Министерство образования и науки Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Курский государственный университет»

Колледж коммерции, технологий и сервиса

Методические рекомендации

по выполнению практических работ

по МДК.02.01 Разработка, внедрение и адаптация программного обеспечения отраслевой направленности

(ЧАСТЬ 1)

для студентов 2-3 курса специальности
09.02.05 Прикладная информатика (по отраслям)

Составитель: преподаватель
Негребецкая В.И., Бобрышева В.В.

Курск 2016

Оглавление

Практическая работа № 1.
Тема: Сбор и анализ информации для определения потребностей клиента при разработке ПО

Цель работы – получения навыков сбора и анализа требований заказчиков и пользователей к автоматизированным системам на основе методики управления требованиями.

Теоретическая часть

Управление требованиями. Один из наиболее важных элементов при разработке программного обеспечения – управление требованиями (Requirements Management, RM). Это систематический подход к сбору, организации, документированию и отслеживанию требований системы. Надлежащее управление требованиями помогает проверять систему, управлять изменениями и анализировать статус проекта. Намного дешевле исправлять проблемы в течение процесса анализа требований, чем на стадии проектирования, тестирования или выпуска релиза.

Использование инструмента для управления требованиями поможет организовать процесс, а также способствовать созданию и настройке требований. Один из наиболее популярных инструментов, например, – IBM Rational RequisitePro.

Требования к ПО. Требование определяется как «условие или особенность, которой должна удовлетворять система». Требованием может быть:

- Функциональность, необходимая заказчику или пользователю для разрешения проблем (или получения прибыли).
- Функциональность, которая должна быть реализована в системе в соответствии с контрактом, стандартом, спецификацией, инструкцией или другим официальным документом.
- Ограничение, наложенное заинтересованным лицом.

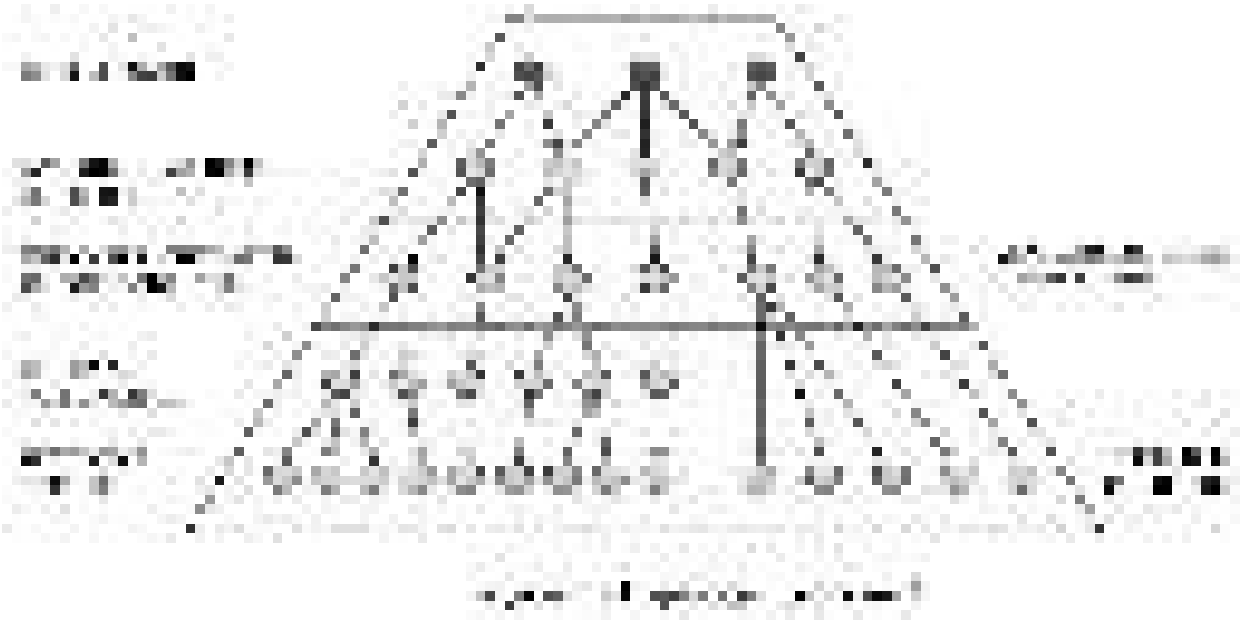
Заинтересованные лица. Обычно под заинтересованным лицом (stakeholder) понимают личность, на которую оказывает влияние разрабатываемая система. Два главных типа заинтересованных лиц – это пользователи и заказчики. Пользователи – это лица, которые будут пользоваться системой. Заказчики –

лица, кто заказывает систему и отвечает в дальнейшем за приемку системы.

В качестве заинтересованного лица можно рассматривать:

- Любого, участвующего в разработке системы (бизнес-аналитики, дизайнеры, кодировщики, тестеры, менеджеры проектов, менеджеры по внедрению, дизайнеры сценариев использования, графические дизайнеры).
- Любого, кто привносит свои знания в систему (эксперты предметной области, авторы документов, которые были использованы для сбора требований, собственники вебсайтов, ссылки на которые были предоставлены).
- Руководство (президент компании, которого представляют заказчики, директор отдела информационных технологий компании, проектирующей и разрабатывающей систему).
- Лица, вовлеченные в управление, настройку и сопровождение системы (хостинговая компания, справочная служба).
- Поставщики стандартов и регламентов (стандарты устанавливаются поисковыми механизмами согласно содержанию веб-сайта, политическим нормам, а также порядку налогообложения).

Пирамида требований. В зависимости от формата, источника и общих характеристик, требования могут быть разделены на различные типы. Эти типы требований могут быть представлены в виде пирамиды, как показано на рисунке 1.



«Хорошее» требование. Требование должно удовлетворять нескольким критериям для того, чтобы считаться «хорошим требованием». Хорошее требование должно иметь следующие характеристики.

Недвусмысленность: должен существовать только один путь интерпретации требования. Иногда двусмысленность проявляется в виде неопределенных акронимов.

Тестируемость (Возможность Проверки): тестеры должны иметь возможность проверить, было ли требование реализовано корректно. Тестирование должно быть либо пройдено, либо нет. Чтобы быть пригодным для тестирования, требование должно быть ясным,точным и недвусмысленным.

Ясность (Краткость, Сжатость, Простота, Точность): требования не должны содержать ненужных выражений или информации. Они должны быть изложены кратко и просто.

Корректность: если требование содержит факты, эти факты должны быть достоверны. Понятность: требования должны быть грамматически правильные, написаны в соответствующем стиле. Должны быть использованы стандартные соглашения. Слово «должен» должно быть использовано вместо «будет», «обязан» или «может».

Правдоподобность (Реальность, Выполнимость): требование должно быть выполнимо в рамках существующих ограничений, таких как время, деньги и доступные ресурсы.

Независимость: чтобы понять требование, не нужно знать какое-либо другое требование.

Элементарность: требование должно содержать отдельный трассируемый элемент (для которого возможно отслеживание связи).

Необходимость: в требовании нет необходимости, если ни одному заинтересованному лицу требование не нужно или удаление требования не повлияет на систему.

Независимость от Реализации (Абстрактность): требования не должны содержать ненужной информации о дизайне и реализации системы.

Постоянство: не должно существовать конфликтов между требованиями. Конфликты могут быть прямые и косвенные.

Немногословность: каждое требование должно быть обозначено только один раз, и не должно перекрываться другим требованием.

Завершенность: требование должно быть описано для всех возможных условий.

Этапы управления требованиями. Самое простое описание процесса управления

требованиями содержит следующие основные пункты:

- Формирование плана управления требованиями.
- Сбор требований.
- Разработка документа Концепции (Vision).
- Создание сценариев использования (Use Cases).
- Дополнительная спецификация.
- Создание тестовых сценариев (Test Cases) из сценариев использования (Use Cases).
- Создание тестовых сценариев (Test Cases) из дополнительной спецификации.
- Проектирование системы.

Сбор требований. В каждой группе заинтересованных лиц выделяется, по крайней мере, один представитель, который будет отвечать за поступление требований. Этот человек должен быть уполномочен представлять группу, обладать соответствующими знаниями и быть доступным для связи с группой аналитиков.

Запросы заинтересованных лиц могут быть собраны с использованием различных методов:

- Интервью (индивидуальный диалог с заинтересованным лицом).
- Анкеты (набор вопросов, отправленный большому количеству заинтересованных лиц).
- Семинары (заинтересованные лица собираются на определенный период

для интенсивных, насыщенных дискуссий).

- Сценарии приложения (использование визуальных/графических инструментов для демонстрирования поведения системы).
- Ролевые игры (каждому члену группы назначается определенная роль, обычно роль одного из пользователей).
- Сессии с применением метода «мозгового штурма» (Brainstorming sessions групповой метод решения вопросов путем изложения идей в течение непродолжительных, интенсивных сессий).
- Использование прототипов (разработка прототипов для получения отклика о системе).
- Сценарии использования (Use Cases – взаимодействие между пользователем и системой, представленное в виде последовательности шагов).
- Анализ существующих документов (извлечение информации из документов Microsoft Word, электронной почты и записей).
- Наблюдение и демонстрирование задач (наблюдение за пользователями, выполняющими определенную задачу).
- Анализ существующих систем (сбор требований от морально устаревших заменяемых систем или от систем, разработанных в ходе конкуренции).

Документ Запросов Заинтересованного лица. Вся полученная в процессе сбора тре бований информация должна быть документально оформлена в документ Запросов Заинтересованного Лица (Stakeholder Requests).

Можно создать один документ, который будет содержать в себе запросы всех заинтересованных лиц, один документ для каждого заинтересованного лица, либо можно собрать запросы нескольких заинтересованных лиц в одном документе.

Документ Запросов Заинтересованного Лица не содержит специально выделенного места для размещения требований типа STRQ (Stakeholder Requests или Запросы Заинтересованного лица). Требования могут быть вложены в качестве ответов на вопросы интервью, либо собраны в последнем параграфе, названным «Резюме Аналитика».

Главными атрибутами требования являются «Текст» и «Название». Если «Текст» – короткий (описание состоит из одного предложения), то не надо создавать отдельное «Название». Для большей точности лучше использовать название для всех требований одного типа, либо не использовать название ни для одного из них.

Представления. Views (или представления) используются для отображения требований, а также для управления требованиями, их атрибутами и их взаимоотношениями с другими требованиями. Вы можете сортировать и фильтровать требования, чтобы получить необходимый отчет.

Описание задачи

Задание выдается для двух студентов. Каждому варианту заданий соответствует задача на разработку программного обеспечения. Необходимо собрать и обработать информацию о требованиях к разрабатываемой программе. Для этого нужно выполнить следующие задания.

1. Взять интервью у партнера, чтобы выяснить список требований. Интервью должно быть зафиксировано письменно в 2х экземплярах (студенты опрашивают друг друга по очереди).
2. Подготовить список возможных заинтересованных лиц.
3. Подготовить анкету для опроса заинтересованных лиц.
4. Систематизировать и оформить в виде запросов заинтересованных лиц собранную и обработанную информацию.

Варианты заданий.

1. Программа для смартфона, позволяющая создавать, редактировать и секретно применять по назначению шпаргалки.
2. Программа для смартфона для получения помощи друга на экзамене (студент должен иметь возможность отправить задачу друзьям и получить обратно решение)
3. Централизованная система помощи студентам на экзамене (веб портал, принимающий запросы на решения задач и т. п., и передающий их централизованной группе консультантов)

4. Программно-технический комплекс для определения качества покупаемого арбуза (Снятие цветовых характеристик, химических проб, отправка данных на центральный сервер для обработки и получение назад результата).

Практическая работа 2.

Тема: Разработка структурно-функциональных диаграмм на проектируемое ПО

Цель работы: Ознакомиться с методологией моделирования прецедентов на основе языка UML

Содержание работы и методические указания к ее выполнению

UML (Universal Modeling Language) - универсальный язык моделирования, который был разработан компанией Rational Software с целью создания наиболее оптимального и универсального языка для описания как предметной области, так и конкретной задачи в программировании. Визуальное моделирование в UML можно представить как некоторый процесс поуровневого спуска от наиболее обшей и абстрактной концептуальной модели системы к логической, а затем и к физической модели соответствующей системы. Любая задача, таким образом, моделируется при помощи некоторого набора иерархических диаграмм, каждая из которых представляет собой некоторую проекцию системы.

Диаграмма (Diagram) - это графическое представление множества элементов. Чаще всего она изображается в виде связного графа с вершинами (сущностями) и ребрами (отношениями).

В UML определено восемь видов диаграмм [1]:

- диаграмма прецедентов (Use case diagram) - диаграмма поведения, на которой показано множество прецедентов и актеров, а также отношения между ними;
- диаграмма деятельности (Activity diagram) - диаграмма поведения, на которой показан автомат и подчеркнуты переходы потока управления от одной деятельности к другой;
- диаграмма классов (Class diagram) - структурная диаграмма, на которой показано множество классов, интерфейсов, коопераций и отношения между ними;
- диаграмма состояний (Statechart diagram) - диаграмма поведения, на которой показан автомат и подчеркнуто поведение объектов с точки зрения порядка получения событий;

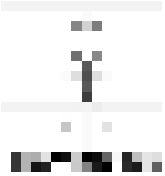
- диаграмма последовательностей (Sequence diagram) - диаграмма поведения, на которой показано взаимодействие и подчеркнута временная последовательность событий;
- диаграмма кооперации (Collaboration diagram) - диаграмма поведения, на которой показано взаимодействие и подчеркнута структурная организация объектов, посылающих и принимающих сообщения;
- диаграмма компонентов (Component diagram) - диаграмма поведения, на которой показан автомат и подчеркнуто поведение объектов с точки зрения порядка получения событий
- диаграмма развертывания (Deployment diagram) - структурная диаграмма, на которой показаны узлы и отношения между ними.

Диаграмма прецедентов

Диаграммы прецедентов применяются для моделирования вида системы с точки зрения внешнего наблюдателя. На диаграмме прецедентов графически показана совокупность прецедентов и Субъектов, а также отношения между ними.

Рассмотрим основные элементы диаграммы прецедентов.

Субъект (actor) - **любая сущность, взаимодействующая с системой извне [2] или множество логически связанных ролей, исполняемых при взаимодействии с прецедентами [1].** Стандартным графическим обозначением субъекта на диаграммах является фигурка "человечка", под которой записывается конкретное имя субъекта, однако субъектом может быть не только человек, но и техническое устройств о, программа или любая другая система, которая может служить источником воздействия на моделируемую систему так, как определит сам разработчик.



Прецеденты (use case) - **это описание множества последовательностей действий (включая их варианты), которые выполняются системой для**

того, чтобы актер получил результат, имеющий для него определенное значение. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие субъектов с системой, это одна из важнейших особенностей разработки прецедентов. Стандартным графическим обозначением прецедента на диаграммах является эллипс, внутри которого содержится краткое название прецедента или имя в форме глагола с пояснительными словами.

Сущность концепции прецедентов подразумевает несколько важных пунктов.

- Прецедент представляет собой **завершенный** фрагмент функциональных возможностей (включая основной поток логики управления, его любые вариации (подпотоки) и исключительные условия (альтернативные потоки)).
- Фрагмент **внешне наблюдаемых функций** (отличных от внутренних функций).
- **Ортогональный** фрагмент функциональных возможностей (прецеденты могут при выполнении совместно использовать объекты, но выполнение каждого прецедента независимо от других прецедентов).
- Фрагмент функциональных возможностей, **инициируемый субъектом.** Будучи инициирован, прецедент может взаимодействовать с другими субъектоми. При этом возможно, что субъект окажется только на принимающем конце прецедента, опосредованно инициированного другим субъектом.
- Фрагмент функциональных возможностей, который предоставляет субъекту **ощутимый полезный результа**т (и этот результат достигается в пределах одного прецедента).

Между субъектами и прецедентами - основными компонентами диаграммы прецедентов - могут существовать различные отношения, которые описывают взаимодействие экземпляров одних субъектов и прецедентов с экземплярами других субъектов и прецедентов. В языке UML имеется несколько стандартных видов отношений между субъектами и прецедентами:

Отношение ассоциации (association) - **определяет наличие канала связи между экземплярами субъекта и прецедента (или между экземплярами двумх субъектов).** Обозначается сплошной линией, возможно наличие стрелки и указание мощности связи.

Отношение расширения (extend) - **определяет взаимосвязь экземпляров отдельного прецедента с более общим прецедентом, свойства которого определяются на основе способа совместного объединения данных экземпляров.** Обозначается пунктирной линией со стрелкой, направленной от того прецедента, который является расширением для исходного прецедента, и помечается ключевым словом "extend" ("расширяет").

Отношение включения (include) - **указывает, что некоторое заданное поведение для одного прецедента включает в качестве составного компонента поведение другого прецедента.** Данное отношение является направленным бинарным отношением в том смысле, что пара экземпляров прецедентов всегда упорядочена в отношении включения. Обозначается пунктирной линией со стрелкой, направленной от базового прецедента к включаемому, и помечается ключевым словом "include" ("включает").

Отношение обобщения (generalization) - **служит для указания того факта, что некоторый прецедент А может быть обобщен до прецедента В. В этом случае прецедент А будет являться специализацией прецедента В.** При этом В называется предком или родителем по отношению к А, а прецедент А - потомком по отношению к прецеденту В. Следует подчеркнуть, что потомок наследует все свойства и поведение своего родителя, а также может быть дополнен новыми свойствами и особенностями поведения. Графически данное отношение обозначается сплошной линией со стрелкой в форме

незакрашенного треугольника, которая указывает на родительский прецедент.

Пример. Магазин видеопродукции.

Магазин продает видеокассеты, DVD-диски, аудио-кассеты, CD-диски и т.д. , а также предлагает широкой публике прокат видеокассет и DVD-дисков.

Товары поставляются несколькими поставщиками. Каждая партия товара предварительно заказывается магазином у некоторого поставщика и доставляется после оплаты счета. Вновь поступивший товар маркируется, заносится в базу данных и затем распределяется в торговый зал или прокат.

Видеоносители выдаются в прокат на срок от 1 до 7 дней. При прокате с клиента взимается залоговая стоимость видеоносителя. При возврате видеоносителя возвращается залоговая стоимость минус сумма за прокат. Если возврат задержан менее чем на 2 дня, взимается штраф в размере суммы за прокат за 1 день* кол-во дней задержки. При задержке возврата более чем на 2 дня - залоговая сумма не возвращается. Клиент может взять одновременно до 4 видеоносителей (прокат-заказ). На каждый видеоноситель оформляется квитанция.

Клиенты могут стать членами видео-клуба и получить пластиковые карточки. С членов клуба не берется залог (за исключением случая описанного ниже), устанавливается скидка на ставку проката и покупку товаров. Члены клуба могут делать предварительные заказы на подбор видеоматериалов для проката или покупки.

Каждый член клуба имеет некоторый стутус. Первоначально - "новичок". При возврате всрок 5 прокат-заказов , статус меняется на "надежный". При задержке хотя бы одного видеоносителя более чем на 2 дня , статус "новичок" или "надежный" меняется на "ненадежный" и клиенту высылается предупреждение. При повторном нарушении правил статус меняется на "нарушитель". Члены клуба со статусом "надежный" могут брать до 8 видеоносителей единовременно, все остальные - 4. С членов клуба со статусом "нарушитель" берется залоговая сумма.

Клиенты при покупке товара или получении видеоносителя в прокат

могут расплачиваться наличными или кредитной картой.

Прокатные видеоносители через определенное количество дней проката списываются и утилизируются по акту. Списываются также товары и прокатные видеоносители, у которых обнаружился брак.

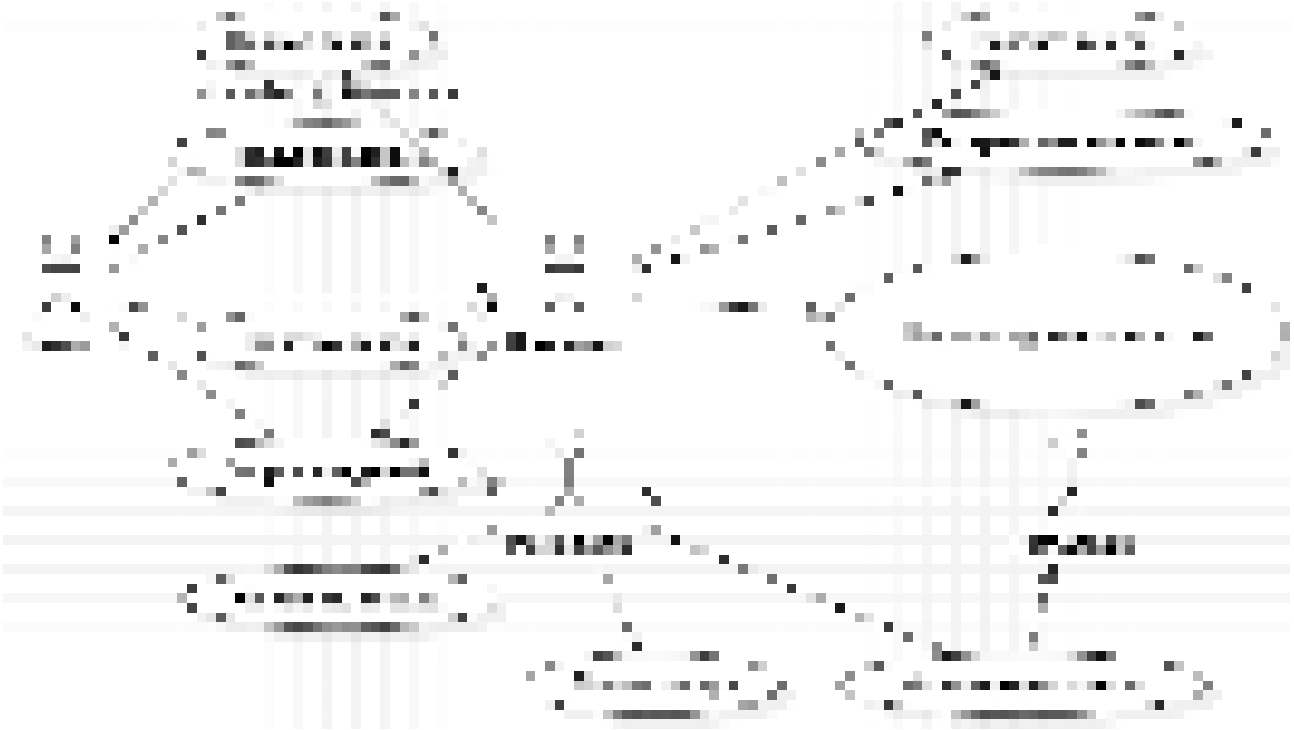
На рисунке ниже приведена диаграмма прецедентов для рассматриваемого примера. В этом примере можно выделить следующие субъекты и соответствующие им прецеденты:

- **Менеджер** - изучает рынок видеопродукции, анализирует продажи (прецедент "Запрос сведений"), работает с поставщиками: составляет заявки на поставки товара (прецедент "Оформление заказа"), оплачивает и принимает товар (прецедент "Прием товара"), списывает товар (прецедент "Списание товара").
- **Продавец** - работает с клиентами: продает товар (прецедент "Продажа видео"), оформляет членство в клубе (прецедент "Сопровождение клиентов"), резервирует (прецедент "Резервирование видио"), выдает в прокат (прецедент "Прокат видео") и принимает назад видеоносители (прецедент "Возврат видео"), отвечает на вопросы клиента (прецедент "Запрос сведений").
- **Поставщик** - оформляет документы для оплаты товара (прецедент "Оформление заказа"), поставляет товар (прецедент "Прием товара"))
- **Клиент** - покупает(прецедент "Продажа видео"), берет на прокат и возвращает видеоносители (прецеденты "Прокат видео" и "Возврат видео"), вступает в клуб (прецедент "Сопровождение клиентов"), задает вопросы (прецедент "Запрос сведений").

Последние два субъекта **Поставщик и Клиент** не будут иметь непосредственного доступа к разрабатываемой системе (второстепенные субъекты), однако именно они являются основным источником событий, инициализирующих прецеденты, и получателями результата работы прецедентов

От прецедента "Прокат видео" к прецеденту "Предупреждения" установлено отношение включения на том основании, что каждый выданный

видеоноситель должен быть проверен на своевременный возврат и, в случае необходимости, выдано предупреждение клиенту.



Дальнейшее развитие модели поведения системы предполагает спецификацию прецедентов. Для этого традиционно используют два способа. Первый - описание с помощью текстового документа. Такой документ описывает, что должна делать система, когда субъект инициировал прецедент. Типичное описание содержит следующие разделы:

- Краткое описание
- Участвующие субъекты
- Предусловия, необходимые для инициирования прецедента
- Поток событий (основной и, возможно, подпотоки, альтернативный
- Постусловия, определяющие состояние системы, по достижении которого прецедент завершается.

Описательная спецификация прецедента "Прокат видео"

Раздел	Описание
Краткое описание	Клиент желает взять на прокат видеокассету или диск, которые снимаются с полки магазина или были предварительно зарезервированы клиентом. При условии, что у клиента нет невозвращенных в срок видеоносителей, сразу после внесения платы фильм

	выдается напрокат. Если невозвращенные в срок видеоносители есть, клиенту выдается напоминание о просроченном возврате.
Субъекты	Продавец, Клиент.
Предусловия	В наличие имеются видеокассеты или диски, которые можно взять напрокат. У клиентов есть клубные карточки. Устройство сканирования работает правильно. Работники за прилавком знают, как обращаться с системой.
Основной поток	Клиент может назвать номер заказа или взять видеоноситель с полки. Видеоноситель и членская карточка сканируются и продавцу не сообщается никаких сведений о задержках, так, что он не задает клиенту соответствующих вопросов. Если клиент имеет статус <надежный>, он может взять до 8 видеоносителей, во всех остальных случаях - до 4-х. Если статус клиента определен как <нарушитель>, его просят внести задаток. Клиент расплачивается наличными или кредитной картой. После получения суммы, информация о наличии фильмов обновляется и видеоносители передаются клиенту вместе с квитанциями на прокат. О прокате каждого видеоносителя делается отдельная запись с указанием идентификационного номера клиента, даты проката, даты возврата, идентификационного номера продавца, полученной суммы. Прецедент генерирует предупреждения о просроченном возврате клиенту, если видеофильм не был возвращен в течение двух дней по истечении даты возврата и изменяет статус клиента на <ненадежный> (первое нарушение) или <нарушитель> (повторное нарушение).
Альтернативный поток	У клиента нет членской карточки. В этом случае прецедент <Сопровождение клиента> может быть активизирован для выдачи новой карточки. Видеофильмы не выдаются, поскольку у клиента есть невозвращенные в срок видеоносители. Попытка взять на прокат слишком много видеоносителей. Видеоноситель или кредитная карта не могут быть отсканированы из-за их повреждения У клиента не хватило наличных или платеж по кредитной карте отклонен.
Постусловия	Видеофильмы сданы напрокат, и база данных соответствующим образом обновлена

Другой способ - построение диаграмм деятельности - рассматривается в лабораторной работе 5.

Выполнение лабораторной работы производится с использованием программного продукта StarUML

Последовательность выполнения практической работы:

1. Ознакомиться с методологией моделирования прецедентовна основе языка UML.

2. Построить диаграмму прецедентов для своей предметной области (Приложение 1).

3. Описать несколько (2-3) прецедентов.

4. Оформить отчет.

Приложение 1.

Предметные области

1. БД «Кадры»
2. БД «Документы»
3. БД «Библиотека»
4. БД «Лесничество»
5. БД «География»
6. БД «Банки»
7. БД «Поликлиника»
8. БД «Юриспруденция»
9. БД «Снабжение»
10. БД «Сбыт»
11. БД «Производство – Детали»
12. БД «ГИБДД».

13. БД «Налоги»
14. БД «Земельный кадастр»
15. БД «Расписание»
16. БД «Учебный план»
17. БД «Учебный процесс»
18. БД «Магазин»
19. БД «Оптовый склад»
20. БД «Социальная работа»
21. БД «Аптека»
22. БД «Пенсионный фонд»
23. БД «Обеспеченность учебного процесса литературой»
24. БД «Факультет»
25. БД «Кафедра»
26. БД «Учебные планы»
27. БД «Картинная галерея»
28. БД «Музей»
29. БД «Продажа и регистрация авиабилетов»
30. БД «Фонотека».
31. БД «Видеотека»
32. БД «Футбольные клубы»
33. БД «Кинозвезды»
34. БД «Баскетбольные клубы»
35. БД «Хоккеисты»

Практическая работа №3.

Тема: Оформление технического задания на разработку модели системы дистанционного обучения с применением технологии «клиент-сервер»

Как писать техническое задание на программный продукт или что значит фраза "по форме ГОСТ 19.201-78"

Рассмотрим, как правильно составить техническое задание на разработку программного продукта.

