РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Б1.В.10 РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Направление подготовки
09.03.03 Прикладная информатика

Профиль подготовки
Прикладная информатика цифровой экономики

Квалификация (степень) выпускника **бакалавр**

Уфа 2023

Составитель: канд. физ.-мат. наук, доцент
Шамсутдинова Т.М.

Программа составлена в соответствии с Федеральным государственным образовательным стандартом высшего образования по направлению подготовки 09.03.03 Прикладная информатика, утвержденным приказом Министерства образования и науки Российской Федерации
19.09.2017 г. № 922, зарегистрирован в Министерстве юстиции Российской Федерации 12.10.2017 г. № 48531. Редакция с изменениями № 1456 от 26.11.2020; С изменениями и дополнениями от 26 ноября 2020 г., 8 февраля 2021 г.

Рабочая программа обсуждена и одобрена на заседании кафедры информатики и ИТ «23» марта 2023 г. (протокол № 9)

Зав. кафедрой информатики и ИТ д-р техн. наук, доцент
Беляева А.С.

Рассмотрена и одобрена на заседании методической комиссии экономического факультета «23» марта 2023 г. (протокол № 7).

Председатель методической комиссии экономического факультета, канд. экон. наук, доцент
_____ Фролова О.Н.

Согласовано:
Руководитель ОПОП ВО динова Т.М.

1 Перечень планируемых результатов обучения по дисциплине Б1.В.11 Разработка мобильных приложений, соотнесенных с планируемыми результатами освоения образовательной программы 38.03.05 Бизнес-информатика

В результате освоения образовательной программы бакалавриата обучающийся должен овладеть следующими результатами обучения по дисциплине:

Код и наименование компетенции	Код и наименование индикаторов достижения компетенции	Планируемые результаты обучения
ПК-2 Способность разрабатывать и адаптировать прикладное программное обеспечение	ПК-2.1 Кодирование на языках программирования	Знания: ПК-2.1/Зн1 основ современных систем управления базами данных, теории баз данных, основ программирования; современных объектноориентированных языков программирования, современных структурных языков программирования, языков современных бизнес-приложений; Умения: ПК-2.1/Ум1 кодировать на языках программирования; ПК-2.1/Ум2 проводить верификацию кода ИС и баз данных ИС; тестировать результаты кодирования; ПК-2.1/Ум3 применять web-технологии для разработки контента и ИТ-сервисов предприятия ПК-2.1/Ум4 применять методы, способы и средства получения, хранения, переработки информации с целью разработки интернет-ресурсов Навыки: ПК-2.1/Нв1 разработки кода ИС и баз данных ИС; ПК-2.1/Нв2 верификации кода ИС и баз данных ИС относительно дизайна ИС и структуры баз данных ИС; устранения обнаруженных несоответствий; ПК-2.1/Нв3 использования основ web-технологий для разработки контента и ИТ-сервисов предприятия ПК-2.1/Нв4 использования инструментальных средств разработки интернет-ресурсов

2 Место дисциплины в структуре образовательной программы

Дисциплина «Разработка мобильных приложений» относится к дисциплинам части, формируемой участниками образовательных отношений, блока 1 «Дисциплины/модули». Данная дисциплина взаимосвязана с такими дисциплинами ОПОП как «Разработка интерфейсов прикладных программ», «Программирование в корпоративных информационных системах», «Разработка интернет-ресурсов», «Web-технологии», «Основы программирования в 1С».

Для изучения дисциплины обучающиеся должны владеть языком программирования 1С, основными механизмами технологической платформы 1С:Предприятие 8.3, знать отличия в пользовательских интерфейсах «десктопного» и мобильного приложений

Дисциплина изучается обучающимися на 4 курсе в 8-ом семестре на очной форме обучения, и на 4 курсе в 7,8-ом семестрах на заочной форме обучения.

Последующие дисциплины (практики) по связям компетенций:

Б2.В.01(П) Преддипломная практика

Б3.О.01 Подготовка к сдаче и сдача государственного экзамена

Б3.О.02 Выполнение и защита выпускной квалификационной работы

3 Объем дисциплины и виды учебной работы

Общая трудоемкость (объем) дисциплины (модуля) составляет 5 зачетных единиц (ЗЕ)

3.1 Очное обучение (срок обучения: 4 года)

Вид учебной работы	Всего часов	Распределение по семестрам
		8 сем.
Контактная работа, всего в т.ч.: занятия лекционного типа (лекции) (Л) занятия семинарского типа (лабораторные работы) (ЛР), в т.ч. направленные на практическую подготовку (ПРП)	58 24 34 6	58 24 34 6
Самостоятельная работа обучающегося (СРО), всего в т.ч.: подготовка к лабораторным работам (ЛР) расчетно-графическая работа (РГР) самостоятельное изучение теоретического материала (СИТМ)	86 34 20 32	86 34 20 32
Вид промежуточной аттестации	экзамен (36 ч)	экзамен (36 ч)
Общая трудоемкость дисциплины часы	180	180
зачетные единицы	5	5

3.2 Заочное обучение (срок обучения: 4 года 6 месяцев)

Вид учебной работы	Всего часов	Распределение по семестрам	
		7	8
Контактная работа, всего в т.ч.: занятия лекционного типа (лекции) (Л) занятия семинарского типа (лабораторные работы) (ЛР), в т.ч. направленные на практическую подготовку (ПРП)	26 8 18 -	10 8 2 -	16 - 16 -
Самостоятельная работа обучающегося (СРО), всего в т.ч.: подготовка к лабораторным работам (ЛР) расчетно-графическая работа (РГР) самостоятельное изучение теоретического материала (СИТМ)	118 42 20 56	26 18 - 8	92 24 20 48
Вид промежуточной аттестации	экза-мен (36 ч)	-	экзамен (36 ч)

Общая трудоемкость дисциплины	часы	180	36	144
	зачетные единицы	5	1	4

4 Содержание дисциплины

4.1 Разделы дисциплины и виды занятий для очного и заочного обучения

№ п/п	Наименование раздела дисциплины	Очное обучение			Заочное обучение		
		Л	ЛР/ПРП	СРО	Л	ЛР/ПРП	СРО
Раздел 1. Основы разработки мобильных приложений							
1	**Раздел 1.** Основы разработки мобильных приложений	12	18/6	36	2	8	48
2	**Раздел 2.** Разработка универсальных прикладных решений	6	8	24	2	6	36
3	**Раздел 3.** Создание мобильных клиентов для облачных сервисов	6	8	26	4	4	34
Итого:		**24**	**34/6**	**86**	**8**	**18**	**118**

4.2 Содержание разделов дисциплины

№ п/п	Наименование **модулей** (раздела) дисциплины	Содержание раздела
1	Раздел 1. Основы разработки мобильных приложений	1.1 Технология разработки и сборки мобильного приложения. Создание тестовой площадки для разработки. Разработка и тестирование мобильного приложения. Особенности отладки мобильного приложения. Установка и настройка сборщика мобильных приложений. Сборка мобильного приложения. Создание многокомпонентных приложений. Особенности использования механизмов платформы. 1.2 Ограничения функциональности мобильной платформы. Работа с файлами. Отчеты. Работа с ролями и пользователями. Внешние компоненты. Работа с картинками. Особенности работы на различных ОС. Разделение данных в мобильном приложении. 1.3 Использование мобильного функционала. Работа с телефонией. Работа с сообщениями (SMS и MMS). Средства геопозиционирования. Работа с мультимедиа. Сканирование штрихкодов. Работа с электронной почтой. Работа с контактами. Работа с календарем. Работа с уведомлениями. Работа с рекламой. Работа со встроенными покупками

2	Раздел 2. Разработка универсальных прикладных решений	2.1 Особенности разработки интерфейса. Настройка интерфейса для мобильного устройства и персонального компьютера. Различия в функциональности начальных страниц. Различия функциональности форм списков. Различия функциональности форм объектов и записей. Проектирование и настройка командного интерфейса для мобильного устройства и персонального компьютера. 2.2 Программирование логики. Реализация клиентсерверной логики. Работа с учетными записями пользователей. Демонстрационные базы в мобильном приложении. Разработка отчетов и использование СКД. Обмен данными между мобильными клиентами
3	Раздел 3. Создание мобильных клиентов для облачных сервисов	3.1 Разработка мобильного клиента для сервиса. Реализация аутентификации в облаке на стороне мобильного клиента. Разделение функциональности мобильного клиента на компоненты. Логика обращения к облачному сервису для командного взаимодействия и обмена данными. 3.2 Разработка облачного сервиса. Создание веб-сервиса регистрации. Создание http-сервиса для обмена данными

5 Тематика контактной работы

5.1 Занятия лекционного типа (лекции)

№ п/п	№ раздела	Наименование лекционных занятий	Объем, часы	
			очное обучение	заочное обучение
Раздел 1				
1	1.1	Создание тестовой площадки для разработки • Разработка и тестирование мобильного приложения • Особенности отладки мобильного приложения	2	1
2	1.1	Установка и настройка сборщика мобильных приложений • Сборка мобильного приложения • Создание многокомпонентных приложений	2	1
3	1.2	Ограничения функциональности мобильной платформы Работа с файлами	2	-
4	1.2	• Отчеты • Работа с ролями и пользователями • Внешние компоненты • Работа с картинками • Особенности работы на различных ОС • Разделение данных в мобильном приложении		

5	1.3	Работа с телефонией • Работа с сообщениями (SMS и MMS) • Средства геопозиционирования • Работа с мультимедиа • Сканирование штрихкодов	3	-
6	1.3	Работа с электронной почтой • Работа с контактами Работа с календарем • Работа с уведомлениями • Работа с рекламой • Работа со встроенными покупками	3	
Раздел 2				
7	2.1	Особенности разработки интерфейса • Настройка интерфейса для мобильного устройства и персонального компьютера • Различия в функциональности начальных страниц • Различия функциональности форм списков Различия функциональности форм объектов и записей • Проектирование и настройка командного интерфейса для мобильного устройства и персонального компьютера	3	1
8	2.2	Программирование логики • Реализация клиент-серверной логики • Работа с учетными записями пользователей Демонстрационные базы в мобильном приложении • Разработка отчетов и использование СКД Обмен данными между мобильными клиентами	3	1
Раздел 3				
9	3.1	Разработка мобильного клиента для сервиса • Реализация аутентификации в облаке на стороне мобильного клиента Разделение функциональности мобильного клиента на компоненты • Логика обращения к облачному сервису для командного взаимодействия и обмена данными	4	2
10	3.2	Разработка облачного сервиса • Создание веб-сервиса регистрации • Создание http-сервиса для обмена данными	2	2
		Итого	**24**	**8**

5.2 Занятия семинарского типа (практические занятия) Не предусмотрены

5.3 Занятия семинарского типа (лабораторные работы)

№ п/п	№ раздела	Наименование лабораторных работ	Объем, часы	
			очное обучение	заочное обучение
Раздел 1				
1	1.1	Технология разработки и сборки мобильного приложения	6	4
2	1.2	Особенности использования механизмов платформы	6	2
3	1.3	Использование мобильного функционала (ПРП)	6	2
Раздел 2				
4	2.1	Особенности разработки интерфейса	4	2
5	2.2	Программирование логики и работа с базами данных	4	4
Раздел 3				
6	3.1	Разработка мобильного клиента для сервиса	4	2
7	3.2	Разработка облачного сервиса	4	2
		Итого	**34**	**18**

6 Самостоятельная работа обучающегося

6.1 Очное обучение

№ п/п	№ раздела	Виды самостоятельной работы	Название (содержание) работы	Объем, часы
1	1.1	Подготовка к лабораторным работам	Технология разработки и сборки мобильного приложения	6
2	1.2	Подготовка к лабораторным работам	Особенности использования механизмов платформы	6
3	1.3	Подготовка к лабораторным работам	Использование мобильного функционала	6
№ п/п	№ раздела	Виды самостоятельной работы	Название (содержание) работы	Объем, часы
4	2.1	Подготовка к лабораторным работам	Особенности разработки интерфейса	4
5	2.2	Подготовка к лабораторным работам	Программирование логики и работа с базами данных	4
6	3.1	Подготовка к лабораторным работам	Разработка мобильного клиента для сервиса	4
7	3.2	Подготовка к лабораторным работам	Разработка облачного сервиса	4
8	1.1-1.3, 2.1-2.2, 3.1-3.2	Расчетно-графическая работа	Разработка мобильного приложения	20

