Investigación sobre Lenguajes de Programación

Aprendiz

Harold Felipe Hernandez Reyes

SENA

Técnico en Programación de Software

Ficha: 3233827

Instructor: Carlos Andres Figueredo Rodriguez

Funza, Cundinamarca

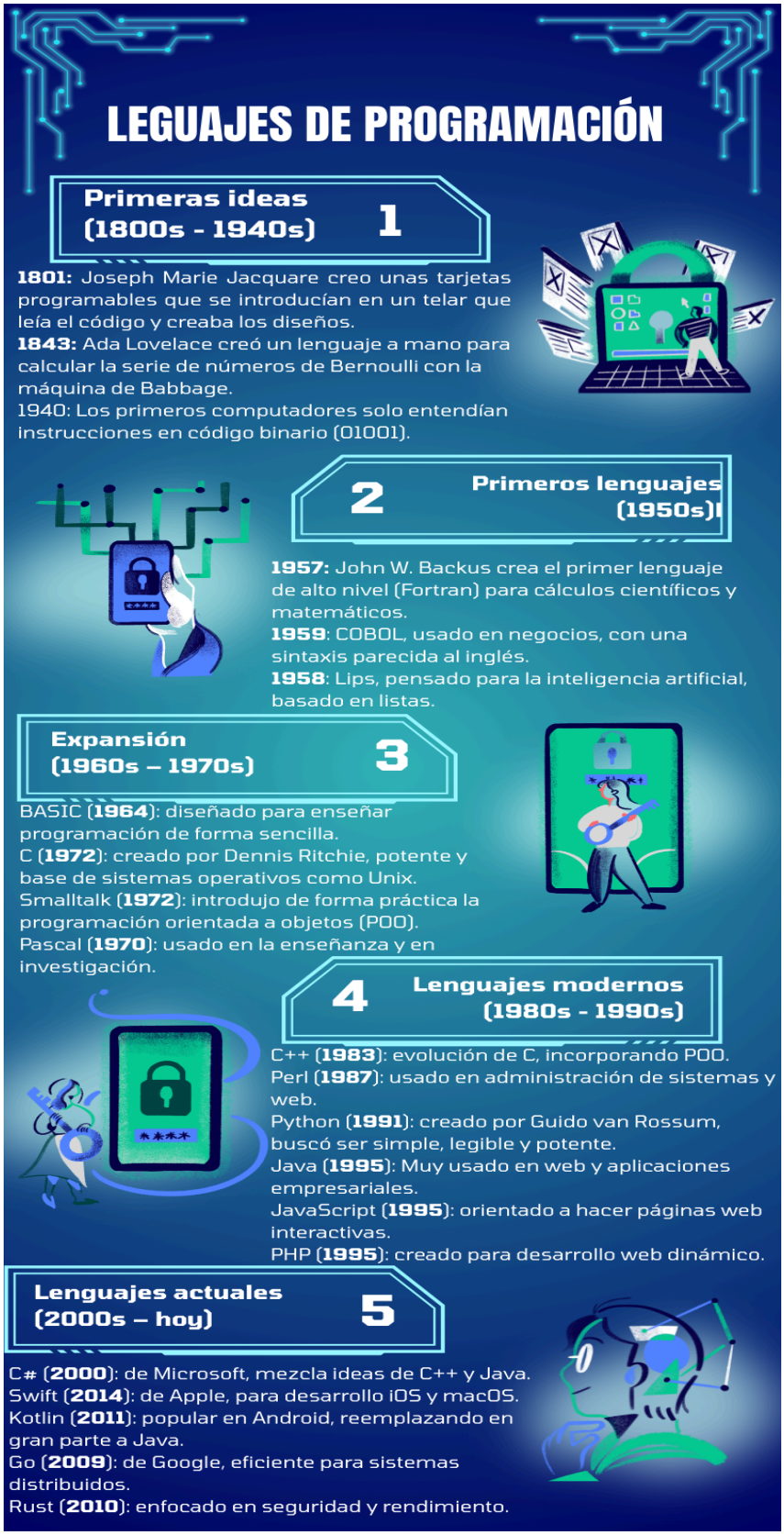
27 de agosto de 2025

Contenido

1. Definición propia

Es el idioma con el que un programador se comunica y le da instrucciones a una computadora para que realice diferentes tareas, como resolver cálculos, manejar datos, entre otros. Cada lenguaje de programación maneja sus reglas y su propia estructura que se deben seguir detalladamente para que el programa haga específicamente lo que se le indique.

