МИНОБРНАУКИ РОССИИ
Ярославский государственный университет им. П.Г. Демидова

Кафедра вычислительных и программных систем

УТВЕРЖДАЮ
Декан факультета ИВТ
_____Д.Ю. Чалый
«_23_»____мая_____2023 г.

Рабочая программа дисциплины
«Разработка программных приложений для ОС Аврора»

Направление подготовки
01.03.02 Прикладная математика и информатика

Профиль
«Программирование и технологии искусственного интеллекта»

Квалификация выпускника
Бакалавр

Форма обучения
очная

Программа рассмотрена
на заседании кафедры
от 21 апреля 2023 г.,
протокол № 8

Программа одобрена НМК
факультета ИВТ
протокол № 6 от
28 апреля 2023 г.

Ярославль

1. Цели освоения дисциплины

Целями дисциплины «Разработка программных приложений для ОС Аврора» являются изучение современных платформ для разработки мобильных приложений на основе ОС Аврора, включенной в реестр отечественного программного обеспечения; изучение основных принципов построения пользовательских интерфейсов приложений для мобильных устройств; изучение основных принципов построения и особенностях современных инновационных мобильных сервисов; формирование представления о современном состоянии и проблемах построения мобильных сервисов; формирование способности к восприятию новых научных фактов и гипотез и использованию полученных знаний.

2. Место дисциплины в структуре ОП бакалавриата

Дисциплина «Разработка программных приложений для ОС Аврора» относится к вариативной части (дисциплина по выбору) ОП бакалавриата.

Содержание курса тесно связано фактически со всеми дисциплинами, которые изучались студентами. Освоению данной программы предшествуют учебные курсы по программированию и современным информационным технологиям.

Дисциплина «Разработка программных приложений для ОС Аврора» обеспечивает закрепление и углубление теоретических знаний и практических навыков по основным дисциплинам ИТ-цикла. Дисциплина способствует профессиональному росту студентов, повышению их общеметодологического уровня, а также дальнейшему развитию навыков научно- исследовательской деятельности.

3. Планируемые результаты обучения по дисциплине, соотнесенные с планируемыми результатами освоения ОП бакалавриата

Процесс изучения дисциплины направлен на формирование следующих элементов компетенций в соответствии с ФГОС ВО, ОП ВО и приобретения следующих знаний, умений, навыков и (или) опыта деятельности:

Формируемая компетенция (код и формулировка)	**Индикатор достижения компетенции (код и формулировка)**	**Перечень планируемых результатов обучения**
Профессиональные компетенции		
ПК-3 Способен к разработке стратегии тестирования и управлению процессом тестирования	ПК-3.2 Умеет использовать методы проектирования и разработки программных продуктов	Знать: ● принципы построения программных приложений; ● особенности разработки приложений с использованием Qt QML для платформы Аврора. Уметь: ● проектировать мобильные приложения в соответствии с требованиями пользователя; ● разрабатывать приложения для платформы Аврора. Владеть навыками: ● разработки

		приложений в среде QtCreator; ● работы с эмулятором мобильного устройства; ● тестирования приложений.

4. Объем, структура и содержание дисциплины

Общая трудоемкость дисциплины составляет 4 зач. ед., 144 акад. час.

№ п/п	Темы (разделы) дисциплины, их содержание	Семестр	Виды учебных занятий, включая самостоятельную работу студентов, и их трудоемкость (в академических часах)						Формы текущего контроля успеваемости Форма промежуточной аттестации (по семестрам)
			Контактная работа						
			лекции	практические	лабораторные	консультации	аттестационные испытания	самостоятельная работа	
1.	Введение в программирование для мобильных устройств	7	1	3				1	
2.	Инструменты разработки мобильных приложений	7	1	3				2	
3.	Основы конструирования интерфейсов пользователя	7	1	3				2	
4.	Типовые элементы интерфейса на мобильных устройствах	7	1	3		2		2	Индивидуальные задания
5.	Средства организации интерфейса мобильных приложений	7	1	3				2	
6.	Управление данными и использование шаблона MVC	7	1	3				2	

7.	Интерфейс пользователя как граф состояний	7	1	3				2	
8.	Интеграция QMl-интерфейсов и логики приложения на языке C++	7	1	3		2		2	Индивидуальные задания
9.	Управление подключением к сети	7	1	3				2	
10.	Использование типовых сервисов мобильной платформы	7	1	3				2	
11.	Графика и мультимедиа	7	1	3				2	
12.	Использование сенсоров	7	1	3				2	
13.	Сервисы геопозиционирования и дополненная реальность	7	2	6				2	
14.	Управление качеством при разработке мобильных приложений	7	2	6		2		2	Индивидуальные задания
15.	Система управления пакетами и распространение приложений	7	3	6		1		2	
	Всего за 7 семестр		**18**	**54**		**7**	**34**	**29**	**Экзамен**
	Всего		**18**	**54**		**7**	**34**	**29**	

Содержание разделов дисциплины:

1. Введение в программирование для мобильных устройств

Современная мобильная ОС с точки зрения пользователя на примере ОС Аврора: история развития, поддерживаемые устройства, интерфейс пользователя и система жестов, многозадачность, безопасность, персонализация и другие возможности. Обзор отличий мобильных устройств от настольных и переносных персональных компьютеров и обусловленных этими различиями особенностей мобильного программирования: архитектурные отличия используемых платформ (Intel/ARM); управление питанием и требования к энергозатратам системы и приложений, основные способы увеличения срока автономной работы; разнообразие размеров экранов и разрешений, и требования к организации пользовательского интерфейса (в частности проблемы абсолютного позиционирования элементов, адаптация к ориентации экрана и ее изменению), типовые элементы управления мобильных интерфейсов; сенсорные интерфейсы, управление жестами и экранные клавиатуры; система встроенных сенсоров и реализация обратной связи; вопросы безопасности. Типовой цикл подготовки приложения в интегрированной среде к запуску в эмулируемой среде или на целевом устройстве на примере простого шаблонного приложения.

2. Инструменты разработки мобильных приложений

Обзор инструментальных средств и использование комплекта разработки (SDK): интегрированная среда разработки на примере QtCreator, режимы и панели, их

назначение; средства кросс-компиляции на примере Mer build engine, эмулятор устройства на примере ОС Аврора Emulator для системы виртуализации VirtualBox, ограничения эмулятора по сравнению с реальным устройством. Процесс сборки на примере простейшего приложения, ключевые понятия (tool, toolchain, cross-compilation, compilation target). Архитектура мобильной платформы, доступные программные интерфейсы, место и роль фреймворка Qt и технологии QtQuick в разработке приложений. Обзор концепций декларативного описания интерфейсов на примере шаблона простейшего приложения на языке QML: визуальные невизуальные элементы и их свойства; доступные типы свойств; присваивание и привязка свойств; древовидная структура описания графического интерфейса; иерархия элементов и наследование свойств; события и их обработка, императивный код на языке JavaScript в обработчиках событий, обзор доступных модулей элементов, в т.ч., Silica.

