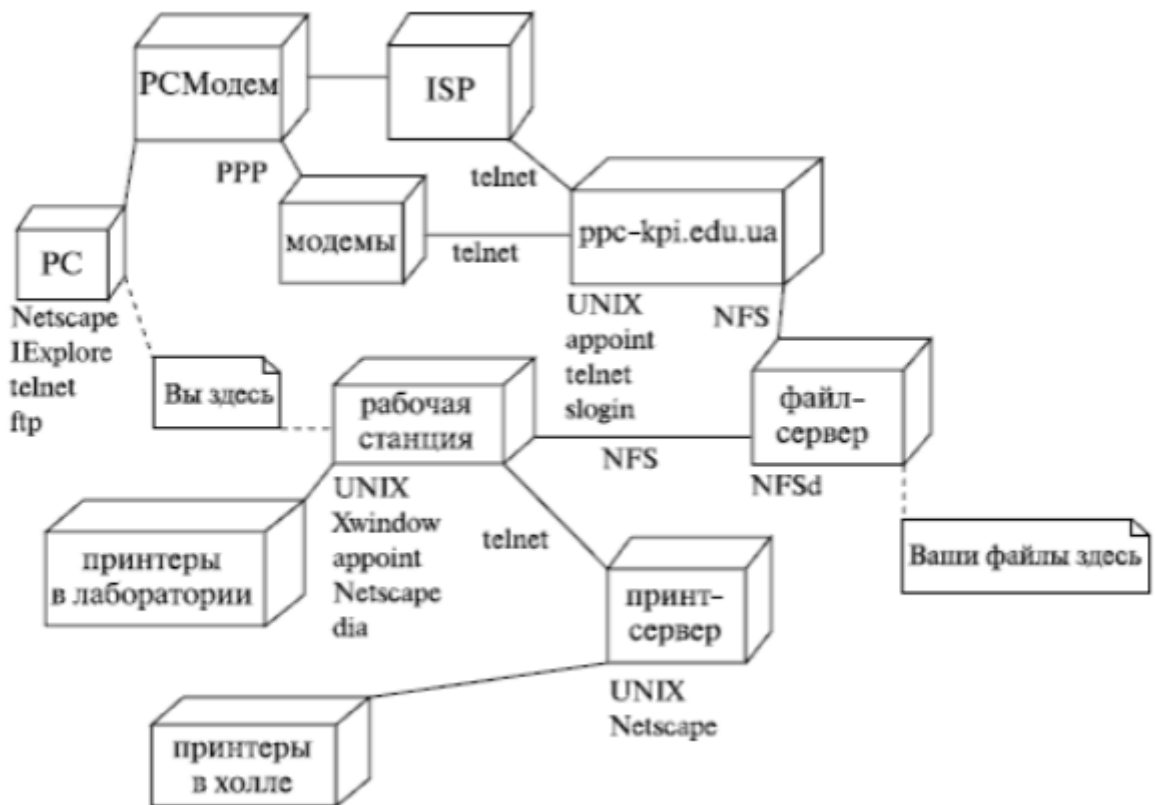
2. Запустите среду разработки диаграмм UML.

3. Диаграмма для данного примера будет иметь следующий вид:



Задание 2. Это инфраструктура некоего учебного заведения, включающая шлюз, файл-сервер, принт-сервер, принтеры в лабораториях и холле и т. д. Пользователь (вероятно, студент или преподаватель) может получить доступ к этим ресурсам либо со своей домашней машины, либо с

рабочих станций, находящихся в лабораториях вуза. Обратите внимание на подписи под линиями, показывающими линии передачи информации, например, видно, что рабочая станция получает доступ к файлам, хранящимся на файл-сервере, посредством NFS. Также хорошая идея - рядом с обозначением узла перечислить программное обеспечение, установленное на данном узле, как это сделано, например, для рабочей станции.



Задание 3. Разработать диаграмму развертывания для информационной системы Банкомат. Сделайте краткое описание диаграммы

Задание 4. Разработать диаграмму развертывания для информационной системы Туроператор. Сделайте краткое описание диаграммы

ПРАКТИЧЕСКАЯ РАБОТА № 19

Тема: Построение диаграммы развертывания

Цель работы: исследование процесса построения диаграмм развертывания в заданной предметной области; научиться строить диаграммы развертывания с применением ПО

Оборудование: ПК, программное обеспечение – MS Word, Smart Draw или UML Diagram или Visual Paradigm Online, инструкции по выполнению работы.

Содержание работы:

Задание 1. Разработать диаграмму развертывания для системы онлайн-магазина — узлы могут включать «Веб-сервер», «Сервер базы данных» и «Клиентское устройство», а компоненты — «Веб-приложение», «БД» и «Интерфейс пользователя». Сделайте краткое описание диаграммы

Задание 2. Разработать диаграмму развертывания для архитектуры «клиент-сервер», иллюстрирующую соединение между клиентскими приложениями и серверными узлами. Сделайте краткое описание диаграммы.

Задание 3. Построить диаграмму развертывания для информационной системы в соответствии с вариантом.

Варианты:

- 1 «Отдел кадров»;
- 2 «Агентство аренды»;
- 3 «Аптека»;
- 4 «Ателье»;
- 5 «Аэропорт»;
- 6 «Библиотека»;
- 7 «Кинотеатр»;
- 8 «Поликлиника»;
- 9 «Автосалон»;
- 10 «Таксопарк».
- 11 «Издательство»;
- 12 «Прокат велосипедов»;
- 13 «Спортивный клуб».

Отчет по практическому занятию выполняется в формате MS Word, который содержит пошаговое описание процесса построения диаграммы, а также скриншоты результатов согласно заданию.

ПРАКТИЧЕСКАЯ РАБОТА № 20

Тема: Построение диаграммы деятельности

Цель работы: исследование процесса построения диаграмм деятельности в заданной предметной области; научиться строить диаграммы деятельности с применением ПО

Оборудование: ПК, программное обеспечение – MS Word, Smart Draw или UML Diagram или Visual Paradigm Online, инструкции по выполнению работы.

Справочный материал:

Диаграмма деятельности (Activity diagram) (также диаграмма активности, диаграмма видов деятельности) — тип диаграммы в языке UML (Unified Modeling Language). Она показывает, как поток управления переходит от одной деятельности к другой, при этом внимание фиксируется на результате деятельности (например, изменении состояния системы или возвращении некоторого значения).

Отличие диаграммы деятельности от традиционной блок-схемы — более высокий уровень абстракции и возможность представления управления параллельными потоками наряду с последовательным управлением.

Элементы

Диаграмма деятельности состоит из ограниченного количества фигур, соединённых стрелками.

Прямоугольники с закруглениями — действия (операции).

Ромбы — решения, определяют правило ветвления и различные варианты дальнейшего развития сценария.

Чёрный круг — начало процесса (начальный узел).

Чёрный круг с обводкой — окончание процесса (конечный узел).

Стрелки идут от начала к концу процесса и показывают потоки управления или потоки объектов (данных).

Для реализации параллельных потоков используются точки разделения (fork node) и точки слияния (join node):

Точка разделения — из неё выходят два и более потока, выполняющихся параллельно.

Точка слияния — синхронизирует потоки: каждый из них ждёт достижения этой точки остальными потоками, после чего продолжается последовательное исполнение.

Правила построения

1. Каждая диаграмма деятельности должна иметь единственное начальное и единственное конечное состояния. На практике иногда можно видеть несколько конечных состояний на одной диаграмме, но это одно и то же состояние, изображённое несколько раз для лучшей читабельности диаграммы.

2. Диаграмму принято располагать так, чтобы действия следовали сверху вниз: начальное состояние изображается в верхней части диаграммы, а конечное — в её нижней части.

3.Используются только нетриггерные переходы — такие, которые срабатывают сразу после завершения деятельности или выполнения соответствующего действия.

Содержание работы:

Задание 1. Построить диаграмму деятельности для бизнес-процесса предприятия.

Чтобы построить диаграмму деятельности, для начала стоит разобраться, какие действия происходят от оформления заказа и передачи его на сборку до выдачи собранного компьютера клиенту.

1.После оформления заказа на сборку компьютера, менеджер по работе с клиентами передает заказ инженеру по сборке.

2. Инженер по сборке, перед началом работы, запрашивает все необходимые компоненты со склада.

3.Завскладом проверяет наличие комплектующих, собирает и передает инженеру по сборке.

4.Инженер по сборке производит сборку компьютера и передает его инженеру по тестированию.

5.Инженер по тестированию тестирует компьютер:

- тестирование пройдено –компьютер передается на склад, завскладом оформляет и хранит его до выдачи;

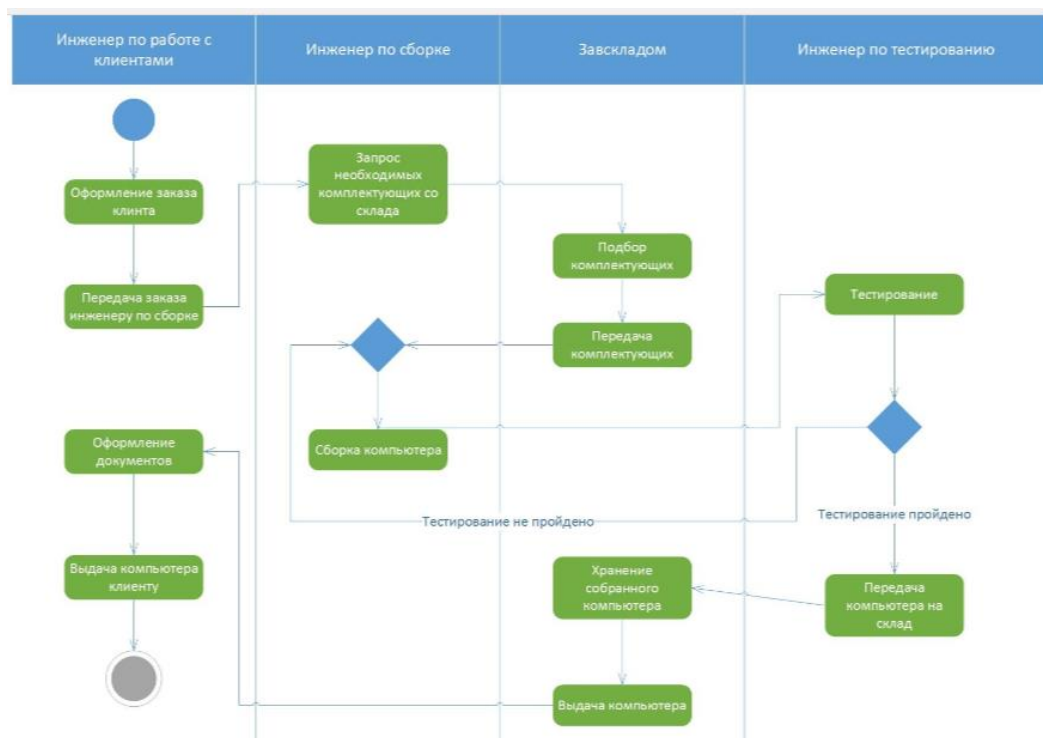
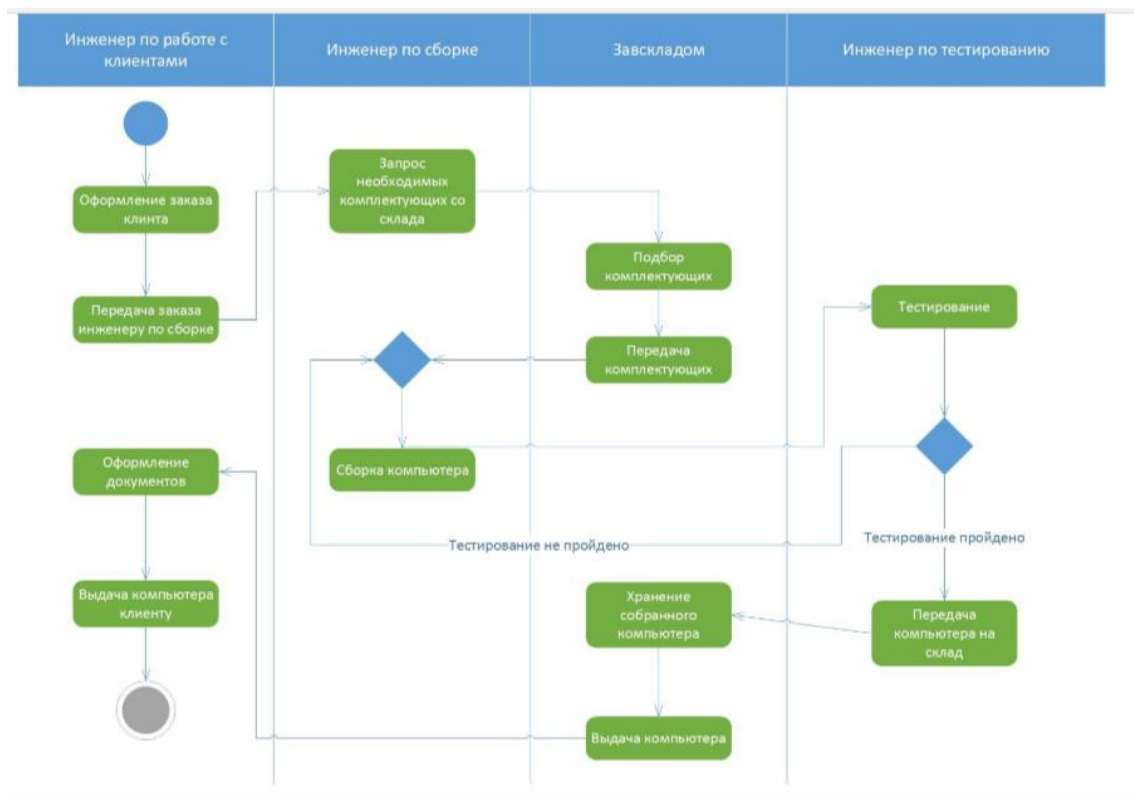
- тестирование не пройдено–компьютер передается на повторную сборку и устранение неполадок.

6.Завскладом, по запросу менеджера по работе с заказом, передает компьютер на выдачу.

7.Менеджер по работе с клиентами оформляет необходимую документацию и выдает клиенту выполненный заказ.

Теперь нужно построить визуальное разделение для каждого из действующих лиц

Инженер по работе с клиентами	Инженер по сборке	Завскладом	Инженер по тестированию



Задание 2. Построить диаграмму деятельности для системы. Моделируется ситуация, возникающая в супермаркетах при оплате товаров. Как правило, заплатить за покупки можно либо наличными, либо по кредитной карточке. Если покупателем выбран вариант оплаты по кредитной карточке, то проверяется сумма баланса предъявленной к оплате кредитной карточки. При этом оплата происходит только в том случае, если общая стоимость приобретаемых товаров не превышает суммы баланса этой

карточки. В противном случае оплаты не происходит, и товар остается у продавца.

Задание 3. Построить диаграмму деятельности для системы. Регистрация пассажиров в аэропорту. Первоначально выполняется деятельность по проверке билета: если билет не действителен, он возвращается пассажиру; если билет действителен, то пассажиру выдается посадочный талон, в дополнение проверяется гражданство и наличие багажа у пассажира. Если есть багаж, то его проверка может быть выполнена параллельно, после чего пассажиру выдается талон на багаж. Если пассажир является иностранным гражданином, то дополнительно проверяется наличие у него визы. Если виза действительна, то проверка завершается успешно, и пассажир может проследовать на посадку. Если же виза не действительна, то для этого пассажира посадка оказывается невозможной, и ему не выдается посадочный талон и талон на багаж. Происходит прекращение всех выполняемых сотрудниками аэропорта действий.

ПРАКТИЧЕСКАЯ РАБОТА № 21

Тема: Построение диаграммы деятельности

Цель работы: исследование процесса построения диаграмм деятельности в заданной предметной области; научиться строить диаграммы деятельности с применением ПО

Оборудование: ПК, программное обеспечение – MS Word, Smart Draw или UML Diagram или Visual Paradigm Online, инструкции по выполнению работы.

Содержание работы:

Задание 1. Построить диаграмму деятельности для системы. Программное средство представляет собой базу данных «Автоматизация процесса составления расписания в учебном заведении». Программное средство обеспечивает корректировку данных, а именно в БД «Группы» может изменяться перечень предметов в соответствии с курсом группы и отделением; в БД «Предметы» могут изменяться номера аудиторий и фамилии преподавателей; осуществляет поиск по ФИО преподавателя, номеру аудитории, названию предмета и номеру группы. Программное средство составляет расписание работы для конкретного преподавателя на неделю; для группы на неделю; для группы по конкретному предмету, а также отчет о загрузке аудиторий на каждый день.

Задание 2. Построить диаграмму деятельности для системы. Торговая компания, обслуживающая клиентов в форме заказов. Подразделениями компании обычно являются отдел приема и оформления заказов, отдел продаж и склад. Этим подразделениям будут соответствовать три дорожки на диаграмме деятельности, каждая из которых специфицирует зону ответственности подразделения. В этом случае диаграмма деятельности включает в себе не только информацию о последовательности выполнения рабочих действий, но и о том, какое подразделение торговой компании должно выполнять то или иное действие.

После принятия заказа от клиента отделом приема и оформления заказов осуществляется распараллеливание деятельности на два потока (переход-разделение). Первый из них остается в этом же отделе и связан с получением оплаты от клиента за заказанный товар. Второй инициирует выполнение действия по регистрации заказа в отделе продаж (модель товара, размеры, цвет, год выпуска и пр.). Однако выдача товара со склада начинается только после того, как будет получена от клиента оплата за товар (переход-слияние). Затем выполняется подготовка товара к отправке и его отправка клиенту в отделе продаж. После завершения этих деятельностей заказ закрывается в отделе приема и оформления заказов.

Задание 3. Построить диаграмму деятельности в соответствии с вариантом.

Отчет по практическому занятию выполняется в формате MS Word, который содержит пошаговое описание процесса построения диаграммы, а также скриншоты результатов согласно заданию.

Варианты:

- 1 «Отдел кадров»;
- 2 «Агентство аренды»;
- 3 «Аптека»;
- 4 «Ателье»;
- 5 «Аэропорт»;
- 6 «Библиотека»;
- 7 «Кинотеатр»;
- 8 «Поликлиника»;
- 9 «Автосалон»;
- 10 «Таксопарк».
- 11 «Издательство»;
- 12 «Прокат велосипедов»;
- 13 «Спортивный клуб».

ПРАКТИЧЕСКАЯ РАБОТА № 22

Тема: Построение диаграммы состояний

Цель работы: исследование процесса построения диаграмм состояний в заданной предметной области; научиться строить диаграммы состояний с применением ПО

Оборудование: ПК, программное обеспечение – MS Word, Smart Draw или UML Diagram или Visual Paradigm Online, инструкции по выполнению работы.

Справочный материал:

Диаграмма состояний (state diagram, statechart diagram) — это визуальное представление поведения объекта при определённых условиях. Она показывает, какие состояния может принимать объект и как он переходит от одного к другому при разных событиях.

Состояние — ситуация в жизненном цикле объекта, во время которой он удовлетворяет некоторому условию, выполняет определённую деятельность или ожидает события.

Элементы

Диаграмма состояний изображается в виде графа с вершинами и дугами (ребрами).

Состояния — обозначаются прямоугольниками с закруглёнными углами.

Переходы — связи между двумя состояниями, показывают, что объект, находящийся в первом состоянии, должен выполнить некоторые действия и перейти во второе, как только произойдёт определённое событие и будут выполнены определённые условия.

События — триггер, внешний сигнал, который вызывает переход состояния.

