Paradigmas de programación

Trabajo Práctico: Comparación de Lenguajes de Programación

**Lenguajes analizados:** C (Imperativo), Haskell (Declarativo), Java (Multiparadigma)

Alumnos:

* Díaz, Matías Hernán 35993466
* Perez, Adrian DNI

Profesor: Gastón Aguilera.

Contenido

[**1. Introducción** 3](#_Toc199541535)

[**2. Lenguaje C (Imperativo)** 3](#_Toc199541536)

[**3. Lenguaje Haskell (Declarativo)** 3](#_Toc199541537)

[**4. Lenguaje Java (Multiparadigma)** 4](#_Toc199541538)

[5. Resolución de un problema 4](#_Toc199541539)

[**6. Comparación general** 6](#_Toc199541540)

[7. Conclusión 6](#_Toc199541541)

[8. Bibliografía 7](#_Toc199541542)

**1. Introducción**

Los lenguajes de programación se pueden clasificar según el paradigma que utilizan. Dos de los principales paradigmas son el **imperativo** y el **declarativo**.

* **Lenguaje imperativo:** describe una secuencia de pasos que el procesador debe ejecutar. Se enfoca en cómo se hace algo. Ejemplos: C, Pascal.
* **Lenguaje declarativo:** se centra en qué resultado se desea, sin especificar el procedimiento para conseguirlo. Ejemplos: Haskell, SQL, Prolog.

El lenguaje Java, aunque inicialmente imperativo y orientado a objetos, ha incorporado características funcionales desde su versión 8.

Este trabajo presenta una comparación detallada entre C, Haskell y Java, evaluando su estructura, usos, ventajas, desventajas y adecuación a diferentes tipos de problemas.

**2. Lenguaje C (Imperativo)**

El lenguaje C representa el paradigma imperativo en su forma más clásica. Fue diseñado para permitir a los programadores un control total sobre el flujo de ejecución y el uso de memoria, por lo que se emplea ampliamente en sistemas operativos y software de bajo nivel. C se caracteriza por el uso intensivo de variables mutables, bucles y estructuras condicionales que definen explícitamente los pasos a seguir por el procesador.

Entre sus principales ventajas se destaca su **eficiencia**, lo que lo convierte en una excelente elección cuando el rendimiento es una prioridad. Además, la **portabilidad** del lenguaje ha contribuido a su adopción en diversas plataformas. Sin embargo, su **bajo nivel de abstracción** exige al programador ocuparse de aspectos técnicos complejos, como la gestión manual de memoria, lo que puede aumentar la **probabilidad de errores** y dificultar el mantenimiento del código. Asimismo, resulta **poco expresivo** en comparación con lenguajes de más alto nivel, lo que puede complicar la implementación de estructuras complejas.

**3. Lenguaje Haskell (Declarativo)**

Haskell es un lenguaje funcional puro que representa de forma clara el paradigma declarativo. En él, la lógica del programa se expresa mediante funciones puras y estructuras inmutables, sin necesidad de detallar los pasos intermedios para alcanzar un resultado. Gracias a la evaluación perezosa, las expresiones solo se calculan cuando son necesarias, lo cual permite optimizaciones significativas en muchos contextos.

Una de las mayores fortalezas de Haskell radica en su capacidad para **expresar conceptos complejos de forma clara y concisa**. La ausencia de efectos secundarios y el uso de funciones puras **facilitan la depuración**, la verificación formal de programas y la **concurrencia segura**. No obstante, su **curva de aprendizaje es pronunciada**, especialmente para quienes provienen de lenguajes imperativos. Además, aunque cuenta con una comunidad activa, su adopción en la industria es limitada, lo cual puede dificultar su integración en proyectos productivos a gran escala.

**4. Lenguaje Java (Multiparadigma)**

Java es un lenguaje **multiparadigma** que combina el enfoque imperativo con la programación orientada a objetos. Desde la versión 8, ha incorporado elementos de la programación funcional, como **expresiones lambda** y estructuras de tipo stream, lo cual permite escribir código más expresivo y declarativo sin abandonar la semántica imperativa tradicional.

Su éxito en el desarrollo de aplicaciones empresariales se debe a su robustez, portabilidad y a un extenso ecosistema de bibliotecas, marcos de trabajo y herramientas. Además, la máquina virtual de Java (JVM) garantiza que el código pueda ejecutarse en múltiples plataformas sin modificaciones. Sin embargo, una de sus principales críticas es la verbosidad del código, lo que puede hacer que soluciones simples requieran implementaciones extensas. Asimismo, aunque Java gestiona automáticamente la memoria, su rendimiento puede verse comprometido frente a lenguajes de bajo nivel como C.