1. ОБЩИЕ ТРЕБОВАНИЯ

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

1.2 Лист утверждения и титульный лист оформляют в соответствии с ГОСТ 19.104-78. Информационную часть (аннотацию и содержание), лист регистрации изменений допускается в документ не включать.

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

1.4. Техническое задание должно содержать следующие разделы:

- введение;
- основания для разработки;
- назначение разработки;
- требования к программе или программному изделию;
- требования к программной документации;
- технико-экономические показатели;
- стадии и этапы разработки;
- порядок контроля и приемки;

в техническое задание допускается включать приложения.

В зависимости от особенностей программы или программного изделия допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них.

2. СОДЕРЖАНИЕ РАЗДЕЛОВ

2.1. В разделе "Введение" указывают наименование, краткую характеристику области применения программы или программного изделия и объекта, в котором используют программу или программное изделие.

2.2. В разделе "Основания для разработки" должны быть указаны:

- документ (документы), на основании которых ведется разработка;
- организация, утвердившая этот документ, и дата его утверждения;
- наименование и (или) условное обозначение темы разработки.

2.3. В разделе "Назначение разработки" должно быть указано функциональное и эксплуатационное назначение программы или программного изделия.

2.4. Раздел "Требования к программе или программному изделию" должен содержать следующие подразделы:

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

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

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

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

2.4.4. В подразделе "Требования к составу и параметрам технических средств" указывают необходимый состав технических средств с указанием их основных технических характеристик.

2.4.5. В подразделе "Требования к информационной и программной совместимости" должны быть указаны требования к информационным структурам на входе и выходе и методам решения, исходным кодам, языкам программирования я программным средствам, используемым программой.

При необходимости должна обеспечиваться защита информации и программ.

2.4.6. В подразделе "Требования к маркировке и упаковке" в общем случае указывают требования к маркировке программного изделия, варианты и способы упаковки.

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

2.5а. В разделе "Требования к программной документации" должен быть указан предварительный состав программной документации и, при необходимости, специальные требования к ней.

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

2.6. В разделе "Стадии и этапы разработки" устанавливают необходимые стадии разработки, этапы и содержание работ (перечень программных документов, которые должны быть разработаны, согласованы и утверждены), а

также, как правило, сроки разработки я определяют исполнителей.

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

2.8. В приложениях к техническому заданию, при необходимости, приводят:

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

Пример заполненного Технического задания.

Техническое задание на разработку модели системы дистанционного обучения с применением технологии «клиент-сервер».

1. Введение

Разработать модель системы дистанционного обучения «» с использованием клиент-серверной технологии. Модель предполагает дальнейшее развитие в программный комплекс, предназначенный для заочных и дистантных форм обучения высших и средних учебных заведений, учебных центров повышения квалификации и центров переподготовки сотрудников.

2. Основания для разработки

Основанием для разработки является учебный план кафедры ИУ6 на 11-й семестр, утвержденный заведующим кафедрой.

3. Назначение разработки

Модель является первым этапом реализации сложного комплекса системы дистанционного обучения, предназначенного для внедрения и использования в учебных заведениях. Назначение системы – реализовать новый подход к обучению, позволяющий людям с периферии иметь возможность изучить учебные программы, подготовленные в крупных ВУЗах страны, а также позволяющий получать образование или повышать квалификацию дома или на рабочем месте без отрыва от производства.

4. Требования к программе или программному изделию.

4.1 Требования к функциональным характеристикам.

Разрабатываемая модель должна обладать следующими функциями:

- Работать под управлением ОС Windows 95/98 или Windows NT/2000.
- Использовать для соединения и обмена данными протокол TCP/IP.
- Использовать свой протокол, как надстройку над TCP/IP для передачи данных и команд.
- Иметь доступный и простой интерфейс пользователя.
- Иметь гибкую систему настроек.
- Серверная часть должна хранить базу данных пользователей, имеющих доступ к системе и обеспечивать аутентификацию пользователей согласно имеющихся записей.
- Серверная часть должна хранить базу данных учебных курсов, доступных для изучения пользователями.
- Серверная часть должна поддерживать соединение до 32000 пользователей одновременно.
- Клиентская часть должна хранить базу данных адресов серверов для подключения.

4.2 Требования к надежности.

Надежность системы в целом зависит от надежности используемой операционной системы. Серверная часть должна обслуживать без сбоев одновременное подключение и работу до 32000 пользователей. Обе части должны без потерь передавать информацию по каналу связи между клиентом и сервером.

4.3 Условия эксплуатации.

Стандартные условия эксплуатации программных продуктов. Необходимые сотрудники для обслуживания серверной части системы – системный администратор для обслуживания собственно сервера (регистрация и удале-

ние пользователей, добавление и настройка учебных материалов) и группа разработчиков учебных курсов, численность и состав которой зависит от конкретной дисциплины курса.

4.4 Требования к составу и параметрам технических средств.

Для нормальной работы как серверной, так и клиентской частей необходимо:

- Компьютер с процессором Intel Pentium-100 или 100%- совместимым.
- Оперативная память не менее 16 Mb.
- Жесткий диск объемом не менее 1 Gb.
- Наличие адаптера подключения к сети (сетевой карты, модема и т.п.).
- Установленная ОС Windows 95/98/NT/2000.
- Настроенный протокол TCP/IP.

4.5 Требования к информационной и программной совместимости.

Модель системы должна работать под управлением ОС Windows 95/98/NT/2000, поэтому требуется совместимость исполняемого модуля и библиотек динамического подключения стандартам, используемым этими ОС на платформе IBM PC. Модель должна использовать свой протокол передачи данных высокого уровня как надстройку над TCP/IP. Для хранения информации требуется использование баз данных формата MDB (Microsoft Access).

Для доступа к базам данных Microsoft Access 97 требуется наличие установленного ядра работы с БД Microsoft JET DAO версии 3.5. В качестве средства разработки требуется использовать интегрированную среду разработки Borland Delphi 5, включающую редактор исходных текстов, компилятор, компоновщик и отладчик. В качестве средства проектирования структуры базы данных и создания файла базы данных требуется использовать Microsoft Access 97.

4.6 Требования к маркировке и упаковке.

Не предъявляются.

4.7 Требования к транспортированию и хранению.

Не предъявляются.

4.8 Специальные требования.

Не предъявляются.

5. Требования к программной документации.

Программной документацией к разрабатываемой модели системы дистанционного обучения является рассчетно-пояснительная записка.

6. Стадии и этапы разработки.

№	Содержание работы	Срок	Исполнитель этапа разработки
1	Исследование концепций дистанционного обучения и имеющихся на сегодняшний день решений.	1-2 недели	Цыганов П.В., Кузнецов Д.Д.
2	Выработка своего решения	3-я неделя	Цыганов П.В., Кузнецов Д.Д.
3	Выработка технического задания	4-я неделя	Цыганов П.В., Кузнецов Д.Д.
4	Разработка протокола прикладного уровня “DECSS Protocol” для передачи команд и данных между клиентом и сервером. Создание библиотеки классов, реализующей разработанный протокол.	5-7 недели	Цыганов П.В.
5	Принятие решения по разработке формата файлов для хранения учебных курсов. Разработка библиотеки классов для поддержки принятого формата.	5-7 недели	Кузнецов Д.Д.
6	На основе разработанного протокола создание «скелета» серверной и клиентской части модели.	8-10 недели	Цыганов П.В.|
7	На основе созданной библиотеки классов для работы с файлом учебного курса создание средств просмотра	8-10 недели	Кузнецов Д.Д.

	курса.		
8	Объединение разработанных частей в единую модель.	11 неделя	Цыганов П.В., Кузнецов Д.Д.
9	Сдача и защита курсового проекта.	12 неделя	Цыганов П.В., Кузнецов Д.Д.

7. Порядок контроля и приемки.

Испытание представленной модели и контроль качества ее работы провести на базе компьютерного класса кафедры ИУ6. Во время испытаний проверить работу системы по следующим позициям:

- Запуск серверной и клиентской частей.
- Соединение клиента (-ов) с сервером, проверка правильности обработки сервером соединения.
- Аутентификация пользователя на сервере. Проверка изменения состава зарегистрированных пользователей и групп.
- Подключение на сервере учебного курса с тем, чтобы он был доступен для просмотра.
- Просмотр учебного курса с клиентского рабочего места.
- Завершение сеанса связи.

Практическая работа №4.

Тема: Составление технического задания на программный продукт

Цель:

- знакомство со стандартами **ЕСПД** ;
- умение пользоваться стандартами ЕСПД .

Задание. Используя ГОСТ 19.201-78 и техническое задание на разработку модели системы дистанционного обучения с применением технологии «клиент-сервер» составить техническое задание по выбранному варианту из практической работы №1.

Как писать техническое задание на программный продукт или что значит фраза "по форме ГОСТ 19.201-78"

Рассмотрим, как правильно составить техническое задание на разработку программного продукта.

1. ОБЩИЕ ТРЕБОВАНИЯ

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

1.2 Лист утверждения и титульный лист оформляют в соответствии с ГОСТ 19.104-78. Информационную часть (аннотацию и содержание), лист регистрации изменений допускается в документ не включать.

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

1.4. Техническое задание должно содержать следующие разделы:

- введение;
- основания для разработки;
- назначение разработки;

- требования к программе или программному изделию;
- требования к программной документации;
- технико-экономические показатели;
- стадии и этапы разработки;
- порядок контроля и приемки;

в техническое задание допускается включать приложения.

В зависимости от особенностей программы или программного изделия допускается уточнять содержание разделов, вводить новые разделы или объединять отдельные из них.

2. СОДЕРЖАНИЕ РАЗДЕЛОВ

2.1. В разделе "Введение" указывают наименование, краткую характеристику области применения программы или программного изделия и объекта, в котором используют программу или программное изделие.

2.2. В разделе "Основания для разработки" должны быть указаны:
- документ (документы), на основании которых ведется разработка;
- организация, утвердившая этот документ, и дата его утверждения;
- наименование и (или) условное обозначение темы разработки.

2.3. В разделе "Назначение разработки" должно быть указано функциональное и эксплуатационное назначение программы или программного изделия.

2.4. Раздел "Требования к программе или программному изделию" должен содержать следующие подразделы:
- требования к функциональным характеристикам;
- требования к надежности;
- условия эксплуатации;
- требования к составу и параметрам технических средств;
- требования к информационной и программной совместимости;
- требования к маркировке и упаковке;
- требования к транспортированию и хранению;
- специальные требования.

2.4.1. В подразделе "Требования к функциональным характеристикам" долж-

ны быть указаны требования к составу выполняемых функций, организации входных и выходных данных, временным характеристикам и т.п.

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

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

2.4.4. В подразделе "Требования к составу и параметрам технических средств" указывают необходимый состав технических средств с указанием их основных технических характеристик.

2.4.5. В подразделе "Требования к информационной и программной совместимости" должны быть указаны требования к информационным структурам на входе и выходе и методам решения, исходным кодам, языкам программирования я программным средствам, используемым программой.

При необходимости должна обеспечиваться защита информации и программ.

2.4.6. В подразделе "Требования к маркировке и упаковке" в общем случае указывают требования к маркировке программного изделия, варианты и способы упаковки.

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

2.5а. В разделе "Требования к программной документации" должен быть указан предварительный состав программной документации и, при необходимости, специальные требования к ней.

2.5. В разделе "Технико-экономические показатели" должны быть указаны:

ориентировочная экономическая эффективность, предполагаемая годовая потребность, экономические преимущества разработки по сравнению с лучшими отечественными и зарубежными образцами или аналогами.

2.6. В разделе "Стадии и этапы разработки" устанавливают необходимые стадии разработки, этапы и содержание работ (перечень программных документов, которые должны быть разработаны, согласованы и утверждены), а также, как правило, сроки разработки я определяют исполнителей.

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

2.8. В приложениях к техническому заданию, при необходимости, приводят:

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

Пример заполненного Технического задания.

Техническое задание на разработку модели системы дистанционного обучения с применением технологии «клиент-сервер».

1. Введение

Разработать модель системы дистанционного обучения «» с использованием клиент-серверной технологии. Модель предполагает дальнейшее развитие в программный комплекс, предназначенный для заочных и дистантных форм обучения высших и средних учебных заведений, учебных центров повышения квалификации и центров переподготовки сотрудников.

2. Основания для разработки

Основанием для разработки является учебный план кафедры ИУ6 на 11-й семестр, утвержденный заведующим кафедрой.

3. Назначение разработки

Модель является первым этапом реализации сложного комплекса системы дистанционного обучения, предназначенного для внедрения и использования

в учебных заведениях. Назначение системы – реализовать новый подход к обучению, позволяющий людям с периферии иметь возможность изучить учебные программы, подготовленные в крупных ВУЗах страны, а также позволяющий получать образование или повышать квалификацию дома или на рабочем месте без отрыва от производства.

4. Требования к программе или программному изделию.

4.1 Требования к функциональным характеристикам.

Разрабатываемая модель должна обладать следующими функциями:

- Работать под управлением ОС Windows 95/98 или Windows NT/2000.
- Использовать для соединения и обмена данными протокол TCP/IP.
- Использовать свой протокол, как надстройку над TCP/IP для передачи данных и команд.
- Иметь доступный и простой интерфейс пользователя.
- Иметь гибкую систему настроек.
- Серверная часть должна хранить базу данных пользователей, имеющих доступ к системе и обеспечивать аутентификацию пользователей согласно имеющихся записей.
- Серверная часть должна хранить базу данных учебных курсов, доступных для изучения пользователями.
- Серверная часть должна поддерживать соединение до 32000 пользователей одновременно.
- Клиентская часть должна хранить базу данных адресов серверов для подключения.

4.2 Требования к надежности.

Надежность системы в целом зависит от надежности используемой операционной системы. Серверная часть должна обслуживать без сбоев одновременное подключение и работу до 32000 пользователей. Обе части

должны без потерь передавать информацию по каналу связи между клиентом и сервером.

4.3 Условия эксплуатации.

Стандартные условия эксплуатации программных продуктов. Необходимые сотрудники для обслуживания серверной части системы – системный администратор для обслуживания собственно сервера (регистрация и удаление пользователей, добавление и настройка учебных материалов) и группа разработчиков учебных курсов, численность и состав которой зависит от конкретной дисциплины курса.

4.4 Требования к составу и параметрам технических средств.

Для нормальной работы как серверной, так и клиентской частей необходимо:

- Компьютер с процессором Intel Pentium-100 или 100%-совместимым.
- Оперативная память не менее 16 Mb.
- Жесткий диск объемом не менее 1 Gb.
- Наличие адаптера подключения к сети (сетевой карты, модема и т.п.).
- Установленная ОС Windows 95/98/NT/2000.
- Настроенный протокол TCP/IP.

4.5 Требования к информационной и программной совместимости.

Модель системы должна работать под управлением ОС Windows 95/98/NT/2000, поэтому требуется совместимость исполняемого модуля и библиотек динамического подключения стандартам, используемым этими ОС на платформе IBM PC. Модель должна использовать свой протокол передачи данных высокого уровня как надстройку над TCP/IP. Для хранения информации требуется использование баз данных формата MDB (Microsoft Access).

Для доступа к базам данных Microsoft Access 97 требуется наличие установ-

ленного ядра работы с БД Microsoft JET DAO версии 3.5. В качестве средства разработки требуется использовать интегрированную среду разработки Borland Delphi 5, включающую редактор исходных текстов, компилятор, компоновщик и отладчик.
В качестве средства проектирования структуры базы данных и создания файла базы данных требуется использовать Microsoft Access 97.
4.6 Требования к маркировке и упаковке.

Не предъявляются.

4.7 Требования к транспортированию и хранению.

Не предъявляются.

4.8 Специальные требования.

Не предъявляются.

5. Требования к программной документации.

Программной документацией к разрабатываемой модели системы дистанционного обучения является рассчетно-пояснительная записка.

6. Стадии и этапы разработки.

№	Содержание работы	Срок	Исполнитель этапа разработки
1	Исследование концепций дистанционного обучения и имеющихся на сегодняшний день решений.	1-2 недели	Цыганов П.В., Кузнецов Д.Д.
2	Выработка своего решения	3-я неделя	Цыганов П.В., Кузнецов Д.Д.
3	Выработка технического задания	4-я неделя	Цыганов П.В., Кузнецов Д.Д.
4	Разработка протокола прикладного уровня “DECSS Protocol” для передачи команд и данных между клиентом и сервером. Создание библиотеки классов, реализующей разработанный протокол.	5-7 недели	Цыганов П.В.
5	Принятие решения по разработке формата файлов для хранения учебных кур-	5-7 недели	Кузнецов Д.Д.

	сов. Разработка библиотеки классов для поддержки принятого формата.		
6	На основе разработанного протокола создание «скелета» серверной и клиентской части модели.	8-10 недели	Цыганов П.В.|
7	На основе созданной библиотеки классов для работы с файлом учебного курса создание средств просмотра курса.	8-10 недели	Кузнецов Д.Д.
8	Объединение разработанных частей в единую модель.	11 неделя	Цыганов П.В., Кузнецов Д.Д.
9	Сдача и защита курсового проекта.	12 неделя	Цыганов П.В., Кузнецов Д.Д.

7. Порядок контроля и приемки.

Испытание представленной модели и контроль качества ее работы провести на базе компьютерного класса кафедры ИУ6. Во время испытаний проверить работу системы по следующим позициям:

- Запуск серверной и клиентской частей.
- Соединение клиента (-ов) с сервером, проверка правильности обработки сервером соединения.
- Аутентификация пользователя на сервере. Проверка изменения состава зарегистрированных пользователей и групп.
- Подключение на сервере учебного курса с тем, чтобы он был доступен для просмотра.
- Просмотр учебного курса с клиентского рабочего места.
- Завершение сеанса связи.

Практическая работа №5.

Тема: Составление технической документации (руководство оператора, программиста, системного администратора)

Цель:

- знакомство со стандартами **ЕСПД** ;
- умение пользоваться стандартами ЕСПД .

Задание. Используя ГОСТ 19.505-79 Руководство оператора и ГОСТ 19. 504-79 Руководство программиста составить техническую документацию по выбранному варианту из практической работы №1.

РУКОВОДСТВО ОПЕРАТОРА.
ТРЕБОВАНИЯ К СОДЕРЖАНИЮ И ОФОРМЛЕНИЮ

United system for program documentation.

Operation's guide. Requirements to contents and form of presentation

Постановлением Государственного комитета стандартов Совета Министров СССР от 12 января 1979 г. ¹ 74 срок введения установлен

с 01.01. 1980 г.

Настоящий стандарт устанавливает требования к содержанию и оформлению программного документа «Руководство программиста», определённого ГОСТ 19.101-77.

Стандарт полностью соответствует СТ СЭВ 2095-80.

1. ОБЩИЕ ПОЛОЖЕНИЯ

1.1. Структуру и оформление документа устанавливают в соответствии с ГОСТ 19.105-78.

Составление информационной части (аннотации и содержания) является обязательным.

1.2. Руководство оператора должно содержать следующие разделы:

- назначение программы;
- условия выполнения программы;

- выполнение программы;
- сообщения оператору.

В зависимости от особенностей документы допускается объединять отдельные разделы или вводить новые.

(Измененная редакция, Изм. № 1).

2. СОДЕРЖАНИЕ РАЗДЕЛОВ

2.1. В разделе «Назначение программы» должны быть указаны сведения о назначении программы и информация, достаточная для понимания функций программы и ее эксплуатации.

2.2. В разделе «Условия выполнения программы» должны быть указаны условия, необходимые для выполнения программы (минимальный и (или) максимальный состав аппаратурных и программных средств и т.п.).

(Измененная редакция, Изм. № 1).

2.3. В разделе «Выполнение программы» должна быть указана последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведено описание функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузки и управляет выполнением программы, а также ответы программы на эти команды.

(Измененная редакция, Изм. № 1).

2.4. В разделе «Сообщения оператору» должны быть приведены тексты сообщений, выдаваемых в ходе выполнения программы, описание их содержания и соответствующие действия оператора (действия оператора в случае сбоя, возможности повторного запуска программы и т.п.).

2.5. Допускается содержание разделов иллюстрировать поясняющими примерами, таблицами, схемами, графиками.

(Измененная редакция, Изм. № 1).

2.6. В приложения к руководству оператора допускается включать различные материалы, которые нецелесообразно включать в разделы руководства.

(Введен дополнительно, Изм. № 1).

Лист утверждения

- **Титульный лист**
- **Аннотация**
- **Содержание**
- **<u>Основная часть</u>**
- Назначение программы
- *Функциональное назначение программы*
- *Эксплуатационное назначение программы*
- *Состав функций*
- Функции (такой-то)
- Функции (этакой)
- Условия выполнения программы
- *Минимальный состав аппаратных средств*
- *Минимальный состав программных средств*
- *Требования к персоналу (пользователю)*
- Выполнение программы
- *Загрузка и запуск программы*
- *Выполнение программы*
- Выполнение функции (такой-то)
- Выполнение функции (этакой)
- *Завершение работы программы*
- Сообщения оператору
- *Сообщение (такое-то)*
- *Сообщение (этакое)*
- **Приложения** (необязательны)
- **Регистрация изменений**

РУКОВОДСТВО ПРОГРАММИСТА.
ТРЕБОВАНИЯ К СОДЕРЖАНИЮ И ОФОРМЛЕНИЮ

United system for program documentation.

Programmer's guide. Requirements to contents and form of presentation

Постановлением Государственного комитета стандартов Совета Министров СССР от 12 января 1979 г. ¹ 74 срок введения установлен

с 01.01. 1980 г.

Настоящий стандарт устанавливает требования к содержанию и оформлению программного документа «Руководство программиста», определённого ГОСТ 19.101-77.

Стандарт полностью соответствует СТ СЭВ 2095-80.

1. ОБЩИЕ ПОЛОЖЕНИЯ

1.1. Структуру и оформление документа устанавливают в соответствии с ГОСТ 19.105-78.

Составление информационной части (аннотации и содержания) является обязательным.

1.2. Руководство программиста должно содержать следующие разделы:

- назначение и условия применения программ;
- характеристика программы;
- обращение к программе;
- входные и выходные данные;
- сообщения.

В зависимости от особенностей документы допускается объединять отдельные разделы или вводить новые.

2. СОДЕРЖАНИЕ РАЗДЕЛОВ

2.1. В разделе «Назначение и условия применения программ» должны быть указаны назначение и функции, выполняемые программой, условия,

необходимые для выполнения программы (объем оперативной памяти, требования к составу и параметрам периферийных устройств, требования к программного обеспечению и т.п.).

2.2. В разделе «Характеристика программы» должно быть приведено описание основных характеристик и особенностей программы (временные характеристики, режим работы, средства контроля правильности выполнения и самовосстанавливаемости программы и т.п.).

2.3. В разделе «Обращение к программе» должно быть приведено описание процедур вызова программы (способы передачи управления и параметров данных и др.).

2.4. В разделе «Входные и выходные данные» должно быть приведено описание организации используемой входной и выходной информации и, при необходимости, ее кодирования.

2.5. В разделе «Сообщения» должны быть указаны тексты сообщений, выдаваемых программисту или оператору в ходе выполнения программы, описание их содержания и действий, которые необходимо предпринять по этим сообщениям.

2.6. В приложении к руководству программиста могут быть приведены дополнительные материалы (примеры, иллюстрации, таблицы, графики и т.п.).

Рекомендуемая структура программного документа (по ГОСТ 19.504-79. ЕСПД)

- **Лист утверждения**
- **Титульный лист**
- **Аннотация**
- **Содержание**
- **<u>Основная часть</u>**
- Назначение и условия применения программ
- *Назначение программы*
- *Функции, выполняемые программой*

- *Условия, необходимые для выполнения программы*
- Объем оперативной памяти
- Требования к составу периферийных устройств
- Требования к параметрам периферийных устройств
- Требования к программного обеспечению
- Требования к персоналу (программисту)
- Характеристика программы
- *Описание основных характеристик программы*
- Временные характеристики программы
- Режим работы программы
- Средства контроля правильности выполнения программы
- *Описание основных особенностей программы*
- Самовосстанавливаемость программы
- Обращение к программе
- *Загрузка и запуск программы*
- *Выполнение программы*
- Выполнение функции (такой-то)
- Выполнение функции (этакой)
- *Завершение работы программы*
- Входные и выходные данные
- *Организации используемой входной информации*
- *Организации используемой выходной информации*
- Сообщения
- Сообщение (такое-то)
- Сообщение (этакое)
- **Приложения** (необязательны)
- **Регистрация изменений**

Практическая работа №6.

Тема: Тестирование технической документации.

Цель:

- знакомство со стандартами **ЕСПД** ;
- умение пользоваться стандартами ЕСПД .

Задание. Используя ГОСТ 19.505-79 Руководство оператора, ГОСТ 19. 504-79 Руководство программиста и ГОСТ 19.201-78 протестировать техническую докумнетацию.

РУКОВОДСТВО ОПЕРАТОРА.
ТРЕБОВАНИЯ К СОДЕРЖАНИЮ И ОФОРМЛЕНИЮ

United system for program documentation.

Operation's guide. Requirements to contents and form of presentation

Постановлением Государственного комитета стандартов Совета Министров СССР от 12 января 1979 г. ¹ 74 срок введения установлен

с 01.01. 1980 г.

Настоящий стандарт устанавливает требования к содержанию и оформлению программного документа «Руководство программиста», определённого ГОСТ 19.101-77.

Стандарт полностью соответствует СТ СЭВ 2095-80.

1. ОБЩИЕ ПОЛОЖЕНИЯ

1.1. Структуру и оформление документа устанавливают в соответствии с ГОСТ 19.105-78.

Составление информационной части (аннотации и содержания) является обязательным.

1.2. Руководство оператора должно содержать следующие разделы:

- назначение программы;
- условия выполнения программы;
- выполнение программы;

- сообщения оператору.

В зависимости от особенностей документы допускается объединять отдельные разделы или вводить новые.

(Измененная редакция, Изм. № 1).

2. СОДЕРЖАНИЕ РАЗДЕЛОВ

2.1. В разделе «Назначение программы» должны быть указаны сведения о назначении программы и информация, достаточная для понимания функций программы и ее эксплуатации.

2.2. В разделе «Условия выполнения программы» должны быть указаны условия, необходимые для выполнения программы (минимальный и (или) максимальный состав аппаратурных и программных средств и т.п.).

(Измененная редакция, Изм. № 1).

2.3. В разделе «Выполнение программы» должна быть указана последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведено описание функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузки и управляет выполнением программы, а также ответы программы на эти команды.

(Измененная редакция, Изм. № 1).

2.4. В разделе «Сообщения оператору» должны быть приведены тексты сообщений, выдаваемых в ходе выполнения программы, описание их содержания и соответствующие действия оператора (действия оператора в случае сбоя, возможности повторного запуска программы и т.п.).

2.5. Допускается содержание разделов иллюстрировать поясняющими примерами, таблицами, схемами, графиками.

(Измененная редакция, Изм. № 1).

2.6. В приложения к руководству оператора допускается включать различные материалы, которые нецелесообразно включать в разделы руководства.

(Введен дополнительно, Изм. № 1).

РУКОВОДСТВО ПРОГРАММИСТА.
ТРЕБОВАНИЯ К СОДЕРЖАНИЮ И ОФОРМЛЕНИЮ

United system for program documentation.

Programmer's guide. Requirements to contents and form of presentation

Постановлением Государственного комитета стандартов Совета Министров СССР от 12 января 1979 г. ¹ 74 срок введения установлен

с 01.01. 1980 г.

Настоящий стандарт устанавливает требования к содержанию и оформлению программного документа «Руководство программиста», определённого ГОСТ 19.101-77.

Стандарт полностью соответствует СТ СЭВ 2095-80.

1. ОБЩИЕ ПОЛОЖЕНИЯ

1.1. Структуру и оформление документа устанавливают в соответствии с ГОСТ 19.105-78.

Составление информационной части (аннотации и содержания) является обязательным.

1.2. Руководство программиста должно содержать следующие разделы:

- назначение и условия применения программ;
- характеристика программы;
- обращение к программе;
- входные и выходные данные;
- сообщения.

В зависимости от особенностей документы допускается объединять отдельные разделы или вводить новые.

2. СОДЕРЖАНИЕ РАЗДЕЛОВ

2.1. В разделе «Назначение и условия применения программ» должны быть указаны назначение и функции, выполняемые программой, условия,

необходимые для выполнения программы (объем оперативной памяти, требования к составу и параметрам периферийных устройств, требования к программного обеспечению и т.п.).

2.2. В разделе «Характеристика программы» должно быть приведено описание основных характеристик и особенностей программы (временные характеристики, режим работы, средства контроля правильности выполнения и самовосстанавливаемости программы и т.п.).

2.3. В разделе «Обращение к программе» должно быть приведено описание процедур вызова программы (способы передачи управления и параметров данных и др.).

2.4. В разделе «Входные и выходные данные» должно быть приведено описание организации используемой входной и выходной информации и, при необходимости, ее кодирования.

