Final project
CSC-421/621 Design and Organization of Programming Languages

Professor: Mark L Nelson

Group member:
Lijie Su
Sheng Kai Liao
Zehao Yu
Yuhao Li
Spring 2021

Introduction

Swift is a programming language used to create iOS and macOS applications. Swift combines the advantages of C and Objective-C while avoiding the drawbacks of C compatibility. Swift employs a stable programming model and adds a slew of new features to make programming easier, more flexible, and more enjoyable. Swift is based on the well-known and popular Cocoa and Cocoa Touch framework, and it alter the way software is designed.

Swift has been in production for a long time. In order to lay a stable foundation for Swift, Apple improved the parser, debugger, and system layout. It employs Automatic Reference Counting to facilitate memory management. To build and standardize the device stack, Foundation and Cocoa are used. Since Objective-C supports blocks, array syntax, and modules, the framework can easily support modern programming language technology. Because of this foundational work, developers are able to release such a new language for future Apple software development.

Among Objective-C programmers, Swift is well-known. It employs called Objective-C parameters and a dynamic object model, is compliant with Objective-C code, and can be seamlessly integrated into the current Cocoa framework. On this base, Swift has a slew of new features and supports procedural and object-oriented programming.

Swift has the benefit of being very user-friendly for beginners. It is the first programming language that not only follows industry expectations but is also expressive and interesting. It allows for code preview. This ground-breaking function enables programmers to run Swift code

and display the results in real time without having to compile and run the script. In this paper, we are going to discuss the features, implementations, syntax, semantics, and some other overviews of Swift.

Design of Swift

Swift is a powerful and intuitive programming language especially for macOS, iOS, watchOS, and Apple tvOS, but it also can be used to develop applications in other operating systems like Linux, Windows, and Android. Swift has concise syntax but strong expressiveness and includes modern features that developers love. Swift code is designed to be safe, and at the same time, software that runs lightning-fast can be developed. Chris Lattner, Doug Gregor, John McCall, Ted Kremenek, Joe Groff, and Apple Inc. designed Swift language to replace the position of Objective-C language in the iOS software development.

In comparison to Objective-C, Swift will prevent all forms of unsafe code. Variables will always be initialized before use, arrays and integers will be tested for overflow, memory will be handled automatically, and the feature of mandatory exclusive access to memory will also avoid many programming errors. The grammar has been changed so that developers can easily identify their target purpose. For example, a three-character keyword can define a variable var or constant let. Furthermore, Swift makes extensive use of value types, especially for popular types such as array and dictionary. This ensures that when developers copy this form of content, it will not be updated elsewhere.

Implementations of Swift

Swift is an open-source language everyone can contribute to it, and Apple Inc. responsible for updating and changing features in the Swift language. Since Swift develops and grows quickly, engineers often encounter problems as a result of rapid changes. Changing rapidly is the source of the issues. Many changes were made in the most recent version, making old code obsolete and uncompilable. This causes a slew of issues for developers because they must update and even rebuild their code.

While Swift is the primary language for designing iOS/macOS applications, it is capable of much more than just creating mobile/desktop user interfaces. Swift's latest growth is aimed at making it a common programming language. Swift is used to write device management software, conduct scientific analysis, write back-end applications, and so on, in addition to mobile apps. The most common application is to write back-end programs in Swift. Many Swift-written back-ends are now being gradually deployed in production environments, demonstrating that using Swift to write back-end programs is feasible and effective.

IBM's Kitura is a server-side Swift platform. It is currently being carried out in an open-source and free manner. The entire project is built on the Swift Package Manager architecture and can run on macOS and Linux. Perfect is another Swift implementation example. Perfect is a comprehensive and versatile toolbox, software framework system, and web application server

that runs on Linux, iOS, and macOS. Swift engineers will use the same approach on both the server and the client because the software framework has tailored a range of solutions for them to build lightweight, easy-to-maintain, flexible Web applications and other REST services. Software projects in multiple languages are created. There are several other Swift implementations that developers can use, indicating that Swift is a powerful and approachable popular language for engineers.

