CIENCIA DE LA COMPUTACIÓN

COMPARACIÓN DE LENGUAJES DE PROGRAMACIÓN

LENGUAJES DE PROGRAMACIÓN

José Armando Chávez Quijahuaman

Antonyjuan Miranda Yquira

Yahaira Valeria Gomez Sucasaca

Rafael Isaac Cano Guitton

7mo semestre

2021

"Los alumnos declaran haber realizado el presente trabajo de acuerdo a las normas de la Universidad Católica San Pablo."

Paradigma	Funcional	Imperativo	Orientado a objetos	Declarativo
Lenguaje de Programación	**Haskell**	**Pascal**	**Kotlin**	**Swift**
Legibilidad	Sí - La naturaleza de haskell hace que tengamos que dividir nuestros programas en pequeñas piezas y recurrir a la recursividad, lo cual hace de nuestro código bastante legible.	Sí - Es facilmente comprensible, ya que es un lenguaje de alto nivel: se parece más a un lenguaje natural hablado que al lenguaje de máquina.	Sí - Es un lenguaje conciso y la mayoria de sus palabras reservadas no tienen doble significado.	Sí - Se basa en una sintaxis que se ha reducido a lo esencial para una legibilidad óptima.
Facilidad de escritura	Sí - Es elegante y conciso, usa varios elementos de alto nivel por lo que los programas son usualmente cortos.	Sí - Es un lenguaje de sintaxis sencilla, muy estructurado y utiliza pocas líneas de código.	Sí - Porque es un lenguaje de alto nivel, tipado y el uso de comillas al final de cada opcional opcional.	Sí - Al ser un lenguaje de programación declarativo indicamos qué queremos hacer y esto permite que Swift sea amigable para programadores principiantes de aplicaciones iOS.
Confiabilidad	Sí - Debido a que haskell es un lenguaje tipado, la sintaxis y semántica te provee de herramientas suficientes para escribir código "correcto" previniendo errores inesperados.	Sí - Porque se caracteriza por ser un lenguaje de programación fuertemente tipado.	Sí - Es un lenguaje que proporciona seguridad y se comporta como es anunciado con los resultados esperados.	Sí - Ya que está enfocado al desarrollo de aplicaciones para iOS y macOS y brinda todas la herramientas necesarias.
Eficiencia	Sí - El compilador GHC llega a tener un desempeño comparable a C, además de el recolector de basura y localización automática de memoria en el heap.	Sí - Pascal puede implementarse en forma eficiente; la traducción se hace en el código de máquina ejecutable.	Sí - La implementación de aplicaciones de Kotlin es más rápida de compilar, liviana y evita que las aplicaciones aumenten de tamaño.	Sí - Es un lenguaje rápido y eficiente que proporciona información en tiempo real.
Facilidad de aprendizaje	No - Los lenguajes funcionales son especialmente difíciles de aprender, y en haskell en vez de aprender algunos conceptos complicados, se aprende y aplica varios conceptos simples. El mayor reto es acostumbrarse a escribir código usando uno o más de estos conceptos simples.	Sí - Ya que por la naturaleza del paradigma imperativo, hace que el código se lea como unas instrucciones paso a paso.	Sí - Porque es un lenguaje de alto nivel, tipado y el uso de comillas al final de cada sentencia opcional .	Sí - De hecho es una de la razones por la cuales se está haciendo popular cada vez más. Y es que soluciona errores comunes que se producen al usar otros lenguajes si no tenemos sumo cuidado.
Ortogonalidad	Sí - A pesar de ser un lenguaje no-escricto, hace uso del heap para hacer la evaluación de las variables cuando se necesite.	Sí - Tiene una secuencia claramente definida de instrucciones para un ordenador.	Sí - Los paquetes son ortogonales de los módulos. Un módulo puede contener muchos paquetes y un solo paquete se puede distribuir en varios módulos.	No - Ya que no se puede combinar en una sola instrucción diversas características del lenguaje.
Reusabilidad	Sí - En Haskell se puede crear funciones que pueden trabajar con valores de cualquier tipo que respeten cierta forma o diseño.	Sí - Permite dividir un programa en módulos (procedimientos y funciones) que luego pueden ser reutilizados.	Sí - Kotlin utiliza funciones de orden superior, se puede utilizar funciones como parámetros y tambien utiliza expresiones lambda.	Sí - Pero es más complicado y puede llevar más tiempo.
Modificalidad	Sí - De hecho haskell es un lenguaje diseñado para ser altamente modificable y mantenible.	No - Ya que la ejecución no esta claramente separada de la programación, se pueden producir efectos colaterales o errores no deseados si se hacen modificaciones.	Sí - Utiliza modificadores de acceso o de visibilidad que se utilizan para definir el alcance de una clase, función, campo, interfaz, etc.	No - Ya que el cambio en alguna parte puede traer errores.
Portabilidad	No - Portabilidad entre sistemas operativos (Windows, GNU/Linux) es relativamente simple en haskell gracias al compilador GHC pero entre arquitecturas es muy difícil.	No - Frente a otros lenguajes tiene el inconveniente de ser menos portable.	Sí - Soporta programación multiplataforma y es uno de sus principales beneficios ya que reduce el tiempo de escribir o mantener el codigo para diferentes plataformas.	No - Las bibliotecas de Switf solo están integradas en iOS, macOS, watchOS y tvOS.