2.5. В разделе «Сообщения» должны быть указаны тексты сообщений, выдаваемых программисту или оператору в ходе выполнения программы, описание их содержания и действий, которые необходимо предпринять по этим сообщениям.

2.6. В приложении к руководству программиста могут быть приведены дополнительные материалы (примеры, иллюстрации, таблицы, графики и т.п.).

Практическая работа №7.

Тема: Освоение основных принципов работы в Rational Rose

Цель работы: изучить принципы работы с Case-пакетом Rational Rose 2003.

Rational Rose – семейство объектно-ориентированных Case-средств, предназначенных для автоматизации процессов анализа и проектирования ПО, для генерации кодов на различных языках программирования и выпуска проектной документации.

Назначение элементов экрана интерфейса Rose:

Браузер (browser) - используется для быстрой навигации по модели. С помощью браузера можно добавлять к модели элементы, просматривать существующие элементы модели и связи между ними, перемещать и переименовывать элементы модели, добавлять элементы модели к диаграмме, группировать элементы в пакеты, связывать элемент с файлом или адресом Интернета, работать с детализированной спецификацией элемента, открывать диаграмму. Браузер поддерживает четыре представления (view): представление вариантов использования, компонентов, размещения и логическое представление.

Окно документации (documentation window) – применяется для работы с текстовым описанием элементов модели. С его помощью можно документировать элементы модели Rose. Например, можно сделать краткое описание каждого действующего лица. При документировании класса все, что будет написано в окне документации, появится затем как комментарий в сгенерированном коде. Документация будет выводиться также в отчетах, создаваемых в среде Rose.

Панели инструментов (toolbars) - применяются для быстрого доступа к наиболее распространенным командам. Панели инструментов Rose обеспечивают быстрый доступ к наиболее распространенным командам. В этой среде существуют два типа панелей инструментов: стандартная панель и панель диаграммы. **Стандартная панель** видна всегда, ее кнопки соот-

ветствуют командам, которые могут использоваться для работы с любой диаграммой. **Панель диаграммы** своя для каждого типа диаграмм UML.

Все панели инструментов могут быть изменены и настроены пользователем. Для этого используется пункт меню **Tools > Options**, затем вкладку **Toolbars**.

Окно диаграммы (diagram window) - используется для просмотра и редактирования одной или нескольких диаграмм UML. В нем показано, как выглядит диаграммы UML-модели. При внесении в элементы диаграммы изменений Rose автоматически обновит браузер. Аналогично при внесении изменений в элемент с помощью браузера Rose автоматически обновит соответствующие диаграммы. Это помогает поддерживать модель в непротиворечивом состоянии.

Журнал (log) - применяется для просмотра ошибок и отчетов о выполнении различных команд. По мере работы над моделью определенная информация будет направляться в окно журнала. Например, туда помещаются сообщения об ошибках, возникающих при генерации кода. Не существует способа закрыть журнал совсем, но его окно может быть минимизировано.

На рис.1 показаны различные части интерфейса Rose.

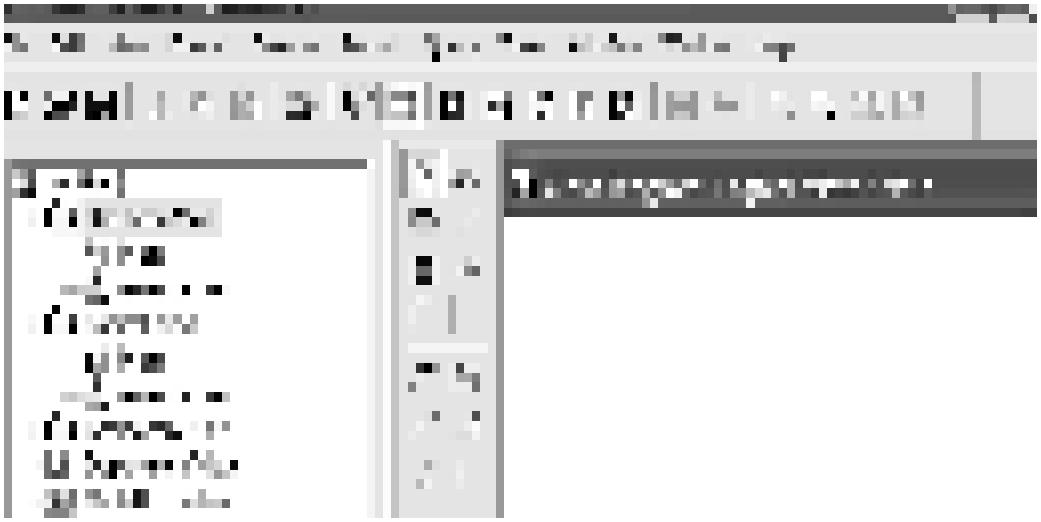


Рис.1. Интерфейс Rose

Четыре представления модели Rose

В модели Rose поддерживаются четыре представления - это **представление вариантов использования**, **логическое представление,**

представление компонентов и представление размещения. Каждое из них предназначено для своих целей.

Представление вариантов использования содержит всех действующих лиц, все варианты использования и их диаграммы для конкретной системы. Оно может также содержать некоторые диаграммы последовательности и кооперативные диаграммы. На рис.2 изображено представление вариантов использования в браузере Rose.

Представление вариантов использования содержит:

1. Действующих лиц.
2. Варианты использования.
3. Документацию по вариантам использования, описывающую происходящие в них процессы (потоки событий), включая обработку ошибок. Эта пиктограмма соответствует внешнему файлу, прикрепленному к модели Rose.
4. Диаграммы вариантов использования. Обычно у системы бывает несколько таких диаграмм, каждая из которых показывает подмножество действующих лиц и/или вариантов использования.
5. Пакеты, являющиеся группами вариантов использования и/или действующих лиц.

Рис.2. Представление вариантов использования

Логическое представление (рис. 3) показывает, как система будет реализовывать поведение, описанное в вариантах использования. Оно дает подробную картину составных частей системы и описывает взаимодействие этих частей. Логическое представление включает конкретные классы, диаграммы классов и диаграммы состояний. С их помощью конструируется детальный проект создаваемой системы.

Рис. 3 Логическое представление системы

Логическое представление содержит:

1. Классы.
2. Диаграммы классов. Как правило, для описания системы используется несколько диаграмм классов, каждая из которых отображает некоторое подмножество всех классов системы.
3. Диаграммы взаимодействия, применяемые для отображения объектов, участвующих в одном потоке событий варианта использования.
4. Диаграммы состояний.
5. Пакеты, являющиеся группами взаимосвязанных классов.

Представление компонентов содержит:

1. Компоненты, являющиеся физическими модулями кода.

2. Диаграммы компонентов.

3. Пакеты, являющиеся группами связанных компонентов.

Представление размещения - это последнее представление Rose. Оно соответствует физическому размещению системы, которое может отличаться от ее логической архитектуры.

<u>В представление размещения входят:</u>

1. Процессы, являющиеся потоками (threads), исполняемыми в отведенной для них области памяти.

2. Процессоры, включающие любые компьютеры, способные обрабатывать данные. Любой процесс выполняется на одном или нескольких процессорах.

3. Устройства, т.е. любая аппаратура, не способная обрабатывать данные (например, терминалы ввода-вывода и принтеры).

4. Диаграмма размещения.

Параметры настройки отображения (изображение атрибутов и операций на диаграммах классов)

<u>В Rose имеется возможность настроить диаграммы классов так, чтобы:</u>

1. Показывать все атрибуты и операции.

2. Скрыть операции/ Скрыть атрибуты.

3. Показывать только некоторые атрибуты или операции.

4. Показывать операции вместе с их полными сигнатурами или только их имена.

5. Показывать или не показывать видимость атрибутов и операций.

6. Показывать или не показывать стереотипы атрибутов и операций.

Значения каждого параметра по умолчанию можно задать с помощью окна, открываемого при выборе пункта меню Tools > Options.

Существуют два способа изменения параметров представления атрибутов на диаграмме. Можно установить нужные значения у каждого класса индивидуально. Можно также изменить значения нужных параметров

по умолчанию до начала создания диаграммы классов. Внесенные таким образом изменения повлияют только на вновь создаваемые диаграммы.

Дня переключения между нотациями видимости Rose и UML:

1. В меню модели выберите пункт Tools > Options.
2. Перейдите на вкладку Notation.
3. Для переключения между нотациями воспользуйтесь переключателем Visibility as Icons. Если этот переключатель помечен, будет использоваться нотация Rose, в противном случае - нотация UML. Изменение этого параметра повлияет только на новые диаграммы. Существующие диаграммы классов останутся прежними.

Задание

1. Выберите пункт **Tools > Options** и откройте вкладку **Toolbars**.

Чтобы сделать видимой или невидимой стандартную панель инструментов, пометьте (или снимите пометку) контрольный переключатель **show standard Too1Bar (или show Diagram ToolBar**).

2. Увеличьте размер кнопок на панели инструментов:

1. Щелкните правой кнопкой мыши по требуемой панели.

2. Выберите во всплывающем меню пункт Use Large Buttons (Использовать большие кнопки), вернитесь к нормальному размеру кнопок.

3. Настройте панель инструментов:

1. Щелкните правой кнопкой мыши по панели диаграммы Main пакета Use Case View.

2. Выберите пункт Customize (настроить) и добавьте несколько кнопок.

Чтобы добавить или удалить кнопки, выберите соответствующую кнопку и затем щелкните мышью по кнопке **Add** (добавить) или **Remove** (удалить).

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».

Практическая работа №8.

Тема: Создание модели вариантов использования

Цель работы: изучить принципы работы с Case-пакетом Rational Rose 2003.

Окончательный вид диаграммы показан на рис. 1.

Рис. 1 Диаграмма вариантов использования задачи о заказе товара.

Этапы выполнения упражнения.

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.

В 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.

Практическая работа № 9.

Тема: Анализ системы**.**

Цель работы: изучить принципы работы с Case-пакетом Rational Rose 2003.

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 упражнения

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 диаграмы последовательности

Практическая работа №10.

Тема: Создание диаграммы прецедентов

Цель работы: изучить принципы работы с Case-пакетом Rational Rose 2003.

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 ОК. 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

Практическая работа №11.

Тема: Создание диаграммы прецедентов**.**

Цель работы: изучить принципы работы с Case-пакетом Rational Rose 2003.

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

Рис 5. Диаграмма состояний для класса *Заказ*

Этапы выполнения упражнения

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 Actions(Действия).
4. В открывшемся меню выберите пункт Insert(Вставить).

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 OK, 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 OK, 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.

Практическая работа №12.

Тема: Создание диаграммы классов

Цель работы: изучить принципы работы с Case-пакетом Rational Rose 2003.

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 упражнения.

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. При-

свойте 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(промежуток 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

Практическая работа № 13.

Тема: Создание диаграммы классов

Цель работы: изучить принципы работы с Case-пакетом Rational Rose 2003.

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 упражнения Создание пакетов

1. Щелкните правой кнопкой мыши на Логическом представлении браузера.
2. В открывшемся меню выберите пункт New > Package здать >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.

Классы и пакеты в браузере показаны на рис. 9

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

Практическая работа №14.

Тема: Создание диаграмм деятельности**.**

Цель работы: изучить принципы работы с Case-пакетом Rational Rose 2003.

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 упражнения* 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 C1ass 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() : Boo1ean

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 ОК, 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 Boo1ean.

8. Перейдите на вкладку Detail(Подробно).

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 ОК, 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 Boo1ean.

17. Нажав на кнопку ОК, 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) : Boo1ean

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

Сохранить заказ(OrderID : Integer) : Boolean *Сохранить информацию*() : Integer

Практическая работа № 15

Тема: Создание диаграмм взаимодействия.

Цель работы: изучить принципы работы с Case-пакетом Rational Rose 2003.

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**

1. Найдите в браузере диаграмму Классов "Ввод нового заказа",
2. Дважды щелкнув на диаграмме, откройте ее.

3. 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.

4. Щелкните правой кнопкой мыши на панели инструментов диаграммы и в открывшемся меню выберите пункт Customize(Настроить),

5. 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

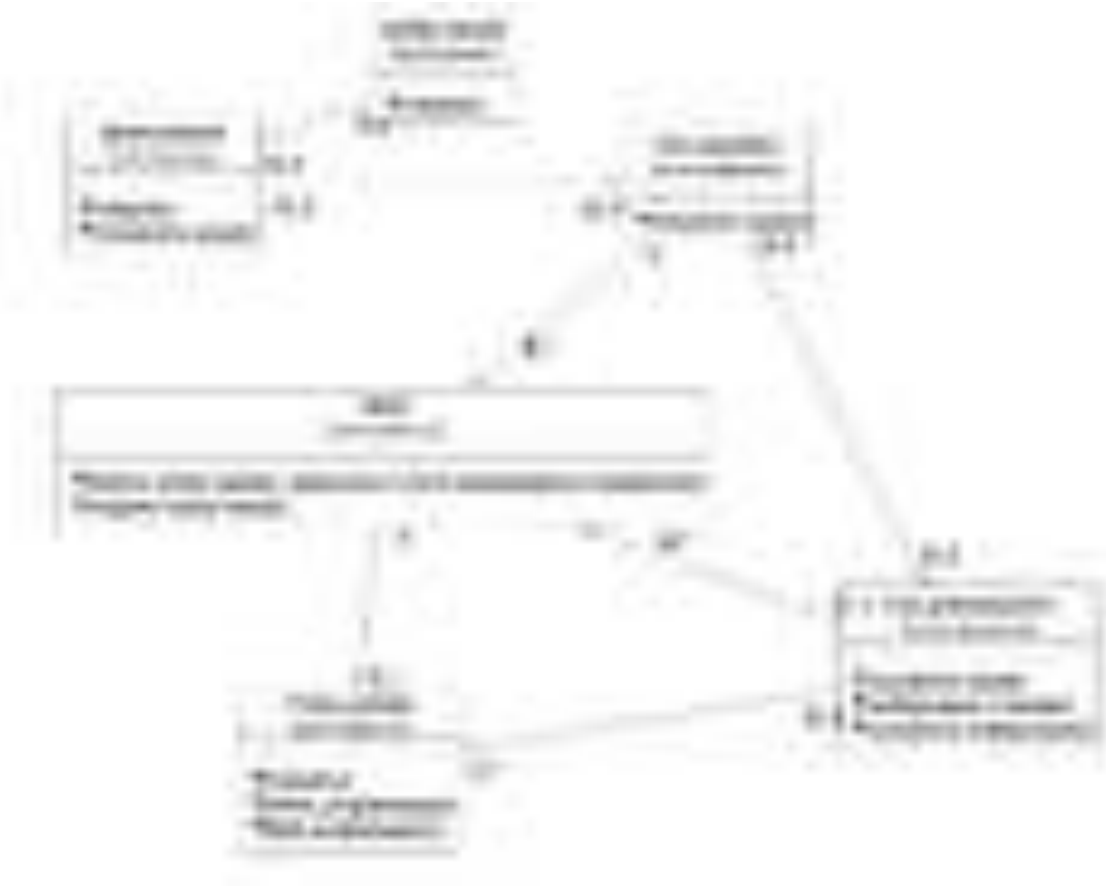


Рис. 10 Ассоциации сценария "Ввести новый заказ"

Практическая работа № 16.

Тема: Создание диаграммы состояний

Цель работы: изучить принципы работы с Case-пакетом Rational Rose 2003.

Этапы выполнения упражнения.

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)

Практическая работа № 17.

Тема: Кодогенерация проекта в Delphi.

Цель работы: изучить принципы работы с Case-пакетом Rational Rose 2003.

В настоящем разделе начинается построение физической модели системы (то есть программной системы).

Этапы выполнения упражнения

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 Con-

trol:

- 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

Рис. 13 Представление компонентов и классов в браузере

компонентов, с которыми этот класс связан (рис. 13)

Практическая работа №18

Тема: Анализ Delphi проекта, добавление визуальных объектов, реинжениринг в Rose.

Цель работы: изучить принципы работы с Case-пакетом Rational Rose 2003.

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.

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. Опишите интерпретацию рис. 17 с позиции объектно-ориентированного проектирования).

Практическая работа №19.

Тема: Кодогенерация модельных элементов. Построение диаграммы размещения

Цель работы: изучить принципы работы с Case-пакетом Rational Rose 2003.

В этом упражнении создается диаграмма Размещения для системы обработки заказов.

Рис. 18 Диаграмма размещения для модельной задачи

Этапы выполнения упражнения 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

Практическая работа №20.

Тема: «Создание элементарного приложения с ГИП средствами win32 Api.»

Цель работы: создать приложение с ГИП средствами Windows API, содержащее главное окно с меню.

Ход работы.

1. Создайте в среде Code::Blocks пустой проект. Для этого выполните в меню «File/New/Project». В появившемся окне выберете «Empty project» (рисунок 1) и нажмите кнопку «Go». Далее в появившемся окне нажмите «Next».

Рисунок 1. Окно выбора типа проекта

В следующем окне укажите имя проекта и путь к нему как показано на рисунке 2.

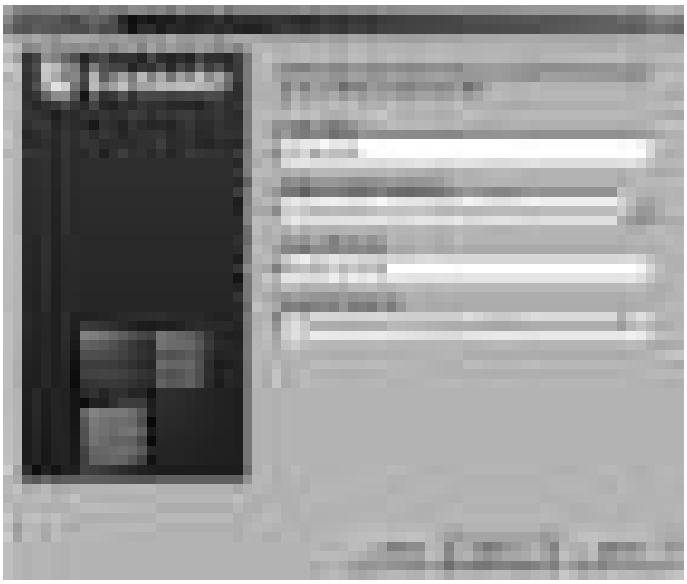


Рисунок 2. Выбор имени и пути проекта

Содержимое следующего окна оставьте как показано на рисунке 3.

Рисунок 3. Настройка целей сборки проекта.

2. Измените тип проекта на «GUI application». Для этого выберете в меню «Project/Properties». В появившемся окне (рисунок 4.) на вкладке «Build targets» укажите тип (Type) GUI Application и нажмите OK.

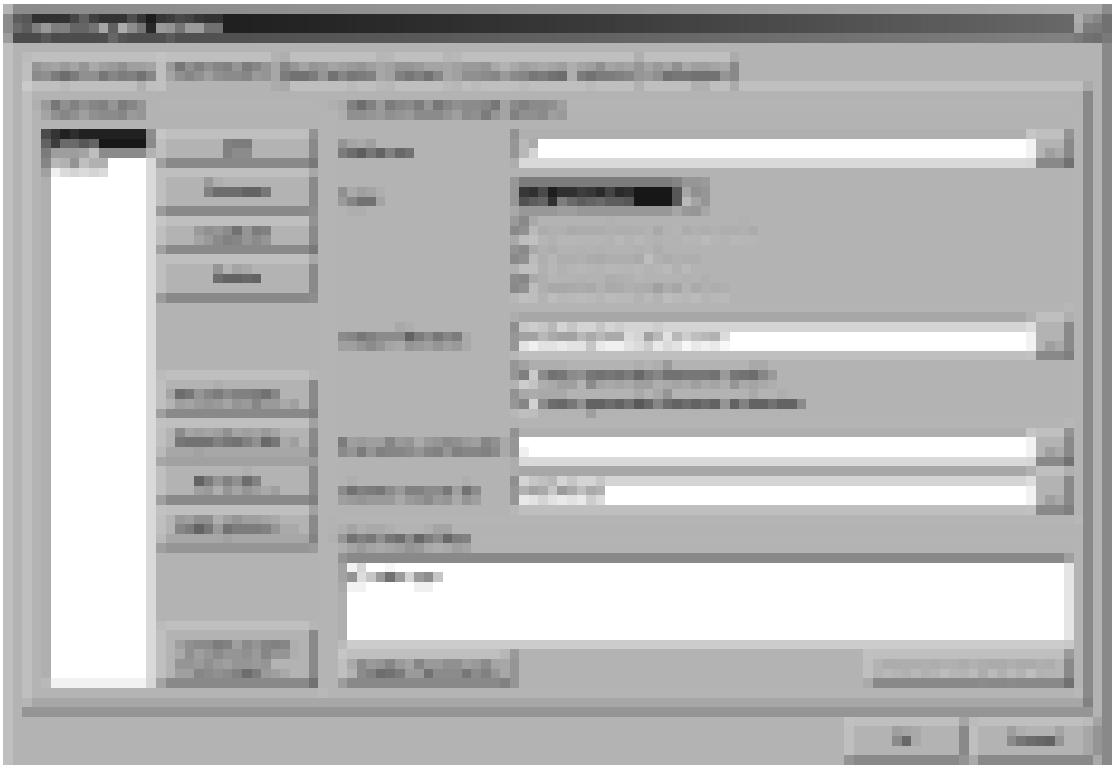


Рисунок 4. Окно настройки свойств проекта

3. Подключите библиотеку контролов libcomctl32. Для этого выберете пункт меню «Project/Build options». В появившемся окне (рисунок 5) выберете вкладку «

Linker settings». В ней нажмите кнопку «Add» и укажите путь к библиотеке. Библиотека libcomctl32.a расположена в <CodeBlocksPath>\MinGW\lib. Где CodeBlocksPath – путь к папке с установленным CodeBlocks.

Рисунок 5. Окно настройки опций компиляции

4. Добавьте в проект заголовочный файл main.h. Содержимое файла указано в листинге 1. Для этого выполните «File/New/Empty file». В появившемся окне нажмите кнопку «Да». В появившемся диалоге выбора файла (рисунок 6) укажите имя файла main.h.

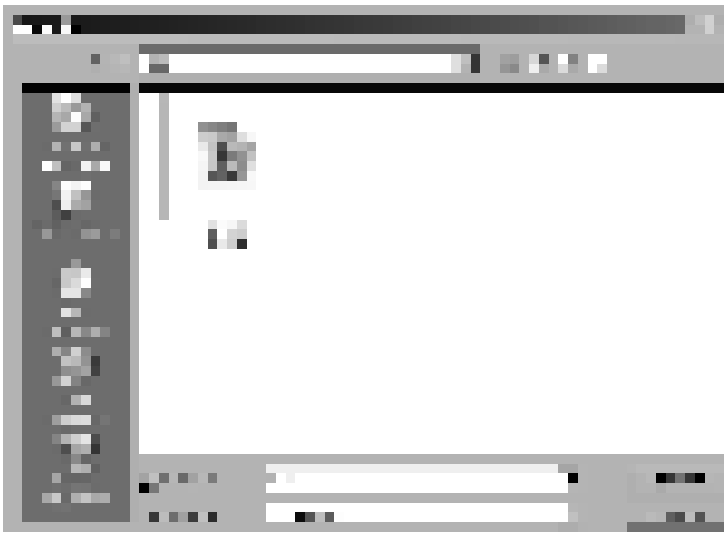


Рисунок 6. Окно сохранения файла

```
#define DLG_ABOUT  1001
#define IDR_MNU_MAIN  2001
#define IDM_ABOUT  6001
#define IDR_ICO_MAIN  8001
```

Листинг 1. Содержимое файла main.h

В появившемся окне укажите все как показано на рисунке 7.

Рисунок 7. Выбор целей сборки для файла

5. Аналогично добавьте в проект файл ресурсов main.rc. Содержимое файла указано в листинге 2

```
#include <windows.h>
#include <commctrl.h>
#include "main.h"

LANGUAGE LANG_ENGLISH,SUBLANG_ENGLISH_US

DLG_ABOUT DIALOGEX DISCARDABLE 6, 18, 195, 77
STYLE
DS_SHELLFONT|WS_POPUP|DS_MODALFRAME|DS_CENTER|WS_CAPTION|WS_S
YSMENU|DS_CONTEXTHELP|DS_3DLOOK|WS_VISIBLE
CAPTION "Диалог ""О программе"""
FONT 8, "MS Shell Dlg", 0, 0, 1
BEGIN
  CONTROL "OK", IDOK, "button", WS_TABSTOP, 75, 55, 45, 15
  CONTROL "Пример программы с применением Win32API", -1,
"static", WS_GROUP, 40, 10, 150, 10
  CONTROL "Версия 1.0", -1, "static", WS_GROUP, 40, 22, 150,
10
END
```

Листинг 2. Содержимое файла main.rc

5. Добавьте в проект файл исходных текстов main.cpp. Содержимое файла указано в приложении 1.
6. Скомпилируйте и запустите получившееся приложение.
7. Покажите готовое приложение преподавателю.
8. Добавьте в приложение свой пункт меню.
9. Подготовьте отчет о выполнении работы

Приложение 1. Содержимое файла main.cpp

```
#define _WIN32_IE 0x0600
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <windowsx.h>
#include <commctrl.h>
#include <tchar.h>
#include "main.h"
#define NELEMS(a)  (sizeof(a) / sizeof((a)[0]))
static LRESULT WINAPI MainWndProc(HWND, UINT, WPARAM, LPARAM);
static void Main_OnPaint(HWND);
static void Main_OnCommand(HWND, int, HWND, UINT);
static void Main_OnDestroy(HWND);
static LRESULT WINAPI AboutDlgProc(HWND, UINT, WPARAM, LPARAM);
static HINSTANCE__* ghInstance = 0;
int PASCAL WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpszCmdLine, int
nCmdShow)
{
    INITCOMMONCONTROLSEX icc;
    WNDCLASS wc;
    HWND hwnd;
    MSG msg;

    ghInstance = hInstance;

    icc.dwSize = sizeof(icc);
    icc.dwICC = ICC_WIN95_CLASSES;
    InitCommonControlsEx(&icc);
    wc.lpszClassName = _T("WinApiExClass");
    wc.lpfnWndProc = MainWndProc;
    wc.style = CS_OWNDC|CS_VREDRAW|CS_HREDRAW;
    wc.hInstance = ghInstance;
    wc.hIcon = LoadIcon(ghInstance, MAKEINTRESOURCE(IDR_ICO_MAIN));
    wc.hCursor = LoadCursor(NULL, IDC_ARROW);
    wc.hbrBackground = (HBRUSH)(COLOR_WINDOW+1);
    wc.lpszMenuName = MAKEINTRESOURCE(IDR_MNU_MAIN);
    wc.cbClsExtra = 0;
    wc.cbWndExtra = 0;
    if (!RegisterClass(&wc))
        return 1;
    hwnd = CreateWindow(_T("WinApiExClass"),
        _T("Пример программы с применением Win32API"),
        WS_OVERLAPPEDWINDOW|WS_HSCROLL|WS_VSCROLL,
        0,
        0,
        CW_USEDEFAULT,
        CW_USEDEFAULT,
        NULL,
        NULL,
        ghInstance,
        NULL
    );
    if (!hwnd) return 1;
    ShowWindow(hwnd, nCmdShow);
    UpdateWindow(hwnd);
    while (GetMessage(&msg, NULL, 0, 0))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
   return msg.wParam;
}
static LRESULT CALLBACK MainWndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam)
```

```
{
    switch (msg)
    {
        HANDLE_MSG(hwnd, WM_PAINT, Main_OnPaint);
        HANDLE_MSG(hwnd, WM_COMMAND, Main_OnCommand);
        HANDLE_MSG(hwnd, WM_DESTROY, Main_OnDestroy);
        /* TODO: enter more messages here */
        default:
            return DefWindowProc(hwnd, msg, wParam, lParam);
    }
}
static void Main_OnPaint(HWND hwnd)
{
    PAINTSTRUCT ps;
    RECT rc;

    BeginPaint(hwnd, &ps);
    GetClientRect(hwnd, &rc);
    EndPaint(hwnd, &ps);
}
static void Main_OnCommand(HWND hwnd, int id, HWND hwndCtl, UINT codeNotify)
{
    switch (id)
    {
        case IDM_ABOUT:
            DialogBox(ghInstance, MAKEINTRESOURCE(DLG_ABOUT), hwnd,
(DLGPROC)AboutDlgProc);
    }
}
static void Main_OnDestroy(HWND hwnd)
{
    PostQuitMessage(0);
}
static LRESULT CALLBACK AboutDlgProc(HWND hDlg, UINT uMsg, WPARAM wParam, LPARAM
lParam)
{
    switch (uMsg)
    {
        case WM_INITDIALOG:
            return TRUE;

        case WM_COMMAND:
            switch (wParam)
            {
                case IDOK:
                case IDCANCEL:
                    EndDialog(hDlg, TRUE);
                    return TRUE;
            }
            break;
    }
    return FALSE;
}
```

Практическая работа №21.

