Технічні науки

УДК 004.852

Петрішенко Сергій Олександрович
студент
Національний технічний університет України
«Київський політехнічний інститут»
Петришенко Сергей Александрович
студент
Национальный технический университет Украины
«Киевский политехнический институт»
Petrishenko S.
student
National Technical University of Ukraine “Kyiv Polytechnic Institute”

ПОРІВНЯЛЬНИЙ АНАЛІЗ SWIFT І OBJECTIVE-C
СРАВНИТЕЛЬНЫЙ АНАЛИЗ SWIFT И OBJECTIVE-C
COMPARATIVE ANALYSIS OF SWIFT AND OBJECTIVE-C

Анотація: Розглянуто основні мови програмування під платформу iOS. Проаналізовано основні переваги Swift.

Ключові слова: мова програмування, мобільна платформа iOS, середовище розробки, управління пам’яттю, WWDC.

Аннотация: Рассмотрены основные языки программирования под платформу iOS. Проанализированы основные преимущества Swift.

Ключевые слова: язык программирования, мобильная платформа iOS, среда разработки, управления памятью, WWDC.

Summary: Were considered main programming languages for iOS. Were analyzed basic advantages of Swift.

Програмування – основа основ комп'ютерної техніки. Корпорація Apple давно відома своїм умінням задавати тон розвитку індустрії на роки вперед, але, з огляду на поступове сходження розробників Apple з ринку професійних рішень, вона асоціюється в першу чергу з споживчими товарами. Однак чергове дослідження громадської думки показує, що розробники ПЗ більш ніж задоволені свіжою пропозицією компанії – мовою програмування Swift. Згідно з інформацією ресурсу Stack Overflow, майже 80 відсотків професіоналів із задоволенням працювали або планують працювати з випущеним не так давно інструментом розробки від Apple (рис. 1) [3]. Дослідникам вдалося опитати 26 тисяч відвідувачів з більш ніж 150 країн світу. Близько третини постійно зайнятих в сегменті написання мобільного ПЗ респондентів працюють в основному на платформі iOS, а менше половини також займаються створенням додатків для конкуруючої ОС Android. 20 % опитаних не змогли визначитися, швидше за все, сюди входять фахівці, які регулярно розробляють програмне забезпечення для різних платформ.

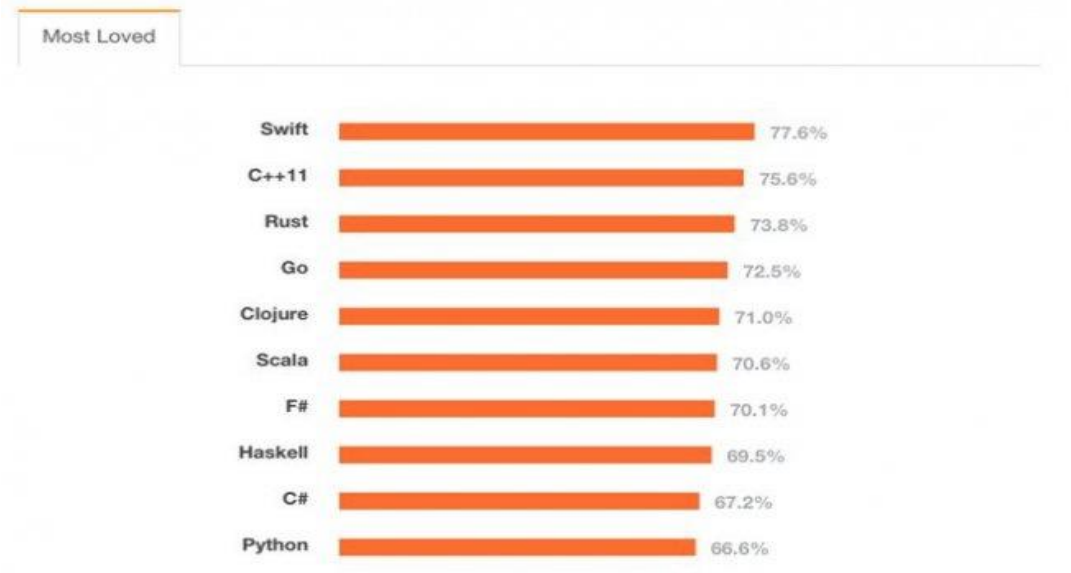

Рисунок 1.4 – Оцінки мов програмування

Swift є багатопарадигмовою компільованою об'єктно-орієнтовною мовою програмування, яка була створена компанією Apple для роботи пліч-о-пліч з Objective-C. Swift була представлена 2 червня 2014 році на

Всесвітній конференції розробників (WWDC). Swift була створена в першу чергу для розробників під платформу iOS. Вона покликана повністю замінити Objective-C, адже комнанія Apple покладає великі надії на свій продукт, тому Apple настільки ефективно оптимізувала компілятор і саму мову в принципі, що багато можливостей ще тільки належить розкрити. Можна сказати, що Swift «призначена для зльоту від «Hello, World» до цілої операційної системи.

