Автономная некоммерческая профессиональная образовательная организация
«МЕЖДУНАРОДНЫЙ ВОСТОЧНО-ЕВРОПЕЙСКИЙ КОЛЛЕДЖ»
Пушкинская ул., д. 268, 426008, г. Ижевск. Тел.: (3412) 77-68-24. E-mail: mveu@mveu.ru, www. mveu.ru
ИНН 1831200089. ОГРН 1201800020641

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

по выполнению лабораторных работ

при изучении профессионального модуля

ПМ.02 Осуществление интеграции программных модулей

по специальности

09.02.07 Информационные системы и программирование

Ижевск, 2023

Лабораторная работа – небольшой научный отчет, обобщающий проведенную учащимся работу, которую представляют для защиты преподавателю.

В процессе лабораторного занятия учащиеся выполняют одну или несколько лабораторных работ (заданий) под руководством преподавателя в соответствии с изучаемым содержанием учебного материала.

Состав и содержание лабораторных занятий направлены на реализацию Государственных требований.

Наряду с формированием умений и навыков в процессе лабораторных занятий обобщаются, систематизируются, углубляются и конкретизируются теоретические знания, вырабатывается способность и готовность использовать теоретические знания на практике, развиваются интеллектуальные умения.

Лабораторные занятия проводятся в форме практической подготовки в виде работ, связанных с будущей профессиональной деятельностью.

К лабораторным работам предъявляется ряд требований, основным из которых является полное, исчерпывающее описание всей проделанной работы, позволяющее судить о полученных результатах, степени выполнения заданий и профессиональной подготовке учащихся.

I. Лабораторные работы:

Тема лабораторной работы № 1. «Построение диаграммы Вариантов использования и диаграммы. Последовательности», объем часов 2
У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.
Цель лабораторной работы: исследование процесса построения диаграмм прецедентов и диаграмм взаимодействий в заданной предметной области.

Задание(я):

Задание 1. Создание диаграммы вариантов использования и действующих лиц.

Окончательный вид диаграммы показан на рис. 1.

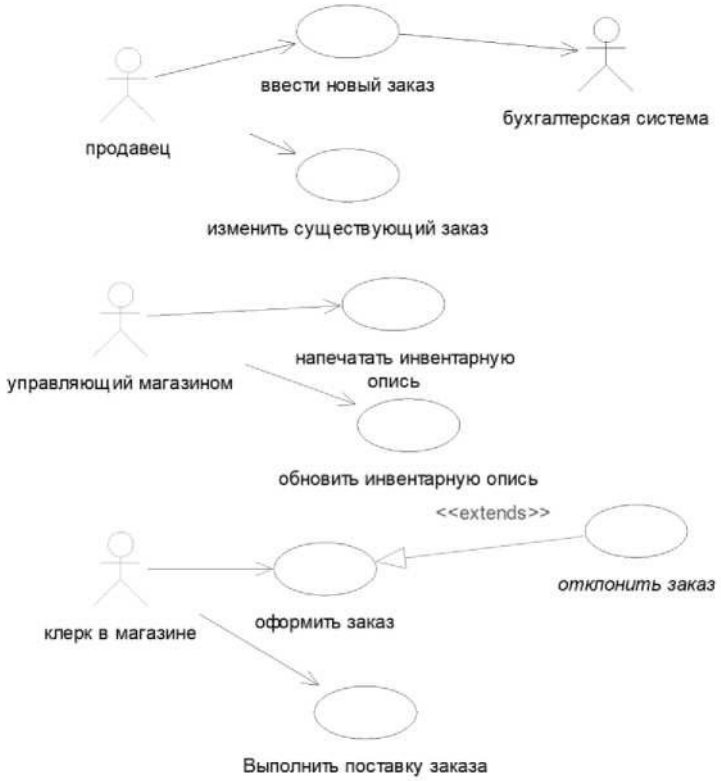


Рис. 1 Диаграмма вариантов использования задачи о заказе товара.

Этапы выполнения

1. Дважды щелкнув мышью на Главной диаграмме Вариантов Использования (Main) в браузере, откройте ее.
2. С помощью кнопки Use Case (Вариант использования) панели инструментов поместите на диаграмму новый вариант использования. Назовите его "Ввести новый заказ".
3. Повторив этапы 2 и 3, поместите на диаграмму остальные варианты использования:

Изменить существующий заказ

Напечатать инвентарную опись

Обновить инвентарную опись

Оформить заказ

Отклонить заказ

Выполнить поставку заказа

4. С помощью кнопки Actor (Действующее лицо) панели инструментов поместите на диаграмму новое действующее лицо.
5. Назовите его " *Продавец*".
6. Повторив шаги 4 и 5, поместите на диаграмму остальных действующих лиц:

Управляющий магазином

Клерк магазина

Бухгалтерская система

7. Создание абстрактного варианта использования (не требующего дальнейшей декомпозиции).

Щелкните правой кнопкой мыши на варианте использования "*Отклонить заказ*" на диаграмме.

8. открывшемся меню выберите пункт Open Specification (Открыть спецификацию).

Установите флажок Abstract (Абстрактный), чтобы сделать этот вариант использования абстрактным.

Добавление ассоциаций

1. С помощью кнопки Unidirectional Association (Однонаправленная ассоциация) панели инструменте нарисуйте ассоциацию между действующим лицом *Продавец* и вариантом использования " *Ввести заказ*".

2. Повторив шаг 1, поместите на диаграмму остальные ассоциации, согласно рис. 1.

Добавление связи расширения

С помощью кнопки Generalization (Обобщение) панели инструментов нарисуйте связь между вариантом использования "*Отклонить заказ*" и вариантом использования "*Оформить заказ*". Стрелка должна быть направлена от первого варианта использования ко второму. Связь расширения означает, что вариант использования "*Отклонить заказ*" при необходимости дополняет функциональные возможности варианта использования " *Оформить заказ*".

Щелкните правой кнопкой мыши на новой связи между вариантами использования " *Отклонить заказ* " и " *Оформить заказ*".

В открывшемся меню выберите пункт Open Specification (Открыть спецификацию).

В раскрывающемся списке стереотипов введите слово extends (расширение), затем нажмите ОК.

Надпись «extends» появится на линии данной связи.

Добавление описаний к вариантам использования

Выделите в браузере вариант использования " *Ввести новый заказ*".

В окне документации введите следующее описание: " Этот вариант использования дает клиенту возможность ввести новый заказ в систему".

С помощью окна документации добавьте описания ко всем остальным вариантам использования.

Добавление описаний к действующему лицу

Выделите в браузере действующее лицо *Продавец.*

В окне документации введите следующее описание: "Продавец — это служащий, старающийся продать товар".

С помощью окна документации добавьте описания к остальным действующим лицам.

Задание 2. Создание диаграммы Последовательности.

Согласовав основные бизнес процессы с Антоном, Павел приступил к построению модели бизнес - процессов, что бы ответить на вопрос - «как это должно делаться в системе». Для начала

он выбрал наиболее важный Вариант использования - «Ввод нового заказа» и построил для него диаграммы взаимодействия.

Диаграммы взаимодействия включают в себя два типа диаграмм - Последовательности и Кооперативную.

Этапы выполнения

Настройка программной среды

1. В меню модели выберите пункт Tools >- Options (Инструменты >- Параметры).
2. Перейдите на вкладку Diagram (Диаграмма).
3. Установите флажки Sequence numbering, Collaboration numbering и Focus of control.
4. Нажмите OK, чтобы выйти из окна параметров.

Создание диаграммы Последовательности

1. Щелкните правой кнопкой мыши на Логическом представлении браузера.
2. В открывшемся меню выберите пункт New > Sequence Diagram (Создать >Диаграмма Последовательности).
3. Назовите новую диаграмму: Ввод заказа.
4. Дважды щелкнув на этой диаграмме, откройте ее.

Добавление на диаграмму действующего лица и объектов

1. Перетащите действующее лицо *Продавец* из браузера на диаграмму.
2. Нажмите кнопку Object (Объект) панели инструментов.
3. Щелкните мышью в верхней части диаграммы, чтобы поместить туда новый объект.
4. Назовите объект *Выбор варианта заказа*.
5. Повторив шаги 3 и 4, поместите на диаграмму объекты:

- *Форма деталей заказа*
- *Заказ №1234*

Добавление сообщений на диаграмму

1. На панели инструментов нажмите кнопку Object Message (Сообщение объекта).
2. Проведите мышью от линии жизни действующего лица *Продавец* к линии жизни объекта *Выбор варианта заказа*.
3. Выделив сообщение, введите его имя — *Создать новый заказ*.
4. Повторив шаги 2 и 3, поместите на диаграмму сообщения:

- *Открыть форму* — между *Выбор Варианта Заказа* и *Форма деталей Заказа*
- *Ввести номер заказа, заказчика и число заказываемых предметов* — между *Продавец* и *Форма Деталей Заказа*
- *Сохранить заказ* — между *Продавец* и *Форма Деталей Заказа*
- *Создать пустой заказ* — между *Форма Деталей Заказа* и *Заказ N1234*
- *Ввести номер заказа, заказчика и число заказываемых предметов* — между *Форма Деталей Заказа* и *Заказ N1234*
- *Сохранить заказ* — между *Форма Деталей Заказа* и *Заказ N1234*

Завершен первый этап работы. Готовая диаграмма Последовательности представлена на рис. 2.

Рис. 2. Диаграмма последовательности без управляющих элементов.

Теперь нужно позаботиться об управляющих объектах и о взаимодействии с базой данных. Как видно из диаграммы, объект *Форма Деталей Заказа* имеет множество ответственностей, с которыми лучше всего мог бы справиться управляющий объект. Кроме того, новый заказ должен сохранять себя в базе данных сам. Вероятно, эту обязанность лучше было бы переложить на другой объект.

Добавление на диаграмму дополнительных объектов

1. Нажмите кнопку Object панели инструментов.
2. Щелкните мышью между объектами *Форма Деталей Заказа* и *Заказ №1234*, чтобы поместить туда новый объект.
3. Введите имя объекта — *Управляющий заказами*.
4. Нажмите кнопку Object панели инструментов.
5. Новый объект расположите справа от *Заказ №1234*.
6. Введите его имя- *Управляющий транзакциями*.

Назначение ответственностей объектам

1. Выделите сообщение 5: *Создать пустой заказ*.
2. Нажав комбинацию клавиш CTRL+D, удалите это сообщение.
3. Повторите шаги 1 и 2 для удаления двух последних сообщений:

- *Вести номер заказа, заказчика и число заказываемых предметов*
- *Сохранить заказ*

4. Нажмите кнопку Object Message панели инструментов.
5. Поместите на диаграмму новое сообщение, расположив его под сообщением 4 между

Форма деталей заказа и *Управляющий заказами*.

6. Назовите его *Сохранить заказ*.
7. Повторите шаги 4 — 6, добавив сообщения с шестого по девятое и назвав их:

- *Создать новый заказ* — между *Управляющий заказами* и *Заказ №1234*
- *Ввести номер заказа, заказчика и число заказываемых предметов*- между *Управляющий заказами* и *Заказ №1234*
- *Сохранить заказ*- между *Управляющий заказами* и *Управляющий транзакциями*
- *Информация о заказе* —между *Управляющий транзакциями* и *Заказ №1234*

8. На панели инструментов нажмите кнопку Message to Self (Сообщение себе).
9. Щелкните на линии жизни объекта *Управляющий транзакциями* ниже сообщения 9, добавив туда рефлексивное сообщение.
10. Назовите его *Сохранить информацию о заказе в базе данных*.

Соотнесение объектов с классами

1. Щелкните правой кнопкой мыши на объекте *Выбор варианта заказа*.
2. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию).

В раскрывающемся списке классов выберите пункт <New> (Создать). Появится окно спецификации классов.

4. В поле Name введите *Выбор заказа*.
5. Щелкните на кнопке ОК. Вы вернетесь в окно спецификации объекта.
6. В списке классов выберите класс *Выбор Заказа*.
7. Щелкните на кнопке ОК, чтобы вернуться к диаграмме. Теперь объект называется *Выбор варианта заказа: Выбор Заказа*
8. Для соотнесения остальных объектов с классами повторите шаги с 1 по 7:

- Класс *Детали заказа* соотнесите с объектом *Форма деталей заказа*
- Класс *Упр_заказами* — с объектом *Управляющий заказами*
- Класс *Заказ* — с объектом *Заказ N 1234*
- Класс *Упр_ транзакциями* — с объектом *Управляющий транзакциями*

Соотнесение сообщений с операциями

1. Щелкните правой кнопкой мыши на сообщении 1: *Создать новый заказ.*
2. В открывшемся меню выберите пункт <new operation> (создать операцию). Появится окно спецификации операции.
3. В поле Name введите имя операции —*Создать*.
4. Нажмите на кнопку ОК, чтобы закрыть окно спецификации операции и вернуться к диаграмме.
5. Еще раз щелкните правой кнопкой мыши на сообщении 1.
6. В открывшемся меню выберите новую операцию *Создать()*.
7. Повторите шаги с 1 по 6, чтобы соотнести с операциями все остальные сообщения:

- Сообщение 2: *Открыть форму* соотнесите с операцией *Открыть()*
- Сообщение 3: *Ввести номер заказа, заказчика и число заказываемых предметов* — с операцией *Ввести номер заказа, заказчика и число заказываемых предметов()*
- Сообщение 4: *Сохранить заказ* — с операцией *Сохранить заказ()*
- Сообщение 5: *Сохранить заказ* — с операцией *Сохранить заказ()*
- Сообщение 6: *Создать пустой заказ - с* операцией *Создать пустой заказ()*
- Сообщение 7: *Ввести номер заказа, заказчика и число заказоваемых предметов*- с одноименной операцией.
- Сообщение 8 *Сохранить заказ* - с операцией *Сохранить заказ()*
- Сообщение 9 *Информация о заказе* - с одноименной операцией
- Сообщение 10 *Сохранить информацию о заказе* с одноименной операцией.

Диаграмма должна выглядеть, как на рис. 3.

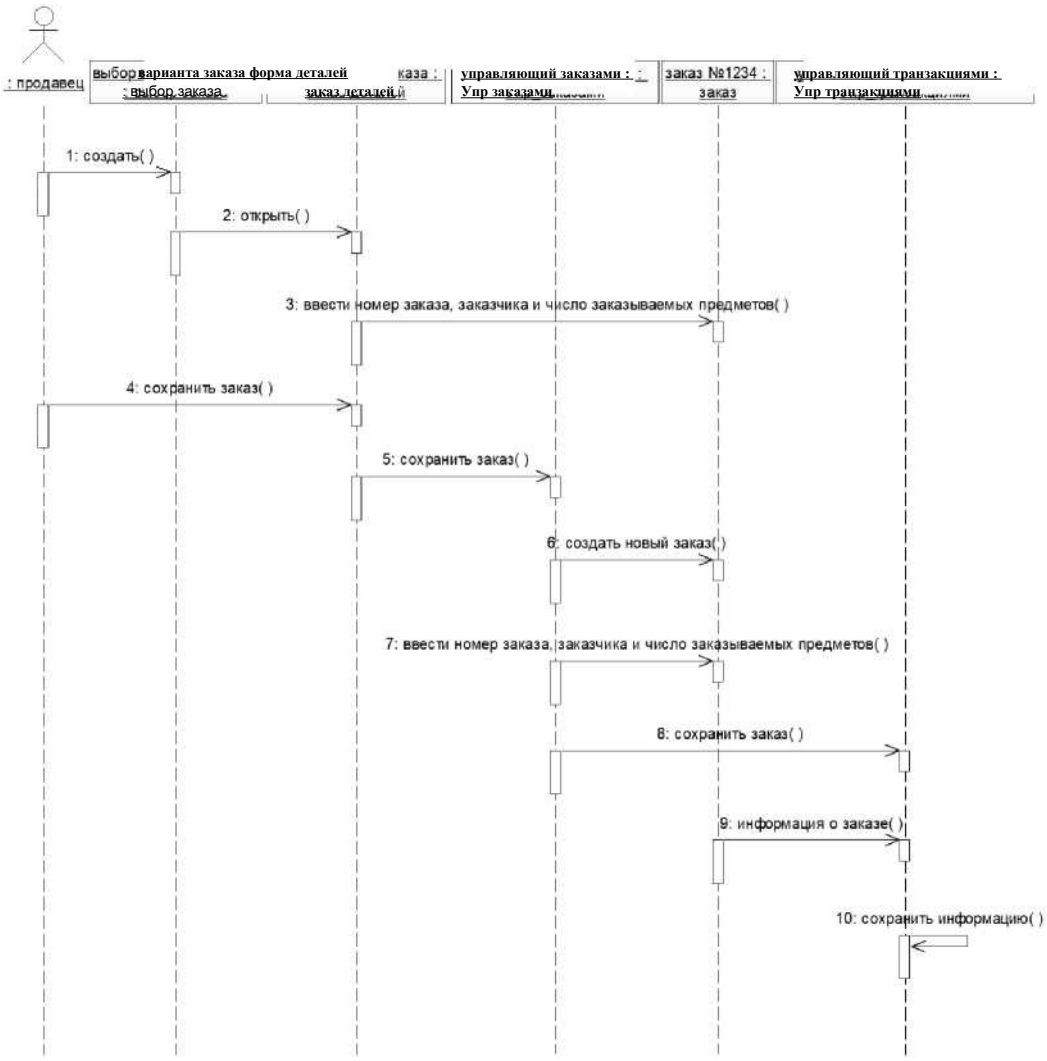


Рис. 3. Окончательный вид диаграммы

Методические указания по ходу выполнения работы

Постановка задачи (описание предметной области).

Магазин осуществляет продажу товаров клиенту путем оформления документов «Заказ». Директор магазина- Антон, принял решение автоматизировать документооборот продаж товара и пригласил для выполнения работ программиста Павла. Поговорив с Антоном, в соответствие с концепцией жизненного цикла (ЖЦ) программы Павел приступил к описанию бизнес процессов, сопровождающих продажу товара. Взяв за основу язык UML, он начал с построения контекстной диаграммы процессов- Use Case diagram. Диаграмма должна ответить на вопрос-«что должно делаться в системе и кто участник этих процессов».

Тема лабораторной работы № 2. «Построение диаграммы Кооперации и диаграммы Развертывания», объем часов 2

У1 использовать выбранную систему контроля версий;

У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: исследование моделирования процессов, описывающих взаимодействие объектов в диаграмме кооперации и диаграмме развертывания в заданной предметной области.

Задание(я):

Задание 1. Построение Создание Кооперативной диаграммы

Конечный вид диаграммы представлен на рис. 4.

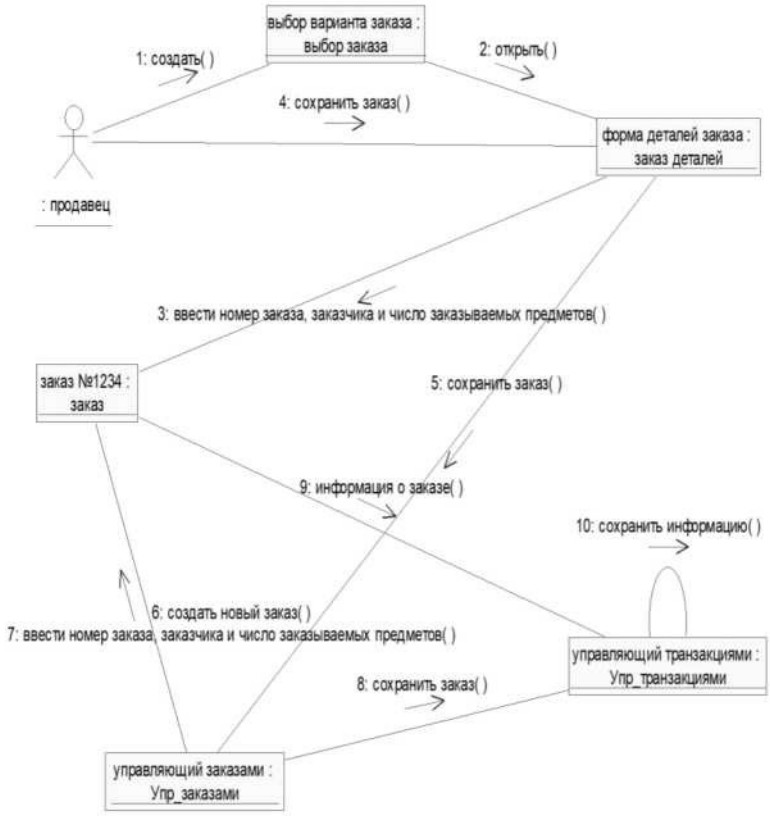


Рис. 4. Окончательный вид кооперативной диаграммы.

1. Щелкните правой кнопкой мыши на Логическом представлении в браузере.
2. В открывшемся меню выберите пункт New > Collaboration Diagram (Создать > Кооперативная диаграмма).
3. Назовите эту диаграмму *Ввод заказа*.
4. Дважды щелкнув мышью на диаграмме, откройте ее.

Добавление действующего лица и объектов на диаграмму

1. Перетащите действующее лицо *Продавец* из браузера на диаграмму.
2. Нажмите кнопку Object (Объект) панели инструментов.
3. Щелкните мышью где-нибудь внутри диаграммы, чтобы поместить туда новый объект.
4. Назовите объект *Выбор варианта заказа*.
5. Повторив шаги 3 и 4, поместите на диаграмму объекты:

- *Форма деталей заказа*
- *Заказ №1234*

Добавление сообщений на диаграмму

1. На панели инструментов нажмите кнопку Object Link (Связь объекта).
2. Проведите мышью от действующего лица *Продавец* к объекту *Выбор варианта заказа*.
3. Повторите шаги 1 и 2, соединив связями следующие объекты:
   - Действующее лицо *Продавец* и объект *Форма деталей Заказа*
   - Объект *Форма деталей Заказа* и объект *Выбор Варианта Заказа*
   - Объект *Форма деталей Заказа* объект *Заказ N1234*
4. На панели инструментов нажмите кнопку Link Message (Сообщение связи).
5. Щелкните мышью на связи между *Продавец* и *Форма деталей Заказа.*
6. Выделив сообщение, введите его имя —*Создать новый заказ*;
7. Повторив шаги с 4 по 6, поместите на диаграмму сообщения:

- *Открыть форму —* между *Выбор Варианта Заказа* и *Форма Деталей Заказа*.
- *Ввести номер заказа, заказчика и число заказываемых предметов —* между *Продавец* и *Форма Деталей Заказа*
- *Сохранить заказ —* между *Продавец* и *Форма деталей Заказа*
- *Создать пустой заказ —* между *Форма деталей Заказа* и *Заказ №1234*
- Ввести номер заказа, заказчика и число заказываемых предметов — между *Форма деталей Заказа* и *Заказ №1234*
- Сохранить заказ — между *Форма деталей Заказа* и *Заказ №1234*

Теперь нужно поместить на диаграмму дополнительные элементы, а также рассмотреть ответственности объектов.

Добавление на диаграмму дополнительных объектов

1. Нажмите кнопку Object панели инструментов.
2. Щелкните мышью где-нибудь на диаграмме, чтобы поместить туда новый объект.
3. Введите имя объекта — *Управляющий заказами*.
4. На панели инструментов нажмите кнопку Object.
5. Поместите на диаграмму еще один объект.
6. Введите его имя — *Управляющий транзакциями*.

Назначение ответственностей объектам

1. Выделите сообщение 5: *Создать пустой заказ*. Выделяйте слова, а не стрелку.
2. Нажав комбинацию клавиш CTRL+D, удалите это сообщение.
3. Повторите шаги 1 и 2 для удаления сообщений 6 и 7:

- *Ввести номер заказа, заказчика и число заказываемых предметов*
- *Сохранить заказ*

4. Выделите связь между объектами *Форма деталей Заказа* и *Заказ №1234*
5. Нажав комбинацию клавиш CTRL+D, удалите эту связь
6. На панели инструментов нажмите кнопку Object Link (Связь объекта).
7. Нарисуйте связь между *Форма деталей Заказа* и *Управляющий Заказами*.
8. На панели инструментов нажмите кнопку Object Link (Связь объекта).
9. Нарисуйте связь между *Управляющий Заказами* и *Заказ №1234*
10. На панели инструментов нажмите кнопку Object Link (Связь объекта).
11. Нарисуйте связь между *Заказ №1234* и *Управляющий Транзакцией*.
12. На панели инструментов нажмите кнопку Object Link (Связь объекта).
13. Нарисуйте связь между *Управляющий Заказами* и *Управляющий Транзакцией*.
14. На панели инструментов нажмите кнопку Link Message (Сообщение связи).
15. Щелкните мышью на связи между объектами *Форма деталей Заказа* и *Управляющий Заказами*, чтобы ввести новое сообщение.
16. Назовите это сообщение Сохранить заказ.
17. Повторите шаги 14 — 16, добавив сообщения с шестого по девятое, и назвав их:

- *Создать новый заказ —* между *Управляющий Заказами* и *Заказ №1234*
- *Ввести номер заказа, заказчика и число заказываемых предметов —* между

Управляющий Заказами и *Заказ №1234*

- *Сохранить заказ* — между *Управляющий Заказами* и *Управляющий Транзакцией*
- *Информация о заказе* — между *Управляющий Транзакцией* и *Заказ №1234*

18. На панели инструментов нажмите кнопку Link to Self (Связь с собой).
19. Щелкнув на объекте *Управляющий Транзакцией*, добавьте к нему рефлексивное сообщение.
20. На панели инструментов нажмите кнопку Link Message (Сообщение связи).
21. Щелкните мышью на рефлексивной связи *Управляющий Транзакциями*, чтобы ввести туда сообщение.
22. Назовите новое *Сохранить информацию о заказе в базе данных*.

Соотнесение объектов с классами (если классы были созданы при разработке описанной выше диаграммы Последовательности)

1. Найдите в браузере класс *Выбор Заказа*.
2. Перетащите его на объект *Выбор варианта заказа* на диаграмме.
3. Повторите шаги 1 и 2 соотнеся остальные объекты и соответствующие им классы:

- Класс *заказ деталей* соотнесите с объектом *Форма деталей заказа*
- Класс *Упр_заказами* — с объектом *Управляющий Заказами*
- Класс *Заказ* — с объектом *Заказ №1234*
- Класс *Упр_транзакциями* — с объектом *Управляющий транзакциями*

Соотнесение объектов с классами (если вы не создавали описанную выше диаграмму Последовательности)

1. Щелкните правой кнопкой мыши на объекте *Форма деталей Заказа*.
2. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию).
3. В раскрывающемся списке классов выберите пункт <New> (Создать). Появится окно спецификации классов.
4. В поле имени введите *Выбор заказа*.
5. Щелкните на кнопке OK. Вы вернетесь в окно спецификации объекта.
6. В списке классов выберите класс *Выбор заказа.*
7. Щелкните на кнопке OK, чтобы вернуться к диаграмме. Теперь объект называется *Выбор варианта заказа: Выбор Заказа*
8. Для соотнесения остальных объектов с классами повторите шаги с 1 по 7:

- Класс *Детали заказа* соотнесите с объектом *Форма деталей заказа*
- Класс *Упр_заказами* — с объектом *Управляющий заказами*
- Класс *Заказ* — с объектом *Заказ N 1234*
- Класс *Упр_ транзакциями* — с объектом *Управляющий транзакциями*

Соотнесение сообщений с операциями (если операции были созданы при разработке описанной выше диаграммы Последовательности)

1. Щелкните правой кнопкой мыши на сообщении 1: Создать новый заказ.
2. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию).
3. В раскрывающемся списке имен укажите имя операции — *Создать()*.
4. Нажмите на кнопку OK.
5. Повторите шаги 1—4 для соотнесения с операциями остальных сообщений:

- Сообщение 2: *Открыть форму* соотнесите с операцией *Открыть()*
- Сообщение 3: *Ввести номер заказа, заказчика и число заказываемых предметов* — с операцией *Ввести номер заказа, заказчика и число заказываемых предметов()*
- Сообщение 4: *Сохранить заказ* — с операцией *Сохранить заказ()*
- Сообщение 5: *Сохранить заказ* — с операцией Сохранить заказ()

Сообщение 6: *Создать пустой заказ - с* операцией Создать пустой заказ()

Сообщение 7: *Ввести номер заказа, заказчика и число заказоваемых предметов-* с одноименной операцией.