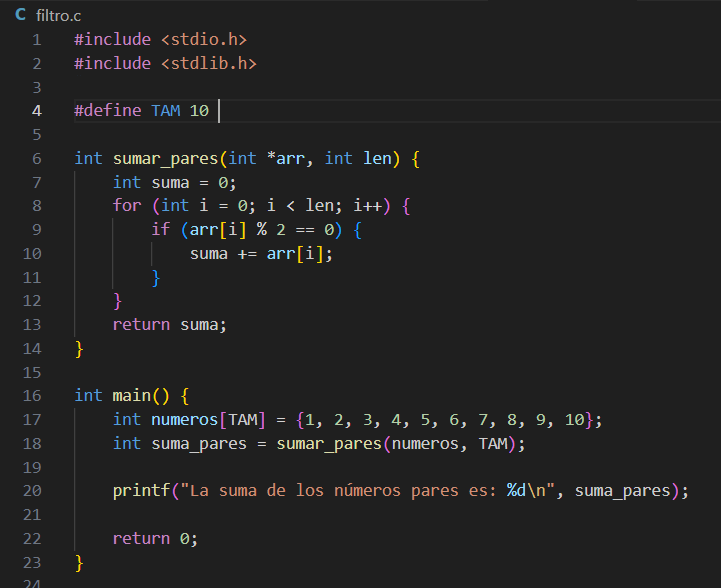
## 5. Resolución de un problema

Se presenta un caso práctico para ver como lo resolvería cada lenguaje acorde a su paradigma.

Problema: filtrar una lista de números para obtener solo los pares y luego calcular la suma de esos números pares.

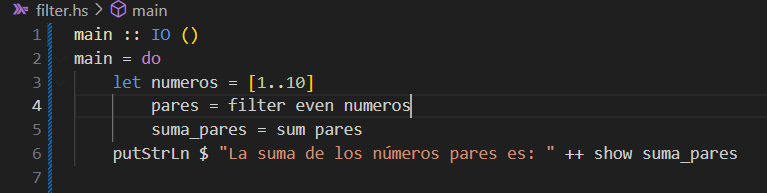
**Lista = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]**

* En C

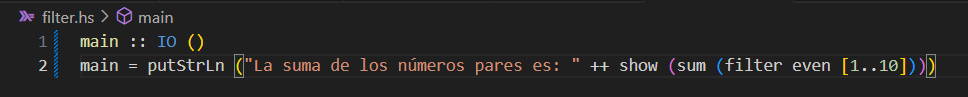


Se detalla cada paso del proceso (iterar, verificar, acumular). El foco está en el cómo se realiza la tarea. Se utilizan variables mutables para almacenar el resultado intermedio.

* En Haskell

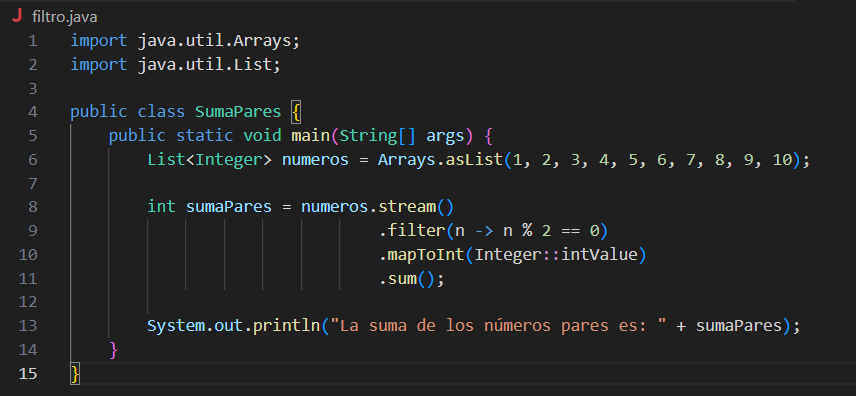


Pudiéndose reducir aún más.



Se describe qué se quiere obtener (la suma de los pares) utilizando funciones que transforman datos. No se especifican los pasos de la iteración o la acumulación. Se favorecen las listas inmutables y las funciones puras.

* En Java



Java (Multiparadigma): Utiliza un enfoque más declarativo con Streams, permitiendo expresar la lógica de filtrado y suma de manera concisa. Aunque Java sigue siendo fundamentalmente imperativo, las características funcionales permiten un estilo más cercano al declarativo para ciertas operaciones.

**6. Comparación general**

La comparación entre C, Haskell y Java permite observar cómo cada paradigma resuelve problemas de programación desde perspectivas distintas. C se destaca por ofrecer un control detallado sobre el hardware y el rendimiento, a costa de una mayor complejidad en la escritura y mantenimiento del código. Haskell, en cambio, prioriza la claridad y la expresividad, lo cual es ideal para programas en los que la corrección lógica es fundamental, aunque puede no ser el más adecuado para tareas que requieran interacciones directas con el sistema. Java, por su parte, ofrece un equilibrio entre abstracción y eficiencia, lo que le permite adaptarse a una gran variedad de contextos, desde sistemas empresariales hasta aplicaciones móviles y de escritorio.

## 7. Conclusión

Cada lenguaje refleja un enfoque diferente para resolver problemas:

* C es ideal para sistemas donde el control del hardware es crítico.
* Haskell promueve la claridad y corrección matemática del código.
* Java se adapta a entornos empresariales con necesidades de escalabilidad.

La elección del lenguaje depende del problema, el contexto del proyecto y la experiencia del equipo.

## 8. Bibliografía

1. Hutton, G. (2016). Programming in Haskell. Cambridge University Press.
2. Kernighan, B. W., & Ritchie, D. M. (1988). The C Programming Language. Prentice Hall.
3. Oracle. (2024). Java Language Specification. <https://docs.oracle.com/javase/specs/>
4. Bird, R. (2014). Introduction to Functional Programming using Haskell. Prentice Hall.
5. Stack Overflow Developer Survey (2024). <https://insights.stackoverflow.com/survey>