Communities of Swift

Because Swift is an open-source language which makes the communities more active than other non open-source programming languages. There are many communities for Swift. For instance, Swift.org is one of the most famous communities. Its sole mission is to create the best general-purpose programming language in the world. Both developers will work together to openly create the language, with contributions from everyone who wishes to participate. This paper explains how the Swift community is structured so that we can collaborate to bring exciting new features to Swift and make it accessible to many more developers across more platforms.

GitHub is also a good community for Swift developers. Github is more than just an online collaboration platform; it also allows developers to communicate with one another and collaborate on open-source projects. The programming project would be shared by developers or businesses, and everyone would be able to use, create, and discuss it. Github is a fantastic Swift group.

Comparison between Swift and Objective-C

Compared with Objective-C, Swift has better readability, maintainability and safety. For example, to better readability, Swift drops @ symbol which is used to define different types in Objective-C and is optional to add semicolons after the end of each line code in Swift. Furthermore, for access control, as many popular languages, Swift uses "." instead of square brackets ("[","]") in Objective-C. For representing the Boolean value, Swift define true and false values as "true" and "false" rather than "YES" and "NO" in Objective-C. Those changes make Swift more similar to the most popular languages, such as C/C++, python and Java.

The "main" function is not needed in Swift, a simple Hello world can be printed by simply type "println("Hello, World")". In addition, the new Swift language will still be compatible with Objective-C. Its environment is the core mechanism used by SwiftUI to propagate values down the view tree, that is, from the parent view to the child view tree it contains. SwiftUI uses the environment extensively, but users can also use it for their own purposes. A key feature of Objective C is its support for categories, methods that can be added to extend classes at runtime. The system is also broadly used as an organizational technique, which allows related code to be gathered into extensions like libraries.

In Swift, there are three collections: Array, Dictionary and Set collection. Each copes with the three collections in Objective-C: NSArray, NSDictionary and NSSet. However, there are still differences between Swift and Objective-C. In Swift, when collections are assigned as variable, constant or function parameter, a copy of the collection will be created so that the usage of those values will not affect the original collection.

“goto” is a handy function in Objective-C which allows programs to jump to the task with a certain label. Swift still preserved this function but with different usage. Similar to Java, Swift manipulates the labels with “Label statement” which allows the user to define a name for a function and it can be controlled by using “continue” and “break” in the function loop.

Objective-C and Swift still share the same enumerator functionality. However, Swift has simpler syntax to define each case. In addition, Swift enumerator allows us to define barcodes, QR codes and raw data which is very convenient to manipulate relevant data for developers.

Swift also is a type-safe language, which means that the language can help users figure out the types of values that the code can use. If part of the code requires String, type safety prevents you from passing it to Int by mistake. Likewise, type safety prevents you from accidentally passing optional strings to code that requires non-optional strings. Type safety can help you find and fix errors early in the development process. Moreover, by default, Swift does not expose pointers and other unsafe accessors, in contrast to Objective C, which uses pointers pervasively to refer to object instances. In this case, it is safer than objective C.

Swift provides various control flow statements, these include while loops to perform tasks multiple times by using “if”, “guard”, “switch” statements etc. to execute different code branches based on specific conditions; “and” statements such as break and continue to move the flow of execution to another point in the code. Swift also provides a for-in loop that allows you to easily traverse arrays, dictionaries, ranges, strings, and other sequences. In many languages like C, Swift's switch statement is much more powerful than its counterpart. Cases can match many different patterns, including interval matching, tuples, and coercion to a specific type. The matching value in the switch case can be bound to the temporary constant or variable used in the

case body, and the complex matching condition can be expressed by the where clause of each case.

Moreover, Swift contains several novel functions which are not included in Objective-C. First, Swift supports in-time compiling which allows developers to see what changes have been made from the last command immediately. And of course, Swift not only inherits the basic unary, binary, ternary and logic operator, but also includes new operators such as range, overflow and custom operator. Range operators using ellipsis "…" to define the range between the values, as example "0…5" means it only contains the value from 0 to 5. Also, you could set a condition in range operators such as "0…<5" to restrain the value range. Overflow operator is used to avoid overflow error. To implement it, you simply add '&' character before the arithmetic operators (+, -, *, /) so that the system will return the overflow and underflow value rather than throwing error. Custom operators allow developers to define their own operators, for example, we can create a new unary operator "+++" to represent the value += 3 by creating a prefix, infix or postfix function in Swift. According to these novelties, Swift provides developers with a more convenient, flexible and maintainable development environment.