Сообщение 8 Сохранить заказ - с операцией Сохранить заказ()

Сообщение 9 *Информация о заказе* - с одноименной операцией

Сообщение 10 *Сохранить информацию о заказе* с одноименной операцией

Соотнесение сообщений с операциями (если вы не создавали описанную выше диаграмму Последовательности)

1. Щелкните правой кнопкой мыши на сообщении 1: *Создать новый заказ()*.
2. В открывшемся меню выберите пункт <new operation> (создать операцию). Появится окно спецификации операции.
3. В поле имени введите имя операции —*Создать().*
4. Нажмите на кнопку ОК, чтобы закрыть окно спецификации операции и вернуться к диаграмме.
5. Еще раз щелкните правой кнопкой мыши на сообщении 1.
6. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию).
7. В раскрывающемся списке Name (Имя) укажите имя новой операции.
8. Нажмите на кнопку ОК.
9. Повторите шаги 1—8, чтобы создать новые операции и соотнести с ними остальные сообщения:

- Сообщение 2: *Открыть форму* соотнесите с операцией *Открыть()*
- Сообщение 3: *Ввести номер заказа, заказчика и число заказываемых предметов* — с операцией *Ввести номер заказа, заказчика и число заказываемых предметов()*
- Сообщение 4: *Сохранить заказ* — с операцией *Сохранить заказ()*
- Сообщение 5: *Сохранить заказ* — с операцией *Сохранить заказ()*

Сообщение 6: *Создать пустой заказ - с* операцией *Создать пустой заказ()*

Сообщение 7: *Ввести номер заказа, заказчика и число заказоваемых предметов-* с одноименной операцией.

Сообщение 8 Сохранить заказ - с операцией *Сохранить заказ()*

Сообщение 9 *Информация о заказе* - с одноименной операцией

Сообщение 10 *Сохранить информацию о заказе* с одноименной операцией

Ваша диаграмма должна выглядеть, как показано на рис. 4

Методические указания по ходу выполнения работы

Чётко следуем инструкциям.

Тема лабораторной работы № 3. «Построение диаграммы Деятельности, диаграммы Состояний и диаграммы Классов», объем часов 2

У1 использовать выбранную систему контроля версий;

У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: исследование процесса построения диаграммы состояний и диаграммы классов в заданной предметной области.

Задание(я):

Задание 1. Постройте диаграмму Состояний

Постройте диаграмму Состояний для класса *Заказ*, показанную на рис. 5.

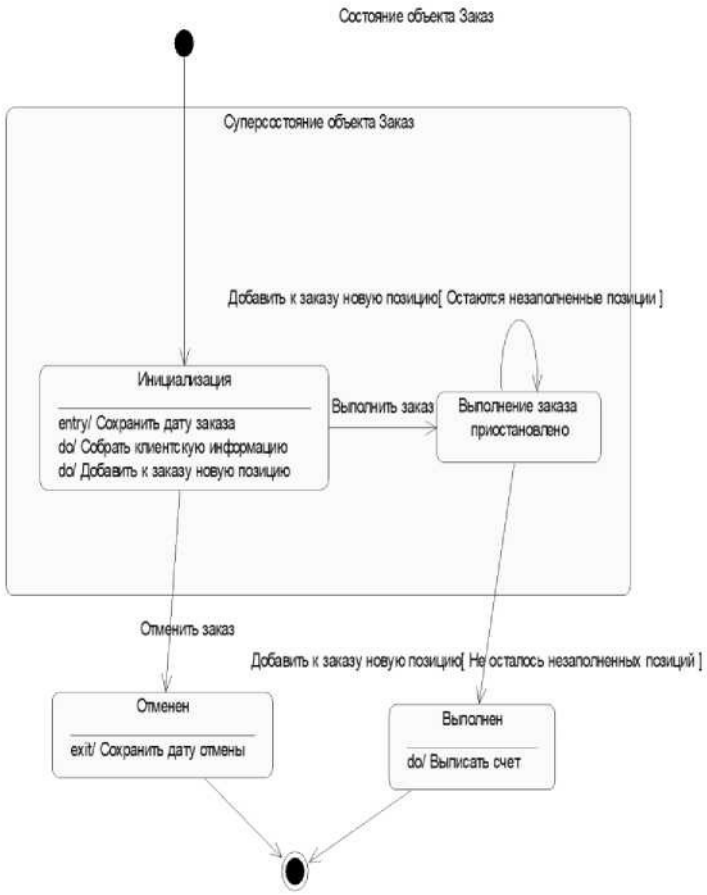


Рис 5. Диаграмма состояний для класса *Заказ*

Этапы выполнения

Создание диаграммы

1. Найдите в браузере класс *Заказ*.
2. Щелкните на классе правой кнопкой мыши и в открывшемся меню укажите пункт New > Statechart Diagram (Создать диаграмму состояний).

Добавление начального и конечного состояний

1. Нажмите кнопку Start State (Начальное состояние) панели инструментов.

2. Поместите это состояние на диаграмму.
3. Нажмите кнопку End State (Конечное состояние) панели инструментов.
4. Поместите это состояние на диаграмму.

Добавление суперсостояния

1. Нажмите кнопку State (Состояние) панели инструментов.
2. Поместите это состояние на диаграмму.

Добавление оставшихся состояний

1. На панели инструментов нажмите кнопку State (Состояние).
2. Поместите состояние на диаграмму.
3. Назовите состояние *Отменен*.
4. На панели инструментов нажмите кнопку State (Состояние).
5. Поместите состояние на диаграмму.
6. Назовите состояние *Выполнен*.
7. На панели инструментов нажмите кнопку State (Состояние).
8. Поместите состояние на диаграмму внутрь суперсостояния.
9. Назовите состояние *Инициализация*.
10. На панели инструментов нажмите кнопку State (Состояние).
11. Поместите состояние на диаграмму внутрь суперсостояния.
12. Назовите состояние *Выполнение заказа приостановлено*.

Описание состояний

1. Дважды щелкните мышью на состоянии *Инициализация*.
2. Перейдите на вкладку Detail (Подробно).
3. Щелкните правой кнопкой мыши в окне Лс1ю1^(Действия).
4. В открывшемся меню выберите пункт й^егЦВставить).
5. Дважды щелкните мышью на новом действии.
6. Назовите его *Сохранить дату заказа*.
7. Убедитесь, что в окне When (Когда) указан пункт On Entry (На входе).
8. Повторив шаги 3—7, добавьте следующие действия:

- *Собрать клиентскую информацию*, в окне When укажите DO (Выполнять между входом и выходом)
- *Добавить к заказу новые позиции*, укажите DO

9. Нажмите два раза на ОК, чтобы закрыть спецификацию.
10. Дважды щелкните мышью на состоянии *Отменен*.
11. Повторив шаги 2—7, добавьте действия:

Сохранить дату отмены, укажите On Exit (На выходе)

12. Нажмите два раза на ОК, чтобы закрыть спецификацию.
13. Дважды щелкните мышью на состоянии *Выполнен*.
14. Повторив шаги 2—7, добавьте действие:

- *Выписать счет*, укажите On Exit

15. Нажмите два раза на ОК, чтобы закрыть спецификацию.

Добавление переходов

1. Нажмите кнопку Transition (Переход) панели инструментов.
2. Щелкните мышью на начальном состоянии.
3. Проведите линию перехода к состоянию *Инициализация*.
4. Повторив шаги с первого по третий, создайте следующие переходы:

- От состояния Инициализация к состоянию *Выполнение заказа приостановлено*
- От состояния *Выполнение заказа приостановлено* к состоянию *Выполнен*
- От суперсостояния к состоянию *Отменен*
- От состояния *Отменен* к конечному состоянию
- От состояния *Выполнен* к конечному состоянию

5. На панели инструментов нажмите кнопку Transition to Self (Переход к себе).

6. Щелкните мышью на *состоянии Выполнение заказа приостановлено*

Описание переходов

1. Дважды щелкнув мышью на переходе от состояния *Инициализация* к состоянию *Выполнение заказа приостановлено*, откройте окно спецификации перехода.
2. В поле Event (Событие) введите фразу *Выполнить заказ*.
3. Щелкнув на кнопке ОК, закройте окно спецификации.
4. Повторив шаги с первого по третий, добавьте событие *Отменить заказ* к переходу между суперсостоянием и состоянием *Отменен*.
5. Дважды щелкнув мышью на переходе от состояния *Выполнение заказа приостановлено* к состоянию *Выполнен*, откройте окно его спецификации.
6. В поле Event (Событие) введите фразу *Добавить к заказу новую позицию*.
7. Перейдите на вкладку Detail (Подробно).
8. В поле Guard Condition (Сторожевое Условие) введите *Не осталось незаполненных позиций*.
9. Щелкнув на кнопке ОК, закройте окно спецификации.
10. Дважды щелкните мышью на рефлексивном переходе (Transition to Self) состояния *Выполнение заказа приостановлено*.
11. В поле Event (Событие) введите фразу *Добавить к заказу новую позицию*.
12. Перейдите на вкладку Detail (Подробно).
13. В поле Guard Condition (Сторожевое Условие) введите *Остаются незаполненные позиции*.
14. Щелкнув на кнопке ОК, закройте окно спецификации.

Задание 2. Построение диаграммы Активности для варианта использования «Выполнить поставку Заказа».

Побеседовав с Павлом, Антон понял, что необходимо согласовать логику реализации еще одного варианта использования «Выполнить поставку заказа». Стало ясно, что здесь возможны несколько альтернативных потоков управления. Для таких ситуаций более удобно использовать не диаграммы взаимодействия, приспособленные для единственного потока, а диаграмму активности.

Описание варианта использования.

При оформлении заказа проверяют каждую содержащуюся в нем позицию, чтобы убедиться в наличии соответствующих товаров на складе. После этого выписываются товары для реализации заказа. Во время выполнения этих процедур одновременно проверяется прохождение платежа. Если платеж прошел, и товары имеются на складе, то осуществляется их поставка. Если платеж прошел, но товары на складе отсутствуют, то заказ ставится в ожидание. Если платеж не прошел, то заказ аннулируется.

Этапы выполнения

1. Найдите в браузере вариант использования «Выполнить поставку заказа»
2. Щелкните на классе правой кнопкой мыши и в открывшемся меню укажите пункт New > Activity Diagram (Создать диаграмму активности).
3. Назвите диаграмму «Выполнить поставку» и откройте ее двукратным щелчком мыши
4. На панели инструментов щелкните мышкой на элементе Swimline, затем на поле диаграммы. На диаграмме появится разделительная линия («водная дорожка»).
5. Установите курсор на заголовок NewSwinlane и нажмите правую клавишу мыши. В выпадающем списке нажмите Select in brousere. В браузере выделится этот объект. Нажав правую клавишу мыши в выпадающем списке выберете Open Specefication и откройте спецификацию. Измените поле Name на *Клерк.* Выберите в поле Class *Клерк в магазине*.
6. Выполните заново пункты 5-6 и присвойте полю Name *Система*, Class- *Бухгалтерская система.*
7. Найдите в браузере сплошной черный кружок (начальное состояние). Перенесите его на дорожку *Клерк.*

8. Выберете из панели инструментов объект Activity и поместите его на диаграмму в “дорожку” *Клерк*. Измените имя объекта на ”*Получить заказ*”.

9. Повторите предыдущий этап, создайте на «дорожке» *Клерк* 4 новых Activity и присвойте им имена *Проверить позицию заказа, закрепить позицию за заказом, Поставить заказ в ожидание, Скомплектовать заказ*

10. Поместите на «дорожку» 2 новых объекта End State (конечное состояние). Одному из них измените поле Name на «*Выполнить поставку*»

11. На дорожку *Система* поместите новый объект Activity и присвойте полю Name “*Проверить платеж*”. На эту же дорожку поместите новый объект End State и измените в его спецификации поле Name на «*Отменить заказ*».

12. Поместить на «дорожку» Клерк 2 объекта Horisontal Sinhronization (горизонтальная синхронизация). Присвойте полю Name спецификации одного объекта «*1*», другого- «*2*».

13. Поместить на «дорожку» *Клерк* объект Dicision (выбор). Через спецификацию присвойте полю Name «*Позиция имеется?*».

14. Поместить на «дорожку» *Система* объект Dicision. Присвойте полю Name «*Деньги поступили?*».

15. Щелкните мышкой на панели инструментов объекте- стрелке State Transition (состояние перехода). Затем щелкните мышкой на диаграмме объекта начальное состояние. Удерживая кнопку мыши перенесите курсор на активность ”*Получить заказ*” и лишь затем отпустить курсор. В результате два объекта будут соединены стрелкой.

16. Выполните этап 14, соединив стрелкой объект Активность «*Получить заказ*» с объектом Horisontal Sinhronization 1.

17. Соедините этими же стрелками объекты 1 и «Проверить платеж», 1 и «Проверить позицию заказа», «Проверить заказ» и «Деньги подступили?», «Деньги поступили?» и «Отменить заказ», «Проверить позицию заказа» и «Позиция имеется», «Позиция имеется» и «Закрепить позицию за заказом», «Деньги получены?» и 2, «Закрепить позицию за заказом» и 2, «Позиция имеется?» и «Поставить заказ в ожидание», 2 и «Скомплектовать заказ», «Скомплектовать заказ» и «Выполнить поставку», «Поставить заказ в ожидание» и объект Конечное состояние (без имени).

18. Присвоим некоторым стрелкам наименование полю Event (условие перехода).Для этого, установим курсор на стрелке, соединяющей «Деньги получены?» и «Отменить заказ». Двукратным щелчком мыши откроем окно спецификации. В поле Event введем «Нет».

19. Выполним пункт 18 для стрелки, соединяющей «Деньги получены?» и 2 и присвойте Event «Да». Аналогично для стрелки соединяющей «Позиция имеется?» и «Закрепить позицию за заказом» присвоить Event «Да». Стрелке, соединяющей «Позиция имеется?» и «Поставить заказ в ожидание» - «Нет».

20. Добавим элементарные действия (Actions) к активности “Проверить позицию заказа». Установим курсор на «Проверить позицию заказа» и двукратным щелчком мыши откроем окно спецификации. Откроем закладку Actions. Установим курсор на свободное поле и нажмем правую клавиши мыши. В выпадающем меню нажмем Insert. В появившейся заставке в поле When выберем Entry(На входе в активность), В поле Name введем «Просмотреть спецификацию к заказу». Нажать Ok. Вновь нажмем курсор правой мыши и введем новое действие. Полю When прсвоим Do(промежумок между входом и выходом), а полю Name «Найти новую позицию». При вводе третьей активности полю When присвойте Exit (выход), а полю Name «Передать результаты поиска».

21. Путем перемещения объектов (установить курсор мыши- нажать- тащить- отпустить) привести диаграмму к виду, показанному на рис. 6.

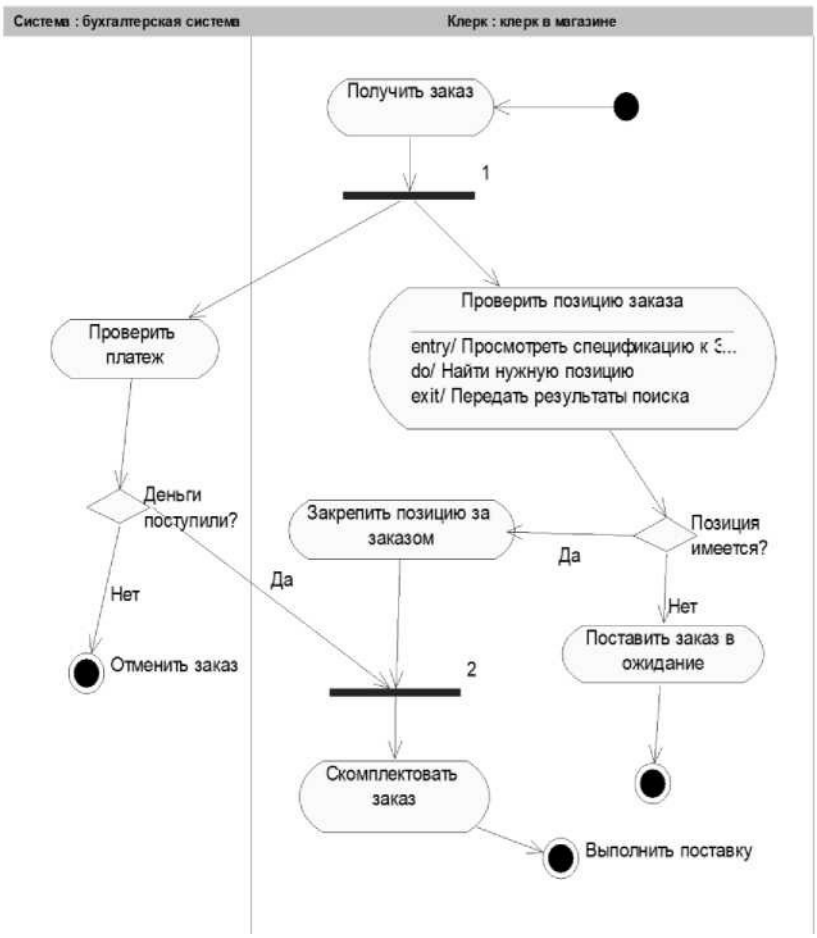


Рис. 6 Диаграмма активности для варианта использования «Выполнить поставку заказа»

Задание 3. Пакеты и классы

В этом упражнении необходимо сгруппировать в пакеты классы, созданные при выполнении предыдущих работ. Затем нужно будет построить несколько диаграмм Классов и показать на них классы и пакеты системы.

Создание диаграммы Классов

Объедините обнаруженные классы в пакеты. Создайте диаграмму Классов для отображения пакетов, диаграммы Классов, для представления классов в каждом пакете и диаграмму Классов для представления всех классов варианта использования "Ввести новый заказ".

Этапы выполнения

Создание пакетов

1. Щелкните правой кнопкой мыши на Логическом представлении браузера.
2. В открывшемся меню выберите пункт New > Package (Создать >Пакет).
3. Назовите новый пакет Сущности.
4. Повторив шаги 1—3, создайте пакеты Границы и Управление.

Создание Главной диаграммы Классов

1. Дважды щелкнув мышью на Главной диаграмме Классов, находящейся под Логическим представлением браузера, откройте ее.
2. Перетащите пакет *Сущности* из браузера на диаграмму.
3. Перетащите пакеты *Границы* и *Управление* из браузера на диаграмму.

Главная диаграмма Классов должна выглядеть, как показано на рис. 7

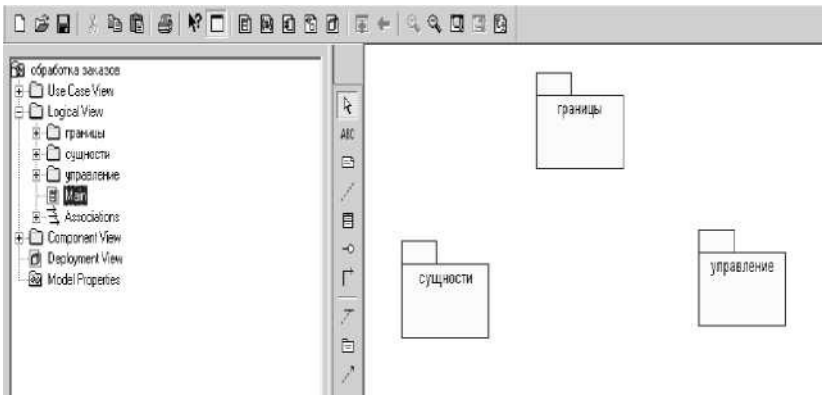


Рис. 7 Главная диаграмма классов в логическом представлении браузера.

Создание диаграммы Классов для сценария "Ввести новый заказ" с отображением всех классов

1. Щелкните правой кнопкой мыши на Логическом представлении браузера.
2. В открывшемся меню выберите пункт New > Class Diagram (Создать > Диаграмма Классов).
3. Назовите новую диаграмму Классов: Ввод нового заказа.
4. Дважды щелкнув мышью на этой диаграмме в браузере, откройте ее.
5. Перетащите из браузера все классы (*Выбор_заказа, Заказ_деталей, упр_заказами, Заказ, Упр_транзакциями.*

Объединение классов в пакеты

1. В браузере перетащите класс *выбор_заказа* на пакет Границы.
2. Перетащите класс *заказ_деталей* на пакет Границы.
3. Перетащите классы *Упр_заказами* и *Упр-транзакциями* на пакет Управление.
4. Перетащите класс *Заказ* на пакет Сущности.
5. ассы и пакеты в браузере показаны на рис. 9

5£l обработка заказов
+ D Use Case View
- D Logical View
- Г~i границы
§ Main
+ Ё! выбор заказа
+ Ё! заказ деталей
Associations
- Г~I сущности
+ Ё! заказ
Associations
- Г~I управление
+ Ё! Упр_заказами
+ Ё! Упр_транзакциями
Associations

Рис. 8 Представление пакетов и классов

Добавление диаграмм Классов к каждому пакету

1. В браузере щелкните правой кнопкой мыши на пакете *Границы*.
2. В открывшемся меню выберите пункт New > Class Diagram (Создать > Диаграмма Классов).
3. Введите имя новой диаграммы — Main (Главная).
4. Дважды щелкнув мышью на этой диаграмме, откройте ее.
5. Перетащите на нее из браузера классы *выбор_заказа* и *заказ_деталей*.
6. Закройте диаграмму.

В браузере щелкните правой кнопкой мыши на пакете *Сущности.*

8. В открывшемся меню выберите пункт New > Class Diagram (Создать > Диаграмма Классов).
9. Введите имя новой диаграммы — Main (Главная).
10. Дважды щелкнув мышью на этой диаграмме, откройте ее.
11. Перетащите на нее из браузера класс *Заказ*.
12. Закройте диаграмму
13. В браузере щелкните правой кнопкой мыши на пакете *Управление*
14. В открывшемся меню выберите пункт New > Class Diagram (Создать > Диаграмма Классов).
15. Введите имя новой диаграммы — Main (Главная).
16. Дважды щелкнув мышью на этой диаграмме, откройте ее.
17. Перетащите на нее из браузера классы *Упр_заказами* и *Упр_транзакциями*
18. Закройте диаграмму

Задание 4. Уточнение методов и свойств классов.

В этом упражнении к описаниям операций будут добавлены детали, включая параметры и типы возвращаемых значений, и определены атрибуты классов

Постановка проблемы

Для определения атрибутов классов был проанализирован поток событий. В результате к классу *Заказ* диаграммы Классов были добавлены атрибуты *Номер заказа* и *Имя клиента*. Так как в одном заказе можно указать большое количество товаров и у каждого из них имеются свои собственные данные и поведение, было решено моделировать товары как самостоятельные классы, а не как атрибуты класса *Заказ*.

Добавление атрибутов и операций

Добавим атрибуты и операции к классам диаграммы Классов "Ввод нового заказа". При этом используем специфические для языка особенности. Установим параметры так, чтобы показывать все атрибуты, все операции и их сигнатуры. Применим нотацию UML.

Этапы выполнения

Настройка

1. В меню модели выберите пункт Tools > Options (Инструменты >Параметры).
2. Перейдите на вкладку Diagram.
3. Убедитесь, что флажок Show visibility (Показать видимость) установлен.
4. Убедитесь, что флажок Show stereotyps (Показать стереотипы) установлен.
5. Убедитесь, что флажок Show operation signatures (Показать сигнатуры операций) установлен.
6. Убедитесь, что флажки Show all attributes (Показать все атрибуты) и Show all operations (Показать вое операции) установлены.
7. Убедитесь, что флажки Suppress attributes (Подавить атрибуты) и Suppress operations (Подавить операции) сброшены.
8. Перейдите на вкладку Notation (Нотация).
9. Убедитесь, что флажок Visibility as icons (Отображать пиктограммы) сброшен.

Добавление нового класса

1. Найдите в браузере диаграмму Классов варианта использования "Ввести новый заказ".
2. Дважды щелкнув мышью на диаграмме, откройте ее.
3. Нажмите кнопку Class панели инструментов.
4. Щелкните мышью внутри диаграммы, чтобы поместить туда новый класс.
5. Назовите его *Позиц_заказа*.
6. Назначьте этому классу стереотип Entity.
7. В браузере перетащите класс в пакет *Сущности.*

Добавление атрибутов

1. Щелкните правой кнопкой мыши на классе *Заказ*.
2. В открывшемся меню выберите пункт New Attribute (Создать атрибут),
3. Введите новый атрибут:
OrderNumber : Integer

4. Нажмите клавишу Enter
5. Введите следующий атрибут:
CustomerName : String.

6. Повторив шаги 4 и 5, добавьте атрибуты:
OrderDate : Date

OrderFillDate : Date

Если тип атрибута не появляется в выпадающем списке, то введите его от руки и он далее будет появляться.

7. Щелкните правой кнопкой мыши на классе *Позиц_заказа*.
8. В открывшемся меню выберите пункт New Attribute (Создать атрибут).
9. Введите новый атрибут:
ItemID : Integer.

10. Нажмите клавишу Enter.
11. Введите следующий атрибут:
ItemDescription : String.

Добавление операций к классу *Позиц_заказа*

1. Щелкните правой кнопкой мыши на классе *Позиц_заказа*.
2. В открывшемся меню выберите пункт New Opration (Создать операцию).
3. Введите новую операцию:
Создать()

4. Нажмите клавишу Enter.
5. Введите следующую операцию:
Взять_информацию()

6. Нажмите клавишу Enter.
7. Введите операцию:
Дать_информацию()

Подробное описание операций с помощью диаграммы Классов

1. Щелкнув мышью на классе *Заказ*, выделите его.
2. Щелкните на этом классе еще раз, чтобы переместить курсор внутрь.
3. Отредактируйте операцию *Создать()*, чтобы она выглядела следующим образом:
Создать() : Boolean

4. Отредактируйте операцию *Взять_информацию*:
Взять_информацию (OrderNum : Integer, Customer : String, OrderDate : Date, FillDate : Date): Boolean

5. Отредактируйте операцию *Дать_информацию*;
Дать_информацию(): String

Подробное описание операций с помощью браузера

1. Найдите в браузере класс *Позиц_заказа*.
2. Раскройте этот класс, щелкнув на значке "+" рядом с ним. В браузере появятся атрибуты и операции класса.
3. Дважды щелкнув мышью на операции *Дать_информацию()* , откройте окно ее спецификации:
4. В раскрывающемся списке Return class (Возвращаемый класс) укажите String.
5. Щелкнув мышью на кнопке OK, закройте окно спецификации операции.
6. Дважды щелкните в браузере на операции *Дать_информацию* () класса *Позиц_заказа*, чтобы открыть окно ее спецификации.
7. В раскрывающемся списке Return class укажите Boolean.
8. Перейдите на вкладку Detail(nogpo6Ho).
9. Щелкните правой кнопкой мыши в области аргументов, чтобы добавить туда новый параметр:
10. В открывшемся меню выберите пункт Insert (Вставить). Rose добавит аргумент под названием argname.
11. Щелкнув один раз на этом слове, выделите его и измените имя аргумента на ID.
12. Щелкните на колонке Type (Тип). В раскрывающемся списке типов выберите Integer (Если этого либо иного необходимого типа не будет- введите его вручную).
13. Щелкните на колонке Default (По умолчанию), чтобы добавить значение аргумента по умолчанию. Введите число 0.
14. Нажав на кнопку OK, закройте окно спецификации операции.
15. Дважды щелкните на операции *Создать()* класса *Позиц_заказа*, чтобы открыть окно ее спецификации.
16. В раскрывающемся списке Return class укажите Boo1eап.
17. Нажав на кнопку OK, закройте окно спецификации операции.

Подробное описание операций

1. Используя браузер или диаграмму Классов, введите следующие сигнатуры операций класса *Заказ_деталей*:

Открыть() : Boolean

Сохранить заказ() : Boolean

2. Используя браузер или диаграмму Классов, введите сигнатуру операций класса *Выбор_заказа*:

Создать() : Boo1ean

3. Используя браузер или диаграмму Классов, введите сигнатуру операций класса *Упр_заказами*:

Сохранить заказ(OrderID : Integer) : Boo1eап

4. Используя браузер или диаграмму Классов, введите сигнатуры операций класса *Упр_транзакциями*:

Сохранить заказ(OrderlD : Integer) : Boolean

Сохранить информацию() : Integer

Задание 5. Описание связей между классами

В этом упражнении определяются связи между классами, участвующими в варианте использования "Ввести новый заказ".