9	1.1-1.3, 2.1-2.2, 3.1-3.2	Самостоятельное изучение теоретического материала	• Настройка интерфейса для мобильного устройства и персонального компьютера • Различия в функциональности начальных страниц • Различия функциональности форм списков • Реализация клиент-серверной логики • Работа с учетными записями пользователей • Демонстрационные базы в мобильном приложении • Создание веб-сервиса регистрации • Создание http-сервиса для обмена данными	32
			Итого	**86**

6.2 Заочное обучение

№ п/п	№ раздела	Виды самостоятельной работы	Название (содержание) работы	Объем, часы
1	1.1	Подготовка к лабораторным работам	Технология разработки и сборки мобильного приложения	6
2	1.2	Подготовка к лабораторным работам	Особенности использования механизмов платформы	6
3	1.3	Подготовка к лабораторным работам	Использование мобильного функционала	6
4	2.1	Подготовка к лабораторным работам	Особенности разработки интерфейса	6
5	2.2	Подготовка к лабораторным работам	Программирование логики и работа с базами данных	6
6	3.1	Подготовка к лабораторным работам	Разработка мобильного клиента для сервиса	4
7	3.2	Подготовка к лабораторным работам	Разработка облачного сервиса	4
8	1.1-1.3, 2.1-2.2,	Расчетно-графическая работа	Разработка мобильного приложения	20
	3.1-3.2			

9	1.1-1.3, 2.1-2.2, 3.1-3.2	Самостоятельное изучение теоретического материала	• Настройка интерфейса для мобильного устройства и персонального компьютера • Различия в функциональности начальных страниц • Различия функциональности форм списков • Реализация клиент-серверной логики • Работа с учетными записями пользователей • Демонстрационные базы в мобильном приложении • Создание веб-сервиса регистрации • Создание http-сервиса для обмена данными • Проектирование и настройка командного интерфейса для мобильного устройства и персонального компьютера.	60
			Итого	**118**

7 Образовательные технологии

Реализация у обучающихся навыков командной работы, межличностной коммуникации, принятия решений, лидерских качеств предусмотрено использование в учебном процессе проведение занятий в виде групповых дискуссий.

№ п/п	№ модуля (раздела)	Наименование темы	Вид учебного занятия	Активные и интерактивные формы проведения обучения
1	3.2	Разработка облачного сервиса	Лекции	Групповые дискуссии

8 Оценочные материалы для проведения промежуточной аттестации обучающихся по дисциплине

Перечень компетенций с указанием этапов их формирования; описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания; типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения образовательной программы; методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности,

характеризующих этапы формирования компетенций) представлены в **Приложение 1 «Фонд оценочных средств по учебной дисциплине».**

9 Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины

а) основная литература

1 Хрусталева, Е. Ю. Знакомство с разработкой мобильных приложений на платформе "1С:Предприятие 8" [Текст] / Е. Ю. Хрусталева - издание 2 – Москва: ООО «1С-Паблишинг», 2015

2 Фигурнов, В. 1С:Облачная подсистема Фреш [Текст] / В. Фигурнов - Москва: ООО «1С-Паблишинг», 2020

б) дополнительная литература (в т.ч. периодические издания)

1 Радченко, М. 1С:Enterprise Development Tools. Руководство разработчика [Текст] / М. Радченко - Москва: ООО «1С-Паблишинг», 2020

2 Радченко М. Г. 1С: Предприятие 8.2. Практическое пособие разработчика. Примеры и типовые приемы [Текст] / М. Г. Радченко, Е. Ю. Хрусталева. - М. : 1С-Паблишинг, 2009. - 872 с. + 1 эл. опт. диск (CD-ROM)

3 Радченко М. Г. 1С: Предприятие 8.2. Коротко о главном. Новые возможности версии
8.2 [Текст] / М. Г. Радченко, Е. Ю. Хрусталева. - М. : 1С-Паблишинг, 2009. - 416 с.

4 1С:Предприятие 8.2. Руководство пользователя [Текст] / [разраб.: А. Алексеев и др.]. - 2-е изд. - М. : Фирма "1С", 2011. - 224 с.

5 1С:Предприятие 8.2. Руководство администратора [Текст] / [разраб. А. Алексеев и др.]. - М. : 1 С, 2009. - 239 с.

6 1С:Предприятие 8.2. Описание встроенного языка [Текст] : в 5 ч. / [разраб.: А. Алексеев и др.]. - М. : Фирма "1С", 2009

7 1С:Предприятие. Версия 8.0. Конфигурирование и администрирование [Текст]. - М. : Фирма "1С", 2004. - 690 с.

10 Профессиональные базы данных и ресурсы «Интернет», к которым обеспечивается доступ обучающихся

Профессиональные базы данных:

1. http://biblio.bsau.ru - Электронная библиотека Башкирского ГАУ;
2. http://znanium.com/ - Электронная библиотечная система; 3. http://elibrary.ru – Электронно-библиотечная система elibrary.
4. https://its.1c.ru/ - Портал информационно-технологического сопровождения программных продуктов фирмы «1С» 1С:ИТС

Ресурсы «Интернет»:

1. https://edu.bsau.ru/ - Система управления обучением Башкирского ГАУ;
2. http://window.edu.ru/ - "Единое окно": доступ к образовательным ресурсам;
3. http://www.gks.ru/ - Федеральная служба государственной статистики.

4. http://devtrainingforum.v8.1c.ru/forum/ - Форум поддержки читателей книги "Практическое пособие разработчика. Примеры и типовые приемы"

Перечень информационно-справочных систем:

1. http://biblio.bsau.ru - Электронная библиотека Башкирского ГАУ;
2. http://www.consultant.ru – Справочная правовая система Консультант плюс;
3. http://garant.ru - Информационно-правовое обеспечение «Система ГАРАНТ».

11 Методические указания для обучающихся по освоению дисциплины (модуля)

Изучаемая дисциплина поделена на 3 раздела, в каждый из которых входят следующие подразделы:

Раздел 1– подразделы 1.1, 1.2, 1.3;
Раздел 2– подразделы 2.1, 2.2.
Раздел 3– подразделы 3.1, 3.2.

Вид учебных занятий	Организация деятельности обучающегося
Вид учебных занятий	Организация деятельности обучающегося
Занятия лекционного типа (лекция)	Написание конспекта лекций: кратко, схематично, последовательно фиксировать основные положения, выводы, формулировки, обобщения; помечать важные мысли, выделять ключевые слова, термины. Проверка терминов, понятий с помощью энциклопедий, словарей, справочников с выписыванием толкований в тетрадь. Обозначить вопросы, термины, материал, который вызывает трудности, пометить и попытаться найти ответ в рекомендуемой литературе. Если самостоятельно не удается разобраться в материале, необходимо сформулировать вопрос и задать преподавателю на консультации, на практическом занятии. .
Занятия семинарского типа (лабораторная работа)	Проработка рабочей программы, уделяя особое внимание целям и задачам структуре и содержанию дисциплины. Решение расчетно-графических заданий, ситуационных заданий, решение задач по соответствующим темам и др.
Самостоятельная работа	Подготовка к занятиям лекционного и семинарского типа. Самостоятельное изучение теоретического материала, основной и дополнительной литературы, включая справочные издания, зарубежные источники и т.д. по разделам (модулям) дисциплины.
Расчетнографическая работа	изучение научной, учебной, нормативной и другой литературы. Отбор необходимого материала; формирование выводов и разработка конкретных рекомендаций по решению поставленной цели и задачи; проведение практических исследований по данной теме. Инструкция по выполнению требований к оформлению работы находится в методических материалах по дисциплине.
Подготовка к экзамену	При подготовке к экзамену необходимо ориентироваться на конспекты лекций, рекомендуемую литературу, список вопросов и заданий,

	приведенных в фонде оценочных средств по дисциплине

Методические указания для обучающихся по освоению дисциплины

№ п/п	Наименование методических указаний, тестов по дисциплине	Назначение (вид занятия, № темы)
1	**Методические указания к** лабораторным работам "Технология разработки мобильного приложения" и самостоятельной работе по дисциплине Б1.В.11 "Разработка мобильных приложений" : [Электронный ресурс] : направление подготовки 09.03.03 Прикладная информатика : квалификация выпускника Бакалавр / Башкирский ГАУ, Каф. информатики и информационных технологий ; сост. Т. М. Шамсутдинова. - Уфа : БГАУ, 2022. - 26 с. - **URL:** http://biblio.bsau.ru/metodic/138154.pdf	ЛР 1-7

12 Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине

№ п/п	Наименование методических указаний, тестов по дисциплине	Назначение (вид занятия, № темы)
1	**Методические указания к** лабораторным работам "Технология разработки мобильного приложения" и самостоятельной работе по дисциплине Б1.В.11 "Разработка мобильных приложений" : [Электронный ресурс] : направление подготовки 09.03.03 Прикладная информатика : квалификация	СИТМ
	выпускника Бакалавр / Башкирский ГАУ, Каф. информатики и информационных технологий ; сост. Т. М. Шамсутдинова. - Уфа : БГАУ, 2022. - 26 с. - **URL:** http://biblio.bsau.ru/metodic/138154.pdf	
2	**Методические указания к** расчетно-графической работе "Разработка мобильного приложения" по дисциплине Б1.В.11 "Разработка мобильных приложений" : [Электронный ресурс] : направление подготовки 09.03.03 Прикладная информатика : квалификация выпускника Бакалавр / Башкирский ГАУ, Каф. информатики и информационных технологий ; сост. Т. М. Шамсутдинова. - Уфа : БГАУ, 2022. - 8 с. - **URL:** http://biblio.bsau.ru/metodic/138155.pdf	РГР

13 Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень программного обеспечения и информационных справочных систем

Внеаудиторное контактное взаимодействие с обучающимися по самостоятельному изучению теоретического материала и выполнению расчетно-графической работы реализовано

в системе управления обучением электронной информационной образовательной среды университета https://edu.bsau.ru .

1 Учебная платформа 1С:Предприятие (Windows) + мобильная платформа [Электронный ресурс]: пакет прикладных программ. – Электрон. дан. и прогр. – фирма «1С», 2020

2 Microsoft Office 2007, 2010 или 2013 Standard Microsoft Open License [Электронный ресурс]: пакет прикладных программ. – Электрон. дан. и прогр. – Microsoft, 2008, 2018, 2015

3 LightShot [Электронный ресурс]: пакет прикладных программ. – Электрон. дан. и прогр. – Skillbrains, 2018

14 Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине

Материально-техническое обеспечение дисциплины

Для проведения лекционных занятий по данной дисциплине используются аудитории, оснащенные мультимедийным оборудованием.

Лабораторные работы проводятся в лабораториях, оснащенных необходимым оборудованием, обеспечивающих получение знаний по дисциплине.

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

.

Материально-техническое обеспечение дисциплины

№ п/п	Наименование	Назначение (виды занятий)
1	*2*	*3*
1	Аудитория для занятий лекционного типа	Лекции
2	Аудитория для занятий семинарского типа	Практические занятия. лабораторные работы
3	Аудитория для групповых и индивидуальных консультаций	Консультации
4	Аудитория для самостоятельной работы обучающегося	Самостоятельная работа обучающихся

Перечень лабораторного оборудования

№ п/п	Наименование	Кол-во, шт.
121/1		
1	Интерактивная .доска SMARTBoard 680	1
2	Компьютер Depo Neos	12
3	Проектор BenQ Multimedia Projector MH FullMW	1

15 Особенности организации обучения по дисциплине для инвалидов и лиц с ограниченными возможностями здоровья

Организация обучения инвалидов и лиц с ограниченными возможностями здоровья (далее ОВЗ) осуществляется на основе адаптированной образовательной программы с использованием специальных методов обучения и дидактических материалов, составленных с учетом особенностей психофизического развития, индивидуальных возможностей и состояния здоровья таких обучающихся (обучающегося).

