Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Российский государственный профессионально-педагогический университет»
Институт инженерно-педагогического образования
Кафедра информационных систем и технологий

УЧЕБНЫЙ МОДУЛЬ «ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ»

Выпускная квалификационная работа бакалавра
по направлению 44.03.04 Профессиональное обучение (по отраслям)
профиля «Энергетика»
профилизация «Компьютерные технологии автоматизации и управления»

Идентификационный номер ВКР: 123

Екатеринбург 2016

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Российский государственный профессионально-педагогический университет»
Институт инженерно-педагогического образования
Кафедра информационных систем и технологий

К ЗАЩИТЕ ДОПУСКАЮ
Заведующий кафедрой ИС
_____ Н. С. Толстова
«__» _____ 2016 г.

УЧЕБНЫЙ МОДУЛЬ «ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ»

Выпускная квалификационная работа бакалавра
по направлению 44.03.04 Профессиональное обучение (по отраслям)
профиля «Энергетика»
профилизация «Компьютерные технологии автоматизации и управления»
Идентификационный номер ВКР: 123

Исполнитель:
студент группы КТэ-401 К. В. Отрутикова
Руководитель:
ст. преподаватель Т. П. Телепова
Нормоконтролер: Т. В. Рыжкова

Екатеринбург 2016

РЕФЕРАТ

Пояснительная записка к выпускной квалификационной работе выполнена на 54 страницах, содержит 3 таблицы, 16 рисунков, 30 источник.

Ключевые слова: ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ, ОБЪЕКТ, КЛАСС, НАСЛЕДОВАНИЕ, ИНКАПСУЛЯЦИЯ.

Объект исследования — процесс обучения основам объектно-ориентированного программирования.

Предмет исследования — методические материалы, пособия, учебники, лабораторные работы по теме.

Цель исследования — разработать учебный модуль по теме «Основы объектно-ориентированного программирования».

Для достижения поставленной цели в работе были решены следующие задачи:

1. Рассмотрены теоретические основы объектно-ориентированного программирования.
2. Разработана структура учебного модуля.
3. Отобрано содержание теоретического раздела.
4. Разработаны практические задания по объектно-ориентированному языку программирования Delphi.
5. Создан электронный вариант учебного модуля.

Разработанный учебный модуль может быть использован для дополнительного изучения обучающихся, которые хотят получить дополнительные навыки в IT-сфере.

СОДЕР0ЖАНИЕ

Введение 4
1 Основы объектно-ориентированного программирования 6
1.1 Основные понятия объектно-ориентированного подхода 6
1.2 Объектно-ориентированное моделирование 10
1.3 Объектно-ориентированные языки программирования 12
1.4 Исторические аспекты объектно-ориентированного программирования 15
1.5 Объектно-ориентированный язык программирования Delphi 20
1.5.1 Элементы интерфейса среды разработки Delphi 23
1.5.2 Компоненты среды разработки Delphi 28
1.5.3 Основы программирования Delphi 30
2 Разработка учебного модуля по теме «Основы Объектно-ориентированного программирования» 32
2.1 Понятие учебного модуля 32
2.2 Назначение и структура учебного модуля 34
2.3 Обоснование программной среды разработки учебного модуля 37
2.4 Описание содержания теоретического раздела учебного модуля 41
2.5 Описание содержания практического раздела 44
Заключение 47
Список использованных источников 49
Приложение ..... **Ошибка! Закладка не определена.**

ВВЕДЕНИЕ

В последнее время большой популярностью среди программистов пользуются объектно-ориентированные языки программирования, так как они позволяют использовать преимущества объектно-ориентированного подхода не только на этапах конструирования и проектирования программных систем, но и на этапах их реализации, тестирования и сопровождения.

Использовать объектно-ориентированное программирование при разработке крупных программных проектов очень удобно. Чем проект объемнее и сложнее, тем больше выгоды можно получить при использовании объектно-ориентированного программирования. Одним из наибольших преимуществ объектно-ориентированного программирования является возможность повторного использования программного кода.

Объектно-ориентированное программирование требует оставить в стороне характерные представления о программировании, которые долгие годы рассматривались как стандартные. Однако после того, как это сделано, объектно-ориентированное программирование становится более доступным в понимании, более наглядным и превосходным средством разрешения ряда задач, доставляющих неприятности традиционному программному обеспечению.

Данный учебный модуль разрабатывается для дополнительно образования обучающихся в учебно-техническом центре (УТЦ) ООО «Омега-1». В качестве среды разработки выбрана Delphi 7, так как именно эта версия совместима с текущим оборудованием в УТЦ «Омега-1». Тем не менее, навыки, полученные при изучении языка Delphi 7, позволят гораздо быстрее освоиться при изучении более современных версий интегрированной среды Delphi.

В разработанном учебном модуле представлена необходимая теория для изучения объектно-ориентированных языков программирования, позволяющая ознакомится с основными понятиями и элементами объектно-

ориентированного подхода к программированию, описание объектно-ориентированных языков программирования и интегрированной среды разработки Delphi.

Также в учебном модуле предусмотрены практические задания в интегрированной среде Delphi.

Объект исследования — процесс обучения основам объектно-ориентированного программирования.

Предмет исследования — методические материалы, пособия, учебники, лабораторные работы по теме.

Цель исследования — разработать учебный модуль по теме «Основы объектно-ориентированного программирования».

Для достижения поставленной цели в работе были поставлены следующие задачи:

1. Рассмотреть теоретические основы объектно-ориентированного программирования.
2. Разработать структуру учебного модуля.
3. Отобрать содержание теоретического раздела.
4. Разработать практические задания по объектно-ориентированному языку программирования Delphi.
5. Создать электронный вариант учебного модуля.

1 ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

1.1 Основные понятия объектно-ориентированного подхода

Прежде чем говорить об объектно-ориентированном программировании, нужно рассмотреть, что такое объектно-ориентированный подход. *Объектно-ориентированный подход* использует объектную декомпозицию, то есть поведение системы описывается в терминах взаимодействия объектов.

Концептуальной основой объектно-ориентированного подхода является объектная модель. Основными ее элементами являются [1]:

- абстрагирование;
- инкапсуляция;
- модульность;
- иерархия.

Помимо основных существуют еще три дополнительных элемента, которые не являются, в отличие от основных, строго необходимыми:

- типизация;
- параллелизм;
- устойчивость.

Далее раскроем понятие каждого из элементов.

Абстрагирование — это выделение немаловажных характеристик некоторого объекта, которые отличают его от всех других видов объектов и, таким образом, четко определяют его концептуальные границы относительно дальнейшего рассмотрения и анализа. Абстрагирование сосредотачивает внимание на внешних особенностях объекта и позволяет отделить самые важные особенности его поведения от деталей их реализации. Выбор правильного набора абстракций для заданной предметной области представляет собой главную задачу объектно-ориентированного проектирования.

Инкапсуляция — это процесс отделения друг от друга отдельных элементов объекта, определяющих его устройство и поведение. Инкапсуляция необходима для того, чтобы изолировать интерфейс объекта, отражающий его внешнее поведение, от внутренней реализации объекта. Объектный подход предполагает, что собственные ресурсы, которыми могут манипулировать только методы самого класса, скрыты от внешней среды. Абстрагирование и инкапсуляция являются взаимодополняющими операциями: абстрагирование фокусирует внимание на внешних особенностях объекта, а инкапсуляция (или, иначе, ограничение доступа) не позволяет объектам-пользователям различать внутреннее устройство объекта.

Модульность — это свойство системы, связанное с возможностью ее декомпозиции на ряд внутренне связных, но слабо связанных между собой модулей. Инкапсуляция и модульность создают препятствия между абстракциями.

Иерархия — это ранжированная или упорядоченная система абстракций, расположение их по уровням. Основными видами иерархических структур применительно к сложным системам являются структура классов (иерархия по номенклатуре) и структура объектов (иерархия по составу). Примерами иерархии классов являются простое и множественное наследование (один класс использует структурную или функциональную часть соответственно одного или нескольких других классов), а иерархии объектов — агрегация.

Типизация — это ограничение, накладываемое на класс объектов и препятствующее взаимозаменяемости различных классов (или сильно сужающее ее возможность). Типизация позволяет защититься от использования объектов одного класса вместо другого или по крайней мере управлять таким использованием.

Параллелизм — свойство объектов находиться в активном или пассивном состоянии и различать активные и пассивные объекты между собой.

Устойчивость — свойство объекта существовать во времени (не зависимо от процесса, породившего данный объект) или в пространстве (при перемещении объекта из адресного пространства, в котором он был создан).

Основными понятиями объектно-ориентированного подхода являются *класс* и *объект*.

Объект — это предмет или явление, имеющее четко определяемое поведение. Объект обладает состоянием, поведением и индивидуальностью; структура и поведение схожих объектов определяют общий для них класс. Термины «экземпляр класса» и «объект» являются равнозначными. Состояние объекта характеризуется перечнем всех возможных (статических) свойств данного объекта и текущими значениями (динамическими) каждого из этих свойств. Поведение характеризует воздействие объекта на другие объекты и наоборот относительно изменения состояния этих объектов и передачи сообщений. Другими словами, поведение объекта всецело определяется его действиями. Индивидуальность — это свойства объекта, отличающие его от всех других объектов [9].

Определенное воздействие одного объекта на другой с целью вызвать соответствующую реакцию называется операцией. Обычно, в объектных и объектно-ориентированных языках операции, выполняемые над выбранным объектом, называются методами и являются составной частью определения класса.

Класс — это множество объектов, связанных общностью структуры и поведения. Любой объект является экземпляром класса. Определение классов и объектов — одна из самых сложных задач объектно-ориентированного проектирования.

Следующую группу важных понятий объектного подхода составляют наследование и полиморфизм. Понятие полиморфизма может быть интерпретировано, как способность класса принадлежать более чем одному типу. Наследование означает построение новых классов на основе существующих с возможностью добавления или переопределения данных и методов.

Объектно-ориентированная система изначально строится с учетом ее эволюции. Наследование и полиморфизм обеспечивают возможность определения новой функциональности классов при помощи создания производных классов — потомков базовых классов. Потомки наследуют характеристики родительских классов без изменения их первоначального описания и добавляют при необходимости собственные структуры данных и методы. Определение производных классов, при котором задаются только различия или уточнения, позволяет в огромной степени сэкономить время и усилия при производстве и использовании спецификаций и программного кода [9].

Рассмотрим пример наследования (рисунок 1).

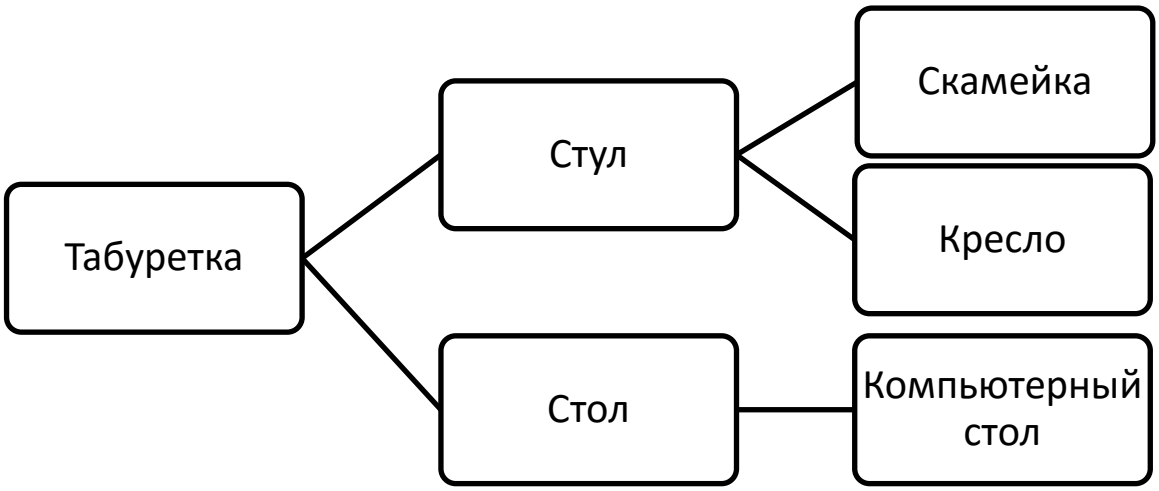

Рисунок 1— Пример наследования — иерархия объекта «табуретка»

Возьмем объект «табуретка», теперь, исходя из этого объекта, описываем объект «стул». Оба эти объекта обладают одинаковыми свойствами — опорные ножки, сидение и т.д. Поэтому желательно за основу взять табуретку и доделать ее в стул. Для этого вы создаете новый объект «стул» и пишете, что он происходит от объекта «табуретка». Ваш новый объект сразу примет все свойства «табуретки», и вам останется добавить недостающие свойства — спинка стула. Так же из объекта «табуретка» можно создать новый объект «стол». Оба эти объекта обладают одинаковыми свойствами — опорные ножки, сидение и т.д. В таком случае дополнительными свойствами будет просто удлинить ножки табуретки и увеличить до нужного размера сиде-

ние, чтобы получилась столешница. В этом примере предком будет объект «табуретка», а потомками объекты «стул» и «стол».