Постановка задачи

Чтобы найти связи, были просмотрены диаграммы Последовательности. Все взаимодействующие там классы нуждались в определении соответствующих связей на диаграммах Классов. После обнаружения связи были добавлены на диаграммы классов.

Добавление связей

Добавим связи к классам, принимающим участие в варианте использования "Ввести новый заказ".

Этапы выполнения упражнения

Настройка

2. Найдите в браузере диаграмму Классов "Ввод нового заказа",
3. Дважды щелкнув на диаграмме, откройте ее.
4. Проверьте, имеется ли в панели инструментов диаграммы кнопка Unidirectional Association (Однонаправленная ассоциация). Если ее нет, продолжите настройку, выполнив шаги 4 и 5. Если есть, приступайте к выполнению самого упражнения.
5. Щелкните правой кнопкой мыши на панели инструментов диаграммы и в открывшемся меню выберите пункт Customize(HacmpouTb),
6. Добавьте на панель кнопку Creates A Unidirectional Association (Создать однонаправленную ассоциацию).

Добавление ассоциаций

1. Нажмите кнопку Unidirectional Association панели инструментов.
2. Проведите ассоциацию от класса *выбор_заказа* к классу *заказ_деталей*.
3. Повторите шаги 1 и 2, создав ассоциации:

- От класса *заказ_деталей* к классу *упр_заказами*
- От класса *упр_заказами* к классу *Заказ*
- От класса *упр_заказами* к классу *упр_транзакциями*
- От класса *упр_транзакциями* к классу Заказ
- От класса *упр_транзакциями* к классу *Позиц_заказа*
- От класса *Заказ* к классу *Позиц_заказа*

4. Щелкните правой кнопкой мыши на однонаправленной ассоциации между классами *выбор_заказа* и *заказ_деталей* класса *выбор_заказа*.
5. В открывшемся меню выберите пункт Multiplicity > Zero or One (Множественность >- Нуль или один),
6. Щелкните правой кнопкой мыши на другом конце однонаправленной ассоциации.
7. В открывшемся меню выберите пункт Multiplicity > Zero or One (Множественность > Нуль или один),
8. Повторите шаги 4—7, добавив на диаграмму значения множественности для остальных ассоциаций, как показано на рис. 10

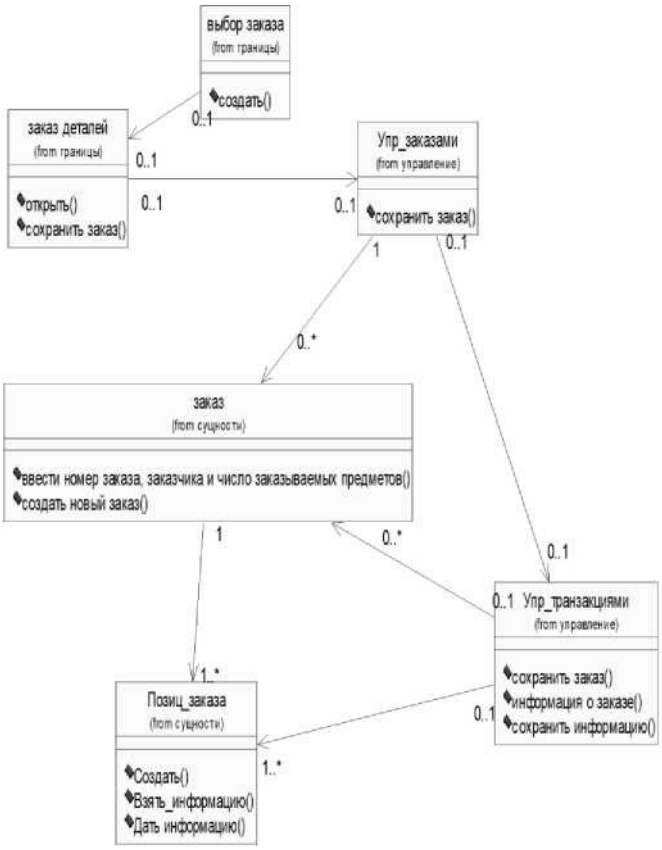


Рис. 10 Ассоциации сценария "Ввести новый заказ"

Задание 6. Исключение кириллизованного текста в информации классов.

Разработанные ранее модели, предназначенные для описания предметной области используют кириллизованную символику, недопустимую для большинства языков программирования. Выполните замену русского текста на латинский шрифт. Для этой цели сохраните предыдущую модель под другим именем и далее работайте с новым файлом (что бы при необходимости можно было бы вернуться к бизнес- процессам, описанным русским шрифтом).

Этапы выполнения

Этап 1. Используя меню (Файл-> Сохранить как) сохраните данную модель под другим именем (например Заказ1) в той же папке, что и исходная модель.

Работайте далее с копией модели (то есть Заказ1).

Этап 2. Переименуйте классы и их спецификации таким образом, чтобы использовался только латинский шрифт. Замените имя класса

Заказ_деталей на *OrderDetail*

Выбор заказа на *OrderOptions*

Заказ на *Order*

Упр_заказами на *OrderMgr*

Позиц_заказа на *OrderItem*

Упр_транзакциями на *TransactionMgr*

Измените имена операций таким образом, чтобы рис.10 преобразовался в рис. 11. Для этого, измените операцию класса *OrderOptions*

Открыть() на *Open()*

Класса *OrderDetail*

Открыть() на *Open()*

Сохранить заказ() на *Save()*

Класса *Order*

Ввести номер заказа, заказчика и число заказываемых предметов() на *SetInfo()*

Сохранить_заказ() на *Save()*

Класса *OrderMgr*

Сохранить заказ() на *SaveOrder()*

Класса *TransactionMgr*

Сохранить заказ() на *SaveOrder()*

Сохранить информацию о заказе() на *Commit()*

Создать_заказ() на *SubmitInfo()*

Класса *OrderItem*

Создать() на *Create()*

Взять_информацию() на *GetInfo()*

Дать_информацию на *SetInfo()*

Переименуйте имена пакетов

Границы на *Boundaries*

Сущности на *Entity*

Контроль на *Control*

Добавление стереотипов к классам

1. Щелкните правой кнопкой мыши на классе OrderOptions диаграммы.
2. В открывшемся меню выберите пункт Open Specification (Открыть спецификацию).
3. В поле стереотипа выберите из выпадающего списка слово Boundary (если его нет, то введите).
4. Нажмите на кнопку ОК.
5. Повторив шаги 1—4, свяжите классы OrderDetail со стереотипом Boundary, OrderMgr и TransactionMgr со стереотипом Control, а класс Order и OderItem— со стереотипом Entity.

Теперь диаграмма Классов должна иметь вид, показанный на рис. 11.

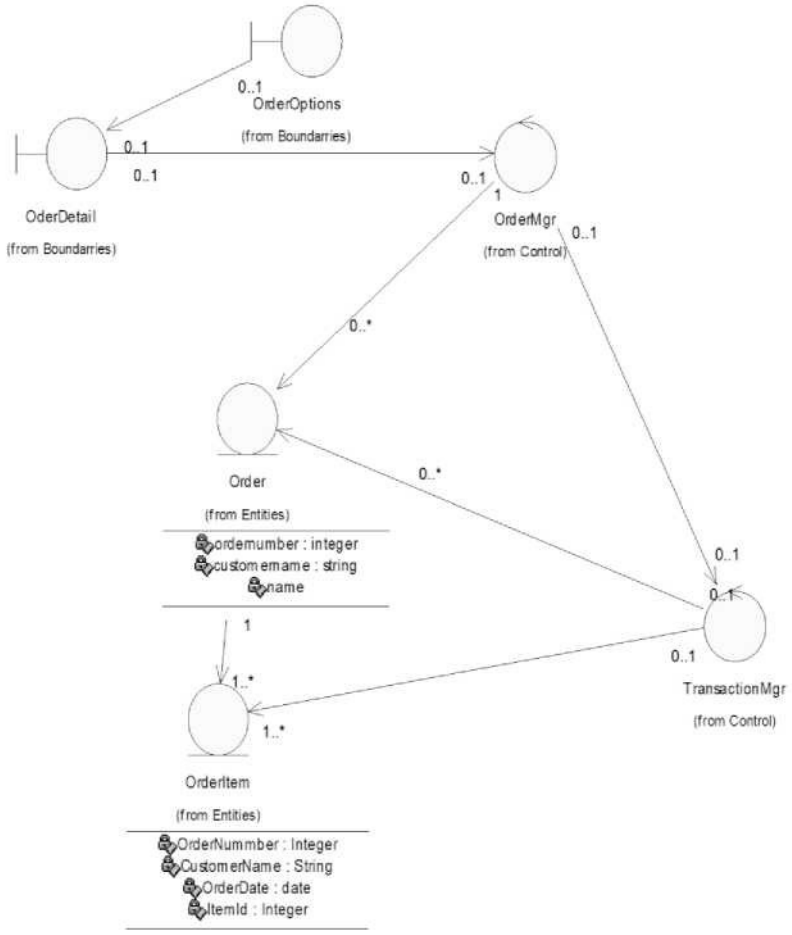


Рис. 11 Основная диаграмма классов
Замечание. На диаграмме рис. 11 возможно визуальное представление классов не в виде иконок, а в виде дополнительной строки текста с именем стереотипа. За этот вид отвечает метка установленная либо на icon либо на label (Class> Open Specefication> Options> Label)

Методические указания по ходу выполнения работы

Чётко следуем инструкциям.

Тема лабораторной работы № 4. «Построение диаграммы компонентов», объем часов 1

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.
Цель лабораторной работы: исследование процесса построения диаграммы компонентов и в заданной предметной области.

Задание(я):

Задание 1. Построение диаграммы Компонентов

Этапы выполнения

Так как эта модель связана с конкретным языком программирования, то в настройках это необходимо отметить. Выполнить Tools>Options>Notations>Default Language и из выпадающего списка языков программирования выбрать Delphi.

Создание пакетов компонентов

1. Щелкните правой кнопкой мыши на представлении компонентов в браузере.
2. В открывшемся меню выберите пункт New > Package (Создать > Пакет}.
3. Назовите пакет Entities (Сущности).
4. Повторив шаги с первого по третий, создайте пакеты Boundaries (Границы) и Control (Управление).

Добавление пакетов на Главную диаграмму Компонентов

1. Откройте Главную диаграмму Компонентов, дважды щелкнув на ней мышью,
2. Перетащите пакеты Entities, Boundary и Control из браузера на Главную диаграмму.

Отображение зависимостей между пакетами

1. Нажмите кнопку Dependency (Зависимость) панели инструментов.
2. Щелкните мышью на пакете Boundary Главной диаграммы Компонентов.
3. Проведите линию зависимости к пакету Control.
4. Повторив шаги 1 — 3, проведите зависимость от пакета Control к пакету Entities.

В результате диаграмма примет вид рис. 12

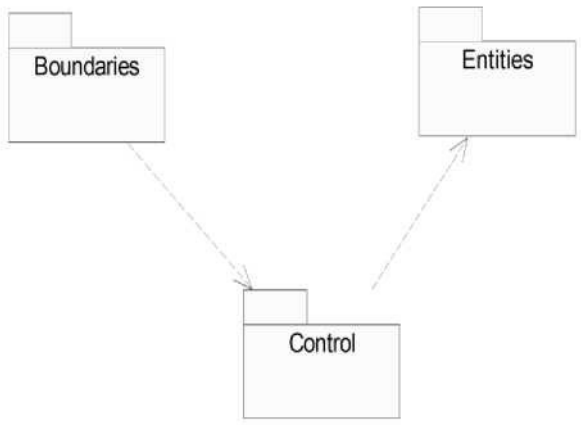


Рис. 12 Зависимости между пакетами

Добавление компонентов к пакетам и отображение зависимостей

1. Дважды щелкнув мышью на пакете Entities Главной диаграммы Компонентов, откройте Главную диаграмму Компонентов этого пакета.
2. Нажмите кнопку Package Specification (Спецификация пакета) панели инструментов.
3. Поместите спецификацию пакета на диаграмму.
4. Введите имя спецификации пакета — *OrderItem_*.
5. Повторив шаги 2—4, добавьте спецификацию пакета *Order_*.
6. Нажмите кнопку Dependency (Зависимость) панели инструментов.
7. Щелкните мышью на спецификации пакета *OrderItem_*.
8. Проведите линию зависимости к спецификации пакета *OrderItem_*.
9. С помощью описанного метода создайте следующие компоненты и зависимости:

Для пакета Boundaries:

- Спецификацию пакета *Orderoptions_*
- Спецификацию пакета *OrderDetail_*

Зависимости в пакете Boundaries:

• От спецификации пакета *Orderoptions_*
к спецификации пакета *OrderDetai_l*

Для пакета Control:

• Спецификацию пакета *OrderMgr_*

• Спецификацию пакета *TransactionMgr_*
Зависимости в пакете Control:

• От спецификации пакета *OrderMgr_*
к спецификации пакета *TransactionMg_*

Создание диаграммы Компонентов системы

1. Щелкните правой кнопкой мыши на представлении Компонентов в браузере.
2. В открывшемся меню выберите пункт New > Component Diagram (Создать > Диаграмма Компонентов).
3. Назовите новую диаграмму System.
4. Дважды щелкните на этой диаграмме мышью.

Размещение компонентов на диаграмме Компонентов системы

1. Разверните в браузере пакет компонентов *Entities*, чтобы открыть его.
2. Щелкните мышью на спецификации пакета *Order_* в пакете компонентов *Entities*.
3. Перетащите эту спецификацию на диаграмму.
4. Повторив шаги 2 и 3, поместите на диаграмму спецификацию пакета *OrderItem_*.
5. С помощью этого метода поместите на диаграмму следующие компоненты:

Из пакета компонентов *Boundaries*:

• Спецификацию пакета *Orderoptions_*
• Спецификацию пакета *OrderDetail_*
Из пакета компонентов Control:

• Спецификацию пакета *OrderMgr_*
• Спецификацию пакета *TransactionMgr_*

6. Нажмите кнопку Task Specification (Спецификация задачи) панели инструментов.
7. Поместите на диаграмму спецификацию задачи и назовите ее *OrderClientExe*.
8. Повторите шаги 6 и 7 для спецификации задачи *OrderServerExe*.

Добавление оставшихся зависимостей на диаграмму Компонентов системы

Уже существующие зависимости будут автоматически показаны на диаграмме Компонентов системы после добавления туда соответствующих компонентов. Теперь нужно добавить остальные зависимости.

1. Нажмите кнопку Dependency (Зависимость) панели инструментов.
2. Щелкните мышью на спецификации пакета *OrderDetail_*
3. Проведите линию зависимости к спецификации пакета *OrderDetail_*
4. Повторив шаги 1 — 3, создайте следующие зависимости:

• От спецификации пакета *OrderMgr_*
к спецификации пакета *Order_*

• От спецификации пакета *TransactionMgr_*
к спецификации пакета *OrderItem_*

• От спецификации пакета *TransactionMgr_*
к спецификации пакета *Order_*

• От спецификации задачи *OrderClientExe* к спецификации пакета *Orderoptions_*

От спецификации задачи *OrderServerExe* к спецификации пакета *OrderMg_r*

Соотнесение классов с компонентами

1. В Логическом представлении браузера найдите класс *Order* пакета *Entities*.
2. Перетащите этот класс на спецификацию пакета компонента *Order_* в представлении Компонентов браузера, В результате класс *Order* будет соотнесен со спецификацией пакета компонента *Order_*.
3. Повторив шаги 1 — 2, соотнесите классы со следующими компонентами:

- Класс *OrderItem* со спецификацией пакета *OrderItem_*
- Класс *Orderoptions* со спецификацией пакета *Orderoptions_*
- Класс *OrderDetail* со спецификацией пакета *OrderDetai_l*
- Класс *OrderMgr* со спецификацией пакета *OrderMgr_*
- Класс *TransactionMgr* со спецификацией пакета *TransactionMgr_*

В результате в браузере после имени класса, в скобках появятся имена компонентов, с которыми этот класс связан (рис. 13)

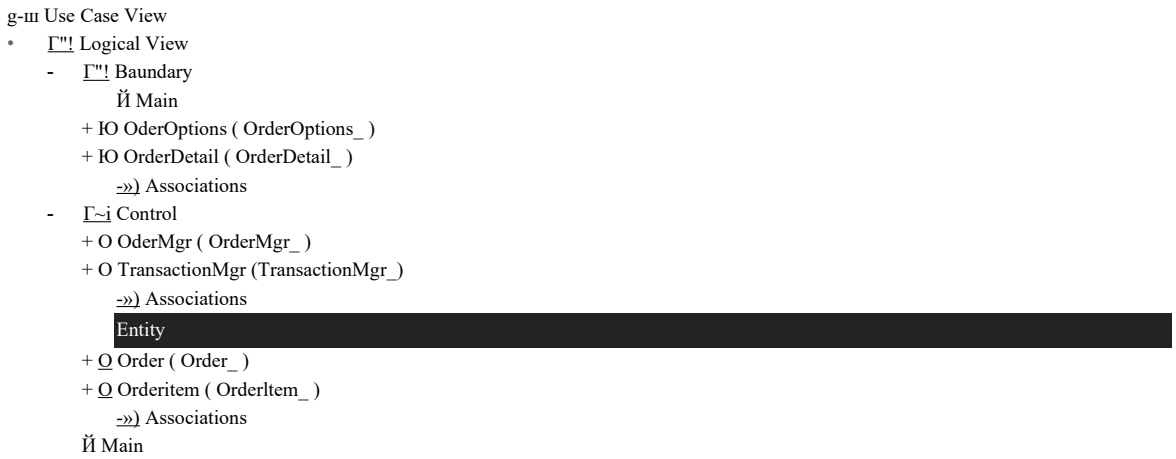


Рис. 13 Представление компонентов и классов в браузере

Методические указания по ходу выполнения работы

Чётко следуем инструкциям.

Тема лабораторной работы № 5. «Построение диаграмм потоков данных», объем часов 1

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: исследование процесса построения диаграммы потоков данных и диаграммы размещения в заданной предметной области.

Задание(я):

Задание 1. Кодогенерация проекта в Delphi.

Теперь вся информация подготовлена к тому, чтобы запрограммировать классы с их методами и операциями.

Для выполнения кодогенерации в среде Delphi необходимо выполнить следующую последовательность действий:

- протестировать модель на логические непротиворечия;
- настроить (или проверить настройки) среду на законы кодогенерации (соответствие элемента модели Rose элементу кода Delphi);
- создать имя проекта Delphi и выполнить кодогенерацию.

Этапы выполнения упражнения.

1) Протестируйте модель Tools->Ceck Model. Просмотрите log файл на наличие ошибок. Если файл не виден- выполните команду file->Save Log As и введите имя файла (по умолчанию error.log). Затем его просмотрите и, при необходимости, исправьте ошибки. К наиболее распространенным ошибкам относятся такие, как неотображение сообщений на операции или несоотнесение объектов с классами на диаграммах взаимодействия. С помощью пункта меню Check Model можно выявить большую часть неточностей и ошибок в модели.

2) Пункт меню Access Violations позволяет обнаружить нарушения правил доступа, возникающие тогда, когда существует связь между двумя классами разных пакетов. При этом связи между самими пакетами нет. Например, если существует связь между классами Order пакета Entities и OrderManager пакета Control, то обязательно должна существовать и связь между пакетами Entities и Control. Если последняя связь не установлена, Rose выявит нарушение правил доступа. Чтобы обнаружить нарушение правил доступа:

Выберите в меню Report > Show Access Violations.

Проанализируйте все нарушения правил доступа.

3) Выполните Tools>Options>Notation>Default Language и из выпадающего списка выберите язык программирования Delphi.

4) Проверьте правильность установок кодогенерации по умолчанию (default). Для этого выполните Tools->Options->Delphi и последовательно переберите из выпадающего списка поля Type все элементы. Сравните установки в поле Model Propities с данными (default) из таблицы Приложения А. В случае несоответствия- исправьте.

5) Выполните Tools> Ensemble Tools>Rose Delphi Link (рис.14)

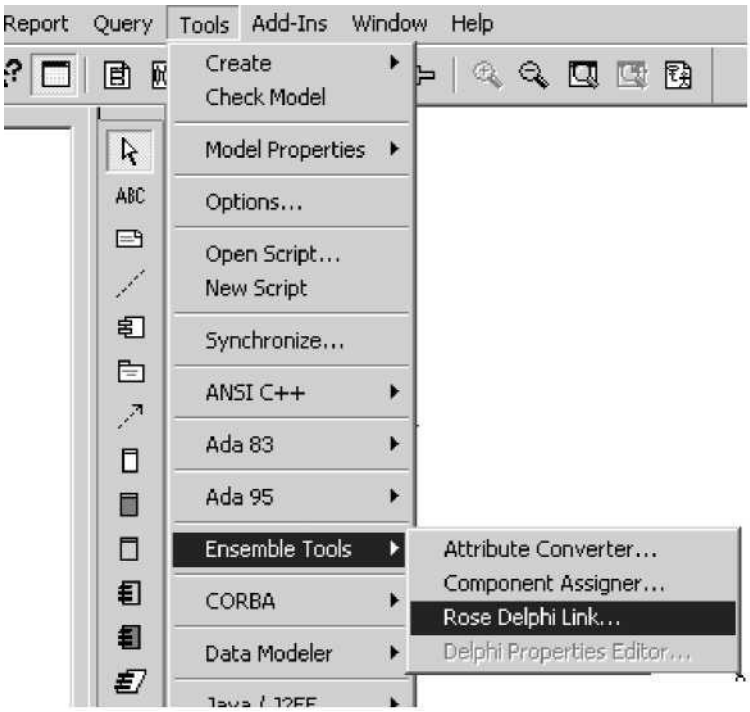


Рис. 14. Меню для выбора процесса кодогенерации

В результате появится соответствующая экранная форма. Выполните на этой форме File>New Proect. Появится форма с браузером. Введите имя файла и место на диске, куда будет сохранено имя сгенерированного проекта в Delphi. Например, NewProect.dpr и нажмите Открыть. В результате форма примет вид (рис. 15)

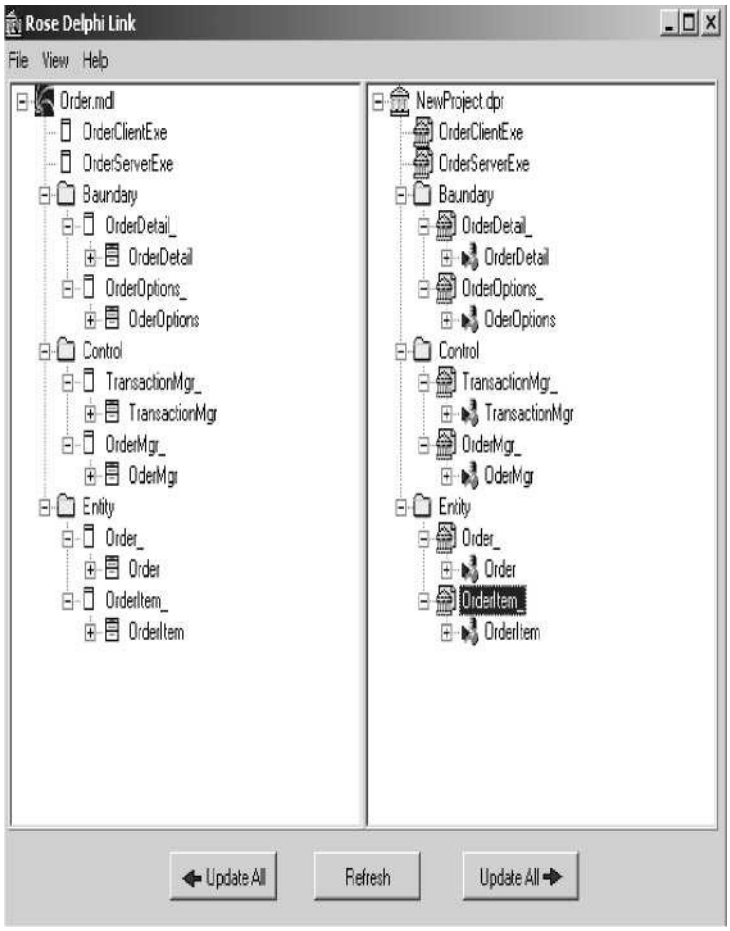


Рис. 15. Представление результатов кодогенерации в окне Rose Delphi Link

5. Через проводник Windows найдите папку проекта Delphi. С помощью программы Блокнот просмотрите содержимое всех файлов. В приложении В к руководству приведено содержимое всех файлов проекта.

Задание 2. Анализ Delphi проекта, добавление визуальных объектов, реинжениринг в Rose

1) Запустите на выполнение программу Delphi и загрузите сгенерированный проект (Proect1.dpr). Проверьте, что проект содержит все модули и присмотрите их содержимое через редактор Delphi.

2) Создайте в проекте Delphi новую форму с Name Form1. Поместите на форму компонент MainMenu (главное меню)

3) С помощью Menu Disigner введите две позиции горизонтального меню с названиями (полями Caption) Oder и OderItem.

4) Для Oder введите две строки вертикального меню с Caption Create и SubmitInfo. Для OderItem введите одну строку вертикального меню с названием GetInfo.

5) Сохраните проект в Delphi.

Реинжениринг Delphi проекта в модель Rose

1) Вернитесь в проект Rose и откройте окно проектов Rose Delphi Link. Проверьте, что открыт именно тот проект, для которого выполнялась кодогенерация.

2) Курсором мыши нажмите клавишу Update ALL со стрелкой влево (обновление модели Rose на основе изменений проекта Delphi). В результате в моделе Rose должны произойти определенные изменения (рис. 16)

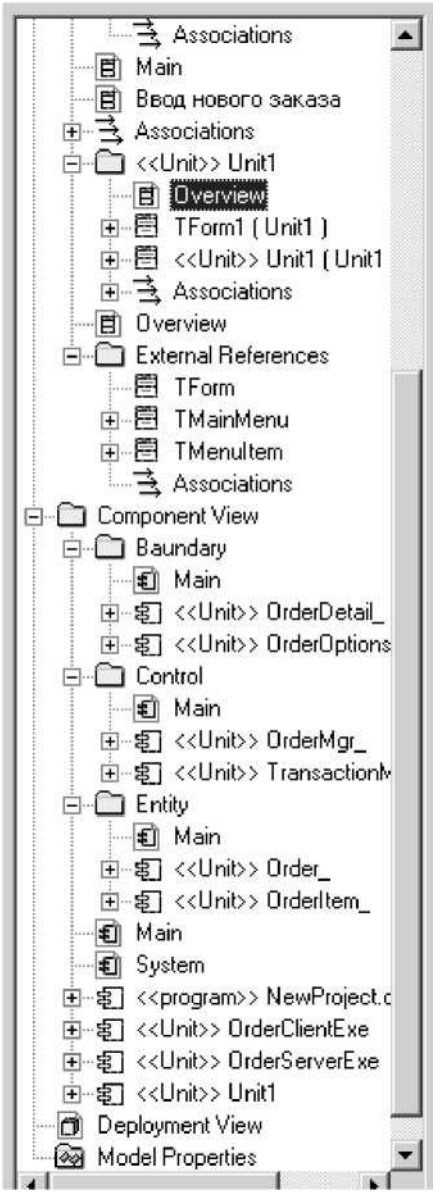


Рис. 16. Окно Rose Delphi Link после кодогенерации

- в представление Logic View создался новый пакет Unit1 и External References (Внешние ссылки). Внутри второго пакета созданы три класса TForm, TMainMenu и TMenuItem, которые использовались при развитии проекта Delphi. Отметим, что эта папка не создалась бы, если бы мы при первоначальном создании проекта включили в него пакет классов Delphi FreimWork.
- в этом же представлении в пакете Unit1 создан класс TForm1 и Unit1 оба соотнесенные с вновь созданным компонентом Unit1. Кроме того, в этом же пакете создалась диаграмма классов Overview, содержимое которой показано на рис.17

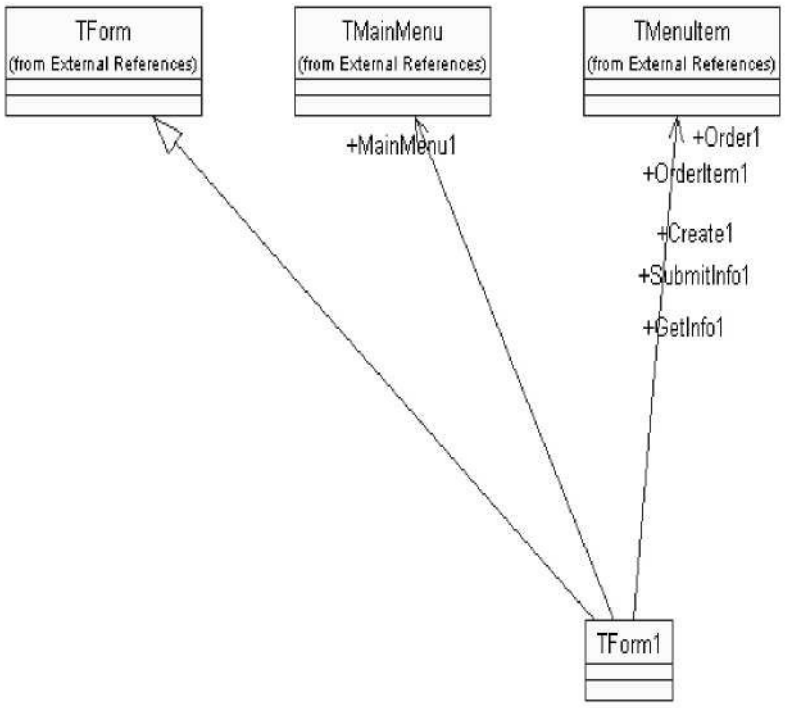


Рис. 17. Результаты реинжениринга проекта Delphi в Rose

Задание 3. Построение диаграммы размещения

В этом задании создается диаграмма Размещения для системы обработки заказов.