Swift працює з фреймворками Cocoa і Cocoa Touch і сумісна з основною кодовою базою, написаною на Objective-C. Вона являється простою і гнучкою мовою, маючи в той же час більш читабельний код в порівнянні з Objective-C. Адже Objective-C має всі болячки, які можуть бути у мови, побудованої на C. Для диференціації ключових слів і типів, Objective-C вводить нові ключові слова, використовуючи символ «@». Так як Swift не побудована на C, то вона може об'єднати всі ключові слова і видалити численні символи «@» перед кожним Objective-C типом або перед пов'язаним з об'єктом ключовим словом. Також більше не потрібно ставити кому в кінці рядка або писати круглі дужки, щоб виділити умовні вирази всередині операторів if/else. Найбільшою перевагою Swift перед Objective-C є те, що виклики методу не розташовуються усередині один одного. Для виклику функцій і методів в Swift використовується стандартний, розділений комами, список параметрів в круглих дужках. В результаті отримуємо мову зі спрощеним синтаксисом і граматикою [2].

Особливістю Swift є відмова від використанням і підтримуванням програмістами одночасно двох файлів: файлу заголовка і файлу реалізації. Середовище розробки Xcode і компілятор LLVM (Low Level Virtual Machine) може з'ясувати залежність і виконати покрокові зміни автоматично в Swift. Тому, поділу змісту від тіла стає справою минулого. Swift поєднує в собі заголовок Objective-C (.h) і файли реалізації (.m) в одному файлі коду (.swift). Двох-файлова система Objective-C накладає

додаткову роботу на програмістів - і це робота, яка відволікає їх від більш важливих завдань. У Objective-C потрібно вручну синхронізувати імена методів і коментарі між файлами. Xcode і компілятор LLVM можуть працювати непомітно для програміста, знижуючи його навантаження. Використовуючи Swift, програмісти роблять менше допоміжних дій і можуть витратити більше часу на створення логіки додатка. Отже, можна сказати, що Swift створена в тому числі для середнього програміста, адже має в своєму арсеналі містить досить розумні інструменти, які допомагають писати власні програми легше. Структура Swift дозволяє уникнути багатьох поширених помилок, які часто допускають розробники-початківці [1].

Swift є type-safe мовою, вона краще розуміє дії програміста і дозволяє робити більше за менший час. Одним з цікавих моментів в Objective-C можна вважати спосіб обробки вказівників, особливо nil (NULL). У Objective-C нічого не трапиться, якщо викликати метод зі змінною вказівника nil. Рядок коду, де міститься виклик цього методу, не спрацює, і програма продовжить свою роботу, але насправді буде повно помилок, що призведуть до непередбачуваної поведінки. Опціональні типи дають можливість існування в Swift коді nil (опціонального значення), що говорить про можливість створення помилки компілятора при написанні поганого коду. Це створює короткий цикл зворотного зв'язку і дозволяє програмістам писати програми більш впевнено. Проблеми можуть бути усунені при вже написаному коді, що значно зменшує кількість часу і грошей, які витрачаються на виправлення помилок, а також кількість коду, в порівнянні з передачею вказівника на NSError в Objective-C [2].

Традиційно в Objective-C якщо значення було повернуто з методу, то програміст був зобов'язаний зафіксувати поведінку вказівника змінної, що повернулась. В Swift опціональні типи і типи значень дозволяють явно визначити метод, якщо значення існує, або якщо воно може бути

опціональним. Для забезпечення передбачуваної поведінки, Swift викликає помилку при виконанні коду, якщо використовується опціональна змінна nil. Ця помилка забезпечує узгоджену поведінку, яка полегшує процес усунення багів, тому що змушує програміста вирішити проблему відразу. Ця помилка виникне на тому рядку коду, де була використана опціональна змінна nil. Це означає, що помилка буде виправлена своєчасно або її вдасться уникнути зовсім [2].

Головною особливістю є те, що Swift має незалежне управління пам'яттю (Automatic Reference Counting - автоматичний підрахунок посилань). Swift уніфікований так, як ніколи не була Objective-C. У Objective-C ARC підтримується всередині Cocoa API, але він не доступний для C коду і API. Це означає, що програміст буде повинен взяти на себе управління пам'яттю при роботі з API, які доступні на iOS. Величезні витоки пам'яті, які програміст може мати в Objective-C, неможливі в Swift. Програмісту не доводиться думати про виділення пам'яті для кожного об'єкта, який він створює, бо ARC обробляє всі управління пам'яттю під час компіляції, і витрати, які пішли б на управління пам'яттю, тепер можуть бути сфокусовані на основній лозіці додатка і нових можливостях. Це відбувається тому, що ARC в Swift працює і на процесуальному, і на об'єктно-орієнтованому коді, і тепер не потрібно контекстних переходів для програмістів, навіть якщо вони пишуть код, який розрахований на більш старі API. Це є проблемою для поточної версії Objective-C. Apple змогли вирішити і довести, що автоматичне і високопродуктивне управління пам'яттю може підвищити продуктивність [1].