Важным качеством объектного подхода является согласованность моделей деятельности организации и моделей проектируемой системы от стадии формирования требований до стадии реализации. Требование согласованности моделей выполняется благодаря возможности применения абстрагирования, модульности, полиморфизма на всех стадиях разработки. Модели ранних стадий могут быть непосредственно подвергнуты сравнению с моделями реализации. По объектным моделям может быть прослежено отображение реальных сущностей моделируемой предметной области (организации) в объекты и классы информационной системы.

1.2 Объектно-ориентированное моделирование

Большинство существующих методов объектно-ориентированного анализа и проектирования (ООАП) включают как язык моделирования, так и описание процесса моделирования. Язык моделирования — это нотация (в основном графическая), используемая методом для описания проектов. Нотация представляет собой совокупность графических объектов, которые используются в моделях; она является синтаксисом языка моделирования. Например, нотация диаграммы классов определяет, каким образом представляются такие элементы и понятия, как класс, ассоциация и множественность. Процесс — это описание шагов, которые необходимо выполнить при разработке проекта [9].

Унифицированный язык моделирования UML (Unified Modeling Language) — это язык для определения, представления, проектирования и документирования программных систем, организационно-экономических систем, технических систем и других систем различной природы. UML содержит стандартный набор диаграмм и нотаций самых разнообразных видов.

UML является наследником методов объектно-ориентированного анализа и проектирования, появившихся ещё в конце 1980-х и начале 1990-х годов. Создание UML началось во второй половине 1994 года с объединения методов Booch и OMT (Object Modeling Technique) под руководством компании Rational Software. К концу 1995 г. Гради Буч и Джеймс Рамбо создали первую спецификацию Unified Method, версия 0.8. Тогда же в 1995 г. к ним присоединился создатель метода OOSE (Object-Oriented Software Engineering) Ивар Якобсон. UML является унификацией методов Буча, Рамбо и Якобсона, а также суммой передового опыта по разработке программного обеспечения (ПО).

Разработка UML преследовала следующие цели:

- предоставить разработчикам единый язык визуального моделирования;
- предусмотреть механизмы расширения и специализации языка;
- обеспечить независимость языка от языков программирования и процессов разработки;
- интегрировать накопленный практический опыт.

Язык UML находится в процессе стандартизации, проводимом OMG (Object Management Group) — организацией по стандартизации в области объектно-ориентированных методов и технологий, в настоящее время используется в качестве стандартного языка моделирования, и получил широкую поддержку в индустрии ПО. Язык UML принят на вооружение большинством крупных компаний, являющимися производителями ПО (Microsoft, IBM, Hewlett-Packard, Oracle, Sybase и др.). Помимо этого, практически все мировые производители CASE-средств, кроме Rational Software (Rational Rose), поддерживают UML в своих продуктах (Paradigm Plus 3.6, System Architec, Microsoft Visual Modeler for Visual Basic, Delphi, PowerBuilder и др.) [1].

1.3 Объектно-ориентированные языки программирования

Объектно-ориентированное программирование (ООП) — это метод программирования, при использовании которого главными элементами программ являются объекты. В языках программирования понятие объекта реализовано как совокупность свойств (структур данных, характерных для данного объекта), методов их обработки (подпрограмм изменения их свойств) и событий, на которые данный объект может реагировать и, которые приводят, как правило, к изменению свойств объекта.

Реализация программного обеспечения связана с использованием одного из языков программирования. Наиболее удобными для реализации программных систем, разработанных в рамках объектно-ориентированного подхода. *Объектно-ориентированные* языки — это языки, основанные на построении объектов как набора данных и операций над ними. Объектно-ориентированные языки объединяют и расширяют возможности, присущие процедурным и аппликативным языкам [9].

На данный момент существует достаточно большое количество объектно-ориентированных языков программирования (таблица 1).

Таблица 1 — Примеры объектно-ориентированных языков программирования

C#	Dylan	JavaScript
C++	ActionScript (3.0)	Ruby
Java	Scala	Smalltalk
Delphi	Python	Ada
Eiffel	PowerBuilder	Xbase++
Simula	Perl	X++
JScript .NET	Visual DataFlex	PHP
Objective-C	Object Pascal	Cyclone

Рассмотрим подробней некоторые объектно-ориентированные языки программирования.

*Я*зык *Smalltalk* был разработан в компании Xerox PARC Аланом Кэйем. В основу были положены идеи Simula, Smalltalk является одновременно и

языком программирования, и средой разработки программ. Это чисто объектно-ориентированный язык, в котором абсолютно все рассматривается как объекты; даже целые числа — это классы. Вслед за Simula, Smalltalk является важнейшим объектно-ориентированным языком, поскольку он не только оказал влияние на последующие поколения языков программирования, но и заложил основы современного графического интерфейса пользователя, на которых непосредственно базируются интерфейсы Macintosh, Windows и Motif.

В основу языка положены две простые идеи [13]:

- все является объектами;
- объекты взаимодействуют, обмениваясь сообщениями.

Основным недостатком языка Smalltalk являются большие требования к памяти и невысокое быстродействие полученных программ. Это связано с не самой удачной реализацией объектно-ориентированных особенностей

Язык программирования C++ был разработан Бьерном Страуструпом. Непосредственным предшественником C++ является C with Classes, созданный тем же автором в 1980 году. Язык C with Classes, в свою очередь, был создан под сильным влиянием C и Simula. C++ — это в значительной степени надстройка над C. В определенном смысле можно назвать C++ улучшенным C, тем C, который обеспечивает контроль типов, перегрузку функций и ряд других удобств. Но главное в том, что C++ добавляет к C объектную ориентированность.

Существует несколько версий C++. В версии 1.0 реализованы основные механизмы объектно-ориентированного программирования, такие как одиночное наследование и полиморфизм, проверка типов и перегрузка функций. В 1989 году была создана версия 2.0, в которой нашли отражение многие дополнительные свойства, возникшие на базе широкого опыта использования языка многочисленным количеством пользователей. В версии 3.0 (1990) появились шаблоны и обработка исключений. C++ продолжает совершенствоваться и в наши дни, так в 1998 году вышла новая версия стандарта, содер-

жащая в себе некоторые довольно существенные изменения. Язык стал основой для разработки современных больших и сложных проектов.

В 1983 году под эгидой Министерства Обороны США был создан язык Ada. Язык замечателен тем, что очень много ошибок может быть выявлено на этапе компиляции. Кроме того, в языке Ada поддерживаются многие аспекты программирования, которые часто отдаются на откуп операционной системе (параллелизм, обработка исключений). В 1995 году был принят стандарт языка Ada 95, который развивает предыдущую версию, добавив в нее объектно-ориентированность и исправив некоторые неточности. Оба этих языка не получили широкого распространения вне военных и прочих крупномасштабных проектов (авиация, железнодорожные перевозки). Основной причиной является сложность освоения языка и достаточно громоздкий синтаксис.

Object Pascal создавался работниками компании Apple Computer (некоторые из которых были участниками проекта Smalltalk) совместно с Никлаусом Виртом (Niklaus Wirth), создателем языка Pascal. Object Pascal известен с 1986 года и является первым объектно-ориентированным языком программирования, который был включен в Macintosh Programmer's Workshop (MPW), среду разработки для компьютеров Macintosh фирмы Apple.

В этом языке отсутствуют методы класса, переменные класса, множественное наследование и метаклассы. Эти механизмы исключены специально, чтобы сделать язык простым для изучения начинающими «объектными» программистами [13].

Сравнительная характеристика объектно-ориентированных языков программирования рассмотренных выше представлена в таблице 2.

Таблица 2 — Сравнение объектно-ориентированных языков программирования

Сравнительные характеристики	**Object Pascal**	**C++**	**Ada**	**Smalltalk**
Абстракции: Переменные экземпляра Методы экземпляра Переменные класса Методы класса	 да да нет нет	 да да да да	 да да нет нет	 да да да да
Инкапсуляция: Переменных методов	 открытые	 открытые защищенные закрытые	 открытые закрытые	 закрытые открытые
Модульность: Разновидности модулей	 модуль (unit)	 файл	 пакет	 нет
Иерархии: Наследование Шаблоны Метаклассы	 одиночное нет нет	 множественное да нет	 нет да нет	 одиночное нет да
Типизация: Сильная типизация Полиморфизм	 да да (одиночный)	 да да (одиночный)	 да нет	 нет да (одиночный)
Параллельность: Многозадачность	 нет	 непрямая (посредством классов)	 да	 непрямая (посредством классов)
Сохраняемость: Долгоживущие объекты	 нет	 нет	 нет	 нет

1.4Исторические аспекты объектно-ориентированного программирования

Объектно-ориентированное программирование появилось в результате развития процедурного программирования, где данные и подпрограммы

(процедуры, функции) их обработки не связанны. Основоположниками объектного подхода в программировании считаются норвежцы Оле Джохан Дал и Кристен Нюгорт, авторы языка Simula.

История Simula началась в 1962 году с проекта Simulation Language, предназначенного для программного моделирования метода Монте-Карло. Нюгорт, занимая в то время должность директора по науке Норвежского компьютерного центра(NCC), занялся созданием языка дискретного моделирования. Он предложил сотрудничество Оле Джохана Дала, коллеге по экспериментальной группе Министерства обороны Норвегии.

В 1965 году у авторов зародилась идея объединить данные с процедурами, их обрабатывающими. После успешного обсуждения возможностей Simula I на саммите НАТО в 1966 г. было решено продолжить его совершенствование. В язык вошли новые средства моделирования и имитации мультипроцессной работы. Авторы также придумали термины «класс» и «объект». В то же время возникла и технология наследования. Создатели Simula ввели в язык возможность использования разными классами общих свойств посредством указания названия класса в виде префикса. После публичного анонса новая технология вызвала интерес в Дании, Германии и СССР. У нас в конце 60–х появилась реализация Simula для УРАЛа-16.

Алан Кей внимательно изучал идеи, заложенные в Simula и еще два оригинальных языка — LISP, применявшийся для задач искусственного интеллекта, и LOGO, предназначенный для обучения базовым понятиям программирования. В ходе ознакомления с данными языками программирования Кей придумал новую концепцию разработки, в соответствии с которой набор последовательно выполняющихся инструкций мог быть заменен на многомерную среду взаимодействия объектов, общающихся друг с другом путем асинхронного обмена сообщениями. В результате чего появлялась возможность поддержки подобной среды не одним, а множеством компьютеров, объединенных в сеть. Правда, для своего времени эта идея оказалась слишком революционной.

Позже Кей перешел в Стэнфордскую лабораторию по искусственному интеллекту, а в 1972 году устроился на работу в хорошо известный научный центр Xerox PARC, где и воплотил свои идеи в новом объектном языке SmallTalk, первоначально названном им Biological System и смоделированном на Бейсике, а затем реализованном на ассемблере. В процессе этой деятельности он предложил знаменитый термин «объектно-ориентированное программирование» (ООП). Заложенные в SmallTalk идеи ООП до сих пор остались непревзойденными в других языках и системах. Если Кея нередко называют отцом SmallTalk, то матерью этого уникального языка можно считать профессора лингвистики Адель Голдберг, работавшую в те годы в тесной связке с Аланом в Xerox PARC. Она разработала первую документацию к SmallTalk, а затем написала несколько книг и большое количество статей по методологиям объектного анализа.