Образование инвалидов и лиц с ОВЗ может быть организовано как совместно с другими обучающимися, так и в отдельных группах или индивидуально.

Учебно-методические материалы для самостоятельной работы обучающихся из числа инвалидов и лиц с ОВЗ предоставляются в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Категория обучающихся	Формы предоставления материалов
С нарушением слуха	- в печатной форме; - в форме электронного документа.
С нарушением зрения	- в печатной форме увеличенным шрифтом; - в форме электронного документа; - в форме аудиофайла.
С нарушением опорно-двигательного аппарата	- в печатной форме увеличенным шрифтом; - в форме электронного документа; - в форме аудиофайла.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

Для обучающихся инвалидов и лиц с ОВЗ предусмотрены следующие оценочные средства:

Категория обучающих-ся	Виды оценочных средств	Формы контроля и оценки результатов обучения
С нарушением слуха	тест	преимущественно письменная провер-ка
С нарушением зрения	собеседование	преимущественно устная проверка (индивидуально)
С нарушением опорнодвигательного аппарата	решение дистанционных тестов, контрольные вопросы	организация контроля с помощью LMS Башкирского ГАУ, письменная проверка.

Обучающимся инвалидам и лицам с ОВЗ увеличивается время на подготовку ответов к зачѐту, допускается готовить ответы с использованием дистанционных образовательных технологий.

При проведении процедуры оценивания результатов обучения инвалидов и лиц с ОВЗ предусматривается использование технических средств, необходимых им в связи с их индивидуальными особенностями. Эти средства предоставляются ФГБОУ ВО Башкирский ГАУ или могут использоваться собственные технические средства обучающихся.

Процедура оценивания результатов обучения инвалидов и лиц с ОВЗ по дисциплине предусматривает предоставление информации в формах, адаптированных к ограничениям их здоровья и восприятия информации.

Так для лиц с нарушениями зрения:

- в печатной форме увеличенным шрифтом; - в форме электронного документа; - в форме аудиофайла.

Для лиц с нарушениями слуха:

- в печатной форме;
- в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

- в печатной форме;
- в форме электронного документа; - в форме аудиофайла.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

При проведении процедуры оценивания результатов обучения инвалидов и лиц с ОВЗ по дисциплине (модулю) обеспечивается выполнение следующих дополнительных требований в зависимости от индивидуальных особенностей обучающихся:

1. Инструкция по порядку проведения процедуры оценивания предоставляется в доступной форме (устно, в письменной форме, устно с использованием услуг сурдопереводчика).
2. Доступная форма предоставления заданий оценочных средств (в печатной форме, в печатной форме увеличенным шрифтом, в форме электронного документа, задания зачитываются ассистентом, задания предоставляются с использованием сурдоперевода).
3. Доступная форма предоставления ответов на задания (письменно на бумаге, набор ответов на компьютере, с использованием услуг ассистента, устно).

При необходимости для инвалидов и обучающихся с ОВЗ процедура оценивания результатов обучения по дисциплине (модулю) может проводиться в несколько этапов. Проведение процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья допускается с использованием дистанционных образовательных технологий.

Для освоения дисциплины инвалидами и лицами с ОВЗ предоставляются основная и дополнительная учебная литература в виде электронного документа в фонде библиотеки и / или в электронно-библиотечных системах. А также предоставляются бесплатно специальные учебники и учебные пособия, иная учебная литература и специальные технические средства обучения коллективного и индивидуального пользования, а также услуги сурдопереводчиков и тифлосурдопереводчиков.

В зависимости от нозологии для пользователей с ОВЗ организован доступ к электронным информационным и образовательным ресурсам библиотеки университета из любой точки с доступом к «Интернет». Заключен договор о сотрудничестве с Башкирской республиканской специальной библиотекой для слепых. Предоставляется возможность аудио прослушивания и сохранения файла электронных изданий ЭБС «Консультант студента. Электронная библиотека технического вуза» (полные тексты изданий доступны пользователям ФГБОУ ВО Башкирский ГАУ, после самостоятельной регистрации в Электронной библиотечной системе Университета).

Предоставляется возможность пользоваться бесплатным мобильным приложением для операционных систем IOS и Android ЭБС издательства «Лань», с синтезатором речи (возможность использования книг в учебном процессе для незрячих и слабовидящих обучающихся).

В освоении дисциплины инвалидами и лицами с ОВЗ большое значение имеет индивидуальная работа. Под индивидуальной работой подразумевается две формы взаимодействия с преподавателем: индивидуальная учебная работа (консультации), т.е. дополнительное разъяснение учебного материала и углубленное изучение материала с теми обучающимися, которые в этом заинтересованы, и индивидуальная воспитательная работа. Индивидуальные консультации по предмету являются важным фактором, способствующим индивидуализации обучения и установлению воспитательного контакта между преподавателем и обучающимся инвалидом или обучающимся с ОВЗ.

Освоение дисциплины инвалидами и лицами с ОВЗ осуществляется с использованием специальных средств обучения. Оборудовано специализированное помещение, в котором установлен мультимедийный проектор и организовано два рабочих места с доступом к электронной информационной образовательной среде и сети Интернет. Данное помещение оснащено: индукционной петлей ИС-50Л (усиление звука для слабослышащих обучающихся); персональными компьютерами, с программой экранного доступа ("Jaws for Windows 16.0 Pro"), брайлевским дисплеем (тактильный дисплей Брайля PAC Mate 20) для студентов с нарушением зрения; специальными партами для обучающихся с нарушением опорно-двигательного аппарата; мобильным видеоувеличителем; портативной информационной индукционной системой "Исток А2" для слабослышащих обучающихся.

Приложение 1 к рабочей программе дисциплины

Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине

1 Перечень компетенций и этапы формирования компетенций в процессе освоения образовательной программы

Код и наименование компетенции	Код и наименование индикаторов достижения компетенции	Этапы формирования
ПК-2 Способность разрабатывать и адаптировать прикладное программное обеспе-чение	ПК-2.1 Кодирование на языках программирования	7,8

2 Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания

2.1 Показатели и критерии оценивания компетенций

ПК-2 Способность разрабатывать и адаптировать прикладное программное обеспечение ПК-2.1 Кодирование на языках программирования

Планируемые результаты (показатели оценивания)	Критерии оценивания			
	Ниже порогового уровня (неудовл.)	Пороговый уро-вень (удовл.)	Повышенный уровень (хорошо)	Высокий уровень (отлично)
	Не зачтено	Зачтено		

Знания	ПК-2.1/Зн1 основ современных систем управления базами данных, теории баз данных, основ программирования; современных объектноориентированных языков программирования, современных структурных языков программирования, языков современных бизнеспри ложений; ПК-2.1/Зн2 современных методик тестирования разрабатываемых ИС: инструментов и методов Разделного тестирования, инструментов и методов тести-	Отсутствие или фрагментарные знание основ программирования; современных объектноориентированных языков программирования, современных структурных языков программирования, языков современных бизнеспри ложений; современных методик тестирования разрабатываемых ИС: инструментов и методов Разделного тестирования, инструментов и методов тестирования нефункциональных и функциональных характеристик	Неполное знание основ программирования; современных объектноориентированных языков программирования, современных структурных языков программирования, языков современных бизнесприложений; современных методик тестирования разрабатываемых ИС: инструментов и методов Разделного тестирования, инструментов и методов тестирования нефункциональных и функциональных характеристик ИС; теоретических основ web-	В целом сформировавшееся знание основ современных систем управления базами данных, теории баз данных, основ программирования; современных объектноориентированных языков программирования, современных структурных языков программирования, языков современных бизнес-приложений; современных методик тестирования разрабатываемых ИС: инструментов и методов Разделного тестирования, инструментов и методов тестирования не-	Сформировавшееся систематическое знание основ современных систем управления базами данных, теории баз данных, основ программирования; современных объектноориентированных языков программирования, современных структурных языков программирования, языков современных бизнес-приложений; современных методик тестирования разрабатываемых ИС: инструментов и методов Разделного тести-

	рования нефункциональных и функциональных характеристик ИС; ПК-2.1/Зн3 теоретических основ web-технологий; ПК-2.1/Зн4 теоретических основ разработки интернет-ресурсов	ИС; теоретических основ webтехнологий	технологий	функциональных и функциональных характеристик ИС; теоретических основ webтехнологий; теоретических основ разработки интернет-ресурсов	рования, инструментов и методов тестирования нефункциональных и функциональных характеристик ИС; теоретических основ webтехнологий; теоретических основ разработки интернет-ресурсов

Умения	ПК-2.1/Ум1 кодировать на языках программирования; ПК-2.1/Ум2 проводить верификацию кода ИС и баз данных ИС; тестировать результаты кодирования; ПК-2.1/Ум3 применять web-технологии для разработки контента и ИТсервисов предприятия ПК-2.1/Ум4 применять методы, способы и средства получения, хранения, переработки информации с целью разработки интернетресурсов	Отсутствие или фрагментарные умения кодировать на языках программирования; проводить верификацию кода ИС и баз данных ИС; тестировать результаты кодирования; применять webтехнологии для разработки контента и ИТсервисов предприятия; применять методы, способы и средства получения, хранения, переработки информации с целью разработки интернет-ресурсов	Неполное умение кодировать на языках программирования; проводить верификацию кода ИС и баз данных ИС; тестировать результаты кодирования; применять webтехнологии для разработки контента и ИТсервисов предприятия; применять методы, способы и средства получения, хранения, переработки информации с целью разработки интернет-ресурсов	В целом сформировавшееся умение кодировать на языках программирования; проводить верификацию кода ИС и баз данных ИС; тестировать результаты кодирования; применять web-технологии для разработки контента и ИТсервисов предприятия; применять методы, способы и средства получения, хранения, переработки информации с целью разработки интернет-ресурсов	Сформировавшееся систематическое умение кодировать на языках программирования; проводить верификацию кода ИС и баз данных ИС; тестировать результаты кодирования; применять webтехнологии для разработки контента и ИТ-сервисов предприятия; применять методы, способы и средства получения, хранения, переработки информации с целью разработки интернет-ресурсов
Навыки	ПК-2.1/Нв1 разработки кода ИС и баз данных ИС; ПК-2.1/Нв2 верификации кода ИС и баз данных ИС относительно дизайна ИС и структуры баз данных ИС; устранения обнаруженных несоответствий; ПК-2.1/Нв3 использования основ web-технологий для разработки контента и ИТсервисов предприятия ПК-2.1/Нв4 использования инструментальных средств разработки интернет-ресурсов	Отсутствие или фрагментарные навыки разработки кода ИС и баз данных ИС; верификации кода ИС и баз данных ИС относительно дизайна ИС и структуры баз данных ИС; устранения обнаруженных несоответствий; использования основ web-технологий для разработки контента и ИТсервисов предприятия; использования инструментальных средств разработки	Неполные навыки разработки кода ИС и баз данных ИС; верификации кода ИС и баз данных ИС относительно дизайна ИС и структуры баз данных ИС; устранения обнаруженных несоответствий; использования основ web-технологий для разработки контента и ИТсервисов предприятия; использования инструментальных средств разработки интернет-ресурсов	В целом сформировавшиеся навыки разработки кода ИС и баз данных ИС; верификации кода ИС и баз данных ИС относительно дизайна ИС и структуры баз данных ИС; устранения обнаруженных несоответствий; использования основ web-технологий для разработки контента и ИТсервисов предприятия; использования инструментальных средств разработки	Сформировавшиеся систематические навыки разработки кода ИС и баз данных ИС; верификации кода ИС и баз данных ИС относительно дизайна ИС и структуры баз данных ИС; устранения обнаруженных несоответствий; использования основ webтехнологий для разработки контента и ИТ-сервисов предприятия; использования инструментальных средств разработки интернет-ресурсов

		интернет-ресурсов		интернет-ресурсов	

2.2 Шкала оценивания компетенций

Виды оценок	Оценки			
Академическая оценка по 5-ти балльной шкале	Неудовлетворительно	Удовлетворительно	Хорошо	Отлично
	Не зачтено	Зачтено		

2.2 Критерии оценивания по пятибалльной системе

Оценка экзаменатора, уровень	Критерии
«отлично», высокий уровень	Обучающийся показал прочные знания основных положений учебной дисциплины, умение самостоятельно решать конкретные практические задачи повышенной сложности, свободно использовать справочную литературу, делать обоснованные выводы из результатов расчетов или лабораторных работ
«хорошо», повышенный уровень	Обучающийся показал прочные знания основных положений учебной дисциплины, умение самостоятельно решать конкретные практические задачи, предусмотренные рабочей программой, ориентироваться в рекомендованной справочной литературе, умеет правильно оценить полученные результаты расчетов или лабораторных работ

«удовлетворительно», пороговый уровень	Обучающийся показал знание основных положений учебной дисциплины, умение получить с помощью преподавателя правильное решение конкретной практической задачи из числа предусмотренных рабочей программой
«неудовлетворительно», ниже порогового уровня	При ответе обучающегося выявились существенные пробелы в знаниях основных положений учебной дисциплины, неумение с помощью преподавателя получить правильное решение конкретной практической задачи из числа предусмотренных рабочей программой учебной дисциплины

3 Фонд оценочных средств для оценки достижения обучающимися результатов освоения дисциплины

Раскрываемые компетенции:

ПК-2 Способность разрабатывать и адаптировать прикладное программное обеспечение

ИДК - ПК-2.1 Кодирование на языках программирования

1. Что вызывает ошибку в этом фрагменте кода? Как вы могли бы ее исправить?

```
let n1: Int = 1

let n2: Float = 2.0

let n3: Double = 3.34

var result = n1 + n2 + n3
```

Ответ

В Swift неявное приведение типов между двумя типами данных невозможно.