3. Основы конструирования интерфейсов пользователя

Базовый элемент Item и основные визуальные элементы (прямоугольник, изображение, текст), их основные свойства. Геометрия элементов. Системы координат и абсолютное позиционирование. Относительное позиционирование, контейнеры элементов. Порядок отрисовки и z-координата. Масштабирование и поворот элементов. Анимация свойств элементов. Организация диалога с пользователем, обработка событий ввода, в том числе жестов.

4. Типовые элементы интерфейса на мобильных устройствах

Типовые элементы управления и их мобильные аналоги на примере элементов Qt Quick Controls и Silica: надписи, кнопки, переключатели, текстовые поля, инструменты выбора, индикаторы прогресса и занятости и т. п.

5. Средства организации интерфейса мобильных приложений

Средства организации мобильных приложений на примере элементов Qt Quick Controls и Silica: окна, системы меню, диалоги, средства организации многостраничных приложений и навигации по страницам. Создание обложки приложения для отображения на рабочем столе.

6. Управление данными и использование шаблона MVC

Объединение и группировка элементов; шаблон проектирования MVC, создание моделей данных и визуальных элементов, отображений с помощью представлений. Поддержка форматов JSON и XML. Получение данных от сервисов сети интернет на примере XMLListModel. Построение моделей данных с использованием источников с SQL-доступом, в частности, хранение данных в локальной базе SQLite. Использование Loca Storage API и элементов для хранения настроек приложения.

7. Интерфейс пользователя как граф состояний

Средства описания состояния в виде набора значений свойств составляющих его элементов. Именование состояний и декларативное определение логики переходов. Определение переходов (например, анимаций). Создание собственных элементов на языке QML, реализация свойств, методов (в виде функций Javascript) и сигналов событий.

8. Интеграция QMl-интерфейсов и логики приложения на языке C++

Основы использования фреймворка Qt и Qt C++. Метаобъектная система, свойства и интроспекция, сигналы и слоты. Контейнерные классы. Создание доступных в QMLсцене методов, объектов и классов Qt C++, создание подключаемых элементов и модулей на Qt C++.

9. Управление подключением к сети

Сетевые возможности мобильных устройств и проблемы, связанные с мобильностью пользователя. Управление подключением к сети. Обзор технологий ОС Linux, обеспечивающих управление соединениями и доступ к средствам обмена данными, в том числе сетей сотовой связи, Bluetooth и NFC. Элементы QML для обеспечения обмена данными. Пример получения данных от умного браслета с использованием Bluetooth-соединения.

10. Использование типовых сервисов мобильной платформы
Управление контактами, обменом сообщениями, событиями календаря, встроенным веб-браузером. Домашний экран. Встраивание компонента браузера в приложения. Гибридные приложения на основе Qt WebChannel.
11. Графика и мультимедиа
Графический холст Canvas. Элементы Sprite и SpriteSequence. Системы частиц Particles. Элементы для встраивания мультимедийного содержимого (проигрывания аудио и видео. Иллюстрация предоставляемых мобильной платформой средств на примере разработки игрового приложения.
12. Использование сенсоров
Обзор типовых встроенных сенсоров мобильных устройств и примеры доступных посредством Bluetooth носимых датчиков. Получение данных сенсоров на примере датчиков освещения, близости и ориентации. Использование акселерометра на примере Qt Sensors и элемента Accelerometer языка QML.
13. Сервисы геопозиционирования и дополненная реальность
Основы функционирования средств геопозиционирования GPS/ГЛОНАСС: спутниковая группировка, способ определения местоположения на основе данных спутников, реализация обработки данных от модуля позиционирования (координат, высоты над уровнем моря, даты и времени, скорости, отклонения от северного направления) на примере Qt Positioning API и соответствующих классов QML. Использование камеры на примере Qt Multimedia и элемента Camera языка QML. Идея дополненной реальности, совместное использование данных позиционирования, компаса, акселерометра и камеры для реализации простого сценария дополненной реальности.
14. Управление качеством при разработке мобильных приложений
Обзор типовых ошибок и их решений, предпочтительная практика программирования. Соглашения о стиле кодирования для Qt и QML, дополнительные рекомендации Аврора. Вопросы потребления памяти и производительности: выделение памяти в QML-приложении и рекомендации по программированию, способствующие эффективному управлению памятью, использование средств Valgrind, применение Memcheck и Callgrind; обзор типичных ресурсоемких операций для элементов QML и способы уменьшения вычислительной нагрузки, использование CPU Usage Analyzer. Оптимизация QML-кода, использование профайлера. Применение средств статического анализа кода на примере Clang Static Analyzer. Модульное тестирование приложений QML.
15. Система управления пакетами и распространение приложений
Управление пакетами приложений в ОС Linux и мобильных Linux: форматы и менеджеры пакетов, репозитории. Создание RPM-пакета приложения и установка на устройстве. Использование OBS. Распространение приложений через магазин приложений на примере Harbour: регистрация, публикация приложения, взаимодействие с QA, использование инструментов Dashboard.

5. Образовательные технологии, используемые при осуществлении образовательного процесса по дисциплине

В процессе обучения используются следующие образовательные технологии:

Вводная лекция – дает первое целостное представление о дисциплине и ориентирует студента в системе изучения данной дисциплины. Студенты знакомятся с назначением и задачами курса, его ролью и местом в системе учебных дисциплин и в системе подготовки в целом. Дается краткий обзор курса, история развития науки и практики, достижения в этой сфере, имена известных ученых, излагаются перспективные направления исследований. На этой лекции высказываются методические и организационные особенности работы в рамках данной дисциплины, а также дается анализ рекомендуемой учебно-методической литературы.

Лекция-беседа или «диалог с аудиторией», является наиболее распространенной и сравнительно простой формой активного вовлечения студентов в учебный процесс. Эта лекция предполагает непосредственный контакт преподавателя с аудиторией. Преимущество лекции-беседы состоит в том, что она позволяет привлекать внимание студентов к наиболее важным вопросам темы, определять содержание и темп изложения учебного материала с учетом особенностей студентов.

Мастер-класс – это особая форма учебного занятия, когда преподаватель-мастер передает свой опыт путем прямого и комментированного показа последовательности действий, методов, приемов и форм педагогической деятельности. Целью проведения мастер-класса является профессиональное, интеллектуальное и эстетическое воспитание студентов, и прежде всего, развитие в ходе мастер-класса способности студента самостоятельно и нестандартно мыслить.

Академическая лекция (или лекция общего курса) – последовательное изложение материала, осуществляемое преимущественно в виде монолога преподавателя. Требования к академической лекции: современный научный уровень и насыщенная информативность, убедительная аргументация, доступная и понятная речь, четкая структура и логика, наличие ярких примеров, научных доказательств, обоснований, фактов.

Практическое занятие – занятие, посвященное освоению конкретных умений и навыков и закреплению полученных на лекции знаний.

6. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень лицензионного программного обеспечения и информационных справочных систем (при необходимости)

В процессе осуществления образовательного процесса используются: для разработки документов, презентаций, для работы с электронными таблицами

OfficeStd 2013 RUS OLP NL Acdmc 021-10232

LibreOffice (свободное)

издательская система LaTeX;

– Среда разработки NetBeans 8.2 : www.netbeans.org. Доступ свободный

– OS Linux (свободная)

– для поиска учебной литературы библиотеки ЯрГУ – Автоматизированная библиотечная информационная система "БУКИ-NEXT" (АБИС "Буки-Next").

7. Перечень основной и дополнительной учебной литературы, ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

а) основная:

1.Лафоре Р. Объектно-ориентированное программирование в С++. / Р. Лафоре; [пер. с англ. А. Кузнецова, М. Назарова, В. Шрага] - 4-е изд. - СПб.: Питер, 2014. - 923 с.

2. Соколова В. В. Вычислительная техника и информационные технологии. Разработка мобильных приложений: учеб. пособие для прикладного бакалавриата. / В. В. Соколова; УМО вузов по университетскому политехническому образование; Томский политехнический ун-т - М.: Юрайт, 2016. - 175 с

б) дополнительная:

1. Лагутина, Н. С., С++. Примеры и задачи : практикум / Н. С. Лагутина ; Яросл. гос. ун-т, Ярославль, ЯрГУ, 2011, 47с

2. Лагутина, Н. С., С++. Примеры и задачи [Электронный ресурс] : практикум / Н. С. Лагутина ; Яросл. гос. ун-т, Ярославль, ЯрГУ, 2011, 47с. http://www.lib.uniyar.ac.ru/edocs/iuni/20110401.pdf

3. Лагутина Н. С. Основы объектно-ориентированного программирования на языке С++: учеб. пособие для вузов. / Н. С. Лагутина; Науч.-метод. совет ун-та ; Яросл. гос. ун-т им. П. Г. Демидова - Ярославль: ЯрГУ, 2008. - 107 с.

в) ресурсы сети «Интернет»

- QML Tutorial http://doc.qt.io/qt-5/qml-tutorial.html Доступ свободный
- Qt Documentation http://doc.qt.io/ Доступ свободный

8. Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине

Материально-техническая база, необходимая для осуществления образовательного процесса по дисциплине включает в свой состав специальные помещения:

-учебные аудитории для проведения занятий лекционного типа и практических занятий (семинаров);

- учебные аудитории для проведения групповых и индивидуальных консультаций,

- учебные аудитории для проведения текущего контроля и промежуточной аттестации;

-помещения для самостоятельной работы;

-помещения для хранения и профилактического обслуживания технических средств обучения.

Специальные помещения укомплектованы средствами обучения, служащими для представления учебной информации большой аудитории.

Для проведения занятий лекционного типа предлагаются наборы демонстрационного оборудования и учебно-наглядных пособий, хранящиеся на электронных носителях и обеспечивающие тематические иллюстрации, соответствующие рабочим программам дисциплин.

Помещения для лабораторных занятий и самостоятельной работы обучающихся оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду организации.

Число посадочных мест в лекционной аудитории больше либо равно списочному составу потока, а в аудитории для практических занятий (семинаров) – списочному составу группы обучающихся.

Автор(ы) :

Старший преподаватель кафедры ВПС_____ А.М.Васильев

Приложение №1 к рабочей программе дисциплины
«Разработка программных приложений для ОС Аврора»
Фонд оценочных средств
для проведения текущей и промежуточной аттестации студентов
по дисциплине

1. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

1.1. Контрольные задания и иные материалы, используемые в процессе текущей аттестации

Типовые индивидуальные задания

1. Написать программу, реализующую шифрование и дешифрование, применяя шифр Цезаря. Шифр Цезаря — это вид шифра подстановки, в котором каждый символ в тексте заменяется символом, находящимся на некотором постоянном числе позиций левее или правее него в алфавите. Меню «Файл» предполагает возможность извлечения текста для шифрования/дешифрования из файла, сохранение результатов шифрования/дешифрования в файл. Меню «Опции» предполагает изменение цвета фона окна приложения и стиля кнопок. Ограничения на алгоритм шифрования: шифровать только буквы латинского алфавита (остальные символы, например знаки препинания или кириллицу, не изменять).

2. Написать программу-блокнот, позволяющую открывать, редактировать и сохранять текстовые документы. Меню «Файл» предполагает возможность извлечения текста для редактирования из файла, сохранения результатов в файл, выход из программы. Меню «Помощь» содержит подменю «О приложении», при нажатии на которое в отдельном окне отображается информация о разработчике приложения. Строка состояния содержит две панели: правая панель – состояние документа «Сохранен/Изменён» (текст на панели =«Сохранён», если документ только что открыли или сохранили, либо «Изменён», если содержимое документа изменено), левая панель – состояние блокнота (если пользователь сохраняет документ, текст панели = «Сохраняю...», открывает = «Открываю...», программа находится в режиме ожидания = «Готов»).

3. Написать программу, предоставляющую возможность составить заказ на приобретение некоторого товара. В главном окне приложения слева отображается список всех товаров и информация о текущем товаре. Справа выводится состояние заказа: список выбранных товаров и их общая стоимость. Для управления работой приложения используются три кнопки: «Добавить товар в заказ», «Новый заказ» и «Сохранить заказ». В комбинированном списке «Товар:» содержатся наименования товаров, доступных для покупки. При выборе товара текст метки «Стоимость:…», текст метки «Изготовитель:…» и текст поля ввода «Описание:» меняют свои значения на значения, соответствующие выбранному товару. Комбинированный список «Товар:» заполняется при запуске программы. Значения для списка хранятся в исходном файле «PriceList.txt» в произвольном формате. В этом же файле хранятся стоимость, изготовитель и описание товара.

4. Написать программу «Учет пользователей». Программа хранит информацию о пользователях: имя, фамилию и адрес электронной почты. Изменять информацию пользователи могут только о себе. Вход в программу осуществляется по логину и паролю. Информация об учетных записях хранится в файле «Accounts.txt» в произвольном формате.Интерфейс программы должен состоять из трёх окон: окно авторизации, где вводятся логин и пароль, окно создания новой учетной записи, где вводится новый логин и пароль, ввод пароля дублируется; главное окно программы, в котором отображаются

имя, фамилия, адрес электронной почты пользователя, а также присутствуют кнопки для редактирования данных. При запуске программы появляется окно авторизации. После проверки введённых данных если пользователя нет в списке авторизированных пользователей, то при нажатии на кнопку «OK» появляется сообщение об ошибке. При нажатии на кнопку «Create User» появляется окно создания новой учетной записи.При создании новой учетной записи происходит проверка на уникальность логина и сложность пароля. Алгоритм определения сложности пароля предлагается выбрать самостоятельно. После ввода данных пользователь нажимает кнопку «Create». Если вновь введённый логин уже существует или пароль недостаточно сложен, появляется соответствующее сообщение и предлагается повторить ввод или отказаться от работы. Если проверка пройдена успешно, появляется главное окно программы. Если же пользователь есть в списке авторизированных пользователей, при нажатии на кнопку «OK» в окне авторизации появляется главное окно программы. Поля ввода в окне доступны только для чтения. Чтобы изменить значение в поле ввода, введены три кнопки «<- Edit». Эти кнопки изменяют состояния полей ввода с «Только для чтения» на «Чтение / запись» и обратно. Главное окно программы содержит также кнопки для выхода из приложения (Exit) и смены пользователя (Logout).