В 1974 году Марвин Мински, родоначальник теории искусственного интеллекта, предложил идею фрейма, отделившего описание класса (структуры) объекта от его конкретного представления (экземпляра) и быстро завоевавшего популярность в языках искусственного интеллекта. Фрейм стал прямым предшественником современного понятия объектов в C++.

В 1976 году Кринстен Нюгорн создал новый язык BETA, в котором ввел концепцию шаблонов — более высокого уровня абстракций, нежели объекты.

В 1980 году Бьерн Страуструп, продолживший дело своих коллег из лаборатории Bell, дополнил язык C концепцией классов, основанной на фреймах и объектных механизмах Simula.

В 1982 году в Мехико прошла 8-я Международная конференция по сверхбольшим базам данных (БД) (VLDB), на которой была предложена концепция объектно-ориентированных БД и рассматривались вопросы расширения существовавших языков запросов к БД новыми, объектными типами данных.

В 1983 году. Страуструп дал своему творению окончательное название C++.

В 1986 году в Портленде прошла первая всемирная конференция по объектно-ориентированным системам программирования. Возможно, именно прозвучавшие на ней доклады оказали главное влияние на Уильяма Аткинсона, инженера Apple, который через год после этого спроектировал систему HyperCard, прообраз современных визуальных сред быстрой разработки. Эффективность новой технологии оказалась столь высокой, что уже в 1989 году одиннадцать компаний: 3Com, American Airlines, Canon, Data General, Hewlett-Packard, Philips Telecommunications, Sun Microsystems и Unisys, основали группу OMG (Object Management Group), призванную формировать индустриальные стандарты на объектное программирование и упрощать интеграцию приложений с помощью универсальных кросс-платформенных технологий. OMG первым делом приступила к выработке единого стандарта компонентной модели CORBA (Common Object Request Broker Architecture), набора спецификаций, определяющих способы объектно-ориентированного взаимодействия компонентов промежуточного уровня в гетерогенных средах без привязки к конкретным языкам программирования. С самого начала CORBA нацеливалась на поддержку крупных, индустриальных проектов, и этот подход со временем себя полностью оправдал.

В 1992 году вышел стандарт CORBA 1.0, определяющий главные аспекты функционирования CORBA-систем. В него были включены базовое описание объектной модели, наборы программных интерфейсов поддержки CORBA-систем, а также декларативный язык определения интерфейсов Interface Definition Language (IDL), созданный OMG для описания распределенных интерфейсов.

В 1994 году опубликован стандарт CORBA 2.0, который быстро получил массовое признание, так как представлял собой богатый и глубоко проработанный набор документов и охватывал большинство востребованных на рынке задач. В нем были исправлены недостатки прежней версии, в резуль-

тате чего CORBA 2.0 начал поддерживать транзакции и понимать универсальную кодировку Unicode, а также появился набор средств обеспечения безопасности и взаимодействия COM- и CORBA-объектов. Гради Буч и Джеймс Румбах из Rational Software объединили две методологии визуального моделирования Booch и OMT и создали на их основе новый язык UML (Unified Modeling Language).

В 1995 году Sun Microsystems выкладывает в свободный доступ элемент технологии Java — среду HotJava, поддерживающую мобильный код, разработку проекта Green, которая к тому времени считалась в Sun практически пропащей, если бы не развитие Сети. Новинку сразу же лицензирует Netscape Communication, а следом за ней к Java проявляют коммерческий интерес десятки компаний, в том числе Microsoft, IBM, Adobe, Borland, Lotus, Oracle.

Корпорация Borland выпустила первую версию среды быстрой визуальной разработки Delphi 1, основанную на концепции библиотек стандартных компонентов.

В 1997 году Sun Microsystems разрабатывает концепцию Enterprise JavaBeans — технологию создания корпоративных Java-компонентов, которые можно исполнять на серверах приложений, реализуя логику крупных, хорошо масштабируемых и защищенных систем на платформенно-независимой основе.

Эксперты OMG понимали не только важность объектных технологий программирования, но и острую необходимость в универсальных методологических концепциях проектирования крупных систем. Секретом стабильности системы и высокой отдачи инвестиций специалисты OMG называют независимую UML-модель и приступают к созданию концепции «Архитектура, управляемая моделью» (Model Driven Architecture, MDA). В ее основу закладывается базовая платформенно-независимая UML-модель системы, несколько платформенно-зависимых моделей и коллекция определений программных интерфейсов. Первую реализацию этой универсальной концепции

(так называемое «отображение в объектный стандарт») OMG выполнила, конечно, для CORBA.

В 2000 году Microsoft анонсирует новую объектную платформу .NET и новый язык программирования C#, объединяющий лучшие свойства C++ и Java. Он был предложен Microsoft во многом в противовес Java.

В 2001 году OMG выпускает спецификацию CORBA 3.0. Она дополнена возможностями асинхронного обмена сообщениями, разработки систем реального времени и создания встраиваемых систем. В ней появились подключаемые компоненты, поддержка XML и средства интеграции различных Интернет-технологий. Особый акцент в третьей версии CORBA сделан на эффективном взаимодействии с Java.

В 2002 году Опубликована последняя официальная версия CORBA 3.0.2 [139].

1.5 Объектно-ориентированный язык программирования Delphi

Прежде чем рассмотреть объектно-ориентированный язык программирования Delphi, нужно для начала понять, что представляет собой визуальное программирование.

Визуальное программирование — это способ создания программы для электронных вычислительных машин путём манипулирования графическими объектами вместо написания её текста. Важно иметь в виду:

- графический язык программирования — это язык программирования со своим синтаксисом;
- визуальные средства разработки — это средства проектирования интерфейсов или какой-либо CASE-системы, для быстрой разработки приложений [20].

Классификация визуальных языков программирования в зависимости от типа и степени визуального выражения представлена на рисунке 2.

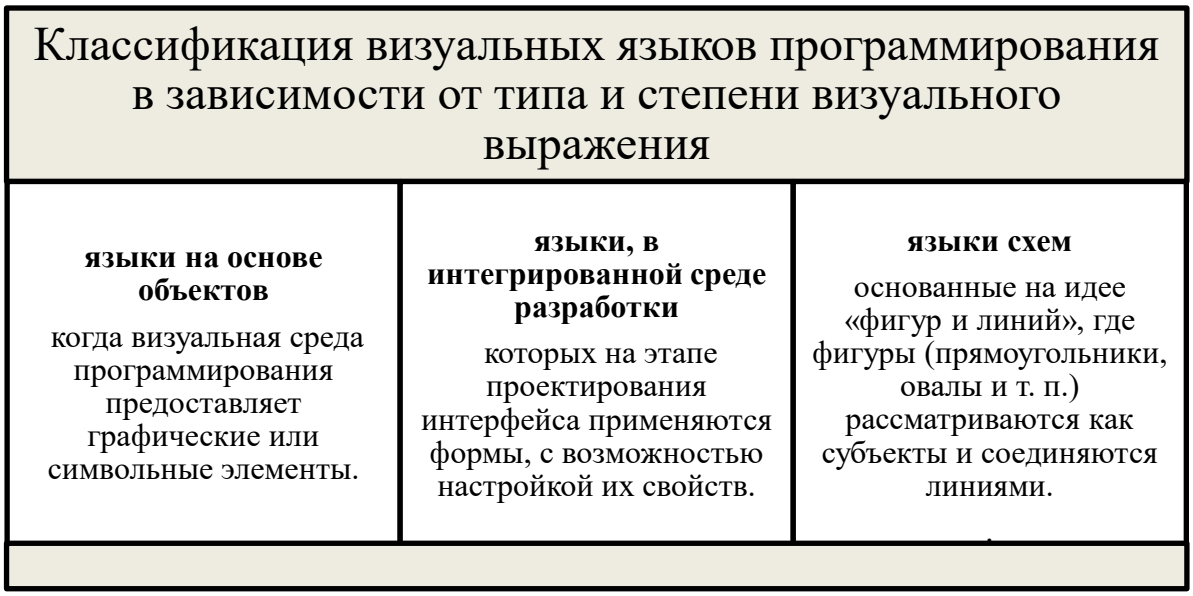

Рисунок 2 — Классификация визуальных языков программирования

Мы рассмотрим такую интегрированную среду, как Delphi — интегрированная среда разработки программного обеспечения для Microsoft Windows, Mac OS, iOS и Android на языке Delphi, созданная первоначально фирмой Borland и на данный момент принадлежащая и разрабатываемая Embarcadero Technologies. Embarcadero Delphi является частью пакета Embarcadero RAD Studio и поставляется в пяти редакциях: Starter, Professional, Enterprise, Ultimate и Architect. Координирующий офис Embarcadero, ответственный за разработку Delphi, находится в Торонто, тогда как сама разработка сконцентрирована главным образом в Канаде и Испании.

В основу интегрированной среды разработки программного обеспечения Delphi заложен язык программирования Object Pascal, унаследовавший основные концепции «классического» языка Turbo Pascal.

Концепция Delphi была реализована в 1995 году, когда вышла первая версия среды разработки. В основу этого программного продукта легли концепции объектно-ориентированного программирования и визуального подхода к построению интерфейса приложения. От версии к версии разработчики улучшают средства для разработки приложений. На данный момент вышло более 20 версий интегрированной среды программирования Delphi.

Каждая новая версия дополняет предыдущую. Большинство версий Delphi выпускается в нескольких вариантах: Standart — стандартном, Professional — профессиональном, Client/Server — клиент/сервер, Enterprise — разработка баз данных предметных областей. Различаются варианты в основном разным уровнем доступа к системам управления базами данных. Последние варианты — Client/Server и Enterprise, в этом отношении наиболее мощные [28].

В учебном модуле будет рассматриваться популярная 7 версия среды разработки Delphi, так как она является простой в изучении и создании приложений.

Изучение языка Delphi довольно таки актуально в наше время. Одно из основных, и, несомненно, самых важных качеств языка Delphi, это то, что он прост в освоении. Ведь всегда надо с чего-то начинать, поэтому изучение языков объектно-ориентированного программирования можно смело начинать именно с Delphi. Не нужно быть очень опытным пользователем, чтобы начать изучение данного языка. Практика показывает, что пользователи, изучавшие язык Delphi, впоследствии, при изучении уже более сложных объектно-ориентированных сред, опережают своих сверстников в процессе обучения. Так же Delphi обладает рядом других преимуществ:

- больше ошибок находится при компиляции из-за строгой типизации и простого синтаксиса;
- отладка в 2-3 раза быстрей, чем для любых вариантов C;
- хорошая читаемость, отсюда простота сопровождения;
- огромное количество готовых компонент, под любые задачи.

Delphi — это комбинация нескольких важнейших технологий:

- высокопроизводительный компилятор в машинный код;
- объектно-ориентированная модель компонентов;
- визуальное построение приложений из программных прототипов;
- масштабируемые средства для построения баз данных.

1.5.1Элементы интерфейса среды разработки Delphi

Среда программирования Delphi — это сложный механизм, обеспечивающий высокоэффективную и продуктивную работу программиста. Визуально она реализуется несколькими одновременно раскрытыми на экране окнами. Окна могут быть свободно перемещены по экрану. Но так же они частично или полностью перекрывают друг друга, что обычно вызывает у пользователя, привыкшего к относительной «строгости» среды текстового процессора Word или табличного процессора Excel, ощущение некоторого дискомфорта. После приобретения опыта работы с Delphi это ощущение пройдет, и вы сможете быстро отыскивать нужное окно, чтобы изменить те или иные функциональные свойства создаваемой вами программы, ибо каждое из окон несет в себе некоторую функциональность, т. е. предназначено для решения определенных задач [3].

Графический интерфейс Delphi 7 представлен на рисунке 3.