2. Historia (infografía)



3. Diferencias entre lenguajes

Lenguaje natural: Es el que usamos los humanos para comunicarnos, por ejemplo, español, inglés. Surgió a lo largo de la historia y evolución de los humanos. Se rige a partir de reglas gramaticales y sus conceptos pueden tener varios significados según el contexto. Lo entienden principalmente los humanos y tiene como finalidad comunicar ideas, emociones o información.

Lenguaje de programación: Es usado para dar instrucciones a una computadora. Se creo por los humanos para automatizar procesos con ayuda de las computadoras. Se rige por reglas estrictas, con un pequeño error se generan fallos en el programa. Lo entienden los humanos los cuales lo escriben y las computadoras las cuales lo ejecutan y tiene como finalidad controlar y dar órdenes precisas a una computadora.

4. Clasificación

Según su nivel:

Bajo nivel: Son rápidos y eficientes pero difíciles de aprender y usar, usan un lenguaje binario.

Alto nivel: Son fáciles de leer y escribir.

Según su paradigma:

Imperativos: Se enfocan en dar instrucciones paso a paso.

POO: Organizan el código en clases y objetos (representan cosas del mundo real).

Funcionales: Basados en funciones matemáticas.

Según su compilación:

Compilados: Todo el código se traduce de una vez a lenguaje máquina y luego se ejecuta. Son muy rápidos.

Interpretados: el código se ejecuta línea por línea. Son más lentos, pero fáciles de probar y depurar.

Según su propósito:

De propósito general: sirven para crear todo tipo de programas (aplicaciones web, juegos, sistemas, etc.).

De propósito específico: diseñados para un área concreta.

Según su tipado

Estático: El tipo de cada variable se define antes de ejecutar el programa, lo que reduce errores.

Dinámico: El tipo de la variable se determina mientras corre el programa.

Fuertemente tipados: No permiten mezclar tipos de datos sin convertirlos.

Débilmente tipados: Permiten combinar tipos sin problema, aunque a veces genera errores extraños.

5. **Características**

Simplicidad: Clara y fácil de aprender, aunque sin perder potencia.

Capacidad: Ofrece herramientas suficientes para resolver diferentes tipos de problemas, o adaptarse bien a un área específica.

Estructuración: Facilita organizar el código de forma ordenada y con menos errores.

Compacidad: Permite expresar operaciones en pocas líneas sin repetir información innecesaria.

Principio de localidad: Aprovecha el uso repetido de ciertos datos o zonas de memoria para mejorar el rendimiento.

6. **Paradigmas**

Programación imperativa/procedural

Programación orientada a objetos (POO)

Programación Funcional

Programación Declarativa:

6.1 Ejemplos

Imperativo: C

Procedural: Pascal

Orientado a objetos (POO): Java

Funcional: Haskell

Lógico: Prolog

Declarativo: SQL

Estructurado: C

Reactivo: Elm

Concurrente / Paralelo: Go, Erlang

Orientado a aspectos: AspectJ

Basado en eventos: JavaScript

Dirigido por datos: R, MATLAB

Orientado a servicios: Jolie

7. ¿Para qué sirven?

Los lenguajes de programación sirven para comunicarnos con las computadoras y darles instrucciones de forma clara y precisa. Con ellos se pueden crear programas y aplicaciones de todo tipo, desde algoritmos, páginas web y videojuegos, hasta apps móviles y software empresarial. También permiten automatizar tareas repetitivas, manejar y analizar grandes cantidades de datos, e incluso controlar dispositivos y robots. En pocas palabras, son la herramienta que transforma nuestras ideas en acciones que una computadora puede ejecutar para resolver problemas o facilitar la vida diaria.

8. Diferencias de nivel

Los lenguajes de bajo nivel trabajan de manera muy detallada y específica, controlando aspectos técnicos como la memoria o los registros del procesador. Esto los hace más rápidos y eficientes, pero más complicados de programar.

