Департамент образования и науки города Москвы

ГАПОУ Колледж предпринимательства №11

Центр информационно-коммуникационных технологий

**Аналитический отчёт**

по дисциплине: “Исследовательская и проектная деятельность”

на тему: Видео руководство: «Подготовка к демонстрационному экзамену по стандартам Ворлдскиллс по направлению: мобильная разработка»

|  |  |
| --- | --- |
| Выполнили:  студенты группы ИСиП-32  Собко София – менеджер  Адногулов Артем – зам. менеджера  Шевцов Владимир | Проверил:  Преподаватель  Е.Ю. Ильина |

Москва, 2022 г.

**Содержание**

Введение3

Глава 1. Натуральное описание 6

Глава 2. Сбор и анализ информации 8

Выводы

Рекомендации

Заключение

**Введение**

Демонстрационный экзамен по стандартам Ворлдскиллс – это форма государственной итоговой аттестации выпускников по программам среднего профессионального образования образовательных организаций высшего и среднего профессионального образования.

Демонстрационный экзамен по стандартам Ворлдскиллс проводится с целью определения у студентов и выпускников уровня знаний, умений, навыков, позволяющих вести профессиональную деятельность в определенной сфере и (или) выполнять работу по конкретным профессии или специальности в соответствии со стандартами Ворлдскиллс Россия.

Демонстрационный экзамен у групп «Информационные системы и программирование» – (ИСИП) представляет из себя создание мобильного приложения согласно Техническому заданию.

**Проект разработки видео руководства является актуальным** для ГАПОУ КП №11 Центра ИКТ благодаря тому, что группа имеет цель и желание создать видео руководство, которое поможет студентам групп ИСИП подготовиться к сдаче демонстрационного экзамена, что является заключительным этапом в обучении перед получением диплома.

Данный проект способствует развитию навыков проектной команды в сфере мобильной разработки.

**Целью проекта** является создание видео руководства по демонстрационному экзамену для ГАПОУ КП №11 Центра ИКТ путем его создания с помощью Xcodе (среда разработки мобильных приложений под IOS) и CapCut (видеоредактора), которое может заинтересовать студентов и сделать их процесс подготовки легче с подробным разбором реального задания прошлого года. Руководство в дальнейшем будет использоваться в учебном заведении для самостоятельной подготовки к демонстрационному экзамену.

Данный проект способствует удобному и понятному применению современных технологий прямо во время обучения. Это упрощает задачу для студентов, успешно сдать демонстрационный экзамен.

**Задачи**

1. Выбрать тему проекта.
2. Собрать команду.
3. Распределить роли в команде.
4. Выбрать источники информации.
5. Найти информацию о существующих похожих видео руководствах.
6. Найти информацию о возможностях среды разработки Xcode.
7. Проанализировать информацию, необходимую для создания приложения.
8. Подробно изучить все необходимые библиотеки среды разработки Xcode.
9. Взять у преподавателя по мобильной разработке полное задание с демонстрационного экзамена прошлого года.
10. Разбить создание мобильного приложения на блоки(уроки).
11. Найти теоретическую информацию для каждого урока.
12. Тезисно расписать, о чем будет говориться на каждом видео.
13. Найти картинки для теоретической части видео, для наглядности.
14. Записать голос с разъяснением теории.
15. Записать экран и голос для практической части.
16. Смонтировать уроки.
17. Утвердить руководство по подготовке к демонстрационному экзамену у преподавателя, который ответственный за его проведение.
18. Исправить ошибки, если таковые будут выявлены.
19. Защитить аналитический отчет.

**Объект исследования** – демонстрационный экзамен по стандартам Ворлдскиллс по направлению: мобильная разработка.

**Предмет** **исследования** – процесс подготовки студентов к демонстрационному экзамену по стандартам Ворлдскиллс.

**Субъект исследования** – проектная команда.

**Выбранные методы исследования** – моделирование, описательный,

кабинетные методы исследования.

**Глава 1. Натуральное описание**

**05.09.2022-.19.09.2022**

Преподаватель по проектной деятельности назвал условия выполнения проекта. Группа успешно с ними ознакомилась и приступила к написанию аналитического отчета.

Сформирована команда из трех человек.

После утверждения темы проекта в команду были отобраны:

Собко София, Манасбек уулу Эрболот, Владимир Шевцов.

Были распределены следующие роли:

Манасбек уулу Эрболот – Подробный разбор задач по каждой теме.

Владимир Шевцов – Поиск и отбор информации.

Манасбек уулу Эрболот – Форматирование электронного документа.

Владимир Шевцов – Ведение аналитического отчёта за периоды.

Собко София предоставила тему проекта, которую одобрила команда. Группа решила его осуществить.

Манасбек уулу Эрболот оформил титульный лист.

Собко София описала актуальность и цель данного проекта.

Собко София написала задачи.

**19.09.2022-03.010.2022**

Владимир Шевцов – оформление аналитического отчета.