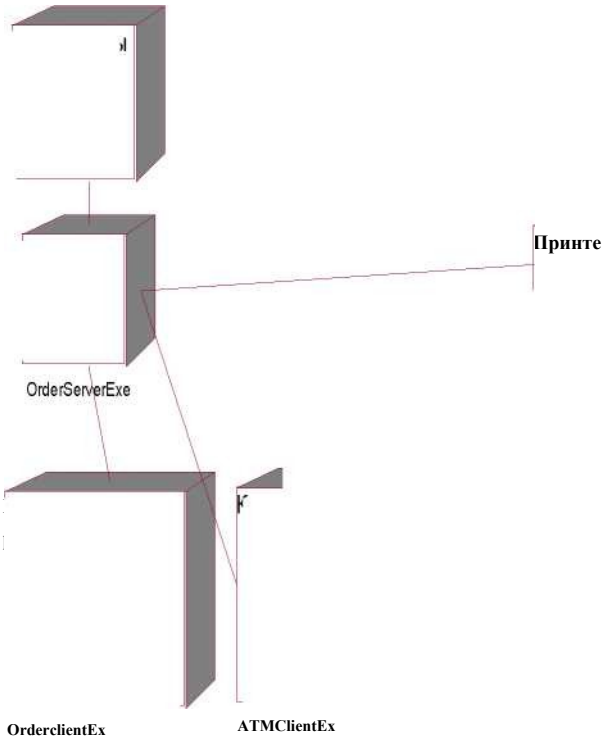


Рис. 18 Диаграмма размещения для модельной задачи

Добавление узлов к диаграмме Размещения

1. Дважды щелкнув мышью на представлении Размещения в браузере, откройте диаграмму Размещения.
2. Нажмите кнопку Processor (Процессор) панели инструментов.
3. Щелкнув мышью на диаграмме, поместите туда процессор.
4. Введите имя процессора "Сервер базы данных".
5. Повторив шаги 2—4, добавьте следующие процессоры:

-Сервер приложения

- Клиентская рабочая станция №1
- Клиентская рабочая станция №2

6. На панели инструментов нажмите кнопку Devices (Устройство).
7. Щелкнув мышью на диаграмме, поместите туда устройство.
8. Назовите его "Принтер".

Добавление связей

1. Нажмите кнопку Connection (Связь) панели инструментов.
2. Щелкните мышью на процессоре "Сервер базы данных".
3. Проведите линию связи к процессору "Сервер приложения".
4. Повторив шаги 1 — 3, добавьте следующие связи;

- От процессора "Сервер приложения" к процессору "Клиентская рабочая станция №1"
- От процессора "Сервер приложения" к процессору "Клиентская рабочая станция №2"
- От процессора "Сервер приложения" к устройству "Принтер"

Добавление процессов

1. Щелкните правой кнопкой мыши на процессоре "Сервер приложения" в браузере.
2. В открывшемся меню выберите пункт New > Process (Создать > Процесс),
3. Введите имя процесса — OrderServerExe.
4. Повторив шаги 1 —- 3, добавьте процессы:

- Процесс OrderclientExe на процессоре "Клиентская рабочая станция №1"
- Процесс ATMClientExe на процессоре "Клиентская рабочая станция №2"

Показ процессов на диаграмме

1. Щелкните правой кнопкой мыши на процессоре "Сервер приложения".
2. В открывшемся меню выберите пункт Show Process (Показать процессы).
3. Повторив шаги 1 и 2, покажите процессы на следующих процессорах:

- Клиентская рабочая станция №1
- Клиентская рабочая станция №2

Методические указания по ходу выполнения работы

Чётко следуем инструкциям.

Тема лабораторной работы № 6. «Разработка тестового сценария», объем часов 2

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: усвоить знание о видах тестирования; освоить способы обнаружения и
фиксирования ошибок.

Задание(я):

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

Программа работы

1. Разработать интерфейс приложения и написать программные коды для событий кнопок.
2. Сохранить проект в отдельной папке, скопировать исполняемый файл на рабочий стол.
3. Составить тесты для проверки работы приложения.
4. Провести тестирование исполняемого файла

Составить отчет по итогам тестирования и рекомендации по устранению выявленных ошибок

Методические указания по ходу выполнения работы

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

Уровни тестирования:

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

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

Системное тестирование**.** Тестируется интегрированная система на ее соответствие

исходным требованиям.

Таблица 1. Виды некоторых ошибок и способы их обнаружения

Виды программных ошибок	Способы их обнаружения

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

Тест - это набор контрольных входных данных совместно с ожидаемыми результатами. Тесты должны обладать определенными свойствами.

Детективность: тест должен с большой вероятностью обнаруживать возможные ошибки.

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

Воспроизводимость: ошибка должна выявляться независимо от изменяющихся условий.

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

Тема лабораторной работы № 7. «Оценка необходимого количества тестов», объем часов 2

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: получить навыки разработки тестовых сценариев.

Задание(я):

Задание № 1

Написать программу решения квадратного уравнения ax2 + Ьx + c = 0.

Задание № 2

Найти минимальный набор тестов для программы нахождения веще-ственных корней квадратного уравнения ax2 + Ьx + c = 0. Решение представлено в таблице.

Номер теста	a	Ь	c	Ожидаемый результат	Что проверяется
1	2	-5	2	xi=2, X2=O,5	Случай вещественных корней
2	3	2	5	Сообщение	Случай комплексных корней
3	3	-12	0	xi=4, X2=0	Нулевой корень
4	0	0	10	Сообщение	Неразрешимое уравнение
5	0	0	0	Сообщение	Неразрешимое уравнение
6	0	5	17	Сообщение	Неквадратное уравнение
7	9	0	0	**X|-X2=0**	Нулевые корни

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

Заповеди по отладки программного средства, предложенные Г. Майерсом.

Заповедь 1. Считайте тестирование ключевой задачей разработки ПС, поручайте его самым квалифицированным и одаренным программистам, нежелательно тестировать свою собственную программу.

Заповедь 2. Хорош тот тест, для которого высока вероятность обнаружить ошибку, а не тот, который демонстрирует правильную работу программы.

Заповедь 3. Готовьте тесты как для правильных, так и для неправильных данных.

Заповедь 4. Документируйте пропуск тестов через компьютер, детально изучайте результаты каждого теста, избегайте тестов, пропуск которых нельзя повторить. Заповедь 5. Каждый модуль подключайте к программе только один раз, никогда не изменяйте программу, чтобы облегчить ее тестирование.

Заповедь 6. Пропускайте заново все тесты, связанные с проверкой работы какой-либо программы ПС или ее взаимодействия с другими программами, если в нее были внесены изменения (например, в результате устранения ошибки).

Задание № *3*

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

Набор тестовых сценариев запишите в виде таблицы, приведенной выше.

Задание № 4

Оформить отчет.

Методические указания по ходу выполнения работы

- Оценка стоимости и причины ошибок в программном обеспечении.
- Виды и методы тестирования.
- Понятие теста.
- Требования к разработке тестовых сценариев.
- Правила разработки тестовых сценариев.

Тема лабораторной работы № 8. «Разработка тестовых пакетов», объем часов 2

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: получить навыки разработки тестовых пакетов.

Задание(я):

Задание № 1

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

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

	1	2	3	4	5
1	A	B	c	D	E
2	F	G	н	I. J	K
3	L	M	N	O	P
4	Q	R	S	T	и
5	V	W	X	Y	Z

Пользуясь изложенным способом создать программу, которая:

а) зашифрует введенный текст и сохранит его в файл;
б) считает зашифрованный текст из файла и расшифрует данный текст.

Задание № 2

Спроектировать тесты по принципу «белого ящика» для программы, разработанной в задании № 1. Выбрать несколько алгоритмов для тестирования и обозначить буквами или

цифрами ветви этих алгоритмов. Выписать пути алгоритма, которые должны быть проверены тестами для выбранного метода тестирования. Записать тесты, которые позволят пройти по путям алгоритма. Протестировать разработанную вами программу. Результаты оформить в виде таблиц:

Тест	Ожидаемый результат	Фактический результат	Результат тестирования

Задание № 3

Проверить все виды тестов и сделать выводы об их эффективности

Задание № 4

Оформить отчет.

Методические указания по ходу выполнения работы

- Системные основы разработки требований к сложным комплексам программ.
- Формализация эталонов требований и характеристик комплекса программ.
- Формирование требований компонентов и модулей путем декомпозиции функций комплексов программ.
- Тестирование по принципу «белого ящика».

Тема лабораторной работы № 9. «Оценка программных средств с помощью метрик», объем часов 1

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

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

Задание(я):

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

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

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

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

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

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

2. Установить веса показателей wi (?wi =1).
3. Для каждого показателя установить конкретную численную оценку ri от 0 до 1, исходя из следующего:

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

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

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

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

общее количество показателей

Результатом выполнения данной работы является отчет

Методические указания по ходу выполнения работы

Необходимая документация**:** ГОСТ 28.195-89

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

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

ГОСТ 28.195-89 «Оценка качества программных средств. Общие положения» устанавливает общие положения по оценке качества программных средств, номенклатуру и применяемость показателей качества.

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

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

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

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

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

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

Определение значений показателей качества ПС *экспертным методом* осуществляется группой экспертов-специалистов, компетентных в решении данной задачи, на базе их опыта и интуиции. Экспертный метод применяется в случаях, когда задача не может быть решена никаким другим из существующих способов или другие способы являются значительно более трудоемкими. Экспертный метод рекомендуется применять при определении показателей наглядности, полноты и доступности программной документации, легкости освоения, структурности.

Социологические методы основаны на обработке специальных анкет-вопросников.

Рис. 1 - Уровни системы показателей качества

Показатели качества объединены в систему из четырех уровней. Каждый вышестоящий уровень содержит в качестве составляющих показатели нижестоящих уровней (рисунок 1).

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

Определенные настоящим стандартом характеристики дополнены рядом требований по выбору метрик и их измерению для различных проектов ПС. Они применимы к любому типу ПС, включая компьютерные программы и данные, содержащиеся в программируемом оборудовании. Эти характеристики обеспечивают согласованную терминологию для анализа качества ПС. Кроме того, они определяют схему для выбора и специфицирования требований к качеству ПС, а также для сопоставления возможностей различных программных продуктов, таких как функциональные возможности, надежность, практичность и эффективность.

Все множество атрибутов качества ПС может быть классифицировано в структуру иерархического дерева характеристик и субхарактеристик. Самый высший уровень этой структуры состоит из характеристик качества, а самый нижний уровень - из их атрибутов. Эта иерархия не строгая, поскольку некоторые атрибуты могут быть связаны с более чем одной субхарактеристикой. Таким же образом, внешние свойства (такие, как пригодность, корректность, устойчивость к ошибкам или временная эффективность) влияют на наблюдаемое качество. Недостаток качества в использовании (например, пользователь не может закончить задачу) может быть прослежен к внешнему качеству (например, функциональная пригодность или простота использования) и связанным с ним внутренним атрибутам, которые необходимо изменить.

Внутренние метрики могут применяться в ходе проектирования и программирования к неисполняемым компонентам ПС (таким, как спецификация или исходный программный текст). При разработке ПС промежуточные продукты следует оценивать с использованием внутренних метрик, которые измеряют свойства программ, и могут быть выведены из моделируемого поведения. Основная цель внутренних метрик - обеспечивать, чтобы было достигнуто требуемое внешнее качество. Внутренние метрики дают возможность пользователям, испытателям и разработчикам оценивать качество ЖЦ программ и заниматься вопросами технологического обеспечения качества задолго до того, как ПС становится готовым исполняемым продуктом.

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

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

Когда требования к качеству ПС определены, в них должны быть перечислены характеристики и субхарактеристики, которые составляют полный набор показателей качества. Затем определяются подходящие внешние метрики и их приемлемые диапазоны значений, устанавливающие количественные и качественные критерии, которые подтверждают, что ПС

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

Метрики качества в использовании измеряют, в какой степени продукт удовлетворяет потребности конкретных пользователей в достижении заданных целей с результативностью, продуктивностью и удовлетворением в заданном контексте использования. При этом результативность подразумевает точность и полноту достижения определенных целей пользователями при применении ПС; продуктивность соответствует соотношению израсходованных ресурсов и результативности при эксплуатации ПС, а удовлетворенность - психологическое отношение к качеству использования продукта. Эта метрика не входит в число шести базовых характеристик ПС, регламентируемых стандартом ИСО 9126, однако рекомендуется для интегральной оценки результатов функционирования комплексов программ.

Оценивание качества в использовании должно подтверждать его для определенных сценариев и задач, оно составляет полный объединенный эффект характеристик качества ПС для пользователя. *Качество в использовании* - это восприятие пользователем качества системы, содержащей ПС, и оно измеряется скорее в терминах результатов использования комплекса программ, чем собственных внутренних свойств ПС. Связь качества в использовании с другими характеристиками качества ПС зависит от типа пользователя, так, например, для конечного пользователя качество в использовании обусловливают, в основном, характеристики функциональных возможностей, надежности, практичности и эффективности, а для персонала сопровождения ПС качество в использовании определяет сопровождаемость. На качество в использовании могут влиять любые характеристики качества, и это понятие шире, чем практичность, которая связана с простотой использования и привлекательностью. Качество в использовании, в той или иной степени, характеризуется сложностью применения комплекса программ, которую можно описать трудоемкостью использования с требуемой результативностью. Многие характеристики и субхарактеристики ПС обобщенно отражаются неявными технико-экономическими показателями, которые поддерживают функциональную пригодность конкретного ПС. Однако их измерение и оценка влияния на показатели качества, представляет сложную проблему.

Тема лабораторной работы № 10. «Инспекция программного кода на предмет соответствия стандартам кодирования», объем часов 1

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

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

Задание(я):

Задание №1

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

Задание №2

Используя шаблоны рефакторинга, выполнить реорганизацию программного кода разрабатываемого ПО.

Задание №3

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

Задание №4

В случае необходимости скорректировать проектную документацию.

Задание №5

Сделать выводы по результатам выполнения работ.

Класс Main

Было:

пакетная игра;

импорт игры. Персонажи. *;

импорт игры. Персонажи. Персонажи;

импорт игры. Энергетика. Энергетика;

импорт игры. Энергетика. Освещение;

импорт игры. Уровни. Блок;

импорт игры. Уровни. Уровень;

импортировать game.Levels.Level_data;

импорт игры. Weapon.Bullet;

импорт игры. Оружие. Оружие;

import javafx.animation.AnimationTimer;

импорт javafx.application.Application;

import javafx.scene.Scene;

import javafx.scene.image.Image;

```
import javafx.scene.input.KeyCode;
import javafx.scene.layout.Pane;
import javafx.stage.Stage;
import java.io.DataInputStream;
import java.io.FileInputStream;
импорт java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
публичный класс Main расширяет приложение {
public static ArrayList <Block> blocks = new ArrayList <> ();
public static ArrayList <Bullet> bullets = new ArrayList <> ();
public static ArrayList <Bullet> eparaBullets = новый ArrayList <> ();
public static ArrayList <EnemyBase> враги = новый ArrayList <> ();
static HashMap <KeyCode, Boolean> keys = new HashMap <> ();
публичная статическая сцена этапа;
публичная статическая сцена;
public static Pane gameRoot = new Pane ();
public static Pane appRoot = new Pane ();
публичное статическое меню;
публичный статический Персонаж букера;
публичная статическая HUD HUD;
статическое оружие общего назначения;
публичная статика елизавета елизавета;
статический VendingMachine vendingMachine;
статическое учебное пособие;
частные статические CutScenes cutScene;
публичная статика Энергетика энергетика;
публичная статическая молния молнии;
public static int levelNumber;
уровень статического уровня;
```

```
public static AnimationTimer timer = new AnimationTimer () {

@Override

public void handle (давно) {

Обновить();
}
};

приватное статическое void update () {

для (EnemyBase враг: враги) { enemy.update ();
if (epara.getDelete ()) { enemies.remove (враг); перемена;
}
}
Bullet.update ();

Controller.update ();

booker.update ();

if (! energetic.getName (). equals (""))

energetic.update ();

if (levelNumber> 0) elizabeth.update (); если (молния! = ноль) { lightning.update ();

if (lightning.getDelete ())

молния = ноль;
}
menu.update ();

hud.update ();

weapon.update ();

if (booker.getTranslateX ()> Level_data.BLOCK_SIZE * 295)

cutScene = новые CutScenes (levelNumber);
}
@Override

public void start (Stage primaryStage) выдает исключение {

stage = primaryStage;

сцена = новая сцена (appRoot, 1280, 720);

. AppRoot.getChildren () добавить (gameRoot);

уровень = новый уровень ();
```

```
try (DataInputStream dataInputStream = new DataInputStream (новый FileInputStream ("C:
/DeadShock/saves/data.dat"))) {

levelNumber = dataInputStream.readInt ();

level.createLevels (levelNumber);

level.changeImageView (levelNumber);

vendingMachine = new VendingMachine ();

букер = новый персонаж ();

booker.setMoney (dataInputStream.readInt ());

booker.setSalt (dataInputStream.readByte ());

booker.setCountLives (2);

оружие = новое оружие ();

weapon.setWeaponClip (dataInputStream.readInt ());

weapon.setBullets (dataInputStream.readInt ());

hud = новый HUD ();
Елизавета = новая Елизавета ();

энергичный = новый Энергетический ();

} catch (IOException e) {

levelNumber = 0;

level.createLevels (levelNumber);

vendingMachine = new VendingMachine ();

букер = новый персонаж ();

оружие = новое оружие ();

hud = новый HUD ();

энергичный = новый Энергетический ();

tutorial = new Tutorial (levelNumber);
}
switch (levelNumber) {

случай 0:

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 117, 200));
```

---

*Level_data.enemyBlocks.add (новый Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый Level_data.BLOCK_SIZE * 13));
Level_data.enemyBlocks.add (новый Level_data.BLOCK_SIZE * 9));
Level_data.enemyBlocks.add (новый Level_data.BLOCK_SIZE * 9));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 127, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 148, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 161, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 171, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 185, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 204, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 215, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 228, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 233, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 243, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 252, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 262, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 277, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 280, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 286, 200));

перемена;

Дело 1:

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 57, 200));

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 67, 200));

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 74, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 87, 200));

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 104, 150));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 117, 200));

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 133, 200));

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 156, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 177, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 193, 200));

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 201, 200));

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 216, 200));

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 224, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 246, 200));

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 260, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 277, 100));

```
34,
блок («невидимый», Level_data.BLOCK_SIZE
36,
блок («невидимый», Level_data.BLOCK_SIZE
60,
блок («невидимый», Level_data.BLOCK_SIZE
61
блок («невидимый», Level_data.BLOCK_SIZELevel_data.enemyBlocks.add (новый блок
(«невидимый», Level_data.BLOCK_SIZE * 106, Level_data.BLOCK_SIZE * 7));

        Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 107,
Level_data.BLOCK_SIZE * 7));

        Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 168,
Level_data.BLOCK_SIZE * 13));

        Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 170,
Level_data.BLOCK_SIZE * 13));

        Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 196,
Level_data.BLOCK_SIZE * 13));

        Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 197,
Level_data.BLOCK_SIZE * 13));

        Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 232,
Level_data.BLOCK_SIZE * 8));

        Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 233,
Level_data.BLOCK_SIZE * 8));

        перемена;
        }
        меню = новое меню ();

        appRoot.getChildren () добавить (menu.menuBox).

        booker.translateXProperty (). addListener (((наблюдаемый, oldValue, newValue) -> {

        int offset = newValue.intValue ();

        if (offset> 600 && offset <gameRoot.getWidth () - 680) {

        gameRoot.setLayoutX (- (смещение - 600));

        level.getBackground (). setLayoutX ((смещение - 600) / 1,5);
        }
        если (смещение <= 100)

        level.getBackground () setLayoutX (0).

        }));

        vendingMachine.createButtons ();

        stage.getIcons (). add (новое изображение ("файл: / C: /DeadShock/images/icon.jpg"));

        stage.setTitle ( "DeadShock");
```

```
stage.setResizable (ложь);

stage.setWidth (scene.getWidth ());

stage.setHeight (scene.getHeight ());

stage.setScene (сцены);

stage.show ();

timer.start ();
}
public static void main (String [] args) {

запуск (арг);
}
}
```

Стало:

```
пакетная игра;

импорт игры. Персонажи. *;

импорт игры. Персонажи. Персонажи;

импорт игры. Энергетика. Энергетика;

импорт игры. Энергетика. Освещение;

импорт игры. Уровни. Блок;

импорт игры. Уровни. Уровень;

импортировать game.Levels.Level_data;

импорт игры. Weapon.Bullet;

импорт игры. Оружие. Оружие;

import javafx.animation.AnimationTimer;

импорт javafx.application.Application;

import javafx.scene.Scene;
import javafx.scene.image.Image;

import javafx.scene.input.KeyCode;

import javafx.scene.layout.Pane;

import javafx.stage.Stage;

import java.io.DataInputStream;

import java.io.FileInputStream;
```

```
импорт java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
публичный класс Main расширяет приложение {
public static ArrayList <Block> blocks = new ArrayList <> ();
public static ArrayList <Bullet> bullets = new ArrayList <> ();
public static ArrayList <Bullet> eparaBullets = новый ArrayList <> ();
public static ArrayList <EnemyBase> враги = новый ArrayList <> ();
static HashMap <KeyCode, Boolean> keys = new HashMap <> ();
публичная статическая сцена этапа;
публичная статическая сцена;
public static Pane gameRoot = new Pane ();
public static Pane appRoot = new Pane ();
публичное статическое меню;
публичный статический Персонаж букера;
публичная статическая HUD HUD;
статическое оружие общего назначения;
публичная статика елизавета елизавета;
статический VendingMachine vendingMachine;
статическое учебное пособие;
частные статические CutScenes cutScene;
публичная статика Энергетика энергетика;
публичная статическая молния молнии;
public static int levelNumber;
уровень статического уровня;
public static AnimationTimer timer = new AnimationTimer () {
@Override
public void handle (давно) {
Обновить();
}
```

```
        };

        private void initContent () {

        . AppRoot.getChildren () добавить (gameRoot);

        уровень = новый уровень ();

        try (DataInputStream dataInputStream = new DataInputStream (новый FileInputStream ("C:
/DeadShock/saves/data.dat"))) {

        levelNumber = dataInputStream.readInt ();

        level.createLevels (levelNumber);

        level.changeImageView (levelNumber);

        vendingMachine = new VendingMachine ();

        букер = новый персонаж ();

        booker.setMoney (dataInputStream.readInt ());

        booker.setSalt (dataInputStream.readByte ());

        booker.setCountLives (2);

        оружие = новое оружие ();

        weapon.setWeaponClip (dataInputStream.readInt ());

        weapon.setBullets (dataInputStream.readInt ());

        hud = новый HUD ();

        Елизавета = новая Елизавета ();

        энергичный = новый Энергетический ();

        } catch (IOException e) {
        levelNumber = 0;

        level.createLevels (levelNumber);

        vendingMachine = new VendingMachine ();

        букер = новый персонаж ();

        оружие = новое оружие ();

        hud = новый HUD ();

        энергичный = новый Энергетический ();

        tutorial = new Tutorial (levelNumber);
        }
        createEnemies ();
```

```
меню = новое меню ();

appRoot.getChildren () добавить (menu.menuBox).

booker.translateXProperty (). addListener (((наблюдаемый, oldValue, newValue) -> { int offset
= newValue.intValue ();

if (offset> 600 && offset <gameRoot.getWidth () - 680) {

gameRoot.setLayoutX (- (смещение - 600));

level.getBackground (). setLayoutX ((смещение - 600) / 1,5);
}
если (смещение <= 100)

level.getBackground () setLayoutX (0).

}));

vendingMachine.createButtons ();
}
public static void createEnemies () {

switch (levelNumber) {

случай 0:

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 117, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 127, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 148, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 161, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 171, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 185, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 204, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 215, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 228, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 233, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 243, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 252, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 262, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 277, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 280, 200));
```

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 286, 200));

перемена;

Дело 1:

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 57, 200));

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 67, 200));

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 74, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 87, 200));

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 104, 150));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 117, 200));

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 133, 200));

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 156, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 177, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 193, 200));

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 201, 200));

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 216, 200));

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 224, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 246, 200));

враги.адд (новый EnemyRedEye (Level_data.BLOCK_SIZE * 260, 200));

враги.адд (новый EnemyComstock (Level_data.BLOCK_SIZE * 277, 100));

Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 34,
Level_data.BLOCK_SIZE * 13));

Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 36,
Level_data.BLOCK_SIZE * 13));

Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 60,
Level_data.BLOCK_SIZE * 9));

Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 61,
Level_data.BLOCK_SIZE * 9));

Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 106, Level_data.BLOCK_SIZE * 7));

```
Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 107,
Level_data.BLOCK_SIZE * 7));

Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 168,
Level_data.BLOCK_SIZE * 13));

Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 170,
Level_data.BLOCK_SIZE * 13));

Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 196,
Level_data.BLOCK_SIZE * 13));

Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 197,
Level_data.BLOCK_SIZE * 13));

Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 232,
Level_data.BLOCK_SIZE * 8));

Level_data.enemyBlocks.add (новый блок («невидимый», Level_data.BLOCK_SIZE * 233,
Level_data.BLOCK_SIZE * 8));

перемена;
}
}
приватное статическое void update () { для (EnemyBase враг: враги) { enemy.update ();

If (enemy.GetDelete()) { enemies.remove(enemy); break;
}
}
Bullet.update();

Controller.update();

booker.update();

If (!energetic.GetName().Equals("")) energetic.update();

if (levelNumber > 0) elizabeth.update();

if (lightning != null) { lightning.update();

If (lightning.GetDelete()) lightning = null;
}
menu.update();

hud.update();

weapon.update();

if (booker.getTranslateX() > Level_data.BLOCK_SIZE * 295) cutScene = new
CutScenes(levelNumber);
}
@Override
```

```
public void start(Stage primaryStage) throws Exception {

stage = primaryStage;

scene = new Scene(appRoot, 1280, 720);

initContent();

stage.getIcons().add(new Image("file:/C:/DeadShock/images/icon.jpg"));

stage.setTitle("DeadShock");

stage.setResizable(false);

stage.setWidth(scene.getWidth());

stage.setHeight(scene.getHeight());

stage.setScene(scene);

stage.show();

timer.start();

public static void main(String[] args) { launch(args);
```

Шаблон рефакторинга: Выделение Memoga(Extract Method)

Методические указания по ходу выполнения работы
Чётко следуем инструкциям.