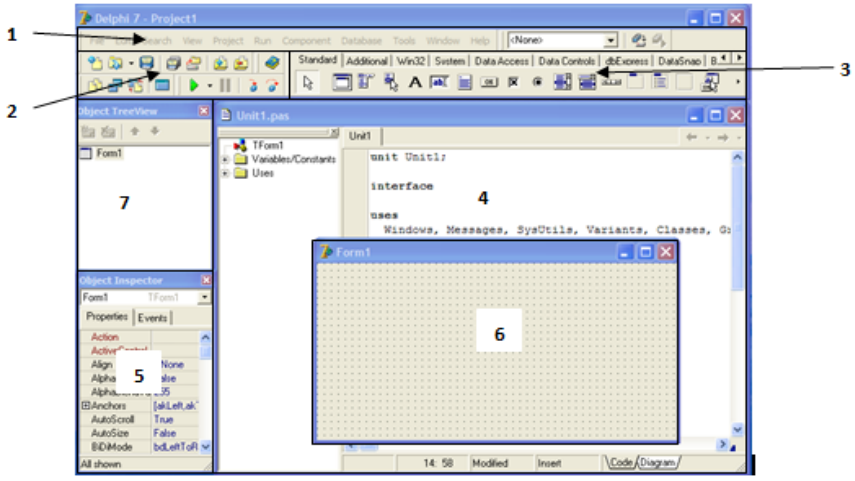

Рисунок 3 — Среда Delphi 7

Элементы среды:

1. *Меню.* Предоставляет доступ к главным командам Delphi. Выбор нужного пункта из него можно осуществить следующими способами. Можно использовать меню для выполнения широкого круга задач, например: для открытия и сохранения файлов, компиляции и запуска программ, управления отладчиком, для настройки среды программирования и многое другое. Следует отметить,

что содержание пунктов Главного Меню в разных версиях Delphi несколько отличаются.

2. *Панель Быстрого Доступа*. Дублирует наиболее часто используемые команды меню. Если задержать мышь над любой из иконок, то появится всплывающая подсказка, объясняющая назначение данной иконки. Первое время, Вам могут понадобиться только три из них: «Open» (Открыть), «Help» (Помощь) и кнопка с зеленым треугольником «Run» (Запуск программы), дублирующая команду Run\Run (Выполнить) в меню.

3. *Палитра Компонентов.* Содержит компоненты, которые можно использовать на форме. Палитра имеет закладки, обеспечивающие быстрый поиск нужного компонента. Под компонентом понимается некий функциональный элемент, содержащий определенные свойства и размещаемый программистом в окне формы. С помощью компонентов создается каркас программы, во всяком случае, ее видимые на экране внешние проявления: окна, кнопки, списки выбора и т. д.

4. *Редактор кода* (*Окно редактора исходного текста*). В этом окне составляется код будущей программы. Часть программы система Delphi формирует автоматически. Даже «пустая» программа для Windows собирается из нескольких тысяч операторов языка Pascal. Добавлять нужные операторы система Delphi начинает еще до того, как программист приступил к созданию своей программы. Поэтому окно редактора кода никогда не бывает пустым.

5. *Инспектор Объектов.* Здесь отображаются свойства, характеристики объекта, компонента, который в данный момент выделен на форме. Можно изменять вид и поведение объекта с помощью Инспектора Объектов. Это окно содержит две страницы - Properties (Свойства) и Events (События). Страница properties служит для установки необходимых свойств компонента, страница Events позволяет определить реакцию компонента на то или иное событие.

6. *Окно формы.* В этой области и будет создаваться приложение, и здесь же будут размещаться объекты. Окно формы представляет собой проект Windows-окна будущей программы. Вначале это окно пусто. Точнее, оно содержит стандартные для Windows интерфейсные элементы — кнопки вызова системного меню, максимизации, минимизации и закрытия окна, полосу заголовка и очерчивающую рамку. Вся рабочая область окна обычно заполнена точками координатной сетки, необходимой для упорядочения размещаемых на форме компонентов (вы можете убрать эти точки, вызвав с помощью меню Tools | Environment options соответствующее окно настроек и убрав флажок в переключателе Display Grid на окне, связанном с закладкой Preferences).

7. *Окно дерева объектов.* Предназначено для наглядного отображения связей между отдельными компонентами, размещенными на активной форме. Щелчок по любому компоненту в этом окне активизирует соответствующий компонент в окне формы и отображает свойства этого компонента в окне Инспектора объектов.

Работа над новым проектом, начинается с создания стартовой формы. Так на этапе разработки программы называют диалоговые окна. Стартовая форма создается путем изменения значений свойств формы Forml и добавления к форме необходимых компонентов (полей ввода и вывода текста, командных кнопок). Свойства формы определяют ее внешний вид: размер, положение на экране, текст заголовка, вид рамки. Для просмотра и изменения значений свойств формы и ее компонентов используется окно Object Inspector. В верху окна Object Inspector указано имя объекта, значения свойств которого отображается в данный момент. В левой колонке вкладки Properties (Свойства) перечислены свойства объекта, а в правой — указаны их значения (таблица 3).

Таблица 3 – Свойства формы

Свойства	**Описание**
Name	Имя формы. В программе имя формы используется для управления формой и доступа к компонентам формы.
Caption	Текст заголовка
Width	Ширина формы
Height	Высота формы
Top	Расстояние от верхней границы формы до верхней границы экрана
Left	Расстояние от левой границы формы до левой границы экрана
BorderStyle	Вид границы. Граница может быть обычной (bsSizeable), тонкой (bsSingle) или отсутствовать (bsNone). Если у окна обычная граница, то во время работы программы пользователь может при помощи мыши изменить размер окна. Изменить размер окна с тонкой границей нельзя. Если граница отсутствует, то на экран во время работы программы будет выведено окно без заголовка. Положение и размер такого окна во время работы программы изменить нельзя
BorderIcons	Кнопки управления окном. Значение свойства определяет, какие кнопки управления окном будут доступны пользователю во время работы программы. Значение свойства задается путем присвоения значений уточняющим свойствам biSystemMenu, biMinimaze, biMaximaze и biHelp. Свойство biSystemMenu определяет доступность кнопки Свернуть и кнопки системного меню, biMinimaze — кнопки Свернуть, biMaximaze — кнопки Развернуть, biHelp — кнопки вывода справочной информации
Color	Цвет фона. Цвет можно выбрать, указав название цвета или привязку к текущей цветовой схеме операционной системы. Во втором случае цвет определяется текущей цветовой схемой, выбранным компонентом привязки и изменяется вместе с цветовой схемой операционной системы
Icon	Значок в заголовке диалогового окна, обозначающий кнопку вывода системного меню

Окончание таблицы 3

Font	Шрифт. Шрифт, используемый «по умолчанию» компонентами, находящимися на поверхности формы. Изменение свойства Font формы приводит к автоматическому изменению свойства Font компонента, располагающегося на поверхности формы. То есть компоненты наследуют свойство Font от формы (имеется возможность запретить наследование)

Форма — это обычное окно. Поэтому его размер можно изменить точно так же, как размер любого другого окна, т, е. захватом и перемещением (с помощью мыши) границы. По окончании перемещения границ автоматически изменятся значения свойств Height и Width. Они будут соответствовать новому размеру формы [3].

Для системы Delphi каждая незавершенная программа — это проект. Проект включает в себя множество файлов. Наиболее важными являются три файла: файл формы, файл кода и файл проекта.

Проект состоит из:

- файла проекта Project1.dpf;
- файла параметров проекта Project1.dof;
- файла ресурсов проекта Project.res;
- файла настроек проекта Project1.cfg;
- файла описания формы Unit1.dfm;
- файла модуля формы Unit1.pas.

Файл модуля формы доступен для редактирования, именно он отображается в Редакторе Кода. Остальные файлы создаются Delphi автоматически. В процессе компиляции программы файлы преобразуются в исполняемый exe-файл, который, по умолчанию, создается в той же папке, в которой расположен файл проекта. В проекте могут быть задействованы несколько форм, а также дополнительные модули и файлы ресурсов, при этом схема компиляции остается похожей. Для сохранения проекта нужно воспользоваться пунктом главного меню Save Project As.… Чтобы не смешивать файлы разных проектов, желательно каждый новый проект сохранять в отдельную

папку. Сначала предлагается ввести имя для модуля формы, а затем имя проекта.

1.5.2Компоненты среды разработки Delphi

Компоненты — это визуальные объекты приложения. Таким образом, компонентами являются сама форма приложения и все визуальные объекты, представленные значками в палитре компонентов [4].

Компоненты первостепенной важности расположены на панели Standard (стандартные) палитры компонентов (рисунок 4).

Рисунок 4 — Базовые визуальные компоненты

Компоненты панели Standard:

- компонент Label предназначен для вывода надписей;
- компонент Edit используется для ввода текста пользователем;
- компонент Memo используют для работы с несколькими строками. Строки хранятся в свойстве Lines;
- компонент Button реализуется как кнопка;
- компонент CheckBox используется для работы с пунктами;
- компонент ComboBox предназначен для выбора текста из нескольких альтернатив;
- компонент RadioGroup используется для множественного выбора. Чтобы добавить в него пункты, необходимо щелкнуть на кнопку свойства Items и ввести названия пунктов. Номер выделенного пункта соответствует свойству ItemIndex, нумерация начинается с нуля;
- компонент Panel может служить простейшим контейнером компонентов. На него можно поместить любой другой компонент, например, кноп-

ку. При перемещении контейнера передвигаются все находящиеся в нем компоненты;

- компонент ListBox отображает прокручиваемый список (обычно текстовых строк), из которого пользователь может выбрать один или несколько элементов. Выбранные элементы отмечаются цветом.

Для того чтобы добавить в форму компонент, необходимо в палитре компонентов выбрать этот компонент, щелкнув левой кнопкой мыши, далее установить курсор в ту точку формы, в которой должен быть левый верхний угол компонента, и еще раз щелкнуть левой кнопкой мыши. В результате в форме появляется компонент стандартного размера.

Размер компонента можно задать в процессе его добавления к форме. Для этого надо после выбора компонента из палитры поместить курсор мыши в ту точку формы, где должен находиться левый верхний угол компонента, нажать левую кнопку мыши и, удерживая ее нажатой, переместить курсор в точку, где должен находиться правый нижний угол компонента, затем отпустить кнопку мыши. В форме появится компонент нужного размера. Каждому компоненту Delphi присваивает собственное имя, состоящее из названия компонента и его порядкового номера [13].

Рассматривая компоненты нужно не забыть про такое понятие, как событие. Событие (Event) — это то, что происходит во время работы программы. В Delphi каждому событию присвоено свое имя.

Реакцией на событие должно быть какое-либо действие. В Delphi реакция на событие реализуется как процедура обработки события. Таким образом, для того чтобы программа выполняла некоторую работу в ответ на действия пользователя, программист сначала должен написать процедуру обработки соответствующего события. Следует обратить внимание на то, что значительную часть обработки событий берет на себя сам компонент. В связи с этим программисту необходимо разрабатывать процедуру обработки события только в том случае, если реакция на событие отличается от стандартной или изначально не определена [13].

1.5.3Основы программирования Delphi

Особенности при написании кода программы [13]:

- кириллицу можно использовать только в текстовых константах, заключенных в апострофы. Использование русских букв в идентификаторах невозможно;
- все ключевые слова можно использовать только по их прямому назначению. Delphi выделяет их жирным шрифтом в тексте программы;
- операторы отделяются друг от друга символом «;» (точка с запятой). После слова begin и перед словом end символ «;» обычно не ставится;
- комментарии помещаются в фигурные скобки, либо можно использовать знак // для того, чтобы закомментировать строку.

Язык Delphi унаследовал от предыдущих версий языка Turbo Pascal все основные операторы:

- оператор присваивания;
- составной оператор;
- условный оператор;
- оператор варианта;
- операторы цикла.

При написании больших программ разработчик неизбежно сталкивается с необходимостью написания подпрограмм. Delphi предоставляет возможность написания подпрограмм двух типов: процедуры и функции. Подпрограммы, по праву, могут рассматриваться как маленькие самостоятельные программы. Так же, как и основная программа, подпрограмма может иметь свои собственные объявления, которые называются локальными, потому что они видимы и доступны только в ее пределах.

Подпрограммы могут иметь параметры. Параметр, используемый как локальная переменная, передает в подпрограмму определенное значение и обеспечивает ее работу с различными наборами исходных данных.

Процедура обеспечивает удобное средство объединения исполняемых операторов в самостоятельную, логически завершенную смысловую группу. Она может иметь любое количество параметров (в том числе, может вообще не иметь параметров), но не может участвовать в выражениях, стоящих в правой части оператора присваивания.