Манасбек уулу Эрболот – нашел информацию о существующих похожих видео руководствах на YouTub, нашел информацию о возможностях среды разработки Xcode.

Собко София – подробно изучила все необходимые библиотеки среды разработки Xcode, связалась с преподавателем по мобильной разработке и взяла полное задание с демонстрационного экзамена прошлого года.

**03.10.2022-17.10.2022**

Манасбек уулу Эрболот покинул проектную группу.

Адногулов Артем присоединился к группе.

При проверке работы были сделаны следующие замечания:

1. Добавить после каждого текста ссылку на источник.

2. Перенести информацию из Введения в Главу 2.

3. Правильно сформулировать предложение «Целью проекта является…» без слова «сдача».

Владимир Шевцов – монтаж первого урока.

Адногулов Артем – правки аналитического отчета согласно замечаниям, оформление аналитического отчета.

Собко София – запись первого урока, распределение тем уроков.

**17.10.2022-31.10.2022**

При проверке работы были сделаны следующие замечания:

1. Прикрепить скриншоты технического задания в “Приложение 1”

2. Неверная отчетность в периодах.

3. Исправить объект исследования

4. Выравнивание маркеров

5. Выбрать готовый вариант оглавления

Владимир Шевцов – правки аналитического отчета согласно замечаниям, оформление аналитического отчета.

Адногулов Артем – монтаж второго урока, поиск информации по темам.

Собко София – запись второго видео урока.

**31.10.2022-14.11.2022**

Владимир Шевцов – оформление аналитического отчета, поиск информации по темам, монтаж урока 3.

Адногулов Артем – закончен монтаж урока 2, шаблон превью.

Собко София – запись третьего видео урока.

**Глава 2. Сбор и анализ информации**

Демонстрационный экзамен по стандартам Ворлдскиллс – это форма государственной итоговой аттестации выпускников по программам среднего профессионального образования образовательных организаций высшего и среднего профессионального образования, которая предусматривает:(Здесь все правильно)

1. Моделирование реальных производственных условий для демонстрации выпускниками профессиональных умений и навыков;

2. Независимую экспертную оценку выполнения заданий демонстрационного экзамена, в том числе экспертами из числа представителей предприятий;

3. Определение уровня знаний, умений и навыков выпускников в соответствии с международными требованиями.

**Существующие каналы с видео курсами по разработке мобильных приложений под IOS:**

<https://www.youtube.com/channel/UCsuV4MRk_aB291SrchUVb4w>

<https://www.youtube.com/c/AmrCodes>

<https://youtube.com/c/IanSolomein>

<https://youtube.com/channel/UCiyiEAeWUuuPj6tt_jePALQ>

С данный каналов мы можем брать пример объяснения теории в начале видео и пояснения процесса при написании самого приложения.

**Xcode**

**Возможности среды разработки Xcode:**

XCode — среда для разработки программного обеспечения под операционные системы Apple. С ее помощью можно писать и проверять код, запускать программы, описывать графические интерфейсы и их связь с кодом.

**Применение Xcode:**

* Быстрое и удобное написание кода с помощью специального текстового

редактора с подсветкой и проверкой синтаксиса, подсчетом строк и другими функциями.

* Создание и организация крупных проектов, которые могут включать в себя много файлов и папок. Среда помогает структурировать работу.
* Компиляция или интерпретация кода, то есть его преобразование в понятной машине формат.
* Выполнение кода. С помощью IDE можно запускать программу и

отслеживать результаты ее работы.

* Отладка и тестирование — работа по поиску и устранению ошибок в коде.
* Создание графических интерфейсов и их связь с кодом.
* Автоматизация сборки готового проекта.
* Упрощение архитектурных и конструкторских задач: их помогают

решать вспомогательные инструменты, которые наглядно показывают структуру кода или проекта.

XCode — среда для разработки программного обеспечения под операционные системы Apple. С ее помощью можно писать и проверять код, запускать программы, описывать графические интерфейсы и их связь с кодом.

**Из чего состоит среда Xcode:**

Инструменты, которые есть в этой IDE, подобраны так, чтобы разработчику было легко пройти полный цикл создания приложения, в первую очередь под платформы Apple.

**Редактор кода.** Он похож на текстовый редактор, но отличается более широкой функциональностью. Редактор кода распознает синтаксис языка, подсвечивает его конструкции для большей читаемости. Он может сигнализировать об ошибках в синтаксисе или давать краткую справку о той или иной команде. Он подсчитывает строки, имеет гибкие возможности для навигации по коду и выполняет множество других функций.

В редакторе программист пишет код, а потом сохраняет его в виде файла в нужном формате. XCode позволяет его открывать и редактировать.

**Редактор графических интерфейсов.** В XCode по умолчанию входит приложение Interface Builder — редактор интерфейсов. С его помощью можно быстро создавать GUI для своих программ. Редактор содержит наборы — палитры стандартных элементов интерфейса: их можно создавать, настраивать и редактировать.

Созданный интерфейс можно связать с кодом и таким образом получить полностью функциональное приложение.

