Этап I

Модуль 1

1. Константы и переменные (let, var)
2. Типы данных
    - Character и String
    - Int
    - Float
    - Double
    - Boolean
    - nil и опциональные типы
    - конвертация типов данных (к примеру, string в int и наоборот)
3. Арифметические операторы
4. Условный оператор
    - if
    - guard
    - switch
5. Логические операторы
    - &&
    - ||
    - !
6. Операторы сравнения
    - “>” и “<”
    - ==
    - !=
    - “>=” и “<=”

Задание:

Создать 5 переменных Int, Double, Float, String, String с числами.

Уточнение:

- Первая строковая переменная принимает числа в виде: "1", "2" и тп.
- Вторая строковая переменная принимает числа в виде: "один", "два" до 10.

Нужно просуммировать все переменные. Необходимо учесть обработку ввода некорректных значений.

Модуль 2

1. Коллекции
    - array
    - dictionary
    - set
2. Циклы
    - while
    - for
3. Перечисления (Enum) в связке с Switch

Задание:

1. Создать два массива, первый массив количество дней, второй массив название месяцев с соответствующей индексацией. К примеру: [январь, февряль, ...] [31, 28, ...]
    - Пройтись по массиву месяцев и вывести текст: В месяце "значение с массива месяцев" дней = "значение с массива количество дней"

2. Создать массив тюплюв. Тюпл отображает количество дней и название месяца.
    - Пройтись по массиву тюплов и вывести текст: Месяц: "значение с массива", количество дней: "значение с массива"

3. Создать дикшенери в котором ключ название месяца, значение количество дней
    - Пройтись по дикшенери и вывести текст: Месяц: "значение с массива", количество дней: "значение с массива"

4. Создать массив чисел, значения которые разного типа(Инт, Флоат, Стринг).
    - Вывести сумму всех чисел в массиве, которые могли посчитать, в виде: Сумма всех чисел = "сумма"

Модуль 3

1. Tuples
2. Функции
    - ничего не принимает, ничего не возвращает
    - принимает один или несколько параметров, ничего не возвращает
    - ничего не принимает, возвращает один или несколько параметров
    - принимает один или несколько параметров, возвращает один или несколько параметров

3. Замыкания
4. Структуры и классы
    - свойства
    - методы
5. “Библиотечные” функции и свойства
    - пройти стандартные функции и свойства для Array, Dictionary, String, Int и т. п.

Задание:

- Создать Класс User в котором будет имя, фамилия, возраст, метод который возвращает небольшую биографию этого юзера типа String.

- Создать массивы с Именами, Фамилиями, Биографиями.

- Создать массив в котором будет 100 юзеров, юзеры создаются с рандомными именами, фамилиями, биографией, возрастом(возраст от 1 до 120).
- Из получившегося массива:
  - вывести количество юзеров которые старше 30 лет.
  - создать новый массив в котором будут юзеры от 18 до 35 лет
  - вывести фамилии юзеров у которых имя к примеру Андрей(любое придуманное вами имя, но с вашего массива Имён)

Этап II

Модуль 1

1. Начало работы с UI элементами
    - создать Single View App. Зайти в main.storyboard
    - узнать что такое outlet и action
    - положить UIButton на текущий экран и прокинуть outlet и action в существующий ViewController
    - по нажатию на кнопку вывести текст “Произошло нажатие на кнопку” и поменять цвет кнопки с белой на синий. А при повторном нажатии наоборот
2. UI элементы
    - Ulimage
    - UILabel
    - UITextField
    - UITextView
    - UIButton
    - нужно уметь выполнять обычную работу с этими элементами. К примеру, сохранить в переменную вводимый текст с UITextField, присвоить текст с переменной в UILabel/UITextField и т. п.
3. Верстка интерфейсов (Auto Layout)

Задание:

- Есть два UITextField, для ввода имени и фамилии и UITextView для ввода дескрипшина.
- Есть две кнопки "Save" сохраняет юзера с вводимыми значениями в массив.
- Есть кнопка "Print n users", где n количество сохраненных юзеров. По нажатию на эту кнопку выводим в консоль всех юзеров.