Тема: «Создание элементарного приложения с диалоговым интерфейсом»

Цель работы: создать диалоговое приложение для копирования файлов.

Ход работы.

1. Запустите QtCreator.

2. Создайте пустой проект Qt, для чего выполните следующий ряд шагов.

а) Выполнете в меню «Файл/Новый файл или проект».

б) В появившемся окне выберете пустой проект и нажмите «Выбрать» (рисунок 1).

Рисунок 1. Окно выбора типа нового проекта

в) В открывшемся мастере создания проекта укажите имя своего проекта как показано на рисунке 8, на следующих страницах нажмите кнопки «Далее» и «Завершить».

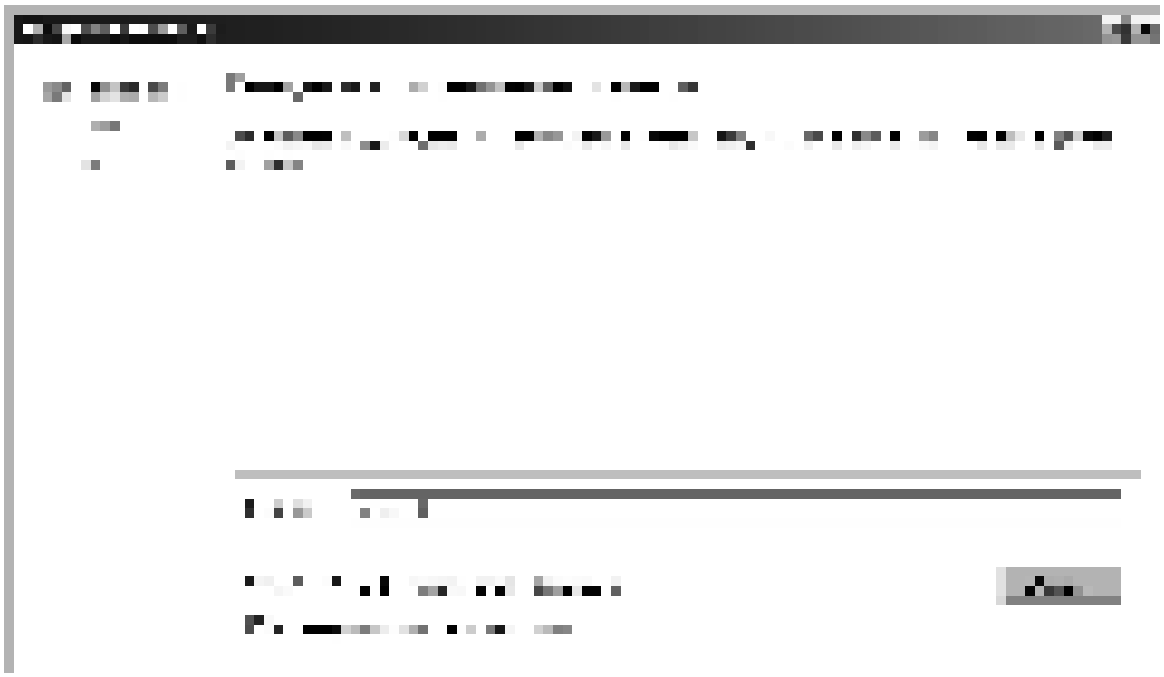


Рисунок 1. Диалог создания проекта: страница выбора имени проекта

г) Откройте двойным щелчком в менеджере проектов файл l3.pro и добавьте в него содержимое, представленное в листинге 1 и сохраните файл.

```
#Файл Qt проекта. Здесь комментарии обозначаются знаком #
#Шаблон проекта – приложение (может быть, например, lib - библиотека)
TEMPLATE=app
#Цель – имя выходного файла (exe или dll)
TARGET=lab-3-dialogs
#Настройки параметров конфигурации может принимать много параметров.
CONFIG += qt
#Используемые части Qt (Ядро и ГИП)
QT += core gui
#Каталоги с зависимостями (здесь указан . – текущий каталог проекта)
DEPENDPATH += .
#Каталоги, содержащие файлы заголовков здесь указан . – текущий каталог
проекта)
INCLUDEPATH += .
```

Листинг 1. Содержимое файла l3.pro

3. Добавьте в проект файл main.cpp. Для этого выполните следующий ряд шагов.

а) Правой кнопкой мыши щелкните на корне проекта в менеджере проектов. В появившемся меню выберете «Добавить новый».

б) В открывшемся окне выберете шаблон «Файл исходных текстов C++» (рисунок 9).

Рисунок 2. Окно выбора шаблона создаваемого файла

в) в появившемся мастере укажите имя файла main.

4. Откройте файл main.cpp (так же как и предыдущий файл) и добавьте в него содержимое листинга 2 и сохраните файл.

```
#include "maindialog.h"
#include <QApplication>
#include <QTextCodec>
int main(int argc, char* argv[])
{
//Создание объекта приложения
      QApplication app(argc, argv);
//Подключение текстового кодека для UTF-8
      QTextCodec::setCodecForCStrings(QTextCodec::codecForName("UTF-8"));
//Создание объекта диалога
      MainDialog dlg;
//Открытие диалога
      dlg.show();
//Запуск главного потока приложения
      return app.exec();
}
```

Листинг 2. Содержимое файла main.cpp

5. Аналогично добавьте в проект файл maindialog.h (только вместо «Файл исходных текстов C++» выберете «Заголовочный файл C++»); Содержимое файла maindialog.h представлено в листинге 3.

6. Аналогично добавьте в проект файл maindialog.cpp; Содержимое файла maindialog.cpp представлено в листинге 4.

```
//Код для предотвращения многократной загрузки заголовка
#ifndef MAINDIALOG_H
#define MAINDIALOG_H
//Подключение заголовка для QDialog
#include <QDialog>
//Предварительное объявления используемых классов
class QDialogButtonBox; //Панель с кнопками
class QLineEdit; //Редактор строки текста (поле ввода)
//Объявление собственного класса диалога, наследующего от QDialog
class MainDialog: public QDialog
{
//Специальный макрос, подключающий дополнительные возможности Qt
      Q_OBJECT
public:
//Конструктор, получающий параметром родительский виджет
      MainDialog(QWidget* parent = 0);
//Деструктор
      virtual ~MainDialog();
private:
//Панель с кнопками
      QDialogButtonBox* buttonBox;
//Поле ввода для ввода имени копируемого файла
      QLineEdit* leSourceFile;
//Поле ввода для ввода имени каталога назначения
      QLineEdit* leDistFolder;
private slots:
//Обработка сигнала accepted (принятие параметров диалога)
      void onAccept();
//Обработка сигнала rejected (отклонение параметров диалога
      void onReject();
//Выбор файла для копирования
      void openSourceFile();
//Выбор каталога назначения.
      void openDestinationFolder();
};

#endif // MAINDIALOG_H
```

Листинг 3. Содержимое файла maindialog.h

```
//Подключение заголовочных файлов Qt и файла maindialog.h
#include "maindialog.h"

#include <QDialogButtonBox>
#include <QGridLayout>
#include <QLineEdit>
#include <QToolButton>
#include <QFile>
#include <QFileInfo>
#include <QDir>
#include <QMessageBox>
#include <QLabel>
#include <QFileDialog>
```

Листинг 4. Содержимое файла maindialog.cpp

```
//Конструктор диалога
MainDialog::MainDialog(QWidget* parent) :
//Инициализация конструктора базового класса
      QDialog(parent),
//Инициализация панели кнопок. Сообщаем, что будут
//Использованы кнопки OK и Cancel.
//Панель будет горизонтальной
      buttonBox(new QDialogButtonBox(QDialogButtonBox::Ok |
QDialogButtonBox::Cancel, Qt::Horizontal, this)),
//Инициализация переменных для полей ввода новыми объектами
      leSourceFile(new QLineEdit(this)), leDistFolder(new
QLineEdit(this))
{
//Установка заголовка диалогового окна
      setWindowTitle("Диалог копирования файлов");
//Подключение сигналов диалога accepted и rejected
//К их обработчикам
      connect(this, SIGNAL(accepted()), this,
SLOT(onAccept()));
      connect(this, SIGNAL(rejected()), this,
SLOT(onReject()));
//Подключение сигналов accepted и rejected
//панели с кнопками (соответствуют кнопкам OK или Cancel)
//к соответствующим слотам (accept и reject) диалога
      connect(buttonBox, SIGNAL(accepted()), this,
SLOT(accept()));
      connect(buttonBox, SIGNAL(rejected()), this,
SLOT(reject()));
//У полей ввода устанавливаем свойство readOnly в true
      leSourceFile->setReadOnly(true);
      leDistFolder->setReadOnly(true);
//Кнопка для вызова диалога выбора файла - источника
      QToolButton* tbSource = new QToolButton(this);
      tbSource->setText("...");
      connect(tbSource, SIGNAL(released()), this,
SLOT(openSourceFile()));
//Кнопка для вызова диалога выбора папки назночения
      QToolButton* tbDest= new QToolButton(this);
      tbDest->setText("...");
      connect(tbDest, SIGNAL(released()), this,
SLOT(openDestinationFolder()));
//Создание объекта лэйоута (для расстановки
//виджетов в окне)
      QGridLayout* gl = new QGridLayout;
//Добавляем виджеты в лэйоут (надпись, поле ввода, кнопка)
      gl->addWidget(new QLabel("Источник", this), 0, 0);
      gl->addWidget(leSourceFile, 0, 1);
      gl->addWidget(tbSource, 0, 2, Qt::AlignLeft);
//Добавляем виджеты в лэйоут (надпись, поле ввода, кнопка)
      gl->addWidget(new QLabel("Папка назначения", this),
1, 0);
      gl->addWidget(leDistFolder, 1, 1);
      gl->addWidget(tbDest, 1, 2, Qt::AlignLeft);
//Добавляем виджеты в лэйоут (панель с кнопками)
      gl->addWidget(buttonBox, 3,0,1,3);
// Устанавливаем стретч-фактор для 2 строки в 1
//(параметр для растягивания строки по вертикали)
      gl->setRowStretch(2, 1);
//Устанавливаем gl главным лэйоутом окна
      setLayout(gl);
}
```

Листинг 4. Содержимое файла maindialog.cpp (продолжение)

```
//Деструктор
MainDialog::~MainDialog()
{
}
//Обработчик сигнала accepted (принятие параметров диалога)
void MainDialog::onAccept()
{
//Получение информации о файле-источнике и папке назначения
      QFileInfo fi_in (leSourceFile->text());
      QFileInfo fi_out (leDistFolder->text());
//Проверка на существование файла и папки
      if(!fi_in.exists() || !fi_out.exists())
      {
//Если файла или папки не существует, выдаем сообщения об
// ошибке
            QMessageBox::critical(this, "Критическая ошибка", "Файл-источник
или каталог назначения не существует");
      }
      else
      {
//Иначе выдаем информационное сообщение
            QMessageBox::information(this, "Копирование файла", "Копируем
файл\n\t" + fi_in.absoluteFilePath() +
"\nВ каталог\n\t" + fi_out.absoluteFilePath());
//Дезактивируем диалог
            setDisabled(true);
//Копируем файл в каталог назначения
            QFile::copy(fi_in.absoluteFilePath(), fi_out.absoluteFilePath() +
"/" + fi_in.fileName());
      }
}
//Обработчик сигнала rejected
//(отклонение параметров диалога
void MainDialog::onReject()
{
//Выводим соответствующее сообзение
      QMessageBox::information(this, "Выход из программы", "Выход из
программы без копирования файла");
}
//Выбор файла для копирования
void MainDialog::openSourceFile()
{
      QString fileName = QFileDialog::getOpenFileName(this, "Выбор файла для
копирования", QDir::homePath(),
"Все файлы (*)");
      if(!fileName.isEmpty()) leSourceFile->setText(fileName);
}
//Выбор каталога назначения
void MainDialog::openDestinationFolder()
{
      QString fileName = QFileDialog::getExistingDirectory(this, "Выбор папки
назначения", QDir::homePath());
      if(!fileName.isEmpty()) leDistFolder->setText(fileName);
}
```

Листинг 4. Содержимое файла maindialog.cpp (окончание)

7. Выполните команду «Сборка/Собрать все»

8. Выполните команду «Сборка/Запустить»

9. Покажите результат преподавателю.

10. Выполните отчет по практической работе.

Практическая работа №22.

Тема: «Создание элементарного SDI приложения»

Цель работы: Изучить на практике средства Qt для создания SDI приложений.

Ход работы.

1. Запустите QtCreator.

2. Откройте проект <QTDIR>/Examples/4.7/mainwindows/sdi/sdi.pro

3. Соберите и запустите проект.

4. Исследуйте содержимое файлов main.cpp, mainwindow.h, mainwindow.cpp.

5. Разберитесь со способом подключения и использования ресурсов в проекте.

6. Выполните отчет по практической работе, в котором письменно ответьте на следующие вопросы.

Вопросы по практической работе

1. Какой класс Qt реализует панель меню?
2. Какой класс Qt реалезует меню?
3. Какой класс Qt реализует действие (команду) меню?
4. Какой класс Qt реализует панель инструментов?
5. Какой класс Qt реализует строку состояния приложения?
6. Какой класс Qt реализует текстовый редактор?
7. Приведите пример кода для добавления меню на панель меню
8. Приведите пример кода создания действия (команды) меню с именем «Пример» и иконкой example.png, подгружаемой из ресурсов.
9. Приведите пример кода добавления действия в меню.
10. Приведите пример кода для добавления действия на панель инструментов.
11. Приведите пример кода для создания панели инструментов «Моя панель» и добавление ее в главное окно программы.

Практическая работа №23.

Тема: «Создание элементарного MDI приложения»

Цель работы: Изучить на практике средства Qt для создания MDI приложений.

Ход работы.

1. Запустите QtCreator.

2. Откройте проект <QTDIR>/Examples/4.7/mainwindows/mdi/mdi.pro

3. Соберите и запустите проект.

4. Исследуйте содержимое файлов main.cpp, mainwindow.h, mainwindow.cpp, mdichild.h, mdichild.cpp.

5. Разберитесь со способом подключения и использования ресурсов в проекте.

6. Выполните отчет по практической работе, в котором письменно ответьте на следующие вопросы.

Вопросы по практической работе

1. Какой класс Qt реализует панель меню?
2. Какой класс Qt реалезует меню?
3. Как0ой класс Qt реализует область подключения дочерних окон?
4. Какой класс Qt функционал дочернего окна?
5. Какой класс Qt реализует действие (команду) меню?
6. Какой класс Qt реализует панель инструментов?
7. Какой класс Qt реализует строку состояния приложения?
8. Какой класс Qt реализует текстовый редактор?
9. Приведите пример кода для добавления дочернего окна в mdiArea.
10. Опишите набор методов класса QMdiArea.
11. Опишите набор методов класса QMdiSubWindow.

Практическая работа №24.

Тема: «Обработка стандартных событий Qt-приложения»

Цель работы: Изучить на практике средства Qt для реализации работы с событиями.

Ход работы.

1. 1. Запустите QtCreator.

2. Создайте пустой проект Qt, для чего выполните следующий ряд шагов.

а) Выполнете в меню «Файл/Новый файл или проект».

б) В появившемся окне выберете пустой проект и нажмите «Выбрать» (рисунок 1).

Рисунок 1. Окно выбора типа нового проекта

в) В открывшемся мастере создания проекта укажите имя своего проекта как показано на рисунке 8, на следующих страницах нажмите кнопки «Далее» и «Завершить».

Рисунок 1. Диалог создания проекта: страница выбора имени проекта

г) Откройте двойным щелчком в менеджере проектов файл l3.pro и добавьте в него содержимое, представленное в листинге 1 и сохраните файл.

```
#Файл Qt проекта. Здесь комментарии обозначаются знаком #
#Шаблон проекта – приложение (может быть, например, lib -
библиотека)
TEMPLATE=app
#Цель – имя выходного файла (exe или dll)
TARGET=lab-3-dialogs
#Настройки параметров конфигурации может принимать много
параметров.
CONFIG += qt
#Используемые части Qt (Ядро и ГИП)
QT += core gui
#Каталоги с зависимостями (здесь указан . – текущий каталог
проекта)
DEPENDPATH += .
#Каталоги, содержащие файлы заголовков здесь указан . –
текущий каталог проекта)
INCLUDEPATH += .
```

Листинг 1. Содержимое файла l3.pro

3. Добавьте в проект файл main.cpp. Для этого выполните следующий ряд шагов.

а) Правой кнопкой мыши щелкните на корне проекта в менеджере

проектов. В появившемся меню выберете «Добавить новый».

б) В открывшемся окне выберете шаблон «Файл исходных текстов C++» (рисунок 9).

Рисунок 2. Окно выбора шаблона создаваемого файла

в) в появившемся мастере укажите имя файла main.

4. Откройте файл main.cpp (так же как и предыдущий файл) и добавьте в него содержимое листинга 2 и сохраните файл.

```
#include "maindialog.h"
#include <QApplication>
#include <QTextCodec>
int main(int argc, char* argv[])
{
//Создание объекта приложения
      QApplication app(argc, argv);
//Подключение текстового кодека для UTF-8
      QTextCodec::setCodecForCStrings(QTextCodec::codecForN
ame("UTF-8"));
//Создание объекта диалога
      MainDialog dlg;
//Открытие диалога
      dlg.show();
//Запуск главного потока приложения
      return app.exec();
}
```

Листинг 2. Содержимое файла main.cpp

5. Аналогично добавьте в проект файл maindialog.h (только вместо «Файл исходных текстов C++» выберете «Заголовочный файл C++»);

Содержимое файла maindialog.h представлено в листинге 3.

6. Аналогично добавьте в проект файл maindialog.cpp; Содержимое файла maindialog.cpp представлено в листинге 4.

```
//Код для предотвращения многократной загрузки заголовка
#ifndef MAINDIALOG_H
#define MAINDIALOG_H
//Подключение заголовка для QDialog
#include <QDialog>
//Предварительное объявления используемых классов
class QDialogButtonBox; //Панель с кнопками
class QLineEdit; //Редактор строки текста (поле ввода)
//Объявление собственного класса диалога, наследующего от
QDialog
class MainDialog: public QDialog
{
//Специальный макрос, подключающий дополнительные
возможности Qt
      Q_OBJECT
public:
//Конструктор, получающий параметром родительский виджет
      MainDialog(QWidget* parent = 0);
//Деструктор
      virtual ~MainDialog();
private:
//Функции-обработчики событий
      virtual void closeEvent(QCloseEvent * e);
      virtual void showEvent(QShowEvent * e);
      virtual void mousePressEvent(QMouseEvent *e);
};

#endif // MAINDIALOG_H
```

Листинг 3. Содержимое файла maindialog.h

```
//Подключение заголовочных файлов Qt и файла maindialog.h
#include "maindialog.h"
#include <QMessageBox>
#include <QCloseEvent>

MainDialog::MainDialog(QWidget *parent)
    : QDialog(parent)
{
}
MainDialog::~MainDialog()
{

}
```

Листинг 4. Содержимое файла maindialog.cpp

```
void MainDialog::closeEvent(QCloseEvent *e)
{
	int btn = QMessageBox::question(this, "Закрытие
прорграммы", "Вы готовы закрыть программу?",

	QMessageBox::Yes, QMessageBox::No);
	if(btn == QMessageBox::Yes) QDialog::closeEvent(e);
	else e->ignore();
}

void MainDialog::showEvent(QShowEvent *e)
{
	int btn = QMessageBox::question(this, "Главное окно
прорграммы",

	"Отображается главное окно программы. Продолжить?",

	QMessageBox::Yes, QMessageBox::No);
	if(btn == QMessageBox::Yes) QDialog::showEvent(e);
	else
	{
		e->ignore();
		close();
	}
}

void MainDialog::mousePressEvent(QMouseEvent *e)
{
	QString btn;
	switch(e->button())
	{
	case Qt::LeftButton: btn = "левая";break;
	case Qt::RightButton: btn = "правая";break;
	case Qt::MidButton: btn = "средняя";break;
	default: btn = "неизвестная";break;
	}
	QMessageBox::information(this,
tr("QMouseEvent"),QString("В позиции (%1, %2) нажата
<b>%3</b> кнопка!")
	.arg(e-
>pos().x()).arg(e->pos().y()).arg(btn));
}
```

Листинг 4. Содержимое файла maindialog.cpp (продолжение)

7. Выполните команду «Сборка/Собрать все»

8. Выполните команду «Сборка/Запустить»

9. Покажите результат преподавателю.

10. Выполните отчет по практической работе.

Практическая работа №25.

Тема: «Создание сигналов и слотов Qt»

Цель работы: Изучить на практике систему сигналов и слотов Qt.

Ход работы.

1. 1. Запустите QtCreator.

2. Создайте пустой проект Qt, для чего выполните следующий ряд шагов.

а) Выполнете в меню «Файл/Новый файл или проект».

б) В появившемся окне выберете пустой проект и нажмите «Выбрать» (рисунок 1).

Рисунок 1. Окно выбора типа нового проекта

в) В открывшемся мастере создания проекта укажите имя своего проекта – 17 – (рисунке 2), на следующих страницах нажмите кнопки «Далее» и «Завершить».

Рисунок 2. Диалог создания проекта: страница выбора имени проекта

г) Откройте двойным щелчком в менеджере проектов файл l7.pro и добавьте в него содержимое, представленное в листинге 1 и сохраните файл.

```
#Файл Qt проекта. Здесь комментарии обозначаются знаком #
#Шаблон проекта – приложение (может быть, например, lib -
библиотека)
TEMPLATE=app
#Цель – имя выходного файла (exe или dll)
TARGET=lab-7-signals_and_slots
#Настройки параметров конфигурации может принимать много
параметров.
CONFIG += qt
#Используемые части Qt (Ядро и ГИП)
QT += core gui
#Каталоги с зависимостями (здесь указан . – текущий каталог
проекта)
DEPENDPATH += .
#Каталоги, содержащие файлы заголовков здесь указан . –
текущий каталог проекта)
INCLUDEPATH += .
```

Листинг 1. Содержимое файла l7.pro

3. Добавьте в проект файхл main.cpp. Для этого выполните следующий ряд шагов.

а) Правой кнопкой мыши щелкните на корне проекта в менеджере

проектов. В появившемся меню выберете «Добавить новый».

б) В открывшемся окне выберете шаблон «Файл исходных текстов C++» (рисунок).

Рисунок 3. Окно выбора шаблона создаваемого файла

в) в появившемся мастере укажите имя файла main.

4. Откройте файл main.cpp (так же как и предыдущий файл) и добавьте в него содержимое листинга 2 и сохраните файл.

```
#include <QApplication>
#include <QTextCodec>
#include <mainwidget.h>

int main(int argc, char* argv[])
{
      QApplication app(argc, argv);

      QTextCodec::setCodecForCStrings(QTextCodec::codecForN
ame("UTF-8"));

      MainWidget* w = new MainWidget(0);
      w->show();

      return app.exec();
}
```

Листинг 2. Содержимое файла main.cpp

5. Аналогично добавьте в проект файл mainwidget.h (только вместо «Файл исходных текстов C++» выберете «Заголовочный файл C++»);

Содержимое файла mainwidget.h представлено в листинге 3.

6. Аналогично добавьте в проект файл mainwidget.cpp; Содержимое файла mainwidget.cpp представлено в листинге 4.

```
#ifndef MAINWIDGET_H
#define MAINWIDGET_H

#include <QWidget>

class QTextEdit;
class QPushButton;

class MainWidget : public QWidget
{
    Q_OBJECT
public:
    explicit MainWidget(QWidget *parent = 0);
      virtual ~MainWidget();

signals:
      void sendText(const QString& text);
private:
      QTextEdit* teText;
      MainWidget* mwDouble;
      QPushButton* pbDouble;
      QPushButton* pbConnect;
      QPushButton* pbDisconnect;
      QPushButton* pbSendText;
      QPushButton* pbCloseDouble;
private slots:
      void setText(const QString& text);
      void sendText();
      void createDouble();
      void closeDouble();
      void createConnections();
      void destroyConnections();
      void onDoubleDestroyed();
};

#endif // MAINWIDGET_H
```

Листинг 3. Содержимое файла mainwidget.h

```
#include "mainwidget.h"

#include <QtCore>
#include <QtGui>
```

Листинг 4. Содержимое файла mainwidget.cpp

```
MainWidget::MainWidget(QWidget *parent) :
      QWidget(parent), teText(new QTextEdit(this)),
mwDouble(0),
      pbDouble(new QPushButton("Создать копию окна",
this)),
      pbConnect(new QPushButton("Подключить сигналы и
слоты", this)),
      pbDisconnect(new QPushButton("Отключить сигналы и
слоты", this)),
      pbSendText(new QPushButton("Послать текст", this)),
      pbCloseDouble(new QPushButton("Закрыть копию", this))
{
      QGridLayout* gl = new QGridLayout;
      gl->addWidget(teText, 0, 0, 1, 2);
      gl->addWidget(pbSendText, 1, 1, Qt::AlignRight);
      gl->addWidget(pbDouble, 2, 0, Qt::AlignLeft);
      gl->addWidget(pbCloseDouble, 2, 1, Qt::AlignRight);
      gl->addWidget(pbConnect, 3, 0, Qt::AlignLeft);
      gl->addWidget(pbDisconnect, 3, 1, Qt::AlignRight);
      gl->setRowStretch(4, 1);
      setLayout(gl);

      connect(pbDouble, SIGNAL(released()), this,
SLOT(createDouble()));
      connect(pbConnect, SIGNAL(released()), this,
SLOT(createConnections()));
      connect(pbDisconnect, SIGNAL(released()), this,
SLOT(destroyConnections()));
      connect(pbSendText, SIGNAL(released()), this,
SLOT(sendText()));
      connect(pbCloseDouble, SIGNAL(released()), this,
SLOT(closeDouble()));
}

MainWidget::~MainWidget()
{
}

void MainWidget::setText(const QString &text)
{
      teText->append(text);
}

void MainWidget::sendText()
{
      emit sendText(teText->toPlainText());
}

void MainWidget::createDouble()
{
      if(!mwDouble)
      {
            mwDouble = new MainWidget(0);
            mwDouble->show();
            connect(mwDouble, SIGNAL(destroyed()), this,
SLOT(onDoubleDestroyed()));
            mwDouble->mwDouble = this;
            connect(this, SIGNAL(destroyed()), mwDouble,
SLOT(onDoubleDestroyed()));
      }
}
```

Листинг 4. Содержимое файла mainwidget.cpp (продолжение)

```
void MainWidget::closeDouble()
{
	if(mwDouble)
	{
		mwDouble->close();
		mwDouble->deleteLater();
	}
}

void MainWidget::createConnections()
{
	if(mwDouble)
	{
		connect(this, SIGNAL(sendText(QString)),
mwDouble, SLOT(setText(QString)));
		connect(mwDouble, SIGNAL(sendText(QString)),
this, SLOT(setText(QString)));
	}
}

void MainWidget::destroyConnections()
{
	if(mwDouble)
	{
		mwDouble->disconnect(this,
SLOT(setText(QString)));
		disconnect(mwDouble, SLOT(setText(QString)));
	}
}

void MainWidget::onDoubleDestroyed()
{
	mwDouble = 0;
}
```

Листинг 4. Содержимое файла mainwidget.cpp (продолжение)

7. Выполните команду «Сборка/Собрать все»

8. Выполните команду «Сборка/Запустить»

9. Воспользовавшись QtAssistant и книгой Жасмин Бланшет «Т 4 программирование GUI на C++», напишите подробные комментарии к каждой строке файлов «mainwidget.h» и «mainwidget.cpp».

10. Покажите результат преподавателю.

11. Выполните отчет по практической работе.

Практическая работа №26.

Тема: «Создание приложения – обработчика клавиатуры для Windows»

Цель работы: Изучить на практике работу с различными видами элементов ввода в Qt.

Ход работы.

1. Запустите QtCreator.

2. Создайте пустой проект Qt, для чего выполните следующий ряд шагов.

а) Выполнете в меню «Файл/Новый файл или проект».

б) В появившемся окне выберете пустой проект и нажмите «Выбрать» (рисунок 1).

Рисунок 1. Окно выбора типа нового проекта

в) В открывшемся мастере создания проекта укажите имя своего проекта – 111 – (рисунке 2), на следующих страницах нажмите кнопки «Далее» и «Завершить».

Рисунок 2. Диалог создания проекта: страница выбора имени проекта

г) Откройте двойным щелчком в менеджере проектов файл l11.pro и добавьте в него содержимое, представленное в листинге 1 и сохраните файл.

```
TEMPLATE=app
TARGET=lab-11-input
CONFIG += qt
QT += core gui
DEPENDPATH += .
INCLUDEPATH += .
```

Листинг 1. Содержимое файла l11.pro

3. Добавьте в проект файхл main.cpp. Для этого выполните следующий ряд шагов.