В приведенном выше коде вы пытаетесь сложить вместе три элемента, каждый из которых представляет собой другой тип данных.

Чтобы исправить это, необходимо преобразовать каждое значение к одному типу данных. Например:

```
var result = Double(n1) + Double(n2) + n3
```

2. Каким будет значение переменной len? Почему?

```
var arr1 = [1, 2, 3]

var arr2 = arr1



arr2.append(4)



var len = arr1.count
```

Ответ
Значение len равно 3, т.е. количество элементов в arr1 равно 3. Это происходит потому, что присвоение arr1 к arr2 фактически означает, что копия arr1присваивается к arr2, поэтому arr1 не затрагивается.

В Swift все основные типы данных (булевы, целые числа и т. д.), перечисления и структуры по своей природе являются типами значений.

Массивы тоже являются типами значений. Перемещая тип значения в Swift, вы, по сути, работаете с его копией. Например, при передаче типа значения в качестве аргумента функции создается его копия, поэтому все, что делает функция, не отражается на исходном значении.

3. В чем здесь проблема и как вы можете ее решить?

Рассмотрим этот фрагмент кода, который пытается получить цвет темы из локального хранилища устройства iOS:

```
var color = UserDefaults.standard.string(forKey: "themeColor")!
```

```
print(color)
```

Смогли ли вы заметить ошибку и исправить ее?
Ответ
Первая строка извлекает цвет темы из *user defaults*. Этот метод, однако, возвращает *optional* (поскольку themeColor может быть не определен). Если ключ не найден, возвращается nil, что приводит к крашу:

```
fatal error: unexpectedly found nil while unwrapping an Optional value
```

Это происходит потому, что в первой строке используется ! для *force unwrapoptional*, которое теперь nil. *Force unwrapping* должно использоваться только тогда, когда вы на 100% уверены, что значение не nil.

Чтобы исправить это, вы можете использовать *optional binding* для проверки, найдено ли значение для ключа:

```
if let color = defaults.stringForKey("themeColor") {
    print(color)
}
```

4. Какие потенциальные улучшения вы здесь видите?

Вы просматриваете пулл-реквест и столкнулись с этим методом:

```
func turnTo(direction: String){
    if direction == "North" {
        northAction()
    } else if direction == "East" {
```

```
        eastAction()
    } else if direction == "South" {
        southAction()
    } else if direction == "West" {
        westAction()
    } else {
        print("No valid direction specified")
    }
}
```

Какие улучшения вы можете предложить автору кода?

Ответ

Даже если этот код может работать, есть два момента, которые следует учитывать.

- Использование жестко закодированных строк типа (например, "West") - плохая идея. Что если кто-то неправильно напишет это слово? Чтобы решить эту проблему, следует отказаться от жестко закодированных строк и вместо них использовать перечисление.
- Кроме того, как насчет использования оператора switch вместо длинного оператора if-else?

Благодаря этим улучшениям код станет более безопасным и читабельным:

```
enum Direction {
    case North
    case East
    case South
```

```
    case West
}
func turnTo(direction: Direction){
    switch direction {
    case .North: northAction()
    case .East: eastAction()
    case .South: southAction()
    case .West: westAction()
    default:
        print("No valid direction specified")
    }
}
```

5. Что такое перечисления (enumerations) в Swift?

Перечисление (enumeration)- это группа связанных значений.

Перечисления позволяют писать безопасный для типов код.

```
enum Direction {
    case North
    case East
```

```
    case South

    case West

}
```

Теперь в своем коде вы можете вызвать, например, Direction.North, вместо того чтобы использовать мистическую строку "North" (которая легко может быть неправильно написана и вызвать раздражающие ошибки).

Больше информации о перечислениях можно прочесть в этой статье.

6. Что такое Optional в Swift? Как его создать?

Optional - это тип, который может хранить либо значение, либо nil. Вы можете создать optional, добавив вопросительный знак ? после любого типа:

```
var number: Int? = 10
```

7. Что такое typealias в Swift? Как его можно создать?

Typealias, как следует из названия, является псевдонимом для существующего типа данных.

Вы можете создать его следующим образом:

```
typealias Weight = Float
```

Теперь вы можете использовать Weight вместо Float:

```
let mass1: Weight = 150.0

let mass2: Weight = 220.0
```

```
let total: Weight = mass1 + mass2
```

8. Назовите некоторые преимущества использования Swift.

Вот лишь некоторые из них:

- Swift - язык с типобезопасностью
- В нем есть поддержка замыканий
- Поддерживаются опциональные типы
- Встроенная обработка ошибок
- Поддерживается сопоставление шаблонов

9. Назовите 5 утверждений передачи управления (Control Transfer Statements) и опишите, как их использовать.

Вот они ~~слева-направо~~ (сверху-вниз):

- Break
- Continue
- Fallthrough
- Throw
- Return

Операторы передачи управления изменяют порядок выполнения вашего кода.

Например, вы можете использовать оператор передачи управления break для завершения выполнения цикла for, когда продолжение цикла считается ненужным:

```
for choice in choices:

    if isCorrect(choice):

        print("Correct choice found!")

        break
```

10. Предложите небольшую доработку для следующего кода.

```
if age >= 18 {

    driveCar()
```

```
} else {

   doNotDrive()

}
```

Этот код хорошо работает - но можете ли вы предложить небольшое улучшение рефакторинга, чтобы сделать его еще лучше?
Ответ
Вы можете использовать тернарный условный оператор для преобразования этого выражения в однострочное, что в данном случае не ухудшает читабельность, а улучшает ее.

```
age >= 18 ? driveCar() : doNotDrive()
```

11. Как можно улучшить читаемость кода?

В нашей компании 20 разработчиков и 20 уникальных стилей кодирования. Как мы можем внедрить некоторые общие стили кодирования/лучшие практики?
Ответ
Можно использовать линтер, например, **Swiftlint**. **Линтер** - это простой в настройке инструмент, который проверяет и исправляет ваши ошибки и внедряет лучшие практики и соглашения от вашего имени.

Вы можете использовать рекомендации по умолчанию, но также можете настроить его в соответствии с предпочтениями вашей компании.

12. Зачем нужен completion handler в Swift?

Completion handlers (обработчики завершения) — это замыкания в действии. Предположим, вы выполняете трудоемкую задачу, например сетевой запрос, и хотите что-то сделать сразу после завершения запроса.

Но вы определенно не хотите тратить ресурсы впустую, проверяя несколько раз, продолжается ли процесс или нет. Здесь используются обработчики завершения. Обработчик завершения — это замыкание, которое «вернется» сразу после завершения трудоемкого процесса. Узнайте больше о замыканиях и о том, как передать функцию в качестве параметра.

13. Как тестировать приложение без физического устройства?

Если у вас нет устройства iOS, вы можете использовать симуляторы устройств iOS от Apple для тестирования своих приложений на Mac.

14. Что делает init() в Swift?

Метод `init()` используется для инициализации экземпляра.

Инициализация означает подготовку экземпляра (класса, структуры или перечисления) к использованию.

В процессе инициализации вы устанавливаете начальные значения для каждого свойства экземпляра. Вы также можете выполнить некоторые другие подготовительные действия, прежде чем экземпляр будет готов к использованию.

15. Let и Var в Swift?

В **Swift** вы можете использовать **let** для создания константы (значения, которое нельзя изменить) и **var** для создания переменной (значения, которое может быть изменено позже).

Некоторые дополнительные подробности вы можете найти в этой статье.

16. Что такое plist?

Plist, или список свойств, - это словарь пар ключ-значение, которые используются для хранения данных в файловой системе вашего проекта. Например, **info.plist**.

17. Для чего нужны Protocols в Swift? Приведите пример.

Протокол действует как чертеж для свойств, методов и т.д. Он описывает, как должен вести себя тип, соответствующий ему.

Вы не можете создавать экземпляры протоколов. Скорее, вы можете сделать так, чтобы, например, класс соответствовал протоколу.

Вот пример протокола, описывающего животное:

```
protocol Animal {

   var name: String { get set }

   var color: String { get set }
```

```
    func makeSound()

}
```

Давайте создадим классы Cat и Dog, которые оба соответствуют протоколу Animal. Таким образом, требуется, чтобы они оба реализовывали поведение, описанное в протоколе Animal- то есть переменные name, color и метод makeSound():

```
class Cat: Animal {

    var name = "Luna"

    var color = "gray"

    func makeSound() {

        print("Meow!")

    }

}

class Dog: Animal {

    var name = "Charlie"

    var color = "black"

    func makeSound() {

        print("Woof!")

    }
```

```
}
```

18. Для чего нужен оператор вида «??» ?

Оператор двойного вопросительного знака ?? известен как оператор объединения (слияния) nil. Он возвращает значение в левой части, если оно не равно nil. Если левая часть равна nil, то возвращается значение в правой части.

Его можно использовать как сокращение для проверки того, является ли опциональное значение nil. Например, вы можете заменить это:

```
var name: String?



if name != nil {

    print(name)

} else {

    print("N/A")

}
```

На это:

```
print(name ?? "N/A")
```

19. Для чего используется Guard?

Оператор guard используется для передачи управления программой за пределы области видимости. Оператор guard похож на оператор if, но он запускается только тогда, когда некоторые условия не выполняются.

Например, оператор guard используется для выхода из функции:

```
func myFun() {

   guard false else {

      print("This block is run")

      return

   }

   print("This is never run")

}



myFun()
```

Вывод:

```
This block is run
```

Узнайте больше о ключевом слове guard, прочитав эту статью.

20. Каковы три основных типа коллекций в Swift?

- **Массивы**: Массив - это упорядоченная коллекция значений.
- **Наборы**: Набор - это неупорядоченная коллекция значений.
- **Словари**: Словарь - это неупорядоченная коллекция пар ключ-значение.

21. Для чего используется Defer в Swift?

Вы можете использовать метод defer для выполнения кода перед выходом из области видимости. В качестве примера, давайте напечатаем что-нибудь прямо перед завершением выполнения функции:

```
func printStuff() {
    defer {
        print("I some printed numbers and now I exit the scope")
    }
    print("4")
}
printStuff()
// Output:
// 4
// I some printed numbers and now I exit the scope
```

Defer обычно используется при открытии и закрытии контекста внутри области видимости - например, при доступе к файлам.

22. Можно ли поменять местами две переменные без третьей переменной-помощника?

Это классический вопрос на собеседовании Swift.

Да, это возможно.

С помощью tuple destructuring вы можете решить проблему следующим образом:

```
var a = 1
var b = 2
```

```
(a, b) = (b, a)
```

23. В чем разница между структурами и классами?