**Поддержка языков программирования.** Программирование в XCode во многом адаптировано под Swift и Objective-C — языки, которые активнее всего используются в экосистеме Apple. Также для комфортного использования IDE поддерживаются другие языки программирования.

**Компиляторы.** Компилятор — это программа, которая переводит написанную программу в машинные коды для ее запуска Также существуют интерпретаторы — они, в отличие от компиляторов, исполняют код построчно, а не преобразуют его полностью.

В IDE должны быть компиляторы для языков, которые она поддерживает: без них программу нельзя запустить и протестировать в среде. XCode пользуется преобразованным GCC, или GNU Compiler Collection: это свободная коллекция компиляторов для популярных языков программирования.

**Отладчики.** Отладчик — составная часть любой масштабной IDE, программа для поиска и отладки ошибок. С его помощью можно:

* трассировать код, то есть выполнять его построчно;
* ставить метки или точки останова в определенных местах

программы;

* следить за состоянием сущностей в разные моменты;
* настраивать логику для трассировки, остановки и других действий.

Так программисту легче отследить, в какой момент времени в коде происходит ошибка и с чем она связана. Он сможет быстрее понять, как ее исправить.

**API.** API помогает связать между собой разные сервисы и программные продукты. В IDE должны быть API для разных направлений разработки. В случае с XCode это в первую очередь API, которые нужны для программирования под Apple: Cocoa, Carbon и другие. Например, с их помощью разработчик связывает между собой интерфейс и код.

**Фреймворки и библиотеки.** Это дополнительные средства разработки, которые помогают расширить язык и добавляют новые возможности: функциональные и архитектурные.

В XCode входят фреймворки для создания серверного, десктопного и мобильного ПО и многого другого.

**Документация.** В XCode по умолчанию входит большая часть документации Apple, которая посвящена среде разработки и ее инструментам. При изучении с нуля с документацией придется сверяться часто, но она должна быть под рукой не только у новичка.

**Контроль версий.** Системы контроля версий позволяют нескольким программистам работать одновременно и вносить в код разные изменения, а потом объединять их. С системами можно откатываться к предыдущим состояниям или иметь сразу несколько «веток» разработки. Наиболее популярная — Git. XCode поддерживает Git «из коробки»: создавать «точки сохранения», откатываться к ним или сливать версии можно прямо из интерфейса IDE.

**Симулятор.** С помощью XCode можно «симулировать» запуск программы на мобильном устройстве Apple. Это помогает понять, как приложение будет работать на реальном телефоне или планшете, и качественнее его протестировать.

**Swift Playgrounds.** Так называется среда для обучения языку Swift и разработки на нем. Имеет характерный интерфейс, поддержку фреймворка для графических интерфейсов SwiftUI и возможность мгновенного отображения результата кода. В Swift Playgrounds входят интерактивные уроки для быстрого обучения концепциям языка, также среда подходит для экспериментов и небольших проектов.

Источник: <https://blog.skillfactory.ru/glossary/xcode/>

**Библиотеки (Pods)**

**Библиотеки необходимые для создания приложения:**

**Alamofire**

Alamofire - это HTTP сетевая библиотека на Swift для iOS и Mac OS X. Она обеспечивает элегантный интерфейс сетевого стека Foundation от Apple, упрощающего ряд общих сетевых задач.

Alamofire предоставляет цепочку методов ответ/запрос, параметр JSON и ответ сериализации, аутентификацию и многие другие функции. В этом туториале по Alamofire вы будете использовать Alamofire для выполнения основных сетевых задач, таких как загрузка файлов и запрос данных со сторонних RESTful API.

Источник: <https://swiftbook.ru/post/tutorials/tutorial-po-alamofire/>

**SwiftyJSON**

SwiftyJSON - это библиотека классов, написанная swift для анализа данных json. SwiftyJSON чрезвычайно удобен в использовании, и нет необходимости беспокоиться о массивах вне границ и других проблемах.

Источник: <https://russianblogs.com/article/9351659114/>

**SDWebImageSwiftUI**

SDWebImageSwiftUI - это платформа загрузки изображений SwiftUI, основанная на SDWebImage.

Он содержит все ваши любимые функции SDWebImage, такие как асинхронная загрузка изображений, кэширование памяти/диска, анимированное воспроизведение изображений и производительность.

**Зачем использовать SDWebImageSwiftUI:**

* Поддерживает управление параметрами заполнителя и

детализации для загрузки изображения в виде SDWebImage

* Поддерживает прогрессивную загрузку изображений (например,

базовую линию)

* Поддерживает событие успеха/сбоя/завершенного изменения для

пользовательской обработки

* Поддерживает индикатор с индикатором активности/прогресса и

настройкой

* Поддерживает встроенную анимацию и переход на базе SwiftUI
* Поддерживает анимированное изображение

Источник: <https://github.com/SDWebImage/SDWebImageSwiftUI>

**RealmSwift**

Realm - это мобильная база данных, которая работает непосредственно внутри телефонов, планшетов или носимых устройств.

