Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования.
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «ИНЖЕНЕРНЫЙ БИЗНЕС И МЕНЕДЖМЕНТ»

КАФЕДРА «ПРОМЫШЛЕННАЯ ЛОГИСТИКА»

Домашнее задание

по дисциплине

«Парадигмы и конструкции языков программирования»

НА ТЕМУ:

SWIFT

Студенты ИБМ3-34Б _____ В.А. Леонтьева
(Подпись, дата)

Преподаватель _____ Ю.Е. Гапанюк
(Подпись, дата)

История создания языка программирования Swift

Именно на языке программирования Swift создавалась в своё время (это были 1989-1995 годы) платформа NeXT. Позже её взяли за основу для системы macOS, а потом ещё и для iOS.

Тот вид, который язык имеет на сегодняшний день, был в процессе разработки с 2010 по 2014 год. Наконец, произошло его официальное представление. Распространялся продукт через «iBook Store».

Новая версия Swift 2.0 увидела мир в 2015 году. Она производительнее предыдущих. Плюс язык программирования Swift имеет новый API для выявления и ликвидации ошибок, оптимизированный синтаксис. Кроме того, есть опция проверки доступности возможностей языка именно для тех операционных систем, под которые готовится разработка. Через полгода после данной версии вышло и ещё одно обновление под номером 3.0.

Следующий вариант Swift 4.0 появился осенью 2017 года, далее, через год – очередная стабильная разработка 4.2. В это же время вышла бета-версия 5.0, и позже она тоже стала стабильной.

Крупное обновление вновь состоялось в сентябре 2019 и было представлено версией 5.1. Актуальная 5.4 увидела свет в конце 2021 года, и, возможно, очередные обновления ещё будут.

Основные характеристики Swift

Swift – это новый язык программирования, на котором создаются приложения для iOS, macOS, watchOS и tvOS. Впрочем, если вы пользовались C и Objective-C, то могли встречать и там многие части Swift.

Только в Swift заложены свои версии для фундаментальных типов C и Objective-C. Имеются в виду int для целых чисел, Double и Float для показателей с плавающей точкой, Bool для булевых показателей и String для текстовых объектов. Кроме того, в разделе Типы коллекций прописано, что Swift включает в себя основные три типа, а именно, Array, Set и Dictionary, причем довольно сильные их версии.

В Swift (как и в C) обращение к значениям выполняется по уникальному имени, а для хранения задействуются переменные. Причем используются и те из них, значения которых меняться не могут. Они считаются константами, и в сравнении с константами в C – они мощнее. Вообще в Swift константы используются очень активно, за счет этого код получается более чистым и безопасным, если в нем есть показатели, которые не должны изменяться.

Кроме уже известных типов в Swift есть ещё и расширенные (в Objective-Сих нет). Среди них – кортежи, с помощью которых создаются и передаются группы значений. Ещё кортежи дают возможность брать несколько значений из функции представлять в виде одного целого показателя.

Опционные типы, работающие с отсутствующими значениями, в языке программирования Swift тоже имеются. Данные типы либо указывают на наличие некоего значения (и определяют его величину, например, x), либо говорят, здесь никакого значения нет.

Это нечто схожее с использованием nil указателей в Objective-C, но тут доступна работа не только с классами, а со всеми типами. Вообще в сравнении с nil указателями в Objective-C, опциональные значения безопаснее и чётче, это, собственно, важный элемент многих мощных особенностей Swift.

Swift считается типобезопасным языком, он сам указывает, для работы с какими типами подходит ваш код. Если куску кода нужен на входе String, то безопасность типов не позволит ошибочно передать ему Int. И это ещё не всё. Если кусок кода ожидает неопциональный String, то безопасность типов предотвратит случайную передачу опционального String. То есть, система «ловит» ошибки и вносит исправления прямо на стадии разработки.

Почему стоит учить Swift

Начинать знакомство с программированием с какого-либо старого языка – плохая идея. Если говорить об изучении языков программирования с

нуля (для новичков), то лучше сразу браться за что-то современное, вроде Python, Ruby, Swift. Кому-то из специалистов не очень нравятся последние версии Swift, но не будьте и вы тоже скептиком. Поверьте, для этого есть причины.

1. Работа с платформами Mac и iOS

За 2016 год разработчики получили с приложений для Mac и iOS около 20 миллиарда долларов. Столько собрал AppStore. Учитывая, что Apple забирает через AppStore 30 % прибыли, объем всего рынка получается равным примерно 28,5 миллиардов долларов. Так почему бы вам не получить здесь свой куш?

Приложения для Mac и iOS пишутся на языке программирования Swift, который по сути уже заменяет Objective-C (всё еще используемый, однако уже морально устаревший, не имеющий перспектив). Поэтому сегодня Swift – актуальный инструмент для разработки приложений под обе платформы, и ут можно неплохо заработать.