5.Написать программу, моделирующую работу регистратуры поликлиники.Главное окно приложения содержит расписание работы врачей с информацией о фамилии, имени, отчестве и специальности врача, дне недели, времени приёма (с 8 до 13 часов или с 14 до 19), номере кабинета. Главное меню приложения должно содержать пункты для создания и изменения данных: чтение данных из файла, сохранение данных в файл, редактирование информации о враче, добавление информации о враче, поиск врача с заданной специальностью (результат поиска выводится в отдельном окне), а также вывод информации о самой программе (пункт меню «О программе»). Все данные при добавлении и изменении должны проверяться на корректность: один и тот же врач может работать только пять дней в неделю и только в один из возможных интервалов времени, два врача не могут одновременно работать в одном и том же кабинете. При попытке ввода некорректных данных должно выдаваться соответствующее сообщение.

6. Написать программу, работающую с информацией о служащих учреждения. Данные об отдельном работнике состоят из имени, фамилии, отчества, даты рождения, образования, должности и названия отдела. Главное окно приложения должно содержать список людей, отвечающий условиям, которые определяются в меню. Главное меню должно содержать пункты для создания, изменения и отображения данных: чтение данных из файла, сохранение данных в файл, редактирование информации о служащем, добавление информации о служащем, вывод списка работников с заданным образованием, вывод списка работников заданного возраста (от a до b лет), вывод списка работников заданного отдела, а также вывод информации о самой программе (пункт меню «О программе»).

6. Написать программу, позволяющую компьютеру и человеку играть в слова. Предварительно программа объясняет правила игры и позволяет уточнить их в любой момент. Тематикой игры могут быть по выбору города, животные, растения и т. д.(не менее 5 тем). Тему из предложенных компьютером вариантов выбирает человек. Для игры компьютер использует собственную базу данных (для каждой тематики свою), хранящуюся в виде текстового файла. Если названное человеком слово отсутствует в базе, уточняется, правильно ли оно названо, и в случае правильности заносится в базу. Правила игры: первый игрок называет слово, затем второй должен предложить другое, начинающееся с той буквы, на которую оканчивается слово, названное первым. Повторять слова в течение одной игры нельзя. Компьютер должен реализовывать стратегию, отличающуюся от случайного выбора подходящего слова, например выбирать слово, заканчивающееся на редко встречающуюся букву.

7. Написать программу для следующей игры: имеется кучка из случайного количества камней (от 10 до 100). Каждый игрок по очереди берёт 1, 2, 3 или 4 камня. Выигрывает тот, кто а) забирает последний камень;б) оставляет противнику последний камень. Вариант выигрыша определяется в настройках игры. В настройках также определяется кто играет: игрок с компьютером; или два игрока. Программа должна реализовывать следующие функции: рисовать в главном окне приложения кучку камней в текущем состоянии; с помощью «мыши» или клавиш выбирать камни, которые следует взять при очередном ходе; реализовать выигрышную стратегию для компьютера в случае игры человека с компьютером; сохранять текущее состояние игры и восстанавливать его при желании пользователя; выводить сообщение о выигрыше или проигрыше.

8. Написать программу, позволяющую играть на бесконечном поле в «крестики-нолики»: игроку с компьютером; или двум игрокам. Если в качестве игрока выступает компьютер, программа делает первый ход. Делая очередной ход, программа анализирует ситуацию, рассчитывая возможные ходы противника вперед на 1—2 хода, и в результате проведенного анализа поступает оптимальным образом.

9. Написать программу, позволяющую играть в «Быки и коровы»: игроку с компьютером; или двум игрокам. Каждый из противников задумывает четырехзначное число, все цифры которого различны (первая цифра числа отлична от нуля). Необходимо разгадать задуманное число. Выигрывает тот, кто отгадает первый. Противники по очереди называют друг другу числа и сообщают о количестве «быков» и «коров» в названном числе («бык» — цифра есть в записи задуманного числа и стоит в той же позиции, что и в задуманном числе; «корова» — цифра есть в записи задуманного числа, но не стоит в той же позиции, что и в задуманном числе). Например, если задумано число 3275 и названо число 1234, получаем в названном числе одного «быка» и одну «корову». Число отгадано в том случае, если получилось 4 «быка».

10. Написать программу для игры человека в головоломки со спичками. На игровом поле находятся несколько спичек, сложенных определенным образом в определенные фигуры. Задача пользователя — убрать определенное количество спичек так, чтобы решить некоторую задачу, например из шести сложенных квадратов сделать три, убрав три спички. В процессе решения головоломок сложность увеличивается. При разработке проекта следует подобрать несколько головоломок разной сложности (не менее 15). В главном окне приложения отображается фигура из спичек, пользователь с помощью «мыши» или клавиш со стрелками выбирает спичку, которую хочет убрать, при нажатии на клавишу «Enter» или при двойном щелчке мыши спичка убирается. На игровом поле должна быть кнопка, позволяющая вернуть спичку, которую убрали. Если заданное количество спичек удалено, программа сообщает о правильности решения. Переход к следующей головоломке осуществляется после правильного решения текущей задачи или по желанию пользователя.

11. Написать программу, обучающую учащихся арифметическим действиям с отрицательными числами, а также предлагающую серию заданий различной сложности для закрепления навыков действий над такими числами. Программа должна содержать справочный материал, который описывает правила выполнения действий и приводит к каждому правилу подходящий пример. В режиме закрепления навыков предлагается набор заданий по возрастанию сложности (не менее 25 примеров), где учащийся вводит ответ к очередному примеру и получает информацию о его правильности. Если ответ не верен, то предлагается решить аналогичный пример, но не более трёх раз для текущего уровня сложности. После завершения тренировки выдаётся информация о количестве и соотношении правильных и неправильных ответов, и анализ результатов (на какие правила допущено больше всего ошибок). Кроме основного режима работы, должна быть реализована возможность отработки навыков для заданного правила или арифметического действия.

12. Написать программу, автоматизирующую процесс построения фигур на плоскости с помощью циркуля и линейки. Программа должна уметь выполнять следующие команды: отметить произвольную точку и обозначить ее; отметить произвольную точку и обозначить ее; построить произвольную прямую; \item построить окружность с заданным центром данного радиуса; построить и обозначить точку пересечения двух линий. Программа должна содержать 10—15 стандартных задач на построение школьного курса геометрии, предлагать их для решения и контролировать процесс построения и полученный результат.