Действия — реакция, операция, выполняемая при переходе состояния или в состоянии.

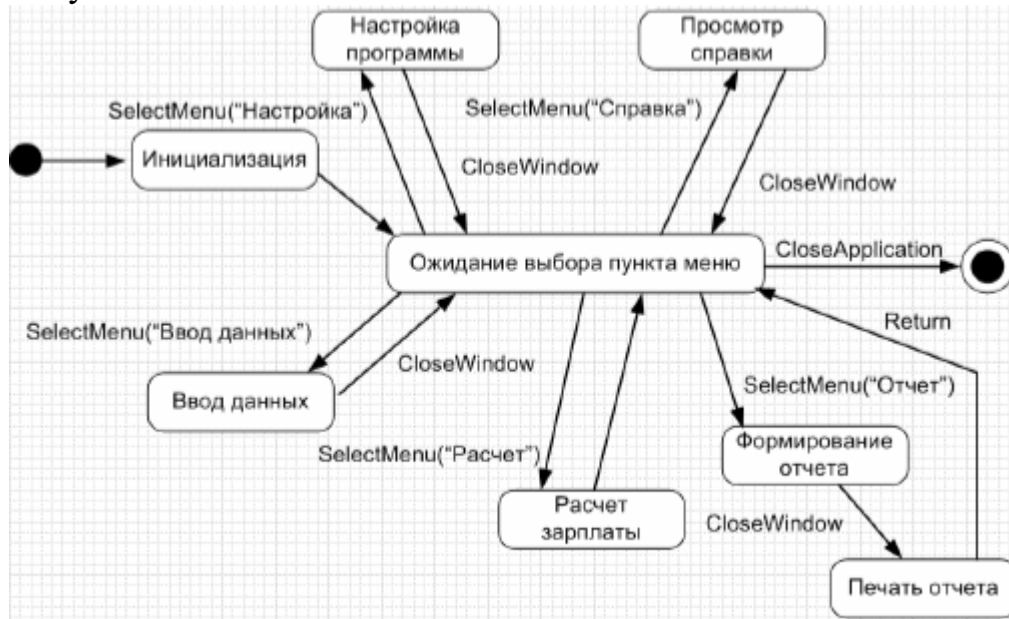
Также на диаграмме могут быть **псевдосостояния**: начальное (обозначается сплошным кружком) и конечное (обозначается кружком, обведённым окружностью).

Правила построения

1. Диаграмма должна начинаться знаком начального состояния и заканчиваться знаком конечного. Начальное состояние указывается только один раз, а конечных может быть несколько, чтобы минимизировать пересечения переходов.
2. Для облегчения восприятия диаграммы рекомендуется использовать декомпозицию со скрытием составных состояний.
3. Диаграмма не должна содержать изолированных состояний и переходов. Переходы и их спецификация должны быть заданы таким образом, чтобы на графе каждое состояние было потенциально достижимо из начального и из любого состояния было потенциально достижимо конечное.

Содержание работы:

Задание 1. Построить диаграмму состояний, моделирующую реакцию системы на выбор пользователем пунктов меню. Описать построенную диаграмму.



Задание 2. Построить диаграмму состояний для системы. Программное средство представляет собой базу данных и обеспечивает корректировку данных, а именно в БД «Группы» может изменяться перечень предметов в соответствии с курсом группы и отделением; в БД «Предметы» могут изменяться номера аудиторий и фамилии преподавателей; осуществляет поиск по ФИО преподавателя, номеру аудитории, названию предмета и номеру группы. Программное средство составляет расписание работы для конкретного преподавателя на неделю; для группы на неделю; для группы по конкретному предмету, а также отчет о загрузке аудиторий на каждый день. Описать построенную диаграмму.

Задание 3. Построить диаграмму состояний для аутентификации клиента для доступа к ресурсам моделируемой информационной системы. Список внутренних действий в данном состоянии может включать следующие действия. Первое действие – входное, которое выполняется при входе в это состояние и связано с получением строки символов, соответствующих паролю клиента. Далее выполняется деятельность по проверке введенного клиентом пароля. При успешном завершении этой проверки выполняется действие на выходе, которое отображает меню доступных для клиента опций. Описать построенную диаграмму.

ПРАКТИЧЕСКАЯ РАБОТА № 23

Тема: Построение диаграммы состояний

Цель работы: исследование процесса построения диаграмм состояний в заданной предметной области; научиться строить диаграммы состояний с применением ПО

Оборудование: ПК, программное обеспечение – MS Word, Smart Draw или UML Diagram или Visual Paradigm Online, инструкции по выполнению работы.

Содержание работы:

Задание 1. Построить диаграммы состояний в соответствии с вариантом (необходимо выбрать по 3 варианта)

Отчет по практическому занятию выполняется в формате MS Word, который содержит пошаговое описание процесса построения диаграммы, а также скриншоты результатов согласно заданию.

Варианты:

- 1 «Отдел кадров»;
- 2 «Агентство аренды»;
- 3 «Аптека»;
- 4 «Ателье»;
- 5 «Аэропорт»;
- 6 «Библиотека»;
- 7 «Кинотеатр»;
- 8 «Поликлиника»;
- 9 «Автосалон»;
- 10 «Таксопарк».
- 11 «Издательство»;
- 12 «Прокат велосипедов»;
- 13 «Спортивный клуб».

ПРАКТИЧЕСКАЯ РАБОТА № 24

Тема: Построение диаграммы классов

Цель работы: исследование процесса построения диаграмм классов в заданной предметной области; научиться строить диаграммы классов с применением ПО

Оборудование: ПК, программное обеспечение – MS Word, Smart Draw или UML Diagram или Visual Paradigm Online, инструкции по выполнению работы.

Справочный материал:

Диаграмма классов (англ. class diagram) — структурная диаграмма языка моделирования UML, которая демонстрирует статическую структуру системы. Она показывает классы, их атрибуты, методы и взаимосвязи (отношения) между ними.

Цель создания диаграммы — графическое представление декларативных элементов системы (классов, типов и т. п.). На диаграмме не указывается информация о временных аспектах функционирования системы.

Элементы

Классы — изображаются в виде прямоугольника, разделённого на три секции:

Верхняя секция — имя класса (выравнивается по центру, пишется полужирным шрифтом, начинается с заглавной буквы). Если класс абстрактный, его имя пишется полужирным курсивом.

Средняя секция — атрибуты (поля) класса, выровнены по левому краю, начинаются с маленькой буквы.

Нижняя секция — методы класса, также выровнены по левому краю, пишутся с маленькой буквы.

Взаимосвязи — линии, соединяющие классы, иллюстрируют ассоциации, показывают отношения, например, «один-к-одному» или «один-ко-многим». Некоторые типы связей:

Обобщение (наследование) — связывает подкласс с его суперклассом.

Ассоциация — показывает статическую взаимосвязь между двумя объектами.

Зависимость — показывает зависимость одного класса от другого, изменение в одном классе приведёт к изменению в другом классе

Виды

Для концептуального моделирования — описывают структуру приложения.

Для детального моделирования — помогают перевести модели в код программирования.

Для моделирования данных — диаграммы классов могут использоваться для описания структуры данных.

Правила создания

1. Начинать с ключевых классов, затем добавлять детали.
2. Использовать абстрактные классы для общих характеристик.

3.Соблюдать уровни детализации: концептуальный уровень (без типов и видимости), уровень спецификации (с типами, без реализации), уровень реализации (полная детализация).

4.Группировать связанные классы с помощью пакетов.

5.Подписывать важные ассоциации для ясности отношений.

Задание 1. Создание физической диаграммы:

1 Запустите программу для создания UML-диаграмм

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

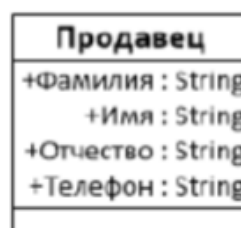
Для чего:

– 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 (один из возможных вариантов представлен на рис. 1).



Рис.1. Формирование статической структуры объектов диаграммы

– 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.

Товар
-Тип : String
-Марка : String
-Атрибут : String

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

Заказ
-Количество : Integer
-Цена : Double
-Статус : String
+Сформировать_Заказ()

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

Заказ_Оплата
-Дата_Получения : Date
-Проплачен : Boolean
-Номер : String
-Цена : Double
+Отправить()
+Закрыть()

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

Клиент
-Имя : String
-Адрес : String
+кредитный_рейтинг() : String

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.

Корпоративный_Клиент
-Контактное_Имя : String
-Кредитный_Рейтинг : Integer
-Кредитный_Лимит : Double
+Сделать()
+Напоминание()
+Счет_За_Месяц() : Integer

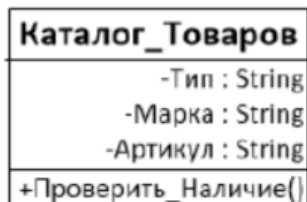
Для класса Частный_Клиент в качестве атрибутов могут выступать данные: номер кредитной карты.

Частный_Клиент
-Номер_Кредитной_Карты : String

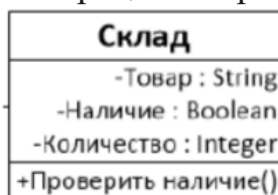
Для класса Вариант_Оплаты в качестве атрибутов могут выступать данные: тип оплаты, а в качестве операций – выбор варианта оплаты.



Для класса Каталог_Товаров в качестве атрибутов могут выступать данные: тип, марка, артикул, а в качестве операций – проверить наличие.



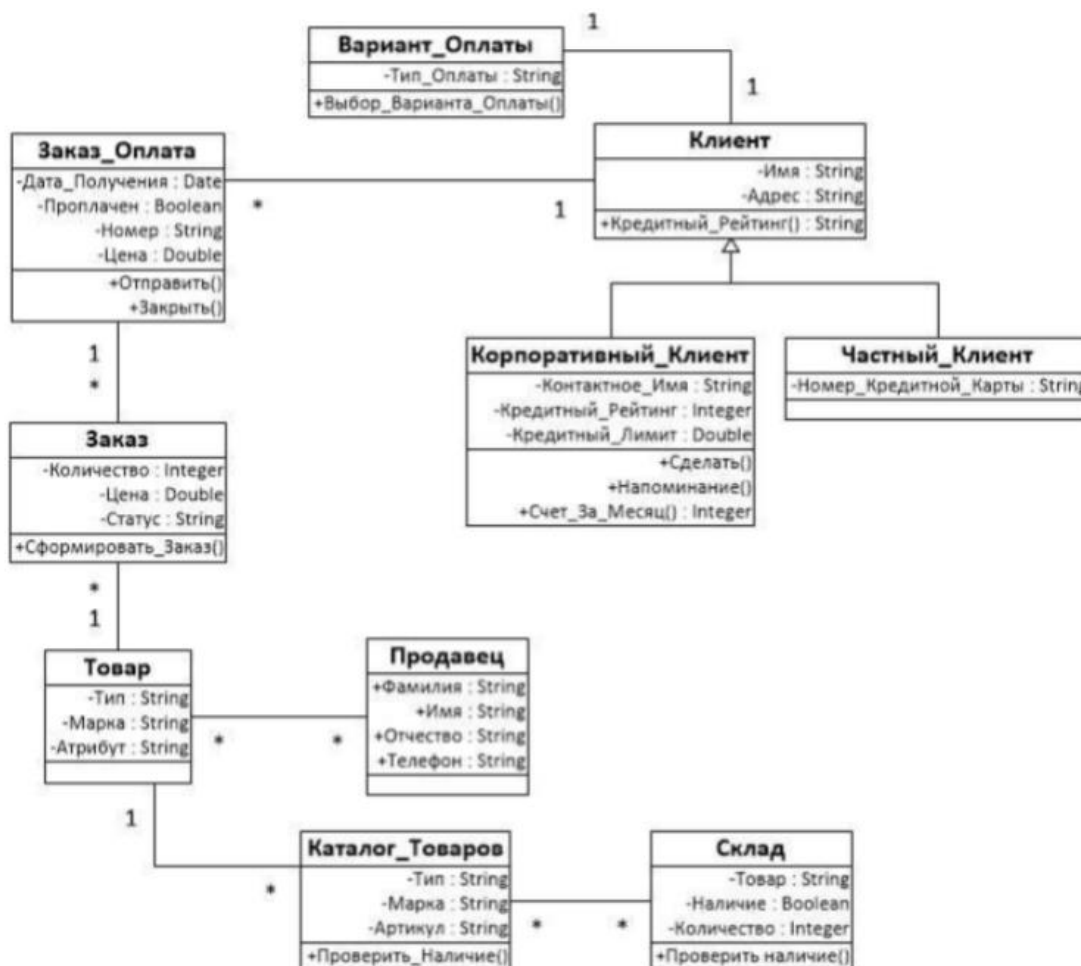
Для класса Склад в качестве атрибутов могут выступать данные: товар, наличие, количество, а в качестве операций – Проверить наличие.



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

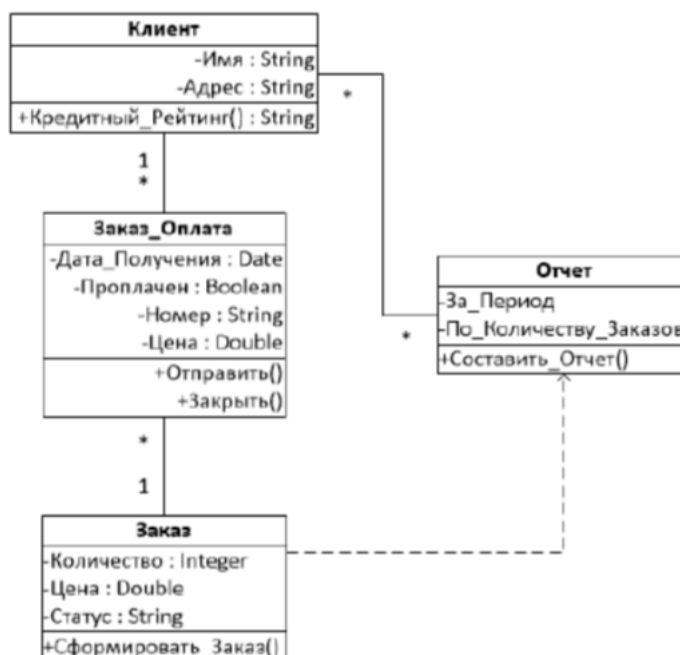
В качестве примера на рис. 2 показан набор классов, описывающих реализацию системы продаж товаров по каталогу. Акцент сделан на классе Клиент, с которым связан класс Заказ_Оплата посредством двусторонней ассоциации «один-ко-многим», Вариант_Оплаты – двусторонней ассоциацией «один-к-одному» и классы Корпоративный_Клиент 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-ко-многим» с классом Склад.



Фрагмент диаграммы классов, описывающей реализацию систем продаж товаров по каталогу

Задание 2. Создайте новую диаграмму классов учета клиентов. 1 Идентифицируйте классы учета клиентов, осуществляющих заказы и создайте диаграмму классов с указанием их имен, атрибутов, операций, например, так как показано на рисунке. Опишите созданную диаграмму.



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

ПРАКТИЧЕСКАЯ РАБОТА № 25

Тема: Построение диаграммы классов

Цель работы: исследование процесса построения диаграмм классов в заданной предметной области; научиться строить диаграммы классов с применением ПО

Оборудование: ПК, программное обеспечение – MS Word, Smart Draw или UML Diagram или Visual Paradigm Online, инструкции по выполнению работы.

Содержание работы:

Задание 1. Создайте диаграмму классов Информационной системы туристического агентства. Отчет по практическому занятию выполняется в формате MS Word, который содержит пошаговое описание процесса построения диаграммы, а также скриншоты результатов согласно заданию.

Задание 2. Выполните построение диаграммы классов компании по прокату авто. Отчет по практическому занятию выполняется в формате MS Word, который содержит пошаговое описание процесса построения диаграммы, а также скриншоты результатов согласно заданию.

Задание 3. Построить диаграмму классов в соответствии с вариантом.

Отчет по практическому занятию выполняется в формате MS Word, который содержит пошаговое описание процесса построения диаграммы, а также скриншоты результатов согласно заданию.

Варианты:

- 1 «Отдел кадров»;
- 2 «Агентство аренды»;
- 3 «Аптека»;
- 4 «Ателье»;
- 5 «Аэропорт»;
- 6 «Библиотека»;
- 7 «Кинотеатр»;
- 8 «Поликлиника»;
- 9 «Автосалон»;
- 10 «Таксопарк».
- 11 «Издательство»;
- 12 «Прокат велосипедов»;
- 13 «Спортивный клуб».

ПРАКТИЧЕСКАЯ РАБОТА № 26

Тема: Построение диаграммы компонентов

Цель работы: исследование процесса построения диаграмм компонентов в заданной предметной области; научиться строить диаграммы компонентов и с применением ПО

Оборудование: ПК, программное обеспечение – MS Word, Smart Draw или UML Diagram или Visual Paradigm Online, инструкции по выполнению работы.

Справочный материал:

Диаграмма компонентов (англ. Component diagram) — элемент языка моделирования UML, статическая структурная диаграмма. Она показывает разбиение программной системы на структурные компоненты и связи (зависимости) между ними.

В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п.

Элементы

Компонент — отдельная часть системы, которая выполняет определённую функцию или имеет определённую роль. Может быть программным модулем, классом, библиотекой, сервисом, физическим устройством и т. д. Обычно отображается в виде прямоугольника с именем или обозначением.

Предоставляемый интерфейс — указывает на то, какие возможности или функции предоставляет конкретный компонент для взаимодействия с другими компонентами или системами. Графически отображается как окружность со сплошной линией.

Требуемый интерфейс — указывает на интерфейсы, которые компонент ожидает от других компонентов. Определяет, какие операции, методы или данные должны быть предоставлены другими компонентами, чтобы компонент мог взаимодействовать с ними. Графически отображается как полукруг со сплошной линией.

Порт — представляет точку входа или выхода компонента, через которую он взаимодействует с другими компонентами. Указывает на интерфейс компонента, через который он предоставляет или получает данные, вызывает методы или осуществляет обмен сообщениями.

Правила построения

- 1.Использовать уже имеющиеся в языке UML компоненты и стереотипы. Для большинства типовых проектов этого набора элементов может быть достаточно для представления компонентов и зависимостей между ними.
- 2.Если проект содержит некоторые физические элементы, описание которых отсутствует в языке UML, использовать механизм расширения: дополнительные стереотипы для отдельных нетиповых компонентов или помеченные значения для уточнения их характеристик.
- 3.Зависимость изображается стрелкой от интерфейса или порта клиента к импортируемому интерфейсу. Стрелка указывает на направление

зависимости: показывает, что изменения в исходном компоненте влияют на зависимый компонент.

Содержание работы:

Задание 1. Для заданной предметной области «Туристическое агентство» построить диаграмму компонентов.