Swift використовує меншу кількістю коду. Наприклад, для додавання двох рядків Swift використовує оператор «+». Ця особливість відсутня у Objective-C. Така підтримка для об'єднаних символів і рядків має важливе значення для будь-якої мови програмування, яка використовує відображення тексту для користувача на екрані.

Система типів в Swift знижує складність в написанні коду, так як компілятор може з'ясовувати типи. Наприклад, Objective-C вимагає, щоб програмісти запам'ятовували спеціальні маркери рядків (% S,% d,% @) і використовували розділений комами список змінних. Swift підтримує інтерполяцію рядків, що усуває необхідність запам'ятовування символів і дозволяє програмістам вставляти змінні, такі як назва ярлика або кнопки, безпосередньо у вбудований рядок користувача. Тип виведення системи і рядки інтерполяції зменшують кількість помилок, які поширені в Objective-C [2].

Компанія Apple велику увагу приділила швидкодії Swift. Як стверджують самі розробники, алгоритм пошуку в Swift виконується до 2,6 разів швидше, ніж в Objective-C, і до 8,4 раза, ніж в Python 2,7. Відповідно до даних Primate Labs, за показниками популярного тесту GeekBench (крос-платформний еталонний тест для вимірювання швидкодії процесора і підсистеми пам'яті комп'ютера) Swift наблизився до експлуатаційних характеристик C++ по обмеженню швидкості обчислень з використанням алгоритму Мандельброта (знаходження множини Мандельброта) в грудні 2014 року [4]. У лютому 2015 року Primate Labs виявили, що Xcode 6.3 Beta поліпшив продуктивність алгоритму GEMM в Swift (алгоритм обмеженої пам’яті з послідовним доступом великих масивів (a memory-bound algorithm with sequential access of large arrays)) до коефіцієнта 1,4. Початкова імплементація FFT (алгоритму обмеженої пам'яті з випадковим доступом великих масивів) - поліпшення продуктивності в 2,6 рази. Swift показав і інші поліпшені показники при подальшому тестуванні: 8,5-кратне підвищення для FFT алгоритму (залишивши C++ тільки з приростом продуктивності в 1,1 рази). Також, Swift перевершує C++ для алгоритму Мандельброта з коефіцієнтом в 1,03. Практично, Swift на одному рівні з C++ для FFT і алгоритму Мандельброта. Відповідно до даних Primate Labs алгоритм GEMM показав, що компілятор Swift не може векторизувати код

так, як може компілятор C++. У C++ є невелика перевага, але Apple обіцяють все виправити в наступних версіях Swift [5].

Одним із важливих нововведень Swift, в порівнянні з Objective-C, є можливість писати код і переглядати результати в реальному часі. Ми можемо внести деякі зміни в програмі і відразу побачити результат, так що нам не потрібно перекомпільовувати і перезапускати програму. Це має назву Interactive Playgrounds. Також тут можна використовувати Quick Look, який відображає графіку або список результатів, Timeline Assistant, який допомагає експериментувати з кодом UI (інтерфейсом користувача) або продивлятися повний цикл створення анімацій. Даний код можна перенести в основний проект. Починаючи з Xcode 7 можна створювати коментарі до коду, використовуючи форматований текст з жирним шрифтом або курсивом, маркований список, вбудовані зображення і посилання. Таким чином, Interactive Playgrounds розроблені для того, щоб зробити програмування більш інтерактивним і доступним [1].

Отже, Swift – це сучасні норми синтаксису, ефективне управління пам'яттю, висока швидкість роботи і інтерактивність. У Objective-C цього немає, але зате є надійність, база документації, прикладів, шаблонів і багато досвідчених програмістів. Підводячи підсумок, можна з повною впевненістю сказати, що Swift – це майбутнє. Все пізнається в порівнянні і, лише відпрацювавши певну кількість проектів, можна стверджувати, чи зручна мова, і чи можна на ній ефективно працювати.

Література:

1. ANON The Swift Programming Language (Swift 2.1) / ANON – Cupertino: Apple Inc., 2014. – 528 p.
2. Vandad Nahavandipoor iOS 8 Swift Programming Cookbook / Vandad Nahavandipoor – Boston: O'Reilly Media., 2014. – 902 p.

3. Офіційний сайт Stack Overflow. – Режим доступу: http://stackoverflow.com/research/developer-survey-2015#tech-super. – Дата доступу 13.03.2016.
4. Офіційний сайт блогу Primate Labs. – Режим доступу: http://www.primatelabs.com/blog/2014/12/swift-performance/. – Дата доступу 20.04.2016.
5. Офіційний сайт блогу Primate Labs. – Режим доступу: http://www.primatelabs.com/blog/2015/02/swift-performance-updated/. – Дата доступу 23.04.2016.
6. Офіційна документація мови програмування Swift. – Режим доступу: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/. – Дата доступу 12.05.2016.