Тема лабораторной работы № 11. «Разработка структуры проекта», объем часов 2

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

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

Задание(я):

Задание

1. Выбрать вариант задания на проектирование и разработку учебной программы.
2. В соответствии с вариантом выполнить разработку технического задания, которое должно включать:
   - o введение;
   - o основание для разработки;
   - o назначение;
   - o требования к программе и программному продукту;
   - o требования к программной документации.
3. Оформить отчет. Содержание отчета:

o тема лабораторной работы

o цель лабораторной работы

o ответы на контрольные вопросы

o задание на лабораторную работу

o разработанное техническое задание

o выводы по проделанной работе.

Варианты заданий

1. Ввести вещественную матрицу размерности n * m построчно, а вывести по столбцам.
2. Выяснить сколько положительных элементов содержит матрица размерности n * m , если a ij = sin(i+j/2).
3. Дана квадратная вещественная матрица размерности n. Является ли матрица симметричной относительно главной диагонали.
4. Дана квадратная вещественная матрица размерности n. Транспонировать матрицу.
5. Дана квадратная вещественная матрица размерности n. Сравнить сумму элементов матрицы на главной и побочной диагоналях.
6. Дана квадратная вещественная матрица размерности n. Найти количество нулевых элементов, стоящих:
   выше главной диагонали;

   ниже главной диагонали;

   выше и ниже побочной.

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

   элементы k строки матрицы упорядочены по убыванию;

   k строка массива симметрична.

8. Дана вещественная матрица размерности n * m. Сформировать вектор b, в котором элементы вычисляются как:
   произведение элементов соответствующих строк;

   среднее арифметическое соответствующих столбцов;

   разность наибольших и наименьших элементов соответствующих строк;

   значения первых отрицательных элементов в столбце.

9. Дана вещественная матрица размерности n * m. Вывести номера столбцов, содержащих только отрицательные элементы.
10. Дана вещественная матрица размерности n * m. Вывести номера строк, содержащих больше положительных элементов, чем отрицательных.
11. Дана вещественная матрица размерности n * m. Найти общую сумму элементов только тех столбцов, которые имеют хотя бы один нулевой элемент.
12. Дана вещественная матрица размерности n * m. Поменять местами строки с максимальным и минимальным элементами.
13. Дана вещественная матрица размерности n * m. Удалить k столбец матрицы.

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

    по столбцам.

15. Дана вещественная матрица размерности m * n. Упорядочить элементы каждой четной строки по возрастанию.
16. Дана вещественная матрица размерности m * n. Расположить все элементы матрицы по убыванию. Обход матрицы осуществлять по строкам.
17. Дана вещественная матрица размерности m * n. Определить индексы первого нулевого элемента матрицы. Обход матрицы осуществлять по столбцам.
18. Известно положение двух ферзей на шахматной доске. Бьют ли они друг друга?

Методические указания по ходу выполнения работы

Чётко следуем инструкциям.

Тема лабораторной работы № 12. «Разработка модульной структуры проекта (диаграммы модулей)», объем часов 1

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: Изучить основы разработки модульной структуры проекта (диаграммы модулей).

Задание(я):

1. Разработайте проект автоматизации библиотечного каталога.
2. Проведите анализ работы деканата и разработайте проект его автоматизации.

Проанализируйте информационные потоки вашего факультета и спроектируйте компьютерную систему их обработки.

Методические указания по ходу выполнения работы

Разработка эскизного проекта

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

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

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

Разработка технического проекта

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

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

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

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

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

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

Рабочий проект

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

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

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

Тема лабораторной работы № 13. «Разработка перечня артефактов и протоколов проекта», объем часов 1

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: разработка перечня артефактов и протоколов проекта.

Задание(я):

1. В соответствии с подготовленным техническим заданием выполнить разработку спецификаций на программный продукт, которые должны включать:
   - спецификации процессов;
   - словарь терминов;
   - диаграммы переходов состояний;
   - диаграммы потоков с детализацией.
2. Оформить отчет. Содержание отчета:
   - тема лабораторной работы;
   - цель лабораторной работы;
   - ответы на контрольные вопросы;
   - задание на лабораторную работу;
   - разработанные спецификации процессов;
   - словарь терминов;
   - диаграммы переходов состояний;
   - диаграммы потоков с детализацией;
   - выводы по проделанной работе.

Методические указания по ходу выполнения работы

Системный анализ и пути решения задачи

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

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

Этап системного анализа состоит из следующих трех стадий:

1. обоснование необходимости разработки программы;
2. научно-исследовательские работы (НИР);
3. разработка и утверждение технического задания.

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

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

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

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

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

1. *Водопадный подход*. При таком подходе разработка ПС состоит из цепочки этапов. На каждом этапе создаются документы, используемые на последующем этапе. В исходном документе фиксируются требования к ПС. В конце этой цепочки создаются программы, включаемые в ПС.
2. *Исследовательское программирование*. Этот подход предполагает быструю (насколько это возможно) реализацию рабочих версий программ ПС, выполняющих лишь в первом приближении требуемые функции. После экспериментального применения реализованных программ производится их модификация с целью сделать их более полезными для пользователей. Этот процесс повторяется до тех пор, пока ПС не будет достаточно приемлемо для пользователей. Такой подход применялся на ранних этапах развития программирования, когда технологии программирования не придавали

большого значения (использовалась интуитивная технология). В настоящее время этот подход применяется для разработки таких ПС, для которых пользователи не могут точно сформулировать требования (например, для разработки систем искусственного интеллекта).

3. *Прототипирование*. Этот подход моделирует начальную фазу исследовательского программирования вплоть до создания рабочих версий программ, предназначенных для проведения экспериментов с целью установить требования к ПС. В дальнейшем должна последовать разработка ПС по установленным требованиям в рамках какого-либо другого подхода (например, водопадного).
4. *Формальные преобразования*. Этот подход включает разработку формальных спецификаций ПС и превращение их в программы путем корректных преобразований.

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

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

Тема лабораторной работы № 14. «Настройка работы системы контроля версий (типов импортируемых файлов, путей, фильтров и др. параметров импорта в репозиторий)», объем часов 1

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: Изучить основы настройки работы системы контроля версий

Задание(я):

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

Методические указания по ходу выполнения работы

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

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

Распространённые системы управления версиями

- Subversion
- Darcs
- Microsoft Visual SourceSafe
- Bazaar
- Rational ClearCase
- Perforce
- BitKeeper
- Mercurial
- Git
- GNU Arch
- CVS — устаревшая. Потомок: Subversion
- RCS — устаревшая. Потомок: CVS

Основные понятия

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

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

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

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

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

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

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

Borland StarTeam

Borland StarTeam - очень мощный и функциональный кросс-платформенный продукт, разрабатываемый в прошлом фирмой StarBase, которую Borland приобрела в конце 2002 г. Заметное преимущество данного решения состоит в том, что версия 2005 выступает центральным элементом стратегии управления жизненным циклом приложений (Application Lifecycle

Management, ALM) компании Borland и обладает расширенными возможностями интеграции со всеми ее ключевыми пакетами, используемыми при разработке программного обеспечения.

MS SourceSafe

Microsoft Visual SourceSafe (Visual SourceSafe, VSS) — программный продукт компании Майкрософт, файл-серверная система управления версиями, предназначенная для небольших команд разработчиков. VSS позволяет хранить в общем хранилище файлы, разделяемые несколькими пользователями, для каждого файла хранится история версий. VSS входит в состав пакета Microsoft Visual Studio и интегрирован с продуктами этого пакета. Доступен только для платформы Windows. Версию для Unix поддерживает компания MainSoft. В ноябре 2005 года вышла обновлённая версия продукта — Visual SourceSafe 2005, обещающая повышенную стабильность и производительность, улучшенный механизм слияния для XML-файлов и файлов в Юникоде, а также работу через HTTP. Visual SourceSafe нацелен на индивидуальных разработчиков либо небольшие команды разработчиков. Там где VSS недостаточно, ему на замену предлагается новый продукт Майкрософт — Team Foundation Server, входящий в состав Visual Studio Team System.

Rational Clear Case

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

- Общий контроль версий не только файлов, но и директорий/поддиректорий;
- Бесконечное число ответвлений от определенной версии;
- Автоматическая компрессия файлов и их кеширование (CC позволяет хранить большое количество данных, при всем при этом база данных остается компактной и быстрой);
- Позволяет легко конвертировать базы данных других средств контроля, например: PVCS, SourceSafe, RCS, CVS и SCCS;
- Поддерживает параллельную разработку и мультикомандные подразделения, расположенные в географически удаленных друг от друга местах;
- Мультиплатформенность (способен объединить единой средой участников, работающих на разных операционных системах);
- Имеет интеграцию со средствами разработки;
- Имеет Web-интерфейс для удаленного контроля.

CVS

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

Subversion

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

Тема лабораторной работы № 15. «Разработка и интеграция модулей проекта (командная работа)», объем часов 1

У1 использовать выбранную систему контроля версий;

У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: Освоить процесс проектирования модулей программного обеспечения..

Задание(я):

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

Отчет по практической работе должен включать:

1. Алгоритм решения задачи.
2. Набор тестов для отладки программы.

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

Варианты индивидуальных заданий.

1. Даны два двумерных массива вещественных элементов. Размер исходных массивов не превосходит 10х10 элементов. Для каждого из массивов указать номера столбцов, содержащих только положительные элементы. Если таковых столбцов в массиве нет, то вывести соответствующее сообщение. Проверку столбца на положительность элементов оформить в виде процедуры с передачей в нее всех элементов текущего столбца.
2. Даны два двумерных массива натуральных элементов. Размер исходных массивов не превосходит 10х10 элементов. Для каждого из массивов указать номера столбцов, содержащих только кратные 5 или 7 элементы. Если таких столбцов в массиве нет, то вывести соответствующее сообщение. Проверку столбца на наличие указанных элементов оформить в виде процедуры с передачей в нее всех элементов текущего столбца.
3. Даны пять одномерных массива вещественных элементов. Размер каждого массива не превосходит 100 элементов. Для каждого из массивов определить, составляют ли его элементы знакочередующуюся последовательность. Если да, то указать порядковый номер такого массива, в противном случае вывести отрицательный ответ. Проверку массива на выполнение условия оформить в виде процедуры с передачей в нее всех элементов рассматриваемого массива.
4. Даны два двумерных массива символьных (буквы русского алфавита) элементов. Размер исходных массивов не превосходит 10х10 элементов. Для каждого из массивов указать номера строк, содержащих элементы только строчных букв, если таких строк нет ни для какого массива, то вывести соответствующее сообщение. Проверку строки на наличие указанных элементов оформить в виде процедуры с передачей в нее всех элементов текущей строки.
5. Даны два двумерных массива вещественных элементов. Размер исходных массивов не превосходит 10х10 элементов. Для каждого из массивов указать количество столбцов, содержащих только не положительные элементы. Если таких столбцов нет ни для одного из массивов, то вывести соответствующее сообщение. Проверку столбца на наличие указанных элементов оформить в виде процедуры с передачей в нее всех элементов текущего столбца.
6. Даны пять одномерных массива вещественных элементов. Размер каждого массива не превосходит 100 элементов. Для каждого из массивов определить, составляют ли его элементы одного знака. Если да, то указать порядковый номер такого массива, в

противном случае вывести отрицательный ответ. Проверку массива на выполнение условия оформить в виде процедуры с передачей в нее всех элементов рассматриваемого массива.

7. Даны два двумерных массива целочисленных элементов. Размер исходных массивов не превосходит 10x10 элементов. Для каждого из массивов указать количество строк, содержащих элементы, четность которых чередуется, а вторым в четных строках является нечетный элемент. Если таких строк нет ни для одного из массивов, то вывести соответствующее сообщение. Проверку строки на наличие указанных элементов оформить в виде процедуры с передачей в нее всех элементов текущего столбца.
8. Даны пять одномерных массива символьных (только латинские буквы) элементов. Размер каждого массива не превосходит 100 элементов. Для каждого из массивов определить, чередуются ли в нем буквы строчные и прописные. Если да, то указать порядковый номер такого массива, в противном случае вывести отрицательный ответ. Проверку массива на выполнение условия оформить в виде процедуры с передачей в нее всех элементов рассматриваемого массива.
9. Даны два двумерных массива целочисленных элементов. Размер исходных массивов не превосходит 10x10 элементов. Для каждого из массивов указать количество строк, для которых сумма элементов, стоящих на нечетных местах в строке, является положительным числом. Если таких строк нет ни для одного из массивов, то вывести соответствующее сообщение. Проверку строки на выполнение условия и расчет оформить в виде процедуры с передачей в нее всех элементов текущей строки.
10. Даны два двумерных массива вещественных элементов. Размер исходных массивов не превосходит 10x10 элементов. Для каждого из массивов указать номера столбцов, произведение отрицательных элементов которых является положительным числом. Если таких столбцов нет ни для одного из массивов, то вывести соответствующее сообщение. Проверку столбца на выполнение условия и расчет оформить в виде процедуры с передачей в нее всех элементов текущего столбца.
11. Даны пять одномерных массива символьных (только латинские буквы) элементов. Размер каждого массива не превосходит 100 элементов. Для каждого из массивов определить, расположены ли в нем строчные буквы в алфавитном порядке. Если да, то указать порядковый номер такого массива, в противном случае вывести отрицательный ответ. Проверку массива на выполнение условия оформить в виде процедуры с передачей в нее всех элементов рассматриваемого массива.
12. Даны два двумерных массива целочисленных элементов. Размер исходных массивов не превосходит 10x10 элементов. Для каждого из массивов проверить выполнение условия: все четные строки массива таковы, что суммы их элементов образуют возрастающую последовательность. Вывести соответствующее сообщение. Вычисление суммы элементов массива и проверку последовательности чисел на выполнение условия оформить в виде процедуры с передачей в нее всех необходимых элементов.
13. Даны два двумерных массива вещественных элементов. Размер исходных массивов не превосходит 10x10 элементов. Преобразовать все нечетные строки каждого массива так, чтобы элементы составляли возрастающую по абсолютной величине последовательность. Вывести преобразованные массивы. Упорядочивание элементов оформить в виде процедуры с передачей в нее всех необходимых элементов.
14. Даны два двумерных массива целочисленных элементов. Размер исходных массивов не превосходит 10x10 элементов. Для каждого столбца массивов вычислить суммы и количества элементов, значения которых находятся в заданном диапазоне. Если чисел, удовлетворяющих этому условию нет, то вывести соответствующее сообщение. Вычисление для элементов столбца массива оформить в виде процедуры с передачей в нее всех необходимых элементов.
15. Даны пять одномерных массива символьных (только латинские буквы) элементов. Размер каждого массива не превосходит 100 элементов. Преобразовать все массивы так, чтобы все строчные буквы были расположены по алфавиту. При этом переставлять только строчные буквы, оставив прописные буквы на своих местах. Преобразование каждого массива оформить в виде процедуры с передачей в нее всех необходимых элементов. Если перестановка элементов не потребовалась, то есть исходные массивы

удовлетворяют требуемому условию, то вывести соответствующее сообщение.

Методические указания по ходу выполнения работы

Чётко следовать инструкциям.

Тема лабораторной работы № 16. «Отладка отдельных модулей программного проекта», объем часов 1

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: Изучить основные подходы к проектированию тестов.

Задание(я):

1. Оформить внешнюю спецификацию.
2. Составить в виде блок-схемы алгоритм решения задачи.
3. Создать программу решения задачи на любом алгоритмическом языке
4. программирования.
5. Составить набор тестов и провести тестирование созданной программы с помощью методов «белого ящика» (покрытия операторов, покрытия решений, покрытия условий, комбинаторного покрытия условий).
6. Оформить отчет по лабораторной работе.

Задача. «Нахождение характерных точек функции». Составить алгоритм и написать программу последовательного вычисления значений заданной функции Y(X) до тех пор, пока не будет пройдена некоторая характерная точка графика функции. Значения аргумента X составляют возрастающую последовательность с шагом h. Начальное значение X_0 и шаг изменения аргумента h задаются пользователем.

Методические указания по ходу выполнения работы

Рассмотрим два основных подхода к проектированию тестов.

Первый подход ориентируется только на стратегию тестирования, называемую стратегией "черного ящика", тестированием с управлением по данным или тестированием с управлением по входу-выходу. При использовании этой стратегии программа рассматривается как черный ящик. Тестовые данные используются только в соответствии со спецификацией программы (т. е. без учета знаний о ее внутренней структуре). Недостижимый идеал сторонника первого подхода — проверить все возможные комбинации и значения на входе. Обычно их слишком много даже для простейших алгоритмов. Так, для программы расчета среднего арифметического четырех чисел надо готовить 10^7 тестовых данных.

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

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

Второй подход использует стратегию "белого ящика", или стратегию тестирования, управляемую логикой программы, которая позволяет исследовать внутреннюю структуру программы. В этом случае тестировщик получает тестовые данные путем анализа только логики программы; стремится, чтобы каждая команда была выполнена хотя бы один раз. При достаточной квалификации добивается, чтобы каждая команда условного перехода выполнялась бы в каждом направлении хотя бы один раз. Цикл должен выполняться один раз, ни разу, максимальное число раз. Цель тестирования всех путей извне также недостижима. В программе из двух последовательных циклов внутри каждого из них включено ветвление на десять путей, имеется 10^{18} путей расчета. Причем выполнение всех путей расчета не гарантирует выполнения всех спецификаций.

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

Последнее утверждение имеет два слабых пункта: во-первых, число не повторяющих друг друга маршрутов — астрономическое; во-вторых, даже если каждый маршрут может быть проверен, сама программа может содержать ошибки (например, некоторые маршруты пропущены).

Свойство пути выполняться правильно для одних данных и неправильно для других — называемое чувствительностью к данным, наиболее часто проявляется за счет численных погрешностей и погрешностей усечения методов. Тестирование каждого из всех маршрутов одним тестом не гарантирует выявление чувствительности к данным.

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

Рассмотрим пример тестирования оператора:

if A and B then..

при использовании разных критериев полноты тестирования.

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

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

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

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

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

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

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

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

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

Тема лабораторной работы № 17. «Организация обработки исключений», объем часов 1

У1 использовать выбранную систему контроля версий;

У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: Изучить основы верстки. Научиться управлять интерфейсом мобильного устройства при разработке программного приложения.

Задание(я):

1. Оформить внешнюю спецификацию.
2. Составить в виде блок-схемы алгоритм решения задачи.
3. Создать программу решения задачи на любом алгоритмическом языке программирования.
4. Составить набор тестов и провести тестирование созданной программы с помощью методов «белого ящика» (покрытия операторов, покрытия решений, покрытия условий, комбинаторного покрытия условий).

Методические указания по ходу выполнения работы

Рассмотрим два основных подхода к проектированию тестов.

Первый подход ориентируется только на стратегию тестирования, называемую стратегией "черного ящика", тестированием с управлением по данным или тестированием с управлением по входу-выходу. При использовании этой стратегии программа рассматривается как черный ящик. Тестовые данные используются только в соответствии со спецификацией программы (т. е. без учета знаний о ее внутренней структуре). Недостижимый идеал сторонника первого подхода — проверить все возможные комбинации и значения на входе. Обычно их слишком много даже для простейших алгоритмов. Так, для программы расчета среднего арифметического четырех чисел надо готовить 10^7 тестовых данных.

При первом подходе обнаружение всех ошибок в программе является критерием исчерпывающего входного тестирования. Последнее может быть достигнуто, если в качестве тестовых наборов использовать все возможные наборы входных данных. Следовательно, приходим к выводу, что для исчерпывающего тестирования программы требуется бесконечное число тестов, а значит, построение исчерпывающего входного теста невозможно. Это подтверждается двумя аргументами: во-первых, нельзя создать тест, гарантирующий отсутствие ошибок; во-вторых, разработка таких тестов противоречит экономическим требованиям. Поскольку исчерпывающее тестирование исключается, нашей целью должна стать максимизация результативности капиталовложений в тестирование (максимизация числа ошибок, обнаруживаемых одним тестом). Для этого необходимо рассматривать внутреннюю структуру программы и делать некоторые разумные, но, конечно, не обладающие полной гарантией достоверности предположения.

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

10^{18} путей расчета. Причем выполнение всех путей расчета не гарантирует выполнения всех спецификаций.

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

Последнее утверждение имеет два слабых пункта: во-первых, число не повторяющих друг друга маршрутов — астрономическое; во-вторых, даже если каждый маршрут может быть проверен, сама программа может содержать ошибки (например, некоторые маршруты пропущены).

Свойство пути выполняться правильно для одних данных и неправильно для других — называемое чувствительностью к данным, наиболее часто проявляется за счет численных погрешностей и погрешностей усечения методов. Тестирование каждого из всех маршрутов одним тестом не гарантирует выявление чувствительности к данным.

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

Рассмотрим пример тестирования оператора:

if A and B then..

при использовании разных критериев полноты тестирования.

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

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

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

Критерием, который решает эти и некоторые другие проблемы, является комбинаторное покрытие условий. Он требует создания такого числа тестов, чтобы все

возможные комбинации результатов условия в каждом решении и все точки входа выполнялись, по крайней мере, один раз.

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

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

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

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

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

Тема лабораторной работы № 18. «Применение отладочных классов в проекте», объем часов 2

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: Получение практических навыков тестирования и отладки программы.

Задание(я):

1. Составить в виде блок-схемы алгоритм решения задачи.
2. Создать программу решения задачи на любом алгоритмическом языке программирования.
3. Составить отчет по лабораторной работе.

Отчет по лабораторной работе должен включать:

1. Алгоритм решения задачи.
2. Текст программы на языке программирования.

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

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

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

Методические указания по ходу выполнения работы

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

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

- требует от программиста глубоких знаний специфики управления используемыми техническими средствами, операционной системы, среды и языка программирования, реализуемых процессов, природы и специфики различных ошибок, методик отладки и соответствующих программных средств;
- психологически дискомфортна, так как необходимо искать собственные ошибки и, как правило, в условиях ограниченного времени;
- возможно взаимовлияние ошибок в разных частях программы, например, за счет затирания области памяти одного модуля другим из-за ошибок адресации;
- отсутствуют четко сформулированные методики отладки.

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

- ручного тестирования;
- индукции;
- дедукции;
- обратного прослеживания.

Метод ручного тестирования

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

Метод индукции

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

Самый ответственный этап - выявление симптомов ошибки. Организуя данные об ошибке, целесообразно записать все, что известно о её проявлениях, причем фиксируют, как ситуации, в которых фрагмент с ошибкой выполняется нормально, так и ситуации, в которых ошибка проявляется. Если в результате изучения данных никаких гипотез не появляется, то необходима дополнительная информация об ошибке. Дополнительную информацию можно получить,

например, в результате выполнения схожих тестов. В процессе доказательства пытаются выяснить, все ли проявления ошибки объясняет данная гипотеза, если не все, то либо гипотеза не верна, либо ошибок несколько.

Метод дедукции

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

Метод обратного прослеживания

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

Тема лабораторной работы № 19. «Отладка проекта», объем часов 2

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: Получение практических навыков тестирования и отладки программы.

Задание(я):
Отладить программу, созданную в лабораторной работе №18.

Методические указания по ходу выполнения работы

Выполнить отладку в IDE.

Отчет по лабораторной работе должен включать:

1. Алгоритм решения задачи.
2. Текст программы на языке программирования.
3. Набор тестов для отладки программы.

Тема лабораторной работы № 20. «Инспекция кода модулей проекта», объем часов 2

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

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

Задание(я):

1. Оформить внешнюю спецификацию.
2. Составить в виде блок-схемы алгоритм решения задачи.
3. Спроектировать и разработать модули программы для решения задачи на любом алгоритмическом языке программирования.

1. Выполнить отладку и тестирование модулей программы.
2. Выполнить инкрементную интеграцию модулей с использованием одного из подходов.

3. Выполнить системное тестирование программы.
4. Оформить отчет по лабораторной работе.

Отчет по лабораторной работе должен включать:

1. Внешнюю спецификацию.
2. Алгоритм решения задачи.
3. Текст программы на языке программирования.
4. Набор тестов для отладки модулей программы.
5. Описание процесса интеграции модулей.

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

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

Методические указания по ходу выполнения работы

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

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

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

Аккуратная интеграция обеспечивает:

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

- лучшее качество кода;
- меньшее количество документации.

Интеграция программ выполняется посредством ***поэтапного*** или ***инкрементного*** подхода.

Поэтапная интеграция состоит из этапов, перечисленных ниже:

1. «Модульная разработка»: проектирование, кодирование, тестирование и отладка каждого класса.

2. «Системная интеграция»: объединение классов в одну огромную систему.

3. «Системная дезинтеграция»: тестирование и отладка всей системы.

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

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

Поэтому поэтапную интеграцию называют еще «интеграцией большого взрыва»

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Нисходящая интеграция

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

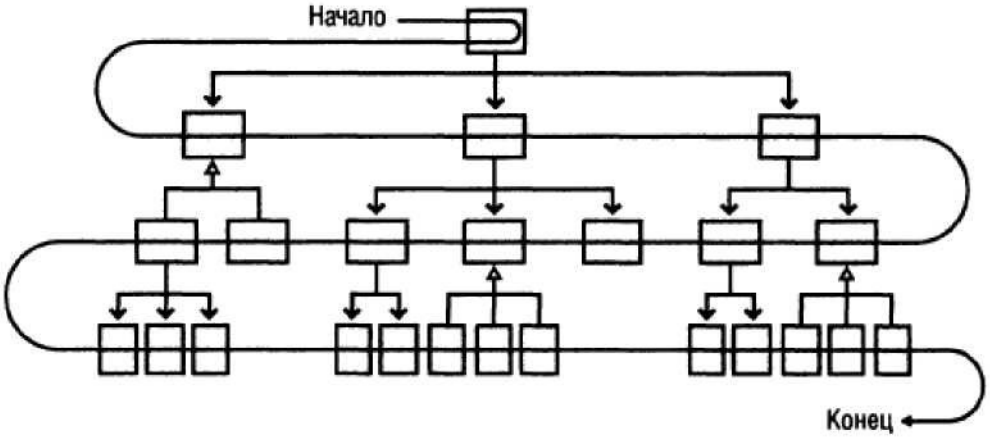
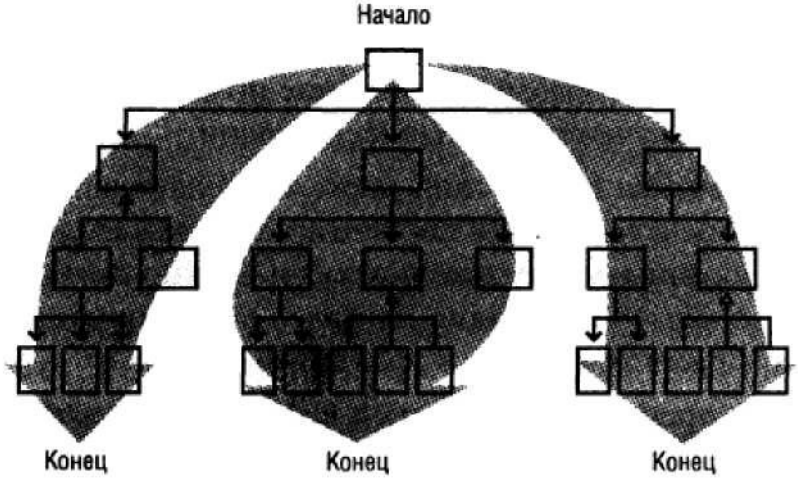


Рис.3 Нисходящая интеграция

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

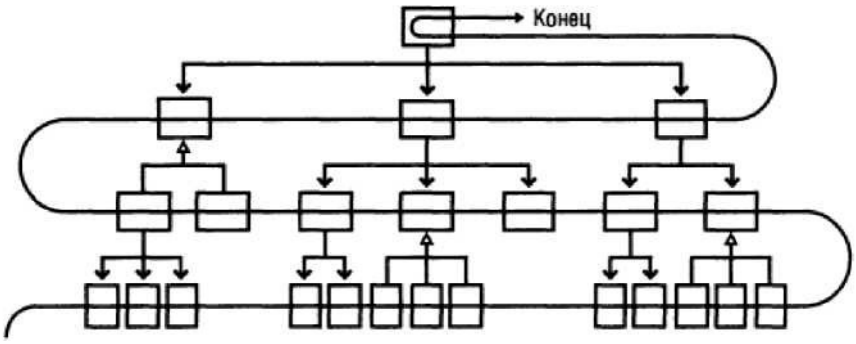
Хорошей альтернативой нисходящей интеграции в чистом виде может стать подход с вертикальным секционированием.