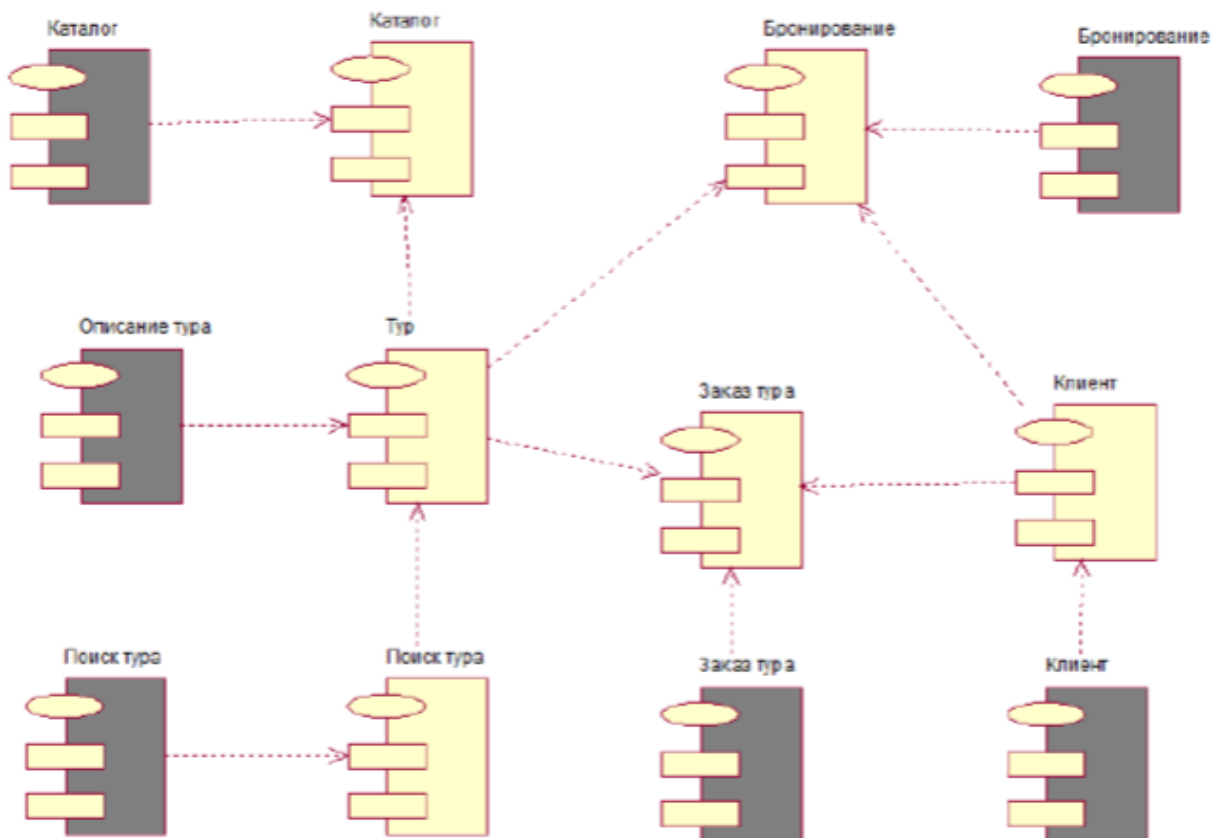
Диаграмма компонентов позволяет определить состав программных компонентов, в роли которых могут выступать исходный, бинарный и исполняемый коды, а также установить зависимости между ними.

Компонент – это физическая часть системы. Компоненты представляют собой файлы с исходным кодом классов, библиотеки, исполняемые модули и т. п., которые должны обладать согласованным набором интерфейсов.

Компоненты могут иметь следующие стандартные стереотипы:

- 1) «file» – любой файл, кроме таблицы;
 - «executable» – программа (исполняемый файл);
 - «library» – статическая или динамическая библиотека;
 - «source» – файл с исходным текстом программы;
 - «document» – остальные файлы (например, файл справки);
- 2) «table» – таблица базы данных.

Диаграмма компонентов для туристического агентства представлена на рисунке



Задание 2. Для заданной предметной области «Розничный магазин» построить диаграмму компонентов. Описать построенную диаграмму.

Розничный магазин занимается продажей продуктов. Основные процессы, на которых основывается деятельность магазина: поступление

товаров, возврат товаров поставщику, реализация товаров, инвентаризация. Каждая партия товаров сопровождается накладной, счет-фактурой и сертификатом качества. Оператор сверяет количество товара с документами, принимает и отправляет товары на склад. Администратор рассчитывает розничные цены для поступившего товара, а также формирует заявку на товар, который необходимо вынести в торговый зал. В случае несоответствия товара требованиям магазина товаровед принимает решение о его возврате по возвратной накладной. В конце дня старший кассир закрывает смену на каждой кассе, формирует отчеты. В процессе инвентаризации осуществляется сверка остатков по базе данных с реальными остатками на складе и в магазине.

Задание 3. Для заданной предметной области «Гостиница» построить диаграмму компонентов. Описать построенную диаграмму.

Основные процессы, на которых основывается деятельность гостиницы: бронирование мест, прием, регистрация и размещение гостей, предоставление услуг проживания и питания, предоставление дополнительных услуг проживающим, окончательный расчет и оформление выезда. При регистрации и оформлении выезда работники гостиницы осуществляют расчет за проживание в гостинице. При выписке проверяется счет гостя, уточняются все его расходы за время проживания, принимается оплата. Дежурный администратор, начиная работу, должен просмотреть журнал с записями предыдущей смены. Перед началом работы необходимо также просмотреть информацию о наличии свободных мест и заявки на текущие сутки. Дежурный администратор контролирует своевременность оплаты услуг, получает плату за проживание при наличном расчете и составляет кассовые отчеты для бухгалтерии.

ПРАКТИЧЕСКАЯ РАБОТА № 27

Тема: Построение диаграммы компонентов

Цель работы: исследование процесса построения диаграмм компонентов в заданной предметной области; научиться строить диаграммы компонентов и с применением ПО

Оборудование: ПК, программное обеспечение – MS Word, Smart Draw или UML Diagram или Visual Paradigm Online, инструкции по выполнению работы.

Содержание работы:

Задание 1. Для заданной предметной области «Стоматологическая клиника» построить диаграмму компонентов. Описать построенную диаграмму.

Клиника оказывает медицинские услуги: лечение и протезирование зубов. Клиент подает заявку на посещение стоматолога в регистратуру. Поступившая заявка записывается в журнал. Журнал ведется в бумажном виде. Во время оформления заявки с клиентом оговариваются условия дальнейшего обследования, время приема и стоимость услуг. После того как условия согласованы, данные клиента заносятся в базу данных и заключается договор. Во время посещения клиенту оформляется медицинская карта, в которую записываются личные данные. В этой карте фиксируются все дальнейшие приемы. После того как клиенту оказаны услуги, лечащий врач заносит информацию об оказанных услугах в медицинскую карту и выдает ее клиенту. На основании записи в медицинской карте бухгалтер в соответствии с прайс-листом выписывает квитанцию, по которой клиент должен будет оплатить услуги в кассе.

Задание 2. Для заданной предметной области «Рекламное агентство» построить диаграмму компонентов. Описать построенную диаграмму.

Основными процессами в рекламном агентстве являются: рассмотрение заявок, обработка заказов, подготовка к выпуску и выпуск рекламной продукции. Рекламное агентство в своей работе использует систему антиплагиата и руководствуется текущим законодательством. Агентство занимается изготовлением щитов, баннеров, рекламных буклетов и продвижением в социальных сетях. Сроки и стоимость заказа согласовываются на этапе заключения договора, но могут меняться в процессе выполнения заказа. В случае изменения условий составляется дополнительное соглашение к договору. Продукция проходит контроль качества.

Задание 3. Для заданной предметной области «Ресторан» построить диаграмму компонентов. Описать построенную диаграмму.

Основные направления деятельности ресторана: производство кулинарной продукции и ее реализация, организация обслуживания, реализация покупных товаров. В процессе производства кулинарные изделия порционируют, оформляют и отпускают потребителю. В процессе

обслуживания осуществляются сервировка столов, уборка помещения, подготовка персонала, размещение и встреча гостей, оформление заказов, передача заказов на производство, расчет с клиентами. Реализация покупных товаров подразумевает продажу сувенирной продукции. Учет заказов, а также расходования продуктов по заказам ведется вручную.

ПРАКТИЧЕСКАЯ РАБОТА № 28

Тема: Построение диаграмм потоков данных

Цель работы: научиться на основании сформированного списка основных бизнес-процессов строить диаграмму потоков данных DFD и диаграмму декомпозиции в нотации IDEF3.

Оборудование: ПК, программное обеспечение – MS Word, Smart Draw или UML Diagram или Visual Paradigm Online, инструкции по выполнению работы.

Справочный материал:

Диаграмма потоков данных (Data Flow Diagram, DFD) — это схема работы системы, которая наглядно показывает, откуда поступает информация, как она обрабатывается, где хранится и кому передаётся. Правильно построенная диаграмма улучшает понимание логики системы и снижает риск ошибок при разработке.

Элементы

- **Процессы** — действия, преобразующие входные данные в выходные. Обозначаются кругами или овалами, имеют уникальные номера и названия.
- **Потоки данных** — показывают, как данные перемещаются между процессами, хранилищами данных и внешними сущностями. Обозначаются стрелками и подписываются именами данных.
- **Хранилища данных** — места, где данные сохраняются на определённое время. Обозначаются двумя параллельными линиями или открытыми прямоугольниками.
- **Внешние сущности** — объекты или лица, которые взаимодействуют с системой, но не являются её частью. Обозначаются прямоугольниками и подписываются именами.

Правила построения

1. Каждый процесс должен сопровождаться как минимум одним входным и одним выходным потоком.
2. В каждое хранилище должен впадать как минимум один поток данных и как минимум один — вытекать.
3. Данные, хранимые в системе, должны проходить через процесс.
4. Каждый процесс диаграммы DFD должен вести либо к другому процессу, либо к хранилищу данных.

Содержание работы:

Задание 1. Построение модели бизнес-процесса рассмотрим на примере работы мебельной фабрики. Во время проведения обследования предприятия были выявлены ее целевые задачи, функциональные деятельности каждого из подразделений предприятия и функциональные взаимодействия между ними; информационные потоки внутри подразделений и между ними; внешние по отношению к предприятию объекты и внешние информационные воздействия, а так же нормативно-справочная документация, данные по имеющимся на предприятии средствам и системам автоматизации.

Целевые функции мебельной фабрики: переработка сырья; изготовление деталей для мебели; сборка изделия; контроль качества.

Нормативные документы мебельной фабрики: чертежи (деталей, сборочный); нормы по переработке сырья; стандарты качества; производственные инструкции; инструкции по технике безопасности.

Подразделения предприятия: цех по обработке сырья и бракованных изделий; цех по изготовлению деталей; сборочный цех; отдел проверки качества изделия.

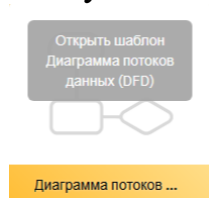
Основным сырьем для изготовления мебели является дерево. Определим основной бизнес-процесс, используя выявленные целевые функции. Так как основное предназначение мебельной фабрики состоит в том, чтобы изготавливать мебель, значит, основным бизнес-процессом является ИЗГОТОВЛЕНИЕ МЕБЕЛИ.

Для построения контекстной диаграммы нам необходимо определить входную информацию (данные или материальные ресурсы), которая преобразуется в процессе для получения результата; выходную информацию - готовый результат; управление, которое влияет на процесс, но не преобразуется процессом; механизмы, которые выполняют процесс.

Для контекстного процесса ИЗГОТОВЛЕНИЕ МЕБЕЛИ определим необходимую информацию:

- ВХОД - сырье;
- УПРАВЛЕНИЕ – чертежи, производственные инструкции, инструкции по технике безопасности (нормативные документы);
- МЕХАНИЗМЫ – персонал, производственное оборудование;
- ВЫХОД – готовая мебель.

1. Запустите программу Smart Draw, выберите шаблон



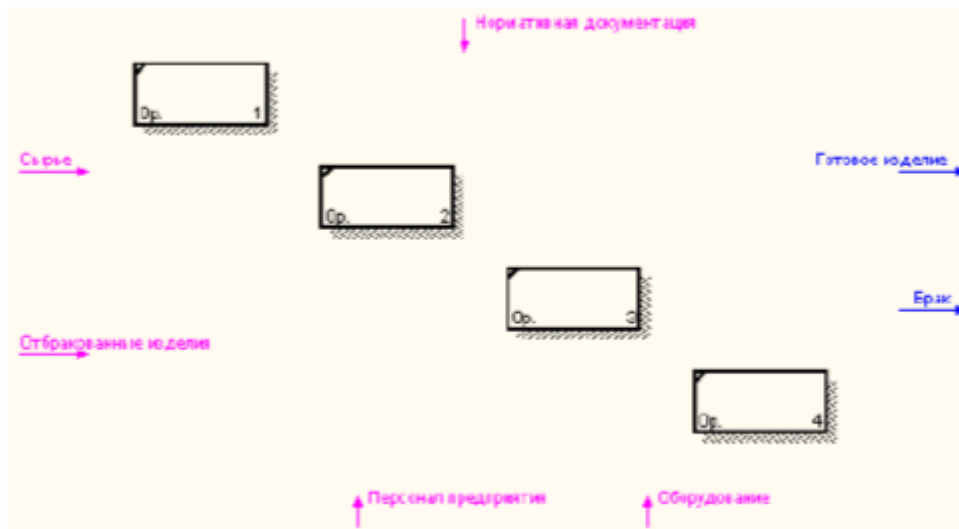
2. Создайте диаграмму, показанную на рисунке



3. Следующим шагом является детализация контекстного процесса с помощью диаграммы верхнего уровня. Эта диаграмма содержит в себе четыре процесса:

- 1) Процесс 1.1 – ПЕРЕРАБОТКА СЫРЬЯ.
- 2) Процесс 1.2 – ИЗГОТОВЛЕНИЕ ДЕТАЛЕЙ.
- 3) Процесс 1.3 – СБОРКА ИЗДЕЛИЯ.
- 4) Процесс 1.4 – КОНТРОЛЬ КАЧЕСТВА.

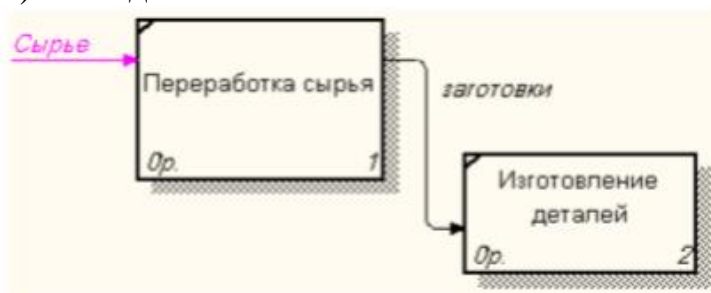
Произведите детализацию процесса «Изготовление мебели», задав нужное количество новых блоков. При декомпозиции функции, входящие в нее и исходящие из нее дуги, автоматически появляются на диаграмме декомпозиции (миграция дуг), но при этом не касаются блоков. Такие стрелки называются несвязанными.



4. Определим входные и выходные потоки для новых процессов.

Процесс 1.1. ПЕРЕРАБОТКА СЫРЬЯ:

- 1) Вход – СЫРЬЁ.
- 2) Вход – ОТБРАКОВАННЫЕ ИЗДЕЛИЯ.
- 3) Выход – ЗАГОТОВКИ.



5. Самостоятельно выполните детализацию процессов:

Процесс 1.2. ИЗГОТОВЛЕНИЕ ДЕТАЛЕЙ:

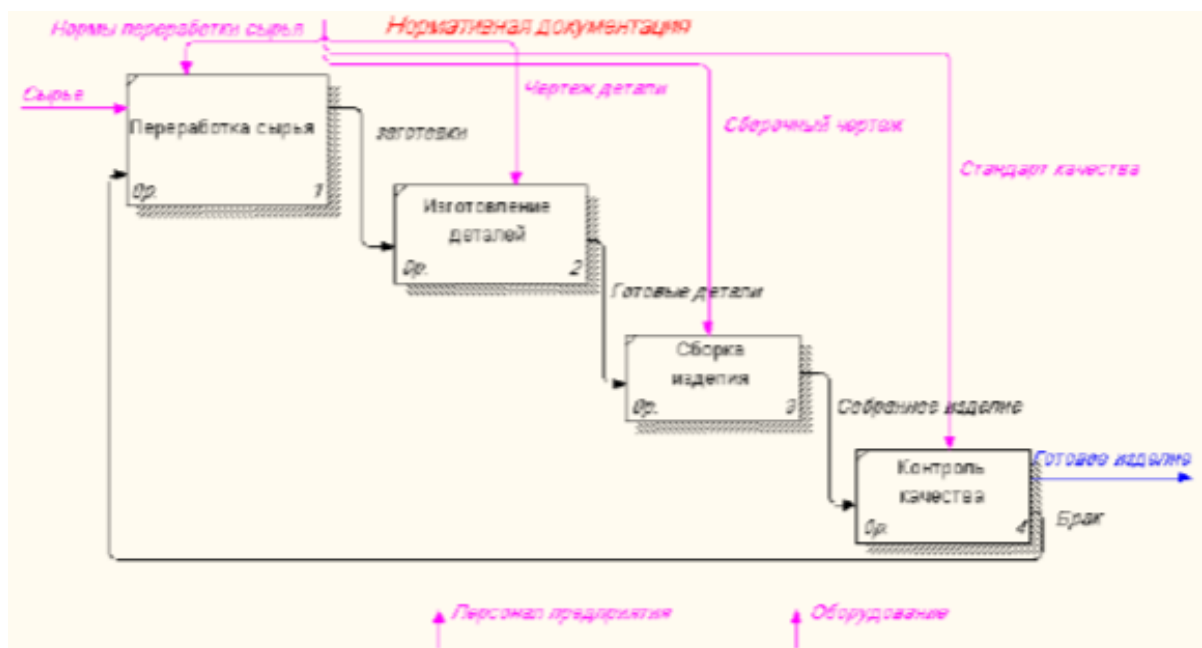
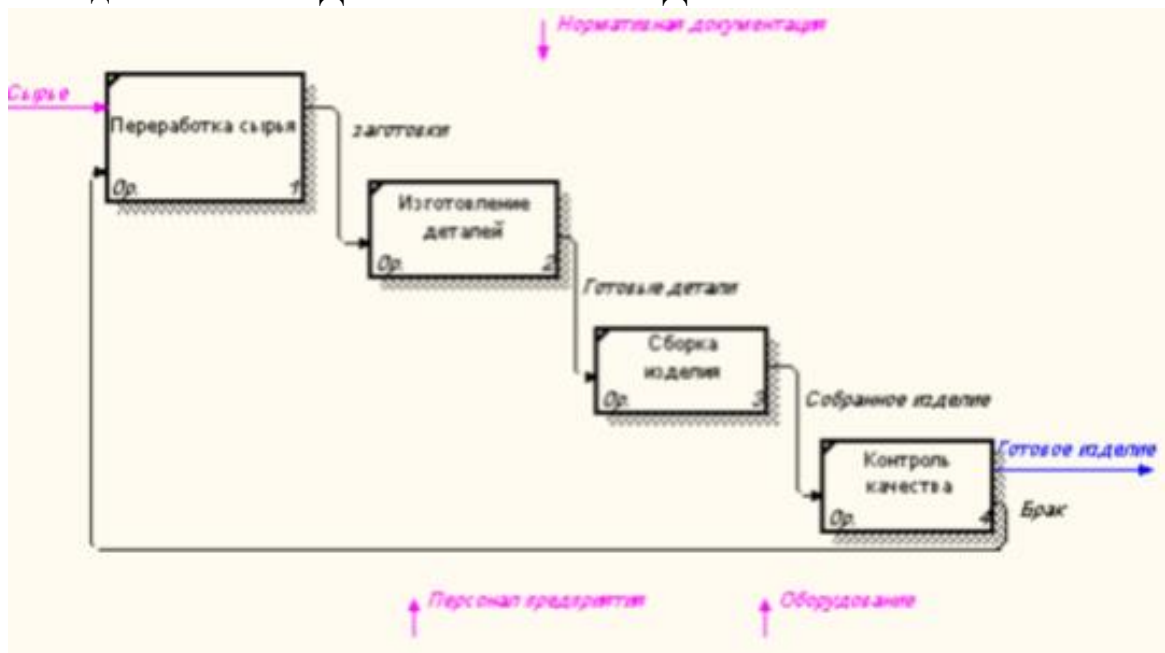
- 1) Вход – ЗАГОТОВКИ.
- 2) Выход – ГОТОВЫЕ ДЕТАЛИ.

