Kotlin For Science

Foundation of Kotlin Scientific Libraries

Independent Society of Knowledge

Last Edited July 12, 2024

1 Introduction

Kotlin, a statically typed language, has gained significant popularity since its release in 2016. Initially, Kotlin became prominent among Android developers after Google announced it as the preferred language for Android app development. However, the language's potential extends far beyond mobile development. The Independent Society of Knowledge has adopted Kotlin as its favored development language due to its readability and robust syntax.

Kotlin's syntax is concise and expressive, making it a high-level language that is accessible and easy to learn. Built on top of the Java Virtual Machine (JVM), Kotlin leverages the rich ecosystem and interoperability of Java, offering a performance advantage over alternatives like Python and Julia in many scenarios.

Despite these advantages, Kotlin has not been widely adopted in scientific programming and computational sciences. Several factors contribute to this gap:

1. **Relative Newness:** Kotlin is a relatively new language, whereas computational sciences have relied on established languages such as Fortran, C, and Python for decades.

2. **Library Ecosystem:** The current ecosystem of libraries for scientific computing in Kotlin is not as mature orextensive as those available for older languages.

3. **Community Focus:** The Kotlin community is primarily concentrated on app and software development, including Acdoird, Multi platform and Backend development. These areas are well-supported with packages and libraries, unlike scientific computing.

The *Kotlin for Science* Foundation aims to bridge this gap by promoting the use of Kotlin in scientific domains. Led by the community and maintained by the independent Society of Knowledge, this initiative will focus on enhancing Kotlin's capabilities for scientific computing and developing a rich ecosystem of libraries.

In this document, we investigate and propose solutions to adapt and integrate pre-developed libraries in C with Kotlin. This will provide a clear pathway for leveraging existing scientific tools with the Kotlin environment.

2 Kotlin Language

Kotlin is a modern, statically typed programming language that runs on the Java Virtual Machine (JVM) and can also be compiled to Javascript or native code. Since its release in 2016, Kotlin has quickly gained popularity due to its expressive syntax, modern features, and strong interoperability with Java.

Kotlin is fully interoperable with Java, which means you can call Java code from Kotlin. This allows developers to leverage the extensive ecosystem of Java libraries and frameworks while enjoying Kotlin's modern language features.

```
val list = ArrayList<String>()
list.add("Hello, World!")
println(list[0])
```

The syntax is designed to be concise and expressive, reducing boilerplate code and making it easier to read and write. This leads to higher productivity and fewer errors.

```
fun greet(name: String) = "Hello \$name !"
fun main() {
   println(greet("ISK")
}
```

Kotlin's type system is designed to eliminate the risk of null pointer exceptions, a common source of runtime errors in many languages. Nullable types and non-nullable types are distinct in Kotlin and the compiler enforces null safety.

```
var nonNullable: String = "This cannot be null."
var nullable: String? = "This can be null."

nullable = null // This is allowed
```

Kotlin also provides built-in support for coroutines, which simplify asynchronous programming. Coroutines allow you to write non-blocking code in a sequential style, making it easier to handle complex asynchronous tasks.

```
import kotlinx.coroutines.*

fun main() = runBlocking {
   launch {
      delay(1000L)
      println("World!")
   }
   println("Hello,")
}
```

Kotlin's interoperability with Java means that developers have access to a vast array of Java libraries and frameworks, which can be seamlessly integrated into Kotlin projects.

```
import org.apache.commons.lang3.StringUtils

fun main() {
   val str = "Hello Kotlin!"
   println(StringUtils.reverse(str))
}
```

As mentioned earlier, Kotlin main purpose in today's world is app and software development. Kotlin supports multiplatform development, allowing developers to write code that can run on multiple platforms, including JVM, Javascript, Android, IOS, and native platforms through Kotlin/Native.

This makes Kotlin an excellent choice for developing cross-platform applications.

Kotlin's Syntax is designed to be both concise and expressive, making it easier to read and write. This readability helps developers understand and maintain code more efficiently.

```
        data class Used(val name:String, val age:Int)

        fun main() {
           val user =User("Alice", 20)

           val doubleNumbers = listOf<Int>(1,2,3,4)
                    .map { it * 2 }
                    .forEach{
                    println(it)
                    }
        }
```

Kotlin's attributes make it a powerful language for various development scenarios, from Android and web applications to cross-platform projects and scientific computing. Its strong support for Java libraries, modern language features, and readable syntax contribute to its growing adoption in the developer community.

3 Binding C with Kotlin

4 Kotlin for Science Strategy

5 Development Plan

6 Costs

7 Promotion and Usage