Рис. 4 Вертикальное секционирование



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

Восходящая интеграция



Начало

Рис.5 Восходящая интеграция

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

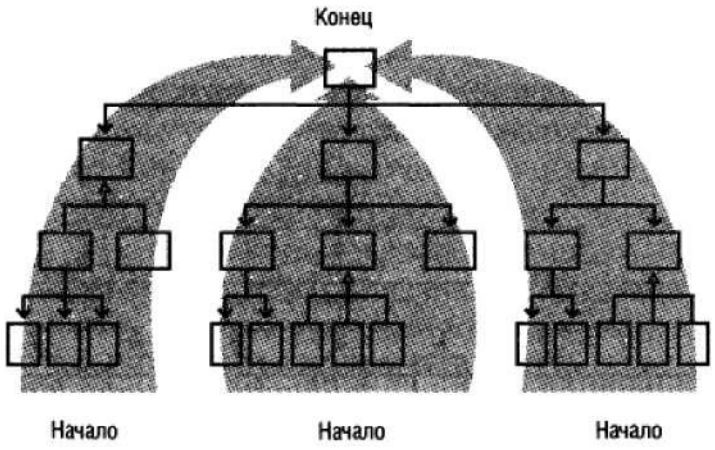


Рис. 6 Гибридный подход при восходящей интеграции

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

Сэндвич-интеграция

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

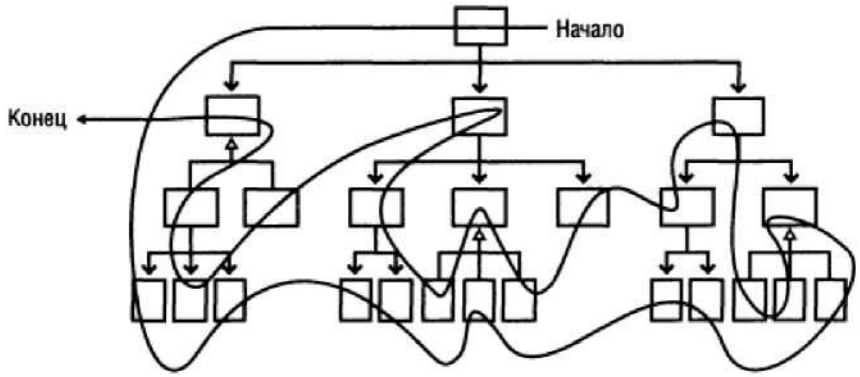


Рис. 7 Сэндвич-интеграция

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

Напоследок вы оставляете классы среднего уровня.

Риск-ориентированная интеграция

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

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

Функционально-ориентированная интеграция

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

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

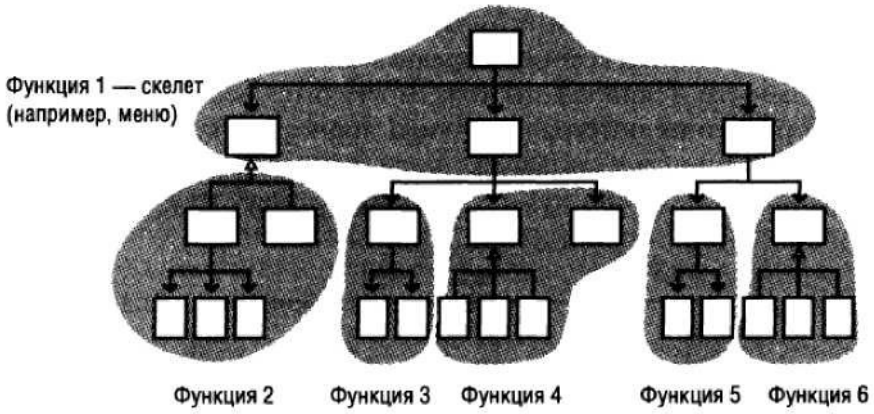


Рис. 8 Функционально-ориентированная интеграция

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

Т-образная интеграция

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

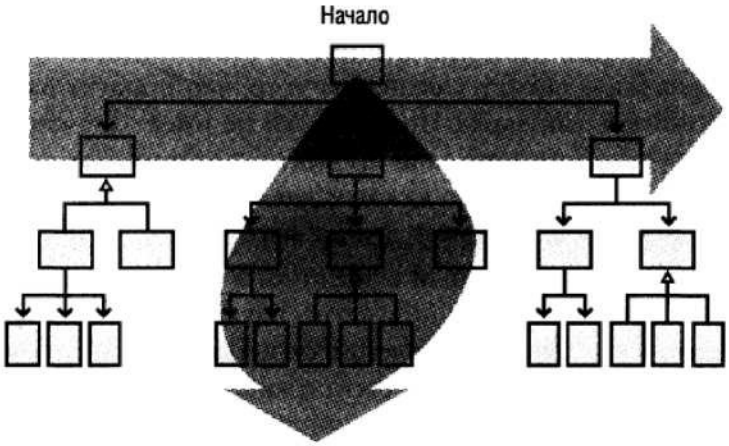


Рис. 9 Т-образная интеграция

Тема лабораторной работы № 21. «Тестирование интерфейса пользователя средствами инструментальной среды разработки»», объем часов 2

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

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

Задание(я):

1. Сформировать диаграмму вариантов использования для задачи лабораторной работы № 1.
2. Сгенерировать набор тестов.
3. Составить отчет по лабораторной работе.

Методические указания по ходу выполнения работы

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

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

Для решения задачи тестирования программ с программным интерфейсом (API - Application Program Interface: вызовы методов или процедур, пересылки сообщений) известны подходы - методы и инструменты - хорошо зарекомендовавшие себя в индустрии создания программного обеспечения. Основа этих подходов следующая: создается формальная спецификация программы, и по этой спецификации генерируются как сами тесты, так и тестовые оракулы - программы, проверяющие правильность поведения тестируемой программы. Спецификации, как набор требований к создаваемой программе, существовали всегда, Ключевым

словом здесь является формальная спецификация. Формальная спецификация - это спецификация в форме, допускающей её формальные же преобразования и обработку компьютером. Это позволяет анализировать набор требований с точки зрения их полноты, непротиворечивости и т.п. Для задачи автоматизации тестирования эта формальная запись должна также обеспечивать возможность описания формальной связи между понятиями, используемыми в спецификации, и сущностями языка реализации программы.

Правильность функционирования системы определяется соответствием реального поведения системы эталонному поведению. Для того чтобы качественно определять это соответствие, нужно уметь формализовать эталонное поведение системы. Распространённым способом описания поведения системы является описание с помощью диаграмм UML (Unified Modeling Language). Стандарт UML предлагает использование трех видов диаграмм для описания графического интерфейса системы:

S Диаграммы сценариев использования (Use Case).

S Диаграммы конечных автоматов (State Chart).

S Диаграммы действий (Activity).

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

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

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

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

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

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

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

- Наличие и состояние окон (заголовок, активность, доступность, статус).
- Наличие и состояние таких объектов, как PushButton, CheckBox, RadioButton, List, Tree и др. (текст, доступность, размер).
- Значение буфера обмена.

- Наличие в оперативной памяти запущенных процессов.
- Существование файлов.

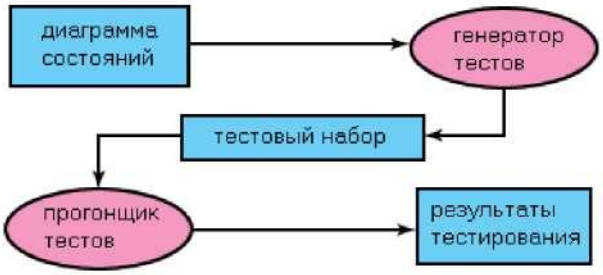


Рис. 2 Общая схема генерации и прогона тестов

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

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

Генератор тестов есть программа (script), написанная на языке ‘Rational Rose Scripting language’ (расширение к языку Summit BasicScriptLanguage). В Rational Rose есть встроенный интерпретатор инструкций, написанных на этом языке, посредством которого можно обращаться ко всем объектам модели (диаграммы состояний).

- Спецификация автоматически интерпретируется (тем самым она проверяется и компилируется в набор тестов).
- Если какая-то функциональность системы изменилась, то диаграмму состояний достаточно изменить в соответствующем месте, и затем сгенерировать новый тестовый набор. Фактически, это снимает большую часть проблем, возникающих при организации регрессионного тестирования.
- Гарантия тестового покрытия. Эта гарантия даётся соответствующим алгоритмом обхода графа состояний.

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

Тема лабораторной работы № 22. «Разработка тестовых модулей проекта для тестирования отдельных модулей», объем часов 2

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: Получение практических навыков использования средств автоматизации тестирования.

Задание(я):

1. Выполнить тестовый набор лабораторной работы № 2.
2. Проанализировать отчёт о прохождении тестов.
3. Составить отчет по лабораторной работе.

Методические указания по ходу выполнения работы

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

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

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

Тема лабораторной работы № 23. «Выполнение функционального тестирования», объем часов 2

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

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

Задание(я):

1. На основе внешней спецификации задачи Практического занятия №5 составить набор тестов на основе подхода «чёрного ящика».
2. Провести тестирование программы.
3. Оформить отчет по лабораторной работе.

Методические указания по ходу выполнения работы

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

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

Эквивалентное разбиение.

Область всех возможных наборов входных данных программы по каждому параметру разбивают на конечное число групп - ***классов эквивалентности***. Наборы данных такого класса объединяют по принципу обнаружения одних и тех же ошибок. Для составления классов эквивалентности нужно перебрать ограничения, установленные для каждого входного значения в техническом задании или при уточнении спецификации. Каждое ограничение разбивают на две или более групп.

Граничные значения.

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

Анализ причинно-следственных связей.

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

Предположение об ошибке.

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

Тема лабораторной работы № 24. «Тестирование интеграции», объем часов 2

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

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

Задание(я):

1. Составить в виде блок-схемы алгоритм решения задачи.
2. . Создать программу решения задачи на любом алгоритмическом языке программирования.
3. Отладить программу с использованием инструментальных средств.
4. Составить отчет по лабораторной работе.

Методические указания по ходу выполнения работы

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

Для сокращения времени отладки необходимо пользоваться научным подходом.

Классический научный подход включает следующие этапы:

1. Сбор данных при помощи повторяющихся экспериментов.
2. Формулирование гипотезы, объясняющей релевантные данные.
3. Разработка эксперимента, призванного подтвердить или опровергнуть гипотезу.
4. Подтверждение или опровержение гипотезы.
5. Повторение процесса в случае надобности.

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

1. Стабилизация ошибки.

2. Определение источника ошибки.
a. Сбор данных, приводящих к дефекту.
b. Анализ собранных данных и формулирование гипотезы, объясняющей дефект.
c. Определение способа подтверждения или опровержения гипотезы, основанного или на тестировании программы, или на изучении кода.

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

3. Исправление дефекта.
4. Тестирование исправления.
5. Поиск похожих ошибок.

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

1. сокращение подозрительной области кода;
2. проверка классов и методов, в которых дефекты обнаруживались ранее;
3. проверка кода, который изменялся недавно.

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

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

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

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

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

- выполнять программу по шагам, причем как с заходом в подпрограммы, так и выполняя их целиком;
- предусматривать точки останова;
- выполнять программу до оператора, указанного курсором;
- отображать содержимое любых переменных при пошаговом выполнении;
- отслеживать поток сообщений и т.п.

Тема лабораторной работы № 25. «Документирование результатов тестирования», объем часов 2

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

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

Задание(я):

1. Выполнить тестирование программы, разработанной в лабораторной работе № 4.
2. Оформить протоколы тестирования.
3. Оформить отчет по лабораторной работе.

Методические указания по ходу выполнения работы

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

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

- административное описание дефекта (дата обнаружения, сотрудник, сообщивший о дефекте, номер сборки программы, дата исправления);
- полное описание проблемы;
- действия, предпринятые для воспроизведения проблемы;
- предложенные способы решения проблемы;
- родственные дефекты;
- тяжесть проблемы (например, критическая проблема, «неприятная» или косметическая);
- источник дефекта: выработка требований, проектирование, кодирование или тестирование;
- вид дефекта кодирования: ошибка занижения или завышения на 1, ошибка присваивания, недопустимый индекс массива, неправильный вызов метода и т. д.;
- классы и методы, измененные при исправлении дефекта;
- число строк, затронутых дефектом;
- время, ушедшее на нахождение дефекта;
- время, ушедшее на исправление дефекта.

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

- число дефектов в каждом классе; все числа целесообразно отсортировать в порядке от худшего класса к лучшему и, возможно, нормализовать по размеру класса;
- число дефектов в каждом методе, все числа целесообразно отсортировать в порядке от худшего метода к лучшему и, возможно, нормализовать по размеру метода;
- среднее время тестирования в расчете на один обнаруженный дефект;
- среднее число обнаруженных дефектов в расчете на один тест;
- среднее время программирования в расчете на один исправленный дефект;
- процент кода, покрытого тестами;
- число дефектов, относящихся к каждой категории тяжести.

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

Тема лабораторной работы № 26. «Построение простейших математических моделей. Построение простейших статистических моделей», объем часов 1

У1 использовать выбранную систему контроля версий;

У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: закрепить практические навыки по построению простейших математических и простейших статистических моделей.

Задание(я):

Задание 1. Составить математическую модель следующей задачи. На складе имеется 300 кг сырья. Надо изготовить два вида продукции. На изготовление первого изделия требуется 2 кг сырья, а на изготовление второго изделия — 5 кг. Определить план выпуска двух изделий.

Решение.

Обозначим, x1 - единица первого изделия, x2 - единица второго изделия. Тогда составим математическая модель: 2x1+5x2=300.

Задание 2. Составить математическую модель следующей задачи. Предположим, что для производства продукции вида А и В можно использовать материал 3-х сортов. При этом на изготовление единицы изделия вида А расходуется 14 кг первого сорта, 12 кг второго сорта и 8 кг третьего сорта. На изготовление продукции вида В расходуется 8 кг первого сорта, 4 кг второго сорта, 2 кг третьего сорта. На складе фабрики имеется всего материала первого сорта 624 кг, второго сорта 541 кг, третьего сорта 376 кг. От реализации единицы готовой продукции вида А фабрика имеет прибыль вида 7 руб., а от реализации единицы готовой продукции вида В фабрика имеет прибыль вида 3 руб. Определить максимальную прибыль от реализации всей продукции видов А и В.

Решение.

Составим математическую модель задачи:

Пусть x_i - единица готовой продукции вида А,

x_2 - единица готовой продукции вида В,

Цель фабрики получить максимальную прибыль от реализации всей продукции видов

А и В, тогда:

$F = 7 \bullet Xy + 3 \bullet x_2 \wedge \max$

Система ограничений:

$14x + 8X_2 < 624$

$< 12Xy + 4X_2 < 541$

$\_ \ 8Xy + 2X_2 < 376$

$Xy > 0, \quad X_2 > 0$ *условие неотрицательности*

Задание 3. Составить математическую модель следующей задачи. Имеются три пункта поставки однородного груза А1, А2, А3 и пять пунктов В1, В2, В3, В4, В5 потребления этого груза. На пунктах А1, А2 и А3 находится груз соответственно в количестве 200, 450, 250 тонн. В пункты В1, В2, В3, В4, В5 требуется доставить соответственно 100, 125, 325, 250, 100 тонн груза. Расстояние между пунктами поставки и пунктами потребления приведено в таблице:

Пункты поставки	Пункты потребления				
	B1	**B2**	**B3**	**B4**	**B5**
A1	5	8	7	10	3
A2	4	2	2	5	6
A3	7	3	5	9	2

Решение:

1. Проверка сбалансированности модели задачи. Модель является сбалансированной, т.к. суммарный объем запасов сырья равен суммарному объему потребности в ней:
   200+450+250=100+125+325+250+100.

2. Построение математической модели - неизвестными в этой задачи является объем перевозок. Пусть x_{ij} - объем перевозок с i -го предприятия в j-го пункт потребления. Суммарные транспортные расходы - это функционал качества (критерий цели):

$$F=\sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij}$$

Где c_{ij} - стоимость перевозки единицы продукции с i -го предприятия в j-й пунктах потребления.

Неизвестные в этой задачи должны удовлетворять следующим ограничениям:

- Объем перевозок не могут быть отрицательными;
- Поскольку модель сбалансирована, то вся продукция должна быть вывезена с предприятия, а потребность всех пунктов потребления должна быть полностью удовлетворены.

Итак, имеем следующую задачу:

$$F=\sum\sum c_{ij}x_{ij} \to \min,$$

- Найти минимум функционала:

$$E \; *_{*} = {}^{12}5, \qquad (5$$

$$E_{x_{ч}} = 2^{00,} \mathrm{j}_{j\,5-}$$

1

]

При ограничениях: $E\ X_{ii} = 100.$ $X_j > 0,\ i \in [1,3],\ j \in [1,5].$

Задания для самостоятельной работы

1 вариант.

Задача 1. Составить математическую модель следующей задачи. Предположим, что для производства продукции вида А и В можно использовать материал трех сортов. При этом на изготовление единицы изделия вида А расходуется ***a1*** кг первого сорта, ***a2*** кг второго сорта и ***a3*** кг третьего сорта. На изготовление продукции вида В расходуется ***b1*** кг первого сорта, ***b2*** кг второго сорта, ***b3*** кг третьего сорта. На складе фабрики имеется всего материала первого сорта ***c1*** кг, второго сорта ***c2*** кг, третьего сорта ***c3*** кг. От реализации единицы готовой продукции вида А фабрика имеет прибыль вида ***a*** руб., а от реализации единицы готовой продукции вида В фабрика имеет прибыль вида ***в*** руб. Определить максимальную прибыль от реализации всей продукции

a1= 19, a2= 16, a3= 19, b1= 26, b2= 17, b3= 8, c1= 868, c2= 638, c3= 853,

a=5, в=4.

Задача 2. Имеются три пункта поставки однородного груза **A1, A2, A3** и пять пунктов **B1, B2, B3, B4, B5** потребления этого груза. На пунктах **A1, A2 и A3** находится груз соответственно в количестве ***a1, a2 и a3*** тонн. В пункты **B1, B2, B3, B4, B5** требуется доставить соответственно ***b1, b2, b3, b4, b5*** тонн груза. Расстояние между пунктами поставки и пунктами потребления приведено в таблице:

Пункты поставки	**Пункты потребления**				
	B1	**B2**	**B3**	**B4**	**B5**
A1	D11	D12	D13	D14	D15
A2	D21	D22	D23	D24	D25
A3	D31	D32	D33	D34	D35

Найти такой план закрепления потребителей за поставщиками однородного груза, чтобы общие затраты по перевозкам были минимальными.

a1=300, a2=250, a3=200,

b1=210, b2=150, b3=120, b4=135, b5=135.

$$D=\begin{vmatrix} 4 & 8 & 13 & 2 & 7 \\ 9 & 4 & 11 & 9 & 17 \\ 3 & 16 & 10 & 1 & 4 \end{vmatrix},$$

Задача 2. Имеются три пункта поставки однородного груза **A1, A2, A3** и пять пунктов **B1, B2, B3, B4, B5** потребления этого груза. На пунктах **A1, A2 и A3** находится груз соответственно в количестве ***a1, a2 и a3*** тонн. В пункты **B1, B2, B3, B4, B5** требуется доставить соответственно ***b1, b2, b3, b4, b5*** тонн груза. Расстояние между пунктами поставки и пунктами потребления приведено в таблице:

Пункты поставки	**Пункты потребления**				
	B1	**B2**	**B3**	**B4**	**B5**
A1	D11	D12	D13	D14	D15
A2	D21	D22	D23	D24	D25
A3	D31	D32	D33	D34	D35

† **вариант.**

Задача 1. Составить математическую модель следующей задачи. Предположим, что для производства продукции вида А и В можно использовать материал трех сортов. При этом на изготовление единицы изделия вида А расходуется ***a1*** кг первого сорта, ***a2*** кг второго сорта и ***a3*** кг третьего сорта. На изготовление продукции вида В расходуется ***b1*** кг первого сорта, ***b2*** кг второго сорта, ***b3*** кг третьего сорта. На складе фабрики имеется всего материала первого сорта ***c1*** кг, второго сорта ***c2*** кг, третьего сорта ***c3*** кг. От реализации единицы готовой продукции вида А фабрика имеет прибыль вида ***a*** руб., а от реализации единицы готовой продукции вида В фабрика имеет прибыль вида ***в*** руб. Определить максимальную прибыль от реализации всей продукции видов А и В.

a1= 14, a2= 15, a3= 20, b1= 40, b2= 27, b3= 4, c1= 1200, c2= 993, c3= 1097,

‡***a=5, в=13.***

го груза, чтобы

Найти такой план закрепления потребителей за поставщиками однородно общие затраты по перевозкам были минимальными.

a1=350, a2=200, a3=300,

(22 14 16 28 30>

^{b}***1=***^{170,} ^{b}***2=***^{140,} ^{b}***3=***^{200,} ^{b}***4=***^{195,} ^{b5}***=***^{145}**-**

$D =$ 19 17 26 36 36
^ 37 30 31 39 41

3 вариант.

Задача 1. Составить математическую модель следующей задачи. Предположим, что для производства продукции вида А и В можно использовать материал трех сортов. При этом на изготовление единицы изделия вида А расходуется ***a1*** кг первого сорта, ***a2*** кг второго сорта и ***a3*** кг третьего сорта. На изготовление продукции вида В расходуется ***b1*** кг первого сорта, ***b2*** кг второго сорта, ***b3*** кг третьего сорта. На складе фабрики имеется всего материала первого сорта ***c1*** кг, второго сорта ***c2*** кг, третьего сорта ***c3*** кг. От реализации единицы готовой продукции вида А фабрика имеет прибыль вида ***а*** руб., а от реализации единицы готовой продукции вида В фабрика имеет прибыль вида ***в*** руб. Определить максимальную прибыль от реализации всей продукции видов А и В.

a1= 9, a2= 15, a3= 15, b1= 27, b2= 15, b3= 3, c1= 606, c2= 802, c3= 840,

а=11, в=6.

Задача 2. Имеются три пункта поставки однородного груза **A1, A2, A3** и пять пунктов **B1, B2, B3, B4, B5** потребления этого груза. На пунктах **A1, A2 и A3** находится груз соответственно в количестве ***a1, a2 и a3*** тонн. В пункты **B1, B2, B3, B4, B5** требуется доставить соответственно ***b1, b2, b3, b4, b5*** тонн груза. Расстояние между пунктами поставки и пунктами потребления приведено в таблице:

Пункты поставки	**Пункты потребления**				
	B1	**B2**	**B3**	**B4**	**B5**
A1	D11	D12	D13	D14	D15
A2	D21	D22	D23	D24	D25
A3	D31	D32	D33	D34	D35

Найти такой план закрепления потребителей за поставщиками однородного груза, чтобы общие затраты по перевозкам были минимальными. ***a1=200, a2=250, a3=200,***

b1=190, b2=100, b3=120, b4=110,

b5=130.

(28 27 18 27 24 >
D 18 26 27 32 21
27 33 23 31 34,

4 вариант.

Задача 1. Составить математическую модель следующей задачи. Предположим, что для производства продукции вида А и В можно использовать материал трех сортов. При этом на изготовление единицы изделия вида А расходуется ***a1*** кг первого сорта, ***a2*** кг второго сорта и ***a3*** кг третьего сорта. На изготовление продукции вида В расходуется ***b1*** кг первого сорта, ***b2*** кг второго сорта, ***b3*** кг третьего сорта. На складе фабрики имеется всего материала первого сорта ***c1*** кг, второго сорта ***c2*** кг, третьего сорта ***c3*** кг. От реализации единицы готовой продукции вида А фабрика имеет прибыль вида ***а*** руб., а от реализации единицы готовой продукции вида В фабрика имеет прибыль вида ***в*** руб. Определить максимальную прибыль от реализации всей продукции видов А и В.

a1= 13, a2= 13, a3= 11, b1= 23, b2= 11, b3= 1, c1= 608, c2= 614, c3= 575,

a=5, в=7.

Задача 2. Имеются три пункта поставки однородного груза **A1, A2, A3** и пять пунктов **B1, B2, B3, B4, B5** потребления этого груза. На пунктах **A1, A2 и A3** находится груз соответственно в количестве ***a1, a2 и a3*** тонн. В пункты **B1, B2, B3, B4, B5** требуется доставить соответственно ***b1, b2, b3, b4, b5*** тонн груза. Расстояние между пунктами поставки и пунктами потребления приведено в таблице:

Пункты поставки	**Пункты потребления**				
	B1	**B2**	**B3**	**B4**	**B5**
A1	D11	D12	D13	D14	D15
A2	D21	D22	D23	D24	D25
A3	D31	D32	D33	D34	D35

Найти такой план закрепления потребителей за поставщиками однородного груза, чтобы общие затраты по перевозкам были минимальными. ***a1=230, a2=250, a3=170,***

b1=140, b2=90, b3=160, b4=110, b5=150.

$$\begin{vmatrix} 40 & 19 & 25 & 25 & 35 \\ 39 & 26 & 27 & 18 & 38 \\ 46 & 27 & 36 & 40 & 45 \end{vmatrix}$$

5 вариант.

Задача 1. Составить математическую модель следующей задачи. Предположим, что для производства продукции вида А и В можно использовать материал трех сортов. При этом на изготовление единицы изделия вида А расходуется ***a1*** кг первого сорта, ***a2*** кг второго сорта и ***a3*** кг третьего сорта. На изготовление продукции вида В расходуется ***b1*** кг первого сорта, ***b2*** кг второго сорта, ***b3*** кг третьего сорта. На складе фабрики имеется всего материала первого сорта ***c1*** кг, второго сорта ***c2*** кг, третьего сорта ***c3*** кг. От реализации единицы готовой продукции вида А фабрика имеет прибыль вида ***a*** руб., а от реализации единицы готовой продукции вида В фабрика имеет прибыль вида ***в*** руб. Определить максимальную прибыль от реализации всей продукции видов А и В.

a1= 19, a2= 16, a3= 19, b1= 31, b2= 9, b3= 1, c1= 1121, c2= 706, c3= 1066,

a=16, в=19.

Задача 2. Имеются три пункта поставки однородного груза **A1, A2, A3** и пять пунктов **B1, B2, B3, B4, B5** потребления этого груза. На пунктах **A1, A2 и A3** находится груз соответственно в количестве ***a1, a2 и a3*** тонн. В пункты **B1, B2, B3, B4, B5** требуется доставить соответственно ***b1, b2, b3, b4, b5*** тонн груза. Расстояние между пунктами поставки и пунктами потребления приведено в таблице:

Пункты поставки	**Пункты потребления**				
	B1	**B2**	**B3**	**B4**	**B5**
A1	D11	D12	D13	D14	D15
A2	D21	D22	D23	D24	D25
A3	D31	D32	D33	D34	D35

Найти такой план закрепления потребителей за поставщиками однородного груза, чтобы общие затраты по перевозкам были минимальными.

a1=200,a2=300, a3=250, (20 10 13 13 18 A

b1=210, b2=150, b3=120, b4=135, D ,7 и 20 16 22

b5=135 [26 17 19 21 23 J

Методические указания по ходу выполнения работы

Построение математической модели процесса, явления или объекта начинается с построения упрощенного варианта модели, в котором учитываются только основные черты. В результате прослеживаются основные связи между входными параметрами, ограничениями и показателем эффективности. Общего подхода к построению модели нет. В каждом конкретном случае при построении математической модели учитывается большое количество факторов: цель построения модели, круг решаемых задач, точность описания модели и точность выполнения вычислений. Математическая модель должна отражать все существенные факторы, определяющие ее поведение, и при этом быть простой и удобной для восприятия результатов. Каждая математическая модель процесса, явления или объекта в своей основе имеет математический количественный метод.

Применение математических количественных методов для обоснования выбора того или иного управляющего решения во всех областях человеческой деятельности называется *исследованием операций.* Целью исследования операций является нахождение с использованием специального математического аппарата решения, удовлетворяющего заданным условиям. На самом деле при решении практически любой задачи имеется неограниченное количество решений. Множество решений, удовлетворяющих заданным условиям (ограничениям), называется допустимым множеством решением. Выбор из множества допустимых решений одного решения, наилучшего в каком-либо смысле, называемого *оптимальным* решением, и есть задача исследования операций.