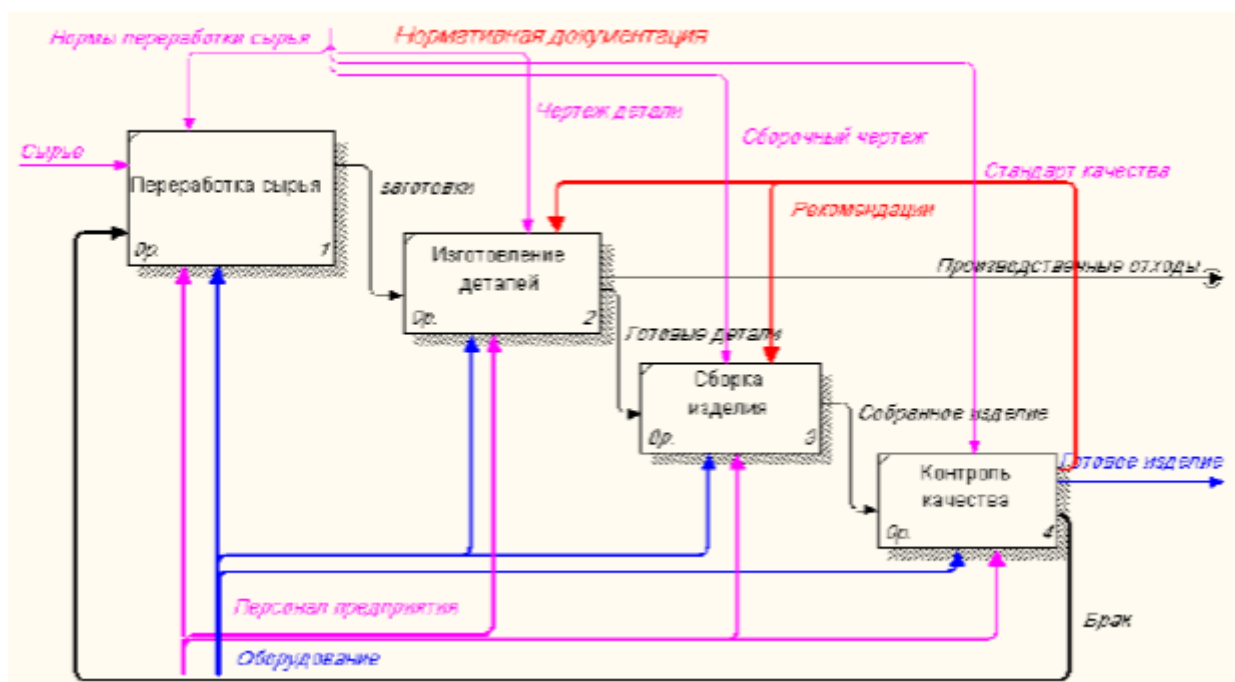
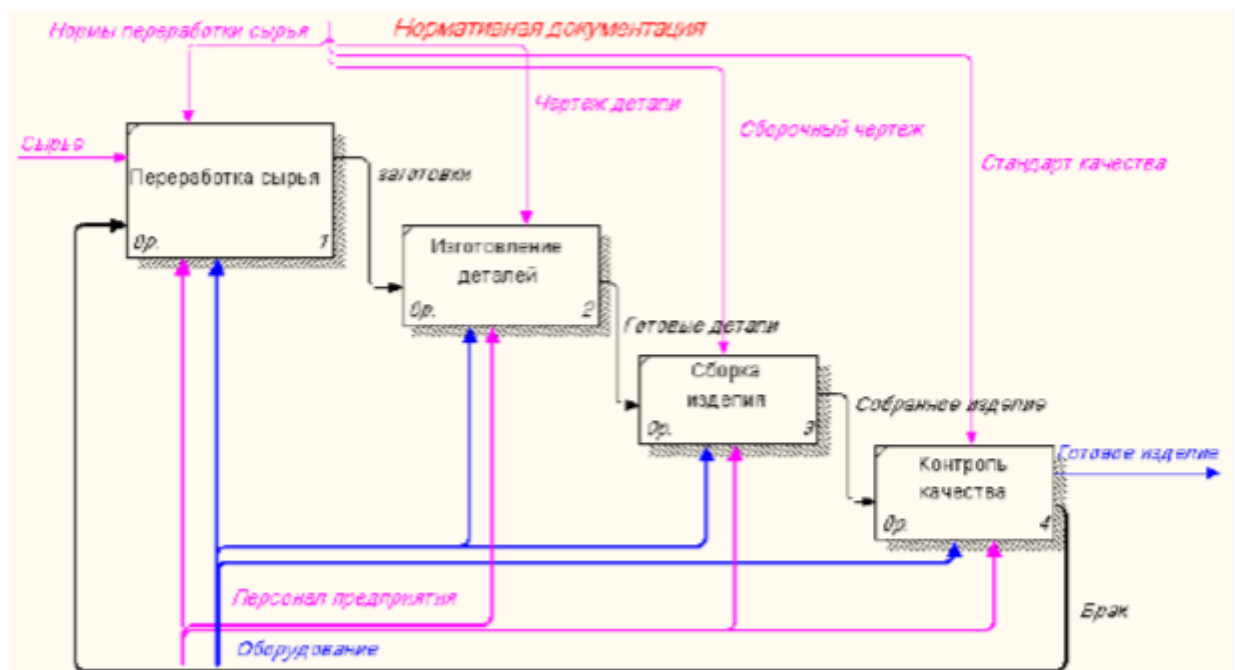
Процесс 1.3. СБОРКА ИЗДЕЛИЯ:

- 1) Вход – ГОТОВЫЕ ДЕТАЛИ.
- 2) Выход – СОБРАННОЕ ИЗДЕЛИЕ.

Процесс 1.4. КОНТРОЛЬ КАЧЕСТВА:

- 1) Вход – СОБРАННОЕ ИЗДЕЛИЕ.
- 2) Выход – ГОТОВОЕ ИЗДЕЛИЕ.
- 3) Выход – БРАК.
- 4) Выход – ПРОИЗВОДСТВЕННЫЕ ОТХОДЫ





Задание 2. Построить диаграмму потоков данных в соответствии с вариантом.

Отчет по практическому занятию выполняется в формате MS Word, который содержит пошаговое описание процесса построения диаграммы, а также скриншоты результатов согласно заданию.

Варианты:

- 1 «Отдел кадров»;
- 2 «Агентство аренды»;
- 3 «Аптека»;
- 4 «Ателье»;
- 5 «Аэропорт»;

ПРАКТИЧЕСКАЯ РАБОТА № 29

Тема: Построение диаграмм потоков данных

Цель работы: исследование процесса построения диаграмм потоков данных в заданной предметной области; научиться строить диаграммы потоков данных с применением ПО

Оборудование: ПК, программное обеспечение – MS Word, Smart Draw или UML Diagram или Visual Paradigm Online, инструкции по выполнению работы.

Содержание работы:

Задание 1. Построить диаграмму потоков данных

Рассмотреть подсистему управления продажами. Менеджер отдела продаж ежедневно получает от клиента Заказ на конкретную номенклатурную единицу железобетонных конструкций. В Заказе номенклатурных единиц клиент указывает желаемую отсрочку платежа.

Менеджер отдела продаж ежедневно проверяет наличие необходимого количества заказанных конструкций на складе. Если конструкций недостаточно для выполнения заказа, то менеджер отдела продаж размещает Заказ в реестре "неудовлетворенный спрос". Затем менеджер ежедневно проверяет возможность выполнения Заказа, размещенного в реестре "неудовлетворенный спрос".

При достаточном количестве товара на складе в отделе продаж на основании Заказа и договора формируется Заявка на номенклатурные единицы. Заявки формируются ежедневно. Ежедневно на основании Заявки менеджер отдела продаж осуществляет резервирование товара.

Менеджер отдела продаж ежедневно контролирует кредитный лимит и дебиторскую задолженность потенциальных покупателей. Если кредитный лимит и дебиторская задолженность не превышают допустимых значений, то Заявка передается на склад в Учетно-операционный отдел.

При превышении кредитного лимита или наличии просроченной дебиторской задолженности свыше допустимого количества дней менеджер отдела продаж заявку в Учетно-операционный отдел не передает, процесс продаж приостанавливается, осуществляются переговоры с клиентом.

Менеджер учетно-операционного отдела, получив Заявку, ежедневно производит подборку номенклатурных единиц. Менеджер учетно-операционного отдела ежедневно формирует упаковочные листы для вложения их в каждый ящик.

Менеджером учетно-операционного отдела ежедневно формируются для клиента следующие документы: счет, расходная накладная, счет-фактура. При фактической отгрузке товара со склада осуществляется его списание. Списание товара осуществляется по расходной накладной и сопровождается формированием проводки.

Задание 2. Построить диаграмму потоков данных

Рассмотрим подсистему управления закупками. Предприятие планирует закупки автозапчастей. Планирование закупок осуществляется в Департаменте маркетинга, в отделе маркетинга и планирования.

Планирование закупок осуществляется следующим образом:

- Менеджер отдела планирования и маркетинга ежедневно получает от контрагентов данные внешней и внутренней статистики продаж автозапчастей в виде отчетов-таблиц собственных продаж и отчетов-таблиц продаж внешних источников.
- Для планирования закупок автозапчастей менеджер отдела планирования и маркетинга еженедельно на основании статистики продаж производит расчет потребности в товаре. В результате расчета формируется «Таблица потребностей в товаре», в которой определено количество и номенклатура заказываемых товаров.

Выбор поставщиков осуществляет менеджер отдела закупок.

- Ежемесячно (или по мере необходимости) в АСУ вводит прайс-листы поставщиков.
- Определив количество и номенклатуру заказываемых товаров, менеджер отдела закупок приступает к анализу предложений поставщиков. Анализ предложений поставщиков и действующих контрактов осуществляется на основании Таблицы потребностей в товаре и прайс-листов. Выбираются наиболее выгодные условия поставки. Для этого сравниваются цены поставщиков. Данные сведения берутся из прайс-листа для закупок. При выборе поставщика важно учесть предоставляемую отсрочку платежа. Эта информация берется из контрактов, отмеченных как приоритетные (действующие). В результате анализа формируется «Список поставщиков» с расстановкой приоритетов (каждой позиции присваивается признак основного и запасного поставщика в порядке убывания приоритета).
- Менеджер отдела закупок ежемесячно на основании «Таблицы потребностей в товаре» и «Список поставщиков» формирует графики поставок с указанием сроков и периодичности, но без количества поставки.

Ежемесячно после определения потребности в товаре менеджер отдела логистики формирует план заявок на месяц.

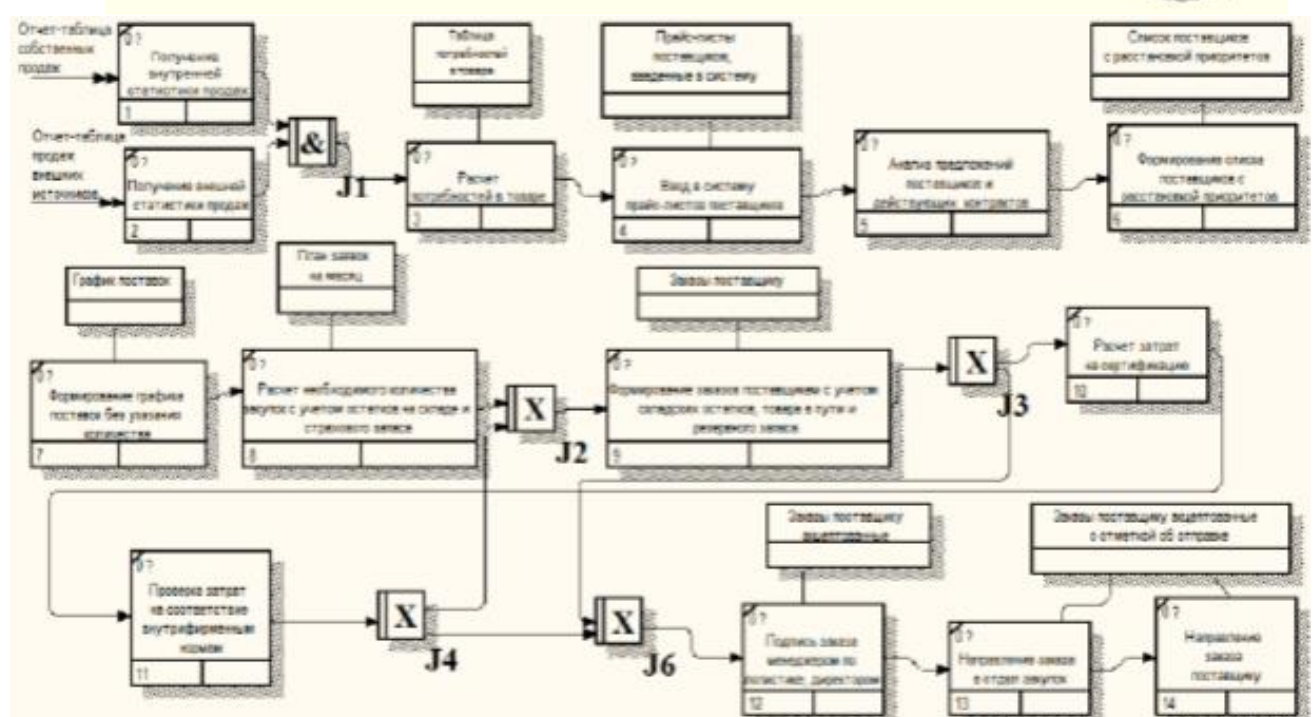
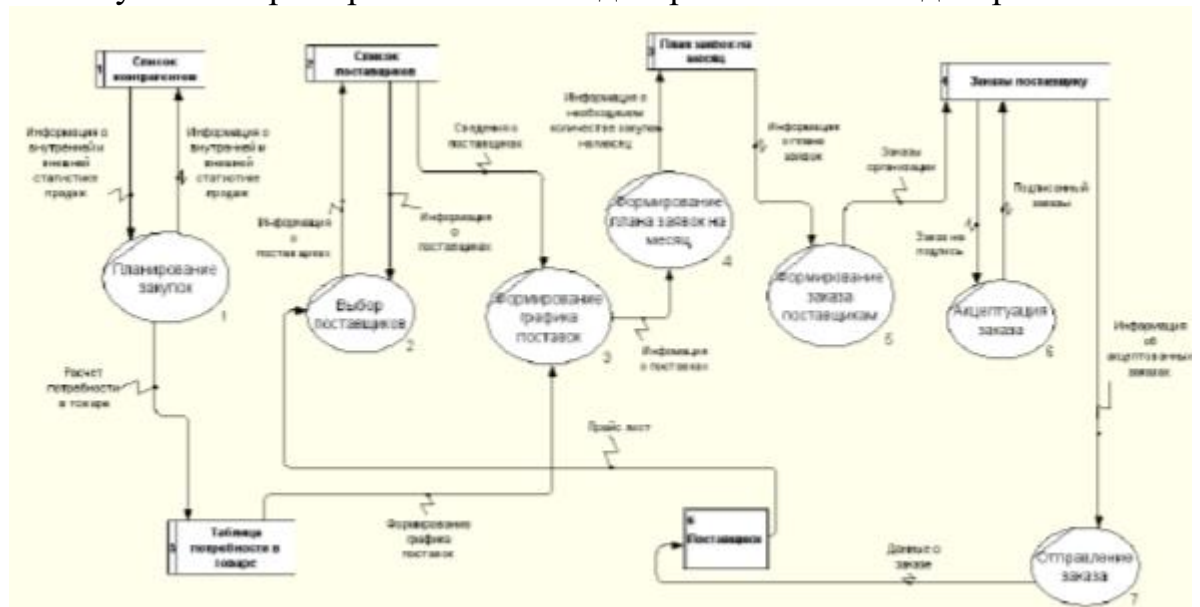
- Для этого рассчитывает необходимое количество закупок.
- Необходимое количество закупок рассчитывается на основании фактических запасов на складе, необходимого минимального и максимального уровня запасов. Нормы минимального и максимального количества запасов устанавливаются в днях. При расчете необходимого количества закупки учитывается также время товара в пути. Таким образом, данный расчет должен обеспечить возможность бесперебойного отпуска товара со склада. По результату расчетов формируется «План заявок на месяц».

Затем в отделе логистики ежедневно по плану заявок, графику поставок, прайс-листам поставщиков формируются заказы поставщикам.

- Если предстоит сделать заказ импортному поставщику, то менеджер отдела логистики рассчитывает затраты на сертификацию, создается отчет о затратах на сертификацию. Затраты на сертификацию проверяются на соответствие внутрифирменным нормам. Данная операция производится по мере необходимости.
- Если затраты на сертификацию превышают внутрифирменные нормы, то менеджер отдела логистики повторяет процесс формирования заказов поставщикам. Формируются новые заказы.

Ежедневно подготовленный заказ поставщику акцептуется, заказ должен подписать менеджер по логистике и директор Департамента маркетинга.

Ежедневно менеджер отдела логистики направляет заказ в отдел закупок. Менеджер отдела закупок направляет заказ поставщику. Должны получиться примерно такие DFD-диаграмма и IDEF3-диаграмма:



ПРАКТИЧЕСКАЯ РАБОТА № 30

Тема: Оценка необходимого количества тестов

Цели работы: усвоить знание о видах тестирования; освоить способы обнаружения и фиксирования ошибок. Научиться оценивать необходимое количество тестов.

Оборудование: ПК, программное обеспечение –MS Word, инструкции по выполнению работы.

Справочный материал:

Сколько тестов понадобится, чтобы обнаружить ошибки? Для ответа на этот вопрос предлагается следующая модель. Утверждается, что вероятность успешности очередного теста зависит от процента оставшихся ошибок. Последние ошибки обнаружить сложнее.

Пусть T_n — среднее количество тестов, необходимых для обнаружения n ошибок, N — число ошибок в программе. Для оценки T_n предлагается следующая формула:

$$T_n = \alpha \sum_{k=0}^{n-1} \frac{1}{N-k}.$$

Здесь α — некий коэффициент, значение которого надо определять экспериментально.

Мы хотим оценить количество тестов, которое потребуется для нахождения всех N ошибок. Посчитать напрямую T_n мы не можем, поскольку не знаем коэффициента α . Чтобы исключить его из вычислений, перейдем от абсолютных величин к относительным. Попробуем оценить, какую часть от всех необходимых тестов придется выполнить для того, чтобы найти первые n ошибок

$$P = \frac{T_n}{T_N} = \frac{\alpha \sum_{k=0}^{n-1} \frac{1}{N-k}}{\alpha \sum_{k=0}^{N-1} \frac{1}{N-k}} = \frac{\sum_{k=0}^{n-1} \frac{1}{N-k}}{\sum_{k=0}^{N-1} \frac{1}{N-k}}.$$

Теперь осталось подставить в формулу для P конкретные значения N и n . Результаты представлены в таблицах. В первой таблице N изменяется от 10 до 100, во второй таблице — от 1 до 10. Последняя таблица построена специально для новичков, программы которых настолько малы по размерам, что в них просто не найдется места для сотни ошибок.