13. Написать программу, моделирующую экологическую модель. Остров размером 20x20 заселен дикими кроликами, волками и волчицами. Имеется по нескольку представителей каждого вида. Кролики довольно глупы: в каждый момент времени они с одинаковой вероятностью 1/9 передвигаются в один из восьми соседних квадратов (за исключением участков, ограниченных береговой линией) или просто сидят неподвижно. Каждый кролик с вероятностью 0,2 превращается в двух кроликов, новый кролик занимает случайно выбранную соседнюю клетку. Волки и волчицы передвигаются случайным образом, пока в одном из соседних восьми квадратов не окажется кролик, за которым они охотятся. Если волк и кролик оказываются в одном квадрате, волк съедает кролика и получает одно очко. В противном случае он теряет 0,1 очка. Волки и волчицы с нулевым количеством очков умирают. В начальный момент времени все волки и волчицы имеют 1 очко. Если волк и волчица окажутся в соседних квадратах и рядом нет кроликов, которых можно съесть, они производят потомство случайного пола. Запрограммировать описанную модель и понаблюдать за изменением популяции в течение заданного периода времени.

14. Написать программу для игры с головоломкой Пятнашки. Игровое поле представляет собой набор одинаковых квадратных фишек с числами в квадратной коробке. Длина стороны коробки в четыре раза больше длины стороны костяшек. В коробке находится набор из 15 пронумерованных от 1 до 15 костяшек, соответственно остаётся незаполненным одна квадратная ячейка. Цель игры — перемещая костяшки по коробке, добиться упорядочивания их по номерам, сделав как можно меньше шагов. Программа должна реализовывать следующие функции: \begin{itemize} \item в начале новой игры создавать игровое поле генератором случайных чисел; \item рисовать в главном окне приложения игровое поле в текущем состоянии; \item с помощью «мыши» или клавиш со стрелками выбирать костяшку для перемещения; \item при нажатии на клавишу «Enter» или при двойном щелчке «мыши» перемещать выбранную костяшку в свободную ячейку, если это возможно; \item сохранять текущее состояние игрового поля и восстанавливать его при желании пользователя; \item если костяшки упорядочены, то выводить сообщение о выигрыше и количестве сделанных перемещений. \end{itemize}

15. Написать программу, моделирующую аквариум. Аквариум содержит камни и рыб. Он представляет собой область главного окна приложения, наполненную водой. Рыбы живут в аквариуме. Рыбка имеет координаты, скорость, размер, цвет, направление движения, поле зрения (небольшой отрезок по направлению движения). Нарисовать её можно в виде стрелки, направленной острием по ходу движения. Рыбка перемещается в текущем направлении на расстояние, зависящее от скорости, иногда случайным образом меняет направление движения. Если рыба видит препятствие, направление движения меняется, пока препятствие не исчезнет из поля зрения. Приложение содержит три кнопки: Init — включает графический режим, заполняет аквариум водой, камнями и рыбами; Run — организует бесконечный цикл, в котором движутся все обитатели аквариума;Done — выключает графический режим.

16. Написать программу, моделирующую Солнечную систему. Необходимо изобразить на экране компьютера Солнце и восемь планет Солнечной системы (от Меркурия до Нептуна), а также основные их спутники (например, для Земли – Луну, для Марса – Фобос и Деймос), в их движении по орбитам. Можно считать, что вращение

планет вокруг Солнца происходит в одной плоскости (поскольку плоскости орбит планет близки к плоскости земной орбиты), к этой плоскости можно отнести и орбиты спутников планет. Вращение планет вокруг своей оси можно не учитывать. При визуализации Солнечной системы на экране компьютера должно быть соблюдено правильное соотношение размеров орбит планет и скоростей их движения (то есть они должны быть пропорциональны их реальным значениям). Соотношение размеров изображений самих планет также должно соответствовать действительности, но при этом для наглядности масштаб их изображения должен быть больше масштаба показа их орбит, иначе некоторые планеты отображаются на экране точками. Кроме того, поскольку при показе на экране сразу всех восьми планет Солнечной системы изображения ближайших к Солнцу планет (и их спутников) получаются слишком мелкими и сливаются друг с другом, следует либо использовать крупный масштаб и предоставить скроллинг изображения, либо предусмотреть визуализацию Солнечной системы в двух масштабах (для всех планет и для ближайших к Солнцу). Пользователь должен иметь возможность включать и отключать показ названий планет и спутников, их орбит и траекторий движения других тел, а также ускорять или замедлять движение тел в Солнечной системе.

17. Написать программу, позволяющую конструировать электрические схемы с помощью графических инструментов, представляющих отдельные элементы электрической цепи. Пользователь системы должен иметь возможность в рабочем окне: \ begin{itemize} \item задавать контур электрической цепи, выбирая его элементы из нескольких возможных; \item располагать в нужных точках заданного контура необходимые элементы; \item изменять расположение отдельных элементов заданной цепи; \item замыкать и размыкать входящие в цепь выключатели (замыкающие ключи); \ item запоминать построенную схему в файле и считывать ее из файла в рабочее окно. \ end{itemize} Необходимо, чтобы указанные действия пользователь мог производить в произвольном, удобном для него порядке. Визуализация электрической схемы должна включать изображение цепи и всех входящих в нее элементов, а также показ текущего состояния выключателей (замыкающих ключей) и горящих лампочек, сигнализирующих о наличии тока на соответствующем участке цепи. Дополнительно можно задавать внутренние характеристики отдельных элементов электрической цепи, величину тока, напряжения и сопротивления в определенных точках/участках построенной схемы, осуществлять расчет для заданной электрической схемы основных ее характеристик, то есть величины тока в каждой ее точке и величины напряжения между двумя произвольными точками.

Требования к выполнению задания

Общие требования

1. Сдаваемая программа должна быть концептуально и текстуально понятной, функционировать правильно, обладать хорошо документированным исходным текстом.
2. Система классов программы должна быть осмысленной и отвечать парадигме объектно-ориентированного программирования.
3. Программа должна быть спроектирована с соответствии архитектурой «модель-вид-контроллер» или одной из её вариаций с обязательным отделением бизнес-логики приложения (уровень модели) от логики представления (пользовательского интерфейса).
4. Настоятельно рекомендуется разрабатывать модульные тесты для всех классов приложения, для которых это представляется рациональным (как минимум, для классов уровня бизнес-логики).
5. Главный класс приложения должен инкапсулировать лишь наиболее общую логику программы. В качестве главного класса графического приложения может выступать класс главной формы, если он осуществляет лишь инициализацию и

отображение этой формы. Рекомендуется использовать шаблоны проектирования во всех случаях, когда это представляется уместным.

6. Не допускается выполнять такие оптимизации программы, которые ухудшают качество её исходного текста (затрудняют восприятие кода человеком, усложняют модификацию, рефакторинг, отладку и т. д.).
7. Программа должна корректно обрабатывать все ошибочные ситуации. Сообщения об ошибках должны быть максимально информативными.

Требования к отдельным деталям реализации программы

1. Обязательно выполнение следующих соглашений по именам:

Имена должны быть осмысленными словами (или словосочетаниями в значении соответствующей части речи) английского или (категорически не рекомендуется!) русского языка (транслитом). Для локальных переменных допускается использование сокращённых имён.

