

Practica 1 Historia, evolución y características de Leng. de Programación

Buenas, lo unico que tengo para decir de esta practica, es que juega de titular chatgpt.

1)

1951 - 1955: Lenguajes tipo assembly

- **Características nuevas:** Primeros lenguajes de bajo nivel, cercanos al lenguaje máquina.
 - **Lenguaje destacado:** Lenguajes ensambladores, como el de la IBM 701.
 - **Innovación clave:** Permitieron escribir código en símbolos mnemotécnicos en lugar de código binario.
-

1956 - 1960: FORTRAN, ALGOL 58, ALGOL 60, LISP

- **Características nuevas:**
 - **FORTRAN:** Introducción del cálculo matemático de alto nivel y compilación eficiente.
 - **ALGOL 58 / ALGOL 60:** Introducción de estructuras de control bien definidas (bloques y sentencias estructuradas).
 - **LISP:** Primer lenguaje funcional, con manejo de listas y recursividad.
-

1961 - 1965: COBOL, ALGOL 60, SNOBOL, JOVIAL

- **Características nuevas:**
 - **COBOL:** Uso de sintaxis similar al inglés, orientado a negocios y procesamiento de datos.
 - **ALGOL 60:** Introducción del concepto de paso de parámetros por valor y por referencia.
 - **SNOBOL:** Manejo avanzado de cadenas de texto.
 - **JOVIAL:** Extensión de ALGOL para aplicaciones militares.
-

1966 - 1970: APL, FORTRAN 66, BASIC, PL/I, SIMULA 67, ALGOL-W

- **Características nuevas:**
 - **APL:** Uso de notación matemática concisa y operaciones vectoriales.
 - **FORTRAN 66:** Estandarización de FORTRAN.
 - **BASIC:** Facilidad de uso para principiantes en programación.
 - **PL/I:** Lenguaje híbrido que combinó características de FORTRAN, COBOL y ALGOL.
 - **SIMULA 67:** Introducción de la programación orientada a objetos (POO).
 - **ALGOL-W:** Mejoras en estructuras de datos y eficiencia.
-

1971 - 1975: Pascal, C, Scheme, Prolog

- **Características nuevas:**
 - **Pascal:** Introducción de la programación estructurada con tipado fuerte.
 - **C:** Creación del lenguaje que influenciaría la mayoría de los lenguajes modernos.
 - **Scheme:** Implementación minimalista del paradigma funcional.
 - **Prolog:** Primer lenguaje de programación lógica.
-

1976 - 1980: Smalltalk, Ada, FORTRAN 77, ML

- **Características nuevas:**
 - **Smalltalk:** Expansión de la programación orientada a objetos.
 - **Ada:** Seguridad y modularidad en sistemas críticos.
 - **FORTRAN 77:** Introducción de estructuras de control modernas.
 - **ML:** Tipado polimórfico y programación funcional avanzada.
-

1981 - 1985: Smalltalk 80, Turbo Pascal, Postscript

- **Características nuevas:**
 - **Smalltalk 80:** Popularización de la POO con interfaces gráficas.
 - **Turbo Pascal:** Introducción de compilación rápida y entornos de desarrollo integrados.

- **Postscript:** Lenguaje de descripción de páginas para impresión avanzada.
-

1986 - 1990: FORTRAN 90, C++, SML

- **Características nuevas:**
 - **FORTRAN 90:** Introducción de programación modular y arrays dinámicos.
 - **C++:** Introducción de POO en un lenguaje eficiente y flexible.
 - **SML:** Avance en la inferencia de tipos en lenguajes funcionales.
-

1991 - 1995: TCL, PERL, HTML

- **Características nuevas:**
 - **TCL:** Lenguaje de scripting embebible.
 - **PERL:** Manejo avanzado de cadenas y expresiones regulares.
 - **HTML:** Creación de la estructura básica de las páginas web.
-

1996 - 2000: Java, Javascript, XML

- **Características nuevas:**
 - **Java:** Programación multiplataforma con la JVM y gestión automática de memoria.
 - **Javascript:** Interactividad en navegadores web.
 - **XML:** Estándar para el intercambio estructurado de datos.