2. Swift – язык для быстрого развития

Swift создавался как замена для Objective-C, и здесь для команды Apple было два важных момента: получить язык простой в изучении и позволяющий выполнять разработку приложений быстрее.

На выходе получился мощный, современный язык, во всех отношениях превосходящий Objective-C.

С использованием Swift у вас появляется больше времени на генерацию и реализацию своих задумок, потому что о возможных сбоях и ошибках в коде можно уже не беспокоиться. Ещё один плюс – синтаксис языка программирования Swift менее многословен, в сравнении с Objective-C, за счет чего проще стал процесс записи и чтения. То есть, опять же, сокращается время на создание кода.

3. Рост популярности, спрос на специалистов

В отчете GitHub Octoverse Swift по популярности занимает 13-е место среди прочих языков, используемых для написания проектов с открытым кодом. На бирже разработчики приложений в цене, и постоянно растет спрос

на специалистов, владеющих языком Swift. Так что подобный навык смело можно закладывать в основу своей карьеры, это будет абсолютно правильный выбор

4. Востребованность языка программирования Swift для Apple

В ближайшие несколько десятков лет Apple ни на что другое Swift не променяет, причин для этого нет. При этом язык постоянно развивается, популярность его растет, как и продажи (а так же и ассортиментный перечень) продукции с «яблоком». Поэтому Swift-специалисты в ближайшем будущем будут неизменно востребованы.

Преимущества языка SWIFT

Принципиальными особенностями языка Swift компания Apple называет защищенность, быстродействие и простоту. Swift продолжает активно развиваться, а популярность обрел благодаря ряду своих ценных преимуществ. Это:

1. **Высокие показатели производительности**. Да, язык Swift рассчитан на быстрое изучение, но притом еще и демонстрирует высокую скорость работы. Он быстрее, чем Objective-C в почти в 2,6 раза, и опережает по скорости Python 2.7 почти в 8,4 (это данные от Apple). Вообще целью ставится опередить по скорости даже C++. При этом язык программирования Swift не просто работает быстро, он обладает широкими современными возможностями для написания действительно функционального кода. Тут доступны кортежи, замыкания, дженерики, итераторы, множественные возвраты, встроенные шаблоны ФП и много всего прочего.
2. **Синтаксис языка достаточно прост**, тут всё логично, минималистично и чётко структурировано, что позволяет обходиться без громоздких кодов. Кроме того, код легко читается, в нем разберется не только опытный специалист, но и новичок. Если сравнивать с Objective-C, то в Swift и синтаксис проще, и объём кода меньше.

3. **Уровень безопасности**. В Swift предусмотрена защита от несанкционированного доступа к данным, от их утечки. Имеется контроль критических сценариев, штатные ошибки исправляются в автоматическом режиме (а не вручную, как в C++). Специально затруднено обращение к неправильным частям памяти, а так же предотвращаются ошибочные действия с данными. Ошибки обрабатываются максимально эффективно, что сводит число сбоев и критических сценариев (а значит, и вариантов непредсказуемого хода событий) к минимуму.

4. **Бесплатный доступ, открытость.** Сейчас есть много языков с открытым исходным кодом, но такой широкий жест от Apple (в случае со Swift) – явление редкое. Apple наоборот обычно старается придерживаться проприетарных технологий, чтобы выделиться на фоне других. Но предоставление открытого исходного кода всему сообществу оправдало себя и принесло свои плоды. Теперь пользователи могут предлагать варианты улучшения функционала, исправления ошибок, адаптировать предложения к другим платформам (кроме Mac и iOS). Собственно, пользователи и выступают в роли главной движущей силы языка.

5. **Удобная рабочая среда.** Имеется в виду интегрированная среда Xcode (от Apple), которая как раз и сделана для того, чтобы разрабатывать ПО для iOS и macOS. Тут есть необходимый набор компиляторов LLVM, инструментарий для создания графических интерфейсов Interface Builder. Имеется и официальная документация разработчика от Apple.

6. **Обратная интеграция с Objective-C.** То есть, проекты, написанные на Objective-C, можно переделать на Swift. Кроме того, внутри Swift-проектов можно задействовать возможности Objective-C.

7. **Динамический (а не статический) формат библиотек.** Благодаря этому в готовые Swift-коды можно быстрее вносить изменения и улучшения, ждать релиза новой версии iOS не нужно. Плюс

программисты для своих приложений могут создавать отдельные библиотеки.

Кроме всего перечисленного тут ещё есть новая система перечислений, хорошо оптимизированный автоматический сборщик мусора, тонкое управление аксессорами. А адаптация C-библиотеки Grand Central Dispatch позволяет еще и обеспечивать многопоточность. Наряду с плюсами у языка программирования Swift, разумеется, есть и минусы.