Общий вид:

Procedure <Имя процедуры> [(<Список формальных параметров>)];

[<Локальные объявления>]

begin

<Блок исполняемых операторов>

end;

Вызов процедуры из основной программы (в Delphi такой программой обычно является обработчик некоторого события) осуществляется по ее имени, за которым следует список фактических параметров: <Имя процедуры> [(<Список фактических параметров>)].

Функция в отличие от процедуры всегда возвращает в основную программу единственное значение определенного типа.

Общий вид:

Function <Имя функции > [(<Список формальных параметров>)]: тип функции;

[<Локальные объявления>]

begin

<Блок исполняемых операторов>

.end;

Вызов функции осуществляется по ее имени, за которым следует список фактических параметров, заключенный в круглые скобки. Вызов функции обычно участвует в выражениях, стоящих в правой части оператора присваивания так же, как и обычные переменные.

2 РАЗРАБОТКА УЧЕБНОГО МОДУЛЯ ПО ТЕМЕ «ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ»

2.1 Понятие учебного модуля

Модульное обучение имеет свои корни, как в педагогической теории, так и в практике. Основная идея этой технологии состоит в разделении содержания каждой дисциплины на составные части в соответствии с профессиональными, педагогическими и дидактическими задачами. П. А. Юцавечене в своих исследованиях подчеркивает: «Сущность модульного обучения состоит в том, что обучающийся более самостоятельно или полностью самостоятельно может работать с предложенной ему индивидуальной учебной программой, содержащей в себе целевую программу действий, банк информации и методические указания по достижению поставленных дидактических целей. При этом функции педагога могут варьироваться от информационно-контролирующей до консультативно-координирующей» [13].

Главным понятием теории модульного обучения является понятие «модуль». Несмотря на достаточную традицию использования технологии модульного обучения существуют различные точки зрения на понимание модуля и технологии его построения как в плане структурирования содержания обучения, так и разработки форм и методов обучения. Некоторые зарубежные авторы (В. Гольдшмидт, М. Гольдшмидт и др.) понимают под модулем формирование самостоятельно планируемой единицы учебной деятельности, помогающей достичь четко определенных целей. Дж. Рассел определяет модуль как автономную порцию учебного материала [13].

В работах Ю. К. Башлова и В. А. Рыжова мы находим понимание модуля как определенного объема информации, необходимой для выполнения какой-либо конкретной профессиональной деятельности. Модуль может

включать несколько модульных единиц, каждая из которых содержит описание одной законченной операции или приема. Модульные единицы могут расширять и дополнять содержание модуля в зависимости от требований конкретной профессиональной деятельности. Авторы отмечают следующие преимущества и особенности модульного обучения:

- разбивка (учебных курсов и дисциплин образовательного стандарта) на законченные части (модули и его элементы), имеющие самостоятельное значение;
- отсеивание материала, являющегося «лишним» для данного конкретного вида работ;
- максимальная индивидуализация продвижения в обучении [13].

Модульная система обучения — это современная педагогическая технология, которая базируется на блочном (модульном) построении материала, который усваивается последовательно и оценивается путем накопления рейтинговых баллов за занятия и самостоятельную работу. Она реализуется в контексте принципов познавательной деятельности, индивидуальной структуризации программы и психологического комфорта. Преподаватель в учебном плане самостоятельно распределяет количество баллов на каждый модуль, за разные виды учебной деятельности, формы контроля знаний.

Главная сущность модульного обучения состоит в том, что обучающийся полностью самостоятельно достигает целей учебно-познавательной деятельности в процессе работы над модулем — целевым функциональным узлом, в который объединены учебное содержание и приемы учебной деятельности по овладению этим содержанием. Основными мотивами внедрения в учебный процесс модульной технологии могут быть [24]:

- гарантированность достижения результатов обучения;
- возможность работать обучающимся в группах или парах;
- возможность общения с товарищами;
- возможность выбора уровня обучения;
- возможность работать в индивидуальном темпе;

- раннее предъявление конечных результатов обучения;
- «мягкий» контроль в процессе освоения учебного материала.

Понятие «модуль» является одним из новых терминов в современном российском образовании. Это структурированная часть образовательной программы, в рамках которой изучается несколько дисциплин, учебных курсов и разделов наук. Термин «модуль» часто употребляют в качестве синонима рабочей программы дисциплины, цикла дисциплин учебного плана, программы учебного курса [16].

Сердцевина модульного обучения — учебный модуль — это самостоятельный блок учебной информации, включающий в себя цели и учебные задачи, методические рекомендации, ориентировочную основу действий преподавателя, систему контроля успешности выполнения учебной деятельности[24].

Учебный модуль может быть предназначен для самостоятельного изучения учебного материала по определенной теме или для поддержки лекционного курса с целью его углубленного изучения.

Кроме того, обучаемый может воспользоваться учебным модулем самостоятельно, без помощи преподавателя или руководителя, находя ответы на интересующие его вопросы.

Учебный модуль позволяет

- объединить материал изучаемой темы;
- представить учебный материал в наглядной форме;
- обеспечить быстрое нахождение необходимой информации;
- структурировать информацию и переходы от одной темы к другой посредством навигации.

2.2 Назначение и структура учебного модуля

Учебный модуль разработан в качестве дополнительного материала, который может использоваться в самостоятельной работе обучающихся, ко-

торые изучают основы программирования. В результате выполнения учебного модуля обучающиеся получают базовые знания по объектно-ориентированному программированию, узнают про интегрированную среду Delphi и научатся программировать в ней. Структура учебного модуля состоит из двух разделов: теоретического и практического.

Выделение теории в отдельный раздел позволяет обучающимся еще до выполнения практических заданий изучить основные понятия по объектно-ориентированному программированию и проконтролировать уровень их усвоения с помощью контрольных вопросов, которые находятся в конце каждой главы. Это повышает эффективность выполнения самих практических заданий. Структура учебного модуля представлена на рисунке 5.

В теоретический раздел учебного модуля входят 8 глав:

1. Основные понятия объектно-ориентированного подхода.
2. Объектно-ориентированное моделирование.
3. Объектно-ориентированные языки программирования.
4. Исторические аспекты объектно-ориентированного программирования.
5. Объектно-ориентированный язык программирования Delphi.
6. Элементы интерфейса среды разработки Delphi.
7. Компоненты среды разработки Delphi.
8. Основы программирования Delphi.

В практическом разделе представлены четыре задания по объектно-ориентированном языку программирования Delphi. В конце раздела находится контрольное задание, которое закрепляет навыки по созданию приложений в среде разработке Delphi.

Для выполнения практических заданий требуется следующее оборудование: персональный компьютер с установленной виртуальной машиной, в которой присутствует интегрированная среда разработки Delphi 7.

Достоинства данного учебного модуля заключается в том, что обучающиеся могу самостоятельно изучить представленный материал дома. Для вы-

явления результатов усвоения показывают преподавателю выполненные задания и отвечают на контрольные вопросы.

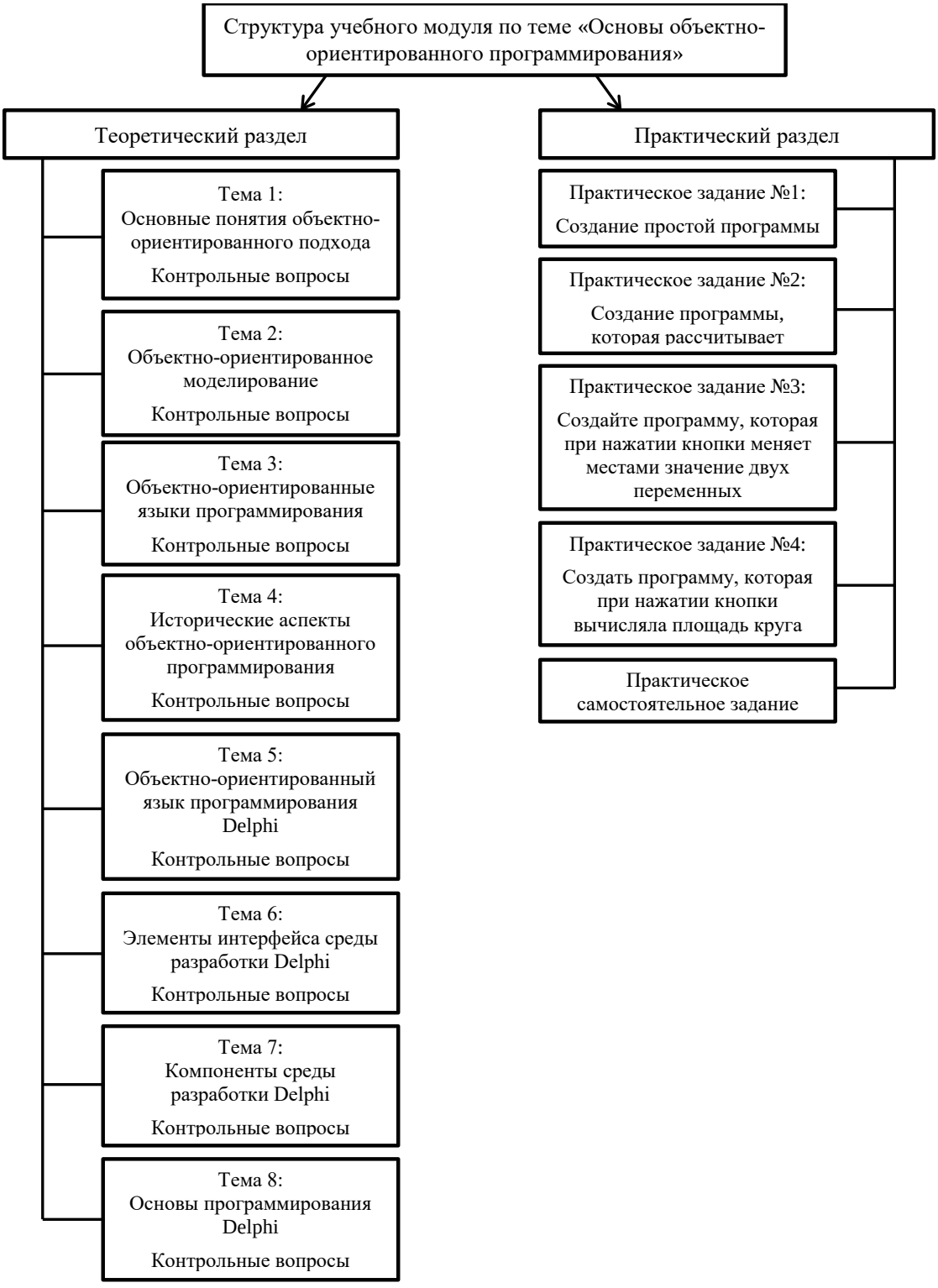

Рисунок 5 — Структура учебного модуля

Контроль хода и результатов обучения является одним из наиболее важных элементов учебного процесса. Организация контроля и методика последующего анализа его результатов оказывают существенное влияние, с одной стороны, на мотивацию заниматься, с другой стороны, по результатам контроля вырабатываются воздействия, корректирующие ход обучения и обеспечивающие оптимальное достижение цели.

В модуле реализована трехуровневая система контроля знаний, в которую входят:

- контрольные вопросы для закрепления изученного материала после каждого теоретического раздела;
- контрольные после каждого практического занятия;
- контрольное задание.

В контрольном задании обучающимся нужно будет самостоятельно разработать приложение на заданную тему. После чего преподавать проверяет, как обучающийся усвоил тему учебного модуля.

2.3 Обоснование программной среды разработки учебного модуля

Для электронного представления учебного модуля был выбран PDF-формат, так как основной акцент в выпускной квалификационной работе был сделан на отборе, структурировании и представлении содержания учебного модуля.

PDF-формат — это возможность надежного предоставления и обмена документов, независимо от программного и аппаратного обеспечения или операционной системы. Документ в PDF-формате может содержать шрифты, графику, мультимедийные элементы, что гарантирует правильное отображение независимо от операционной системы, программного обеспечения и пользовательских настроек конкретного компьютера [7]. Документ PDF-формата имеет следующие преимущества:

1 *Кроссплатформенность.* Документ содержит в себе необходимые для правильного отображения элементы и выглядит одинаково на любой платформе и в любом приложении.

2 *Компактность.* Различные алгоритмы компрессии (архивации) позволяют эффективно сжимать как текст, так и графику.