2)

Historia del lenguaje Java ☕

Java fue desarrollado por **James Gosling** y su equipo en **Sun Microsystems** a principios de los años 90. Su historia se puede dividir en varias etapas clave:

Expansión y crecimiento (1996 - 2006)

- En **1996**, Sun Microsystems lanzó **Java 1.1**, mejorando la gestión de eventos y agregando soporte para JDBC (conexión a bases de datos).

- En **1998**, con **Java 2 (J2SE 1.2)**, se introdujo el **Swing** para interfaces gráficas y la división en ediciones:
 - **J2SE** (Java 2 Standard Edition): Aplicaciones de escritorio y generales.
 - **J2EE** (Java 2 Enterprise Edition): Aplicaciones empresariales y web.
 - **J2ME** (Java 2 Micro Edition): Aplicaciones para dispositivos móviles y embebidos.
- En **2004**, **Java 5** introdujo grandes mejoras como **genéricos**, **autoboxing** y el **bucle for** mejorado.

Era de Oracle y modernización (2010 - Actualidad)

- En **2010**, **Oracle** adquirió **Sun Microsystems**, convirtiéndose en el dueño de Java.
- En **2011**, Java 7 introdujo mejoras en rendimiento y sintaxis, como el **try-with-resources**.
- **Java 8 (2014)** trajo cambios revolucionarios con **expresiones lambda**, la API de **Streams** y el **nuevo API de fechas y tiempos**.
- Desde **Java 9 (2017)**, Oracle adoptó un modelo de lanzamientos cada **6 meses**.
- **Java 17 (2021)** se convirtió en una **versión LTS (Long-Term Support)**, optimizando seguridad y rendimiento.
- Actualmente, Java sigue evolucionando con versiones más eficientes, enfocadas en rendimiento y compatibilidad con la nube.

Importancia de Java






- Java es ampliamente usado en **desarrollo web**, **móvil (Android)**, **empresarial** y **cloud**.
- Su comunidad y ecosistema siguen siendo muy fuertes, con frameworks como **Spring**, **Jakarta EE** y **Quarkus**.

3)

Atributos de un buen lenguaje de programación

Un buen lenguaje de programación debe cumplir con varias características que faciliten su uso y mantenimiento. Aquí están algunos atributos clave con ejemplos:

- **Legibilidad** 📖
 - El código debe ser fácil de entender y estructurar.
 - **Ejemplo: Python** destaca por su sintaxis clara y cercana al lenguaje natural.
- **Simplicidad** 🧠💻
 - Debe evitar características innecesarias o redundantes.
 - **Ejemplo: Go** elimina elementos complejos como herencia múltiple y reduce la sintaxis.

- **Ortogonalidad** 
 - Permite combinar elementos del lenguaje sin restricciones inesperadas.
 - **Ejemplo: LISP** permite aplicar funciones de manera consistente a diferentes estructuras de datos.
- **Expresividad** 
 - Permite escribir código de manera concisa y eficiente.
 - **Ejemplo: Ruby**, con su sintaxis elegante y flexible.
- **Portabilidad** 
 - El código debe poder ejecutarse en diferentes plataformas sin cambios.
 - **Ejemplo: Java**, gracias a la **JVM** (*Write Once, Run Anywhere*).
- **Eficiencia** 
 - Debe ejecutar programas rápidamente y usar recursos de manera óptima.
 - **Ejemplo: C**, por su cercanía al hardware y optimización.
- **Seguridad** 
 - Debe evitar errores comunes y vulnerabilidades.
 - **Ejemplo: Rust**, con su sistema de gestión de memoria sin necesidad de un recolector de basura.

4)

Elijo JAVA y C#, son lenguajes muy parecidos en cuanto a sintaxis y algunos otros aspectos.