а) Правой кнопкой мыши щелкните на корне проекта в менеджере проектов. В появившемся меню выберете «Добавить новый».

б) В открывшемся окне выберете шаблон «Файл исходных текстов C++» (рисунок 3).

Рисунок 3. Окно выбора шаблона создаваемого файла

в) в появившемся мастере укажите имя файла main.

4. Откройте файл main.cpp (так же как и предыдущий файл) и добавьте в него содержимое листинга 2 и сохраните файл.

```
#include <QApplication>
#include <QTextCodec>
#include <mainwidget.h>

int main(int argc, char* argv[])
{
      QApplication app(argc, argv);

      QTextCodec::setCodecForCStrings(QTextCodec::codecForN
ame("UTF-8"));
      QTextCodec::setCodecForTr(QTextCodec::codecForName("U
TF-8"));
      MainWidget* w = new MainWidget(0);
      w->show();
      return app.exec();
}
```

Листинг 2. Содержимое файла main.cpp

5. Аналогично добавьте в проект файл mainwidget.h (только вместо «Файл исходных текстов C++» выберете «Заголовочный файл C++»); Содержимое файла mainwidget.h представлено в листинге 3.

```
#ifndef MAINWIDGET_H
#define MAINWIDGET_H

#include <QWidget>

class QTextBrowser;
class QLineEdit;
class QTextEdit;
class QDoubleSpinBox;
class QSpinBox;
class QSlider;
class QDial;
class QTimeEdit;
class QDateEdit;
class QDateTimeEdit;

class MainWidget : public QWidget
{
    Q_OBJECT
public:
    explicit MainWidget(QWidget *parent = 0);
      virtual ~MainWidget();
private:
      QTextBrowser* tbLog;
      QLineEdit* lineEdit;
      QTextEdit* textEdit;

      QDoubleSpinBox* doubleSpinBox;
      QSpinBox* spinBox;
      QDial* dial;
      QSlider* slider;
      QTimeEdit* timeEdit;
      QDateEdit* dateEdit;
      QDateTimeEdit* dateTimeEdit;

      QWidget* prepareText();
      QWidget* prepareNumbers();
      QWidget* prepareDateTime();


private slots:

      void lineEditToLog();
      void textEditToLog();
      void textEditToLogAsHtml();
      void numbersToLog();
      void dateTimeToLog();

};
```

Листинг 3. Содержимое файла mainwidget.h

6. Аналогично добавьте в проект файл mainwidget.cpp; Содержимое файла mainwidget.cpp представлено в листинге 4.

```
#include "mainwidget.h"

#include <QtCore>
#include <QtGui>

MainWidget::MainWidget(QWidget *parent) :
    QWidget(parent)
{
      QTabWidget* tb = new QTabWidget(this);
      tbLog = new QTextBrowser(this);

      QVBoxLayout* vl = new QVBoxLayout;
      vl->addWidget(tb);
      vl->addWidget(tbLog);
      setLayout(vl);

      tb->addTab(prepareText(), tr("Текст"));
      tb->addTab(prepareNumbers(), tr("Числа"));
      tb->addTab(prepareDateTime(), tr("Дата и время"));
}

MainWidget::~MainWidget()
{
}

QWidget * MainWidget::prepareText()
{
      QWidget* cont = new QWidget(this);

      lineEdit = new QLineEdit(this);
      textEdit = new QTextEdit(this);

      QPushButton* pbLineEdit = new QPushButton(tr("В
журнал"), this);
      connect(pbLineEdit, SIGNAL(released()), this,
SLOT(lineEditToLog()));
      QPushButton* pTextEdit = new QPushButton(tr("В
журнал"), this);
      connect(pTextEdit, SIGNAL(released()), this,
SLOT(textEditToLog()));
      QPushButton* pTextEditAsHtml = new QPushButton(tr("В
журнал как html"), this);
      connect(pTextEditAsHtml, SIGNAL(released()), this,
SLOT(textEditToLogAsHtml()));

      QGridLayout* gl = new QGridLayout;
      gl->addWidget(new QLabel("QLineEdit", this), 0, 0);
      gl->addWidget(lineEdit, 0, 1);
      gl->addWidget(pbLineEdit, 0, 2);
      gl->addWidget(new QLabel("QTextEdit", this), 1, 0);
      gl->addWidget(pTextEditAsHtml, 1, 1);
      gl->addWidget(pTextEdit, 1, 2);
      gl->addWidget(textEdit, 2, 0, 1, 3);

      cont->setLayout(gl);

      return cont;
}
```

Листинг 4. Содержимое файла mainwidget.cpp

```
QWidget * MainWidget::prepareNumbers()
{
      QWidget* cont = new QWidget(this);

      doubleSpinBox = new QDoubleSpinBox(this);
      doubleSpinBox->setMinimum(-99.99);
      doubleSpinBox->setMaximum(99.99);

      spinBox = new QSpinBox(this);
      spinBox->setMinimum(-100);
      spinBox->setMaximum(100);

      slider = new QSlider(this);
      slider->setMinimum(-100);
      slider->setMaximum(100);
      slider->setValue(spinBox->value());
      slider->setOrientation(Qt::Horizontal);

      dial = new QDial(this);
      dial->setRange(-100, 100);
      dial->setValue(spinBox->value());

      connect(spinBox, SIGNAL(valueChanged(int)), slider,
SLOT(setValue(int)));
      connect(spinBox, SIGNAL(valueChanged(int)), dial,
SLOT(setValue(int)));
      connect(slider, SIGNAL(valueChanged(int)), spinBox,
SLOT(setValue(int)));
      connect(slider, SIGNAL(valueChanged(int)), dial,
SLOT(setValue(int)));
      connect(dial, SIGNAL(valueChanged(int)), spinBox,
SLOT(setValue(int)));
      connect(dial, SIGNAL(valueChanged(int)), slider,
SLOT(setValue(int)));

      QPushButton* pbLog = new QPushButton(tr("В журнал"),
this);
      connect(pbLog, SIGNAL(released()), this,
SLOT(numbersToLog()));

      QGridLayout* gl = new QGridLayout;

      gl->addWidget(new QLabel("QDoubleSpinBox", this), 0,
0);
      gl->addWidget(doubleSpinBox, 0, 1);
      gl->addWidget(new QLabel("QSpinBox", this), 1, 0);
      gl->addWidget(spinBox, 1, 1);
      gl->addWidget(new QLabel("QSLider", this), 2, 0);
      gl->addWidget(slider, 2, 1);
      gl->addWidget(new QLabel("QDial", this), 0, 2,
Qt::AlignCenter);
      gl->addWidget(dial, 1, 2, 2, 1);
      gl->addWidget(pbLog, 3, 1, Qt::AlignCenter);

      gl->setRowStretch(4, 1);

      cont->setLayout(gl);
      return cont;}
```

Листинг 4. Содержимое файла mainwidget.cpp (продолжение)

```
QWidget * MainWidget::prepareDateTime()
{
      QWidget* cont = new QWidget(this);

      timeEdit = new QTimeEdit(this);
      timeEdit->setTime(QTime::currentTime());
      dateEdit = new QDateEdit(this);
      dateEdit->setDate(QDate::currentDate());
      dateEdit->setCalendarPopup(true);
      dateTimeEdit = new QDateTimeEdit(this);
      dateTimeEdit-
>setDateTime(QDateTime::currentDateTime());
      dateTimeEdit->setCalendarPopup(true);

      QPushButton* pbLog = new QPushButton(tr("В журнал"),
this);
      connect(pbLog, SIGNAL(released()), this,
SLOT(dateTimeToLog()));

      QGridLayout* gl = new QGridLayout;
      gl->addWidget(new QLabel("QTimeEdit", this), 0, 0);
      gl->addWidget(timeEdit, 0, 1);
      gl->addWidget(new QLabel("QDateEdit", this), 1, 0);
      gl->addWidget(dateEdit, 1, 1);
      gl->addWidget(new QLabel("QDateTimeEdit", this), 2,
0);
      gl->addWidget(dateTimeEdit, 2, 1);
      gl->addWidget(pbLog, 3, 1, Qt::AlignRight);
      gl->setRowStretch(4, 1);
      cont->setLayout(gl);
      return cont;
}
void MainWidget::lineEditToLog()
{
      tbLog-
>setPlainText(QString("QLineEdit: %1").arg(lineEdit-
>text()));
}
void MainWidget::textEditToLog()
{
      tbLog->setPlainText(textEdit->toPlainText());
}
void MainWidget::textEditToLogAsHtml()
{
      tbLog->setPlainText(textEdit->toHtml());
}
```

Листинг 4. Содержимое файла mainwidget.cpp (продолжение)

```
void MainWidget::numbersToLog()
{
      QStringList out;
      out << tr("Данные выводятся в следующей
плследовательности: \n"
                    "Виджет: minimum, maximum, шаг, текущее
значение");
      out <<
QString("QDoubleSpinBox: %1, %2, %3, %4").arg(doubleSpinBox
->minimum())
               .arg(doubleSpinBox-
>maximum()).arg(doubleSpinBox-
>singleStep()).arg(doubleSpinBox->value());
      out <<
QString("QSpinBox: %1, %2, %3, %4").arg(spinBox->minimum())
               .arg(spinBox->maximum()).arg(spinBox-
>singleStep()).arg(spinBox->value());\
      out << QString("QSlider: %1, %2, %3, %4").arg(slider-
>minimum())
               .arg(slider->maximum()).arg(slider-
>singleStep()).arg(slider->value());
      out << QString("QDial: %1, %2, %3, %4").arg(dial-
>minimum())
               .arg(dial->maximum()).arg(dial-
>singleStep()).arg(dial->value());
      tbLog->setText(out.join("\n"));
}

void MainWidget::dateTimeToLog()
{
      QStringList out;
      out << QString("QTimeEdit: %1 или %2").arg(timeEdit-
>time().toString())
               .arg(timeEdit->time().toString("hh:mm:ss"));
      out << QString("QDateEdit: %1 или %2").arg(dateEdit-
>date().toString())
               .arg(dateEdit->date().toString("dd.MM.yy"));
      out << QString("QDateTimeEdit: %1
или %2").arg(dateTimeEdit->dateTime().toString())
               .arg(dateTimeEdit-
>dateTime().toString("dd.MM.yy hh:mm:ss"));
      tbLog->setText(out.join("\n"));
}
```

Листинг 4. Содержимое файла mainwidget.cpp (продолжение)

7. Выполните команду «Сборка/Собрать все»

8. Выполните команду «Сборка/Запустить»

9. Воспользовавшись QtAssistant и книгой Жасмин Бланшет «Т 4 программирование GUI на C++», напишите подробные комментарии к каждой строке файлов «mainwidget.h» и «mainwidget.cpp».

10. Покажите результат преподавателю.

11. Выполните отчет по практической работе.

Практическая работа №27.

Тема: «Создание Windows - приложения, использующего клавиатуру и мышь»

Цель работы: Изучить на практике работу с различными видами элементов вывода в Qt.

Ход работы.

1. Запустите QtCreator.

2. Создайте пустой проект Qt, для чего выполните следующий ряд шагов.

а) Выполнете в меню «Файл/Новый файл или проект».

б) В появившемся окне выберете пустой проект и нажмите «Выбрать» (рисунок 1).

Рисунок 1. Окно выбора типа нового проекта

в) В открывшемся мастере создания проекта укажите имя своего проекта – 112 – (рисунке 2), на следующих страницах нажмите кнопки «Далее» и «Завершить».

Рисунок 2. Диалог создания проекта: страница выбора имени проекта

г) Откройте двойным щелчком в менеджере проектов файл l12.pro и добавьте в него содержимое, представленное в листинге 1 и сохраните файл.

```
TEMPLATE=app
TARGET=lab-12-output
CONFIG += qt
QT += core gui
DEPENDPATH += .
INCLUDEPATH += .
```

Листинг 1. Содержимое файла l12.pro

3. Добавьте в проект файхл main.cpp. Для этого выполните следующий ряд шагов.

а) Правой кнопкой мыши щелкните на корне проекта в менеджере проектов. В появившемся меню выберете «Добавить новый».

б) В открывшемся окне выберете шаблон «Файл исходных текстов C++» (рисунок 3).

Рисунок 3. Окно выбора шаблона создаваемого файла

в) в появившемся мастере укажите имя файла main.

4. Откройте файл main.cpp (так же как и предыдущий файл) и добавьте в него содержимое листинга 2 и сохраните файл.

```
#include <QApplication>
#include <QTextCodec>
#include <mainwidget.h>

int main(int argc, char* argv[])
{
      QApplication app(argc, argv);

      QTextCodec::setCodecForCStrings(QTextCodec::codecForN
ame("Windows-1251"));
      QTextCodec::setCodecForTr(QTextCodec::codecForName("W
indows-1251"));
      MainWidget* w = new MainWidget(0);
      w->show();
      return app.exec();
}
```

Листинг 2. Содержимое файла main.cpp

5. Аналогично добавьте в проект файл mainwidget.h (только вместо «Файл исходных текстов C++» выберете «Заголовочный файл C++»); Содержимое файла mainwidget.h представлено в листинге 3.

```
#ifndef MAINWIDGET_H
#define MAINWIDGET_H

#include <QWidget>

class QLabel;
class QTextBrowser;
class QProgressBar;
class QLCDNumber;
class QCalendarWidget;

class MainWidget : public QWidget
{
    Q_OBJECT
public:
      explicit MainWidget(QWidget *parent = 0);
      virtual ~MainWidget();
private:
//Надпись. будет содержать link на google
      QLabel* mHrefLabel;
//Надпись. будет содержать имя открываемого файла
      QLabel* mFileNameLabel;
//Отображает rich текст. то же самое, что QTextEdit, но без
редактирования
      QTextBrowser* mText;
//Строка прогресса
      QProgressBar* mProgressBar;
//Отображает числа как семисенгментный индикатор
      QLCDNumber* mIndicator;
//Просто календарь
      QCalendarWidget* mCalendar;

private slots:
      void openFile();
      void onLabelLinkActivated(const QString& link);

};

#endif // MAINWIDGET_H
#endif // MAINWIDGET_H
```

Листинг 3. Содержимое файла mainwidget.h

6. Аналогично добавьте в проект файл mainwidget.cpp; Содержимое файла mainwidget.cpp представлено в листинге 4.

```
#include "mainwidget.h"
#include <QtCore>
#include <QtGui>
MainWidget::MainWidget(QWidget *parent) :
    QWidget(parent)
{
      //Создаем надпись, содержащую ссылку на гугл
      mHrefLabel = new QLabel("<a
href=\"http://www.google.ru\">google</a>", this);
      mHrefLabel->setAlignment(Qt::AlignCenter);
      //При нажатии на ссылку вызываем обработчик
      connect(mHrefLabel, SIGNAL(linkActivated(QString)),
SLOT(onLabelLinkActivated(QString)));
      //Создаем надпись для имени файла
      mFileNameLabel = new QLabel(this);
      //Создаем текстовый броузер
      mText = new QTextBrowser(this);
      //Создаем прогрессбар
      mProgressBar = new QProgressBar(this);
      //Создаем QLCDNumber. устанавливаем вывод в
шестнадцатиричном виде
      mIndicator = new QLCDNumber(12, this);
      mIndicator->setHexMode();
      //Создаем календарь. первый день недели будет
понедельник
      mCalendar = new QCalendarWidget(this);
      mCalendar->setFirstDayOfWeek(Qt::Monday);
      //Создаем кнопку. по ее нажатии выберем открываемый
файл
      QPushButton* pbOpen = new QPushButton(tr("Открыть
файл"), this);
      connect(pbOpen, SIGNAL(released()),
SLOT(openFile()));
      //Размещаем компоненты на виджете
      QGridLayout* gl = new QGridLayout;
      gl->addWidget(mHrefLabel, 0, 0, 1, 2,
Qt::AlignCenter);
      gl->addWidget(new QLabel(tr("Размер файла"), this),
1, 0);
      gl->addWidget(pbOpen, 1, 1, Qt::AlignRight);
      gl->addWidget(mIndicator, 2, 0, 1, 2);
      gl->addWidget(mProgressBar, 3, 0, 1, 2);
      gl->addWidget(mCalendar, 0, 2, 4, 1);
      gl->addWidget(mFileNameLabel, 4, 0, 1, 3);
      gl->addWidget(mText, 5, 0, 1, 3);
      setLayout(gl);
}
MainWidget::~MainWidget()
{
}
void MainWidget::onLabelLinkActivated(const QString &link)
{
  mText->setText(tr("Мы хотим зайти на сайт %1, но пока не
умеем!").arg(link));
}
```

Листинг 4. Содержимое файла mainwidget.cpp

```
void MainWidget::openFile()
{
  mText->clear();
  //Для выбора имени файла вызываем диалог (см лаб 13)
  QString fileName = QFileDialog::getOpenFileName(this, tr("Открыть файл"),
QDir::homePath(),
tr("Текстовые файлы (*.txt);;"
      "HTML (*.html);;"
      "Все файлы (*)"));
  if(!fileName.isEmpty())
  {
    QFileInfo fi(fileName);//Fi получает информацию о файле
    mIndicator->display((int)fi.size());//на индикатор выводим размер файла
    //В надпись выводим полное имя файла
    mFileNameLabel->setText(fi.absoluteFilePath());
    //Проверяем расширение файла. html и txt будет
    // обрабатывать QTextBrowser стандартными средствами.
    //Другие файлы - выводить в шестнадцатиричном виде
    if(fi.suffix() == "html" || fi.suffix() == "txt")
      mText->setSource(QUrl(fi.absoluteFilePath()));
    else
    {
      QFile f(fi.absoluteFilePath());
      //Открываем файл
      if(f.open(QIODevice::ReadOnly))
      {
        QDataStream ds(&f);//Для чтения файла в бинарном виде
        quint8 byte;
        QString line;
        int cnt = 0;
        //Максимум прогресса - размер файла
        mProgressBar->setMaximum(fi.size());
        //Читаем по одному байту. Каждые 8 байт добавляем 2 пробела
        //Каждые 16 байт - переводим строку.
        while(!ds.atEnd())
        {
          ds >> byte;
          line += QString("%1  ").arg(byte, 2, 16, QChar('0'));
          if(!((cnt + 1) % 8)) line += "  ";
          if(!((cnt + 1) % 16))
          {
            mText->append(line);
            line.clear();
          }
          ++cnt;
          //Текущее значение прогресса - количество обработанных байт
          mProgressBar->setValue(cnt);
          qApp->processEvents();//Это чтоб интерфейс не зависал
        }
        //закрываем файл
        f.close();
      }    }  }}
```

Листинг 4. Содержимое файла mainwidget.cpp (продолжение)

7. Выполните команду «Сборка/Собрать все»

8. Выполните команду «Сборка/Запустить»

9. Воспользовавшись QtAssistant и книгой Жасмин Бланшет «Т 4 программирование GUI на C++», напишите подробные комментарии к каждой строке файлов «mainwidget.h» и «mainwidget.cpp».

10. Покажите результат преподавателю.

11. Выполните отчет по практической работе.

Практическая работа №28.

Тема: «Использование меню, курсоров и пиктограмм»

Цель работы: Изучить на практике средства Qt для создания меню и панелей инструментов

Ход работы.

1. Запустите QtCreator.

2. Откройте проект <QTDIR>/Examples/4.7/mainwindows/mdi/mdi.pro

3. Соберите и запустите проект.

4. Исследуйте содержимое файлов main.cpp, mainwindow.h, mainwindow.cpp.

5. Разберитесь с реализацией меню и панелей инструментов в проекте.

6. Откройте проект <QTDIR>/Examples/4.7/mainwindows/menus/menus.pro

7. Соберите и запустите проект.

8. Исследуйте содержимое файлов main.cpp, mainwindow.h, mainwindow.cpp.

8. Разберитесь с реализацией меню в проекте.

9. Выполните отчет по практической работе, в котором письменно ответьте на следующие вопросы.

Вопросы по практической работе

1. Какой класс Qt реализует панель меню?
2. Какой класс Qt реалезует меню?
3. Какой класс Qt реализует действие (команду) меню?
4. Какой класс Qt реализует панель инструментов?
5. Приведите пример кода для добавления меню на панель меню
6. Приведите пример кода создания действия (команды) меню с именем «Пример» и иконкой example.png, подгружаемой из ресурсов.
7. Приведите пример кода добавления действия в меню.
8. Приведите пример кода для добавления действия на панель инструментов.
9. Приведите пример кода для создания панели инструментов «Моя панель» и добавление ее в главное окно программы.

10. Опишите назначение, основные члены-функции классов QMenuBar, QMenu, QAction, QToolBar;

11. Можно ли добавить на панель инструментов произвольный виджет. Если да, то укажите члены-функции класса QToolBar, предназначенные для этого.

12. Укажите способ сделать элемент меню активным/неактивным.

Практическая работа № 29

Тема: «Использование диалоговых окон с элементами editbox и button»

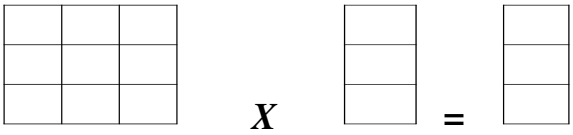
Цель урока:

- Приобрести базовые навыки работы с языком С++.

Задание

1. Умножение матрицы на вектор.

Матрица Вектор Результат



X **=**

и кнопку: ***Выход****.*

2. *Ячейки матрицы и векторов оформить как поля ввода текста (Edit Box), поля 'Результат' сделать доступными только для чтения.*

3. *По нажатию на клавишу* ***X*** *выполнить перемножение матриц.*

Диалоговое окно вызывать из функции WinMain.

Листинг программы:

```
#define STRICT
#define _CRT_SECURE_NO_WARNINGS
#define _CRT_NON_CONFORMING_SWPRINTFS
#include <windows.h>
#include <windowsx.h>
#include <tchar.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include "resource.h"
HINSTANCE g_hApp;
HWND g_hWndMain;
HDC g_hdcMain;
LPCTSTR g_szWndClassName = _T("SomeWindowClass"),
      g_szTitle       = _T
/*****************************************************************/
/*                ПРОТОТИПЫ  ФУНКЦИЙ                    */
/*****************************************************************/
BOOL RegisterWndClasses (void);
void UnregisterWndClasses (void);
```

```
BOOL CreateWindows (void);
LRESULT CALLBACK MainWindowProc (HWND, UINT, WPARAM, LPARAM);
 // --- оконная функция и обработчики сообщений диалогового окна ---
BOOL CALLBACK DlgProc (HWND, UINT, WPARAM, LPARAM);
BOOL DlgOnInitDialog (HWND);
BOOL DlgOnCommand (HWND, int, HWND, UINT);
BOOL DlgOnDestroy ();
// --- обработчики меню ---
BOOL MainOnCommand (HWND, int, HWND, UINT);
BOOL MenuOnOf (HWND hwnd , int   , int  );
BOOL MainOnCreate (HWND);
BOOL MainOnDestroy (HWND);
BOOL InitApp (void);
void UninitApp (void);
int APIENTRY WinMain (HINSTANCE, HINSTANCE, LPSTR, int);
/*****************************************************************/
/*                ГЛОБАЛЬНЫЕ  ПЕРЕМЕННЫЕ                  */
/*****************************************************************/
// --- тип  диалогового окна ---
#define MODAL_DIALOG    1    //  модальное
#define NO_MODAL_DIALOG 0    //  немодальное
int     type_dialog_window=MODAL_DIALOG;
int n,i, a[3][3],
                b[3][1],
                c[3][1];
/*****************************************************************/
/*                RegisterWndClasses                    */
/*                ==================                    */
/*          Регистрация классов окон в БД Windows          */
/*****************************************************************/
BOOL RegisterWndClasses ()
{
// --- инициализация структуры описания класса окон ---
     WNDCLASSEX wc;
     memset (&wc, 0, sizeof(wc));
     wc.cbSize = sizeof(wc);
     wc.hInstance = g_hApp;
     wc.style = CS_VREDRAW | CS_HREDRAW | CS_OWNDC;
     wc.hIcon = LoadIcon(g_hApp, MAKEINTRESOURCE(IDI_ICON1));
     wc.lpfnWndProc = (WNDPROC)MainWindowProc;
     wc.hbrBackground = (HBRUSH)GetStockObject(LTGRAY_BRUSH);
     wc.lpszClassName = g_szWndClassName;
     return RegisterClassEx(&wc);
}  //func RegisterWndClasses
 // --- очистка структуры регистрации классов окон в БД Windows ---
void UnregisterWndClasses ()
{
     UnregisterClass (g_szWndClassName, g_hApp);
```

```
}  //func UnregisterWndClasses
/*****************************************************************/
/*                CreateWindows                        */
/*                =============                        */
/*          Создание основного окна приложения         */
/*****************************************************************/
BOOL CreateWindows ()
{
    g_hWndMain = CreateWindow ( g_szWndClassName, g_szTitle,
            WS_OVERLAPPED | WS_SYSMENU | WS_MINIMIZEBOX |
            WS_MAXIMIZEBOX | WS_THICKFRAME | WS_CLIPCHILDREN,
            370, 150, 600, 400, NULL, NULL, g_hApp, 0);

    if (! g_hWndMain)
            return FALSE;

    return TRUE;
}  //func CreateWindows
/*****************************************************************/
/*                MainWindowProc                       */
/*                ==============                       */
/*         Оконная функция главного  окна              */
/*****************************************************************/
LRESULT CALLBACK MainWindowProc (HWND hwnd, UINT msg, WPARAM wParam,
LPARAM lParam)
{
    switch (msg)
    {
            case WM_CREATE:
                    return MainOnCreate (hwnd);
    // --- сообщения от МЕНЮ ---
            case WM_COMMAND:
                    return MainOnCommand (hwnd, LOWORD(wParam), HWND(lParam),
HIWORD(wParam));

            case WM_DESTROY:
                    return MainOnDestroy (hwnd);
            default:
                    return DefWindowProc (hwnd, msg, wParam, lParam);
    }
    return 0L;
}  //func MainWndProc
/*****************************************************************/
/*                MainOnCommand                        */
/*                =============                        */
/*            обработчик WM_COMMAND                    */
/*****************************************************************/
BOOL MainOnCommand (HWND hwnd, int cmdId, HWND hwndCtrlItem, UINT ntfCode)
```

```
{
   HMENU  hm;
   hm = GetMenu (hwnd);
     switch (cmdId)
     {
    case IDCANCEL:
                        // передача сообщения, требующего закрытия окна
                        SendMessage (hwnd, WM_CLOSE, 0, 0L);
              break;

     }
     return FALSE;
}  //func MainOnCommand
/*****************************************************************/
/*                    DlgProc                          */
/*                    =======                          */
/*          Оконная функция  диалогового окна                */
/*****************************************************************/
BOOL CALLBACK DlgProc (HWND hdlg, UINT msg, WPARAM wParam, LPARAM lParam)
{
     switch (msg)
     {
              case WM_INITDIALOG:
                        return DlgOnInitDialog (hdlg);
              case WM_COMMAND:
                        return DlgOnCommand (hdlg, LOWORD(wParam), HWND(lParam),
HIWORD(wParam));
              case WM_DESTROY:
                        return DlgOnDestroy ();
              default:
                        return FALSE;
     }
     return FALSE;
}  //func DlgProc
/*****************************************************************/
/*                DlgOnInitDialog                        */
/*                ===============                        */
/*             обработчик WM_INITDIALOG                  */
/*****************************************************************/
BOOL DlgOnInitDialog (HWND hdlg)
{
     EnableMenuItem (GetMenu(g_hWndMain), IDC_DIALOG_W, MF_BYCOMMAND |
MF_GRAYED);

     return FALSE;
}  //func DlgOnInitDialog
/*****************************************************************************
*/
```

```
/*                          DlgOnCommand                          */
/*                          ============                          */
/* обработчик WM_COMMAND - команд (или изменения состояния) элементов управления */
/* cmdId        командный идентификатор                          */
/* hwndCtrlItem  дескриптор окна диалогового элемента            */
/* ntfCode      код нотификации (дополнительные данные о происшедшем событии)   */
/*****************************************************************************
*/
BOOL DlgOnCommand (HWND hdlg, int cmdId, HWND hwndCtrlItem, UINT ntfCode)
{
     switch (cmdId)
     {
// клавиша 'OK'
      case IDOK:
                //mass A
                TCHAR buff[300];
                GetDlgItemText(hdlg, IDC_a11,buff,sizeof(buff)/sizeof(buff[0]));
        a[1][1] = _tstof(buff);
        GetDlgItemText(hdlg, IDC_a12,buff,sizeof(buff)/sizeof(buff[0]));
        a[1][2] = _tstof(buff);
        GetDlgItemText(hdlg, IDC_a13,buff,sizeof(buff)/sizeof(buff[0]));
        a[1][3] = _tstof(buff);
        GetDlgItemText(hdlg, IDC_a21,buff,sizeof(buff)/sizeof(buff[0]));
        a[2][1] = _tstof(buff);
        GetDlgItemText(hdlg, IDC_a22,buff,sizeof(buff)/sizeof(buff[0]));
                a[2][2] = _tstof(buff);
        GetDlgItemText(hdlg, IDC_a23,buff,sizeof(buff)/sizeof(buff[0]));
        a[2][3] = _tstof(buff);
        GetDlgItemText(hdlg, IDC_a31,buff,sizeof(buff)/sizeof(buff[0]));
        a[3][1] = _tstof(buff);
        GetDlgItemText(hdlg, IDC_a32,buff,sizeof(buff)/sizeof(buff[0]));
        a[3][2] = _tstof(buff);
        GetDlgItemText(hdlg, IDC_a33,buff,sizeof(buff)/sizeof(buff[0]));
        a[3][3] = _tstof(buff);
                //mass B
        GetDlgItemText(hdlg, IDC_b11,buff,sizeof(buff)/sizeof(buff[0]));
        b[1][1] = _tstof(buff);
        GetDlgItemText(hdlg, IDC_b22,buff,sizeof(buff)/sizeof(buff[0]));
        b[2][1] = _tstof(buff);
        GetDlgItemText(hdlg, IDC_b33,buff,sizeof(buff)/sizeof(buff[0]));
        b[3][1] = _tstof(buff);
       for(n=1; n<=3; n++)
       c[n][1]=0;
                        for(n=1; n<=3; n++)
                        {
                                c[1][1]+=a[1][n]*b[n][1];
                         c[2][1]+=a[2][n]*b[n][1];
                                c[3][1]+=a[3][n]*b[n][1];
```

```
                        }
SetDlgItemInt (hdlg, IDC_c11, c[1][1] , TRUE);
SetDlgItemInt (hdlg, IDC_c22, c[2][1] , TRUE);
SetDlgItemInt (hdlg, IDC_c33, c[3][1] , TRUE);
break;
            case IDCANCEL:
              // закрыть диалог
                    if (type_dialog_window==MODAL_DIALOG)
                          EndDialog (hdlg, IDCANCEL);
                    else
                      DestroyWindow (hdlg);
        return MainOnDestroy (g_hWndMain);
            break;
            default:
                    return FALSE;
     }
     return FALSE;
}  //func DlgOnCommand
/*****************************************************************/
/*              DlgOnDestroy                        */
/*              ============                          */
/*          Удаление диалогового окна                    */
/*****************************************************************/
BOOL DlgOnDestroy ()
{
     EnableMenuItem ( GetMenu(g_hWndMain), IDC_DIALOG_W,
                      MF_BYCOMMAND | MF_ENABLED
                              );
     return FALSE;
}  //func DlgOnDestroy
/*******************************************************************/
/*                  MainOnCreate                        */
/*                  ============                          */
/*                обработчик WM_CREATE                        */
/*******************************************************************/
BOOL MainOnCreate (HWND hwnd)
{
     int i;
      // получить и настроить собственный контекст окна
     g_hdcMain = GetDC(hwnd);
     SetBkMode (g_hdcMain, TRANSPARENT);

     return TRUE;
}  //func MainOnCreate
     /*******************************************************************/
/*                MainOnDestroy                        */
/*                =============                          */
/*                Закрытие окон                        */
```