Модель — это материальный или идеальный объект, заменяющий оригинал, наделенный основными характеристиками (чертами) оригинала и предназначенный для проведения некоторых действий над ним с целью получения новых сведений об оригинале.

Рис. 1. Классификация моделей

Рисунок 1. Классификация математических моделей

При построении математической модели необходимо обеспечить *достаточную* точность вычислений (точность решения) и *необходимую* подробность модели. Любая математическая

модель включает в себя описание основных, т. е. *необходимых* для исследования свойств и законов функционирования исследуемого объекта, процесса или явления. В своей основе каждая математическая модель имеет целевую функцию, которая описывает функционирование реального объекта, процесса или явления. В зависимости от исследуемого (моделируемого) объекта, явления или процесса *целевая функция* может быть представлена одной функциональной зависимостью, системой уравнений (линейных, нелинейных, дифференциальных и т. д.), набором статистических данных и т. д. При работе с целевой функцией исследователь воздействует на нее через ***набор входных параметров*** (рис. 2).

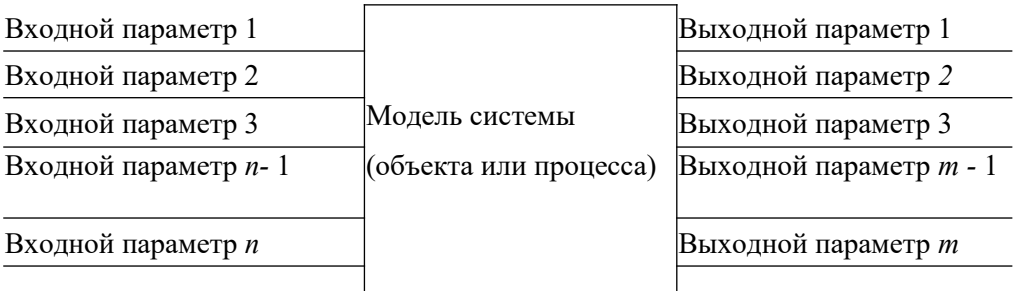


Рисунок 2. Обобщенная схема математической модели

По способу реализации математические модели можно разделить следующим образом.

1. Линейное программирование.

Математическая модель целиком (целевая функция и ограничения) описывается уравнениями первого порядка. Линейное программирование включает в себя несколько методов решения (задач):

- симплексный;
- графический;
- транспортная задача;
- целочисленное программирование.

2. Нелинейное программирование.

Целевая функция и ограничения, составляющие математическую модель, содержат хотя бы одно нелинейное уравнение (уравнение второго порядка и выше). Нелинейное программирование содержит несколько методов решения (задач):

- графический;
- регулярного симплекса;
- деформируемого многогранника (Нелдера - Мида);
- градиентный.

3. Динамическое программирование.

Ориентировано на решение задач прокладки магистралей кратчайшим путем и перераспределения различных видов ресурсов.

4. Сетевое планирование.

Решает проблему построения графика выполнения работ, распределения производственных, финансовых и людских ресурсов.

5. Принятие решений и элементы планирования.

В этом случае и качестве целевой функции выступает набор статистических данных или некоторые данные прогноза. Решением задачи являются рекомендации о способах поведения (стратегии). Решение носит рекомендательный характер (приблизительное решение). Выбор стратегии целиком остается за человеком — ответственным лицом, принимающим решение. Для принятия решения разработаны следующие теории:

- теория игр;

- системы массового обслуживания.

Тема лабораторной работы № 27. «Решение простейших однокритериальных задач», объем часов 1

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: определить оптимальное решение однокритериальных и многокритериальных задач в простейших случаях.

Задание(я):

Задание 1. Решить графическим способом задачу. Для производства двух видов,

P P S_1,S_2,S_3

изделии 1 и 2 используется, три вида сырья 1 2 3, запасы которого соответственно равны

P

100, 60, 180 единиц. Для производства одной единицы продукции 1 используется 2 единицы сырья S^1 и по 1 единице сырья 2 и 3. Для производства одной единицы продукции P_2 используется по 1 единице сырья $S^{1и}S^2$ и 4 единицы сырья 3^3. Прибыль от реализации 1 единицы PP

каждой продукции 1 и 2 соответственно равна 30 и 20 единиц. Необходимо составить такой

PP

план выпуска продукции 1 и 2, при котором суммарная прибыль будет наибольшей.

Решение.

Составим математическую модель задачи:

P

Пусть x_i - единица готовой продукции вида 1

P

x_2 - единица готовой продукции вида 2

P

Цель фабрики получить максимальную прибыль от реализации всей продукции видов 1 и P_2, тогда:

$F = 30 \bullet x_x + 20 \bullet x_2 \wedge \max$

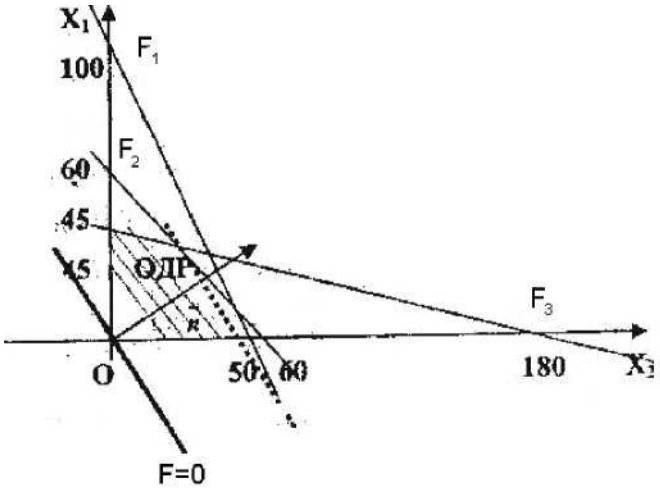
Система ограничений:

$2x + x_2 < 100$

$< x_i + x_2 < 60$

$\_x + 4x_2 < 180$

$X_j > 0, \quad x_2 > 0$ *условие неотрицательности*



Алгоритм решения:

1. Используя систему ограничений и условия неотрицательности, строим область допустимых решений.
2. Строим линию уровня $F = 0$. Линией уровня функции двух переменных называется линия, вдоль которой функция сохраняет свое постоянное значение.
3. Строим градиент целевой функции. Градиент функции - это вектор, имеющий своими координатами частные производные функции и показывающий направление наискорейшего роста значения функции. Так как целевая функция ЗЛП линейная, то линии

$$gradF = n$$

уровня целевой функции - прямые и , n- вектор нормали к этим прямым. :

4. Перемещаем линию уровня $F = 0$ вдоль градиента функции. Если ЗЛП на минимум, то оптимальное решение находится в первой точке, принадлежащей ОДР; если ЗЛП на максимум, то оптимальное решения находится в последней точке, принадлежащей ОДР.
Замечание.
При построении ОДР возможны случаи:

1. ОДР оказалась пустым множеством. В этом случае ЗЛП не имеет решения из-за несовместности системы ограничений.
2. ОДР оказалась либо выпуклым многоугольником, либо не ограниченной выпуклой многоугольной областью. Тогда ЗЛП имеет оптимальное решение, которое совпадает по крайней мере с одной из вершин ОДР.

Используя алгоритм решения и систему ограничений и условия неотрицательности, построим ОДР. Для этого во всех неравенствах системы ограничений и условия неотрицательности знак неравенства заменим на знак равенства. В результате будем иметь уравнения прямых:

$F : 2x + x = 100$

$F : x + x = 60$

$F : x + 4x = 180$

$x = 0, x = 0$

$X OX$

В системе координат 1 2 построим эти прямые. В результате будем иметь ОДР. В

$F = 0$ $gradF = n$

этой же системе координат строим линию уровня = и вектор

Так как задача на максимум, будем перемещать линию уровня F=0 вдоль вектора n до тех пор, пока она не пересечет ОДР в самом крайнем своем положении, т.е. при дальнейшем

перемещении она не будет с ОДР иметь общие точки. Такой точкой оказалась точка пересечения *FF* прямых [1] и [2].

Вычислим ее координаты.

$$\begin{cases} 2x_1 + x_2 = 100 \\ x_1 + x_2 = 60 \end{cases} \Rightarrow x_1 = 40, x_2 = 20, \quad F_{max} = 30 \cdot 40 + 20 \cdot 20 = 1600$$

PP

Таким образом, если предприятие будет выпускать продукцию вида [1] и [2], в количестве 40 и 20 единиц соответственно, то получит максимальную прибыль в размере 1600 единиц.

Задание 2. Фирме необходимо выбрать наилучший вариант закупки оборудования, если задана закупочная цена каждого из вариантов оборудования и время изготовления и доставки. Под наилучшим вариантом понимается вариант с минимальными закупочной стоимостью и временем доставки.

Если их несколько, выбираем из них точку с наименьшим значением $F_2(x)$. Включаем се в множество Парето. Отсекаем точки с большими либо равными значениями Fj(x) и $F_2(x)$ (северо-восточный угол с вершиной в выбранной точке). Повторяем процедуру для оставшейся части допустимой области.

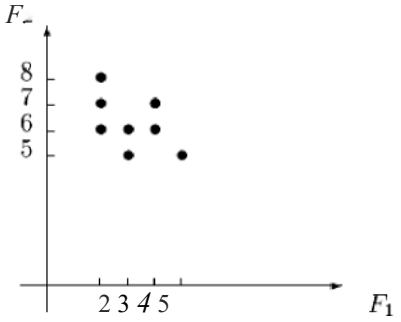
Из рисунка видно, что для нас представляют интерес пары (Fj.K,)е {(2,б). (3,5)} и соответствующие решения (xj,x_n)е j(2.2),(1.2)j, которые являются недоминируемыми и образуюг множество Парето рассматриваемой задачи.

Тема лабораторной работы № 28. «Задача Коши для уравнения теплопроводности», объем часов 1

Обозначим, соответственно, через $x_($ - номер, *F^x^)* - время изготовления и доставки, $F_2(x_l)$ - закупочную стоимость варианта закупки оборудования. Значения функций *F^Xf)* и $F,(x_(}$ заданы таблицей:

x_t	1	2	3	4	5	6	7	8
Fl(^l)	2	2	2	3	3	4	4	5
$F_2(X_l)$	6	7	8	5	6	6	7	5

Задача отыскания множества Парето в случае двух критериев вида $F_x(x)$->minH $F_2(x)$—> min может быть решена графически следующим образом. Находим все точки с наименьшим значением *F^x)*.



У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: определить оптимальное решение однокритериальных и многокритериальных задач в простейших случаях.

Задание(я):

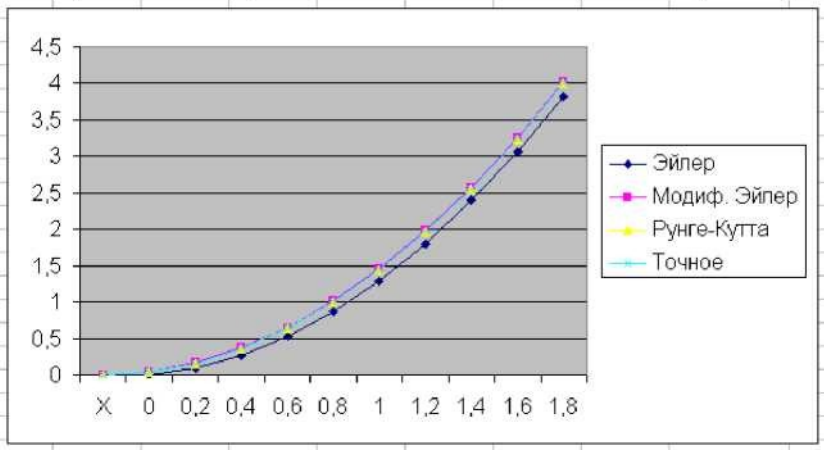
Решение обыкновенных дифференциальных уравнений (ОДУ)

В приведённом примере решается задача Коши, то есть, ищется решение дифференциального уравнения первого порядка вида $dy/dx = f(x,y)$ на интервале $x \in [x_0,x_n]$ при условии $y(x_0)=y_0$ и равномерном шаге сетки по x.

Решение выполняется методами Эйлера, "предиктор-корректор" (он же модифицированный метод Эйлера) и методом Рунге-Кутта 4 порядка точности. Пример может служить образцом для Ваших решений, правда, функцию придётся перепрограммировать несколько раз при различных значениях аргумента - поскольку без применения макросов на VBA Excel не позволяет создать полноценную функцию, которую было бы удобно вызывать с разными значениями аргументов.

Здесь решается уравнение $dy/dx = 2x-y+x^2$ на интервале [0,2], начальное значение y(0)=0, для оценки точности задано также точное решение в виде функции $u(x)=x^2$. Оценка погрешности делается в норме L_1, как и принято в данном случае.

	A	B	C	D	E	F	G	H	I	J	K	L
1	X	Эйлер	y~	Модиф. Эйлер	k1	k2	k3	k4	Рунге-Кутта	Точное		
2	0	0	0	0					0	0	A=	0
3	0,2	0	0	0,044	0	0,21	0,189	0,4022	0,040006667	0,04	B=	2
4	0,4	0,088	0,1232	0,16728	0,399993	0,609994	0,588994	0,802195	0,160012125	0,16	N=	10
5	0,6	0,2624	0,325824	0,36997	0,799988	1,009989	0,988989	1,20219	0,360016594	0,36	H=	0,2
6	0,8	0,52192	0,607976	0,652175	1,199983	1,409985	1,388985	1,602186	0,640020252	0,64		
7	1	0,865536	0,96974	1,013984	1,59998	1,809982	1,788982	2,002183	1,000023248	1		
8	1,2	1,292429	1,411187	1,455467	1,999977	2,209979	2,188979	2,402181	1,440025701	1,44		
9	1,4	1,801943	1,932373	1,976683	2,399974	2,609977	2,588977	2,802179	1,960027709	1,96		
10	1,6	2,393554	2,533346	2,57768	2,799972	3,009975	2,988975	3,202177	2,560029353	2,56		
11	1,8	3,066844	3,214144	3,258497	3,199971	3,409974	3,388973	3,602176	3,240030699	3,24		
12	2	3,821475	3,974798	4,019168	3,599969	3,809972	3,788972	4,002175	4,000031801	4		
13	Погрешн.	0,178525		0,019168					3,18007E-05			



Методические указания по ходу выполнения работы

Для выполнения работы пользуемся рекомендуемыми учебниками.

Тема лабораторной работы № 29. «Решение задач линейного программирования симплекс–методом», объем часов 1

У1 использовать выбранную систему контроля версий;

У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: научиться решать задачи геометрическим методом, научиться решать задачи симплексным методом, закрепить навыки записи взаимосвязи показателей задачи в виде математической модели.

Задание(я):

Задание 1

На предприятии имеется сырье видов I, II, III. Из него можно изготавливать изделия типов A и B. Пусть запасы видов сырья на предприятии составляют b1, b2, b3 ед. соответственно, изделие типа A дает прибыль c1 ден.ед., а изделие типа B - c2 ден.ед. Расход сырья на изготовление одного изделия задан в словных единицах таблицей.

Составить план выпуска изделий, при котором предприятие имеет наибольшую прибыль. Решить задачу графическим методом.

1 вариант								
Изделие	Сырье			Ь1	Ь2	Ь3	c1	c2
	I	II	III					
A	6	3	2	102	91	105	5	9
B	3	4	5					
2вариант								
Изделие	Сырье			Ь1	Ь2	Ь3	c1	c2
	I	II	III					
A	1	1	3	20	36	40	2	5
B	3	2	1					
3 вариант								
Изделие	Сырье			Ь1	Ь2	Ь3	c1	c2
	I	II	III					
A	2	1	3	40	34	46	1	2
B	2	2	1					
4 вариант								
Изделие	Сырье			Ь1	Ь2	Ь3	c1	c2
	I	II	III					
A	3	4	3	300	520	600	6	3
B	1	3	4					

Задание 2

Предприятие выпускает три вида изделий (N1, N2, N3), используя три вида ресурсов (P1, P2, P3). Запасы ресурсов (З) ограничены. Прибыль от реализации (П) единицы изделия и нормы расхода ресурсов представлены в таблице. Определить ассортимент и объемы выпуска продукции, получаемую прибыль, величину остатков. Найти решение задачи симплексным методом с представлением всех симплексных таблиц и проанализировать полученные результаты.

Вариант 1				
	N1	N2	N3	З
P1	1	8	5	43
P2	4	1	6	74
P3	5	2	2	35
П	5	7	6	

Вариант 2				
	N1	N2	N3	З
Р1	3	5	4	81
Р2	6	1	3	74
Р3	1	5	2	33
П	4	8	7	

Вариант 3				
	N1	N2	N3	З
Р1	6	7	2	57
Р2	6	6	1	97
Р3	3	7	8	63
П	5	6	8	

Вариант 4				
	N1	N2	N3	З
Р1	7	8	3	81
Р2	4	1	6	68
Р3	5	1	7	54
П	2	5	6	

Методические указания по ходу выполнения работы

Для выполнения работы пользуемся рекомендуемыми учебниками.

Тема лабораторной работы № 30. «Нахождение начального решения транспортной задачи. Решение транспортной задачи методом потенциалов», объем часов 1

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.;

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

Задание(я):

1 вариант.

Задача. Имеются три пункта поставки однородного груза **A1, A2, A3** и пять пунктов **B1, B2, B3, B4, B5** потребления этого груза. На пунктах **A1, A2 и A3** находится груз соответственно в количестве ***a1, a2 и a3*** тонн. В пункты **B1, B2, B3, B4, B5** требуется доставить соответственно ***b1, b2, b3, b4, b5*** тонн груза. Расстояние между пунктами поставки и пунктами потребления приведено в таблице:

Пункты	**Пункты потребления**

поставки	**B1**	**B2**	**B3**	**B4**	**B5**
A1	D11	D12	D13	D14	D15

A2	D21	D22	D23	D24	D25
A3	D31	D32	D33	D34	D35

Найти такой план закрепления потребителей за поставщиками однородного груза, чтобы общие затраты по перевозкам были минимальными.

a1=200, a2=350, a3=300,

b1=270, b2=130, b3=190, b4=150, b5=110

$$D = \begin{vmatrix} '24 & 50 & 55 & 27 & 16 \\ 50 & 47 & 23 & 17 & 21 \\ 35 & 59 & 55 & 27 & 41> \end{vmatrix}$$

2 вариант.

Задача. Имеются три пункта поставки однородного груза **A1, A2, A3** и пять пунктов **B1, B2, B3, B4, B5** потребления этого груза. На пунктах **A1, A2 и A3** находится груз соответственно в количестве ***a1, a2 и a3*** тонн. В пункты **B1, B2, B3, B4, B5** требуется доставить соответственно ***b1, b2, b3, b4, b5*** тонн груза. Расстояние между пунктами поставки и пунктами потребления приведено в таблице:

Пункты поставки	**Пункты потребления**				
	B1	**B2**	**B3**	**B4**	**B5**
A1	D11	D12	D13	D14	D15
A2	D21	D22	D23	D24	D25
A3	D31	D32	D33	D34	D35

Найти такой план закрепления потребителей за поставщиками однородного груза, чтобы общие затраты по перевозкам были минимальными.

a1=200, a2=300, a3=250,

b1=210, b2=150, b3=120, b4=135, b5=135

$$D = \begin{pmatrix} 20 & 10 & 13 & 13 & 18 \\ 27 & 19 & 20 & 16 & 22 \\ ^{\wedge}26 & 17 & 19 & 21 & 23_{y} \end{pmatrix}$$

3 вариант.

Задача. Имеются три пункта поставки однородного груза **A1, A2, A3** и пять пунктов **B1, B2, B3, B4, B5** потребления этого груза. На пунктах **A1, A2 и A3** находится груз соответственно в количестве ***a1, a2 и a3*** тонн. В пункты **B1, B2, B3, B4, B5** требуется доставить соответственно ***b1, b2, b3, b4, b5*** тонн груза. Расстояние между пунктами поставки и пунктами потребления приведено в таблице:

Пункты поставки	**Пункты потребления**				
	B1	**B2**	**B3**	**B4**	**B5**
A1	D11	D12	D13	D14	D15
A2	D21	D22	D23	D24	D25
A3	D31	D32	D33	D34	D35

Найти такой план закрепления потребителей за поставщиками однородного груза, чтобы общие затраты по перевозкам были минимальными.

a1=230, a2=250, a3=170,

b1=140, b2=90, b3=160, b4=110, b5=150.

$$D = \begin{pmatrix} 40 & 19 & 25 & 25 & 35^{n} \\ 49 & 26 & 27 & 18 & 38 \\ 46 & 27 & 36 & 40 & 45 \end{pmatrix}$$

4 вариант.

Задача. Имеются три пункта поставки однородного груза **A1, A2, A3** и пять пунктов **B1, B2, B3, B4, B5** потребления этого груза. На пунктах **A1, A2 и A3** находится груз соответственно в количестве ***a1, a2 и a3*** тонн. В пункты **B1, B2, B3, B4, B5** требуется доставить соответственно ***b1, b2, b3, b4, b5*** тонн груза. Расстояние между пунктами поставки и пунктами потребления приведено в таблице:

Пункты поставки	Пункты потребления				
	B1	**B2**	**B3**	**B4**	**B5**
A1	D11	D12	D13	D14	D15
A2	D21	D22	D23	D24	D25
A3	D31	D32	D33	D34	D35

Найти такой план закрепления потребителей за поставщиками однородного груза, чтобы общие затраты по перевозкам были минимальными.

a1=350, a2=200, a3=300,

b1=170, b2=140, b3=200, b4=195, b5=145

$$D = \begin{pmatrix} 22 & 14 & 16 & 28 & 30 \\ 19 & 17 & 26 & 36 & 36 \\ 30 & 31 & 39 & 41 & \end{pmatrix}$$

5 вариант.

Задача. Имеются три пункта поставки однородного груза **A1, A2, A3** и пять пунктов **B1, B2, B3, B4, B5** потребления этого груза. На пунктах **A1, A2 и A3** находится груз соответственно в количестве ***a1, a2 и a3*** тонн. В пункты **B1, B2, B3, B4, B5** требуется доставить соответственно ***b1, b2, b3, b4, b5*** тонн груза. Расстояние между пунктами поставки и пунктами потребления приведено в таблице:

Пункты поставки	Пункты потребления				
	B1	**B2**	**B3**	**B4**	**B5**
A1	D11	D12	D13	D14	D15
A2	D21	D22	D23	D24	D25
A3	D31	D32	D33	D34	D35

b1=210, b2=150, b3=120, b4=135, b5=135.

$$D = \begin{pmatrix} 4 & 8 & 13 & 27 \\ 4 & 11 & 9 & 17 \\ 16 & 10 & 1 & 4 \end{pmatrix}$$

Найти такой план закрепления потребителей за поставщиками однородного груза, чтобы общие затраты по перевозкам были минимальными.

a1=300, a2=250, a3=200,

Методические указания по ходу выполнения работы

Для выполнения работы пользуемся рекомендуемыми учебниками.

Тема лабораторной работы № 31. «Применение метода стрельбы для решения линейной краевой задачи», объем часов 1

У1 использовать выбранную систему контроля версий;

У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: научиться применять метод стрельбы для решения линейной краевой задачи.

Задание(я):

1. Начните выполнение работы с темы «*Линейная краевая задача*». Выбрав с помощью меню один из методов решения линейной краевой задачи, перейдите к пункту меню *«Параметры»*.
   1.1. Найдите решение этой задачи методом построения общего решения и методом прогонки для разных *p*, начиная с умеренных значений и увеличивая их до величины порядка 1200. Сравните получаемые решения с точным и объясните наблюдаемые эффекты. Попытайтесь найти решение этой же задачи методом стрельбы. Проанализируйте, как влияет при разных *p* точность задания недостающего начального условия на левом конце интервала на успешное решение задачи методом стрельбы.
   1.2. Объясните полученные результаты..
2. Получите численное решение следующих нелинейных краевых задач:
3. Рассмотрите краевые задачи.
   3.1. Рассмотреть две сингулярно-возмущенные задачи (с малым параметром при старших производных):

Методические указания по ходу выполнения работы

Для выполнения работы пользуемся рекомендуемыми учебниками.

Тема лабораторной работы № 32. «Задача о распределении средств между предприятиями», объем часов 1

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

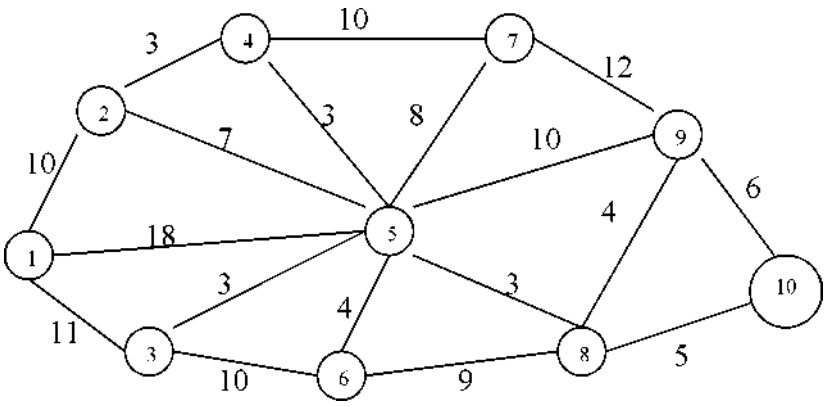
Цель лабораторной работы: Решить простейшие задачи методом динамического программирования.

Задание(я):

Задача 1. Планируется работа двух отраслей производства А и В на 4 года. Количество x средств, вложенных в отрасль А, позволяет получить доход 2x и уменьшается до 0,6x. Количество y средств, вложенных в отрасль В, позволяет получить доход 3y и уменьшается до *S S S>00^ S 5 5,, 50 0 S 500 S₀ S S0 S S₀* = 850

0,2y. Необходимо распределить выделенные ресурсы в количестве ◦ единиц между отраслями по годам планируемого периода для получения максимальной прибыли за весь период.

Задача 2. По заданной схеме, соединяющей 10 точек, найти кратчайшее расстояние от 1 точки до 10.



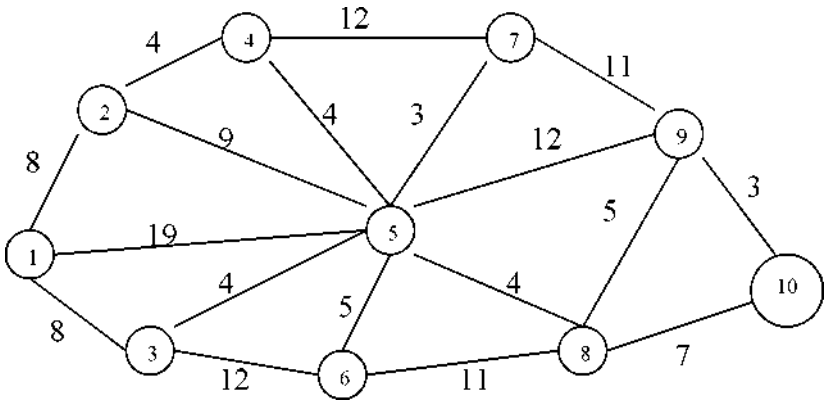
1 вариант.

S_0 ^ 900

Задача 1. Двум предприятиям А и В на 4 квартала выделено ◦ единиц средств.

Каждый квартал предприятие А получает x средств, предприятие В - y средств. При этом от выделенных средств предприятие А получает 4x единиц и остаток средств 0,3x единиц, а предприятие В - доход 5y единиц и остаток выделенных средств 0,1y единиц. Необходимо распределить средства между предприятиями поквартально таким образом, чтобы за весь год оба предприятия получили максимальный доход.

Задача 2. По заданной схеме, соединяющей 10 точек, найти кратчайшее расстояние от 1 точки до 10.



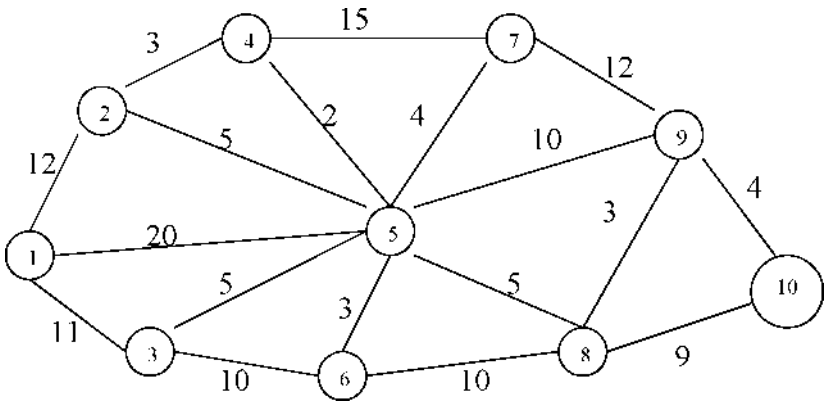
2 вариант.