3 *Интерактивность.* В PDF файле можно использовать мультимедийные средства, такие как: видео, аудио ролики. Также активно используются гиперссылки, формы, данные из которых хранятся во внешних базах, данных.

4 *Безопасность.* Формат поддерживает многоуровневый механизм защиты и проверки подлинности. Есть возможность установить пароль на просмотр/редактирование, создать электронную подпись для идентификации автора [17].

Этот формат в настоящее время успешно использует:

1. Конвертации PostScript-файлов в PDF-формат с одновременным уменьшением их в десятки раз в результате использования специальных алгоритмов сжатия (компрессии).

2. Быстрая передача клиенту эскизов в PDF-формате, например, по электронной почте с максимальным сжатием и минимальным разрешением порядка 72 dpi.

PDF-файл может быть создан из файлов с расширением DOC, используя при этом Microsoft Office Word.

PDF-файл может содержать, помимо общепринятых, также несколько специальных вариантов навигации (ориентировки):

1. Посредством кнопок и гипертекстовых ссылок на страницах.
2. Посредством закладок.
3. С помощью статей, когда информация может подразделяться на несколько параллельных или пересекающихся тем, а не просто излагаться страница за страницей, как в обычной книге.

PDF-формат способен содержать: контурную или векторную графику (встраивается); пиксельную или растровую графику (встраивается); формат MPEG (ссылка на видеофрагменты). Встраиваемую графику в программах-генераторах PDF-файлов можно сжимать [7]**.**

Для электронного представления учебного модуля была выбрана программа Adobe Reader, которая используется для просмотра файлов PDF-формата. Adobe Reader создан разработчиками данного формата и поэтому позволяет работать с файлами любой сложности, а также поддерживает все его спецификации. Кроме того, данная программа абсолютно бесплатна. Программа Adobe Reader позволяет осуществлять с документами PDF только два действия — просматривать и распечатывать. Естественно, можно использовать все функциональные возможности — масштабировать листы при просмотре, осуществлять поиск информации или предварительный просмотр перед печатью. Но, по сути, программа Adobe Reader — удобное средство для чтения документов PDF в электронном или печатном виде [**Ошибка! Источник ссылки не найден.**].

Возможности Adobe Reader:

- поддержка огромного количества форматов;
- создание аннотаций к файлам;
- работа с цифровыми подписями;
- многоязычный интерфейс (включая русский);
- удобное создание форм для последующего заполнения;
- удобное чтение документов.

В программой Adobe Reader очень просто и комфортно работать. Для этого не нужно каких-то специальных умений и навыков. Новичок вполне способен разобраться с функциональностью Adobe Reader за несколько минут. Графический интерфейс программы Adobe Reader представлен на рисунке 6.

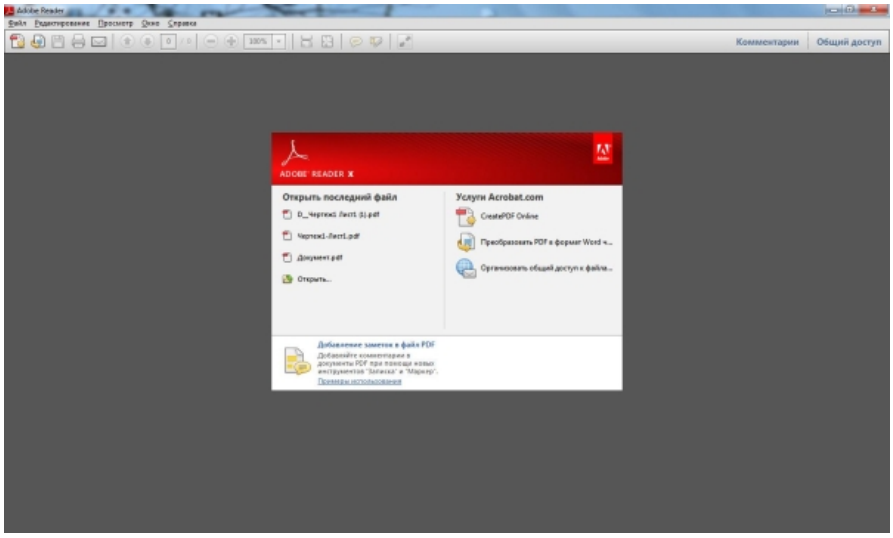

Рисунок 6 — Графический интерфейс программы Adobe Reader

Панель навигации, расположена слева, можно использовать «Эскиз», для перемещения позиции на текущей странице, а также можно использовать «Закладки», для перемещения по содержанию документа. На рисунке 7 продемонстрирована панель навигации.

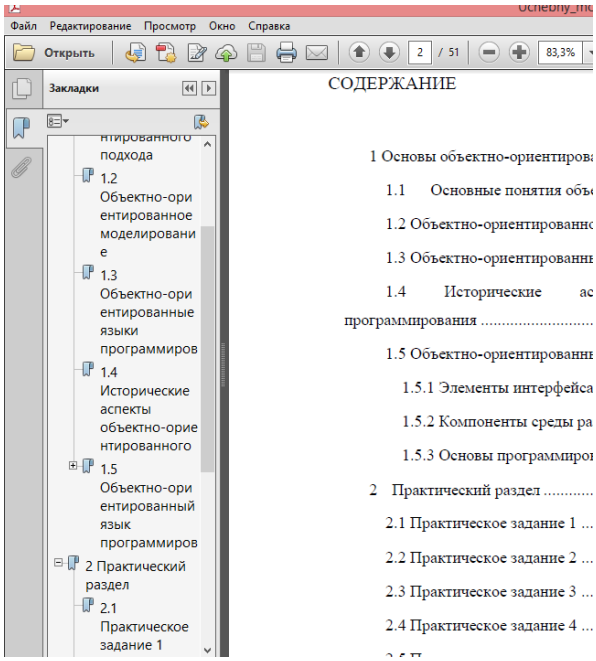

Рисунок 7 — Панель навигации

Также для удобства навигации в Microsoft Office Word были созданы кнопки с гиперссылками, которые позволяют переходить к содержанию, к следующей главе и к предыдущей (рисунок 8).

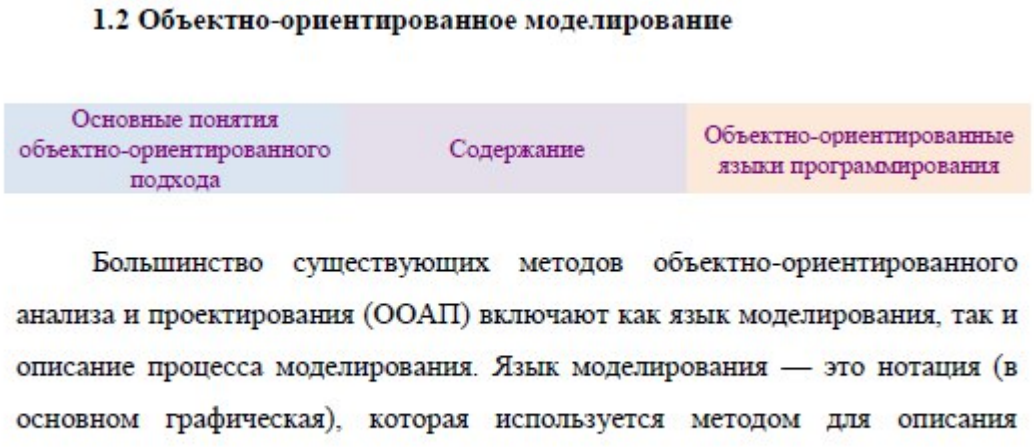

Рисунок 8 — Навигация гиперссылками

Таким образом, используя возможности Microsoft Office Word, был сформирован электронный формат учебного модуля, содержание которого представлено на рисунке 9.

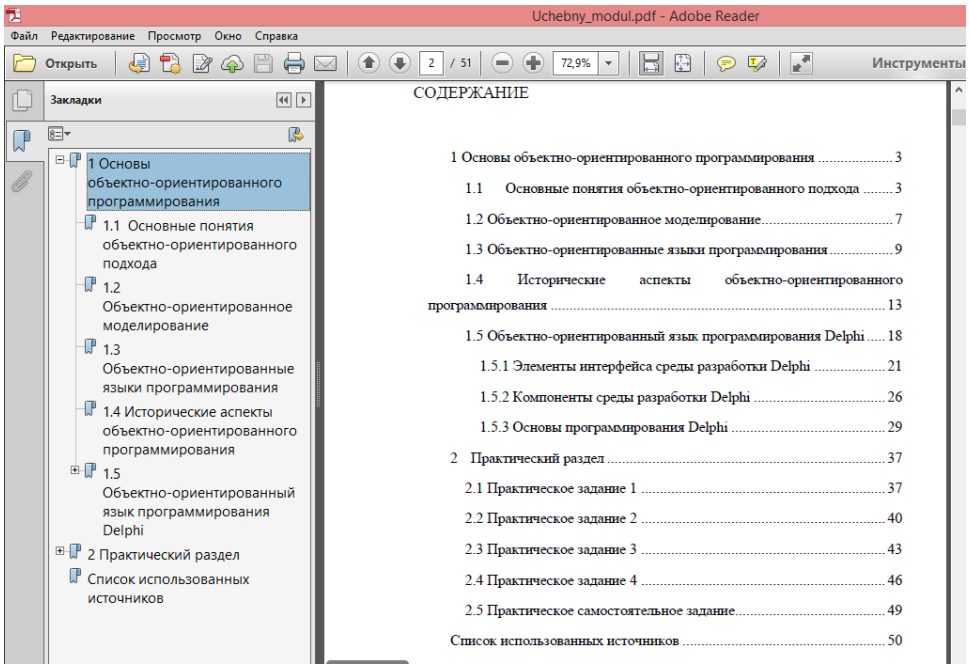

Рисунок 9 — Содержание учебного модуля

2.4 Описание содержания теоретического раздела учебного модуля

В первой главе теоретического раздела «Основные понятия объектно-ориентированного подхода» описываются такие понятия как: объектно-

ориентированный подход, объект, класс, элементы объектно-ориентированного подхода. Кроме того представлен пример наследования объектов.

Во второй главе теоретического раздела «Объектно-ориентированное моделирование» описываются следующие понятия: язык моделирования, унифицированный язык моделирования UML. В данной представлены цели разработки UML.

В третьей главе теоретического раздела «Объектно-ориентированные языки программирования» описываются следующие понятия: объектно-ориентированные языки программирования, объектно-ориентированное программирование. Кроме того подробно рассмотрены некоторые объектно-ориентированные языки программирования и дана сравнительная характеристика.

В четвертой главе теоретического раздела «Исторические аспекты объектно-ориентированного программирования» рассмотрена история объектно-ориентированных языков программирования и представлены основные идеи, заложенные в них.

В пятой главе теоретического раздела «Объектно-ориентированный язык программирования Delphi» описывается следующие понятия: визуальное программирование, интегрированная среда разработки Delphi. Так же представлена классификация визуального программирования и сравнительна характеристика Delphi и Turbo Pascal. Кроме того пятая глава содержит в себе три под главы:

1. Элементы интерфейса среды разработки Delphi.
2. Компоненты среды разработки Delphi.
3. Основы программирования Delphi.

В под главах подробно описываются элементы интерфейса среды разработки Delphi, стандартные компоненты.

На рисунке 10 представлен PDF фрагмент первой главы теоретического раздела. На рисунке продемонстрирована тема первой главы, фрагмент тео-

ретической части первой главы, а также показана структура учебного модуля расположенного в панели навигации расположенная с левой стороны.