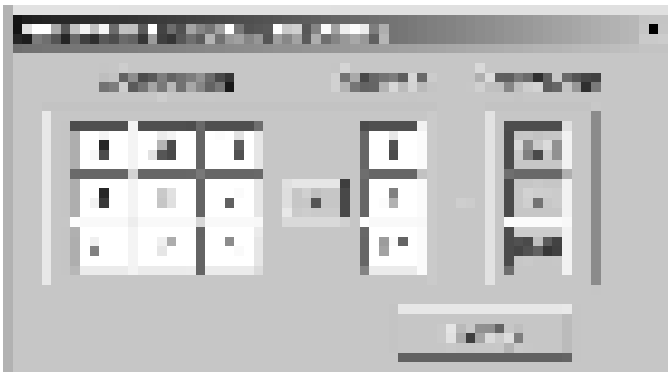
```
/*****************************************************************/
BOOL MainOnDestroy (HWND hwnd)
{

    ReleaseDC (hwnd,g_hdcMain);

    PostQuitMessage (0);
    return TRUE;
}  //func MainOnDestroy
 /*****************************************************************/
/*                InitApp                          */
/*                =============                    */
/*          Инициализация  приложения              */
/*****************************************************************/
BOOL InitApp ()
{
    if (! RegisterWndClasses())
            return FALSE;
    CreateWindows ();
    ShowWindow (g_hWndMain, SW_SHOW);
    UpdateWindow (g_hWndMain);
    return TRUE;
}  //func InitApp
// Удаление приложения из памяти
void UninitApp ()
{
    UnregisterWndClasses ();
}  //func UninitApp
/*****************************************************************/
/*                WinMain                          */
/*                =======                          */
/*             Главная функция                     */
/*****************************************************************/
int APIENTRY WinMain (HINSTANCE hi_crr, HINSTANCE hi_prv, LPSTR pc_cmdLine, int
cmdShow)
{
    MSG msg;
    g_hApp = hi_crr;
    InitApp ();
if(type_dialog_window==MODAL_DIALOG)
                        DialogBox (g_hApp, MAKEINTRESOURCE(IDD_DIALOG1),
g_hWndMain, DlgProc);
                    else
    CreateDialog (g_hApp, MAKEINTRESOURCE(IDD_DIALOG1), g_hWndMain, DlgProc);
    while (GetMessage (&msg, NULL, 0, 0)) {
            // трансляция сообщений нижнего уровня
            // в сообщения верхнего уровня (для клавиатуры)
            // ( WM_KEYDOWN -> WM_CHAR / WM_SYSCHAR )
```

```
            TranslateMessage (&msg);
            DispatchMessage (&msg);
        }
        UninitApp ();
        return msg.wParam;
}  //func WINMAIN
```

Результат:



Практическая работа № 30

Тема: «Установка «Microsoft SQL Server 2008». Создание файла данных и журнала транзакций»

Цель урока:

- научиться устанавливать "Microsoft *SQL Server* 2008".
- научиться создавать файлы данных и журнал транзакций.

Ход работы.

Начнем создание *БД* с установки "Microsoft *SQL Server* 2008". Вставьте *диск* с сервером в привод компакт дисков вашего компьютера. Установка запустится автоматически. *Компьютер* будет проверен на наличие пакета "Microsoft *.NET* Framework". Если данный пакет не установлен, то появится окно начала установки пакета (рис. 1).

Рис 1.

В появившемся окне представлено лицензионное соглашение об использовании пакета "Microsoft *.NET* Framework". Для того чтобы установить пакет необходимо согласиться с соглашением.
Включите *переключатель* "Я прочитал(а) и ПРИНИМАЮ условия лицензионного соглашения" и нажмите кнопку "Установить".

После завершения установки появится окно с сообщением о завершении установки (рис. 2).

Рис. 2.

В данном окне нажмите кнопку "Выход".

Если на вашем компьютере не установлен пакет обновлений для "Windows XP", называемый "KB942288-v2", то появится следующее окно (рис. 3):

Рис. 3.

Для установки обновления нажмите кнопку "Далее". После установки обновления появится вопрос о перезагрузке компьютера (рис. 4).

Рис. 4.

После перезагрузки компьютера появится окно выбора типа установки сервера (рис. 5).

Рис. 5.

Выберите тип установки "New SQL Server stand-alone installation or add to an existing installation". Начнется установка файлов обеспечивающих установку сервера (рис. 2.6).

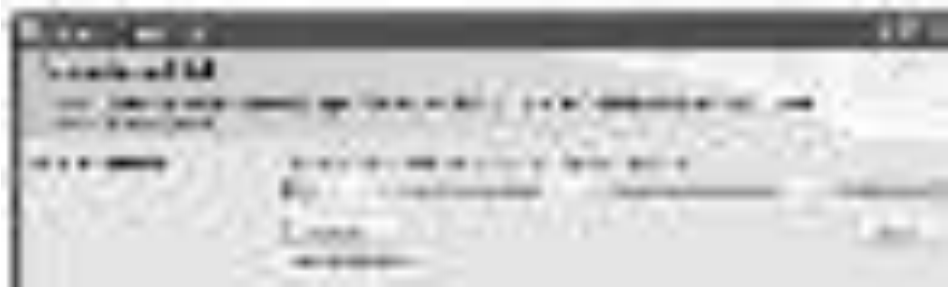


Рис. 6.

После установки вышеперечисленных файлов в окне установки нажмите кнопку "Ok". Появится окно выбора типа лицензии использования, где можно выбрать один из двух видов лицензии:

Specify a free edition - бесплатная версия сервера, работает 180 дней, после чего требует приобретения коммерческой версии;

Enter the product key - коммерческая версия, в поле ввода, расположенного ниже необходимо ввести ключ коммерческой лицензии продукта.

После выбора вида лицензии нажмите кнопку "Next" (Далее). Появится окно, предназначенное для ознакомления с лицензией. Прочитайте лицензионное соглашение, включите опцию "I accept the license terms" (Я согласен с условиями лицензии) и нажмите кнопку "Next" (рис. 7).

Рис. 7.

Появится окно начала установки дополнительных файлов (рис. 8).

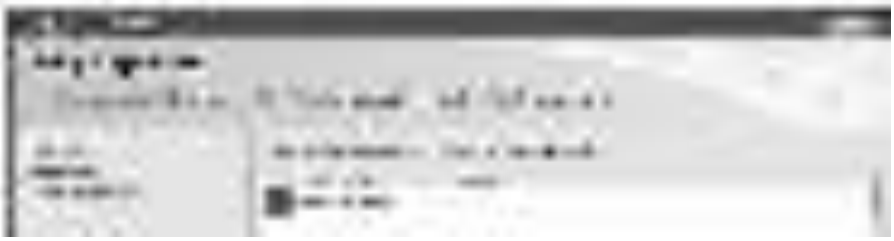


Рис 8.

В окне начала установки дополнительных файлов нажмите кнопку "Install" (Установить). Появится окно определения правил установки дополнительных файлов (рис. 9).

Рис. 9.

В выше представленном окне нажмите кнопку "Next". Появится окно выбора устанавливаемых компонентов сервера (рис. 10).

Рис. 10.

Выберите все компоненты сервера и нажмите кнопку "Next". Появится окно настройки устанавливаемого экземпляра сервера (рис. 11).

Рис. 11.

В данном окне определите следующие настройки:

"Default instance" или "Named instance" - установка экземпляра по умолчанию или поименованного экземпляра;

"Instance ID" - имя устанавливаемого экземпляра сервера;

"Instance root directory" - папка на диске, куда будет установлен экземпляр сервера.

Задайте вышеописанные параметры и нажмите кнопку "Next". Появится окно с отчетом о количестве свободного места на диске, куда устанавливается экземпляр сервера (рис. 12)

Рис. 12.

Нажмите кнопку "Next". Появится окно настройки экземпляра сервера (рис. 2.13).

Рис. 13.

Нажмите кнопку "Next". Появится окно создания учетной записи пользователя сервера (рис. 14).

Рис. 14.

В данном окне оставьте настройки по умолчанию и нажмите кнопку "Ok". Появится окно настройки ядра сервера (рис. 15).

Рис. 15.

В выше представленном окне выберите смешанный режим подключения (Mixed Mode). Задайте пароль (Enter password) и его подтверждение (Confirm password). Добавьте нового администратора сервера, нажав кнопку "Add Current User" (Добавить текущего пользователя) и нажмите кнопку "Next". Появится окно для определения пользователя управляющего всеми службами сервера (рис. 16).

Рис. 16.

Для добавления пользователя нажмите кнопку "Add Current User", а затем нажмите кнопку "Next". Появится окно настройки сервера отчетов (рис. 17).

Рис. 17.

В данном окне можно выбрать один из трех вариантов настройки отчета:

Install the native mode default configuration - установить настройки по умолчанию;

Install the SharePoint integrated mode configuration - настройка отчетов для интеграции их в систему SharePoint.

Install, but do not configure the report server - установить но не настраивать сервер отчетов.

Выберите первый вариант настройки и нажмите кнопку "Next". Появится окно настройки отчетов об ошибках в работе сервера (рис. 18).

Рис. 18.

Если в окне настройки отчета об ошибках включить все опции, то в случае ошибки вся информация об ошибке будет отправлена на сервер разработчика (Microsoft) для анализа. Если вам это необходимо то можете включить все опции, если нет, то не включайте. Нажмите кнопку "Next". Появится окно, отображающее правила установки (рис. 19).

Рис. 19.

В выше представленном окне нажмите кнопку "Next". Появится окно с полным отчетом по установке сервера (рис. 20).

Рис. 20.

Для начала установки нажмите кнопку "Install" (Установить). Начнется процесс установки сервера. Появится окно, отображающее ход установки сервера (рис. 21).

Рис. 21.

После окончания процесса установки нажмите кнопку "Next". Появится окно отчета по результатам установки (рис. 22).

Рис. 22.

Для завершения установки сервера в данном окне нажмите кнопку "Close" (Закрыть).

Создание файла данных и журнала транзакций.

Создание любой БД начинается с создания файла данных. Рассмотрим этот процесс в "Microsoft SQL Server 2008" на примере создания простой БД по учету успеваемости студентов.

Для начала необходимо запустить среду разработки "SQL Server Management Studio". Для этого в меню "Пуск" выбираем пункт"Программы\Microsoft SQL Server 2008\SQL Server Management Studio" (рис. 23).

Рис. 23.

После запуска среды разработки появится окно подключения к серверу "Connect to Server" (рис. 24).

Рис. 24.

В этом окне необходимо нажать кнопку "Connect"

Замечание: Если при установке "Microsoft SQL Server 2008" был задан логин и пароль подключения к серверу, то перед нажатием кнопки "Connect", в выпадающем списке "Authentication" нужно выбрать "SQL Server Authentication", а затем необходимо ввести заданные при установке логин и пароль.

После нажатия кнопки "Connect" появится окно среды разработки "SQL Server Management Studio" (рис. 25).

Рис. 25.

Данное окно имеет следующую структуру (рис. 26):

1. Оконное меню - содержит полный набор команд для управления сервером и выполнения различных операций.

2. Панель инструментов - содержит кнопки для выполнения наиболее часто производимых операций. Внешний вид данной панели зависит от выполняемой операции.

3. Панель "Object Explorer" - обозреватель объектов. Обозреватель объектов - это панель с древовидной структурой, отображающая все объекты сервера, а также позволяющая производить различные операции, как с самим

сервером, так и с БД. Обозреватель объектов является основным инструментом для разработки БД.

4. Рабочая область. В рабочей области производятся все действия с БД, а также отображается ее содержимое.

Замечание: В обозревателе объектов сами объекты находятся в папках. Чтобы открыть папку необходимо щелкнуть по знаку "+"слева от изображения папки.

Теперь перейдем непосредственно к созданию файла данных. Для этого в обозревателе объектов щелкните ПКМ на папке"Databases" (*Базы данных*) (рис. 4.3) и в появившемся *меню* выберите *пункт* "New Database" (Новая *БД*). Появится окно настроек параметров файла данных новой *БД* "New Database" (рис. 26). В левой части окна настроек имеется *список* "Select a page". Этот *список* позволяет переключаться между группами настроек.

Рис. 26.

Для начала настроим основные настройки "General". Для выбора основных настроек нужно просто щелкнуть мышью по пункту"General" в списке "Select a page". В правой части окна "New Database" появятся

основные настройки (рис. 26)

Рассмотрим их более подробно. В верхней части окна расположено два параметра: "Database name" (Имя *БД*) и "Owner"(Владелец).
Задайте *параметр* "Database
name" равным "Students". *Параметр* "Owner" оставьте без изменений.

Под вышеприведенными параметрами в виде таблицы располагаются настройки файла данных и журнала транзакций. *Таблица*имеет следующие столбцы:

- Logical Name - логическое имя файла данных и журнала транзакций. По этим именам будет происходить обращение к вышеприведенным файлам в БД. Можно заметить, что файл данных имеет то же имя что и БД, а имя файла журнала транзакций составлено из имени БД и суффикса "_log".
- File Type - тип файла. Этот параметр показывает, является ли файл файлом данных или журналом транзакций.
- Filegroup - группа файлов, показывает к какой группе файлов относится файл. Группы файлов настраиваются в группе настроек "Filegroups".
- Initial Size (MB) - начальный размер файла данных и журнала транзакций в мегабайтах.
- Autogrowth - автоувеличение размера файла. Как только файл заполняется информацией его размер автоматически увеличивается на величину, указанную в параметре "Autogrowth". Увеличение можно задавать как в мегабайтах так и в процентах. Здесь же можно задать максимальный размер файлов. Для изменения этого параметра надо нажать кнопку "…". В нашем случае (рис. 26) размер файлов не ограничен. Файл данных увеличивается на 1 мегабайт, а файл журнала транзакций на 10%.
- Path - путь к папке, где хранятся файлы. Для изменения этого параметра также надо нажать кнопку "…".
- File Name - имена файлов. По умолчанию имена файлов аналогичны логическим именам. Однако файл данных имеет расширение "mdf", а файл журнала транзакций - расширение "ldf".

Замечание: Для добавления новых файлов данных или журналов транзакций используется кнопка "Add", а для удаления кнопка"Remove".

Теперь перейдем к другим второстепенным настройкам файла данных. Для доступа к этим настройкам необходимо щелкнуть мышью по пункту "Options" в списке "Select a page". Появится следующее окно (рис. 27).

Рис. 27.

В правой части окна мы видим следующие настройки:

- Collation - этот параметр отвечает за обработку текстовых строк, их сравнение, текстовый поиск и т.д. Рекомендуется оставить его как "<server default>". При этом данный параметр будет равен значению, заданному на вкладке "Collation", при установке сервера.
- Recovery Model - модель восстановления. Данный параметр отвечает за информацию, предназначенную для восстановления БД, хранящуюся в файле транзакций. Чем полнее модель восстановления, тем больше вероятность восстановления данных при сбое системы или ошибках пользователей, но и больше размер файла журнала транзакций. При наличии места на диске, рекомендуется оставить этот параметр в значении "Full".

- Compatibility level - уровень совместимости, определяет совместимость файла данных с более ранними версиями сервера. Если планируется перенос данных на другую, более раннюю версию сервера, то ее необходимо указать в этом параметре.
- Other options - второстепенные параметры. Данные параметры являются необязательными для изменения.

В нашем случае все параметры в разделе "Options", рекомендуется оставить как на рис. 27.

Наконец рассмотрим последнюю группу настроек "Filegroups". Данная *группа* настроек отвечает за группы файлов. Для ее отображения в списке "Select a page" необходимо щелкнуть мышью по пункту "Filegroups". Отобразятся настройки групп файлов (рис. 4.6).

Рис. 28.

Группы файлов представлены в таблице "Rows" в правой части окна (рис. 28). Данная *таблица* имеет следующие столбцы:

- Name - имя группы файлов.
- Files - количество файлов входящих в группу.

- Read only - файлы в группе будут только для чтения. То есть, их можно только просматривать, но нельзя изменять.
- Default - группа по умолчанию. Все новые файлы данных будут входить в эту группу.

Замечание: Как и в случае с файлами данных, для добавления новых групп используется кнопка "Add", а для удаления кнопка"Remove".

В рассматриваемой *БД* нет необходимости добавлять новые группы файлов. Поэтому оставим группу настроек "Filegroups" без изменений.

На этом мы заканчиваем настройку свойств наших файлов. Для принятия всех настроек и *создание файла* данных и журнала транзакций нашей *БД* в окне "New Database" нажмем кнопку "Ok".

Произойдет возврат в окно среды разработки "SQL Server Management Studio". На панели обозревателя объектов в папке"Databases" появится новая *БД* "Students" (рис. 29).

Рис. 29.

Замечание: Для переименования *БД* необходимо в обозревателе объектов щелкнуть по ней ПКМ и в появившемся *меню* выбрать*пункт* "Rename". Для удаления в это же *меню* выбираем *пункт* "Delete", для обновления - *пункт* "Refresh", а для

изменения свойств описанных выше - *пункт* "Properties".

Практическая работа № 31

Тема: «Создание и заполнение таблиц»

Цель урока:

- научиться создавать и заполнять таблицы.

Ход работы.

Перейдем теперь к созданию таблиц. Все таблицы нашей *БД* находятся в подпапке "Tables" папки "Students" в окне обозревателя объектов (рис. 1).

Рис. 1.

Создадим таблицу "Специальности". Для этого щелкните ПКМ *по* папке "Tables" и в появившемся *меню* выберите *пункт* "New Table". Появится окно создания новой таблицы (рис. 2).

Рис. 2.

В правой части окна расположена *таблица* определения полей новой таблицы. Данная *таблица* имеет следующие столбцы:

- Column Name - имя поля. Имя поля должно всегда начинаться с буквы и не должно содержать различных специальных символов и знаков препинания. Если имя поля содержит пробелы, то оно автоматически заключается в квадратные скобки.
- Data Type - тип данных поля.
- Allow Nulls - допуск значения Null. Если эта опция поля включена, то в случае незаполнения поля в него будет автоматически подставлено значение Null. То есть, поле необязательно для заполнения.

Замечание: Под таблицей определения полей располагается *таблица* свойств выделенного поля "Column Properties". В данной таблице настраиваются свойства выделенного поля. Некоторые из них будут рассмотрены ниже.

Перейдем к созданию полей и настройке их свойств. В таблице определения полей задайте значения столбцов "Column Name","Data Type" и "Allow Nulls", как показано на рисунке ниже (рис. 3).

Рис. 3.

Из рис. 3 следует, что наша *таблица* "Специальности" имеет три поля:

- Код специальности - числовое поле для связи с таблицей студенты,
- Наименование специальности - текстовое поле, предназначенное для хранения строк, имеющих длину не более 50 символов.
- Описание специальности - текстовое поле, предназначенное для хранения строк, имеющих неограниченную длину.

Замечание: Так как, *поле* "Код специальности" будет являться первичным полем связи в запросе, связывающем таблицы"Студенты" и "Специальности". То мы должны сделать его числовым счетчиком. То есть данное *поле* должно автоматически заполняться числовыми значениями. Более того, оно должно быть ключевым.

Сделаем *поле* "Код специальности" счетчиком. Для этого выделите *поле*, просто щелкнув *по* нему мышкой в таблице определения полей. В таблице свойств поля отобразятся свойства поля "Код специальности". Разверните группу свойств"Identity Specification" (Настройка особенности). Свойство "(Is Identity)" (Особенное) установите в *значение* "Yes" (Да). Задайте свойства "Identity Increment" (Увеличение особенности, шаг счетчика) и "Identity Seed" (Начало особенности,

начальное *значение* счетчика) равными 1 (рис. 3). Эти настройки показывают, что *значение* поля "Код специальности" у первой записи в таблице будет равным 1, у второй - 2, у третьей 3 и т.д.

Теперь сделаем *поле* "Код специальности" *ключевым полем*. Выделите *поле*, а затем на панели инструментов нажмите кнопку с изображением ключа

В таблице определения полей, рядом с полем "Код специальности" появиться изображение ключа, говорящее о том, что *поле*ключевое.

На этом настройку таблицы "Специальности" можно считать завершенной. Закройте окно создания новой таблицы, нажав кнопку закрытия

в верхнем правом углу окна, над таблицей определения полей. Появиться окно с запросом о сохранении таблицы (рис. 4).

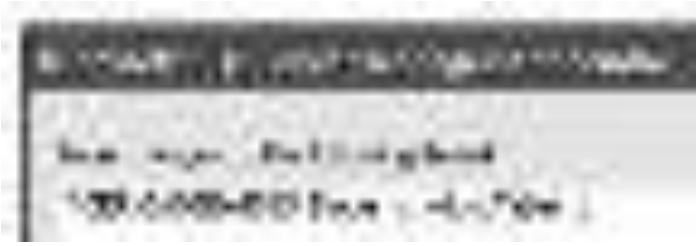


Рис. 4.

В этом окне необходимо нажать "Yes" (Да). Появиться окно "Choose Name" (Задайте имя), предназначенное для определения имени новой таблицы (рис. 6.5).

Рис. 5.

В этом окне задайте имя новой таблицы как "Специальности" и

нажмите кнопку "Ok". *Таблица* "Специальности" отобразиться в обозревателе объектов в папке "Tables" *БД* "Students" (рис. 6).

Замечание: В обозревателе объектов *таблица* "Специальности" отображается как "dbo.Специальности". *Префикс* "dbo" обозначает, что *таблица* является объектом *БД* (*Data Base Object*). В дальнейшем при работе с объектами *БД префикс* "dbo" можно опускать.

Теперь перейдем к созданию таблицы "Предметы". Как и в случае с таблицей "Специальности" щелкните ПКМ *по* папке"Tables" и в появившемся *меню* выберите *пункт* "New Table". Создайте поля представленные на рисунке ниже (рис. 6).

Рис. 6.

Сделайте *поле* "Код предмета" числовым счетчиком и *ключевым полем*, как это было сделано в таблице "Специальности". Закройте окно создания новой таблицы. В появившемся окне "Chose Name" задайте имя "Предметы" (рис. 7).

Рис. 7.

Таблица "Предметы" появится в папке "Tables" в обозревателе объектов

(рис. 8).

После создания таблицы "Предметы" создайте таблицу "Студенты". Создайте новую **таблицу аналогичную таблице представленной на рис. 8.**

Рис. 8.

Рассматривая поля новой таблицы можно придти к следующим выводам:

- Поле "Код студента" - это первичное поле для связи с таблицей оценки. Следовательно, данное поле необходимо сделать числовым счетчиком и ключевым (см. создание таблицы "Специальности" выше);
- Поля "ФИО", "Пол", "Родители", "Адрес", "Телефон", "Паспортные данные" и "Группа" являются текстовыми полями различной длины (для задания длины выделенного текстового поля необходимо в таблице свойств выделенного поля установить свойство Length равное максимальному количеству знаков текста вводимого в поле);
- Поля "Дата рождения" и "Дата поступления" предназначены для хранения дат. Поэтому они имеют тип данных "date";
- Поле "Очная форма обучения" является логическим полем. В "*Microsoft SQL Server 2008*" такие поля должны иметь тип данных "bit";
- Поля "Номер зачетки" и "Курс" являются целочисленными. Единственным отличием является размер полей. Поле "Номер зачет-

ки" предназначено для хранения целых чисел в диапазоне $-2^{63}\ldots+2^{63}$ (тип данных "bigint"). Поле "Курс " предназначено для хранения целых чисел в диапазоне 0…255 (тип данных "tinyint");

- Поле "Код специальности" - это поле связи с таблицей "Специальности". Однако, данное поле связи является вторичным, поэтому его можно сделать просто целочисленным, то есть, "bigint".

После определения полей таблицы "Студенты", закройте окно создания новой таблицы. В появившемся окне "Chose Name"задайте имя новой таблицы как "Студенты" (рис. 9).

Рис. 9.

Таблица "Студенты" появится в папке "Tables" в обозревателе объектов (рис. 10).

Наконец, создадим таблицу "Оценки". Создайте поля, представленные на рис. 10.

Рис. 10.

Таблица "Оценки" не имеет первичных полей связи. Следовательно, эта *таблица* не имеет ключевых полей. Поля "Код предмета 1", "Код предмета 2" и "Код предмета 3" являются вторичными полями связи,

предназначенными для связи с таблицей "Предметы", поэтому они являются *целочисленными (тип* данных "bigint"). Поля "Дата экзамена 1", "Дата экзамена 2" и "Дата экзамена 3" предназначены для хранения дат (*тип данных* "*date*"). Поля "Оценка 1", "Оценка 2", и "Оценка 3"предназначены для хранения оценок. Задайте *тип данных* для этого поля "tinyint". Наконец, *поле* "Средний балл" хранит дробные числа и имеет тип "*real*".

Закройте окно создания новой таблицы, задав *имя таблицы* как "Оценки" (рис. 11).

Рис. 11.

На этом мы заканчиваем создание таблиц *БД* "Students". После создания всех таблиц окно обозревателя объектов будет выглядеть так (рис. 12):

Рис. 12.

Теперь рассмотрим операцию заполнения таблиц начальными данными.

Для начала заполним таблицу "Специальности". Для заполнения этой

таблицы в обозревателе объектов щелкните правой кнопкой мыши *по* таблице "Специальности" (рис. 12) и в появившемся *меню* выберите *пункт* "Edit Top 200 Rows"(Редактировать первые 200 записей.). В рабочей области "*Microsoft SQL Server Management* Studio" проявится окно заполнения таблиц. Заполните таблицу "Специальности", как показано на рис. 13.

Рис. 13.

Замечание: Заполнение таблиц происходит полностью аналогично табличному процессору "Microsoft *Excel* 2000".

Замечание: Так как *поле* "Код специальности" является первичным полем связи и ключевым числовым счетчиком, то оно заполняется автоматически (заполнять его не нужно).