- Структуры - это типы значений, в то время как классы - ссылочные типы.
- Структуры не поддерживают наследование, а классы поддерживают.
- В классе мы можем создать экземпляр с помощью ключевых слов let и попытаться изменить его свойство, в то время как в структурах такой возможности нет.
- Структуры не поддерживают приведение типов, а классы поддерживают.

24. Что такое необязательная цепочка (Optional Chaining) ?

Необязательная цепочка означает, что вы можете безопасно вызвать свойство чего-то, что может быть nil.

Optional chaining работает, как следует из названия, путем объединения одного или нескольких необязательных значений с помощью оператора со знаком вопроса ?, например, так:

```
something?.someValue?.someMethod()
```

Если nil встречается в любой точке вышеприведенной цепочки, приложение не крашится - вместо этого возвращается nil.

25. Что такое опциональное связывание (optional binding) ?

Опциональное связывание проверяет, содержит ли опция значение или нет. Если опция имеет значение, опциональное связывание делает это значение временно доступным:

Например, следующий код проверяет, является ли имя nil или нет. Если нет, то создается временная константа realName и ей присваивается значение name.

```
var name: String? = "Charles"



if let realName = name {

   print (realName)
```

```
}
```

Вывод:

```
Charles
```

26. Объясните архитектуру MVC

MVC (Model-View-Controller) - это программная архитектура для разработки приложений для iOS. Это одна из фундаментальных концепций разработки приложений для iOS.

Множество iOS-фреймворков используют MVC.

Идея MVC заключается в передаче данных из одного места в другое. Это означает, что любой объект попадает в одну из этих трех категорий:

- **Model:** Модель представляет данные приложения. Она хранит информацию, например, товары в магазине. Модель управляет состоянием приложения.
- **View:** Вью отвечает за отображение и взаимодействие с пользовательским интерфейсом. Например, вью отображает таблицу товаров для пользователя вашего приложения.
- **Controller:** Контроллер - это то, что склеивает модель и представление. Он отвечает за управление логикой, которая происходит между ними.

27. Что такое параметр In-Out в Swift?

Параметр inout позволяет изменять значение параметра внутри функции.

Чтобы сделать параметр in-out, используйте ключевое слово inout перед типом параметра.

Чтобы передать переменную в качестве in-out, используйте & перед ее именем.

Например:

```
func change(_ number: inout Int){

   number = 2

}
```

```
var number = 1

change(&number)

print(number)

// Output:

// 2
```

Здесь можно прочитать подробнее о параметрах inout.

28. Что такое tuple? Продемонстрируйте, как работать с ними

Tuple (кортеж) - это значение, которое можно использовать для объединения нескольких значений вместе, например, в виде пары.

Значения tuple не обязательно должны быть одного типа.

Вы можете создать tuple, разделив значения запятыми внутри круглых скобок.

Например:

```
var coordinates3D = (1.0, 2.0, 5.0)
```

Чтобы получить доступ к значению внутри tuple, используйте точечную нотацию и индекс:

```
let xPos = coordinates3D.0
```

Кортежи также могут быть созданы таким образом, чтобы каждое значение имело имя:

```
var coordinates3D = (x: 1.0, y: 2.0, z: 5.0)
```

В этом случае вы можете получить доступ к определенному значению кортежа по его имени:

```
let xPos = coordinates3D.x
```

29. Что такое Swift Messages?

Swift Messages - это библиотека, используемая для отображения сообщений в виде строки состояния в верхней или нижней части экрана устройства iOS.

30. Можно ли задать параметру функции значение по умолчанию?

Можно задать значение по умолчанию для параметра:

```
func eat(food: String = "spaghetti") {

   print("Yum! I ate some good \(food).")

}
```

31. Что такое дженерики? Приведите пример использования дженериков

Дженерики позволяют писать гибкий и многократно используемый код, который может работать с любым типом данных.

Представьте, что вы пишете трехмерную векторную структуру, но хотите иметь возможность создавать векторы, используя целые, плавающие и двойные числа. Вы определенно не хотите писать один и тот же код для каждого типа данных отдельно.

Именно здесь вы можете использовать дженерики.

Например, вы можете создать общий тип для параметров (для представления любого типа), используя букву, например T, следующим образом:

```
struct Vec3D<T> {

   let x, y, z: T
```

```
    init(x: T, y: T, z: T) {
        self.x = x
        self.y = y
        self.z = z
    }
}
let intVector = Vec3D(x: 1, y: 2, z: 5)
let floatVector = Vec3D(x: 1.0, y: 2.0, z: 5.0)
```

32. Чем будет свойство pounds в следующем примере?

```
class Weight {
    var kilograms: Float = 0.0
    var pounds: Float {
        get {
            return (kilograms * 2.205)
        }
        set(newWeight) {
            kilograms = newWeight / 2.205
```

```
        }

    }

}

let weight = Weight()

weight.kilograms = 100

print(weight.pounds) // prints '220.5'

weight.pounds = 315

print(weight.kilograms) // prints '142.85715'
```

Ответ

33. В чем разница между операторами == и ===?

- == - оператор равенства.
- === - оператор тождества.

Оператор равенства == используется для проверки равенства двух типов Equatable:

```
"Hello" == "Hello"

10.0 == 5.0 + 5.0
```

Оператор тождества === может быть использован для проверки идентичности двух классов, т.е. указывают ли они на один и тот же адрес памяти. Рассмотрим пример:

```
class Fruit {

    var name = "Banana"

}
```

```
let fruit1 = Fruit()

let fruit2 = fruit1 // fruit2 now points to same address as fruit



fruit1 === fruit2 // true
```

Узнайте больше о разнице между `==` и `===` **здесь.**

34. Что такое расширения?

В Swift вы можете использовать расширения для добавления функциональности к существующему типу.

В Swift вы можете создать расширение с помощью ключевого слова `extension`:

```
extension SomeExistingType {

    // add new functionality here

}
```

35. Что такое вложенная функция?

Вложенная функция - это комбинация функции внутри функции:

```
func outer() {

    func inner() {

        // Do something here
```

```
    }

}
```

36. Как создать базовый класс в Swift?

Вы можете создать базовый класс, просто определив класс без суперкласса.

37. Что такое Force Unwrapping (принудительное разворачивание) ? Когда его следует использовать?

Force Unwrapping (принудительное разворачивание) пытается преобразовать опциональное значение в значение независимо от того, содержит оно значение или нет.

Принудительное разворачивание небезопасно, потому что если опция nil и вы попытаетесь ее развернуть, это вызовет ошибку, которая приведет к краху приложения. Таким образом, ее следует избегать, если вы не уверены на 100%, что опция не является nil.

38. Перечислите преимущества функций высшего порядка.

- Они обеспечивают гибкость.
- Они полезны в асинхронных вызовах, где обычные функции не могут быть использованы.
- Иногда они улучшают качество кода и делают его короче и лаконичнее.

39. Fileprivate vs Private?

- Свойство fileprivate может быть прочитано в любом месте того же файла Swift, но не за его пределами.
- Свойство private можно прочитать только внутри типа, в котором оно было объявлено (а также в расширениях этого типа в том же файле).

Подробнее о private и fileprivate **здесь.**

40. Какие функции есть в Swift?

Функция позволяет определять многократно используемые блоки кода. Функция может выполнять задачу, которая является частью вашей программы.

Обычно функции принимают некоторые значения, с которыми они могут работать.

41. Nil vs None в Swift?

Между ними нет разницы:

```
nil == .none // returns true
```

Пожалуй, единственное "отличие" заключается в том, что использование nil встречается чаще, чем использование none.

42. Что такое dictionary (словарь) в Swift?

Словарь - это основной тип коллекции в Swift. Он может использоваться для хранения пар ключ-значение.

Вы можете легко получить доступ к значению, зная ключ:

```
let dict = ["a": 1, "b": 2]

let valueOfA = dict["a"]
```

43. Что делает ключевое слово Mutating?

Вы можете использовать ключевое слово mutating, чтобы разрешить изменение свойств структуры в методе, пометив этот конкретный метод mutating.

Например:

```
struct Fruit {

   var type: String

   mutating func convertToBanana() {

      self.type = "Banana"

   }

}

var fruit = Fruit(type: "Apple")
```

```
fruit.convertToBanana()

print(fruit.type) // prints "Banana"
```

По умолчанию это невозможно для типов значений (структур и перечислений), но возможно для ссылочных типов (классов).

44. Можете ли вы устранить проблему в этом коде?

Приведенный ниже код выдает ошибку компилятора. Что не так? Как вы можете это исправить?

```
struct Apple {}

func pick(apple: Apple?) {

  guard let apple = apple else {

    print("No apple found!")

  }

  print(apple)

}
```

Ответ

45. Что такое Deinitializer (деинициализатор) ? Как его создать?

Деинициализатор запускается до того, как экземпляр класса будет деаллоцирован.

Вы можете создать деинициализатор, используя ключевое слово `deinit`.

Этот метод полезен только в том случае, если вам нужно сделать некоторую уборку перед деаллокацией экземпляра класса. В большинстве случаев достаточно позволить Swift сделать это автоматически от вашего имени.

Вот пример деинициализатора, который устанавливает number обратно в 0 при деаллокации экземпляра Example.

```
var number = 15



class Example {

   init() {

      number *= 10

   }



   deinit {

      number = 0

   }

}
```

46. В чем разница между функциями и методами?

Между функцией и методом есть небольшая разница. Оба являются многократно используемыми фрагментами кода, однако методы принадлежат классам, структурам или перечислениям, а функции - нет.

47. Как запретить наследование класса?

Сделать класс конечным, используя ключевое слово final. Например:

```
final class Animal {
```

```
    let name = "I'm a furry animal"

}
```

Подробнее о преимуществах final можно прочитать **здесь.**

48. Что такое Lazy Variables (ленивые переменные)? Когда их следует использовать?

Начальное значение ленивой переменной вычисляется при первом обращении к ней. Ленивые переменные можно использовать для оптимизации кода, не выполняя ненужную работу раньше времени.

Например:

```
lazy var tallest: Person? = {

    return people.max(by: { $0.height < $1.height })

}()
```

Чтобы узнать больше о lazy, ознакомьтесь с этой статьей.

49. Что такое (autoclosure) автозамыкание в Swift? Как и когда его следует использовать?

Автозамыкание оборачивает аргумент функции в замыкание.

Когда вызывается autoclosure, оно возвращает значение выражения, завернутого внутрь.

Автозамыкание - это не что иное, как синтаксическое удобство для написания более чистого кода.

Иногда синтаксически удобно использовать `autoclosure` при работе с функцией, которая принимает аргумент замыкания.

Это происходит потому, что `autoclosure` позволяет не использовать фигурные скобки `{}`.

Это может сделать код более читабельным.

Однако помните, что Apple говорит об использовании автозамыканий :
Обычно принято вызывать функции, которые принимают autoclosure, но не принято реализовывать такого рода функции.

Вот пример того, как autoclosure упрощает код. В первом фрагменте используется обычное замыкание, а во втором - autoclosure. Посмотрите, как вызов функции I_will стал более читабельным во втором фрагменте:

```
func I_will(_ perform_action: () -> Void) {
    perform_action()
}
I_will({
    print("Hello, world!")
})
func I_will(_ perform_action: @autoclosure () -> Void) {
    perform_action()
}
I_will(print("Hello, world"))
```

Как видите, вызов функции I_will больше не требует использования фигурных скобок.

50. Чего не хватает в этом фрагменте кода?

```
enum Example {
  case something(Int, Example)
```

```
}
```

Ответ

В Swift можно создавать рекурсивные перечисления, подобные приведенным выше. Ниже приведен абстрактный пример - однако использование рекурсивных перечислений по умолчанию отключено, вам необходимо включить его с помощью ключевого слова indirect:

```
enum Example {

  indirect case something(Int, Example)

}
```

51. Объясните процесс сборки в Android.