Оказывается, что при $N = 10$ первые 22% тестов обнаружат половину всех ошибок (5 штук). Для того чтобы обнаружить две следующие ошибки, количество тестов придется увеличить в 1,7 раза — до 37%. Поиск следующих двух ошибок потребует увеличения количества тестов еще в 1,8 раза — до 66%. И наконец, поиск последней ошибки потребует оставшихся 34% тестов. Чем больше ошибок в программе, тем дороже обойдется поиск последних ошибок. При $N = 50$ для обнаружения 50% ошибок достаточно 15% тестов, 70% ошибок будут найдены с помощью 26%, 90% ошибок — с помощью 49% тестов. Поиск последних 10% ошибок будет стоить дороже, чем поиск первых 90%!

Имея такие оценки относительного количества тестов, в конкретном проекте можно перейти к абсолютным величинам. Поскольку нам известно, сколько тестов нам понадобилось для нахождения n ошибок, легко оценить, сколько понадобится для нахождения оставшихся. Если количество ошибок в программе оценено в 10 и для обнаружения первых пяти потребовалось, например, 7 тестов, то для поиска двух следующих ошибок количество тестов придется довести до 12 (5 дополнительных тестов на 2 ошибки), для поиска следующих двух — до 21 (9 дополнительных тестов на 2 ошибки). Все 10 ошибок есть надежда найти за 32 теста.

Подобные оценки можно использовать для планирования времени и ресурсов, выделяемых для процесса тестирования.

Содержание работы:

Задание 1. Найти минимальный набор тестов для программы нахождения вещественных корней квадратного уравнения $ax^2 + bx + c = 0$

Решение представлено в таблице.

Но- мер теста	a	b	c	Ожидаемый результат	Что проверяется
1	2	-5	2	$x_1=2, x_2=0,5$	Случай вещественных корней
2	3	2	5	Сообщение	Случай комплексных корней
3	3	-12	0	$x_1=4, x_2=0$	Нулевой корень
4	0	0	10	Сообщение	Неразрешимое уравнение
5	0	0	0	Сообщение	Неразрешимое уравнение
6	0	5	17	Сообщение	Неквadraticное уравнение
7	9	0	0	$x_1=x_2=0$	Нулевые корни

Таким образом, для этой программы предлагается минимальный набор функциональных тестов, исходя из 7 классов выходных данных.

Следовательно, легко оценить, сколько понадобится для нахождения оставшихся. Если количество ошибок в программе оценено в 4 и для обнаружения первых двух потребовалось, например, 4 теста, то для поиска двух следующих ошибок количество тестов придется довести до 10 (6 дополнительных тестов на 2 ошибки). Все 4 ошибки есть надежда найти за 10 тестов.

Задание 2. Найти минимальный набор тестов для следующих программ:

1. Вывести на экран все целые числа от 100 до 200, кратные трем.
2. Найти сумму положительных нечетных чисел, меньших 50.
3. Определить количество трехзначных натуральных чисел, сумма цифр которых равна целому числу n ($0 < n \leq 27$).

ПРАКТИЧЕСКАЯ РАБОТА № 31

Тема: Оценка необходимого количества тестов

Цели работы: усвоить знание о видах тестирования; освоить способы обнаружения и фиксирования ошибок. Научиться оценивать необходимое количество тестов.

Оборудование: ПК, программное обеспечение –MS Word, инструкции по выполнению работы.

Содержание работы:

Задание 1. Найти минимальный набор тестов для следующих программ:

1. Известны данные о количестве страниц в каждой из нескольких газет и в каждом из нескольких журналов. Число страниц в газете не более 16. Найти общее число страниц во всех журналах (количество журналов неизвестно, но известно, что объем любого журнала превышает объем любой газеты).
2. Известно число детей, учащихся во всех первых классах, во всех вторых, ..., во всех одиннадцатых классах. Определить общее число детей, учащихся в первых, вторых, третьих и т.д. классах школы. Оператор цикла с шагом, отличным от 1 и -1, не использовать.
3. Известен год рождения каждого человек из группы. Определить число людей, родившихся до 1985 года, и число людей, родившихся после 1990 года.

ПРАКТИЧЕСКАЯ РАБОТА № 32

Тема: Разработка тестового сценария.

Цель работы: усвоить знание о видах тестирования; получить навыки разработки тестовых сценариев.

Оборудование: ПК, программное обеспечение –MS Word, инструкции по выполнению работы

Справочный материал:

Для разработки тестового сценария проекта вам понадобится четко определить цели тестирования, описать шаги, которые необходимо выполнить, и критерии, по которым будет оцениваться успешность теста.

Вот примерный план создания тестового сценария:

1. Определение целей тестирования. Что именно вы хотите проверить? Это может быть функциональность, производительность, безопасность или совместимость.
2. Выбор тестовых данных. Какие данные будут использоваться для тестирования? Убедитесь, что они репрезентативны и соответствуют реальным условиям использования.
3. Описание шагов тестирования. Какие действия необходимо выполнить? Опишите последовательность действий, которые должен выполнить тестирующий.
4. Ожидаемые результаты. Чего вы ожидаете после выполнения тестовых шагов? Это поможет определить, прошел ли тест успешно.
5. Фактические результаты и оценка. Запишите, что произошло в результате тестирования и сравните это с ожидаемыми результатами.
6. Документирование и анализ. Все результаты должны быть задокументированы. Если тест не пройден, проанализируйте причины и определите шаги для устранения проблем.

Для создания детализированного тестового сценария для программы на Visual Studio, вам нужно будет следовать этим шагам:

1. Описание проекта и тестируемой функциональности. Кратко опишите проект и функциональность, которую вы собираетесь тестировать.
2. Тестовое окружение. Укажите конфигурацию тестового окружения, включая версию Visual Studio, операционную систему и любое другое ПО, необходимое для тестирования.
3. Предварительные условия. Перечислите все условия, которые должны быть выполнены перед началом теста (например, запущенные сервисы, настроенные параметры).
4. Тестовые данные. Определите и подготовьте тестовые данные, которые будут использоваться в тесте.
5. Шаги тестирования.

Описание действия.

- Ожидаемый результат: Что должно произойти после выполнения шага.
- Фактический результат: Записывается во время выполнения теста.
- Статус: Успешно / Не успешно.

Продолжите описание шагов до завершения тестового сценария.

6. Постусловия: Опишите состояние системы после выполнения теста.
7. Критерии успеха: Определите, как будет измеряться успех теста.
8. Очистка: Опишите шаги для возврата системы в исходное состояние после тестирования.
9. Документирование результатов: Запишите результаты тестирования и сделайте выводы о качестве программы.
10. Анализ и действия по улучшению: Если тесты выявили проблемы, опишите шаги для их устранения.

Содержание работы:

Задание 1. Пример тестового сценария для функции логина в приложении.

Тестовый сценарий: Проверка функции логина.

Тестовое окружение: Visual Studio 2022, Windows 10

Предварительные условия: Приложение установлено и запущено.

Тестовые данные: Пользовательские учетные данные (логин и пароль).

Шаги тестирования:

1. Введите корректный логин и пароль.

- Ожидаемый результат: Пользователь успешно вошел в систему.

- Фактический результат: ...

- Статус: ...

2. Введите некорректный логин и пароль.

- Ожидаемый результат: Система отображает сообщение об ошибке.

- Фактический результат: ...

- Статус: ...

Постусловия: Пользователь выходит из системы.

Критерии успеха: Все шаги выполнены успешно, и результаты соответствуют ожидаемым.

Очистка: Закрыть приложение.

Документирование результатов: Результаты записаны и проанализированы.

Анализ и действия по улучшению: Определены шаги для исправления обнаруженных проблем.

Задание 2. Разработайте набор тестовых сценариев (как позитивных, так и негативных) для следующей программы:

Имеется консольное приложение (разработайте самостоятельно). Ему на вход подается 2 строки. На выходе приложение выдает число вхождений второй строки в первую. Например:

Строка 1	Строка 2	Вывод
абвгабвг	аб	2
стстсап	стс	2

Задание 3. Напишите тестовые сценарии для приложения электронной коммерции.

Тестовый сценарий 1. Проверка функциональности входа

Тестовый сценарий 2. Проверка функциональности поиска

Тестовый сценарий 3: проверьте страницу описания продукта
Тестовый сценарий 4: Проверьте функциональность платежей
Тестовый сценарий 5: Проверка истории заказов

ПРАКТИЧЕСКАЯ РАБОТА № 33

Тема: Разработка тестового сценария

Цель работы: усвоить знание о видах тестирования; получить навыки разработки тестовых сценариев.

Оборудование: ПК, программное обеспечение –MS Word, инструкции по выполнению работы

Содержание работы:

Задание 1. Напишите тестовые сценарии для банковского сайта

Тестовый сценарий 1: Проверьте функциональность входа и аутентификации

Тестовый сценарий 2: Проверить перевод денег можно

Тестовый сценарий 3. Проверьте выписку со счета.

Тестовый сценарий 4: Проверка фиксированного депозита / периодического депозита может быть создана

Задание 2. Написать тестовый сценарий для проверки регистрации нового пользователя в веб-приложении

Задание 3. Написать тестовый сценарий для проверки покупки товара в онлайн-магазине

ПРАКТИЧЕСКАЯ РАБОТА № 34

Тема: Разработка тестового сценария

Цель работы: усвоить знание о видах тестирования; получить навыки разработки тестовых сценариев.

Оборудование: ПК, программное обеспечение –MS Word, инструкции по выполнению работы

Содержание работы:

Задание 1. Написать тестовый сценарий для web-приложения или сайта.

- 1.Тестирование функциональности: проверьте все ссылки, проверьте формы, тестирование файлов cookie, проверьте HTML/CSS, тестирование базы данных, ссылки, формы, база данных
- 2.Тестирование удобства использования (юзабилити сайта): проверка навигации, проверка контента, другая информация для пользователей
- 3.Тестирование пользовательского интерфейса
- 4.Проверка совместимости: совместимость с браузерами, совместимость с операционными системами, просмотр на мобильных устройствах, параметры печати
- 5.Тестирование производительности сайта: скорость соединения, нагрузка, стрессовая нагрузка
- 6.Тестирование безопасности

Проверьте все ссылки, присутствующие на веб-странице, а также ссылки на базы данных, формы, используемые для подтверждения действий и получения информации от пользователей, файлы Cookie и т.д. Проверьте ссылки, исходящие от всех страниц к конкретному домену.

- Внутренние ссылки.
- Ссылки на другие элементы, расположенные внутри страниц.
- Ссылки для отправления электронной почты администратору или другим пользователям веб-страниц.
- Проверьте, нет ли ссылок на изолированные страницы.

Формы используются для получения информации от пользователей и взаимодействия с ними. Что нужно проверить в формах:

- Правильность работы валидации в каждом поле формы.
- Значения полей, используемые по умолчанию.
- Опции для создания форм, удаления, просмотра и редактирования форм (если такие имеются).

Тестирование юзабилити — это анализ взаимодействия пользователя и сайта, поиск ошибок и их устранение. При этом проверяется: легкость обучения; навигация; субъективная удовлетворенность пользователей; общий вид.

Под навигацией подразумеваются средства для просмотра страниц пользователем. Это кнопки, блоки. А также то, как посетитель сайта использует ссылки на другие страницы.

Проверка юзабилити:

- Сайт должен быть простым в использовании;

- Инструкции должны быть очень четкими;
- Проверьте, достигают ли предоставленные инструкции поставленной цели;
- Главное меню должно быть доступно на каждой странице;
- Главное меню должно быть построено в логической последовательности.

Проверьте, что происходит, когда пользователь прерывает какое-либо действие. А также, что происходит при повторном подключении к серверу в ходе выполнения какой-либо операции. Нужно проверить: совместимость с браузерами; совместимость с операционными системами; просмотр на мобильных устройствах; параметры печати.

ПРАКТИЧЕСКАЯ РАБОТА № 35

Тема: Разработка тестовых пакетов

Цель работы: получить навыки разработки тестовых пакетов.

Оборудование: ПК, программное обеспечение –MS Word, инструкции по выполнению работы.

Справочный материал:

Тестовые пакеты — это совокупность тестовых случаев, которые проверяют определенную функциональность или компонент программного обеспечения. Они помогают обеспечить, что ПО работает корректно и соответствует требованиям.

Практическая работа по разработке тестовых пакетов предполагает создание комплекса тестов, которые проверяют различные аспекты программного обеспечения.

Шаги, которые помогут вам разработать тестовые пакеты:

1. Определение области тестирования. Определите, какие функции или модули программы будут тестироваться.
2. Разработка тестовых случаев. Создайте список тестовых случаев, которые покрывают все возможные сценарии использования функций или модулей.
3. Подготовка тестовых данных. Соберите или сгенерируйте данные, которые будут использоваться в тестах.
4. Настройка тестового окружения. Убедитесь, что у вас есть все необходимое для выполнения тестов (например, доступ к серверам, настроенные инструменты).
5. Выполнение тестов. Запустите тесты и зафиксируйте результаты.
6. Анализ результатов. Оцените результаты тестов и определите, соответствуют ли они ожиданиям.
7. Документирование. Запишите процесс тестирования и результаты в отчеты.
8. Улучшение и исправление. Если были найдены ошибки, разработайте план их исправления.

Пример структуры документа для тестового пакета

Тестовый пакет	для функции Авторизации
Область тестирования	- Функция входа в систему - Восстановление пароля - Регистрация нового пользователя
Тестовые случаи	
Тестовый случай 1	Успешный вход в систему
Предварительные условия	Пользователь зарегистрирован в системе
Тестовые данные	Корректные логин и пароль
Шаги	1. Открыть страницу входа. 2. Ввести корректные логин и пароль.

	3. Нажать кнопку "Войти".
Ожидаемый результат	Пользователь перенаправлен на главную страницу
Фактический результат	
Статус	
Тестовый случай 2	Вход с неверным паролем
Предварительные условия	Пользователь зарегистрирован в системе
Тестовые данные	Корректный логин и некорректный пароль
Шаги	1. Открыть страницу входа. 2. Ввести корректный логин и некорректный пароль. 3. Нажать кнопку "Войти".
Ожидаемый результат	Система отображает сообщение об ошибке
Фактический результат	
Статус	
Анализ результатов	
Общие выводы:	Расписать: Обнаруженные проблемы Рекомендации по улучшению
Документирование	Отчеты о тестировании Логи ошибок
Улучшение и исправление	1. План исправления ошибок 2. Меры по предотвращению подобных ошибок в будущем

Содержание работы:

Задание 1. Напишите тестовый пакет для проверки функции логина в приложении.

Задание 2. Напишите тестовый пакет для приложения электронной коммерции.

ПРАКТИЧЕСКАЯ РАБОТА № 36

Тема: Разработка тестовых пакетов

Цель работы: получить навыки разработки тестовых пакетов.

Оборудование: ПК, программное обеспечение –MS Word, инструкции по выполнению работы.

Содержание работы:

Задание 1. Напишите тестовый пакет для банковского сайта

Задание 2. Написать тестовый пакет для проверки регистрации нового пользователя в веб-приложении

Задание 3. Написать тестовый пакет для проверки покупки товара в онлайн-магазине

ПРАКТИЧЕСКАЯ РАБОТА № 37

Тема: Разработка тестовых пакетов

Цель работы: получить навыки разработки тестовых пакетов.

Оборудование: ПК, программное обеспечение – MS Word, инструкции по выполнению работы.

Содержание работы:

Задание 1. Написать тестовый пакет для сайта электронной библиотеки

Задание 2. Написать тестовый пакет для web-приложения

Задание 2. Написать тестовый пакет для мобильного приложения

ПРАКТИЧЕСКАЯ РАБОТА № 38

Тема: Оценка программных средств с помощью метрик

Цель работы: научить оценивать ПО с помощью метрик, применяя разные методы.

Оборудование: ПК, программное обеспечение – MS Word, инструкции по выполнению работы.

Содержание работы:

Задание 1. Разработать программу для вычисления значений функции:

$$F = \begin{cases} \sin(x) + \cos^2(y), & \text{при } x < y; \\ \ln(x), & \text{при } x = y; \\ \sin^2(x) + \cos(y), & \text{при } x > y. \end{cases}$$

Значения аргументов функции ввести с клавиатуры. На экран монитора вывести значение функции. Определить значения метрик Холстеда, на основе которых дать оценку качества разработанного исходного текста программы.

Реализация программы. Текст программы для реализации возможного решения поставленной задачи, разработанной с использованием языка программирования C#, приведен в таблице 1.

Номер строки	Строки программы
1.	using System;
2.	namespace Ex
3.	{
4.	class Program
5.	{
6.	static void Main()
7.	{
8.	double x, y, F;
9.	char check;
10.	do
11.	{
12.	Console.WriteLine("Введите значение переменной x");
13.	Console.Write("x=");
14.	x = double.Parse(Console.ReadLine());
15.	Console.WriteLine("Введите значение переменной y");
16.	Console.Write("y=");
17.	y = double.Parse(Console.ReadLine());
18.	if (x < y)
19.	F = Math.Sin(x) + Math.Cos(y)* Math.Cos(y);
20.	else
21.	if (x == y)
22.	F = Math.Log(x);
23.	else
24.	F = Math.Sin(x) * Math.Sin(x) + Math.Cos(y);
25.	Console.WriteLine("F = "+F);

26.	Console.WriteLine("Хотите запустить программу вновь? Y/N");
27.	chek=char.Parse(Console.ReadLine());
28.	} while (check= 'Y' check= 'y');
29.	}
30.	}
31.	}

Словарь программы. В таблице 2 приведены операторы и операции, используемые в программе.

№ п/п	Операторы, операции	Номера строк исходной программы, где встречается оператор или операция	Количество повторений с тексте исходной программы
1	using ...;	1	1
2	namespace ...	2	1
3	class ...	4	1
4	static void...	6	1
5	double...	8	1
6	char...	9	1
7	do... while()	10-28	1
8	Console.WriteLine()	12, 15. 25.26	4
9	Console.Write()	13. 16	2
10Parse()	14. 17.27	3
II	Console. ReadLine()	14. 17. 27	3
12	if ()... else...	18.21	2
13	Math.Sin()	19. 24.24	3
14	Math.Cos()	19. 19. 24	3
15	Math.Log()	22	1
16	;	1.8.9. 12. 13. 14. 15. 16. 17. 19. 22. 24. 25. 26. 27. 28	16
17	,	8.8	2
18	*	19. 24	2
19	=	14. 17. 19. 22. 24. 27	
20	+	19.24. 25	3
21	<	18	1
22	= =	21.28. 28	3
23	{}	3(31). 5(30). 7(29). 11(28)	4
24	()	6. 12. 13. 14. 14. 15. 16. 17. 17. 18. 23 19. 19. 19.21.22. 24. 24. 24. 25. 26. 27.27. 28	
25		28	1
26	" "	12. 13. 15. 16.25. 26.	6
27	' '	28. 28	2
28	-	12, 13, 14. 14. 15. 16. 17. 17, 19, 19, 19, 22.24.24.24. 25. 26. 27.27	19
Всего			116

В таблице 3 приведены операнды программы.