**Зачем использовать Realm:**

* Интуитивно понятная для разработчиков: объектно-ориентированная модель данных Realm проста в освоении, не нуждается в ORM и позволяет писать меньше кода.
* Предназначен для автономного использования: Локальная база данных Realm сохраняет данные на диске, поэтому приложения работают так же хорошо, как и в Интернете.
* Создан для мобильных устройств: Realm полностью функционален, легок и эффективно использует память, дисковое пространство и время автономной работы.

Источники: <https://github.com/realm/realm-swift>

<https://swiftbook.ru/post/tutorials/tutorial-po-freymvorku-realm/>

**Задание с прошлого года**

Техническое задание доступно по ссылке:

<https://docs.google.com/document/d/1zjk8zbWIEWjupgy18RgVG_rcw_YH8k_6/edit>

Макет доступен по ссылке:

<https://www.figma.com/file/yLAIsuVrxWDIWgCXzdyD0j/KOD1.4-Variant1?node-id=0%3A1>

Описание протокола API доступно по ссылке:

<https://app.swaggerhub.com/apis-docs/WorldSkills-MAD/WorldCinema/1.0.0>

**Темы уроков**

* + - 1. Интро
      2. Переменные, базовые типы, опциональные типы, массивы, словари
      3. Switch, if else , for, break
      4. Функции, перечисления
      5. Базовые элементы
      6. Расположение элементов
      7. Библиотеки( SDWebImageSwiftUI, Realm)
      8. Библиотеки( Alamofire, SwiftyJSON), Swagger
      9. Postman, Figma
      10. Git

**Создание видеоуроков**

**Урок 1 – Интро**

**План урока:**

* 1. **Вступительное слово:**

Всем привет! В этом видео-курсе мы с вами изучим необходимый стэк технологий для успешной сдачи демонстрационного экзамена. Первые 10 уроков мы будем разбирать основы SwiftUI, научимся работать в Xcode, Postman, Swagger и Figma. В последующих видео будем применять полученные знания на практике в виде выполнения технического задания демонстрационного экзамена прошлого года.

Начнем с изучения среды разработки Xcode.

XCode — среда для разработки программного обеспечения под операционные системы Apple. С ее помощью можно писать и проверять код, запускать программы, описывать графические интерфейсы и их связь с кодом.

* 1. **Разбор среды разработки Xcode**
* Создание нового проекта
* Ветки, репозитории

Системы контроля версий позволяют нескольким программистам работать одновременно и вносить в код разные изменения, а потом объединять их. С системами можно откатываться к предыдущим состояниям или иметь сразу несколько «веток» разработки.

Наиболее популярная — Git. XCode поддерживает Git «из коробки»: создавать «точки сохранения», откатываться к ним или сливать версии можно прямо из интерфейса IDE.

* Настройки приложения
* Библиотеки

Это дополнительные средства разработки, которые помогают расширить язык и добавляют новые возможности: функциональные и архитектурные.

* Симуляторы (FaceID, клавиатура)

С помощью XCode можно «симулировать» запуск программы на мобильном устройстве Apple.

Это помогает понять, как приложение будет работать на реальном телефоне или планшете, и качественнее его протестировать.

* Превью

Когда вы создаете пользовательское представление с помощью SwiftUI, Xcode может отображать предварительный просмотр содержимого представления, которое остается актуальным по мере внесения изменений в код представления.

Источник: <https://developer.apple.com/documentation/swiftui/previews-in-xcode>

* UI и Unit тесты
* Assets

Здесь вы будете хранить все необходимые вам ресурсы, такие как изображения, цвета, значки приложений, текстуры, стикеры и данные.