Процесс сборки в Android состоит из трех этапов:

- Первый шаг состоит из компиляции папки ресурсов с помощью инструмента упаковки активов Android (AAPT). Они компилируются в один файл класса, известный как R.java, который содержит только константы.
- На втором этапе исходный код java необходимо скомпилировать в файлы .class с использованием javac, которые затем преобразуются в байт-код Dalvik с помощью инструмента ‘dx’, который является одним из инструментов в наборе для разработки программного обеспечения. Окончательный выходной файл - classes.ex.
- На третьем и последнем этапе Android apk Builder должен принять все исходные данные и создать файл Android Packaging Key (APK).

52. Назовите несколько языков программирования Android.

Ниже приведен список самых популярных языков программирования, которые можно использовать для разработки приложений для Android:

- **Java**: Один из самых популярных языков программирования, Java всегда был отправной точкой для новых разработчиков и используется многими, кто занимается разработкой для Android.
- **Kotlin**: Kotlin - относительно новый, современный, безопасный и объектно-ориентированный кроссплатформенный язык программирования. Когда в октябре 2017 года была выпущена Android Studio 3.0, Kotlin был объявлен официальным языком программирования для Android. Многие популярные приложения, такие как Trello, Square и Corda, с тех пор перешли на Kotlin.
- **C #**: используя язык C #, разработчики могут создавать собственные мобильные приложения для iOS и Android.
- **Python**: Python стал одним из самых популярных языков программирования в последнее время. Динамичный и объектно-ориентированный язык программирования Python очень популярен в машинном обучении.

53. Какие различные инструменты доступны в разработке Android? Объясните их функции.

В помощь разработчикам Android доступно множество инструментов:

- **Комплект для разработки программного обеспечения для Android (SDK) и диспетчер виртуальных устройств: Этот инструмент используется для создания виртуальных устройств Android (AVD) и SDK-файлов и их обработки. С помощью эмулятора в AVD вы можете указать поддерживаемую версию SDK, хранилище на SD-карте, разрешение экрана и другие возможности, такие как GPS и сенсорный экран.**

- **Эмулятор Android: AE - это реализация виртуальной машины Android, предназначенная для запуска процессов внутри самого виртуального устройства, которое можно использовать на компьютере разработчика. Основное применение этого инструмента - тестирование и отладка приложений для Android.**
- **Android Debug Bridge (ADB): ADB - это приложение для отладки командной строки, поставляемое вместе с SDK. Это позволяет разработчикам взаимодействовать с устройством и облегчает такие действия, как установка и отладка приложения.**
- **Инструмент упаковки активов Android (AAPT):** AAPT создает файл пакета Android, распространяемый в формате .apk.

54. Объясните язык определения интерфейса Android.

Язык определения интерфейса Android или AIDL облегчает коммуникацию между клиентом и сервисом. Для процедуры обмена данными между процессами данные разбиваются на небольшие части, которые легко распознаются платформой Android.

55. Опишите папку, файл и описание приложений для Android

Ниже приведены краткие пояснения к этим концепциям:

- **gen**: gen содержит сгенерированный компилятором файл .R, который ссылается на все ресурсы проекта
- **src:** src содержит исходные файлы .java в нашем проекте
- **bin:** bin содержит файл .apk, созданный ADT в процессе сборки, а также все другие вещи, необходимые для запуска приложения для Android
- **AndroidManifest.xml:** Этот файл представляет собой файл манифеста, который объясняет основные функции приложения и определяет все его компоненты
- **res / values:** res / values - это каталог для других различных XML-файлов, содержащих такие ресурсы, как строки, определения цветов и многое другое
- **res / drawable-hdpi:** Это каталог объектов, которые можно рисовать и предназначены для экранов с высокой плотностью изображения
- **res / layout:** Это каталог файлов, которые определяют пользовательский интерфейс вашего приложения

56. Что такое ‘действия’? Опишите жизненный цикл действия.

Действия называются окном в пользовательском интерфейсе. Это помогает отображать выходные данные или может даже запрашивать входные данные, чтобы можно было выполнять диалоговые окна и другие роли для создания пользовательского интерфейса. Жизненный цикл действия выглядит следующим образом

- **onCreate ():** создаются просмотры и собираются данные из пакетов
- **OnStart():** вызывается, если действие видно пользователю. За ним может последовать onResume (), если действие выходит на передний план, и onStop (), если оно преобразуется в скрытое.
- **onResume():** вызывается, когда действие начинает взаимодействие с пользователем

- **onPause ():** вызывается, когда действие переходит в фоновый режим, но еще не было остановлено
- **onStop ():** вызывается, когда вы больше не видны пользователю
- **onDestroy():** вызывается по завершении действия
- **onRestart():** вызывается после остановки действия, прежде чем оно будет запущено снова

57. Назовите некоторые преимущества Android.

Вот некоторые преимущества Android:

- **Низкие инвестиции и более высокая отдача:** разработка Android имеет низкий барьер для входа и подходит для новых разработчиков, желающих стать опытными в области программирования
- **Бесплатный SDK:** Одной из наиболее заметных особенностей Android является то, что набор для разработки программного обеспечения имеет открытый исходный код и предоставляется бесплатно, что исключает затраты на лицензирование, распространение и разработку
- **Простота внедрения:** приложения для Android написаны на Java, которая является одним из наиболее используемых языков программирования в мире
- **Многоразовое использование:** компоненты Android могут быть использованы повторно и даже заменены фреймворком
- **Поддержка нескольких платформ:** Платформа Android поддерживает основные операционные системы, такие как Linux, Mac OS и Windows
- **Поддержка носимых устройств:** рынок сейчас наводнен носимыми устройствами, и Android стал лидером по поддержке таких устройств, которые теперь легко доступны на рынке

58. Каковы некоторые недостатки Android?

Ниже приведены некоторые недостатки операционной системы Android

- **Поддельные приложения:** В любой момент времени на рынке доступны тысячи поддельных приложений, которые при установке могут попытаться украсть ваши данные.
- **Проблемы с оптимизацией:** На рынке доступно множество устройств Android с разными размерами экрана, но, что более важно, с разными операционными системами Android. Каждый разработчик приложений должен постоянно работать над обновлением своего приложения для новой ОС, но с различными версиями ОС и обновлениями этот процесс довольно сложный. Приложение, которое бесперебойно работает на одной версии ОС Android, может выйти из строя на другой ОС Android.
- **Фоновые процессы:** Обилие запущенных процессов в фоновом режиме всегда является проблемой, поскольку они быстро разряжают батарею.

59. Определите архитектуру Android.

Архитектура Android состоит из пяти компонентов. Это:

- **Ядро Linux:** Ядро Linux составляет основу платформы Android, и оно поддерживает такие функции, как управление памятью и питанием, а также различные драйверы. Он служит уровнем абстракции перед другими уровнями.

- **Библиотеки платформы:** Библиотеки платформы Android - это собственные библиотеки C и C ++, которые обеспечивают поддержку графики и мультимедиа, а также библиотека WebKit. Это позволяет разработчикам, среди прочего, реализовывать графическую функциональность и отображать веб-контент.
- **Среда выполнения Android:** Среда выполнения Android (ART) и основные библиотеки являются одними из наиболее важных частей архитектуры. Она служит основой для платформы приложений и обладает такими функциями, как оптимизированная сборка мусора.
- **Приложения для Android:** Это то, что вы видите при использовании своего телефона, и это верхний уровень архитектуры. Основные приложения, такие как электронная почта, SMS и контакты, поставляются с предустановленной версией.
- **Платформа приложения: Платформа приложения содержит классы, которые используются при создании приложения. Они предоставляют строительные блоки, с помощью которых вы можете создавать приложения и предоставлять такие сервисы, как менеджер ресурсов, менеджер уведомлений и менеджер активности.**

60. Что такое "Эмулятор”?

"Эмулятор" в Android помогает разработчикам играть с интерфейсом, который действует как настоящее мобильное устройство. Таким образом, разработчикам становится проще писать и тестировать различные коды для приложения. Процесс отладки также становится возможным с помощью эмуляторов. Таким образом, эмуляторы обеспечивают безопасную платформу для тестирования кодов на ранних этапах, а также на более поздних стадиях, когда необходимо устранить ошибки.

61. Что такое Activitycreator?

Activitycreator - это начальный шаг к созданию проекта Android. Он состоит из сценария оболочки, который можно использовать для создания новой файловой системы, необходимой для написания кодов в системе Android.

62. Что такое Android Runtime?

Android Runtime (ART) - это приложение, используемое ОС Android в качестве среды выполнения. Теперь оно заменило Dalvik, снятую с производства виртуальную машину (VM). ART переводит байт-код приложения в собственные инструкции, которые выполняются средой выполнения устройства.

63. Объясните термин ANR.

Термин ANR является сокращением от "Приложение не отвечает". ОС Android отображает его в виде уведомления каждый раз, когда приложение перестает отвечать на действия пользователя в течение значительного периода времени.

64. Объясните, какие диалоговые окна поддерживаются на Android.

Android поддерживает четыре диалоговых окна:

- **AlertDialog:** AlertDialog поддерживает 0-3 кнопки, а также список выбираемых элементов, таких как переключатели и флажки
- **DatePickerDialog:** используется для выбора даты пользователем
- **TimePickerDialog:** используется для выбора времени пользователем
- **ProgressDialog:** используется для отображения индикатора выполнения и является расширением AlertDialog. Он также поддерживает добавление кнопок.

65. Что такое фреймворки Android?

Фреймворки Android представляют собой набор API, упрощающих процесс разработки. Разработчики могут быстро создавать приложения, поскольку API предоставляют такие инструменты, как намерения, текстовые поля и многое другое. По сути, это набор программных инструментов, который позволяет быстро создавать каркас приложения.

66. Что такое API в Android?

Интерфейс прикладного программирования (API) - это просто набор спецификаций или правил, которые определяют, как программные компоненты могут взаимодействовать друг с другом. Компании, выпускающие API, делают это для того, чтобы разработчики могли разрабатывать продукты на основе предоставляемых им сервисов.

67. Что такое датчики в Android?

Устройства на базе Android имеют ряд встроенных датчиков, которые измеряют определенные параметры, такие как движение, ориентация и многое другое. Эти датчики помогают отслеживать позиционирование и перемещение устройства с высокой точностью. В зависимости от природы они могут быть как программными, так и аппаратными. Три основные категории датчиков в устройствах Android:

- **Датчик положения:** используется для измерения физического положения устройства Android. Сюда входят датчики ориентации и магнитометры
- **Датчики движения: Эти датчики включают в себя датчики силы тяжести, активности вращения и ускорения, которые измеряют вращение устройства или ускорение и многое другое.**
- **Датчик окружающей среды: включает в себя датчики, которые измеряют температуру, давление, влажность и другие факторы окружающей среды**

68. Назовите несколько сценариев тестирования для реальных устройств, а не на эмуляторах.

Эмуляторы - это устройства, которые используются для выполнения задач, сопоставимых с задачами реальных устройств Android, и используются для снижения стоимости тестирования.

Но некоторые сценарии могут быть выполнены только на реальных устройствах. Эти сценарии включают:

- Обмен сообщениями
- Bluetooth
- Установка и размонтирование карты памяти
- Проверка сценариев работы от батареи
- Проблемы, связанные с памятью
- Проверка эффективности

69. Что такое контекст?

Контекст в Android, как следует из названия, - это контекст текущего состояния вашего приложения или объекта. Контекст поставляется с такими сервисами, как предоставление доступа к базам данных и настройкам, разрешение ресурсов и многое другое.

Существует два типа контекста:

- **Контекст действия:** Этот контекст привязан к жизненному циклу действия. Его следует использовать, когда вы передаете контекст в рамках действия или вам нужен контекст, жизненный цикл которого привязан к текущему контексту.

70. Как вы обнаруживаете утечки памяти в приложении на платформе Android?

Диспетчер устройств Android (ADM) помогает находить утечки памяти в приложении на Android. Когда вы открываете ADM в Android Studio, вы можете увидеть такие параметры, как размер кучи и анализ памяти, а также многие другие во время запуска приложения.

71. Укажите архитектуру приложения для Android.