№ n/n	Операнды	Номера строк	Количество повторений
1	System	1	1
2	Ex	2	1
3	Program	4	1
4	Main	6	1
5	x	8. 14. 18. 19.21.22. 24. 24	8
6	y	8. 17. 18. 19. 19.21.24	7
7	check	9. 27. 28, 28	4
8	"Введите значение переменной x"	12	1
9	"x="	13	1
10	"Введите значение переменной y"	15	1
II	"y="	16	1
12	F	8. 19, 22. 24. 25	5
13	"Хотите запустить программу вновь? Y/N"	26	1
14	'Y'	28	1
15	'y'	28	1
16	"F =*	25	1
Всего			36

Таким образом, количество строк таблицы 3 есть число уникальных операторов и операций, появляющихся в данном тексте. Если вычислить сумму значений из четвертого столбца, то получим общее число всех операторов и операций, используемых в исходном тексте программы. Отметим, что для фигурных скобок, определяющих блок, приведены два номера строки. Первый определяет левую фигурную скобку, открывающую блок, а второй – закрывающую. Отметим, что такая пара в словаре учитывается только один раз.

Проведем подробный анализ исходного текста программы в соответствии с полученной таблицей 1, начиная с ее первой позиции. Первая строка программы `using System;` (таблица 1, п.1). Ключевое слово `using` представляет собой команду (инструкцию), обеспечивающую доступ к именам пространства имен `System`. Следовательно, команду `using` можно отнести к выполняемым операторам (таблица 2, п. 1). Оператор `using` встречается в программе всего один раз.

Слово `System` представляет собой имя, над которым осуществляется операция `using`. Таким образом, имя `System` заносится в таблицу словаря операндов (таблица 3, п. 1). Имя `System` встречается в программе один раз.

Следующая строка программы `namespace Ex` (таблица 1, п. 2). Состоит из оператора `namespace` и операнда `Ex`, которые также присутствуют в тексте программы в единственном экземпляре. Оператор занесен в таблицу операторов (таблица 2, п. 2), а операнд `Ex` – в таблицу операндов (таблица 3, п. 2).

Строки `class Program`, `static void Main()`, `double x, y, F;` `char check;` также представляют собой сочетание операторов и операндов, которые встречаются в тексте один раз (таблица 1, п. 4, 6 и 8), где ключевые слова

class, static void, double и char представляют соответственно операции, а Program, Main, x, y, F и check – имена (операнды). Все операции попадают в словарь операторов (таблица 2), а имена – в словарь операндов (таблица 3).

Следующий оператор `do ... while()` (таблица 1, п. 10–28). Представляет собой инструкцию реализации циклического алгоритма, которая используется в тексте программы один раз.

Рассмотрим тело цикла (блок операторов, заключенных между ключевыми словами `do ... while`). Первой строчкой цикла является операция вызова функции вывода строк на экран монитора `Console.WriteLine()`. Данная операция повторяется в тексте программы 4 раза (таблица 1, п. 12, 15, 25, 26). В каждом из этих случаев применения оператора вызова функции (метода) `Console.WriteLine()` входным параметром функции является строка (строковая константа). Значение строковой константы в каждом случае применения оператора разное:

- "Введите значение переменной x";
- "Введите значение переменной y";
- "F = ";
- "Хотите запустить программу вновь? Y/N".

Все перечисленные константы являются операндами и заносятся в таблицу 3. Каждый из перечисленных операндов используется один раз.

Следующим по ходу выполнения программы выполняется оператор `Console.Write()`; вызова функции вывода символов на экран (таблица 1, п. 13, 16). Оператор используется 2 раза с разными операндами: "x="; "y=", каждый из которых используется в программе однократно. Оператор включается в таблицу операторов, операнды – в таблицу операндов.

Следующая операция `x = double.Parse(Console.ReadLine());`

Включает три оператора:

= – оператор присваивания (таблица 1, п. 14, 17, 19, 22, 24, 27) используется в программе 6 раз;

... `Parse()` – оператор вызова функции преобразования строки в заданный тип (таблица 1, п. 14, 17, 27) используется в программе 3 раза;

`Console.ReadLine()` – оператор вызова функции считывания строки с клавиатуры (таблица 1, п. 14, 17, 27) используется в программе 3 раза.

Оператор `if()...else...` (таблица 1, п. 18, 21) используется дважды в тексте программы для ветвления алгоритма.

Операции (таблица 1, п. 19, 22, 24):

`F = Math.Sin(x) + Math.Cos(y) * Math.Cos(y);`

`F = Math.Log(x);`

`F = Math.Sin(x) * Math.Sin(x) + Math.Cos(y);`

включают следующие ранее не рассмотренные операторы:

`Math.Sin(x)` - оператор вызова функции вычисления синуса, используется 3 раза;

`Math.Cos(y)` - оператор вызова функции вычисления косинуса, применяется 3 раза;

Math.Log(x) – оператор вызова функции вычисления логарифма, используется один раз.

Имена F, x и y являются операндами: F используется 5 раз, x – 8 раз, y – 7 раз. Символы «;», «,», «*» и «+», используемые в программе, обозначают следующие операции:

; – операция определения завершения оператора, используется 16 раз;

, – операция отделения элементов списка, используется 2 раза;

* – операция умножения, используется 2 раза;

+ – операция сложения (сцепления строк), используется 3 раза.

Символы «<» и «= =» используются для определения логических операций сравнения:

< – операция сравнения «меньше», используется 1 раз;

= = – операция сравнения «равно», используется 3 раза.

В позициях 23 и 24 таблицы 2 представлены символы, определяющие следующие операции:

{ } – операция начала и завершения блока инструкций, используется 4 раза;

() – операция начала и завершения списка параметров или условия, используется 22 раза.

Оставшиеся четыре позиции таблицы 2 содержат символы:

|| – операция логического сложения (дизъюнкция), используется один раз;

— || – операция определения строковых констант, используется 6 раз;

= ‘ – операция определения символьных констант, используется 2 раза;

. – операция связывания имен, используется 19 раз.

Проанализируем содержание таблицы 3. Позиции 1, 2, 3 и 4 содержат имена операндов System, Ex, Program, Main(), которые используются в программе по одному разу. Строковые константы (п. 8, 9, 10, 11, 13 и 16 таблицы 3):

"Введите значение переменной x"; "x=";

"Введите значение переменной y"; "y=";

"Хотите запустить программу вновь? Y/N";

"F = "используются в тексте программы однократно (таблица 3).

Символьные константы 'Y' и 'y' применяются также по одному разу (таблица 3). Имена переменных x, y, check и F повторяются в программе соответственно 8, 7, 4 и 5 раз.

Для рассматриваемой программы список входных и выходных параметров (таблица 4) не обладает большим разнообразием. Входными параметрами являются значения переменных:

x = double.Parse(Console.ReadLine());

y = double.Parse(Console.ReadLine());

check=char.Parse(Console.ReadLine()).

В таблице 4 представлены входные и выходные переменные программы.

Входные переменные	Выходные переменные
x	"Введите значение переменной x"
y	"x="
check	"Введите значение переменной y"
	"y= "
	"F ="
	"Хотите запустить программу вновь? Y/N"
	F

Выходными значениями являются шесть констант, для которых имена совпадают со значениями, и одна переменная F:

Console.WriteLine("Введите значение переменной x");

Console.Write("x=");

Console.WriteLine("Введите значение переменной y");

Console.Write("y=");

Console.WriteLine("F = "+F) – в этом случае два выходных параметра: строковая константа "F = " и переменная F;

Console.WriteLine("Хотите запустить программу вновь? Y/N").

Оценка характеристик программы

Используя сформированные таблицы с необходимыми параметрами для расчета и применяя соотношения Холстеда, вычислим характеристики рассматриваемой программы. Сведем все результаты расчетов метрик Холстеда в таблицу 5.

Наименование характеристики	Обозначение и формула для вычисления	Значение
Число простых (уникальных) операторов и операций	n_1	28
Число простых (уникальных) операндов	n_2	16
Общее число всех операторов и операций	N_1	116
Общее число всех операндов	N_2	36
Число входных и выходных переменных (параметров)	n_2^*	10
Словарь программы	$n = n_1 + n_2$	44
Длина реализации программы	$N = N_1 + N_2$	152
Длина программы	$\tilde{N} = n_1 \cdot \log_2 n_1 + n_2 \cdot \log_2 n_2$	198,68
Объем программы (в битах)	$V = N \log_2 n$	830
Потенциальный объем программы	$V^* = (n_2^* + 2) \log_2 (n_2^* + 2)$	43
Уровень реализации программы	$L = \frac{V^*}{V}$	0,052
Уровень реализации языка	$\lambda = LV^*$	2,23
Работа программирования	$E = V / L$	15960
Число переданных ошибок в программе	$B = \frac{LE}{3000}$	0,3

Выводы:

- 1) Длина реализации меньше длины программы, следовательно, несовершенства в программе отсутствуют.
- 2) Уровень реализации исследуемой программы весьма низкий, так как потенциальный объем программы в значительной степени меньше ее реального объема.
- 3) Возможности языка программирования использованы на достаточном уровне.

Задача 2. «Замена строк таблицы». Дана вещественная таблица размером $M \times N$ элементов. Размер таблицы вводится с клавиатуры. Поменять местами строки таблицы по правилу: строка с номером 0 меняется с последней, строка с номером 1 с предпоследней и т.д.

Пример проведения данной операции приведен ниже.

Таблица в исходном виде	Преобразованная таблица
1,0 2,0 3,0 4,0	9,0 10,0 11,0 12,0
5,0 6,0 7,0 8,0	5,0 6,0 7,0 8,0
9,0 10,0 11,0 12,0	1,0 2,0 3,0 4,0

Разработать программу и определить значения метрик Холстеда, на основе которых дать оценку качества разработанного исходного текста программы.

Таблица 1. Реализация программы для задачи «Замена строк таблицы» на языке C#.

Номера строк	Строки программы
1	using System;
2	namespace Exemp
3	{
4	class Replace
5	{
6	static void Прочитать(double[,] a, int n, int m)
7	{
8	int i, j;
9	for (i = 0; i < n; i++)
10	for (j = 0; j < m; j++)
11	{
12	Console.Write("Элемент[{0},{1}]: ", i, j);
13	a[i, j] = double.Parse(Console.ReadLine());
14	}
15	}
16	static void Вывести(double[,] a, int n, int m, string формат)
17	{
18	int i, j;
19	for (i = 0; i < n; i++, Console.WriteLine())
20	for (j = 0; j < m; j++)
21	Console.Write(формат, a[i, j]);
22	}

23	static void Переставить (double[,] a, int ном1, int ном2, int m)
24	{
25	int j;
26	double b;
27	for (j = 0; j < m; j++)
28	{
29	b = a[ном1, j];
30	a[ном1, j] = a[ном2, j];
31	a[ном2, j] = b;
32	}
33	}
34	static void Main()
35	{
36	int n, m, mp, ном1, ном2;
37	double[,] a;
38	ConsoleKeyInfo клавиша;
39	do
40	{
41	Console.Clear();
42	Console.WriteLine("Сколько строк:");
43	n = int.Parse(Console.ReadLine());
44	Console.WriteLine ("Сколько столбцов:");
45	m = int.Parse(Console.ReadLine());
46	a = new double[n, m];
47	Прочитать(a, n, m);
48	Console.WriteLine("\nИсходная таблица");
49	Вывести(a, n, m, "{0,8:f2}");
50	mp = n/2;
51	for(ном1=0,ном2=n-1; ном1 < mp; ном1++,ном2--)
52	Переставить(a, ном1, ном2 , m);
53	Console.WriteLine ("\nТаблица после перестановки строк");
54	Вывести(a, n, m, "{0,8:f2}");
55	Console.WriteLine ("\nДля выхода нажмите клавишу ESC");
56	клавиша = Console.ReadKey(true);
57	} while (клавиша.Key != ConsoleKey.Escape);
58	}
59	}
60	}

Таблица 2. Словарь операторов и операций программы

№ п/п	Операторы, операции	Номера строк	Количество повторений
1	using...;	1	1
2	namespace	2	1
3	class ...	4	1
4	static void...	6, 16, 23, 34	4
5	double [,]	6, 16, 23, 37	4
6	int...	6, 6, 8, 16, 16, 18, 23, 23, 23, 23, 25, 36	12
7	string...	16	1
8	ConsoleKeyInfo	38	1
9	for ()	9, 10, 19, 20, 27, 51	6
10	do {} while()	39	1
11	Console.Write()	12, 21, 42, 44	4
12	Console.WriteLine()	19, 48, 53, 55	4
13	Console.ReadLine()	13, 43, 45	3
14	...Parse()	13, 43, 45	3
15	Console.Clear()	41	1
16	Console.ReadKey()	56	1
17	new double[]	46	1
18	new int	41	1
19	Прочитать()	47	1
20	Вывести()	49, 54	2
21	Переставить()	52	1
22	.Key	57	1
23	ConsoleKey.Escape	57	1
24	;	1, 8, 9, 9, 10, 10, 12, 13, 18, 19, 19, 20, 20, 21, 25, 26, 27, 27, 29, 30, 31, 36, 37, 38, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 51, 52, 53, 54, 55, 56, 57	42
25	,	6, 6, 6, 8, 12, 12, 12, 16, 16, 16, 16, 18, 19, 21, 21, 23, 23, 23, 23, 29, 30, 30, 31, 36, 36, 36, 36, 37, 46, 47, 47, 49, 49, 49, 49, 51, 51, 52, 52, 52, 54, 54, 54, 54,	44
26	:	49, 54	2
27	/	50	17
28	=	9, 10, 13, 19, 20, 27, 26, 29, 30, 31, 43, 45, 46, 50, 51, 51, 56	1
29	-	51	17
30	++	9, 10, 19, 20, 26, 27, 51	1
31	--	51	1
32	{ }	3(60), 5(59), 7(15), 11(14), 17(22), 24(33), 28(32), 35(58), 40(57)	9
33	()	6, 9, 10, 12, 13, 13, 16, 19, 19, 20, 21, 23, 27, 34, 41, 42, 43, 43, 44, 45, 45, 47, 48, 49, 51, 52, 54, 55, 56, 57	30
34	[]	6, 13, 16, 23, 29, 30, 30, 31, 37, 46	10
35	""	12, 42, 44, 48, 49, 54, 55	7
36	/	45	1
37	.	12, 13, 13, 19, 21, 41, 42, 43, 43, 44, 45, 48, 53, 55, 56, 57, 57	17
38	<	9, 10, 19, 20, 27, 51	6
39	!=	57	1
Всего			252

Таблица 3. Словарь операндов программы

№ п/п	Операторы, операции	Номера строк	Количество повторений
1	System	1	1
2	Exemp	2	1
3	Replace	4	1
4	Прочитать	6	1
5	a	6, 13, 16, 21, 23, 29, 30, 30, 31, 37, 46, 47, 49, 54	14
6	n	6, 9, 16, 19, 36, 43, 46, 47, 49, 50, 51, 54	12
7	m	6, 10, 16, 20, 23, 27, 36, 45, 46, 47, 49, 54	12
8	i	8, 9, 9, 12, 13, 18, 19, 19, 21	9
9	j	8, 10, 10, 12, 13, 18, 20, 20, 25, 27, 27, 29, 30, 31	14
10	"Элемент[{0},{1}]:"	12	1
11	формат	16, 21	2
12	ном1	23, 29, 30, 51, 51, 51, 52	7
13	ном2	23, 30, 31, 36	4
14	b	26, 29, 31	3
15	Main	34	1
16	mp	36, 51	2
17	клавиша	38, 56, 57	3
18	"Сколько строк: "	42	1
19	"Сколько столбцов: "	44	1
20	"\nИсходная таблица"	48	1
21	"{0,8:f2}"	49, 54	1
22	"\nТаблица после перестановки строк"	53	1
23	"\nДля выхода нажмите клавишу ESC"	55	1
24	true	56	1
Всего			95

Таблица 4. Входные и выходные переменные программы

Входные переменные	Выходные переменные
a	"Элемент[{0},{1}]:"
n	формат
m	"Сколько строк:"
клавиша	a
	"Сколько столбцов:"
	"\nИсходная таблица"
	"\nТаблица после перестановки строк"
	"\nДля выхода нажмите клавишу ESC"

Таблица 5. Значения метрик Холстеда для программы

Наименование характеристики	Обозначение и формула для вычисления	Значение
Число простых (уникальных) операторов и операций	n_1	39
Число простых (уникальных) операндов	n_2	24
Общее число всех операторов и операций	N_1	252
Общее число, всех операндов	N_2	95
Общее число входных и выходных переменных (параметров)	n_2^*	12
Словарь программы	$n = n_1 + n_2$	63
Длина реализации	$N = N_1 + N_2$	347
Длина программы	$N = n_1 \cdot \log_2 n_1 + n_2 \cdot \log_2 n_2$	316
Объем программы (в битах)	$V = (N_2 + N_1) \cdot \log_2(n_1 + n_2)$	2074
Потенциальный объем программы	$V^* = (n_2 + 2) \cdot \log_2(n_2 + 2)$	107
Уровень реализации программы	$L = V^* / V$	0,051
Уровень языка	$\lambda = L \cdot V^*$	5,48
Работа по программированию	$E = V / L$	40354

Выводы:

1) Длина реализации больше расчетной длины программы на 9%. В программе присутствуют несовершенства. Проявлением несовершенства программы являются строки:

- $b = a[\text{ном1}, j];$
- $a[\text{ном1}, j] = a[\text{ном2}, j];$
- $a[\text{ном2}, j] = b.$

Представленная последовательность присваиваний очень близка к неоднозначности операндов.

2) Уровень реализации исследуемой программы весьма низкий, так как потенциальный объем программы в значительной степени меньше ее реального объема.

3) Возможности языка программирования использованы на низком уровне.

Задание 3. В следующих задачах, предлагаемых на выбор, необходимо выполнить следующее:

- разработать программу, реализующую заданный алгоритм;
- сформировать словарь программы, охватывающий операнды, операторы и операции;
- словари оформить в виде таблиц;
- рассчитать метрики Холстеда, оформив результат в виде итоговой таблицы;
- провести анализ полученных результатов, сформировав содержательные выводы содержащие оценку качества разработанного текста программы:

1) оценить уровень программы, сравнив потенциальный и реальный объемы;

- 2) выявить наличие несовершенств на основе сравнения длины реализации и расчетной длины программы, указать строки программы, содержащие несовершенства и тип несовершенств;
- 3) оценить уровень использования возможностей языка реализации.

Варианты заданий:

1. Написать и протестировать функцию, которая «переворачивает» строку, передаваемую ей в качестве параметра, в зеркальное состояние.
2. Дано натуральное число N. Вывести на экран число, которое получится после выписывания цифр числа N в обратном порядке. Для получения нового числа составить функцию.
3. Написать и протестировать функцию, подсчитывающую количество минимальных элементов в каждой строке целочисленной матрицы.
4. Написать и протестировать функцию, которая подсчитывает, сколько раз в заданной строке встретился указанный символ.
5. Написать и протестировать функцию, подсчитывающую количество положительных элементов в массиве.

ПРАКТИЧЕСКАЯ РАБОТА № 39

Тема: Оценка программных средств с помощью метрик

Цель работы: научить оценивать ПО с помощью метрик, применяя разные методы.

Оборудование: ПК, программное обеспечение – MS Word, инструкции по выполнению работы.

Справочный материал:

Оценка программного средства на основе метрик Чепина.

Все множество переменных, составляющих список ввода-вывода, разбивается на четыре функциональные группы:

- Р – вводимые переменные для расчетов и для обеспечения вывода. Примером такой переменной может служить используемая в программах лексического анализатора переменная, содержащая строку исходного текста программы – в этом случае сама переменная не модифицируется, а применяется только как контейнер для исходной информации;
- М – модифицируемые, или создаваемые внутри программы, переменные. К таким переменным относится большинство традиционно применяемых переменных, декларируемых в программных модулях;
- С – переменные, участвующие в управлении работой программного модуля (управляющие переменные). Такие переменные предназначены для передачи управления, изменения логики вычислительных процессов и т. д.;
- Т – не используемые в программе (так называемые паразитные) переменные. Такие переменные не принимают непосредственного участия в реализации процесса обработки информации, ради которого написана анализируемая программа, однако они заявлены в программном модуле. Такие переменные могут использоваться для выполнения промежуточных действий и не играют принципиальной роли в решении основной задачи.