* LaunchScreen

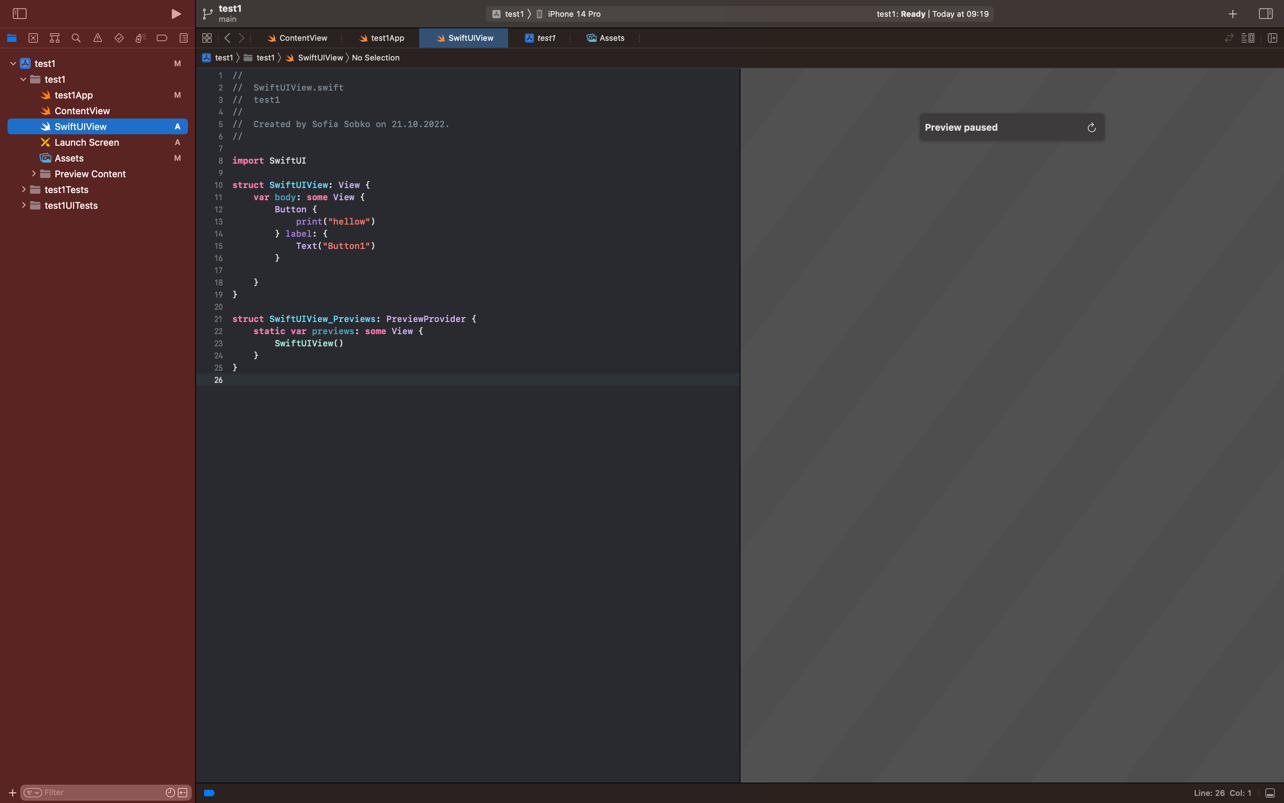
Каждое приложение для iOS должно содержать экран запуска, который отображается во время запуска вашего приложения.

Экран запуска появляется мгновенно при запуске вашего приложения и быстро заменяется первым экраном приложения.

Источник: <https://developer.apple.com/documentation/xcode/specifying-your-apps-launch-screen>

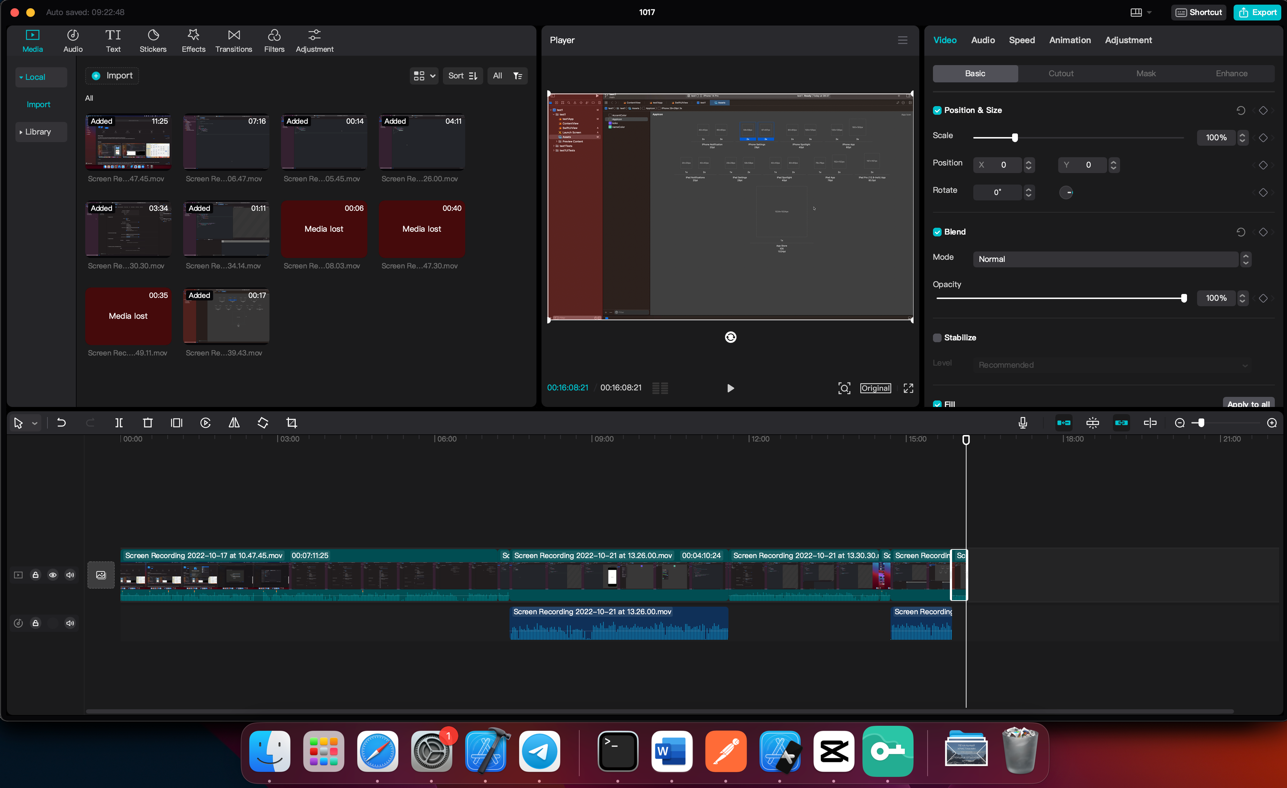
* Debug (консоль, breakpoints)
* Создание новых файлов