Имена классов – существительные или словосочетания в значении существительных: в нижнем регистре, первые буквы слов – в верхнем регистре, разделители слов не используются.

Имена полей и локальных переменных – существительные в нижнем регистре, первые буквы слов начиная со второго – в верхнем регистре, разделители слов не используются.

Имена методов – глаголы в нижнем регистре (либо словосочетания, отражающие действия), первые буквы слов начиная со второго – в верхнем регистре, разделители слов не используются. Для имён методов, возвращающих значения величин допускается использование существительных, а не глаголов, если только это не приведёт к неоднозначности.

Имена методов, возвращающих результат типа boolean, начинаются на is (isOk); выполняющих чтение/изменение значений полей класса – на get и set соответственно (getFileAttr, setFileAtt}); выполняющих преобразование к другому типу данных – на to (toString);

Имена констант – существительные или словосочетания в верхнем регистре, слова разделены подчёркиваниями.

2. Все элементы программы должны иметь минимально возможную область видимости.

3. Запрещается использование статических полей и методов при наличии возможности достичь результата другими средствами.

4. Запрещается использование внутренних анонимных классов, за исключением классов-обработчиков событий. Последние должны выполнять лишь переброс вызова другому методу класса и не должны содержать других операторов.

5. Запрещается использование методов, выполняющих две или более самостоятельные операции. Каждый такой метод подлежит разбиению на более мелкие.

6. Не рекомендуется использование методов, занимающих более 15 строк. Использование методов, занимающих более 50 строк, запрещается, кроме исключительных случаев (исключительность должна быть обоснована!).

7. Запрещается собственная реализация средств, имеющих аналоги в стандартной библиотеке Qt. Во всех случаях, когда возможно использование библиотечных классов, они должны быть использованы.

8. Запрещается посимвольная обработка строковых данных в стиле языка C. Вместо этого необходимо использовать средства стандартной библиотеки.

9. Метод \main(), являющийся точкой входа в программу, не должен выбрасывать исключений.

Требования по оформлению исходного текста программы

- Исходный текст каждого класса программы должен быть размещён в отдельном файле (кроме вложенных классов).
- Программы должны быть выровнены в соответствии с одним из двух допустимых стилей: стиль Кёрнигана и Ричи или стиль Олмана. Недопустимо смешение различных стилей выравнивания в рамках одного проекта.
- Величина отступа всюду должна быть одинаковой (рекомендуется 4 символа).
- Все операторы линейной части программы должны иметь один и тот же отступ. Отступ увеличивается для объявлений вложенных классов, полей и методов классов, тел методов, субоператоров (в т. ч. для всех операторов блока, образующего субоператор).
- При использовании множественного ветвления допускается и рекомендуется размещать конструкции else if строго друг под другом без дополнительных отступов.
- Фигурная скобка, открывающая класс, метод или блок размещается либо на той же строке, что и описание конструкции, которую она открывают (стиль Кёрнигана и Ричи), либо на отдельной строке без дополнительного отступа (стиль Олмана).
- Закрывающая фигурная скобка размещается на отдельной строке с отступом влево относительно той конструкции которую она закрывает.
- Точка с запятой, завершающая пустой оператор, должна размещаться на отдельной строке.
- Не допускается использование строк, выходящих за пределы экрана. Не вмещающиеся части строки переносятся на следующие строки с отступом относительно предыдущей строки. Рекомендуется использовать отступ вдвое больше определённого для программы. Перенос строк, содержащих объявление метода, допускается осуществлять без отступа.
- Не рекомендуется размещение нескольких операторов в одной строке, кроме случаев, когда это не ухудшает удобочитаемости программы.
- Рекомендуется использование пробелов для отбивок отдельных лексем в программе.Способ расстановки пробелов в этом случае должен соответствовать полиграфическим правилам.
- Многострочный документационный комментарий выравнивается следующим образом: первая и последняя строки содержать только символы /** и */ (или **/) соответственно. Все промежуточные строки начинаются со звёздочки, за которой следует текст комментария.

Требования по документированию программы

- Для сдаваемой программы создаётся описание в формате HTML всех пакетов, классов, полей и методов.
- Обязательными являются комментарии для всех классов, полей и методов.
- Комментарий в начале каждого файла должен содержать: имя проекта и его описание (краткое или полное), полное имя класса, содержащегося в файле, описание этого класса, Ф. И. О. автора, название учебной группы.
- Если смысл хотя бы одного из принимаемых или возвращаемых методом значений или

выбрасываемых исключений не является интуитивно понятным, то соответствующий комментарий должен содержать описание всех указанных элементов.

- В случае использования меток в операторах break и continue обязательно наличие комментария в строке, содержащей один из указанных операторов.
- Комментарии не должны содержать орфографических и пунктуационных ошибок.

Критерии оценивания индивидуальных заданий

Оценка	Критерии
Отлично Уровень формирования компетенций: высокий	ПК-5: Выполнены все описанные выше требования к заданию. Знает принципы построения приложений под мобильные платформы. Выбирает оптимальные средства и инструменты разработки. Умеет работать с документацией. Анализирует поставленную задачу, использует при этом несколько источников информации: учебники, статьи, лекционные материалы.
Хорошо Уровень формирования компетенций: продвинутый	ПК-5: Выполнены почти все описанные выше требования к заданию, нарушено не более одного – трех пунктов из каждого набора требований. Знает принципы построения приложений под мобильные платформы. Выбирает оптимальные средства и инструменты разработки для большинства используемых алгоритмов. Умеет работать с документацией. Использует при решении задачи в основном лекционный материал, но может пользоваться другими источниками информации, возможно не полностью понимая качество получаемого решения.
Удовлетворительно Уровень формирования компетенций: пороговый	ПК-5: Выполнены описанные выше требования к заданию, нарушено не более половины пунктов из каждого набора требований. Знает принципы построения приложений под мобильные платформы. Выбирает подходящие средства и инструменты разработки, возможно не достаточно эффективные. Работает с документацией с затруднениями. Использует при решении задачи в только лекционный материал. Другие источники информации воспринимает с трудом или не пытается анализировать.
Неудовлетворительно	ПК-5: Описанные выше требования к заданию не выполнены по большинству пунктов из каждого набора требований. Не знает принципы построения приложений под мобильные платформы. Не может выбрать средства и инструменты разработки. Не умеет работать с документацией. Не умеет использовать лекционный материал, а также другие источники информации для решения поставленной задачи.

Список вопросов к экзамену

1. Отличия мобильных устройств от настольных и переносных персональных компьютеров.
2. Особенности мобильного программирования: архитектурные отличия используемых платформ (Intel/ARM); управление питанием и требования к энергозатратам системы и приложений, основные способы увеличения срока автономной работы; разнообразие размеров экранов и разрешений.