Любое приложение для Android содержит следующие компоненты:

- **Уведомление:** Предлагает такие функции, как свет, звук, иконки и т.д.
- **Сервисы:** выполняет фоновые функции
- **Намерение:** обеспечивает взаимосвязь между действиями и механизмами, которые передают данные
- **Экстернализация ресурсов:** предлагает такие функции, как строки и графика
- **Контент-провайдеры:** обмениваются данными между приложениями

72. Как вы устраняете неполадки в приложении, которое часто выходит из строя?

Следующие действия могут помочь диагностировать приложение Android, которое часто выходит из строя:

- **Свободная память:** Поскольку на мобильных устройствах ограничено пространство, вы можете попробовать освободить его, чтобы приложение работало должным образом
- **Проверка совместимости:** Возможно, это не аппаратная проблема, а скорее программная. Не всегда возможно протестировать приложение для всех устройств и операционных систем. Возможно, приложение несовместимо с вашей операционной системой, поэтому проверьте совместимость на странице приложения в Google Play Store.
- **Управление памятью:** Некоторые приложения отлично работают на одном мобильном устройстве, но могут давать сбой на других устройствах. Здесь в игру вступают вычислительная мощность, управление памятью и скорость процессора. Проверьте требования к памяти приложения, если приложение постоянно выходит из строя.
- **Приложение использования данных:** вы можете удалить приложение данных, которая позволит очистить его кэш-памяти и позволяют немного свободного места на вашем устройстве и может повысить производительность приложения

73. Кратко объясните DDMS.

Сервер Dalvik Debug Monitor Server (DDMS) - это инструмент отладки в Android Studio. Он обладает широким спектром функций отладки, таких как:

- Переадресация портов
- Подмена данных о местоположении
- Снимок экрана
- Logcat
- Информация о состоянии радио
- Информация о потоках и кучах

Инструмент DDMS устарел, и Android теперь предлагает пользователям использовать вместо него Android Profiler.

74. Объясните разницу между скрытым и явным намерением.

Ниже приведена разница между двумя намерениями

- **Явное намерение:** явное намерение - это когда вы сообщаете системе о том, какое действие или системный компонент она должна использовать для генерации ответа на это намерение.
- **Неявное намерение:** неявное намерение позволяет вам объявить действие, которое вы хотите выполнить, после чего система Android проверит, какие компоненты зарегистрированы для выполнения этого конкретного действия.

75. Что это за файл AndroidManifest.xml и зачем он вам нужен?

AndroidManifest.xml Файл содержит информацию о приложении, которая затем предоставляется системе Android. Эти данные могут включать название пакета, такие компоненты, как активность, услуги, поставщики контента и многое другое. Этот файл выполняет следующие задачи:

- 75. Что это за файл AndroidManifest.xml и зачем он вам нужен?
- AndroidManifest.xml Файл содержит информацию о приложении, которая затем предоставляется системе Android. Эти данные могут включать название пакета, такие компоненты, как активность, услуги, поставщики контента и многое другое. Этот файл выполняет следующие задачи:
- Объявление Android API, который будет использоваться приложением
- Информация о файле библиотеки, связанная с приложением

76. Объясните различные режимы запуска в Android.

Это различные режимы запуска в Android:

- **Стандарт:** Этот режим запуска генерирует новый экземпляр действия в задаче, из которой оно возникло. Можно создать несколько экземпляров одного и того же действия, которые могут быть добавлены к одним и тем же или разным задачам.
- **singleTop: Этот режим запуска аналогичен стандартному режиму запуска, за исключением того, что если существует предыдущий экземпляр действия на вершине стека, то новый экземпляр не будет создан, но намерение будет отправлено существующему экземпляру действия.**
- **Однозадачность: В этом режиме запуска всегда создается новая задача и добавляется новый экземпляр к задаче в качестве корневого.**
- **singleInstance: Этот режим запуска такой же, как и режим запуска одиночной задачи, но система не запускает никаких новых действий в рамках той же задачи. В сценарии, при котором запускается новое действие, оно запускается в виде отдельной задачи.**

Вопросы для интервью продвинутого уровня

77. Укажите компоненты, необходимые для нового проекта Android

При создании нового проекта Android требуются следующие компоненты:

- **Манифест:** Содержит XML-файл
- **Build /:** содержит выходные данные сборки
- **Res /:** Содержит ресурсы, не связанные с кодом, такие как растровые изображения, строки пользовательского интерфейса и многое другое
- **Src / : Содержит код и ресурсы**

- **Assets /: Содержит файл, который можно преобразовать в файл .apk**

78. Насколько важна настройка разрешений при разработке приложений для Android?

Если код доступен для всех и без ограничений, может возникнуть сценарий, при котором код будет скомпрометирован, что приведет к утечке дефекта. После установки разрешений код становится доступен только авторизованным пользователям.

79. Какие различные типы данных поддерживаются AIDL?

AIDL или язык определения интерфейса Android облегчает общение между клиентом и сервисом. Типы данных, поддерживаемые AIDL, следующие:

- Строка
- Список
- Карта
- Последовательность символов
- INT, Long, Char, Boolean (типы данных Java)

80. Назовите различные варианты хранения данных, доступные на платформе Android.

Платформа Android предоставляет множество вариантов хранения данных, которые могут использоваться в зависимости от потребностей пользователя. Варианты хранения следующие:

- **SharedPreference: Хранит данные в XML-файлах**
- **SQLite: хранит структурированные данные в частной базе данных**
- **Внутреннее хранилище:** хранит данные в файловой системе устройства, где они не могут быть прочитаны другими приложениями
- **Внешнее хранилище:** хранит данные в файловой системе, но к ним можно получить доступ из всех приложений устройства

81. Есть ли какая-либо разница между видами деятельности и услугами?

Да, существует много различий между видами деятельности и услугами. Эти различия указаны в разделе:

Мероприятия	**Услуги**
Закрыто	Открыть
Может быть прекращено в любое время	Не могут быть прекращены в любое в

Не предназначено для работы за кулисами	Разработан для работы за кулисами
Не действует независимо	Действуйте независимо

82. В чем польза макетов на основе XML?

Макеты на основе XML помогают использовать непоследовательный и стандартный формат для настройки формата определения графического интерфейса пользователя. Детали макета помещаются в XML-файлы, а остальные элементы - в исходные файлы.

83. Объясните, какие контейнеры есть в системе Android.

Контейнеры в системе Android помогают удерживать объекты и виджеты вместе, чтобы можно было выполнять определенные элементы и компоновки. Эти контейнеры включают метки, кнопки, поля и т.д.

84. Что такое ADB?

ADB - это Android Debug Bridge, который помогает разработчикам создавать команды удаленной оболочки. Основная функция ADB - разрешать и контролировать процесс передачи данных к порту эмулятора и получать от него ответ.

85. Каковы состояния в деятельности?

В упражнении есть четыре состояния. К ним относятся следующие.

1. **Активное состояние:** активность находится на переднем плане
2. **Приостановленное состояние:** активность выполняется в фоновом режиме и видна
3. **Остановленное состояние:** действие выполняется в фоновом режиме, но не видно, даже скрыто или заслоняет другие действия
4. **Состояние уничтожения:** действие полностью прекращено или уничтожено / удалено

86. Каковы разрешения в процессе разработки приложения?

Разрешения в процессе разработки приложения включают ограничения, налагаемые для защиты данных и кода. Эти разрешения применяются для защиты приложения от взлома, вирусных атак, кражи информации пользователей и возникновения ошибок.

87. Что такое фильтры намерений?

Фильтры намерений являются жизненно важными компонентами системы Android, поскольку они помогают отвечать, фильтровать и получать вводимую информацию.

88. Существуют ли какие-либо критические циклы при мониторинге активности?

Да, при мониторинге активности есть три ключевых цикла. Это:

- **Цикл 1, весь срок службы:** действие происходит между onCreate и onDestroy.
- **Цикл 2, видимый срок службы:** действие происходит между OnStart и onStop
- **Цикл 3, время жизни на переднем плане:** действие происходит между onResume и onPause

89. На каких возможных состояниях основан процесс?

Возможные состояния, на которых основан процесс, включают следующее:

- **Состояние 1:** активность на переднем плане
- **Состояние 2:** Видимая активность
- **Состояние 3:** Фоновая активность
- **Состояние 4:** Пустая активность

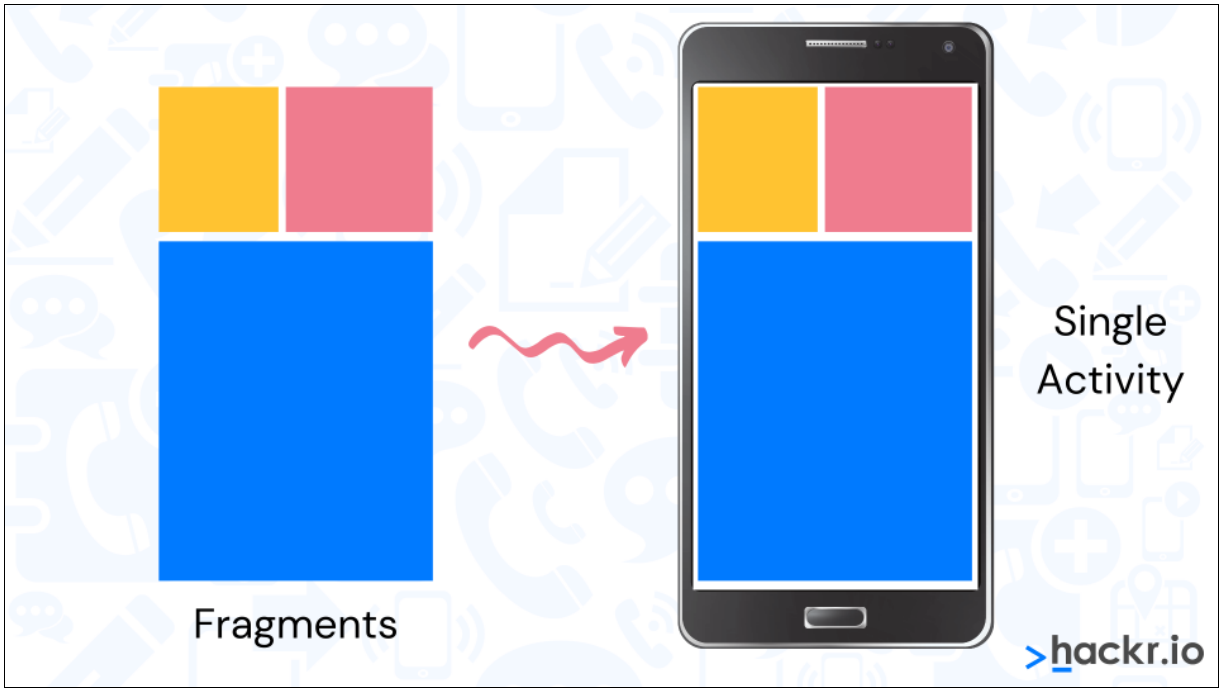
90. Можем ли мы предотвратить ANR в системе Android?

Да, мы можем предотвратить ANR в системе Android. ANR мешает системе Android завершать работу с кодом, который долгое время был адаптивным. Предотвращение может быть осуществлено путем создания дочернего потока, в котором может быть применена большая часть фактической работы кодов, и может быть указан минимальный период времени отсутствия ответов.

91. Какие ресурсы используются по умолчанию? Чем они полезны?

Ресурсы по умолчанию включают строки и файлы по умолчанию. Их отсутствие приведет к появлению ошибок на экране, а также может помешать запуску загруженного приложения. Они полезны, поскольку размещены в виде подкаталогов в каталоге project, который поддерживает запуск загруженного приложения.

92. Что такое фрагмент в Android?



Фрагменты в системе Android являются частью действия, которое в некотором смысле является модульным. Они могут перемещаться и даже объединяться с другими фрагментами, образуя единое действие. Фрагменты можно использовать повторно.

.

93. Что такое мобильное приложение?
- Мобильное приложение - программное обеспечение, предназначенное для работы на мобильных устройствах, таких как смартфоны и планшеты.

94. Какие основные платформы для разработки мобильных приложений существуют?
- Основные платформы для разработки мобильных приложений: iOS (Apple), Android (Google) и Windows Mobile (Microsoft).