**Проект в Xcode:**



**Рис. 1.** **Intro – в Xcode**

**Монтаж:**

****

**Рис. 2. Intro – монтаж**

**Урок в YouTube:**

****

**Рис. 3. Intro – Урок в YouTube**

**Урок 2 –** **Переменные, базовые типы, опциональные типы, массивы, словари**

**План урока:**

* 1. **Вступительное слово:**

Всем привет! В этом видео-уроке мы с вами изучим такие темы как: переменные, базовые типы, опциональные типы, массивы, словари.

Начнем с темы «Переменные»

* 1. **Разбор тем:**
* **Переменные**

Программа на Swift обладает двумя качествами: она может хранить некоторые данные и может выполнять действия. Для хранения данных в Swift, как и в других языках программирования, используются переменные. Переменная представляет именованный участок в памяти, в котором хранится некоторое значение.

Переменные имеют имя и значение. Для определения переменной используется ключевое слово var.

Отличительной особенностью переменных является то, что мы можем изменять их значение многократно во время работы программы.

Кроме переменных для хранения данных в программе могут использоваться константы. Константы подобны переменным, они также хранят некоторое значение, за тем исключением, что определяются с помощью ключевого слова let, и мы не можем после их инициализации изменить их значение.

Таким образом, если значение некоторой переменной в течении программы меняться не будет, то вместо этой переменной лучше использовать константу.

Мы можем определить сразу несколько переменных и констант на одой строке. В этом случае они должны разделяться запятой.

**Правила именования**

Переменные и константы должны иметь уникальные имена. Нельзя использовать в программе несколько переменных и (или) констант с одними и теми же именами.

Причем хорошей практикой является использование названий в так называемом "верблюжем регистре" или CamelCase. То есть названия начинаются с маленькой буквы. Если название состоит из нескольких слов, то только первое из них начинается с маленькой буквы.

Источник: <https://metanit.com/swift/tutorial/2.1.php>

* **Базовые типы**

При написании кода мы работаем с разными типами данных Swift. Базовые типы, которые используются чаще всего:

String (текстовая строка);

Int (целое число);

Float (32-битное число с плавающей точкой);

Double (64-битное число с плавающей точкой для более длинных переменных);

Bool (логическое значение «true» или «false»).

Swift обрабатывает их всех по отдельности. Посмотрим на особенности типов данных Swift.

Источник: <https://stfalcon.com/ru/blog/post/swift-basics-data-types>

* **Опциональные типы**

Опциональные типы представляют объекты, которые могут иметь, а могут и не иметь значение. Опциональные типы выступают двойниками базовых типов. Все они имеют в конце вопросительный знак: Int?, String? и т.д. Вопросительный знак как раз указывает, что это опциональный тип.

Если объект не имеет значения, то ему присваивается специальное значение nil.

чтобы полноценно работать с объектами опциональных типов, следует извлечь из них значение. Для извлечения значения используется оператор ! - восклицательный знак после названия объекта опционального типа. Данный оператор еще называют unwrap operator или forced unwrap operator.

Источник: <https://metanit.com/swift/tutorial/2.8.php>

* **Массивы**

Массив представляет упорядоченную коллекцию элементов одноготипа, к которым можно обратиться по индексу - номеру элемента в массиве. Индекс представляет число - объект типа Int и начинается с нуля. По сути массив представляет собой обычную переменную или константу, которая хранит несколько объектов наподобие кортежа.

Рассмотреть: Объявление массива, Доступ к элементам массива, Размер массива, Перебор массива, Сравнение массивов, Копирование массива, Добавление в массив, Удаление из массива, Сортировка массива, Фильтрация массивов

Источник: <https://metanit.com/swift/tutorial/2.9.php>

* **Словари**

Словарь (dictionary) представляет собой хранилище, в котором каждый элемент имеет ключ и значение. Разные элементы не могут иметь одинаковые ключи. Все ключи в словаре должны быть уникальными. По ключу мы можем найти элемент в словаре изменить его или удалить. Чтобы определить словарь, достаточно просто написать в квадратных скобках несколько элементов в формате [Key: Value].