◆ Java

- **Aritméticas:** `int resultado = 5 + 3;`
- **Lógicas:** `boolean esMayor = (a > b) && (b < c);`
- **Condicionales (ternarias):** `String mensaje = (edad >= 18) ? "Mayor de edad" : "Menor de edad";`
- **Lambdas (desde Java 8):** `Function<Integer, Integer> cuadrado = x -> x * x;`
- **De instancia:** `boolean esString = obj instanceof String;`

◆ C#

- **Similares a Java** (aritméticas, lógicas y condicionales).
- **Expresiones lambda:** `(x) => x * x;`
- **Expresiones LINQ** (consultas sobre colecciones):

Característica	Java	C#
Programación orientada a objetos	Sí (clases, interfaces, herencia, polimorfismo)	Sí (con características más avanzadas como interfaces explícitas)
Espacios de nombres (namespaces)	Usa <i>packages</i> para organizar clases	Usa <code>namespace</code> para organizar clases
Modularización	Desde Java 9 con modulos	Desde .NET con ensamblados
Manejo de excepciones	<code>try-catch-finally</code> , <code>throw</code>	<code>try-catch-finally</code> , <code>throw</code>
Estructuras de concurrencia	<code>Thread</code> , <code>ExecutorService</code> , <code>CompletableFuture</code>	<code>Task</code> , <code>async/await</code> , <code>Parallel</code>
Inyección de dependencias	Spring Boot	.NET Core

Atributo	Java	C#	Justificación
Legibilidad 📖	✅	✅	Ambos tienen sintaxis clara y bien estructurada.
Simplicidad 🤖💻	⚠️	✅	Java puede requerir más código (boilerplate) para ciertas tareas; C# tiene más atajos.
Ortogonalidad 🔄	✅	✅	Ambos permiten combinar estructuras sin conflictos.
Expresividad 📝	✅	✅	Lambdas, streams (Java) y LINQ (C#) facilitan el código conciso.
Portabilidad 🌐	✅	⚠️	Java es multiplataforma por la JVM; C# depende de .NET pero con .NET Core mejoró en este aspecto.
Eficiencia ⚡	⚠️	✅	C# tiene mejor optimización en ejecución gracias a .NET JIT.
Seguridad 🛡️	✅	✅	Ambos manejan excepciones, acceso restringido y recolección de basura.

Conclusión

- **Java** es más portable y ampliamente usado en aplicaciones empresariales y Android.
- **C#** tiene más características modernas y mejor integración con Windows/.NET.

- Ambos son lenguajes robustos, con diferencias en simplicidad y optimización.

¿Cuál elegir?

Si buscas **portabilidad** y un ecosistema empresarial fuerte, elige **Java**.

Si buscas **rendimiento** y herramientas más modernas, elige **C#**.

6)

Lenguaje ADA - Características Principales

Ada es un lenguaje de programación creado en los años 80 por el Departamento de Defensa de EE.UU. para aplicaciones críticas y sistemas embebidos. Se destaca por su **seguridad, modularidad y concurrencia**.

◆ 1. Tipos de Datos

Ada tiene un **sistema de tipos fuerte y estricto**, lo que ayuda a evitar errores en tiempo de ejecución.

✓ Tipos primitivos

- Integer → Enteros de tamaño definido.
- Float → Números de punto flotante.
- Boolean → True o False .
- Character y String → Caracteres y cadenas.

✓ Tipos definidos por el usuario

Ada permite definir tipos personalizados con rangos específicos,

◆ 2. Tipos Abstractos de Datos y Paquetes

Ada fomenta la **modularidad** con **paquetes** (packages), similares a los módulos en otros lenguajes.

◆ 3. Estructuras de Datos

- ✓ Registros (estructuras de datos compuestas)
- ✓ Arrays con rangos personalizados
- ✓ Listas, Pilas y Colas

◆ 4. Manejo de Excepciones

Ada tiene un **sistema robusto de manejo de errores**, evitando fallos en tiempo de ejecución.

✓ **Excepciones personalizadas**

◆ 5. Manejo de Concurrency

Ada fue uno de los primeros lenguajes en soportar **concurrency de forma nativa**, usando tareas (`task`).

◆ Conclusión

✓ Ada es un **lenguaje seguro y confiable**, usado en **sistemas críticos** como **aviación, defensa y control industrial**.

✓ Sus características avanzadas en **concurrency, excepciones y modularidad** lo hacen **ideal para software de alto rendimiento**.

⚠ Sin embargo, es menos popular que lenguajes como C o Java en desarrollo general.

7

¿Para qué fue creado Java?