Los lenguajes de alto nivel permiten enfocarse en la lógica del problema sin preocuparse por esos detalles técnicos, lo que facilita el aprendizaje y el desarrollo de programas más grandes y complejos.

8.1 Ejemplos

Lenguajes de bajo nivel:

Hacer que un semáforo cambie de luces en una ciudad.

Programar la tarjeta de un cajero automático para que lea la clave y libere dinero.

Lenguajes de alto nivel:

Crear una aplicación de mensajería como WhatsApp.

Diseñar una tienda en línea como Amazon o Mercado Libre.

9. Tabla de Lenguajes de Programación

Lenguaje	Creador(es)	Año de creación	Año de mayor popularidad	Paradigma(s) predominante(s)	Logo
Visual Basic	Alan Cooper / Microsoft	1991	1990s, especialmente primeros 2000s	Orientado a objetos, educativo	
Java	James Gosling (Sun Microsystems)	1995	Finales de los 1990s y hasta hoy	Orientado a objetos	
PHP	Rasmus Lerdorf	1995	Finales de los 1990s, web dinámico	Imperativo, orientado a objetos, scripting	
Python	Guido van Rossum	1989 (implementación)	2000s–2020s (muy popular)	Multi-paradigma (OO, funcional, imperativo)	
C++	Bjarne Stroustrup	1983	1990s–2000s	Orientado a objetos, genérico, imperativo	
JavaScript	Brendan Eich	1995	2000s–2020s (web)	Multi-paradigma, funcional, orientado a objetos	
Objective-C	Brad Cox y Tom Love (50s), formalizado 1980s	Años 1980s	2000s (Apple macOS iOS)	Orientado a objetos	
Swift	Chris Lattner (Apple)	2014	Desde 2014, en auge	Multi-paradigma (funcional, OO)	
Ruby	Yukihiro "Matz" Matsumoto	1995	Finales de los 1990s–2000s (Rails)	Multi-paradigma (OO, funcional)	
Perl	Larry Wall	1987	1990s–2000s (CGI scripts)	Imperativo, scripting, funcional	
TypeScript	Anders Hejlsberg (Microsoft)	2012	2016 en adelante (web moderno)	Orientado a objetos, tipado estático	
Dart	Google (Lars Bak, Kasper Lund)	2011	A partir de 2018 (Flutter)	Orientado a objetos	
Go (Golang)	Robert Griesemer, Rob Pike, Ken Thompson	2009	2010s–2020s (sistemas concurrentes)	Imperativo, concurrente	
Kotlin	JetBrains (Dmitry Jemerov)	2011 (publicado 2012)	2017 (Android oficial)	Orientado a objetos, funcional	
Rust	Graydon Hoare (Mozilla)	2010	Segunda mitad de 2010s–2020s	Multi-paradigma (sistemas, funcional)	
R	Ross Ihaka & Robert Gentleman	1993	2000s–2020s (estadística, data science)	Funcional, vectorizado	
Delphi	Anders Hejlsberg (Borland)	1995	Finales de 1990s–2000s	Orientado a objetos (Object Pascal)	
C#	Anders Hejlsberg (Microsoft)	2000	2000s–2020s (aplicaciones .NET)	Orientado a objetos, funcional	
Pascal	Niklaus Wirth	1970	1970s–1980s (educación, compilers)	Procedural estructurado	

10. Tipos de Software de Programación

Editores / IDE: Programas donde se escribe el código.

Compiladores: Traducen todo el programa para que la computadora lo pueda ejecutar.

Intérpretes: Leen y ejecutan el código paso a paso, sin traducir todo de una vez.

Ensambladores: Convierten instrucciones muy básicas (ensamblador) en código que entiende el procesador.

Depuradores: Sirven para encontrar y corregir errores en el programa.

Herramientas de construcción: Ayudan a automatizar tareas como compilar y crear el programa final.

Control de versiones: Guardan los cambios del código y permiten trabajar en equipo sin perder nada.

11. Completa la frase

A. Las computadoras procesan los datos bajo el control de conjuntos de instrucciones llamados **programas**.

B. A los programas que traducen programas escritos en un lenguaje de alto nivel a lenguaje máquina se les llama **compiladores**.