95. Что такое нативное приложение?
- Нативное приложение - это приложение, разработанное специально для определенной платформы, используя язык программирования и инструменты, рекомендованные разработчиком платформы.

96. В чем отличие нативного приложения от кросс-платформенного приложения?
- Нативное приложение разрабатывается специально для одной платформы, в то время как кросс-платформенное приложение может работать на нескольких платформах с помощью общего кода.

97. Какие языки программирования используются для разработки мобильных приложений под iOS?

- Для разработки мобильных приложений под iOS используются языки программирования Objective-C и Swift.

98. Какие языки программирования используются для разработки мобильных приложений под Android?
- Для разработки мобильных приложений под Android используется язык программирования Java и Kotlin.

99. Какие инструменты разработки мобильных приложений под iOS вы знаете?
- Некоторые инструменты разработки мобильных приложений под iOS: Xcode, Swift, Objective-C, SwiftUI.

100. Какие инструменты разработки мобильных приложений под Android вы знаете?
- Некоторые инструменты разработки мобильных приложений под Android: Android Studio, Java, Kotlin.

101. Что такое гибридное приложение?
- Гибридное приложение - это комбинация веб-технологий (HTML, CSS, JavaScript) и нативного кода, которая позволяет запускать приложение на разных платформах.

102. Какие фреймворки используются для разработки гибридных приложений?
- Некоторые фреймворки для разработки гибридных приложений: React Native, Flutter, Ionic, PhoneGap.

103. Что такое API?
- API (Application Programming Interface) - это набор правил и инструкций, которые определяют, как различные компоненты программного обеспечения взаимодействуют друг с другом.

104. Какие типы API используются в разработке мобильных приложений?
- В разработке мобильных приложений используются различные типы API, такие как веб-сервисы API, социальные API (например, API Facebook или Twitter), API карт и геолокации и т. д.

105. Что такое SDK?
- SDK (Software Development Kit) - это набор инструментов и ресурсов для разработки приложений под определенную платформу или для использования определенных сервисов.

106. Какие SDK используются в разработке мобильных приложений под iOS?
- Некоторые SDK, используемые в разработке мобильных приложений под iOS: iOS SDK, Facebook SDK, Google Maps SDK.

107. Какие SDK используются в разработке мобильных приложений под Android?
- Некоторые SDK, используемые в разработке
мобильных приложений под Android: Android SDK, Facebook SDK, Google Maps SDK.

108. Что такое UI/UX дизайн в мобильной разработке?
- UI (User Interface) - это то, как пользователь взаимодействует с приложением, а UX (User Experience) - это общее впечатление и удовлетворенность пользователя от использования приложения.

109. Какие инструменты используются для создания дизайна мобильных приложений?
- Некоторые инструменты для создания дизайна мобильных приложений: Sketch, Adobe XD, Figma.

110. Что такое MVP в контексте мобильной разработки?
- MVP (Minimum Viable Product) - это минимально жизнеспособная версия приложения, содержащая основные функции и возможности, необходимые для запуска и тестирования.

111. Что такое адаптивный дизайн?
- Адаптивный дизайн - это подход к разработке, при котором мобильное приложение автоматически адаптируется к разным размерам экранов и устройствам.

112. Какие методологии разработки используются в мобильной разработке?
- Некоторые методологии разработки, используемые в мобильной разработке: Agile, Scrum, Waterfall.

113. Что такое тестирование мобильных приложений?
- Тестирование мобильных приложений - это процесс проверки функциональности, производительности и качества приложения перед его выпуском.

114. Какие виды тестирования мобильных приложений существуют?
- Некоторые виды тестирования мобильных приложений: функциональное тестирование, тестирование совместимости, тестирование производительности, тестирование пользовательского интерфейса и другие.

115. Что такое отладка (debugging) в мобильной разработке?
- Отладка (debugging) - это процесс идентификации, анализа и исправления ошибок в коде приложения для обеспечения его правильной работы.

116. Какие инструменты используются для отладки мобильных приложений?
- Некоторые инструменты для отладки мобильных приложений: Android Studio Debugger, Xcode Debugger, Chrome Developer Tools.

117. Что такое хранение данных в мобильных приложениях?
- Хранение данных в мобильных приложениях - это процесс сохранения и получения информации, необходимой для работы приложения, на устройстве пользователя или удаленном сервере.

118. Какие методы хранения данных используются в мобильных приложениях?

- Некоторые методы хранения данных в мобильных приложениях: использование баз данных (например, SQLite), файловой системы устройства, удаленного сервера.

119. Что такое облачные сервисы в мобильной разработке?
- Облачные сервисы - это сервисы и инфраструктура, предоставляемые в облаке (например, Amazon Web Services, Google Cloud), которые могут использоваться разработчиками для хранения данных, обработки или других задач.

120. Как обеспечить безопасность мобильных приложений?
- Некоторые методы обеспечения безопасности мобильных приложений: шифрование данных, аутентификация пользователей, обновления приложений, тестирование на уязвимости.

121. Что такое пуш-уведомления в мобильных приложениях?
- Пуш-уведомления - это сообщения, отправляемые разработчиками приложений на устройства пользователей для уведомления о важных событиях или обновлениях.

122. Какие инструменты используются для отправки пуш-уведомлений?
- Некоторые инструменты для отправки пуш-уведомлений: Firebase Cloud Messaging (FCM), Apple Push Notification Service (APNS).

123. Что такое монетизация мобильных приложений?
- Монетизация мобильных приложений - это процесс генерации дохода от приложения, например, через платные загрузки, рекламу, покупки внутри приложения.

124. Какие методы монетизации мобильных приложений существуют?
- Некоторые методы монетизации мобильных приложений: платные загрузки, реклама (например, баннеры, видеоролики), покупки внутри приложения (например, дополнительные функции или контент).

125. Что такое аналитика мобильных приложений?
- Аналитика мобильных приложений - это сбор, анализ и интерпретация данных о поведении пользователей в приложении с целью оптимизации его функциональности и использования.

126. Какие инструменты используются для аналитики мобильных приложений?
- Некоторые инструменты для аналитики мобильных приложений: Google Analytics, Firebase Analytics, Flurry Analytics.

127. Что такое версионирование приложений?
- Версионирование приложений - это процесс присвоения уникального номера или имени каждой новой версии приложения для идентификации и управления обновлениями.

128. Какие подходы к версионированию приложений существуют?
- Некоторые подходы к версионированию приложений: семантическое версионирование (например, 1.0.0), версионирование по дате (например, 2022.09.01), кодовые имена версий (например, "Малиновый мармелад").

129. Что такое приложения с открытым исходным кодом (Open-source)?
- Приложения с открытым исходным кодом - это приложения, код которых доступен для просмотра, использования и изменения общественности.

130. Какие преимущества и недостатки приложений с открытым исходным кодом?
- Преимущества приложений с открытым исходным кодом: прозрачность, возможность быстрой адаптации и улучшения, широкое сообщество разработчиков.
- Недостатки приложений с открытым исходным кодом: меньшая защищенность, возможность злоумышленников использовать уязвимости.

131. Какие требования нужно учитывать при разработке мобильного приложения?
- Некоторые требования, которые нужно учитывать при разработке мобильного приложения: функциональные требования, требования к производительности, требования к безопасности, требования к пользовательскому интерфейсу.

132. Какие этапы включает жизненный цикл разработки мобильного приложения?
- Жизненный цикл разработки мобильного приложения включает следующие этапы: планирование, анализ и проектирование, разработка, тестирование, развертывание и поддержка.

133. Что такое альфа-тестирование мобильного приложения?
- Альфа-тестирование мобильного приложения - это первый этап тестирования, когда приложение проверяется внутренней командой разработчиков перед передачей его на более широкое тестирование
134. Что такое бета-тестирование мобильного приложения?
- Бета-тестирование мобильного приложения - это этап тестирования, когда приложение предоставляется ограниченной группе пользователей для оценки и предоставления обратной связи.

135. Что такое релиз мобильного приложения?
- Релиз мобильного приложения - это процесс выпуска приложения на основные платформы (например, App Store, Google Play) для скачивания и использования пользователями.

136. Какие методы распространения мобильных приложений существуют?
- Некоторые методы распространения мобильных приложений: публикация в официальных магазинах приложений (например, App Store, Google Play), скачивание из веб-сайта или по прямой ссылке, предустановка на устройства.

137. Что такое пользовательский интерфейс (UI) в мобильных приложениях?
- Пользовательский интерфейс (UI) - это визуальная часть мобильного приложения, через которую пользователи взаимодействуют с приложением (кнопки, меню, разметка и т. д.).

138. Что такое пользовательский опыт (UX) в мобильных приложениях?

- Пользовательский опыт (UX) - это общее впечатление и удовлетворенность пользователей от использования мобильного приложения, включая удобство, эффективность и удовлетворение целям пользователя.

139. Что такое аутентификация пользователей в мобильных приложениях?
- Аутентификация пользователей - это процесс проверки подлинности пользователей, чтобы обеспечить доступ к определенным функциям или данным в мобильном приложении.

140. Какие методы аутентификации пользователей используются в мобильных приложениях?
- Некоторые методы аутентификации пользователей в мобильных приложениях: парольная аутентификация, аутентификация по отпечатку пальца, аутентификация с помощью социальных сетей (например, Facebook, Google).

141. Что такое мобильный интерфейс программирования приложений (API)?
- Мобильный интерфейс программирования приложений (API) - это набор инструкций и функций, предоставляемых разработчиками приложений для взаимодействия с определенной платформой или сервисом.

142. Какие меры безопасности следует применять при разработке мобильных приложений?
- Некоторые меры безопасности, которые следует применять при разработке мобильных приложений, включают шифрование данных, аутентификацию пользователя, проверку входных данных и защиту от вредоносных атак.

143. Что такое монетизация мобильных приложений?
- Монетизация мобильных приложений - это процесс генерации дохода от приложения, например, через платные загрузки, рекламу, покупки в приложении или подписки.

144. Какие способы тестирования мобильных приложений существуют?
- Некоторые способы тестирования мобильных приложений включают модульное тестирование, функциональное тестирование, тестирование пользовательского интерфейса, нагрузочное тестирование и тестирование совместимости.

145. Какие основные принципы следует учитывать при проектировании интерфейса мобильного приложения?
- Некоторые основные принципы проектирования интерфейса мобильного приложения включают простоту, консистентность, интуитивность, доступность и эффективность.

146. Какие требования к производительности следует учитывать при разработке мобильного приложения?
- Некоторые требования к производительности, которые следует учитывать при разработке мобильного приложения, включают быструю загрузку, отзывчивость интерфейса и эффективное использование ресурсов устройства.

147. Какие основные проблемы могут возникнуть при разработке мобильного приложения?

- Некоторые основные проблемы, которые могут возникнуть при разработке мобильного приложения, включают несовместимость с разными устройствами, проблемы с производительностью, ошибки программирования и проблемы безопасности.

148. Что такое MVP (Minimum Viable Product) в контексте разработки мобильных приложений?
- MVP (Minimum Viable Product) - это минимально необходимый набор функций и возможностей, которые позволяют запустить и протестировать мобильное приложение с целью получения обратной связи от пользователей.

149. Какие факторы следует учитывать при выборе платформы для разработки мобильного приложения?
- Некоторые факторы, которые следует учитывать при выборе платформы для разработки мобильного приложения, включают целевую аудиторию, бюджет, требования к функциональности и наличие ресурсов и опыта в разработке для конкретной платформы.

150. Какие виды хранилищ данных можно использовать в мобильных приложениях?
- В мобильных приложениях можно использовать различные виды хранилищ данных, включая локальные базы данных, кэширование, удаленные серверы и облака.

4 Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Контроль результатов обучения обучающимися, этапов и уровня формирования компетенций по дисциплине осуществляется через проведение входного, текущего, рубежных, выходного контролей и контроля самостоятельной работы.

Процедура проведения экзамена приведена в Положении о текущем контроле успеваемости и промежуточной аттестации.

Использование модульно-рейтинговой системы обучения и оценки успеваемости обучающихся для оценивания знаний, умений, навыков и (или) опыта деятельности остается на усмотрение преподавателя.