В качестве особенности применения метрики следует назвать необходимость учета каждой переменной в каждой функциональной группе, поскольку каждая переменная может выполнять одновременно несколько функций.

Исходное выражение для определения метрики Чепина записывается в следующем виде: $Q = P + 2M + 3C + 0,5T$.

Следует отметить, что метрика сложности программы Чепина также основана на анализе исходных текстов программ, что обеспечивает единый подход к автоматизации их расчета и может быть рассчитана с использованием специально разработанного программного анализатора.

Содержание работы:

Задание 1. Простые числа в матрице. Дана целочисленная матрица размером $N \times M$. Вычислить и записать в одномерный массив количество простых чисел в каждом столбце матрицы. Размерность матрицы задается с клавиатуры, заполнение матрицы осуществляется посредством датчика случайных чисел. Разработать программу для решения задачи. На основе лексического анализа исходного текста программы определить значение метрики Чепина.

Реализация программы

Текст программы для реализации возможного алгоритма решения поставленной задачи представлен на языке C# (таблица 1). Т

Таблица 1. Реализация программы для задачи «Простые числа в матрице»

Номера строк	Строки программы
1	using System;
2	namespace EX2
3	{
4	class Program
5	{
6	static void Main()
7	{
8	int n,m
9	
10	int[,] a;
11	int[] b;
12	ConsoleKeyInfo клавиша;
13	int i,j,k,d;
14	bool p=false;
15	Random g;
16	do
17	{
18	Console.Clear();
19	Console.Write("Сколько строк: ");
20	n = int.Parse(Console.ReadLine());
21	Console.Write("Сколько столбцов: ");
22	m = int.Parse(Console.ReadLine());
23	g=new Random();
24	a = new int[n, m];
25	for (i = 0; i < n; i++)
26	for (j = 0; j < m; j++)
27	{
28	a[i, j] = int.Parse(g.Next(0,101));
29	}
30	
31	Console.WriteLine("\nИсходная матрица");
32	for (i = 0; i < n; i++, Console.WriteLine())
33	for (j = 0; j < m; j++)
34	Console.Write("{0,8:d}", a[i, j]);
35	b = new int[m];
36	
37	for (j = 0; j < m; j++)
38	{
39	for (i = k = 0; i < n; i++)
40	{
41	p = true;
42	for (d = 2; d < a[i,j]; d++)
43	if (a[i,j] % d == 0) p = false;

44	
45	
46	if (p) k++;
47	b[j] = k;
48	}
49	}
50	
51	
52	Console.WriteLine("\nКоличество простых");
53	for (i = 0; i < b.Length; i++)
54	Console.Write("{0,8:d}", b[i]);
55	
56	Console.WriteLine("\nДля выхода нажмите клавишу ESC");
57	клавиша = Console.ReadKey(true);
58	} while (клавиша.Key != ConsoleKey.Escape);
59	
60	}
61	}
62	}

Оценка характеристик программы

Рассмотрим текст программы для оценки ее качества с помощью метрики Чепина, которая позволяет оценить меру трудности понимания программы на основе входных и выходных данных. Результат анализа объявленных переменных представлен в таблице 2

Таблица 2. Результат анализа объявленных переменных

№ п/п	Наименование переменных	Номера строк
<i>P</i> (для расчётов и обеспечения вывода)		
1	<i>m</i>	8
2	<i>n</i>	8
3	<i>a</i>	10
<i>M</i> (модифицируемые или создаваемые)		
1	<i>i</i>	13
2	<i>j</i>	13
3	<i>k</i>	13
4	<i>d</i>	13
5	<i>b</i>	11
<i>C</i> (управляющие переменные)		
1	<i>g</i>	15
2	<i>p</i>	14
3	<i>клавиша</i>	12
<i>T</i> (не используемые в программе)		
	Отсутствуют	

Переменные *m*, *n* и *a* используются в качестве исходных данных. Переменные *i*, *j*, *k*, *d* и *b* в процессе выполнения программы создаются и модифицируются. Переменные *g*, *p* и *клавиша* используются для управления выполнением программы.

Таким образом, исходя из результатов анализа исходного текста программы, получаем следующие значения характеристик: $P = 3$, $M = 5$, $C = 3$, $T = 0$

Метрика Чепина: $Q = P + 2M + 3C + 0,5T = 3 + 2 \cdot 5 + 3 \cdot 3 + 0,5 \cdot 0 = 22$

Выводы. На основе полученных значений метрики Чепина уровень сложности данного решения можно считать сравнительно низким, как так в исходном тексте программы используется незначительное количество переменных, что не затрудняет понимание программы.

Задача 2. Заправка топливных баков. Определить класс «Бак», описывающий понятие «Топливный бак». При этом должны быть использованы следующие поля: ширина, длина и высота в сантиметрах; вид топлива. Следует учитывать операции (методы):

- полная заправка бака заданным видом топлива;
- вычисление стоимости полной заправки бака.

Бак в общем случае имеет вид параллелепипеда. Стандартным считается бак, имеющий вид куба (ширина, длина и высота совпадают).

Возможные разновидности баков:

- бак с одинаковой шириной и длиной, но с индивидуальной высотой;
- бак с индивидуальными размерами по ширине, длине и высоте.

Стоимость одного кубического сантиметра топлива определяется полями класса «Топливо». Стоимость топлива в рамках решения задачи неизменна.

Выдача стоимости топлива реализуется специальной операцией этого класса.

Заполнить и вывести на экран стоимость заправки четырех баков:

- 10x20x30 см: топливо–газ;
- 10x10x20 см: топливо–керосин;
- 10x10x10 см: топливо–бензин;
- 20x20x20 см: топливо–бензин

Реализация программы

Текст программы для реализации возможного алгоритма решения поставленной задачи представлен на языке программирования C#.

Номера строк	Строки программы
1	using System;
2	namespace EX2_1
3	{
4	class Топливо
5	{
6	private static double ценаГаз = 0.05
7	private static double ценаГаз = 0.05
8	private static double ценаБензии = 0.1;;
9	public static double L[eHa(string вид)
10	{

11	switch (вид)
12	{
13	case "газ": return ценаГаз;
14	case "керосин": return ценаКеросин;
15	case "бензин": return ценаБензин;
16	default: return 0.0
17	}
18	}
19	}
20	class Бак
21	{
22	private double x, y, h;
23	private string видТоплива;
24	public void Заполнить(double xb, string вид)
25	{
26	x = xb; y = xb; h = xb; видТоплива = вид;
27	{
28	public void Заполнить(ref double xb, string вид)
29	{
30	x = xb; y = xb; h = xb; видТоплива = вид;
31	}
32	public void Заполнить(double xb, double yb, string вид)
33	}
34	x = xb; y = yb; h = xb; видТоплива = вид;
35	}
36	public void Заполнить(double xb, double yb, double hb, string вид)
37	{
38	x = xb; y = yb; h = hb; видТоплива = вид;
39	}
40	public double Оплата()
41	{
42	return x * y * h * Топливо.Цена(видТоплива);
43	}
44	}
45	class Program
46	{
47	static void Main(string[] args)
48	{
49	double x = 20.0;
50	Бак b;
51	b = new Бак();
52	b.Заполнить(10.0,20.0,30.0, "газ");
53	Console.WriteLine("Стоимость заправки: " + b.Оплата());
54	b.Заполнить(10.0, 20.0,"керосин");
55	Console.WriteLine("Стоимость заправки: " + b.Оплата());
56	b.Заполнить(10.0,"бензин");
57	Console.WriteLine("Стоимость заправки:" + b.Оплата());
58	b.Заполнить(ref x, "бензин");
59	Console.WriteLine("Стоимость заправки:" + b.Оплата());
60	Console.ReadKey();
61	}
62	}
63	}

Оценка характеристик программы

Необходимо отметить, что в исходном тексте программы используются одноименные программные модули и имена переменных входных параметров этих модулей (строки 24, 28, 32 и 36). Несмотря на совпадение имен, эти переменные являются различными элементами программы, так как принадлежат различным программным модулям.

На основе анализа исходного текста составим таблицу. Для одноименных переменных введем дополнительный столбец.

№ п/п	Наименования переменных	Номера строк	Количество переменных
Р (для расчетов и для обеспечения вывода)			
1	xb	24,28, 32, 36	4
2	yb	32, 36,	2
3	hb	36	1
М (модифицируемые или создаваемые)			
1	ценаГаз	6	1
2	ценаКеросин	7	1
3	ценаБензин	8	1
4	x	22, 50	2
5	y	22	1
6	h	22	1
7	видТоплива	23	1
8	b	51	1
С (управляющие переменные)			
1	вид	9	1
Т (не используемые в программе)			
Отсутствуют			

Исходя из полученных на основе анализа данных имеем: $P = 7$, $M = 9$, $C = 1$, $T = 0$ $Q = 7 + 2 \cdot 9 + 3 \cdot 1 + 0,5 \cdot 0 = 28$

Выводы. Уровень сложности задачи является достаточно высоким, так как реализация задачи имеет многомодульную схему, и, как следствие, используется большое количество переменных для расчета ($P = 7$) и модифицируемых переменных ($M = 9$).

Задача 3. Расчет платежей за электроэнергию. Предметная область данной задачи - коммунальные платежи за электроснабжение. Определить класс, описывающий операцию «Платеж за электроэнергию» с использованием следующих полей:

- фамилия плательщика;
- потребление электроэнергии за оплачиваемый месяц;
- нормативное среднеемесячное потребление;
- тариф (стоимость одного киловатт-часа).

При решении задачи следует учитывать методы:

- вычисление суммы оплаты;
- формирование сводной информации по одному платежу (вид платежа, фамилия, сумма, потребление);

Платеж может выполняться по показаниям счетчика или по нормативно установленному среднемесячному потреблению. В процессе эксплуатации программы тариф и нормативно установленное среднемесячное потребление не изменяются. Для создания конкретного платежа необходимо предусмотреть соответствующий конструктор. Все платежи сохраняются в архиве.

Запросы по ведению архива выполняются статическими методами класса «Запрос»: занесение платежей в архив. Данные платежа вводятся с клавиатуры. Ввод отрицательного показания счетчика означает оплату по нормативно установленному среднемесячному потреблению. Вывод сводной информации проводится из архива. Архив моделируется массивом объектов. В основном классе «Платежи» следует сформировать архив платежей. По данным архива выдать сводную информацию о платежах.

ПРАКТИЧЕСКАЯ РАБОТА № 40

Тема: Оценка программных средств с помощью метрик

Цель работы: научить оценивать ПО с помощью метрик, применяя разные методы.

Оборудование: ПК, программное обеспечение – MS Word, инструкции по выполнению работы.

Справочный материал:

Оценка программного средства на основе метрик Джилба

В качестве меры логической трудности Джилб предложил число логических «двоичных принятий решений». Наиболее ценным для практики является то, что такая оценка может быть получена вручную на основе зрительного анализа текста программы либо автоматически с помощью специально разработанных программных анализаторов, причем относительно несложных.

Логическая сложность программы определяется как насыщенность программы условными операторами и операторами цикла. Вводятся следующие характеристики программного средства:

- CL – абсолютная сложность программы, характеризуемая количеством операторов условий;
- cl – относительная сложность программы, определяющая насыщенность программы операторами условия (вычисляется как отношение абсолютной сложности CL к общему числу операторов L).
- количество операторов цикла L_{loop} ;
- количество операторов условия L_{IF} ;
- число модулей или подсистем L_{mod} ;

$$f = \frac{N_{SV}^4}{L_{mod}}$$

- отношение числа связей между модулями к числу модулей
- отношение числа ненормальных выходов из множества операторов к

общему числу операторов $f^* = \frac{N_{SV}^*}{L}$.

- надежность программы (возможность того, что данная программа проработает определенный период времени без логических сбоев), равную единице минус отношение числа логических сбоев к общему числу запусков;
- мера точности (свободы от ошибок) - отношение количества правильных данных ко всей совокупности данных;
- прецизионность (мера того, насколько часто появляются ошибки, вызванные одинаковыми причинами) – отношение числа фактических ошибок на входе к общему числу наблюдаемых, ошибок, причинами которых явились эти ошибки на входе.

Содержание работы:

Задание 1. Функция копирования элементов массива. Необходимо разработать функцию, которая копирует положительные элементы из одного одномерного целочисленного массива в другой массив. Используя этот

метод, следует выполнить копирование положительных элементов двух исходных массивов А и В в массив С. Размер исходных массивов А и В ввести с клавиатуры. Эти массивы заполнить случайными числами из диапазона от –100 до 300. Сформированный массив С вывести на экран. Выдать сообщение на экран в случае, когда массив С оказывается пустым. Для разработанной программы на основе лексического анализа исходного текста определить значения метрик Джилба.

Реализация программы.

Рассмотрим вариант реализации возможного алгоритма решения поставленной задачи, разработанный с использованием языка программирования С#, в котором применяются программные модули.

Пример реализации такой программы приведен в таблице 1

Номера строк	Строки программы
1	using System;
2	class Exempl
3	{
4	public static int[] Copy(int[] a)
5	{
6	int [] b = new int[a.Length];
7	int j=0;
8	for(int i=0; i<a.Length; i++)
9	{
10	if(a[i]>=0) {b[j]=a[i];j++;}
11	}
12	return b;
13	}
14	
15	public static void Zapoln(ref int[] a, Random g)
16	{
17	for (int i = 0; i < a.Length; i++)
18	{
19	a[i] = g.Next(-100, 300);
20	}
21	}
22	
23	public static void Print(int[] a, string str, string strl)
24	{
25	Console.WriteLine(strl);
26	for (int i = 0; i < a.Length; i++)
27	{
28	if(a[i]!=0) Console.Write(str, a[i]);
29	}
30	Console. WriteLine();
31	}
32	
33	public static void Main()
34	{
35	int[] a, b, c, p;
36	Random g = new Random();
37	char r;
38	do
39	{
40	Console.Clear();

41	Console.WriteLine("Определите размер первого массива!");
42	a = new int[int.Parse(Console.ReadLine())];
43	Console.WriteLine("Определите размер второго массива!");
44	b = new int[int.Parse(Console.ReadLine())];
45	c = new int[a.Length + b.Length];
46	Exempl.Zapoln(ref a, g);
47	Exempl.Zapoln(ref b, g);
48	Exempl.Print(a, " {0,5}", "Первый исходный массив!!!");
49	Exempl.Print(b, " {0,5}", "Второй исходный массив!!!");
50	p = Exempl.Copy(a);
51	Array.Copy(p, 0, c, 0, a.Length);
52	p = Exempl.Copy(b);
53	Array.Copy(p, 0, c, a.Length, b.Length);
54	Exempl.Print(c, " {0,5}", "Результатный массив!!!");
55	Console.WriteLine();
56	Console.WriteLine("Выполнить повтор программы? Y/N");
57	r = char.Parse(Console.ReadLine());
58	} while (r == 'Y' r == 'y');
59	}
60	}

В 6-й, 42-й, 44-й и 50-й строках (таблица 1) ключевые слова `int[...]` представляют собой операторы вызова метода конструктора. Ключевое слово `Random` в 36-й строке используется дважды: в первом случае это оператор описания типа, во втором – вызов метода-конструктора. В таблице 2 приведены операторы и операции, используемые в программе.

№п/п	Операторы, операции	Номера строк	Количество повторений
1	using...	1	1
2	class...	2	1
3	public static...	4, 15, 23, 33	4
4	int[]	4, 6, 15, 23, 35	5
5	new	6, 36, 42, 44, 45	5
6	int	7, 8, 15, 17, 26,	5
7	string	23, 23	2
8	char	37	1
9	Random	15, 36	2
10	.Length	17, 6, 8, 26, 45, 45, 51, 53	8
11	.Next	19	1
12	Console.WriteLine()	25, 30, 41, 43, 55, 56	6
13	Console.Write()	28,	1
22	Console.ReadLine()	42, 44, 57	3
15	for()	8, 17, 26	3
16	do{}while()	38-58	1
17	if()	10, 28	1
18	Console.Clear()	40	1
19	Exempl.Zapoln()	46, 47	2
20	.Parse()	42, 44, 57	3
21	Exempl.Print()	48, 49, 54	3
22	Exempl.Copy()	50, 52	2
23	Array.Copy()	51, 53	2
24	=	6, 7, 8, 10, 17, 19, 26, 36, 42, 44, 45, 50, 52, 57	14
25	>=	10	1

26	!=	28	1
27	= =	58, 58	2
28	<	8, 17, 26	3
29	++	8, 10, 17, 26	4
30	return	12	1
31	[]	4, 4, 6, 6, 10, 10, 10, 15, 19, 23, 28, 28, 35, 42, 44, 45,	16
32	()	4, 8, 10, 15, 17, 19, 23, 25, 26, 28, 28, 30, 33, 36, 40, 41, 42, 42, 43, 44, 44, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 57, 58	35
33	{}	3(60), 5(13), 9(11), 16(21), 18(20), 24(31), 27(29), 34(59), 39(58), 10, 48, 49, 54.	13
34	,	15, 19, 23, 23, 28, 35, 35, 35, 46, 47, 48, 48, 49, 49, 51, 51, 51, 51, 53, 53, 53, 53, 54, 54, 54	25
35	;	1, 6, 7, 8, 8, 10, 10, 12, 17, 17, 19, 25, 26, 26, 28, 30, 35, 36, 37, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58	38
36	.	6, 8, 17, 19, 26, 28, 30, 40, 41, 42, 42, 43, 44, 44, 45, 45, 46, 47, 48, 49, 50, 51, 51, 52, 53, 53, 53, 54, 55, 56, 57, 57	32
37	int[...]	6, 42, 44, 45	4
38	Random()	36	1
39	“ ”	41, 43, 48, 48, 49, 49, 54, 56	8
40	‘ ’	58, 58	2
Всего			263

Оценка характеристик программы

Число операторов условий LIF равно 2 (таблица 2, п. 17); Значение характеристики L_{loop} определяется количеством используемых в программе циклов. В исходном тексте данной программы содержится 4 цикла: 3 оператора for и 1 do... while (таблица 2, п. 15 и 16). Значение характеристики L_{mod} определяется количеством используемых программных модулей в решении.

В представленном решении используется четыре программных модуля, каждый из которых определяется следующими строками:

- Public static void Main() (таблица 1, строка 33);
- Public static int[] Copy(int[] a) (таблица 1, строка 4);
- Public static void Zapoln(ref int[] a, Random g) (таблица 1, строка 15);
- Public static void Print(int[] a, string str, string str1) (таблица 1, строка 23).

Общее количество операторов условия 6, из них 2 оператора if и четыре оператора цикла. Общее число всех используемых операторов $L = 263$ (таблица 2). Таким образом:

- $CL = 6$ – абсолютная сложность программы;
- $cl = CL / L = 6 / 263 = 0,0228$.

Количество связей между NSV модулями равно трем – по одной связи между основным и каждым из дополнительных модулей. Отношение числа связей к числу модулей определяется следующим образом:

$$f = \frac{N_{SV}^4}{L_{mod}} = \frac{3^4}{4} = \frac{81}{4} = 20,25$$

Из полученных результатов анализа текста программы следует, что исходный код имеет невысокую сложность, так как на 263 оператора текста приходится всего лишь 6 операторов условий. Общее число программных модулей решения также невелико (4 модуля), что подтверждает низкий уровень сложности программы.