3. Требования к организации пользовательского интерфейса, типовые элементы управления мобильных интерфейсов; сенсорные интерфейсы, управление жестами и экранные клавиатуры; система встроенных сенсоров и реализация обратной связи; вопросы безопасности.
4. Типовой цикл подготовки приложения в интегрированной среде к запуску в эмулируемой среде или на целевом устройстве на примере простого шаблонного приложения.
5. Интегрированная среда разработки QtCreator, режимы и панели, их назначение;
6. Средства кросс-компиляции Mer build engine, эмулятор устройства ОС Аврора Emulator для системы виртуализации VirtualBox, ограничения эмулятора по сравнению с реальным устройством.
7. Процесс сборки приложения, ключевые понятия (tool, toolchain, cross-compilation, compilation target).
8. Архитектура мобильной платформы, доступные программные интерфейсы, место и роль фреймворка Qt и технологии QtQuick в разработке приложений.
9. Шаблон простейшего приложения на языке QML: визуальные невизуальные элементы и их свойства; доступные типы свойств; присваивание и привязка свойств;
10. Древовидная структура описания графического интерфейса; иерархия элементов и наследование свойств;
11. События и их обработка, императивный код на языке JavaScript в обработчиках событий, обзор доступных модулей элементов, в т.ч., Silica.
12. Базовый элемент Item и основные визуальные элементы (прямоугольник, изображение, текст), их основные свойства.
13. Геометрия элементов. Системы координат и абсолютное позиционирование. Относительное позиционирование, контейнеры элементов. Порядок отрисовки и z-координата. Масштабирование и поворот элементов.
14. Анимация свойств элементов.
15. Организация диалога с пользователем, обработка событий ввода, в том числе жестов.
16. Типовые элементы управления и их мобильные аналоги на примере элементов Qt
17. Quick Controls и Silica: надписи, кнопки, переключатели, текстовые поля, инструменты выбора, индикаторы прогресса и занятости и т. п.
18. Средства организации мобильных приложений на примере элементов Qt Quick Controls и Silica: окна, системы меню, диалоги, средства организации многостраничных приложений и навигации по страницам.
19. Создание обложки приложения для отображения на рабочем столе.
20. Объединение и группировка элементов; шаблон проектирования MVC, создание моделей данных и визуальных элементов, отображений с помощью представлений.
21. Поддержка форматов JSON и XML. Получение данных от сервисов сети интернет на примере XMLListModel.
22. Построение моделей данных с использованием источников с SQL-доступом, в частности, хранение данных в локальной базе SQLite. Использование Loca Storage API и элементов для хранения настроек приложения.
23. Средства описания состояния в виде набора значений свойств составляющих его элементов. Именование состояний и декларативное определение логики переходов. Определение переходов (например, анимаций).

24. Создание собственных элементов на языке QML, реализация свойств, методов (в виде функций Javascript) и сигналов событий.
25. Основы использования фреймворка Qt и Qt C++. Метаобъектная система, свойства и интроспекция, сигналы и слоты.
26. Контейнерные классы. Создание доступных в QMLсцене методов, объектов и классов Qt C++, создание подключаемых элементов и модулей на Qt C++.
27. Сетевые возможности мобильных устройств и проблемы, связанные с мобильностью пользователя. Управление подключением к сети.
28. Технологии ОС Linux, обеспечивающие управление соединениями и доступ к средствам обмена данными, в том числе сетей сотовой связи, Bluetooth и NFC. Элементы QML для обеспечения обмена данными.
29. Управление контактами, обменом сообщениями, событиями календаря, встроенным веб-браузером. Домашний экран. Встраивание компонента браузера в приложения.
30. Гибридные приложения на основе Qt WebChannel.
31. Графический холст Canvas. Элементы Sprite и SpriteSequence. Системы частиц Particles.
32. Элементы для встраивания мультимедийного содержимого (проигрывания аудио и видео.
33. Типовые встроенные сенсоры мобильных устройств и примеры доступных посредством Bluetooth носимых датчиков.
34. Основы функционирования средств геопозиционирования GPS/ГЛОНАСС: спутниковая группировка, способ определения местоположения на основе данных спутников, реализация обработки данных от модуля позиционирования (координат, высоты над уровнем моря, даты и времени, скорости, отклонения от северного направления) на примере Qt Positioning API и соответствующих классов QML.
35. Использование камеры на примере Qt Multimedia и элемента Camera языка QML.
36. Соглашения о стиле кодирования для Qt и QML, дополнительные рекомендации Аврора. Вопросы потребления памяти и производительности: выделение памяти в QML-приложении и рекомендации по программированию, способствующие эффективному управлению памятью.
37. Применение средств статического анализа кода: Clang Static Analyzer. Модульное тестирование приложений QML.

Методические указания по выставлению оценки за экзамен

Оценка за экзамен выставляется по результатам выполнения индивидуального задания в соответствии с описанными выше критериями. Индивидуальное задание может быть сдано в течение семестра последовательно в процессе освоения материала или на экзамене. В случае необходимости преподаватель в ходе сдачи задания может провести беседу по вопросам к экзамену, связанным в первую очередь с тематикой индивидуального задания.

2. Перечень компетенций, этапы их формирования, описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкалы оценивания

2.1. Шкала оценивания сформированности компетенций и ее описание

Оценивание уровня сформированности компетенций в процессе освоения дисциплины осуществляется по следующей трехуровневой шкале:

Пороговый уровень - предполагает отражение тех ожидаемых результатов, которые определяют минимальный набор знаний и (или) умений и (или) навыков, полученных студентом в результате освоения дисциплины. Пороговый уровень является обязательным уровнем для студента к моменту завершения им освоения данной дисциплины.

Продвинутый уровень - предполагает способность студента использовать знания, умения, навыки и (или) опыт деятельности, полученные при освоении дисциплины, для решения профессиональных задач. Продвинутый уровень превосходит пороговый уровень по нескольким существенным признакам.

Высокий уровень - предполагает способность студента использовать потенциал интегрированных знаний, умений, навыков и (или) опыта деятельности, полученных при освоении дисциплины, для творческого решения профессиональных задач и самостоятельного поиска новых подходов в их решении путем комбинирования и использования известных способов решения применительно к конкретным условиям. Высокий уровень превосходит пороговый уровень по всем существенным признакам.

3. Методические рекомендации преподавателю по процедуре оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Целью процедуры оценивания является определение степени овладения студентом ожидаемыми результатами обучения (знаниями, умениями, навыками и (или) опытом деятельности).

Процедура оценивания степени овладения студентом ожидаемыми результатами обучения осуществляется с помощью методических материалов, представленных в разделе «Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций»

3.1 Критерии оценивания степени овладения знаниями, умениями, навыками и (или) опытом деятельности, определяющие уровни сформированности компетенций

Пороговый уровень (общие характеристики):

- владение основным объемом знаний по программе дисциплины;
- знание основной терминологии данной области знаний, стилистически грамотное, логически правильное изложение ответа на вопросы без существенных ошибок;
- владение инструментарием дисциплины, умение его использовать в решении стандартных (типовых) задач;
- способность самостоятельно применять типовые решения в рамках рабочей программы дисциплины;
- усвоение основной литературы, рекомендованной рабочей программой дисциплины;
- знание базовых теорий, концепций и направлений по изучаемой дисциплине;
- самостоятельная работа на практических и лабораторных занятиях, периодическое участие в групповых обсуждениях, достаточный уровень культуры исполнения заданий.