Источник: <https://metanit.com/swift/tutorial/2.12.php>

**Проект в Xcode:**

Text

Description automatically generated

**Рис. 4. Урок 2 – в Xcode**

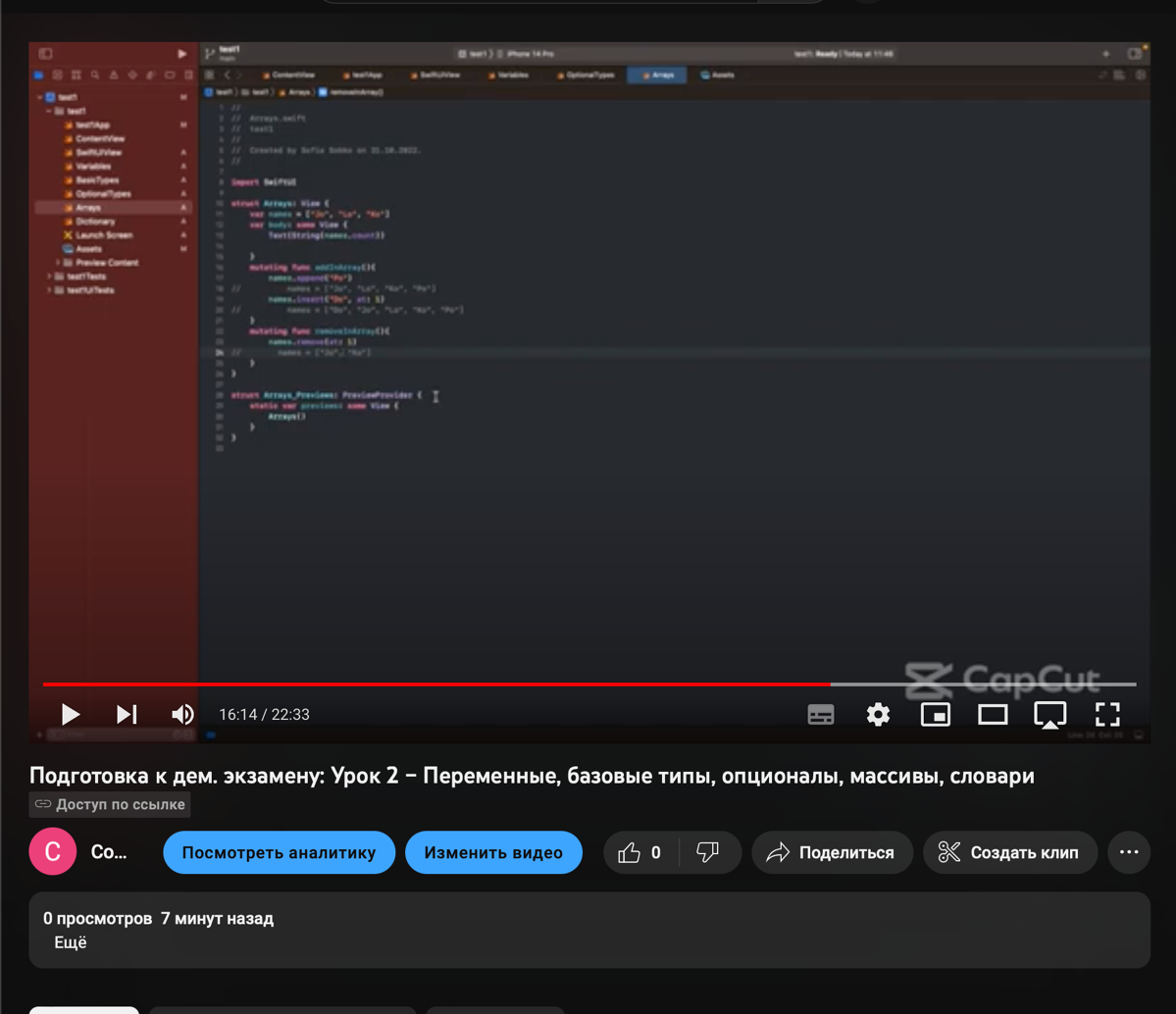
**Монтаж:**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Рис. 5. Урок 2 – монтаж**

**Урок в YouTube:**

****

**Рис. 6. Урок 2 – в YouTube**

**Урок 3 – Switch, i****f else , for, break**

**План урока:**

* 1. **Вступительное слово:**

Всем привет! В этом видео-уроке мы с вами изучим такие темы как: Switch, if else , for, break.

Начнем с темы «If else»

* 1. **Разбор тем:**
* If else

Конструкция if проверяет истинность некоторого условия и в зависимости от результатов проверки выполняет определенный код.

Важно, что после слова if в этой конструкции должно идти значение типа Bool.

Если при проверке условия нам надо выполнить какие-либо альтернативные действия, то мы можем использовать блок else.

Но при сравнении чисел мы можем насчитать три состояния: первое число больше второго, первое число меньше второго и числа равны. Используя конструкцию else if, мы можем обрабатывать дополнительные условия.