Пример экрана

Модуль 2

1. UITableView
2. UIColectionView
3. “Custome cell” для UITableView и UICollectionView
4. UINavigationController
    - навигация между контроллерами (push/present)
5. Переходы между экранами и передача данных между ними
    - segue

- кодом (подымая экран со сториборда)
- кодом (подымая экран с xib)

Задание:

- Используем наработки с "Этап 2 - Модуль 1 - Задание"
- Изменить кнопку "Print n users" на "Show n users", где n количество сохраненных юзеров. При нажатии на эту кнопку переходим на экран с UITableView где отображаем список юзеров. В ячейке отображается имя, фамилия и описание. Уточнение: в ячейке находится три UILabel.
- Отображение информации должно быть полным, то есть дескрипшн не должен быть обрезан.

Модуль 3

UITabBarController

Задание:

- Используем наработки с "Этап 2 - Модуль 1 - Задание" - первый экран UITabBarController. Убираем вторую кнопку, оставляем только кнопку "Save".
- Используем наработки с "Этап 2 - Модуль 2 - Задание" - второй экран UITabBarController. Отображаем список юзеров, которых создали на первом экране.

Этап III

Модуль 1

1. NSCoding
2. NSKeyedArchiver / NSKeyedAnarchiver
3. UserDefaults
4. CoreData

Задание:

- Используем наработки с "Этап 2 - Модуль 2 - Задание"
- Сохраняем массив юзеров, чтобы при повторном запуске отображались юзеры которые были сохранены.
- Сохранить юзеров сначала в UserDefaults потом в CoreData

Модуль 2

Парсить json в объект

- dictionary to object
- Codable / Decodable

Задание:

Положить json файл в объект:

- распарсить в объект с dictionary
- распарсить с помощью Decodable

Модуль 3

Networking

- что такое API
- URLSession

Задание:

- Используем любое публичное API. Распарсить полученный респонс с помощью Decodable
- Для простоты можно использовать это API:
  https://openweathermap.org/api
  path: https://api.openweathermap.org
  api key: 92ee2a79fa0115d18652f5b6f9f540cf

Этап IV

- Наследование
- Протоколы
  - протокольно-ориентированное программирование (POP)
- Extension
- Паттерны
  - архитектурный паттерн MVC
  - Singleton
  - Factory
  - Abstract factory
  - Decorator
  - Adapter
  - Facade
- Система контроля версий Git
- CocoaPods

Тестовое задание:

Можно использовать любое публичное api.

Используем UITabBarController для отображения двух списков:
1 Список загруженный от сервера.
2 Список лайкнутых элементов.

Задача 1:
Отобразить список в котором отображается информация о запрашиваемых данных с сервера. (Использовать пагинацию *) - пагинацию используем по желанию.
Стиль ячейки: (Отобразить картинку *), отобразить название, (отобразить краткое описание *). В правом углу добавить кнопку для лайк/дизлайк выбранного элемента.

Задача 2:
Экран деталей полученной информации.
Стиль экрана: (Отобразить картинку *), отобразить название, (отобразить описание *). В правом верхнем углу экрана добавить кнопку для лайк/дизлайк выбранного элемента.

Задача 3:
Отобразить список в котором отображаются лайкнутые элементы.
Стиль ячейки: (Отобразить картинку *), отобразить название, (отобразить краткое описание *). В правом углу добавить кнопку для лайк/дизлайк выбранного элемента.

Задача 4:
Экран деталей лайкнутого элемента.
Стиль экрана: (Отобразить картинку *), отобразить название, (отобразить описание *). В правом верхнем углу экрана добавить кнопку для лайк/дизлайк выбранного элемента.

Уточнение:
Допустим мы делаем "лайк/дизлайк" элемента на экране деталей - ожидаю что при возвращении на список этот элемент будет также "лайкнут/дизлайкнут" и наоборот.

* если это предусмотрено выбранным api

Внешний вид приложения
https://projects.invisionapp.com/share/9UPHVP6SB2Y#/screens/335559481)

Советуемый ресурс:

https://swiftbook.ru/content/languageguide/basics/