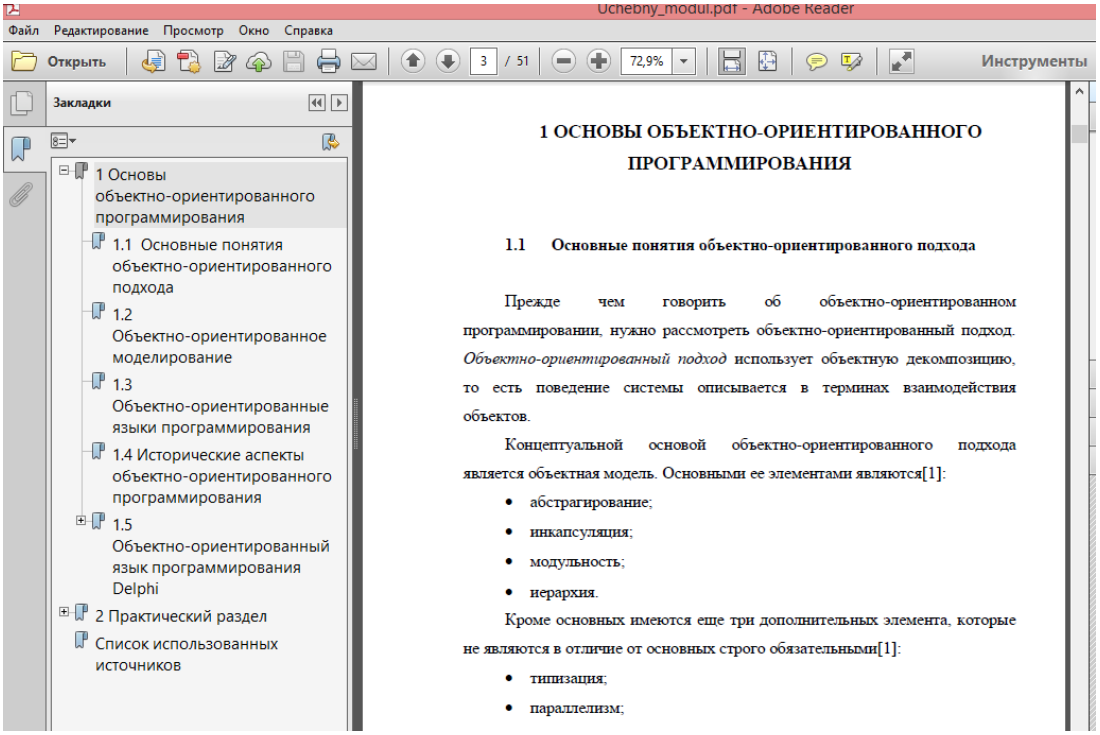

Рисунок 10 — Фрагмент первой главы теоретического раздела

К каждой главе сформулированы контрольные вопросы для самостоятельной проверки уровня усвоенного материала. Контрольные вопросы к первому параграфу представлены на рисунке 11.

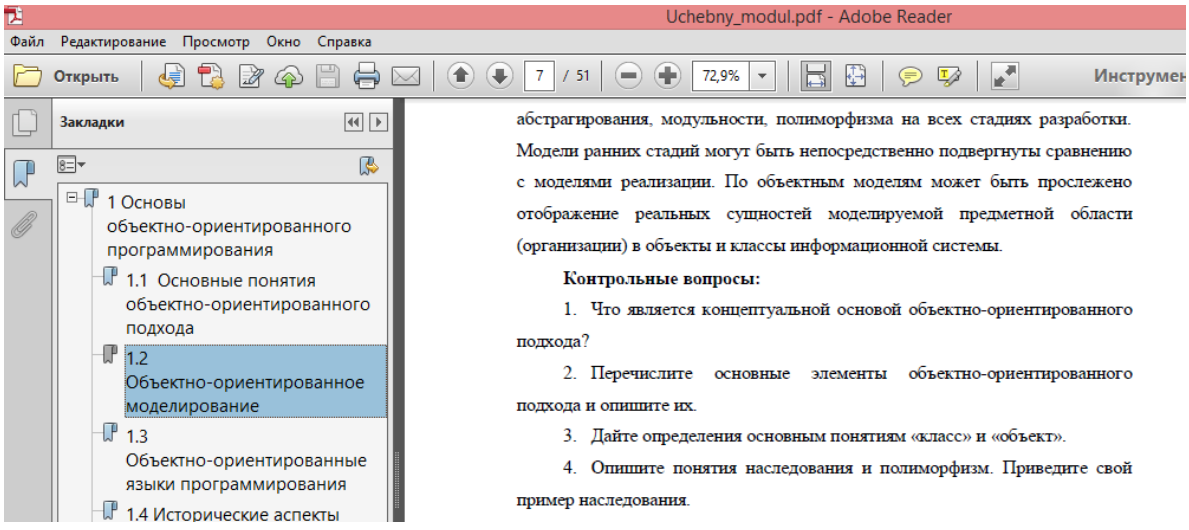

Рисунок 11 — Контрольные вопросы первой главы

2.5 Описание содержания практического раздела

В практическом разделе учебного модуля представлены 4 практических задания и одно контрольное задание, также после каждого практического задания предусмотрены контрольные вопросы.

В первом практическом задании обучающиеся знакомятся с интегрированной средой разработки Delphi. Создают самое простое приложение (рисунок 12).

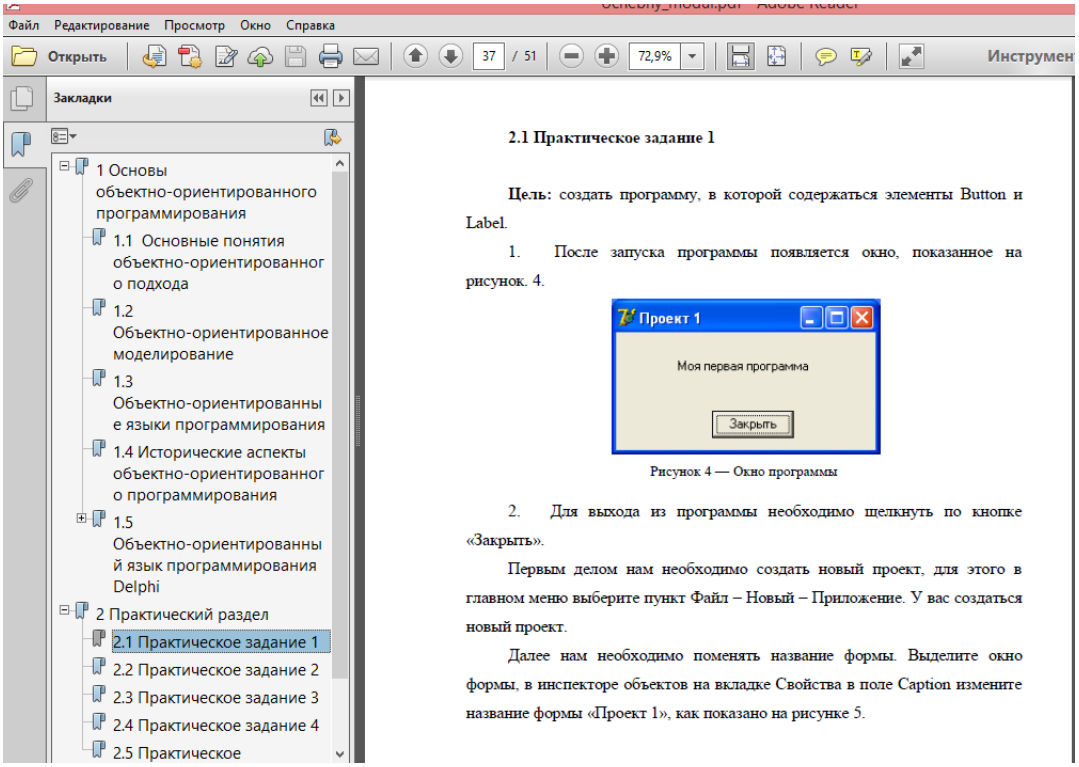

Рисунок 12 — Практическое задание №1

Во втором практическом задании обучающимся необходимо создать программу, которая рассчитывает расстояние и знакомятся с функциями IntToStr и StrToInt. StrToInt — преобразует текс в целое число (integer). IntToStr — преобразует целое число(integer) в текс (рисунок 13).

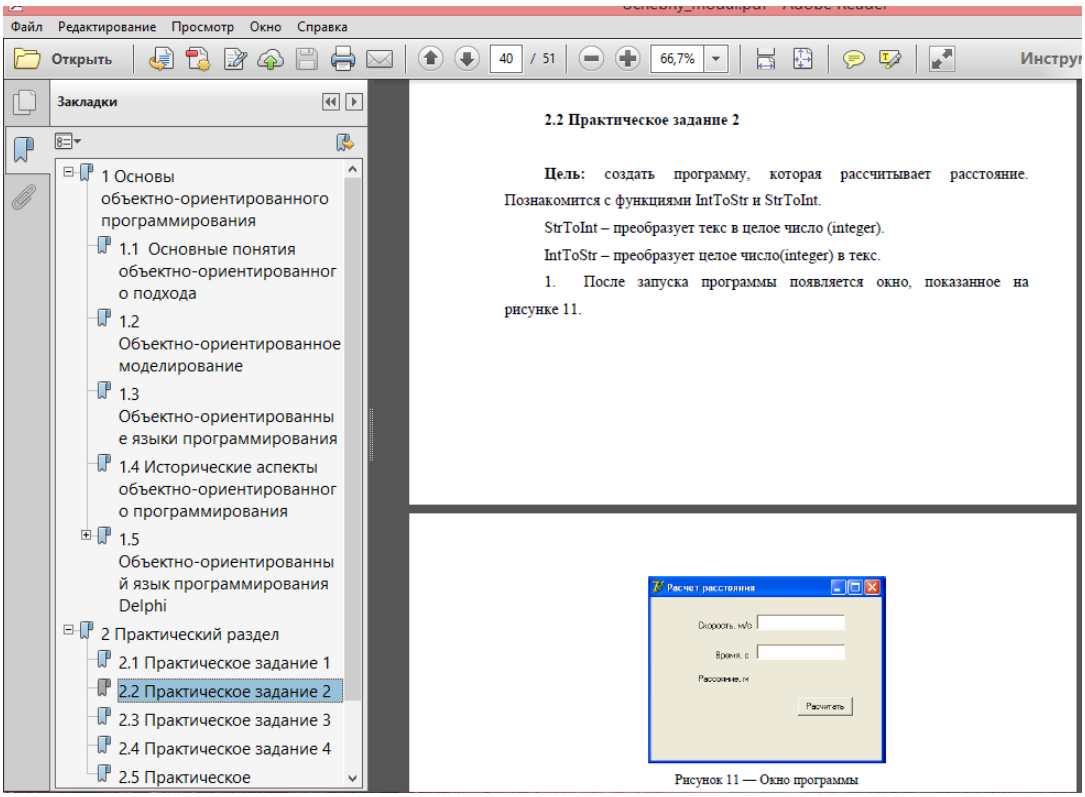

Рисунок 13 — Практическое задание №2

В третьем практическом задании обучающие создаю программу, которая меняет местами две переменные (рисунок 14).

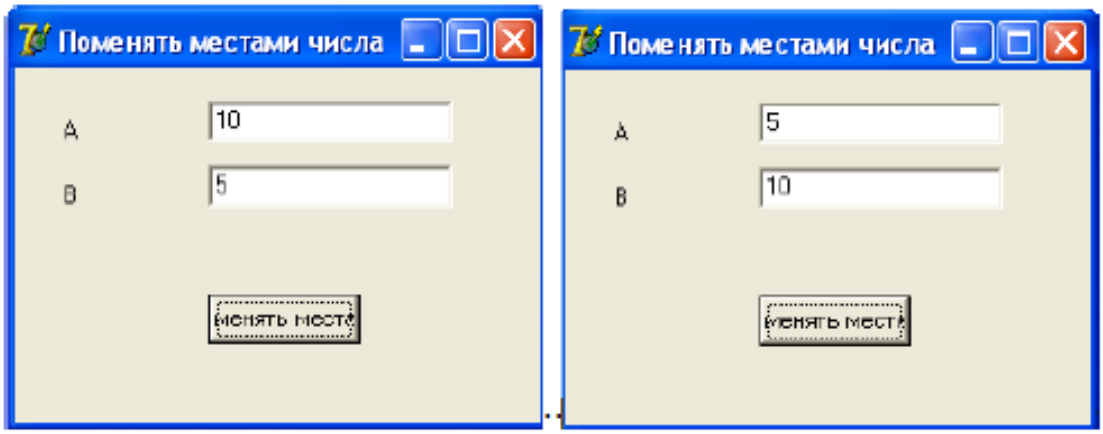

Рисунок 14 — Готовая форма к практическому заданию №3

В четвертом практическом задании обещающие создают программу, которая рассчитывает площадь круга (рисунок 15).