Задача 1. Планируется работа двух отраслей производства А и В на 4 года. Количество x средств, вложенных в отрасль А, позволяет получить доход 5x и уменьшается до 0,1x. Количество y средств, вложенных в отрасль В, позволяет получить доход 3y и уменьшается до

S^K 1 nn,, io_a *s_oS 0,^10* *s S S₀* = 1100 ₀

0,5y. Необходимо распределить выделенные ресурсы в количестве ◦ единиц между отраслями по годам планируемого периода для получения максимальной прибыли за весь период.

Задача 2. По заданной схеме, соединяющей 10 точек, найти кратчайшее расстояние от 1 точки до 10.



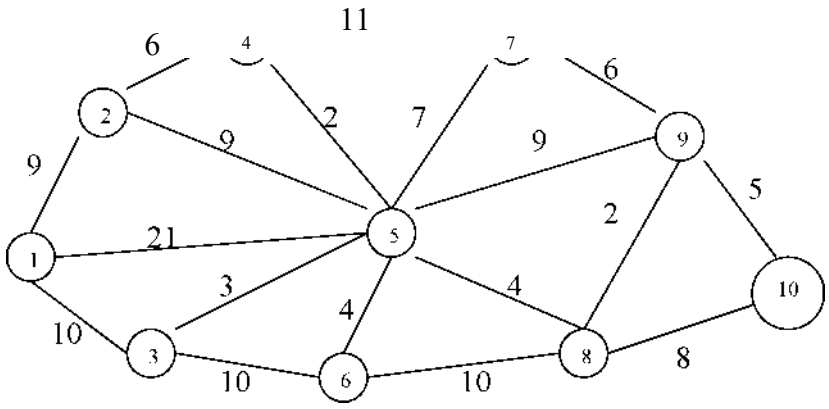
3 вариант.

$S\,S\,S$ $S_n = 750$

Задача 1. Двум предприятиям А и В на 4 квартала выделено ° единиц средств.

Каждый квартал предприятие А получает х средств, предприятие В - у средств. При этом от выделенных средств предприятие А получает 4х единиц и остаток средств 0,3х единиц, а предприятие В - доход 3у единиц и остаток выделенных средств 0,6у единиц. Необходимо распределить средства между предприятиями поквартально таким образом, чтобы за весь год оба предприятия получили максимальный доход.

Задача 2. По заданной схеме, соединяющей 10 точек, найти кратчайшее расстояние от 1 точки до 10.



4 вариант.

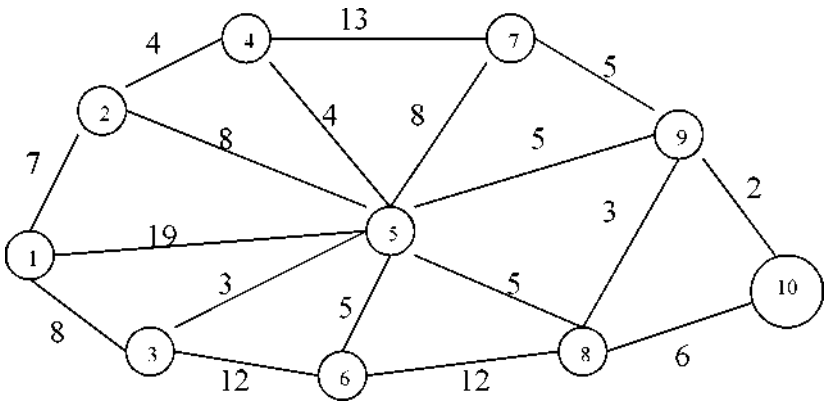
Задача 1. Планируется работа двух отраслей производства А и В на 4 года. Количество х средств, вложенных в отрасль А, позволяет получить доход 5х и уменьшается до 0,3х. Количество у средств, вложенных в отрасль В, позволяет получить доход 6у и уменьшается до

^o^ 800

0,1у. Необходимо распределить выделенные ресурсы в количестве ° единиц между

отраслями по годам планируемого периода для получения максимальной прибыли за весь период.

Задача 2. По заданной схеме, соединяющей 10 точек, найти кратчайшее расстояние от 1 точки до 10.



Методические указания по ходу выполнения работы

Для выполнения работы пользуемся рекомендуемыми учебниками.

Тема лабораторной работы № 33. «Задача о замене оборудования», объем часов 1

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.;
Цель лабораторной работы: научиться решать задачи динамического программирования, научиться разбивать весь процесс решения задачи на этапы, научиться выбирать оптимальную стратегию поведения.

Задание(я):

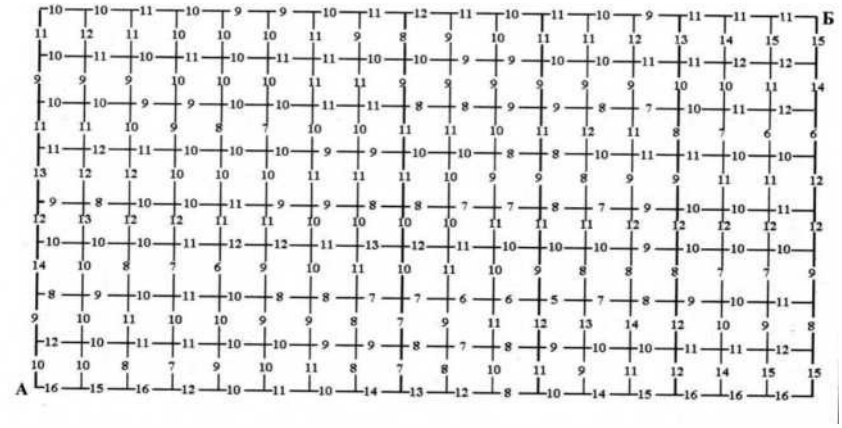
Задание 1. Распределите оптимальным образом денежные средства инвестора величиной *5 у.е.* между четырьмя предприятиями. Доход каждого предприятия от вложения в него *u* у.е.определяется функцией дохода f(u). Эти функции приведены в таблице___

u	Прибыль от внедрения по предприятиям			
	f4(u)	f3(u)	f2(u)	f1(u)
1	f4(1)	6	3	4
2	10	f3(2)	4	6
3	11	11	f2(3)	8
4	12	13	11	f1(4)
5	18	15	18	16

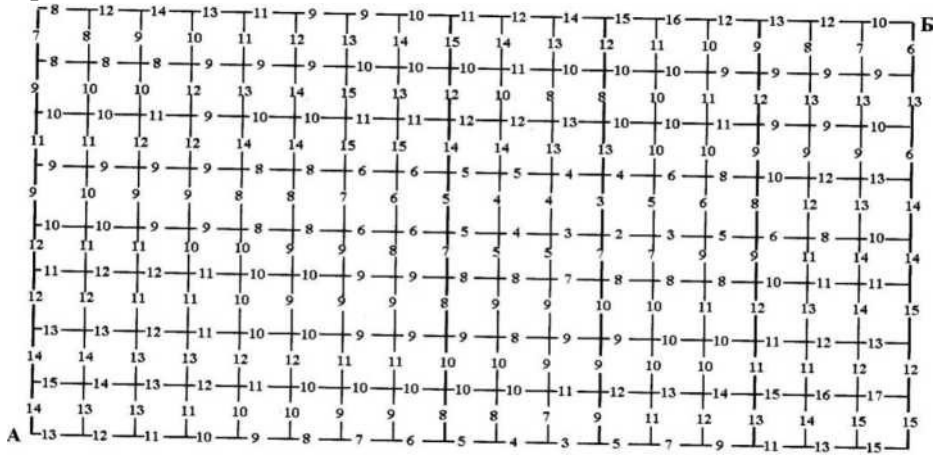
Вариант	**1**	**2**	**3**	**4**
f4(1)	9	5	6	8
f3(2)	10	10	7	7
f2(3)	7	5	8	9
f1(4)	10	9	13	15

Задание 2. Из пункта А в пункт В необходимо проложить автомобильную трассу по самому экономичному пути.

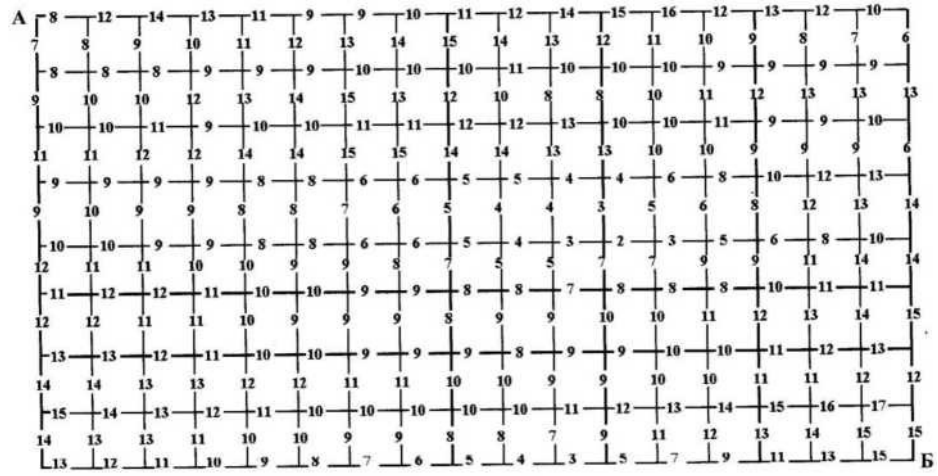
Вариант 1



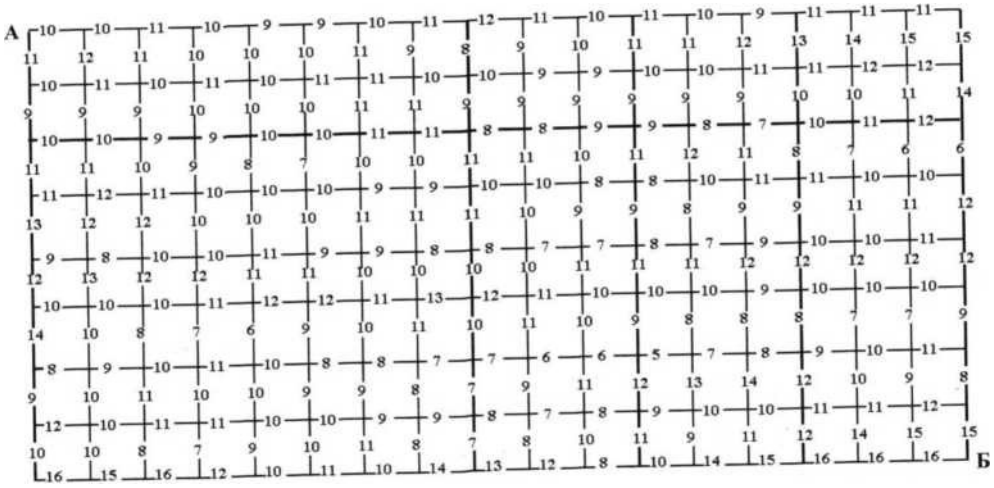
Вариант 2



Вариант 3



Вариант 4



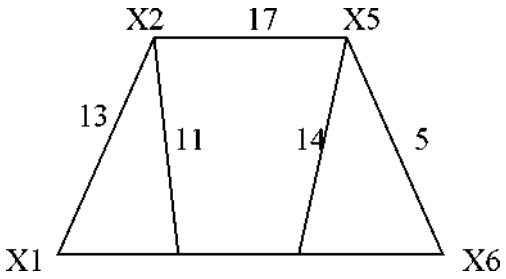
Методические указания по ходу выполнения работы

Для выполнения работы пользуемся рекомендуемыми учебниками.

Тема лабораторной работы № 34. «Нахождение кратчайших путей в графе. Решение задачи о максимальном потоке», объем часов 1
У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.
Цель лабораторной работы: Решить задачу о нахождении кратчайших путей в графе. Решить задачу о нахождении максимального потока.

Задание(я):

На представленном графе найдите: а) минимальный остов дерева, б) найдите кратчайший путь от начальной точки X1 до всех остальных точек.



Методические указания по ходу выполнения работы

Для выполнения работы пользуемся рекомендуемыми учебниками.

Тема лабораторной работы № 35. «Моделирование прогноза», объем часов 1

У1 использовать выбранную систему контроля версий;

У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: Изучить основы верстки. Научиться управлять интерфейсом мобильного устройства при разработке программного приложения. изучение возможностей и формирование умения использования универсальной компьютерной технологии для решения задач выявления тенденций и прогнозирования развития процесса на основе моделирования рядов динамики

Задание(я):

Имеются две наблюдаемые величины *x* и *y*, например, объем реализации фирмы, торгующей кондитерскими изделиями, за ряд лет ее работы. Необходимо выяснить какая из наиболее распространенных функциональных зависимостей подходит для описания процесса реализации товара, и какого результата по объемам продаж можно ожидать в последующие годы работы фирмы. Для того чтобы построить прогноз развития какой-либо ситуации на практике зачастую необходимо знать закономерность изменения исследуемой величины или объекта.

Для выявления тенденций развития процесса продаж необходимо построить тренды и осуществить их анализ. Построим и проанализируем, как описывают процесс динамики продаж линейная, логарифмическая, полиномиальная, степенная и экспоненциальная зависимости.

Создайте новую рабочую книгу.

1. Выберите таблицу с данными согласно своему индивидуальному варианту.
2. Сохраните результат работы в файл.
3. В ячейку A1 введите - описание переменной x, в ячейку B1 - описание переменной y.
4. Осуществите ввод исследуемых данных в столбцы A и B ниже описанных переменных.
5. Оформите созданную расчетную таблицу
6. Сохраните результат работы в файл.
7. Установить курсор в ячейку C1 и постройте диаграмму “Объем реализации продукции за наделю” по диапазону значений столбца B.
8. Произведите оформление построенной диаграммы
9. Сохраните результат работы в файл.
10. Выберите Зависимость 1 согласно индивидуальному варианту тип для первой линии тренда.
11. Постройте первый тренд для диаграммы.
12. Произведите настройку оформления вида полученного тренда
13. Выберите Зависимость 2 согласно индивидуальному варианту тип для второй линии тренда.
14. Постройте второй тренд для диаграммы.
15. Произведите настройку оформления вида построенных трендов
16. Произведите анализ полученных результатов.
17. Сохраните результат работы в файл.
18. Предъявите работу преподавателю.

Заключительные действия

19. Закройте все открытые файлы электронной таблицы.

Вариант 1 _____ _____

День	1	2	3	4	5	6	7	8
Количество проданных ящиков деталей	13	19	29	30	37	44	49	55

Исследуемые зависимости: линейная, степенная.

Вариант 2

Неделя	1	2	3	4	5	6	7	8	9	10
Количество поступивших упаковок продукции	9	16	20	27	34	39	44	52	58	64

Исследуемые зависимости: экспоненциальная, логарифмическая.

Вариант 3

День	1	2	3	4	5	6	7	8	9
Количество отпущенных флаконов пеногерметика	7	17	19	28	35	42	41	52	57

Исследуемые зависимости: полиномиальная, экспоненциальная.

Вариант 4

День	1	2	3	4	5	6	7	8	9
Количество заказанных пачек медикамента *С*	12	21	30	36	44	54	61	70	78

Исследуемые зависимости: логарифмическая, линейная

Вариант 5.....

Месяц	1	2	3	4	5	6	7	8	9	10	11
Количество заказов на переплетные работы	12	17	23	32	35	40	48	54	59	65	72

Исследуемые зависимости: степенная, полиномиальная.

Вариант 6

Час	1	2	3	4	5	6	7	8	9
Количество проданных бутылок напитка К	10	18	22	28	34	39	46	51	54

Исследуемые зависимости: линейная, экспоненциальная.

Вариант 7

Неделя	1	2	3	4	5	6	7	8
Количество проданных подержанных машин	12	18	25	32	40	46	53	60

Исследуемые зависимости: экспоненциальная, линейная.

Вариант 8

День	1	2	3	4	5	6	7	8	9
Количество заказов на хлебобулочное изделие N	14	23	30	39	45	54	63	70	78

Исследуемые зависимости: полиномиальная, линейная.

Вариант 9

Месяц	1	2	3	4	5	6	7	8	9	10	11
Количество проданных сувениров А	15	22	26	33	40	45	52	58	63	69	78

Исследуемые зависимости: логарифмическая, экспоненциальная.

Вариант 10

Неделя	1	2	3	4	5	6	7	8
Количество заказов на установку машинной сигнализации	9	15	24	29	38	46	52	58

Исследуемые зависимости: степенная, логарифмическая.

Вариант 11.....

Неделя	1	2	3	4	5	6	7	8	9	10	11
Количество	9	12	17	23	30	36	40	48	54	65	76

заказов на ремонт стиральных машин											

Исследуемые зависимости: линейная, полиномиальная.

Вариант 12

День	1	2	3	4	5	6	7	8
Количество абитуриентов интересующихся специальностью Z	13	19	26	30	37	44	49	55

Исследуемые зависимости: экспоненциальная, линейная.

Вариант 13

Месяц	1	2	3	4	5	6	7	8
Количество заказов на литературу типа X	12	18	25	32	40	46	53	60

Исследуемые зависимости: полиномиальная, экспоненциальная.

Вариант 14

День	1	2	3	4	5	6	7	8	9

Количество проданных флаконов шампуня В	7	17	19	28	35	42	41	52	57

Исследуемые зависимости: логарифмическая, линейная.

Вариант 15

Неделя	1	2	3	4	5	6	7	8
Количество проданных ящиков кондитерской продукции типа Ш	9	15	24	29	38	46	52	58

Исследуемые зависимости: степенная, полиномиальная.

Методические указания по ходу выполнения работы

Для выполнения работы пользуемся рекомендуемыми учебниками.
Задачу решить с помощью электронных таблиц.

Тема лабораторной работы № 36. «Выбор оптимального решения с помощью дерева решений», объем часов 1

У1 использовать выбранную систему контроля версий;
У2 использовать методы для получения кода с заданной функциональностью и степенью качества.

Цель лабораторной работы: освоение основных методов и способов построения деревьев решений, приобретение практических навыков по использованию инструментария Deductor.

Задание(я):

1. Разработать сценарии построения дерева решений и проведения анализа «что - если».
2. По таблице (например, продаж) создать таблицу транзакций с полями (например, Менеджер, Организация, Вид товара). Таблицу получить путем слияния соответствующих полей из разных таблиц и последующей группировки.
3. Разработать сценарии построения дерева решений с представлением правил, наиболее популярных наборов и анализа «что - если» с входными полями (например, Менеджер и Организация) и выходным полем (например, Вид товара).
4. Создать отчеты по всем разработанным сценариям.
5. Продемонстрировать проект преподавателю с использованием тестовых наборов данных и защитить работу.

Методические указания по ходу выполнения работы

Деревья решений (decision trees) являются одним из самых мощных средств решения задачи отнесения какого-либо объекта (строчки набора данных) к одному из заранее известных классов. Дерево решений - это классификатор, полученный из обучающего множества, содержащего объекты и их характеристики, на основе обучения. Дерево состоит из узлов и листьев, указывающих на класс.

Результатом работы алгоритма является список иерархических правил образующих дерево. Каждое правило - это интуитивно-понятная конструкция вида «Если...то...» (if - then). Дерево может использоваться для классификации объектов, не вошедших в обучающее множество. Чтобы принять решение, к какому классу следует отнести некоторый объект или ситуацию, требуется ответить на вопросы, стоящие в узлах этого дерева, начиная с его корня. Вопросы имеют вид «значение параметра A больше B?». Если ответ положительный, осуществляется переход к правому узлу следующего уровня; затем снова следует вопрос, связанный с соответствующим узлом и т. д.

Настройка назначения полей

Необходимо определить, как будут использоваться поля исходного набора данных при обучении дерева и дальнейшей практической работе с ним.

В левой части окна представлен список всех полей исходного набора данных. Для настройки поля следует выделить его в списке, при этом в правой части окна будут отображены текущие параметры поля:

1. Имя поля - идентификатор поля, определенный для него в источнике данных. Изменить его здесь нельзя.
2. Тип данных - тип данных, содержащихся в поле (вещественный, строковый, дата). Он также задается в источнике данных и здесь изменен быть не может.
3. Назначение - здесь необходимо выбрать порядок использования данного поля при обучении и работе дерева решений. Выбор производится с помощью списка, открываемого кнопкой и содержащего следующие варианты:

- Входное - значения поля будут являться исходными данными для построения и дальнейшей практической работы дерева решений, на их основе будет производиться классификация.
- Выходное - будет содержать результаты классификации. Выходное поле может быть только одно и оно должно быть дискретным.
- Информационное - поле не будет использоваться при обучении дерева, но будет помещено в результирующий набор в исходном состоянии.
- Неиспользуемое - поле не будет использоваться при построении и работе дерева решений и будет исключено из результирующей выборки. В отличие от непригодного, такое поле может быть использовано, если в этом возникнет необходимость.
- Непригодное - поле не может быть использовано при построении и работе алгоритма, но будет помещено в результирующий набор в исходном состоянии.

4. Вид данных - указывает на характер данных, содержащихся в поле (непрерывный или дискретный). Изменить это свойство здесь нельзя.

Статус непригодного поля устанавливается только автоматически и в дальнейшем может быть изменен только на неиспользуемое или информационное. Поле будет запрещено к использованию если:

- поле является дискретным и содержит всего одно уникальное значение;
- непрерывное поле с нулевой дисперсией;
- поле содержит пропущенные значения.

В случае если текущее поле содержит непрерывные (числовые) данные, отображается секция «Статистика», где показываются максимальное и минимальное значения поля, его среднее значение и стандартное отклонение. Если выделенное поле содержит дискретные (строковые) данные, то для него открывается секция «Уникальные значения», в которой отображается общее число уникальных значений поля, а также список самих уникальных значений.

Нормализация полей

Целью нормализации значений полей является преобразование данных к виду, наиболее подходящему для обработки средствами пакета Deductor. Для дерева решений данные, поступающие на вход, должны иметь числовой тип. В этом случае нормализатор может преобразовать дискретные данные к набору уникальных индексов.

Окно настройки нормализации полей вызывается с помощью кнопки «Настройка нормализации». В окне слева приведен полный список входных и выходных полей. При этом каждое поле помечено значком, обозначающим вид нормализации поля:

- линейная - линейная нормализация исходных значений;
- уникальные значения - преобразование уникальных значений в их индексы.

Для числовых (непрерывных) полей с линейной нормализацией дополнительные параметры недоступны. В полях «Минимум» и «Максимум» секции «Диапазон значений» можно посмотреть минимальное и максимальное значения этого поля.

Для дискретных полей справа находится список уникальных значений поля, где для каждого значения указывается количество его повторений. Поле «Количество значений» показывает общее число уникальных значений, принимаемых полем.

Настройка обучающей выборки

Обучающая выборка может быть разбита на три множества - обучающее, тестовое и валидационное.

1. Обучающее множество - включает записи (примеры), которые будут использоваться в качестве входных данных, а также соответствующие желаемые выходные значения.
2. Тестовое множество - также включает записи, содержащие входные и желаемые выходные значения, но используемое не для обучения модели, а для проверки его результатов.
3. Валидационное множество - множество примеров, используемое как для оценки результатов обучения модели, так и определения ее параметров.

Для разбиения исходного множества на обучающее, тестовое и валидационное необходимо настроить несколько параметров:

1. Из списка «Способ разделения исходного множества» выбирается порядок отбора записей во все три множества. Если выбран вариант «по порядку», то порядок следования записей при их разделении не меняется. Множества последовательно формируются в соответствии с определенным для них числом записей. Если выбран вариант «случайно», то отбор записей происходит случайным образом.
2. Затем необходимо указать, какие множества будут использоваться. Для того чтобы множество было сформировано, нужно установить флажок слева от его названия. Если флажок сброшен, то множество использовано не будет. Обучающее множество используется всегда, поэтому сбросить флажок для него нельзя.
3. Для каждого из используемых множеств необходимо задать его размер. Размер может быть задан непосредственно количеством записей или в процентах от объема исходной выборки. Для этого достаточно дважды щелкнуть мышью в соответствующей клетке и ввести нужное значение с клавиатуры. При этом размер, введенный в процентах, автоматически пересчитывается в количество строк и наоборот. В поле «Количество строк (всего)» отображается общее количество записей в исходной выборке данных, которое может быть задействовано для формирования множеств. Если суммарное число строк для всех используемых множеств меньше полного числа строк исходной выборки, то размеры множеств можно задавать произвольно. Можно, например, использовать не все записи, а только часть из них. Если же суммарное указанное число строк превышает максимальное для данной исходной выборки, то автоматически включается баланс множеств, т.е. при указании для одного из множеств размера, в

результате которого будет превышено максимальное число записей в исходной выборке, размер остальных множеств будет автоматически уменьшен таким образом, чтобы суммарный размер множеств не превышал доступного числа записей. В строке «Итого» указывается количество записей, задействованных во всех используемых множествах, а также процент от полного числа записей исходной выборки, который они составляют.

4. В столбце «Порядок сортировки» можно определить порядок следования записей внутри каждого множества. Для этого необходимо дважды щелкнуть мышью в столбце «Порядок сортировки» для соответствующего множества и с помощью появившейся кнопки выбора открыть список, в котором выбрать один из возможных вариантов:

- по возрастанию - записи в данном множестве будут следовать в порядке возрастания;
- по убыванию - записи в данном множестве будут следовать в порядке убывания;
- случайно - записи в данном множестве будут следовать в случайном порядке.

Для того чтобы обучающее множество было репрезентативным необходимо, чтобы все уникальные значения всех дискретных столбцов содержались в данном наборе данных.

Настройка параметров обучения

Необходимо установить параметры, в соответствии с которыми будет проводиться обучение дерева:

1. «Минимальное количество примеров, при котором будет создан новый узел» - задается минимальное количество примеров, которое возможно в узле. Если примеров, которые попадают в данный узел, будет меньше заданного - узел считается листом (т.е. дальнейшее ветвление прекращается).
2. «Строить дерево с более достоверными правилами в ущерб сложности» - установка данного флажка включает специальный алгоритм, который, усложняя структуру дерева, увеличивает достоверность результатов классификации. Сброс данного флажка хотя и приводит к упрощению дерева, снижает достоверность результатов классификации.
3. «Уровень доверия, используемый при отсечении узлов дерева». Значение этого параметра задается в процентах и должно лежать в пределах от 0 до 100. Эти значения выбираются из списка. Чем больше уровень доверия, тем более ветвистым получается дерево, и, соответственно, чем меньше уровень доверия, тем больше узлов будет отсечено при его построении.

II. Общие рекомендации

По всем вопросам, связанным с изучением дисциплины (включая самостоятельную работу), консультироваться с преподавателем.

III. Контроль и оценка результатов

Оценка за выполнение лабораторной работы выставляется в форме *по пятибалльной системе* и учитывается как показатель текущей успеваемости студента.

Качественная оценка индивидуальных образовательных достижений		**Критерии оценки результата**
балл (оценка)	**вербальный аналог**	
5	отлично	Представленные работы высокого качества, уровень выполнения отвечает всем

		требованиям, теоретическое содержание курса освоено полностью, без пробелов, необходимые практические навыки работы с освоенным материалом сформированы, выполнены все предусмотренные работой задания.
4	хорошо	Уровень выполнения работы отвечает всем требованиям, теоретическое содержание курса освоено полностью без пробелов, некоторые практические навыки работы с освоенным материалом сформированы недостаточно, все предусмотренные работой задания выполнены, некоторые из выполненных заданий, возможно, содержат ошибки.
3	удовлетворительно	Уровень выполнения работы отвечает большинству основных требований, теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые практические навыки работы с освоенным материалом в основном сформированы, большинство предусмотренных работой заданий выполнено, некоторые виды заданий выполнены с ошибками.
2	не удовлетворительно	Теоретическое содержание курса освоено частично, необходимые практические навыки работы не сформированы, большинство предусмотренных работой заданий не выполнено.