Закройте окно заполнения таблицы "Специальность" щелкнув *по* кнопке закрытия окна

в верхнем правом углу, над таблицей.

После заполнения таблицы "Специальности" заполним таблицу "Предметы". Откройте ее для заполнения как описано выше, и заполните, как показано на рис. 14.

Рис. 14.

Закройте окно заполнения таблицы "Предметы" и перейдите к заполнению таблицы "Студенты". Откройте таблицу"Студенты" для

заполнения и заполните ее как показано ниже (рис. 15).

Рис. 15.

Замечание: Для заполнения дат в качестве разделителя можно использовать знак ".". Даты можно заполнять в формате "день.месяц.год".

Замечание: *Поле* "Код специальности" является вторичным полем связи (для связи с таблицей "Специальности"). Следовательно, значения этого поля необходимо заполнять значениями поля "Код специальности" таблицы "Специальности". В нашем случая это значения от 1 до 5 (рис. 13). Если у Вас коды специальностей в таблице "Специальности" имеют другие значения, то внесите их в таблицу "Студенты".

По окончании заполнения, закройте окно заполнения таблицы "Студенты".

Наконец заполним таблицу "Оценки", как это показано на рис. 16.

Рис. 16.

Замечание: Поля с датами заполняются, как и в таблице "Студенты" (см. выше).

Замечание: Поля "Код предмета 1", "Код предмета 2" и "Код предмета 3" являются вторичными полями связи с таблицей"Предметы". Поэтому они должны быть заполнены значениями поля "Код предмета из этой таблицы",

то есть значениями от 1 до 5 (см. рис. 14).

Закройте окно заполнения таблицы "Оценки". На этом мы заканчиваем создание и заполнение таблиц нашей *БД* "Students".

Практическая работа № 32.

Тема: Создание запросов и фильтров.

Цель: научиться создавать запросы и фильтры.

Перейдем к созданию статических запросов. В обозревателе объектов "*Microsoft SQL Server 2008*" все запросы *БД* находятся в папке "Views" (рис. 1).

Рис. 1.

Создадим *запрос* "Запрос Студенты+Специальности", связывающий таблицы "Студенты" и "Специальности" *по* полю связи"Код специальности". Для создания нового запроса необходимо в обозревателе объектов в *БД* "Students" щелкнуть ПКМ *по*папке "Views", затем в появившемся *меню* выбрать *пункт* "New View". Появится окно "Add Table" (Добавить таблицу), предназначенное для выбора таблиц и запросов, участвующих в новом запросе (рис. 2).

Рис. 2.

Добавим в новый *запрос* таблицы "Студенты" и "Специальности". Для этого в окне "Add Table" выделите таблицу "Студенты"и нажмите кнопку "Add" (Добавить). Аналогично добавьте таблицу "Специальности". После добавления таблиц участвующих в запросе закройте окно "Add Table" нажав кнопку "Close" (Закрыть). Появится окно конструктора запросов (рис. 3).

Рис. 3.

Замечание: Окно конструктора запросов состоит из следующих панелей:

1. Схема данных - отображает поля таблиц и запросов, участвующих в запросе, позволяет выбирать отображаемые поля, позволяет устанавливать связи между участниками запроса по специальным полям связи. Эта панель включается и выключается следующей кнопкой на панели инструментов

2. Таблица отображаемых полей - показывает отображаемые поля (столбец "Column"), позволяет задавать им псевдонимы (столбец "Alias"), позволяет устанавливать тип сортировки записей по одному или нескольким полям (столбец "Sort Type"), позволяет задавать порядок сортировки (столбец "Sort Order"), позволяет задавать условия отбора записей в фильтрах (столбцы "Filter" и "Or…"). Также эта таблица позволяет менять порядок отображения полей в запросе. Эта панель включается и выключается следующей кнопкой на панели инструментов

3. Код SQL - код создаваемого запроса на языке T-SQL. Эта панель включается и выключается следующей кнопкой на панели инструментов

4. Результат - показывает результат запроса после его выполнения. Эта панель включается и выключается следующей кнопкой на панели инструментов

Замечание: Если необходимо снова отобразить окно "Add Table" для добавления новых таблиц или запросов, то для этого на панели инструментов "Microsoft SQL Server 2008" нужно нажать кнопку

Замечание: Если необходимо удалить таблицу или *запрос* из схемы данных, то для этого нужно щелкнуть ПКМ и в появившемся*меню* выбрать *пункт* "Remove" (Удалить).

Теперь перейдем к связыванию таблиц "Студенты" и "Специальности" *по* полям связи "Код специальности". Чтобы создать*связь* необходимо в схеме данных перетащить мышью *поле* "Код специальности" таблицы "Специальности" на такое же *поле*таблицы "Студенты". *Связь* отобразится в виде ломаной линии соединяющей эти два поля связи (рис. 3).

Замечание: Если необходимо удалить *связь*, то для этого необходимо щелкнуть *по* ней ПКМ и в появившемся *меню* выбрать*пункт* "Remove".

Замечание: После связывания таблиц (а также при любых изменениях в запросе) в области кода Т-*SQL* будет отображаться Т-*SQL*код редактируемого запроса.

Теперь определим поля, отображаемые при выполнении запроса. Отображаемые поля обозначаются галочкой (слева от имени поля) на схеме данных, а также отображаются в таблице отображаемых полей. Чтобы сделать *поле* отображаемым при выполнении запроса необходимо щелкнуть мышью *по* пустому квадрату (слева от имени поля) на схеме данных, в квадрате появится галочка.

Замечание: Если необходимо сделать *поле* невидимым при выполнении запроса, то нужно убрать галочку, расположенную слева от имени поля на схеме данных. Для этого просто щелкните мышью *по* галочке.

Замечание: Если необходимо отобразить все поля таблицы, то необходимо установить галочку слева от пункта "* (All Columns)"(Все поля), принадлежащего соответствующей таблице на схеме данных.

Определите отображаемые поля нашего запроса, как это показано на рис. 3 (Отображаются все поля кроме полей с кодами, то есть полей связи).

На этом настройку нового запроса можно считать законченной. Перед сохранением запроса проверим его работоспособность, выполнив его. Для запуска запроса на панели инструментов нажмите кнопку

Либо щелкните ПКМ в любом месте окна конструктора запросов и в

появившемся *меню* выберите *пункт* "Execute SQL"(Выполнить *SQL*). Результат выполнения запроса появиться в виде таблицы в области результата (рис. 3).

Замечание: Если после выполнения запроса результат не появился, а появилось *сообщение об ошибке*, то в этом случае проверьте, правильно ли создана *связь*. Ломаная *линия связи* должна соединять поля "Код специальности" в обеих таблицах. Если *линия связи* соединяет другие поля, то ее необходимо удалить и создать заново, как это описано выше.

Если *запрос* выполняется правильно, то необходимо сохранить. Для сохранения запроса закройте окно конструктора запросов, щелкнув мышью *по* кнопке закрытия

расположенной в верхнем правом углу окна конструктора (над схемой данных). Появится окно с вопросом о сохранении запроса (рис. 4).

Рис. 4.

В данном окне необходимо нажать кнопку "Yes" (Да). Появится окно "Choose Name" (Выберите имя) (рис. 5).

Рис. 5.

В данном окне зададим имя нового запроса "Запрос Студенты+Специальности" и нажмем кнопку "Ok". *Запрос* появится в папке "Views" *БД* "Students" в обозревателе объектов (рис. 6).

Рис. 6.

Проверим работоспособность созданного запроса вне конструктора запросов. Запустим вновь созданный *запрос* "Запрос Студенты+Специальности" без использования конструктора запросов. Для выполнения уже сохраненного запроса необходимо щелкнуть ПКМ *по* запросу и в появившемся *меню* выбрать *пункт* "Select top 1000 rows" (Отобразить первые 1000 записей). Выполните эту операцию для запроса "Запрос Студенты+Специальности". Результат представлен на рис. 6.

Перейдем к созданию запроса "Запрос Студенты+Оценки". В обозревателе объектов в *БД* "Students" щелкните ПКМ *по* папке"Views", затем в появившемся *меню* выберите *пункт* "New View". Появиться окно "Add Table" (рис. 2).

В запросе "Запрос Студенты+Оценки" мы связываем таблицы "Студенты" и "Оценки" *по* полям связи "Код студента". Следовательно, в окне "Add Table" в новый *запрос* добавляем таблицы "Студенты" и "Оценки". Более того, в данном запросе*таблица* "Оценки" связывается с таблицей "Предметы" не *по* одному полю, а *по* трем полям. То есть поля "Код предмета 1","Код предмета 2" и "Код предмета 3" таблицы "Оценки" связаны с полем "Код предмета" таблицы "Предметы". *По* этому добавим в *запрос* три экземпляра таблицы "Предметы" (*по* одному экземпляру для каждого поля связи таблицы оценки). В итоге в запросе должны участвовать таблицы "Студенты", "Оценки" и три экземпляра таблицы "Предметы" (в запросе они будут называться "Предметы", "Предметы_1" и "Предметы_2"). После добавления таблиц закройте окно "Add Table", появится окно конструктора запросов.

В окне конструктора запросов установите связи между таблицами и определите отображаемые поля, как показано на рис. 7.

Рис. 7.

Теперь поменяем порядок отображаемых полей в запросе, для этого в

таблице отображаемых полей необходимо перетащить поля мышью вверх или вниз за заголовок строки таблицы (столбец перед столбцом "Column"). Расположите отображаемые поля в таблице отображаемых полей как показано на рис. 8.

Рис. 8.

Задайте псевдонимы для каждого из полей, просто записав псевдонимы в столбце "Alias" таблицы отображаемых полей, как на рис. 8.

Проверьте работоспособность нового запроса, выполнив его. Обратите внимание на то, что реальные названия полей были заменены их псевдонимами. Закройте окно конструктора запросов. В появившемся окне "Choose Name" задайте имя нового запроса "Запрос Студенты+Оценки" (рис. 9).

Рис. 9.

Проверьте работоспособность нового запроса вне конструктора. Для этого запустите *запрос*. Результат выполнения запроса"Запрос Студенты+Оценки" должен выглядеть как на рис. 10.

Рис. 10.

На этом мы заканчиваем рассмотрение обычных запросов и переходим к созданию фильтров.

На основе запроса "Запрос Студенты+Специальности" создадим фильтры, отображающие студентов отдельных специальностей. Создайте новый *запрос*. Так как он будет основан на запросе "Запрос Студенты+Специальности", то в окне"Add Table" перейдите на вкладку "Views" и добавьте в новый *запрос* "Запрос Студенты+Специальности" (рис. 11). Затем закройте окно "Add Table".

Рис. 11.

В появившемся окне конструктора запросов определите в качестве отображаемых полей все поля запроса "Запрос Студенты+Специальности" (рис. 12).

Рис. 12.

Замечание: Для отображения всех полей запроса, в данном случае, мы не можем использовать *пункт* "* (All Columns)" (Все поля). Так как в этом случае мы не можем устанавливать *критерий отбора* записей в фильтре, а

также невозможно установить сортировку записей.

Теперь установим *критерий отбора* записей в фильтре. Пусть наш фильтр отображает только студентов имеющих специальность "ММ". Для определения условия отбора записей в таблице отображаемых полей в строке, соответствующей полю, на которое накладывается условие, в столбце "Filter", необходимо задать условие. В нашем случае условие накладывается на *поле*"Наименование специальности". Следовательно, в строке "Наименование специальности", в столбце "Filter" нужно задать следующее условие отбора "='ММ'" (рис. 12).

В заключение настроим сортировку записей в фильтре. Пусть при выполнении фильтра сначала происходит *сортировка* записей *по*возрастанию *по* полю "Очная форма обучения", а затем *по* убыванию *по* полю "Курс". Для установки сортировки записей *по*возрастанию, в таблице определяемых полей, в строке для поля "Очная форма обучения", в столбце "Sort Type" (Тип сортировки), задайте "Ascending" (*По* возрастанию), а в строке для поля "Курс" - задайте "Descending" (*По* убыванию). Для определения порядка сортировки для поля "Очная форма обучения" в столбце "Sort Order" (Порядок сортировки) поставьте 1, а для поля "Курс" поставьте 2 (рис. 12). То есть, при выполнении запроса записи сначала сортируются *по* полю "Очная форма обучения", а затем *по* полю "Курс".

Замечание: После установки условий отбора и сортировки записей на схеме данных напротив соответствующих полей появятся специальные значки. Значки

и

обозначают сортировку *по* возрастанию и убыванию, а значок

показывает наличие условия отбора.

После установки сортировки записей в фильтре проверим его работоспособность, выполнив его. Результат выполнения фильтра должен выглядеть как на рис. 12. Закройте окно конструктора запросов. В качестве имени нового фильтра в окне "Choose Name" задайте "Фильтр ММ" (рис. 13) и нажмите кнопку "Ok".

Рис. 13.

Фильтр "Фильтр ММ" появится в обозревателе объектов. Выполните созданный фильтр вне окна конструктора запросов. Результат должен быть таким же как на рис. 14.

Рис. 14.

Самостоятельно создайте фильтры для отображения других специальностей. Данные фильтры создаются аналогично фильтру"Фильтр ММ" (смотри выше). Единственным отличием является условие отбора,

накладываемое на *поле* "Наименование специальности", оно должно быть не "='ММ'", а "='ПИ'", "='СТ'", "='МО'" или "='БУ'". При сохранении фильтров задаем их имена соответственно их условиям отбора, то есть "Фильтр ПИ", "Фильтр СТ", "Фильтр МО" или "Фильтр БУ". Проверьте созданные фильтры на работоспособность.

Теперь на основе запроса "Запрос Студенты+Специальности" создадим фильтры, отображающие студентов имеющих отдельных родителей. Для начала создадим фильтр для студентов, из родителей только "Отец". Создайте новый *запрос* и добавьте в него *запрос* "Запрос Студенты+Специальности" (рис. 11). После закрытия окна "Add Table" сделайте отображаемыми все поля запроса (рис. 15).

Рис. 15.

В таблице отображаемых полей в строке для поля "Родители", в столбце "Filter", задайте условие отбора равное "='Отец'". Проверьте работу фильтра, выполнив его. В результате выполнения фильтра окно конструктора запросов должно выглядеть как на рис. 15.

Закройте окно конструктора запросов. В окне "Choose Name" задайте имя нового фильтра как "Фильтр Отец" (рис. 16).

Рис. 16.

Выполните фильтр "Фильтр Отец" вне конструктора запросов. Результат должен быть аналогичен рис. 17.

Рис. 17.

Создайте фильтры для отображения студентов с другими вариантами родителей. Данные фильтры создаются аналогично фильтру"Фильтр Отец" (смотри выше). Единственным отличием является условие отбора, накладываемое на *поле* "Родители", оно должно быть не "='Отец'", а "='Мать'", "='Отец, Мать'" или "='Нет'". При сохранении фильтров задаем их имена соответственно их условиям отбора, то есть "Фильтр Мать", "Фильтр Отец и Мать" или "Фильтр Нет родителей". Проверьте созданные фильтры на работоспособность.

Наконец создадим фильтры для отображения студентов очной и заочной формы обучения. Начнем с очной формы обучения. Создайте

новый *запрос* и добавьте в него *запрос* "Запрос Студенты+Специальности". Как и ранее сделайте все поля запроса отображаемыми (рис. 18).

Рис. 18.

В таблице отображаемых полей в столбце "Filter", в строке для поля "Очная форма обучения" установите условие отбора равное "=1"

Замечание: *Поле* "Очная форма обучения" является логическим полем, оно может принимать значения либо "True" (*Истина*), либо "False" (*Ложь*). В качестве синонимов этих значений в "*Microsoft SQL Server 2008*" можно использовать 1 и 0 соответственно.

Установите сортировку *по* возрастанию, *по* полю курс, задав в строке для этого поля, в столбце "Sort Type", *значение*"Ascending".

Проверьте работу фильтра, выполнив его. После выполнения фильтра окно конструктора запросов должно выглядеть точно также как на рис. 18.

Закройте окно конструктора запросов. Сохраните фильтр под именем "Фильтр очная форма обучения" (рис.19).

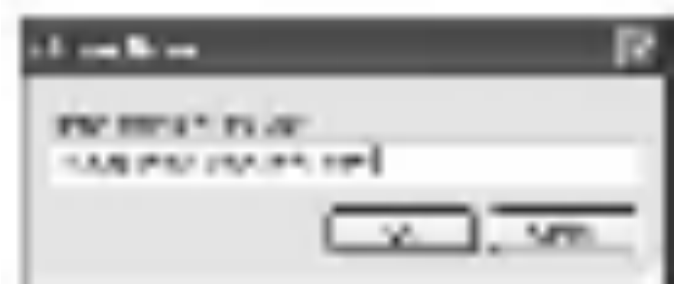


Рис. 19.

После появления фильтра "Фильтр очная форма обучения" в обозревателе объектов выполните фильтр вне окна конструктора запросов. Результат выполнения фильтра "Фильтр очная форма обучения" представлен на рис. 20.

Рис. 20.

Самостоятельно создайте фильтр для отображения студентов заочной формы обучения. Данный фильтр создается точно также как и фильтр "Фильтр очная форма обучения". Единственным отличием является условие отбора, накладываемое на *поле*"Очная форма обучения", оно должно быть не "=1", а "=0". При сохранении фильтра задайте его имя как "Фильтр заочная форма обучения". Проверьте созданный фильтр на работоспособность.

В итоге, после создания всех запросов и фильтров окно обозревателя

объектов должно выглядеть следующим образом (рис. 21):

Рис. 21.

Практическая работа № 33.

Тема: Работа с хранимыми процедурами.

Цель: научиться работать с хранимыми процедурами.

Перейдем к *созданию хранимых процедур*. Для работы с хранимыми процедурами в обозревателе объектов необходимо выделить папку "Programmability/Stored Procedures" *базы данных* "Students" (рис. 1).

Рис. 1.

Создадим процедуру, вычисляющую среднее трех чисел. Для создания новой хранимой процедуры щелкните ПКМ по папке"Stored Procedures" (рис. 10.1) и в появившемся *меню* выберите *пункт* "New Stored Procedure". Появиться окно кода новой хранимой процедуры (рис. 2).

Рис. 2.

Хранимая процедура имеет следующую структуру (рис. 2):

1. Область настройки параметров синтаксиса процедуры. Позволяет настраивать некоторые синтаксические правила, используемые при наборе кода процедуры. В нашем случае это:

   - SET ANSI_NULLS ON - включает использование значений NULL (Пусто) в кодировке ANSI,
   - SET QUOTED_IDENTIFIER ON - включает возможность использования двойных кавычек для определения идентификаторов;

2. Область определения имени процедуры (Procedure_Name) и параметров передаваемых в процедуру (@Param1, @Param2). Определение параметров имеет следующий синтаксис:

   @<Имя параметра> <Тип данных> = <Значение по умолчанию>

   Параметры разделяются между собой запятыми;

3. Начало *тела процедуры*, обозначается служебным словом "BEGIN" ;

4. *Тело процедуры*, содержит команды языка программирования запросов T-SQL;

5. Конец *тела процедуры*, обозначается служебным словом "END".

Замечание: В коде зеленым цветом выделяются комментарии. Они не обрабатываются сервером и выполняют функцию пояснений к коду. Строки комментариев начинаются с подстроки "--". Далее в коде, мы не будем отображать комментарии, они будут свернуты. Слева от раздела с комментариями будет стоять знак "+", щелкнув по которому можно развернуть комментарий.

Наберем код процедуры вычисляющей среднее трех чисел, как это показано на рис. 3.

Рис. 3.

Рассмотрим код данной процедуры более подробно (рис. 3):

1. *CREATE PROCEDURE* [Среднее трех величин] - определяет имя создаваемой процедуры как "Среднее трех величин";

2. @Value1 Real = 0, @Value2 Real = 0, @Value3 Real = 0 - определяют три параметра процедуры Value1, Value2 иValue3. Данным параметрам можно присвоить дробные числа (Тип данных Real), значения по умолчанию равны 0;

3. SELECT 'Среднее значение'=(@Value1+@Value2+@Value3)/3 - вычисляет среднее и выводит результат с подписью "Среднее значение".

Остальные фрагменты кода рассмотрены выше (рис. 2).

Для *создания процедуры*, выполним вышеописанный код, нажав кнопку

(Выполнить) на панели инструментов. В нижней части окна с кодом появиться сообщение "Command(s) completed successfully.". Закройте окно с кодом, щелкнув мышью по кнопке закрытия

расположенной в верхнем правом углу окна с кодом процедуры.

Проверим работоспособность созданной хранимой процедуры. Для запуска хранимой процедуры необходимо создать новый пустой*запрос*, нажав на кнопку

(Новый *запрос*) на панели инструментов. В появившемся окне с пустым запросом наберите команду *EXEC* [Среднее трех величин] 1, 7, 9 и нажмите кнопку  на панели инструментов (рис. 10.4).

Рис. 4.

В нижней части окна с кодом появится результат выполнения новой хранимой процедуры: Среднее значение 5,66667 (рис. 4).

Теперь создадим хранимую процедуру для отбора студентов из таблицы студенты по их "ФИО". Для этого создайте новую хранимую процедуру, как это описано выше, и наберите код новой процедуры как на рис. 5.

Рис. 5.

Рассмотрим код процедуры "Отображение студентов по ФИО" более подробно (рис. 5):

1. *CREATE PROCEDURE* [Отображение студентов по ФИО] - определяет имя создаваемой процедуры как "Отображение студентов по ФИО";
2. @FIO Varchar(50)='' - определяют единственный параметр процедуры FIO. Параметру можно присвоить текстовые строки переменной длины, длинной до 50 символов (Тип данных Varchar(50)), значения по умолчанию равны пустой строке;
3. SELECT * FROM dbo.Студенты WHERE ФИО=@FIO - отобразить все поля (*) из таблицы студенты (dbo.Студенты), где значение поля ФИО равно значению параметра FIO (ФИО=@FIO).

Выполним вышеописанный код и закроем окно с кодом, как описано выше.

Проверим работоспособность созданной хранимой процедуры. Создайте новый пустой *запрос*. В появившемся окне с пустым запросом наберите команду *EXEC* [Отображение студентов по ФИО] 'Иванов А.И.' и нажмите кнопку

на панели инструментов (рис. 6).

Рис. 6.

В нижней части окна с кодом появиться результат *выполнения хранимой процедуры* "Отображение студентов по ФИО" (рис. 6).

Теперь перейдем к более сложной задаче - отобразить студентов, у которых средний балл выше заданного. Создайте новую хранимую процедуру и наберите код новой процедуры как на рис. 7**.**

Рис. 7.

Рассмотрим код процедуры "Отображение студентов по среднему баллу" более подробно (рис. 7):

1. *CREATE PROCEDURE* [Отображение студентов по среднему баллу] - определяет имя создаваемой процедуры как "Отображение студентов по среднему баллу";

2. @*Grade* Real=0 - определяют параметр процедуры Grade. Параметру можно присвоить дробные числа (Тип данных Real), значения по умолчанию равны 0;

3. SELECT * FROM [Запрос Студенты+Оценки] WHERE ([Оценка первого экзамена]+[Оценка второго экзамена]+[Оценка третьего экзамена])/3>@*Grade* - отобразить все поля (*) из запроса "Запрос Студенты+Оценки" (Запрос Студенты+Оценки), где средний балл больше чем значение параметра Grade (([Оценка первого экзамена]+[Оценка второго экзамена]+[Оценка третьего экзамена])/3>@Grade).

Выполним вышеописанный код и закроем окно с кодом, как описано выше. Проверим, как работает *запрос*, описанный выше. Для этого, создайте новый *запрос* и в нем наберите команду *EXEC* [Отображение студентов по среднему баллу] 3.5 и выполните ее (Смотри выше) (рис. 8).

Рис. 8.

В нижней части окна с кодом появиться результат *выполнения хранимой*

процедуры "Отображение студентов по среднему баллу" (рис. 8).

В заключение решим более сложную задачу - *отображение* студентов старше заданного возраста. При чем возраст будет автоматически вычисляться в зависимости от даты рождения.

Создадим новую хранимую процедуру и наберем код новой процедуры как представлено на рис. 9.

Рис. 9.

Рассмотрим код создаваемой процедуры "Отображение студентов по возрасту" более подробно (рис. 9):

1. *CREATE PROCEDURE* [Отображение студентов по возрасту] - определяет имя создаваемой процедуры как "Отображение студентов по возрасту";

2. @Age int=0 - определяют параметр процедуры Age. Параметру можно присвоить целые числа (Тип данных int), значения по умолчанию равны 0;

3. ФИО, [Запрос Студенты+Специальности].[Дата рождения], 'Возраст'=DATEDIFF(yy,[Запрос Студенты+Специальности].[Дата рождения], GETDATE()) - отображает из запроса "Запроса Студенты+Специальности" (FROM [Запрос Студенты+Специальности]) поля "ФИО" (ФИО) и "Дата рождения" ([Запрос Студенты+Специальности].[Дата рождения]), а также отображает возраст студента

('Возраст') в годах (yy), вычисленный исходя из его даты рождения и текущей даты (DATEDIFF(yy,[Запрос Студенты+Специальности].[Дата рождения], GETDATE())). Более того, выводятся студенты возраст которых больше определенного в параметре "Age" (DATEDIFF(yy,[Запрос Студенты+Специальности].[Дата рождения], GETDATE())>@Age).

Замечание: Встроенная *функция* DATEDIFF вычисляющая количество периодов между двумя датами, имеет следующий*синтаксис*: DATEDIFF(<период>,<начальная дата>, <конечная дата>)

Выполним код запроса "Отображение студентов по возрасту", а затем закроем окно с кодом, как описано выше. Проверим, как работает *запрос*. Для этого, создадим новый *запрос* и в нем наберем команду *EXEC* [Отображение студентов по возрасту] 26и выполните ее. Должен появиться результат аналогичный результату, представленному на рис. 10.

Рис. 10.

На этом мы заканчиваем описание хранимых процедур и переходим к рассмотрению *пользовательских функций*. В итоге, обозреватель объектов должен иметь вид как на рис. 11.

Рис. 11.

Практическая работа №34.

Тема: Создание в «Microsoft SQL Server 2008» пользовательских функций.

Цель: научиться работать с пользовательскими функциями.

Теперь рассмотрим создание и применение *пользовательских функций*. В *БД* "*Microsoft SQL Server 2008*" все *пользовательские функции* находятся в папке "Functions" расположенной в папке "Programmability" в обозревателе объектов (рис. 1).

Рис. 1.

Начнем с создания скалярных *пользовательских функций*. Для создания новой скалярной пользовательской функции в обозревателе объектов, в *БД* "Students", в папке "Programmability", щелкните ПКМ *по* папке "Functions" и в появившемся*меню* выберите *пункт* "New/Scalar-valued Function". Появится окно новой скалярной пользовательской функции (рис. 2)

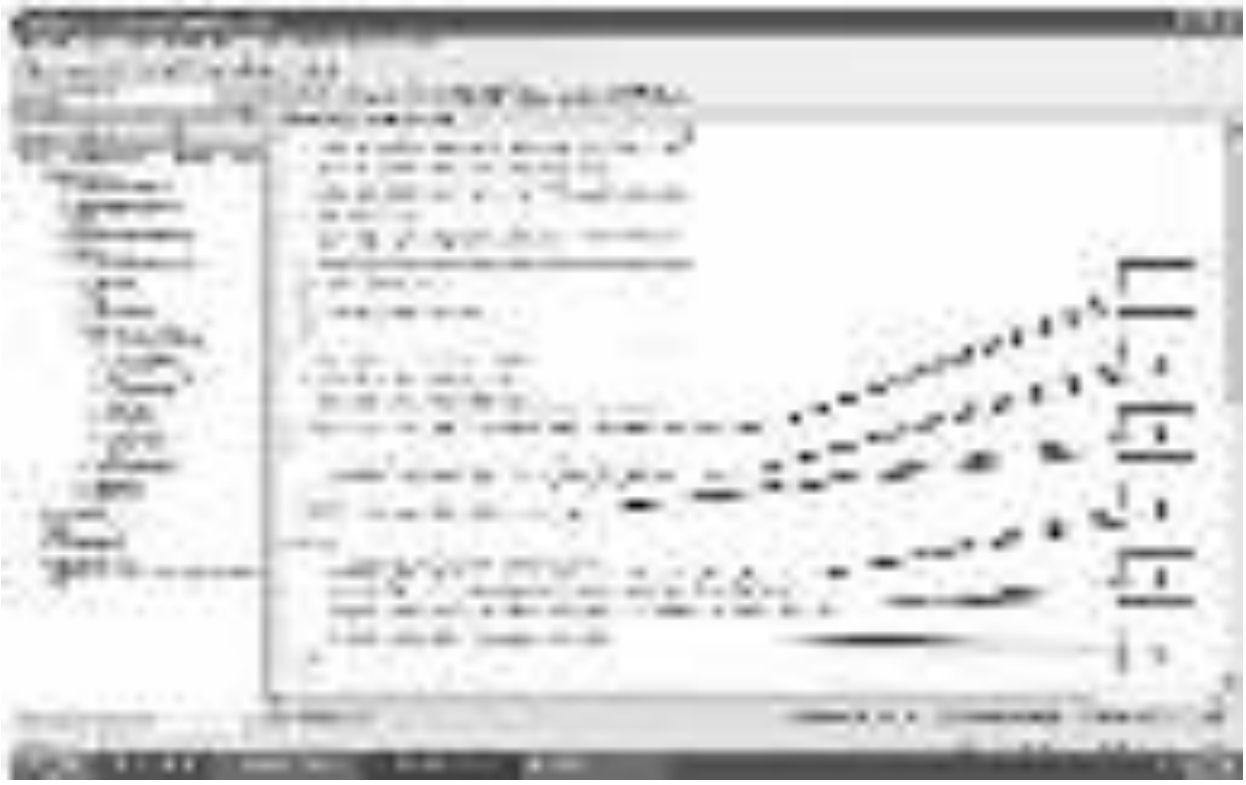


Рис. 2.