2.4 Практическое задание 4

Цель: создать программу, которая при нажатии кнопки вычисляло площадь круга.

Цель: Создайте приложение, которое при нажатии кнопки меняет местами значение двух переменных.

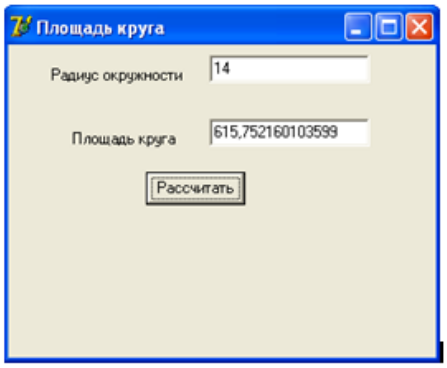

Рисунок 21 — Готовая программа

Рисунок 15 — Практическое здание №4

Для определения усвоения знания предусмотрено контрольное практическое задание (рисунок 16).

2.5 Практическое самостоятельное задание

Создать программу, которая будет выполнять следующие действия:

1. После запуска программы появляется надпись «Брось кубик»(рисунок 22).

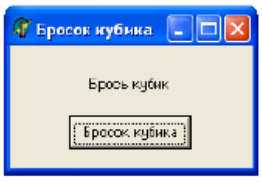

Рисунок 22 — Форма после запуска программы

2. По щелчку мышью на кнопке «Бросок кубика» появляется сообщение, выдающее число в диапазоне 0 – 6 (рисунок 23).

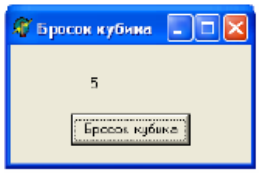

Рисунок 23 — Форма после нажатия на кнопку

Рисунок 16 — Контрольное практическое задание

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы был разработан учебный модуль по теме «Основы объектно-ориентированного программирования». Учебный модуль разработан в качестве дополнительного материала, который может использоваться в самостоятельной работе обучающихся, изучающие основы программирования. В результате выполнения учебного модуля обучающиеся получают базовые знания по объектно-ориентированному программированию.

В результате тщательного сбора, анализа соответствующего материала, его методической переработки была разработана структура учебного, которая составляет два основных раздела: теоретический и практический. В теоретический раздел учебного модуля входят 8 глав:

1. Основные понятия объектно-ориентированного подхода.
2. Объектно-ориентированное моделирование.
3. Объектно-ориентированные языки программирования.
4. Исторические аспекты объектно-ориентированного программирования.
5. Объектно-ориентированный язык программирования Delphi.
6. Элементы интерфейса среды разработки Delphi.
7. Компоненты среды разработки Delphi.
8. Основы программирования Delphi.

Практические задания были разработаны в качестве дополнительного учебного материала, который может использоваться в самостоятельной работе обучающихся СПО, изучающих основы программирования.

Для достижения цели были выполнены следующе задачи:

1. Рассмотрены теоретические основы объектно-ориентированного программирования.
2. Разработана структура учебного модуля.

3. Отобрано содержание теоретического раздела.

4. Разработаны практические задания по объектно-ориентированному языку программирования Delphi.

5. Создан электронный вариант учебного модуля.

Для электронного представления учебного модуля был выбран PDF-формат, так как основной акцент в дипломной работе был сделан на отборе, структурировании и представлении содержания учебного модуля.

Цель работы достигнута, поставленные задачи решены.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Арлоу Дж. UML 2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование [Текст]: / Арлоу Дж., Нейштадт А. — СПб. Символ-Плюс, 2007.

2. Архангельский А.Я. Программирование в Delphi 7 [Текст]: учебник / А.Я. Архангельский — М.: ООО «Бином-Пресс», 2003 г. — 1152 с.: ил.

3. Бобровский С. И. Delphi 7. Учебный курс [Текст]: учебник / С. И. Бобровский. — СПб.: Питер, 2004. — 736 С.: ил.

4. Вальвачев А.Н Программирование на языке Delphi [Электронный ресурс]. — Режим доступа: http://rsdn.ru/article/delphi/delphi_7_07.xml (дата обращения: 05.06.2016).

5. Вендров А.М. Проектирование программного обеспечения экономических информационных систем [Текст]: учебник / А.М. Вендров М.: — Финансы и статистика, 2002.

6. Визуальное программирование [Электронный ресурс]. — Режим доступа: http://life-prog.ru/view_zam2.php?id=162 (дата обращения: 08.06.2016).

7. Иванова А. Формат .PDF [Электронный ресурс] — Режим доступа: http://www.kakprosto.ru/kak-98055-kak-perevesti-doc-fayl-v-pdf (дата обращения: 16.06.2016).

8. Иванова Г.С. Объектно-ориентированное программирование [Текст]: учебник / Г. С. Иванова, Т. Н. Ничушкина. — Москва: Изд-во МГТУ, 2014. — 456 с.: ил.

9. История объектно-ориентированного программирования [Электронный ресурс] — Режим доступа: http://www.pcweek.ru/infrastructure/article/detail.php?ID=64959 (дата обращения: 01.06.2016).

10. Колесов Ю. Б. Моделирование систем. Объектно-ориентированный подход [Текст]:Учебное пособие / Ю. Б. Колесов, Ю. Б. Сениченков. — СПб.: БХВ-Петербург, 2012. — 192 с.: ил.

11. Керман М Программирование и отладка в Delphi. [Текст]: учебный курс / Керман М. Программирование и отладка в Delphi — М.: Издательский дом "Вильямс", 2002. — 672 с.

12. Культин Н. Б. Основы программирования в Delphi 7 [Текст]: учебник / Н. Б. Культин — СПб.: БХВ-Петербург, 2003. — 608 с.: ил.

13. Кьоу Джим Объектно-ориентированное программирование [Текст]: учебник / Джим Кьоу, Марио Джеанини Объектно-ориентированное программирование 7 — СПб.: Питер, 2005. — 240 с.

14. Максудова Л.Г. Разработка и построение учебных модулей для системы дистанционного обучения [Текст]: Методическое пособие / Максудова Л.Г., Литвиненко М.В., Абросимов В.В — М.: МИИГАиК, 2006. — 59 с.

15. Мельников С. В. Delphi и Turbo Pascal на занимательных примерах [Текст]: учебник / С. В. Мельников — СПб.: БХВ-Петербург, 2006. — 448с.

16. Модульная форма обучения [Электронный ресурс]. — Режим доступа: http://fulledu.ru/articles/mba/formy-obucheniya/article/816_chto-takoe-modulnaya-forma-obucheniya.html (дата обращения: 10.06.2016).

17. Новицкого Ф. Функции.PDF [Электронный ресурс]. — Режим доступа: http://pdf-reader.ru/programms.html (дата обращения: 16.06.2016).

18. Обзор языков программирования [Электронный ресурс]. — Режим доступа: http://bourabai.ru/alg/classification04.htm (дата обращения: 05.06.2016).

19. Объектно-ориентированное моделирование Delphi [Электронный ресурс]. — Режим доступа: http://studopedia.info/1-75616.html (дата обращения: 02.06.2016).

20. Объектно-ориентированные языки программирования [Электронный ресурс]. — Режим доступа: http://www.newreferat.com/ref-22729-7.html (дата обращения: 05.06.2016).

21. Онлайн учебник по Delphi 7 [Электронный ресурс]. — Режим доступа: http://delphi.support.uz/ (дата обращения: 17.06.2016).

22. Основы программирования [Электронный ресурс]. — Режим доступа: http://www.studfiles.ru/preview/3004491/ (дата обращения: 08.06.2016).

23. Рамбо Дж. UML 2.0. Объектно-ориентированное моделирование и разработка [Текст]: / Рамбо Дж., Блаха М. — 2-е изд.: Пер. с англ. — СПб.: Питер, 2006.

24. Технология модульного обучения. [Электронный ресурс]. — Режим доступа: http://studopedia.su/4_18396_tehnologiya-modulnogo-obucheniya.html (дата обращения: 17.06.2016).

25. Тузовский А. Ф. Объектно-ориентированное программирование [Текст]: учебное пособие / А. Ф. Тузовский Объектно-ориентированное программирование — Изд.: Юрайт, 2016. — 208 с.

26. Фаулер М. Основы UML. [Текст]: / Краткое руководство по стандартному языку объектного моделирования/ Фаулер М. — 3-е издание..: Пер. с англ. — СПб: Символ-Плюс, 2005.

27. Финогенов К.Г Основы объектно-ориентированного программирования. Лабораторный практикум [Текст]: Учебное пособие. М.:МИФИ, 2008. — 92с.

28. Фленов М.Е. Библия Delphi [Текст]: / М.Е. Фленов — 3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2011. — 688 с.

29. . Яковлев А. Тонкости Adobe: чем Acrobar отличается от Reader [Электронный ресурс] – Режим доступа. http://pdf.cnews.ru/doc/article_doc_14.shtml (дата обращения: 17.06.2016).

30. Adobe Reader [Электронный ресурс]. — Режим доступа: http://elhow.ru/programmnoe-obespechenie/raznye-voprosy-pro-po/dlja-chego-adobe-reader (дата обращения: 16.06.2016).

ПРИЛОЖЕНИЕ

Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Российский государственный профессионально-педагогический университет»

Институт инженерно-педагогического образования
Кафедра информационных систем и технологий
направление 44.03.04 Профессиональное обучение (по отраслям)
профиль «Энергетика»
профилизация «Компьютерные технологии автоматизации и управления»

УТВЕРЖДАЮ
Заведующий кафедрой
_____Н. С. Толстова
«_____»____2016 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра

студентки 4 курса, группы КТэ-401 Отрутиковой Кристины Владимировны
1. Тема Учебный модуль «Основы объектно-ориентированного программирования» утверждена распоряжением по институту от 28.03.2016 г. № 57
2. Руководитель Телепова Татьяна Петровна, старший преподаватель кафедры ИС
3. Место преддипломной практики Екатеринбургский экономико-технологический колледж
4. Исходные данные к ВКР:
К.Г Финогенов «Основы объектно-ориентированного программирования.», Г. С. Иванова, Т. Н. Ничушкина «Объектно-ориентированное программирование».
5. Содержание текстовой части ВКР (перечень подлежащих разработке вопросов)
Теоретический раздел;
Основы объектно-ориентированного программирования;
Объектно-ориентированное моделирование;
Объектно-ориентированный язык программирования Delphi;
Практический раздел.

6. Перечень демонстрационных материалов
Презентация, созданная в PowerPoint 2010

7. Календарный план выполнения выпускной квалификационной работы

№ п/п	Наименование этапа дипломной работы	Срок выполнения этапа	Процент выполнения ВКР	Отметка руководителя о выполнении
1	Сбор информации по выпускной работе и сдача зачета по преддипломной практике	*25.04.2016*	*15*	
2	Выполнение работ по разрабатываемым вопросам их изложение в выпускной работе:			
	Выполнение и оформление теоретического раздела ВКР	*01.05.2016*	*20*	
	Работа над практическим разделом	*10.05.2016*	*10*	
	Выполнение и оформление практического раздела ВКР	*22.05.2016*	*10*	
	Работа над методическим разделом	*01.06.2016*	*15*	
	Выполнение и оформление методического раздела	*10.06.2016*	*10*	
3	Оформление текстовой части ВКР	*15.06.2016*	*5*	
4	Выполнение демонстрационных материалов к ВКР	*20.06.2016*	*5*	
5	Нормоконтроль	*24.06.16*	*5*	
6	Подготовка доклада к защите в ГЭК		*5*	

8. Консультанты по разделам выпускной квалификационной работы

Наименование раздела	Консультант	Задание выдал		Задание принял	
		подпись	дата	подпись	дата
Методическая часть					
Нормоконтроль					
Предварительная защита					

Руководитель _____ *Задание получил* _____
подпись дата подпись студента дата

9. Выпускная квалификационная работа и все материалы проанализированы.
Считаю возможным допустить Отрутикову К.В. к защите выпускной квалификационной работы в государственной экзаменационной комиссии.

Руководитель _____ _____
подпись дата

10. Допустить Отрутикову К.В. к защите выпускной квалификационной работы в государственной экзаменационной комиссии (протокол заседания кафедры от 08.06.2016 № 15)

Заведующий кафедрой _____ _____
подпись дата