Некоторые недостатки Swift

Если говорить о недостатках, то все они связаны лишь с тем, что язык программирования Swift пока еще довольно молод, поэтому есть проблемы, требующие доработок. Apple, разумеется, этим занимается, но недочёты пока ещё есть. Ознакомьтесь с ними заранее, если язык вас заинтересовал, и вы планируете начать его изучение.

- Библиотек здесь немного. Их число растет, но в том же Python дополнений больше. Кроме того, библиотеки новых версий не подходят для предыдущих релизов языка.
- Обратной совместимости со старыми версиями нет. И это плохо. Ведь как только выходит новая версия, разработчики вынуждены переписывать коды в соответствии с ней. А крупные обновления, заметьте, появляются примерно каждые два года (это довольно часто).
- Ранние версии iOS не поддерживаются. То есть, написать приложение для iOS ниже версии 7 на Swift нельзя из-за их несовместимости. Впрочем, iOS версии 6 и более низких установлены лишь на 5% «яблочных» устройств.
- Немногочисленное сообщество разработчиков. Конечно, их число очень быстро растет вместе с популярностью инструмента. И всё же, новичкам не так просто бывает найти решение проблемы, если что-то не так с кодом, написанным на StackOverflow (например).

Сферы применения языка

Для чего же нужен язык программирования Swift? Как известно, это инструмент экосистемы Apple. То есть, у него довольно узкая область применения, а именно – разработка приложений под платформы iOS и MacOS.

Впрочем, есть и другие возможности для использования данной технологии. В частности – для разработки серверных решений в виде альтернативных языков Go и Rust.

Существует уже несколько фреймворков языка программирования Swift, которые задействуются в бэкенд-проектах.

- Perfect. С его помощью пишутся серверные части приложений. Это, пожалуй, самый активно используемый фреймворк. Совместим с WebSocket, ORM и с коннекторами баз данных.
- Vapor. По структуре – проще предыдущего варианта, но обладает таким же широким функционалом и больше подходит для веб-разработки. Плюсом является обширная документация.
- Zewo. Из всех существующих фреймворков – наименее продвинутый вариант. В сегодняшних реалиях им практически невозможно пользоваться, нужно ждать улучшений.
- Kitura. Документация тут гораздо скромнее, чем у Perfect и Vapor, однако инструмент поддерживается даже IBM. По дизайну продукт похож на js.

На языке Swift можно было бы писать программы и для Linux (при соблюдении определенных условий), однако на сегодняшний день у продукта для этого недостаточно библиотек.

И всё же есть энтузиасты, жаждущие экспериментов. Они создают пробные биндинги для использования вместе с GTK+. Вполне вероятно, что очень скоро появятся все необходимые библиотеки, тогда на Swift будет спокойно разрабатывать продукты для Linux.

На языке программирования Swift написаны многие приложения для Apple App Store. Собственно, именно на нем разработана большая часть

приложений (кроме, разве что, устаревших), используемых в устройствах Apple. Потому что для данных целей именно Swift является рекомендованным продуктом.

Сложности в изучении языка программирования Swift

Продукт достаточно прост в изучении, в том числе и для новичков. Но трудности, пусть и небольшие, всё-таки будут. Обучение кодингу с нуля на примере языка программирования Swift лучше не начинать, для этого есть варианты попроще, тот же Python, к примеру.

Хотя, стоит признать, именно Swift весьма дружелюбен с новичками, потому что там задействуются новейшие программные паттерны, подстраховывающие от совершения большого числа распространенных ошибок в программировании. Плюс есть песочница, что дает возможность не писать сразу громоздкие коды, а работать по необходимости с отдельными участками, а затем проверять промежуточные результаты без компиляции и выполнения кода.

Сложности в изучении появляются из-за наличия в языке строгой типизации. И ещё нужно учитывать, что Swift не получится охватить, что называется, «слёту», придется сначала приложить усилия и изучить Objective-C.

Каким количеством времени нужно располагать, чтобы изучать язык программирования Swift? Если будете тратить на это примерно час в день, то за пару месяцев освоите базу. Конечно, охватить основы платформы получится быстрее, если вы сможете посвящать этому рабочий день целиком. А вот, чтобы писать собственные приложения, нужно будет как следует попрактиковаться в течение нескольких месяцев.

Не менее полугода, а скорее и целый год уйдёт на то, чтобы стать профессионалом в создании ПО для iOS или Apple. Освоение навыков в полном объёме займет много времени, если вы хотите стать действительно хорошим специалистом по Swift.

И тут ещё с самого начала нужно понимать, что за какой бы язык программирования вы не взялись, его изучение будет продолжаться до бесконечности. Для Swift постоянно выходят обновления и фреймворки, там всегда есть над чем поработать. В целом окончательный результат зависит от вашего желания овладеть языком и времени, которое вы сможете этому уделять.