Задание 2. В массивах А и В хранятся целые числа. Массивы заполняются исходными данными от датчика случайных чисел в диапазоне значений от 10 до 50. Добавить в конец массива А все четные по значению элементы массива В. Вывести на экран следующие элементы:

- исходный массив А;
- исходный массив В;
- модифицированный массив А.

На основе лексического анализа исходного текста программы определить значения метрик Джилба.

Задание 3. Необходимо разработать программу, реализующую алгоритм решения по приведенному условию, а затем оценить характеристики разработанной программы на основе лексического анализа текста и применения метрик Джилба.

1. Определить число, образованное k старшими цифрами введенного с клавиатуры натурального числа. Исходное число и значение k вводятся с клавиатуры. Пример: для числа 456771 и k=2 искомое число равно 45
2. Вывести на экран таблицу квадратов первых десяти целых положительных чисел.

ПРАКТИЧЕСКАЯ РАБОТА № 41

Тема: Инспекция программного кода на предмет соответствия стандартам кодирования

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

Оборудование: ПК, программное обеспечение –MS Word, инструкции по выполнению работы.

Справочный материал:

Инспекция в целом

- Инспекция - (от лат. inspectio - осмотр) - орган, осуществляющий контроль за соблюдением установленных государством правил.
- Инспекция - орган управления, призванный следить за выполнением установленных правил и совмещающий контрольные функции с определенными административными правами. В их задачи входит также принятие на месте мер к исправлению недостатков.
- Программный код программы - это текст, набор команд, выполненный на особом языке программирования, понятном машине.
- Код программы необходим в первую очередь для написания и редактирования его человеком. Код программы также называют исходным кодом или исходным текстом программы.
- Стандарт кодирования — набор правил и соглашений, которые описывают базовые принципы оформления программного кода, используемого совместно группой разработчиков.
- Цель использования стандарта — упрощение восприятия программного кода человеком, сокращение нагрузки на память и зрение при чтении программы.

Некоторые из стандартов кодирования приведены ниже:

1 Ограниченное использование глобалов: Эти правила говорят о том, какие типы данных могут быть объявлены глобальными, а какие нет.

2 Стандартные заголовки для разных модулей: Для лучшего понимания и обслуживания кода заголовков различных модулей должен соответствовать стандартному формату и информации. Формат заголовка должен содержать ниже вещи, которые используются в различных компаниях:

- 1.Наименование модуля
- 2.Дата создания модуля
- 3.Автор модуля
- 4.История изменений
- 5.Краткое описание модуля о том, что делает модуль
- 6.В модуле поддерживаются различные функции, а также их входные и выходные параметры
- 7.Глобальные переменные, доступные или измененные модулем

3 Соглашения об именах для локальных переменных, глобальных переменных, констант и функций: Некоторые из соглашений об именах приведены ниже:

1. Содержательное и понятное название переменных помогает любому понять причину его использования.
2. Локальные переменные должны быть названы с использованием букв верблюда, начинающихся с маленькой буквы (например, `localData`), тогда как имена глобальных переменных должны начинаться с заглавной буквы (например, `GlobalData`). Имена констант должны быть сформированы только заглавными буквами (например, `CONSDATA`).
3. Лучше избегать использования цифр в именах переменных.
4. Названия функции должны быть написаны в верблюжьем регистре, начиная с маленьких букв.
5. Название функции должно четко и кратко описывать причину ее использования.

4 Отступ: Правильный отступ очень важен для улучшения читабельности кода. Чтобы сделать код читабельным, программисты должны правильно использовать пробелы. Некоторые из интервалов даны ниже:

1. После запятой между аргументами функции должен быть пробел.
2. Каждый вложенный блок должен иметь правильные отступы и интервалы.
3. Надлежащий отступ должен быть в начале и в конце каждого кадра в программе.
4. Все фигурные скобки должны начинаться с новой строки, а код, следующий за окончанием фигурных скобок, также начинается с новой строки.

5 Возвращаемые значения ошибок и соглашения об обработке исключений: Все функции, которые сталкиваются с ошибкой, должны возвращать 0 или 1 для упрощения отладки. С другой стороны, руководящие принципы кодирования дают некоторые общие рекомендации относительно стиля кодирования, которому необходимо следовать для улучшения понятности и читабельности кода. Некоторые из руководств по кодированию приведены ниже

6 Избегайте использования стиля кодирования, который слишком сложен для понимания: Код должен быть легко понятным. Сложный код делает обслуживание и отладку трудной и дорогой.

7 Избегайте использования идентификатора для нескольких целей: Каждой переменной должно быть дано описательное и осмысленное имя, указывающее причину ее использования. Это невозможно, если идентификатор используется для нескольких целей, что может привести к путанице у читателя. Более того, это приводит к большим трудностям при будущих улучшениях.

8 Код должен быть хорошо документирован: Код должен быть правильно прокомментирован для понимания. Комментарии относительно утверждений повышают понятность кода.

9 Длина функций не должна быть очень большой: Длинные функции очень трудно понять. Вот почему функции должны быть достаточно маленькими, чтобы выполнять небольшую работу, а длинные функции должны быть разбиты на маленькие для выполнения небольших задач.

10 Старайтесь не использовать оператор GOTO: Оператор GOTO делает программу неструктурированной, что снижает ее понятность и затрудняет отладку.

Цель инспекции программного кода: обнаружение и исправление ошибок, которые были пропущены, остались незамеченными при разработке. Результат инспекции, как правило, – улучшение качества ПО и навыки разработчика.

Задание. Переписать программный код, используя общепринятые соглашения и рекомендации по именованию и форматированию переменных, операторов, выражений.

Требования:

- добавить комментарии в программный код;
- проверить правильность именования переменных, констант, методов;
- проверить правильность объявления переменных и констант.

Было

Класс Main

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 blocks = new ArrayList <> (); public static

ArrayList bullets = new ArrayList <> ();

public static ArrayList врагаBullets = новый ArrayList <> (); public static

ArrayList враги = новый ArrayList <> (); static HashMap 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); переменная;
}    }
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.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 117, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 127, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 148, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 161, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 171, 200));

```

враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 185, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 204, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 215, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 228, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 233, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 243, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 252, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 262, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 277, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 280, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 286, 200));
перемена;

Дело 1:

враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 57, 200));
враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 67, 200));
враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 74, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 87, 200));
враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 104, 150));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 117, 200));
враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 133, 200));
враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 156, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 177, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 193, 200));
враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 201, 200));
враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 216, 200));
враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 224, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 246, 200));
враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 260, 200));
враги.аdd (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      }
если (смещение <= 100) level.setBackground () 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

Класс Main

```

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;
import 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;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;

```

```

публичный класс Main расширяет приложение {
public static ArrayList blocks = new ArrayList <> ();
public static ArrayList bullets = new ArrayList <> ();
public static ArrayList врагаBullets = новый ArrayList <> ();
public static ArrayList враги = новый ArrayList <> ();
static HashMap 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.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.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 117, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 127, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 148, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 161, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 171, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 185, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 204, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 215, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 228, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 233, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 243, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 252, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 262, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 277, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 280, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 286, 200));
перемена;
случай 1:

```



```

враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 57, 200));
враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 67, 200));
враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 74, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 87, 200));
враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 104, 150));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 117, 200));
враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 133, 200));
враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 156, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 177, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 193, 200));
враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 201, 200));
враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 216, 200));
враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 224, 200));
враги.аdd (новый EnemyComstock (Level_data.BLOCK_SIZE * 246, 200));
враги.аdd (новый EnemyRedEye (Level_data.BLOCK_SIZE * 260, 200));
враги.аdd (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);
}
}

```

Задание 2. В соответствии с индивидуальным вариантом, переписать программный код, используя общепринятые соглашения и рекомендации по именованию и форматированию переменных, операторов, выражений. Провести инспекцию кода с помощью любого из перечисленных сервисов. Сравнить полученные результаты. Сделать вывод.

Варианты заданий:

Вариант №1.

```
main()
{float tr[4][2]; float xe;float ye; float xa, ya, xb, yb,xc, yc, xd, yd; float a, b, c, d, f, h; int i;clrscr();printf("Введите координаты точек параллелограмма"); for (i=0; i<4; i++)
{printf("\nкоординаты: %d-й точки: x=",i+1); scanf("%f", &tr[i][0]); printf("y="); scanf("%f", &tr[i][1]);
} xa=tr[0][0]; ya=tr[0][1]; xb=tr[1][0]; yb=tr[1][1]; xc=tr[2][0]; yc=tr[2][1]; xd=tr[3][0]; yd=tr[3][1];
printf("%f, %f\n %f %f\n %f %f\n %f %f\n",xa,ya,xb,yb,xc,yc,xd,yd); a=sqrt((xa-xb)*(xa-xb)+(ya-yb)*(ya-yb));
b=sqrt((xb-xc)*(xb-xc)+(yb-yc)*(yb-yc)); c=sqrt((xc-xd)*(xc-xd)+(yc-yd)*(yc-yd));
d=sqrt((xd-xa)*(xd-xa)+(yd-ya)*(yd-ya)); f=sqrt((xa-xc)*(xa-xc)+(ya-yc)*(ya-yc)); h=sqrt((xb-xd)*(xb-xd)+(yb-yd)*(yb-yd));
if((a+b>f)&&(b+c>h)&&(d+c>f)&&(a+d>h)) { xe=(xa+xc)/2; ye=(ya+yc)/2;
printf("xe=%f, ye=%f\n",xe,ye); } else printf("Данный параллелограмм ((%f, %f),(%f, %f),(%f, %f),(%f, %f) вырожденный\n",xa,ya,xb,yb,xc,yc,xd,yd); getch(); }
```

Вариант №2.

```
main() { int n, m, r; float a[50][50]; float y[50]; float b; float s; int i,j,k; clrscr(); printf("vvedite dannie matrici"); scanf(" %d %d", &m, &n); while( m<=1 && n<=1 ) { printf("Vi owiblis, povtorite vvod");
scanf(" %d %d", &m, &n); printf("n=", "m="); } for(i=0; i<n; i++) { printf("vvesti koord vekt"); scanf("%f", &y[i]); }
for(i=0; i<m; i++) for(j=0; j<n; j++) a[i][j]=pow(y[j], i); for(i=0; i<m; i++) for(j=0; j<n; j++) printf("a[%d,%d]=%f\n", i,j,a[i][j]); }
```

Вариант №3.

```
#define L 100 main() { clrscr(); char s1[L]; char s2[L]; char seps[] = " ,\t\n"; char *token; printf("Enter the sentence\n"); gets(s1); token=strtok(s1,seps); while (token !=NULL) { if (strstr(token, "ать") || strstr(token, "ять") || strstr(token, "уть") || strstr(token, "ють")) {strcat(s2, "не"); strcat(s2, token);} else strcat(s2, token); strcat(s2, " "); token=strtok(NULL,seps); } printf("%s\n", s2); getch(); }
```

Вариант №4.

```
#define PI float form(int k, float, float, float); main() { float x; int k=1; float e; float s=0; printf("Enter the X:\n"); scanf("%f", &x); while (x<-0 || x>PI) { printf("Error\n"); scanf("%f", &x); } printf("\nEnter the e\n"); scanf("%f", &e); while (e<0) { printf("Error"); scanf("%f", &e); } printf("%f \n", form(k,s,x,e)); getch(); }
float form(int k, float y, float x, float e) { float a; a=cos((2*k-1)*x)/(2*k-1) ; y=y+a; printf("%f\n",y); if (a<e) return(y); else return(form(k+1,y, x, e)); }
```

Вариант №5.

```
main() { float min=999999; struct icx { char kultura[20]; float p38; float p57; float z38; float z57; } a; struct abc { char kultura[20]; float izmen; } b; FILE *filein; clrscr(); if((filein=fopen("TABL1.txt","r"))==NULL) {printf("owibka\n");exit(-1);}
while(fscanf(filein,"%s%f%f%f%f",&a.kultura,&a.p38,&a.p57,&a.z38,&a.z57)!=EOF) { if(a.z57-a.z38<min) { min=a.z57-a.z38; strcpy(b.kultura, a.kultura); b.izmen = a.p57/a.p38*100 - 100; } printf("|%15s| %10.2f| %10.2f| %10.2f| %10.2f\n",a.kultura, a.p38, a.p57, a.z38, a.z57); getch(); }
printf("Минимальное увеличение сбора культуры:\n"); printf("| Kultura |Izmenenie\n"); printf("|%13s| %9.2f\n", b.kultura, b.izmen); getch(); }
```

ПРАКТИЧЕСКАЯ РАБОТА № 42

Тема: Инспекция программного кода на предмет соответствия стандартам кодирования

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

Оборудование: ПК, программное обеспечение –MS Word, инструкции по выполнению работы.

Содержание работы:

Задание 1. В следующих вариантах переписать программный код, используя общепринятые соглашения и рекомендации по именованию и форматированию переменных, операторов, выражений. Провести инспекцию кода с помощью любого из перечисленных сервисов. Сравнить полученные результаты. Сделать вывод.

Варианты заданий:

Вариант №6.

```
main() { float tr[3][2]; float x,y; float xa,ya,xb,yb,xc,yc; float a1,b1,c1,a2,b2,c2,a3,b3,c3,a4,b4,c4,a5,b5,c5; float a,b,c; float r; int i; clrscr(); printf("Enter coordinats :\n"); for(i=0;i<3;i++) { printf("\ncoordinats:%d: x=", i+1); scanf("%f",&r); tr[i][0]=r; printf("y="); scanf("%f", &r); tr[i][1]=r; } xa=tr[0][0]; ya=tr[0][1]; xb=tr[1][0]; yb=tr[1][1]; xc=tr[2][0]; yc=tr[2][1]; a=sqrt((xa-xb)*(xa-xb)+(ya-yb)*(ya-yb)); b=sqrt((xb-xc)*(xb-xc)+(yb-yc)*(yb-yc)); c=sqrt((xa-xc)*(xa-xc)+(ya-yc)*(ya-yc)); if(!(a+b>c)&&(b+c>a)&&(a+c>b)) printf("Triangle ((%f,%f), (%f,%f),(%f,%f)) virog\n", xa,ya,xb,yb,xc,yc); else { a1=yb-ya; b1=xa-xb; c1=-a1*xa-b1*ya; a2=yc-yb; b2=xb-xc; c2=-a2*xb-b2*yb; a3=yc-ya; b3=xa-xc; c3=-a3*xa-b3*ya; a4=b2; b4=-a2; c4=-a4*xa-b4*ya; a5=b3; b5=-a3; c5=-a5*xb-b5*yb; x=((-c4*b5)-(b4*(-c5)))/((a4*b5)-(b4*a5)); y=((a4*(-c5))-(-c4*a5))/((a4*b5)-(b4*a5)); printf("((%f,%f),)\n", x,y); getch(); }
```

Вариант №7.

```
void main () { const int size= 10; int a[size]; srand(time(NULL)); for (int i = 0; i < size; i++) a[i] = rand() % 11 - 5; for (i = 0; i < size; i++) cout << a[i] << " "; for (i = 0; i < size; i++) for (int j = i+1; j < size; j++) if (a[i] < a[j]) { int buf = a[i]; a[i] = a[j]; a[j] = buf; } cout << endl << endl; for (i = 0; i < size; i++) cout << a[i] << " "; getch(); return 0; }
```

Вариант №8.

```
float f(float z) { return pow(z,3)+6*pow(z,2)+6*z-7; } void main() { float a=-3.0, b=2.0, e=0.001, x;// объявление переменных while (fabs(a-b)>=e) { if((f(a)>0&&f((a+b)/2)<0)||f(a)<0&&f((a+b)/2)>0)) b=(a+b)/2; else if ((f((a+b)/2)>0&&f(b)<0)||f((a+b)/2)<0&&f(b)>0)) a=(a+b)/2; else { printf("! Net kornej!"); return; getch(); } } x=(a+b)/2; printf("x=%f F(x)=%f |a-b|=%f",x,f(x),fabs(a-b)); getch(); }
```

Вариант №9.

```
int main() { clrscr(); int i; float x[10], max, min; for (i=0;i<10;i++) { printf("x[%d]=",i+1); scanf("%f",&x[i]); } max=x[0]; min=x[0]; for(i=1;i<10;i++) { if (x[i]>max) max = x[i]; if (x[i]<min) min=x[i]; } x[0] = max+min; printf("\nmax=%f ",max); printf("\nmin=%f \n",min); for(i=0;i<10;i++) printf("\nx[%d]=%f ",i+1,x[i]); getch(); return 0; }
```

Вариант №10.

```
void main() { cout<<"Sluchaino zapolnenii massiv 8x8:"<<endl; int mass[8][8] = {}; for (int i=0; i<=7; i++) { for(int j=0; j<=7; j++) { mass[i][j]=rand()%500-100; if ((mass[i][j]<10) && (mass[i][j]>=0)) { cout<<" "; } else if (((mass[i][j]>=10) && (mass[i][j]<=99)) { cout<<" "; } else if (((mass[i][j]>=100) && (mass[i][j]<=999)) { cout<<" "; } else if (((mass[i][j]<0) && (mass[i][j]>=-10)) { cout<<" "; } else if (((mass[i][j]<=-10) && (mass[i][j]>=-99)) { cout<<" "; } cout<<mass[i][j]<<" "; } cout<<endl; } int s, max, s1, k, l, t; s=0; for (int i=0; i<=7; i++) { s1=s+abs(mass[i][0]); } for (int j=0; j<=7; j++) { for (int i=0; i<=7; i++) { s=s+abs(mass[i][j]); } if (s>s1) { max=s; s1=max; k=j+1; } s=0; } l=mass[0][k-1]; for (int i=0; i<=7; i++) { t=mass[i][k-1]; if (t<l) { l=t; } } cout<<"Maximal'naya summa elementov v stolbike:"<<k<<endl<<"Minimal'noe znachenie v stolbike:"<<l<<endl; getch(); }
```

Информационное обеспечение обучения

Печатные и электронные издания:

Основные учебные издания:

1. Абрамов, Г. В. Проектирование и разработка информационных систем: учебное пособие для СПО / Г. В. Абрамов, И. Е. Медведкова, Л. А. Коробова. — 2-е изд. — Саратов: Профобразование, 2024. — 169 с. — ISBN 978-5-4488-2259-9. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROФобразование: [сайт]. — URL: <https://profspo.ru/books/143685>
2. Грекул, В. И. Проектирование информационных систем: учебное пособие / В. И. Грекул, Г. Н. Денищенко, Н. Л. Коровкина. — 4-е изд. — Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2024. — 299 с. — ISBN 978-5-4497-3335-1. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROФобразование: [сайт]. — URL: <https://profspo.ru/books/142298>

Дополнительные учебные издания

3. Долженко, А. И. Технологии командной разработки программного обеспечения информационных систем: учебное пособие / А. И. Долженко. — 4-е изд. — Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2024. — 300 с. — ISBN 978-5-4497-2486-1. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROФобразование: [сайт]. — URL: <https://profspo.ru/books/133985>
4. Рощин, П. Г. Командная разработка программного обеспечения с помощью системы контроля версий GIT: конспект лекций: учебное пособие / П. Г. Рощин. — Москва: Национальный исследовательский ядерный университет «МИФИ», 2022. — 106 с. — ISBN 978-5-7262-2846-4. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROФобразование: [сайт]. — URL: <https://profspo.ru/books/132682>

Электронно-библиотечная система:

5. ЭБС «IPRbooks», ООО «Ай Пи Ар Медиа»
6. ЭБС «Znanium»
7. ЭБС «PROФобразование»
8. ЭБС «Book.ru»