Продвинутый уровень (общие характеристики):

- достаточно полные и систематизированные знания в объёме программы дисциплины;
- использование основной терминологии данной области знаний, стилистически грамотное, логически правильное изложение ответа на вопросы, умение делать выводы;
- владение инструментарием дисциплины, умение его использовать в решении учебных и профессиональных задач;
- способность самостоятельно решать сложные задачи (проблемы) в рамках рабочей программы дисциплины;
- усвоение основной и дополнительной литературы, рекомендованной рабочей программой дисциплины;
- умение ориентироваться в базовых теориях, концепциях и направлениях по изучаемой дисциплине и давать им сравнительную оценку;
- самостоятельная работа на практических и лабораторных занятиях, участие в групповых обсуждениях, высокий уровень культуры исполнения заданий.

Высокий уровень (общие характеристики):

- систематизированные, глубокие и полные знания по всем разделам дисциплины;
- точное использование терминологии данной области знаний, стилистически грамотное, логически правильное изложение ответа на вопросы, умение делать обоснованные выводы;

- безупречное владение инструментарием дисциплины, умение его использовать в постановке и решении научных и профессиональных задач;
- способность самостоятельно и творчески решать сложные задачи (проблемы) в рамках рабочей программы дисциплины;
- полное и глубокое усвоение основной и дополнительной литературы, рекомендованной рабочей программой дисциплины;
- умение ориентироваться в основных теориях, концепциях и направлениях по изучаемой дисциплине и давать им критическую оценку;
- активная самостоятельная работа на практических и лабораторных занятиях, творческое участие в групповых обсуждениях, высокий уровень культуры исполнения заданий.

3.2 Описание процедуры выставления оценки

В зависимости от уровня сформированности каждой компетенции по окончании освоения дисциплины студенту выставляется оценка «отлично», «хорошо», «удовлетворительно», «неудовлетворительно».

Показатели и критерии, используемые при выставлении оценки подробно описаны в разделе «Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций».

Высокий уровень формирования компетенций соответствует оценке «отлично» за индивидуальное задание.

Продвинутый уровень формирования компетенций соответствует оценке «хорошо» за индивидуальное задание.

Пороговый уровень формирования компетенций соответствует оценке «удовлетворительно» за индивидуальное задание.

Оценка «отлично» выставляется студенту, у которого каждая компетенция (полностью или частично формируемая данной дисциплиной) сформирована на высоком уровне.

Оценка «хорошо» выставляется студенту, у которого каждая компетенция (полностью или частично формируемая данной дисциплиной) сформирована не ниже, чем на продвинутом уровне.

Оценка «удовлетворительно» выставляется студенту, у которого каждая компетенция (полностью или частично формируемая данной дисциплиной) сформирована не ниже, чем на пороговом уровне.

Оценка «неудовлетворительно» выставляется студенту, у которого хотя бы одна компетенция (полностью или частично формируемая данной дисциплиной) сформирована ниже, чем на пороговом уровне.

Оценка «зачет» выставляется студенту, у которого каждая компетенция (полностью или частично формируемая данной дисциплиной) сформирована не ниже, чем на пороговом уровне.

Оценка «незачтено» выставляется студенту, у которого хотя бы одна компетенция (полностью или частично формируемая данной дисциплиной) сформирована ниже, чем на пороговом уровне.

Приложение №2 к рабочей программе дисциплины
«Разработка программных приложений для ОС Аврора»

Методические указания для студентов по освоению дисциплины

Занятия проводятся в интерактивной форме с использованием мультимедиа-технологий. Занятия предполагают наличие дискуссий по поводу тех или иных вопросов разработки программных приложений осуществляемых в результате соответствующего предложения преподавателя.

Практическое применение полученных знаний отрабатывается и во время практических занятий, ориентированных помимо закрепления лекционного материала на разбор различных модельных ситуаций, характерных для разработки современных мобильных сервисов. Разработка программного приложения по индивидуальному заданию является одним из основным результатов работы студента над материалом курса. Изложение материала предполагает постепенное усложнение разрабатываемого приложения по мере изучения технологий. Тема задания выбирается студентом самостоятельно в начале обучения и обсуждается с преподавателем. Результат работы студент представляет по окончании курса на зачете в виде готовой программы.

Текущий контроль успеваемость осуществляется в форме опросов по основным понятиям и концепциям курса, осуществляемый в ходе практических занятий. Для самостоятельной работы студентам предлагается разрабатывать собственные идеи мобильных сервисов и систем с применением разобранных во время лекций и лабораторных занятий подходов и методик. Окончательная аттестация осуществляется в форме зачета, основную часть которого составляет представление результатов индивидуальной работы, а также собеседование по тематике курса.

Учебно-методическое обеспечение самостоятельной работы студентов по дисциплине

Для самостоятельной работы особенно рекомендуется использовать учебную литературу, указанную в разделе № 7 данной рабочей программы.

Также для подбора учебной литературы рекомендуется использовать широкий спектр интернет-ресурсов:

1. Электронно-библиотечная система «Университетская библиотека online» (www.biblioclub.ru) - электронная библиотека, обеспечивающая доступ к наиболее востребованным материалам-первоисточникам, учебной, научной и художественной литературе ведущих издательств (*регистрация в электронной библиотеке – только в сети университета. После регистрации работа с системой возможна с любой точки доступа в Internet.).

2. Для самостоятельного подбора литературы в библиотеке ЯрГУ рекомендуется использовать:

1. Личный кабинет (http://lib.uniyar.ac.ru/opac/bk_login.php) дает возможность получения on-line доступа к списку выданной в автоматизированном режиме литературы, просмотра и копирования электронных версий изданий сотрудников университета (учеб. и метод. пособия, тексты лекций и т.д.) Для работы в «Личном кабинете» необходимо зайти на сайт Научной библиотеки ЯрГУ с любой точки, имеющей доступ в Internet, в пункт меню «Электронный каталог»; пройти процедуру авторизации, выбрав вкладку «Авторизация», и заполнить представленные поля информации.

2. Электронная библиотека учебных материалов ЯрГУ (http://www.lib.uniyar.ac.ru/opac/bk_cat_find.php) содержит более 2500 полных текстов учебных и учебно-методических материалов по основным изучаемым дисциплинам, изданных в университете. Доступ в сети университета, либо по логину/паролю.

3. Электронная картотека «Книгообеспеченность»

(http://www.lib.uniyar.ac.ru/opac/bk_bookreq_find.php) раскрывает учебный фонд научной библиотеки ЯрГУ, предоставляет оперативную информацию о состоянии книгообеспеченности дисциплин основной и дополнительной литературой, а также цикла дисциплин и специальностей. Электронная картотека «Книгообеспеченность» доступна в сети университета и через Личный кабинет.