Синтаксис скалярной пользовательской функции похож на *синтаксис* хранимой процедуры Однако имеется ряд существенных отличий (рис. 2):

1. Область определения имени функции (Inline_Function_Name);
2. Параметры, передаваемые в процедуру (@Param1). Определение параметров аналогично определению параметров в хранимой процедуре
3. Тип данных значения возвращаемого процедурой;
4. Область объявления переменных, используемых внутри функции. Объявление переменных имеет следующий синтаксис:

DECLARE @<Имя переменной> <Тип данных>

5. Тело самой пользовательской функции, содержит команды языка программирования запросов T-SQL;
6. Команда RETURN возвращающая результат выполнения функции. Имеет следующий синтаксис:

RETURN @<Имя переменной с результатом>

Переменная должна быть того же типа данных, который был указан в пункте 3.

Создадим скалярную пользовательскую функцию, вычисляющую среднее трех величин. В окне новой пользовательской функции наберите код представленный на рис. 3.

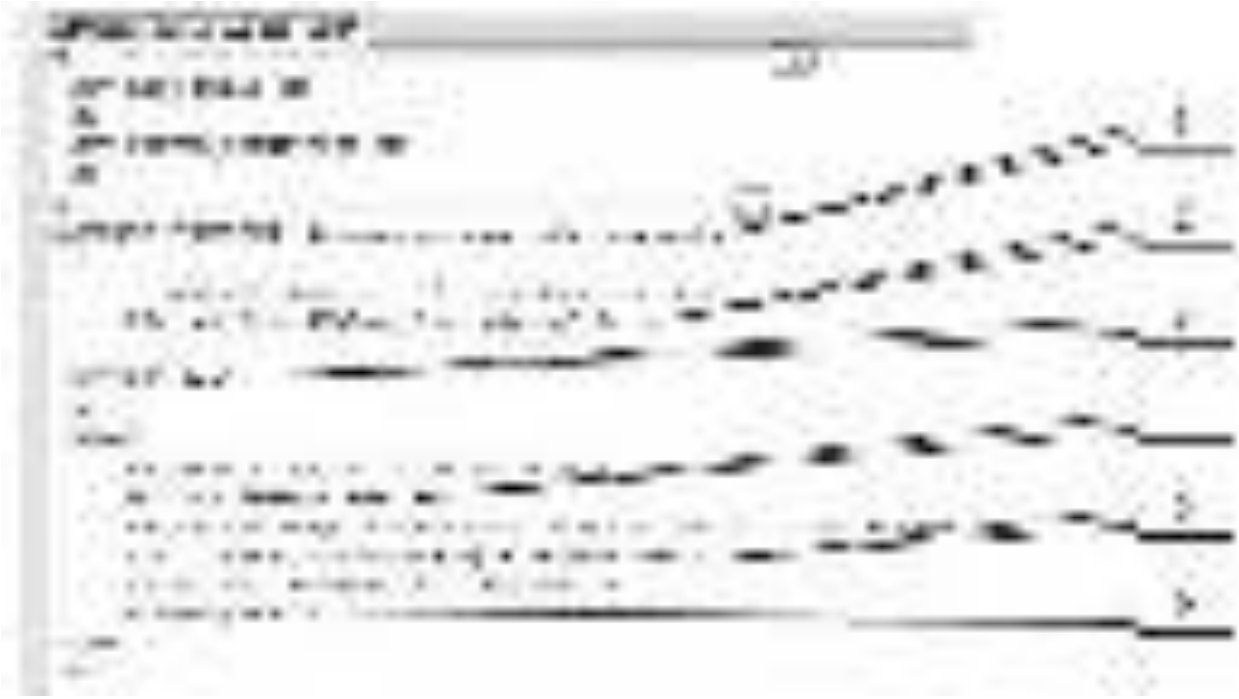


Рис. 3.

Рассмотрим более подробно код данной скалярной пользовательской функции (рис. 3):

1. *CREATE FUNCTION* [Функция средних трех величин] - определяет имя создаваемой функции как "Функция средних трех величин";
2. @Value1 Real, @Value2, @Value3 - определяют три параметра процедуры Value1, Value2 и Value3. Данным параметрам можно присвоить целые числа (Тип данных Int);
3. RETURNS Real - показывает, что функция возвращает дробные числа (Тип данных Real);
4. DECLARE @Result Real - объявляется переменная @Result для хранения результата работы функции, то есть дробного числа (Тип данных Real);
5. SELECT @Result=(@Value1+@Value2+@Value3)/3 - вычисляет среднее и помещает результат в переменную @Result ;
6. RETURN @Result - возвращает значение переменной @Result.

Остальные фрагменты кода рассмотрены выше (рис. 2).

Для *создания функции*, выполним вышеописанный код, нажав кнопку (Выполнить) на панели инструментов. В нижней части окна с кодом появиться сообщение "Command(s) completed successfully.". Закройте окно с кодом, щелкнув мышью *по* кнопке закрытия расположенной в верхнем правом углу окна с кодом функции.

Проверим работу созданной скалярной пользовательской функции. Для запуска пользовательской функции необходимо создать новый пустой *запрос*, нажав на кнопку (Новый *запрос*) на панели инструментов. В появившемся окне с пустым запросом наберите команду SELECT dbo.[Функция средних трех величин] (3, 5, 4) и нажмите кнопку на панели инструментов (рис. 4).

Рис. 4.

В нижней части окна с кодом появится результат выполнения новой скалярной пользовательской функции: 4 (рис. 4).

Теперь создадим более сложную скалярную пользовательскую функцию, предназначенную для определения последнего дня месяца

введенной даты.

Создайте новую скалярную пользовательскую функцию, так как об этом сказано выше. В окне новой пользовательской функции наберите следующий код (рис. 5):

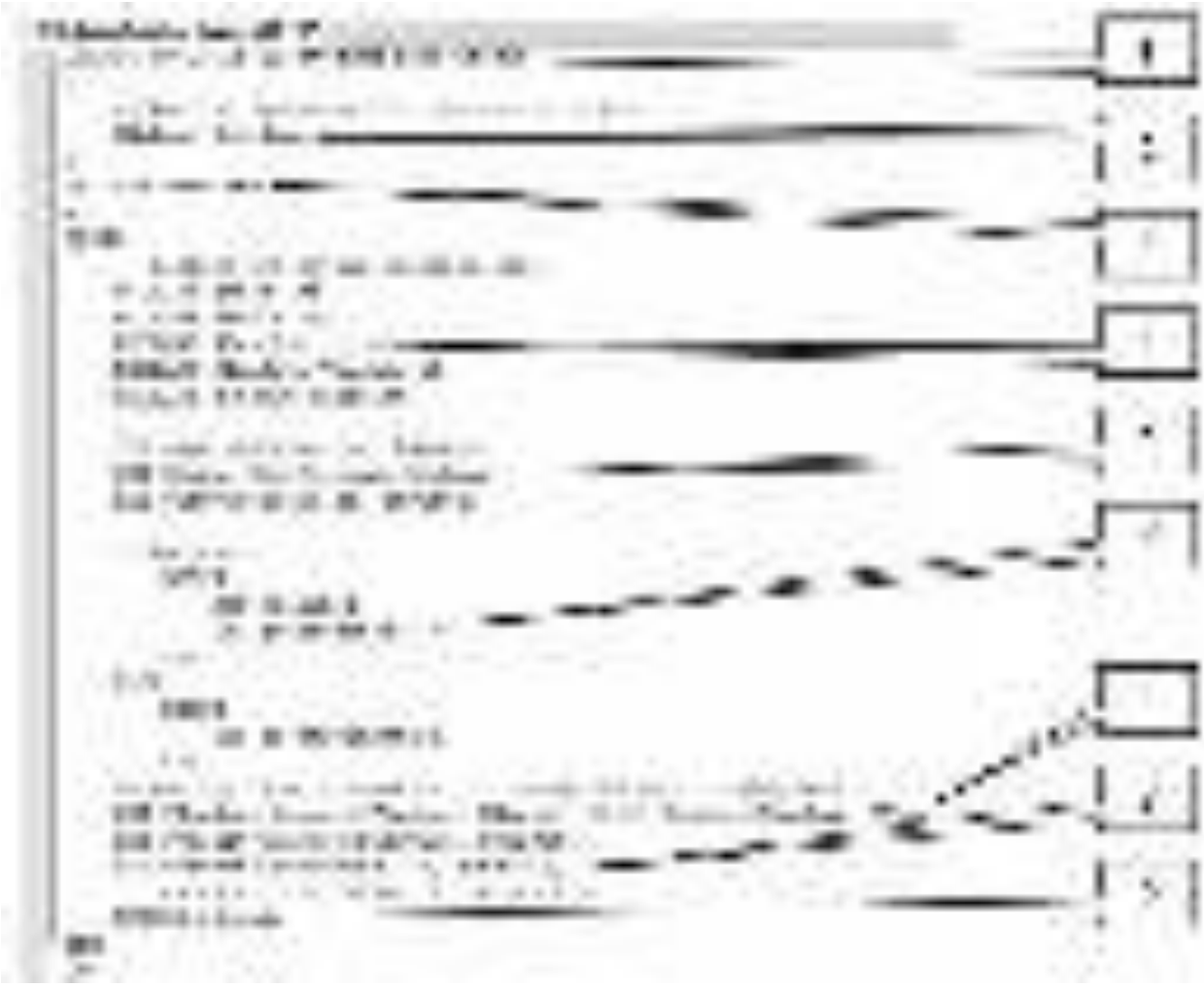


Рис. 5.

Перейдем к рассмотрению вышеприведенного кода (рис. 5). Код состоит из следующих групп команд:

1. *CREATE FUNCTION* [Последний день месяца] - определяет имя создаваемой функции как "Последний день месяца";

2. @MyDate - определяют параметр процедуры MyDate. Параметру можно присвоить значения дат или времени (Тип данных DateTime);

3. RETURNS DateTime - показывает, что функция возвращает дату или время (Тип данных DateTime);

4. DECLARE @Year Int, DECLARE @Month Int, DECLARE @Day Int - объявляются переменные @Year, @Month и @Daydля хранения целочисленных значений года, месяца и дня введенной даты (Тип данных Int).

DECLARE @TmpDate VarChar(10) объявляет переменную "TmpDate" для хранения промежуточного значения даты в строке длинной до 10 символов (Тип данных VarChar(10)).

DECLARE @Result DateTime объявляет переменную "Result" для хранения результата - даты последнего дня месяца (Тип данных DateTime).

5. SET @Year=DatePart(yy, @MyDate), SET @Month=DatePart(mm, @MyDate), SET @Day=DatePart(dd, @MyDate) - определяются части введенной даты и помещаются в переменныне @Year, @Month и @Day. Для определения частей даты используется функция DatePart, имеющая следующий синтаксис: DatePart(<часть даты>, <дата>). Здесь "часть даты" - это закодированная специальными символами определяемая часть даты (yy - год, mm - месяц, dd - день), "дата" - это дата, части которой определяем.

```
IF @Month=12
        BEGIN
                SET @Month=1
                SET @Year=@Year+1
        END
ELSE
        BEGIN
                SET @Month=@Month+1
        END
```

Вышепреведенный фрагмент кода выполняет следующие действия: Если номер месяца равен 12 то установить номер месяца (@Month) равным 1 и увеличить год (@Year) на 1, иначе увеличить месяц на 1.

6. SET @TmpDate=Convert(Varchar, @Month)+'/01/'+Convert(Varchar, @Year), SET @Result=Convert(DateTime, @TmpDate) - переводит числовые значения даты в дату в строковом формате и записывает ее в переменную @TmpDate, затем переводит дату в строковом формате в тип данных даты и времени и помещает ее в переменную @Result. Для конвертации используется функция Convert, имеющая следующий синтаксис:

Convert(<тип данных>, <значение>), здесь "тип данных" это тип

данных в который переводится "значение".

7. SET @Result=DateAdd(dd, -1, @Result) - из даты, хранимой в переменной @Result вычитается 1 день, для этого используется функция DateAdd, имеющая следующий синтаксис:

DateAdd(<часть даты>, <количество периодов>, <дата>) - здесь "часть даты" - это закодированная специальными символами определяемая часть даты (см. функцию DatePart), "количество периодов" - это количество частей даты прибавляемой к введенной дате (параметр "дата").

8. RETURN @Result - возвращает значение, хранимое в переменной @Result.

Для *создания функции*, выполним вышеописанный код, как и в случае с предыдущей функцией, нажав кнопку

После появления сообщения "Command(s) completed successfully." закройте окно с кодом.

Проверим работу функции "Последний день месяца" выполнив ее. Создайте новый пустой *запрос*, затем в окне с пустым запросом наберите команду SELECT dbo.[Последний день месяца] ('12/07/08') и нажмите кнопку

на панели инструментов (рис. 6).

Рис. 6.

Появится результат выполнения новой скалярной пользовательской функции: 2008-12-31 (рис. 6).

Теперь перейдем к созданию табличных *пользовательских функций*. Для создания табличной пользовательской функции в обозревателе объектов, в *БД* "Students", в папке "Programmability",
щелкните ПКМ *по* папке "Functions" и в
появившемся*меню* выберите *пункт* "New/Table-valued Function". Появится окно новой табличной пользовательской функции (рис. 7)

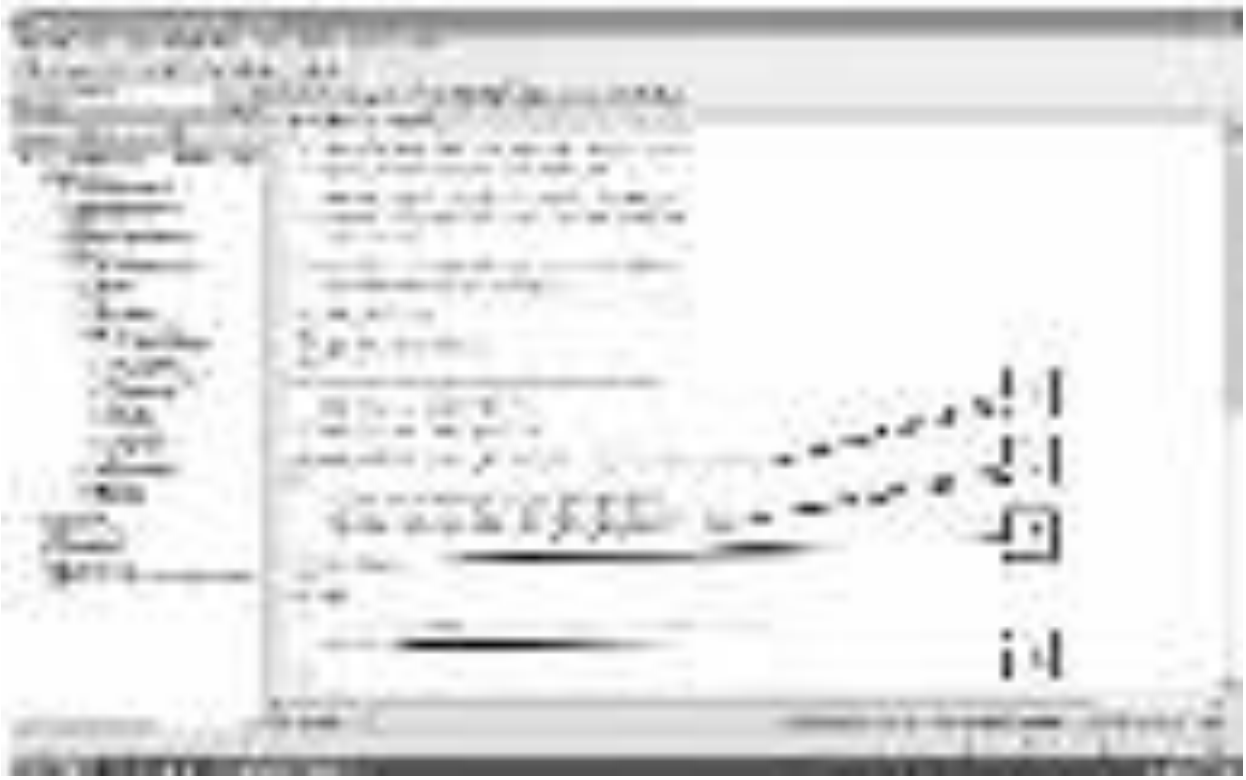


Рис. 7.

Рассмотрим структуру кода табличной пользовательской функции. Табличная пользовательская *функция* состоит из следующих разделов:

1. Область определения имени функции (Inline_Function_Name);
2. Параметры, передаваемые в процедуру (@Param1, @Param2);
3. RETURNS TABLE показывает что функция является табличной, то есть возвращает таблицу;
4. Тело самой пользовательской функции, состоит из команды SELECT языка программирования запросов T-SQL.

Остальные *разделы* табличной пользовательской функции аналогичны таким же разделам хранимых процедур и скалярных*пользовательских функций*.

В заключение рассмотрим создание табличной пользовательской функции "Функция отбора по возрасту", вычисляющих текущий возраст студентов в зависимости от их даты рождения. В окне новой пользовательской функции (рис. 7) наберите следующий код (рис. 8):

Рис. 8.

Из кода представленного на рис. 8 видно, что данная табличная *функция* не имеет параметров и реализуется командой

SELECT ФИО, [Дата рождения], Возраст = DateDiff(yy, [Дата рождения], GetDate())

FROM Студенты

Из вышепредставленной команды видно, что из таблицы "Студенты" отображаются поля "ФИО" и "Дата рождения", а также вычислимое *поле* "Возраст". *Поле* "Возраст" вычисляется при помощи встроенной функции DateDiff вычисляющей различие между датами в определенных единицах измерения (частях даты) и имеющей следующий *синтаксис*:

DateDiff(<часть даты>, <начальная дата>, <конечная дата>).

Здесь "часть даты" - это закодированные специальными символами единицы измерения (часть даты) (yy - год, mm - месяц, dd - день), "начальная дата" - дата начала периода и "конечная дата" - дата конца периода. В нашем случае в качестве начальной даты берем дату рождения студента, а в качестве конечной даты берем текущую дату (*функция* GetDate()).

Для *создания функции*, выполним вышеописанный код, как и в случае с предыдущей функцией. После появления сообщения"Command(s) completed

successfully." закройте окно с кодом.

Проверим работоспособность новой табличной пользовательской функции. Создайте новый пустой *запрос*, затем в окне с пустым запросом наберите команду SELECT * FROM dbo.[Функция отбора по возрасту]() и нажмите кнопку

на панели инструментов (рис. 9).

Рис. 9.

В нижней части окна появиться *таблица* с фамилиями, датами рождения и возрастом студентов на данный момент времени (рис. 9).

Замечание: Обратите внимание на тот факт, что мы работаем с табличной функцией как с обыкновенной таблицей.

На этом мы заканчиваем рассмотрение *пользовательских функций* и переходим к рассмотрению целостности данных, диаграмм и триггеров. *По* окончании выполнения главы 6 обозреватель объектов будет иметь следующий вид (рис.10):

Рис. 10.

Практическая работа №35.

Тема: Создание диаграмм и триггеров, применяемых для обеспечения целостности данных.

Цель: научиться создавать диаграммы и триггеры.

Перейдем теперь к созданию диаграмм.

В *БД* "Microsoft *SQL Server* 2008" все диаграммы находятся в папке "Database Diagrams"обозревателя объектов (рис. 1).

Рис. 1.

Создадим диаграмму, обеспечивающую *целостность данных* нашей *БД* "Students". Для создания новой диаграммы в *БД*"Students" щелкните ПКМ по папке "Database Diagrams" и в появившемся *меню* выберем *пункт* "New Database Diagram". Сначала появится окно с вопросом о добавлении нового объекта "Диаграмма". В этом окне нужно нажать кнопку "Yes". Затем появится окно "Add Table" предназначенное для добавления таблиц в новую диаграмму (рис. 2).

Рис. 2.

В окне добавления таблиц выделите все таблицы нашей *БД* и нажмите кнопку "Add" (рис. 2). Закройте окно "Add Table"нажатием на кнопку "Close".

Появится окно диаграммы, где будут отображены отобранные таблицы. Теперь необходимо определить связи между таблицами.

Перетащите *поле* "Код специальности" из таблицы "Специальности" на такое же *поле* в таблице "Студенты". Появится окно создания связи между таблицами "Tables and Columns" (рис. 3).

Рис. 3.

В окне создания связи нажмите кнопку "Ok". Появится окно настройки свойств связи "Foreign Key Relationship" (рис. 4).

Рис. 4.

Оставьте свойства связи без изменений и в окне свойств связи нажмите кнопку "Ok".В диаграмме между таблицами "Студенты" и"Специальности" появится *связь* в виде ломанной линии (рис. 5).

Аналогичным образом создайте *связь* таблицы "Студенты" с таблицей "Оценки", перетащив *поле* "Код студента" из таблицы"Студенты" на одноименное *поле* в таблице "Оценки". Затем, свяжите таблицы "Предметы" и "Оценки", перетащив *поле*"Код предмета" из таблицы "Предметы" на поля "Код предмета 1", "Код предмета 2" и "Код предмета 3" таблицы"Оценки". После выполнения вышеперечисленных действий *диаграмма* примет следующий вид (рис. 5).

Рис. 5.

Закройте окно с диаграммой, щелкнув мышью по кнопке закрытия

расположенной в верхнем правом углу окна с диаграммой. Появится окно с вопросом о сохранении новой диаграммы, где необходимо нажать кнопку "Yes" (рис. 6).

Рис. 6.

Появится окно определения имени новой диаграммы "Choose Name". В окне определения имени, задайте имя диаграммы как "*Диаграмма БД* Студенты" и нажмите кнопку "Ok" (рис. 7).

Рис. 7.

Появится окно "Save" с запросом сохранения таблиц, входящих в диаграмму. В данном окне необходимо нажать кнопку "Yes" (рис. 8).

Рис. 8.

Перейдем к созданию триггеров. Создадим триггеры для таблицы "Студенты". Триггеры создаются отдельно для каждой таблицы и располагаются в обозревателе объектов в папке "Triggers". В нашем случае, *папка* "Triggers" входит в состав таблицы"Студенты" (рис. 9).

Рис. 9.

Для начала создадим *триггер*, выводящий сообщение "Запись добавлена" при добавлении записи в таблицу "Студенты". Создадим новый *триггер*, щелкнув ПКМ по папке "Triggers" в таблице "Студенты" и в появившемся *меню* выбрав *пункт* "New Trigger". Появится следующее окно с новым триггером (рис. 10):

Рис. 10.

Рассмотрим структуру триггеров:

1. Область определения имени функции (Trigger_Name);
2. Область, показывающая для какой таблицы создается триггер (Table_Name);
3. Область, показывающая когда выполнять триггер (INSERT - при создании записи в таблице, DELETE - при удалении и UPDATE- при изменении) и как его выполнять (AFTER - после выполнения операции, INSTEAD OF - вместо выполнения операции);
4. Тело триггера, содержит команды языка программирования запросов T-SQL.

В окне нового триггера наберите код как показано на рис. 11.

Рис. 11.

Из рис. 11 видно, что создаваемый *триггер* "Индикатор добавления" выполняется после добавления записи (AFTER INSERT) в таблицу "Студенты" (ON dbo.Студенты). После добавления записи *триггер* выведет на экран сообщение "*Запись* добавлена" (PRINT 'Запись добавлена'). Выполните набранный код, нажав кнопку

на панели инструментов. В нижней части окна с кодом появится сообщение "Command(s) completed successfully.".

Проверим, как работает новый *триггер*. Создайте новый пустой *запрос* и в нем наберите следующую команду для добавления новой записи в таблицу "Студенты" (рис. 12):

Рис. 12.

Выполните набранную команду, нажав кнопку

на панели инструментов. В таблицу будет добавлена новая *запись*, и *триггер* выведет сообщение "Запись добавлена" (рис. 12).

Теперь создадим *триггер* отображающий сообщение "Запись изменена". Создайте новый *триггер*, как в предыдущем случае. В окне нового триггера наберите следующий код (рис. 13):

Рис. 13.

Из рис. 14.13 видно, что новый *триггер* "Индикатор изменения" выполняется после изменения записи (AFTER UPDATE) в таблице "Студенты" (ON dbo.Студенты). После изменения записи *триггер* выведет на экран сообщение "*Запись* изменена" (PRINT 'Запись изменена'). Выполните набранный код. В нижней части окна с кодом появится сообщение "Command(s) completed successfully.".

Проверим работоспособность созданного триггера. Создайте новый *запрос* и в нем наберите команду, представленную на рис. 14.

Рис. 14.

Выполните набранную команду, нажав кнопку

на панели инструментов. В таблицу будет добавлена новая *запись*, и *триггер* выведет сообщение "Запись изменена" (рис. 14).

Для полноты картины создадим *триггер*, выводящий сообщение при удалении записи из таблицы "Студенты". Создайте новый*триггер* и в нем наберите код, показанный на рис. 15.

Рис. 15.

Создаваемый *триггер* "Индикатор удаления" выполняется после удаления записи (AFTER DELETE) из таблицы студенты (ON dbo.Студенты). После удаления записи *триггер* выводит сообщение "*Запись* удалена" (PRINT 'Запись удалена').

Выполните код, представленный рис. 15. В нижней части окна с кодом появится сообщение "Command(s) completed successfully.".

Проверим работу триггера "Индикатор удаления" удалив созданную ранее *запись* из таблицы "Студенты". Для этого создайте новый *запрос* и в нем наберите следующую команду (рис. 16):

Рис. 16.

Выполните вышеприведенную команду. После удаления записи *триггер* "Индикатор удаления" отобразит сообщение "Запись удалена" (рис. 16).

В заключение рассмотрим пример применения триггеров для обеспечения целостности данных. Создадим *триггер* "Удаление студента", который при удалении записи из таблицы студенты сначала удаляет все связанные с ней записи из таблицы "Оценки", а затем удаляет саму *запись* из таблицы "Студенты", тем самым обеспечивается *целостность данных*.

Создайте новый *триггер* и в нем наберите следующий код (рис. 17):

Рис. 17.

Создаваемый *триггер* "Удаление студента" выполняется вместо удаления записи (INSTEAD OF DELETE) из таблицы "Студенты" (ON dbo.Студенты).

Замечание: При срабатывании триггера вместо удаления записи создается временная константа Deleted, содержащая *имя таблицы* из которой должно было быть произведено удаление.

После срабатывания триггера из таблицы "Оценки" удаляется *запись*, у которой *значение* поля "Код студента" равно значению такого же поля у удаляемой записи из таблицы "Студенты". Эту операцию выполняют следующие команды:

DELETE dbo.Оценки FROM Deleted

WHERE Deleted.[Код студента] = Оценки.[Код студента]

Затем удаляется *запись* из таблицы "Студенты", которую удаляли до срабатывания триггера. Удаление выполняется следующими командами:

DELETE dbo.Студенты

FROM Deleted

WHERE Deleted.[Код студента] = Студенты.[Код студента]

Выполните код, представленный на рис. 17. В нижней части окна с кодом появиться сообщение "Command(s) completed successfully.".

Проверим, как работает *триггер* "Удаление студента". Для этого создайте новый *запрос* и в нем наберите следующий код (рис. 18):

Рис. 18.

При срабатывании триггера сначала из таблицы "Оценки" удалятся все связанные с удаляемой записью записи, а затем удаляется сама удаляемая *запись* из таблицы "Студенты", при этом сохраняется *целостность данных*.

Замечание: Хотелось бы заметить, что без использования триггера "Удаление студента" нам бы не удалось удалить *запись* из таблицы "Студенты". *Команда* удаления была бы заблокирована диаграммой "Диаграмма БД Студенты" во избежание нарушения целостности данных.

На этом мы завершаем работу с диаграммами и триггерами. После выполнения всех вышеописанных действий обозреватель объектов будет

иметь следующий вид (рис. 19):

Рис. 19.