* Тернарный оператор

Тернарный оператор аналогичен простой конструкции if и имеет следующий синтаксис:

[первый операнд - условие] ? [второй операнд] : [третий операнд]

Источник: <https://metanit.com/swift/tutorial/2.6.php>

* Switch

Конструкция switch/case похожа на конструкцию if/else, так как позволяет обработать сразу несколько условий

После ключевого слова switch идет сравниваемое выражение. Это может быть переменная или константа. Значение этого выражения последовательно сравнивается со значениями, помещенными после оператора сase. И если совпадение будет найдено, то будет выполняться определенный блок сase.

Если совпадение не будет найдено, то выполняется оператор default.

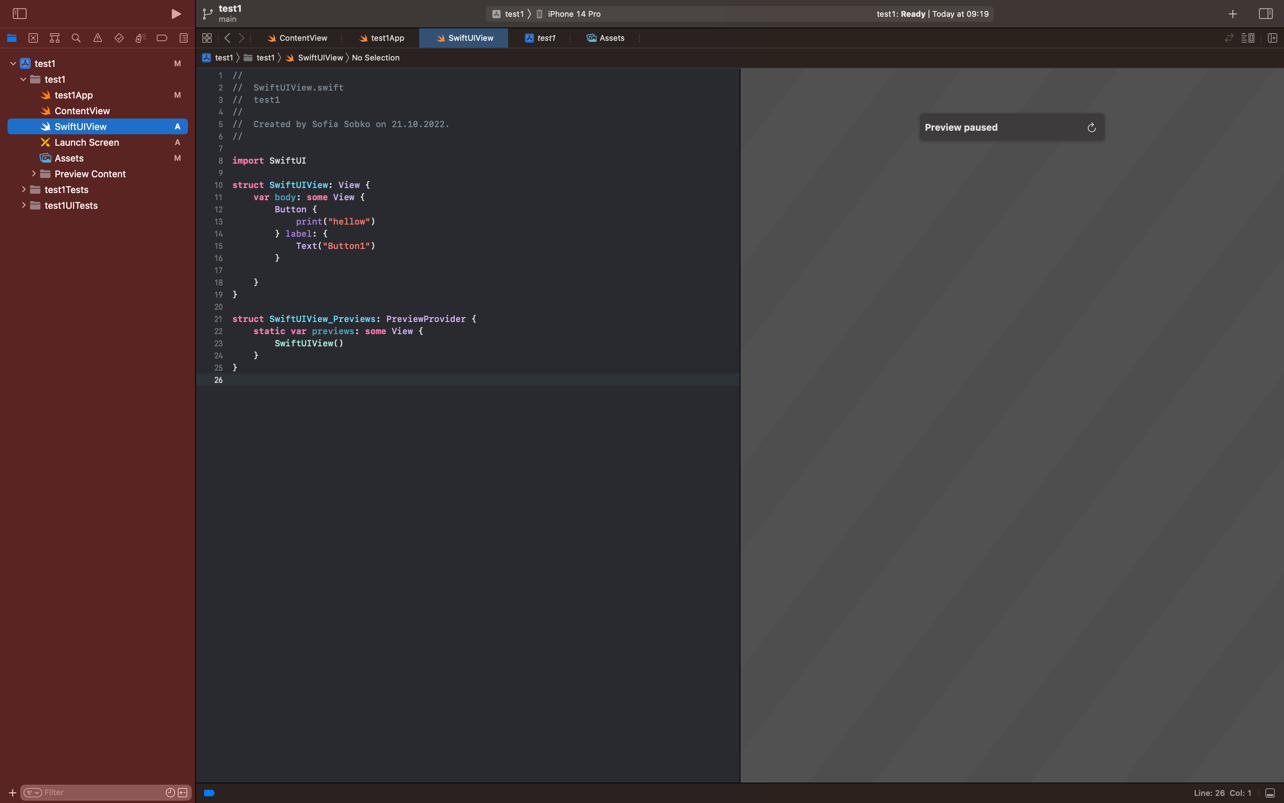
Источник: <https://metanit.com/swift/tutorial/2.7.php>

* For

С помощью цикла for-in мы можем перебрать элементы коллекции (массивы, множества, словари) или последовательности.

Источник: <https://metanit.com/swift/tutorial/2.10.php>

**Проект в Xcode:**



**Рис. 7. Урок 3 – в Xcode**

**Монтаж:**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Рис. 8. Урок 3 – монтаж**

**ПревьюA picture containing logo

Description automatically generated**

**Рис. 10. Превью в YouTube**

**Готовые уроки**

Урок 1 – Intro: <https://youtu.be/ToH2kdfYKbA>

Урок 2 – <https://youtu.be/S18JQOutP90>

**Приложение 1**

**Приложение 1**

Техническое задание:

Text, letter

Description automatically generated

Table

Description automatically generatedText, application

Description automatically generatedText, letter

Description automatically generatedText

Description automatically generatedText

Description automatically generatedText

Description automatically generatedGraphical user interface, text, application, email

Description automatically generatedText, letter

Description automatically generatedText, letter

Description automatically generatedGraphical user interface

Description automatically generated