Finally, in Objective-C, when the called function gets a nil input, the program will automatically stop the function process and continue others. However, in this case, it may create a difficulty for finding error or debugging. On the other hand, in Swift, we can create an optional type which has two status, one is that there is a value and other is that there is no value at all. While the optional type turns to nil, the system will call runtime crash as a predictable behavior for notification. It avoids the difficulties I just mentioned in Objective-C, and as a result brings better maintainability and security.

Swift code example

Code safe problems is the main aspect that will be expanded and several key instances could support the core idea to illustrate why Swift is safer than objective-C and C language. Swift developed numerous API and standard librarian to cover unreasonable, illegitimate, and harmful to the system errors since it was released. Some programmers would like to use unsafe points to avoid system errors, even more serious consequences that could cause the system to lose important data.

Using pointer's parameters among C, Objective-C and Swift is an efficient way to operate code. An unacceptable operation could cause damaging memory and losing data. For instance, the system will report an error when a pointer runs to a certain address with deallocated the storage location. But Swift provides all the low-level operations so that programmers do not need to fix the errors manually. The direct mapping between C and Swift pointers' type holds a guarantee of the security of data processing. According to Lorentey's (2020) speech, some typical instances might show the safe issue directly:

//C	//Swift
Const Pointee *	UnsafePointer<Pointee>
Pointee *	UnsafeMutablePointer<Pointee>
const void *, opaque pointers	UnsaferawPointer
void *, opaque pointers	UnsafeMutablePointer

// C

void process_integers(const int *start, size_t count);

// Swift

func process_integers(_ start: UnsafePointer<CInt>! , _ count Int)

In another word, statistics will be transferred to certain Swift's unsafe pointers for making sure that the data are safe. But it has limitations because these unsafe pointers could only keep bulk interface data secure. Specifically, in Swift language, to find a proper pointer and to use a static allocate way could maintain a variable to create a reasonable dynamic buffer. It is a method to gain the pointer from C.

// Swift

let start = UnsafeMutablepointer<CInt>.allocate(capacity: 4)

Then, using pointers' algorithms and initialization methods to make the buffer's elements to specific value.

// Swift

start.initialize(to:0)

(start+1).initialize(to:3)

(start+2).initialize(to:7)

C function could be called after all the initialize steps are finished.

// Swift

process_integers(start, 7)

Reinitialize and the buffer could be reallocated after the initialized function returns. In short, using unsafe pointers is an irreplaceable method to keep data safe when other language pointers transfer to Swift language. In addition, it could save important data from unnecessary operating mistakes to guarantee the Swift could run in a safe way.

Research Paper overview

In the paper Swift vs. Objective-C: A New Programming Language the authors compare the differences between Swift and Objective-C, the characteristics of Swift, and the novelties. Swift provides full compatibility with Objective-C which means old Objective-C projects to be compiled with Swift libraries, also, developers could insert Objective-C code files into Swift. Besides compatibility with old code, Swift has more useful features to help developers. For instance, Swift adds Optionals new special value type. Optionals are used to assign a type when the value could be different types or nil, or null in other languages. This feature makes Swift more secure than Objective-C. Another example is the keyword lazy, it is a property that makes the variable not calculated until its first use. Even though there are several new features added to Swift, Swift even becomes more concise than Objective-C. It simplified the syntax and made it could be used easier, but still needed the same number of lines and words to program. Furthermore, developers can be more comfortable with the developing process by using Swift. Since it has more abstraction level than Objective-C to facilitate the application development.

Reference

Apple, & Lorentey, K. (2020). Unsafe Swift-WWDC 2020-Videos. Apple Developer. Apple. https://developer.apple.com/videos/play/wwdc2020/10648/.

García, CG, Espada, JP, G-Bustelo, BCP, Manuel, J., & Lovelle, C. (2015). Swift vs. Objective-C: a new programming language. *International Journal of Artificial Intelligence and Interactive Multimedia, 3(3),* 74-81. https://doi.org/10.9781/ijimai.2015.3310