Алгоритм обучения Swift

Да, Swift – язык довольно молодой, но неплохие ресурсы для его изучения уже есть, и для новичков они тоже отлично подходят.

Начинать нужно с самых основ. Но в первую очередь загрузите себе приложение Xcode. Это продукт от Apple, специально предназначенный для разработки программного обеспечения. И изучите документацию Apple Swift. Там вы найдете кучу руководств, среди которых будут и те, что окажутся полезными для первичного ознакомления с основами языка.

Далее перечислены основные темы, которые нужно будет освоить, если вы планируете стать специалистом по Swift.

Синтаксис и переменные языка программирования Swift

Знакомство с любым языком программирования начинается с изучения принципов его работы и задействованных в нём правил. И здесь, конечно, без синтаксиса не обойтись.

После того, как с базовым синтаксисом вы познакомитесь, можно начинать изучение переменных. Для чего они нужны? Чтобы хранить внутри программы те или иные данные. Ниже представлен перечень тем, необходимых для освоения в рамках изучения синтаксиса и переменных языка

Простые типы данных. Используйте `let` для создания константы и `var` для создания переменной. Тип константы указывать не нужно, вы можете присвоить ей значение лишь единожды.

```
var myVariable = 42
myVariable = 50
let myConstant = 42
```

Условия и циклы. Для создания условий используются операторы `if` и `switch`, для создания циклов – `for-in`, `for`, `while` и `do-while`. При этом выделять круглыми скобками условия и инициализирующие выражения необязательно, тогда как фигурные скобки обязательны.

```
let individualScores = [75, 43, 103, 87, 12]
var teamScore = 0
for score in individualScores {
    if score > 50 {
        teamScore += 3
    } else {
        teamScore += 1
    }
}
teamScore
```

Ресурсы для изучения языка

Возможности для изучения Swift существуют самые разные: уроки в видеоформате, материалы для чтения, с преподавателем и без. Стиль и темп обучения выбирайте на своё усмотрение, ищите ресурсы, максимально подходящие вам по формату.

- Уроки в онлайне

Воспользуйтесь онлайн руководствами, следуйте описываемым там инструкциям, применяйте полученные знания на практике. С помощью учебных пособий вы научитесь писать коды, а затем и вносить в них изменения.

- Интерактивное обучение на игровых площадках Apple Swift

У Apple есть для этого специальное приложение, называется Swift Playgrounds. Оно разработано специально для обучения программированию и

знакомит пользователя с основами Swift, в интерактивной форме учит работе с ним.

- Специальные ресурсы для программистов от Apple

На сайте Apple Developer ресурсы есть самые разные, рассчитанные и на новичков, в том числе. Плюс вы можете воспользоваться специальными книгами либо курсами для изучения языка программирования Swift, опубликованными Apple. Ресурсы очень хорошие и притом бесплатные.

- Станьте членом сообщества разработчиков

Изучать программирование самостоятельно, без поддержки, возможно, не так уж и просто. Наверняка вы столкнетесь с трудностями. Поэтому не будет лишним присоединиться к сообществу разработчиков, когда начнете осваивать Swift. В подобном сообществе разработчики самых разных уровней поддерживают друг друга, делятся идеями, предлагают полезное сотрудничество.

Ресурсы для изучения языка

На начальной стадии изучения Swift рассмотрите для себя участие в следующих сообществах:

- r / iOSProgramming. Платформа, объединяющая разработчиков для iOS. Тут вы найдете много для себя полезного, свежие новости касательно iOS.
- Dev.to. Ещё одно сообщество, в котором есть отдельная ветка, посвященная iOS. Тут обитают и новички, и профессионалы, они активно обмениваются идеями и оказывают друг другу всяческую поддержку.
- StackOverflow. Субреддит, работающий по принципу «вопрос-ответ». Рано или поздно большинство разработчиков сюда приходят, потому что если у вас возникает какой-то вопрос, то скорее всего, на StackOverflow его уже задавали, и там есть готовый ответ.

Став членом сообщества, разберитесь с тем, как в нем налажен рабочий процесс. Попробуйте действовать по примеру тех участников, что там уже

есть. По возможности старайтесь помогать другим, вносите свою посильную лепту.

- Осваивайте язык в процессе практики

Итак, сначала вы изучаете основы, базовые знания по Swift. Далее необходимо пробовать применять на практике полученные знания, то есть, приступать к реализации какого-либо проекта. Постепенно вы освоите концепции программирования, и тогда нужно будет взяться за проект, где можно будет реально применить полученные навыки. Так вы закрепите полученные теоретические знания и одновременно глубже освоите все тонкости языка программирования Swift.