C. La versión de C conocida como **ANSI C** que recientemente fue estandarizada a través de la American National Standards Institute.

D. El lenguaje **Pascal** fue desarrollado por Wirth para la enseñanza de la programación estructurada.

E. El departamento de defensa de los Estados Unidos desarrolló el lenguaje Ada con una capacidad llamada **paralelismo**, la cual permite a los programadores especificar la realización de varias tareas en paralelo.

F. La **programación** de **bajo** nivel son los que te permiten hacer modificaciones en el hardware.

G. PHP, Hypertext Preprocessor: es un lenguaje orientado hacia el diseño Backend de páginas web. Es muy adecuado porque es

12. Cuadro comparativo (motores de bases de datos)

Tipo	Motores	Características principales	Ventajas	Desventajas
Relacionales (SQL)	MySQL, PostgreSQL, Oracle, SQL Server	Basados en tablas (filas y columnas), usan SQL para consultas, mantienen relaciones entre datos.	- Muy estructurados. - Fáciles de entender. - Integridad y consistencia de datos.	- Poca flexibilidad con datos no estructurados. - Menor rendimiento con grandes volúmenes de datos distribuidos.
No relacionales (NoSQL)	MongoDB, Cassandra, Redis, CouchDB	Basados en documentos, claves-valor, grafos o columnas; no usan SQL tradicional.	- Flexibilidad para datos variados. - Escalan fácilmente. - Buen rendimiento en grandes volúmenes de datos.	- Menor estandarización. - Pueden carecer de integridad estricta. - Más difíciles de aprender para usuarios acostumbrados a SQL.

13. Principios POO

Adaptabilidad: Un sistema debe poder ajustarse fácilmente a cambios en los requisitos o el entorno sin necesidad de rehacer todo el código.

Extensibilidad: Permite agregar nuevas funciones o características sin modificar lo que ya funciona, solo ampliando el sistema.

Mantenibilidad: El código debe ser fácil de leer, corregir errores y actualizar con el paso del tiempo.

Reusabilidad: El software debe permitir reutilizar partes del código en otros programas o proyectos, ahorrando tiempo y esfuerzo.

Desempeño: El sistema debe responder de manera rápida y consumir los recursos de forma adecuada.

Escalabilidad: Capacidad del software para seguir funcionando bien cuando aumenta la cantidad de usuarios, datos o procesos.

Confiabilidad: El sistema debe ser estable y producir resultados correctos bajo diferentes condiciones, sin fallos inesperados.

Eficiencia: Lograr que el sistema realice las tareas con el menor uso posible de tiempo, memoria y recursos.

14. Tecnologías o servicios en la nube

Servicio	Propietario	Qué ofrece	Configuración	Costo
AWS	Amazon	Soporta muchos lenguajes (Java, Python, PHP…), bases de datos (MySQL, MongoDB…) y Linux/Windows	Desde 1 CPU y 1 GB RAM hasta servidores muy grandes	Pago según uso, tiene 12 meses gratis limitado
GCP	Google	Lenguajes: Python, Node.js, Java… BD: Cloud SQL, Firestore	Desde 1 CPU y 0.6 GB RAM, escalable	Pago según uso, algunas cosas gratis
Azure	Microsoft	Lenguajes: .NET, Python, PHP… BD: SQL Server, MySQL	Desde 1 CPU y 1 GB RAM hasta servidores grandes	Pago según uso, 12 meses gratis con límite
Oracle	Oracle	Lenguajes: Java, PHP, Python… BD: Oracle DB, MySQL	Desde 1 CPU y 1 GB RAM	Algunas cosas gratis siempre, resto pago
DigitalOcean	DigitalOcean	Lenguajes: Node.js, Python, PHP… BD: MySQL, PostgreSQL	Desde 1 CPU y 1 GB RAM	Planes desde $5/mes, crédito inicial gratis
IBM Cloud	IBM	Lenguajes: Java, PHP, Python… BD: Db2, MongoDB	Desde 1 CPU y 2 GB RAM	Algunas cosas gratis, resto pago
Netlify	Netlify Inc.	Ideal para sitios web y apps frontend (React, Vue, Angular)	Usa servidores cloud escalables, no configurables	Gratis limitado, planes pagos desde $19/mes