Java fue creado por Sun Microsystems en 1995 con el objetivo de ser un lenguaje de programación portátil, seguro y eficiente. Inicialmente, se diseñó para dispositivos embebidos, pero rápidamente se adaptó a la programación de aplicaciones empresariales, de escritorio y web.

¿Qué cambios le introdujo a la Web?

Java revolucionó la web al permitir el desarrollo de aplicaciones interactivas y dinámicas a través de **applets** y **servlets**. Antes de Java, las páginas web eran estáticas y dependían de scripts en el cliente. Java introdujo la posibilidad de ejecutar código en el navegador y en el servidor, lo que permitió el desarrollo de aplicaciones web más avanzadas y seguras.

¿Java es dependiente de la plataforma donde se ejecuta? ¿Por qué?

No, Java es **independiente de la plataforma** gracias a la **Java Virtual Machine (JVM)**. El código fuente se compila en **bytecode**, que puede ejecutarse en cualquier sistema operativo que tenga una JVM compatible. Esto sigue el principio de *"Write Once, Run Anywhere"* (escribe una vez, ejecuta en cualquier lugar).

8

¿Qué son los applets?

Los **applets** eran pequeños programas escritos en Java que se ejecutaban en un navegador web dentro de una página. Fueron muy populares en los años 90 y principios de los 2000, ya que permitían añadir interactividad a las páginas web. Sin embargo, fueron eliminados por

problemas de seguridad y la llegada de tecnologías más eficientes como **JavaScript, HTML5 y CSS3**.

¿Qué son los servlets?

Los **servlets** son programas en Java que se ejecutan en un servidor y manejan solicitudes HTTP. Son la base de muchas aplicaciones web, ya que permiten generar páginas dinámicas y procesar datos en el backend. A diferencia de los applets, los servlets aún se utilizan, aunque han sido en gran parte reemplazados por frameworks como **Spring y Jakarta EE**.

9

¿Cómo es la estructura de un programa escrito en C?

Un programa en C sigue una estructura básica que incluye las siguientes secciones:

1. **Directivas de preprocesador:** Se colocan al inicio e incluyen bibliotecas necesarias, como `<stdio.h>`.
2. **Definición de constantes y macros** (opcional): Se pueden definir con `#define`.
3. **Declaraciones globales** (opcional): Variables o estructuras que estarán disponibles en todo el programa.
4. **Funciones:** El código se organiza en funciones, siendo `main()` la principal.
5. **Cuerpo del programa:** La ejecución comienza en `main()`, donde se llama a otras funciones.

¿Existe anidamiento de funciones en C?

No, **C no permite definir funciones dentro de otras funciones**. Cada función debe declararse de forma separada dentro del archivo fuente. Sin embargo, una función puede llamar a otra dentro de su ejecución.

10

¿Cómo maneja C las expresiones?

El lenguaje C proporciona un conjunto de operadores y reglas para evaluar expresiones.

1. **Operadores aritméticos:**
 - Suma (+), resta (-), multiplicación (*), división (/), módulo (%).
 - Ejemplo: `int resultado = 5 + 3 * 2;` (respetar la precedencia de operadores).
2. **Operadores relacionales** (devuelven 0 o 1 como resultado booleano):
 - `==`, `!=`, `<`, `>`, `<=`, `>=`.
 - Ejemplo: `if (a > b) { printf("a es mayor que b"); }`
3. **Operadores lógicos:**

- AND lógico (`&&`), OR lógico (`||`), NOT (`!`).
- Ejemplo: `if (x > 0 && x < 10) { printf("x está en el rango"); }`

4. Operadores de asignación:

- `=`, `+=`, `-=`, `*=`, `/=`, `%=`.
- Ejemplo: `a += 5; // Equivalente a a = a + 5;`

5. Operadores bit a bit:

- `&` (AND), `|` (OR), `^` (XOR), `~` (NOT), `<<` (desplazamiento a la izquierda), `>>` (desplazamiento a la derecha).
- Ejemplo: `x = y << 2; // Desplaza los bits de y dos posiciones a la izquierda`

6. Operador ternario:

- Sintaxis: `condición ? valor_si_verdadero : valor_si_falso;`
- Ejemplo: `int menor = (a < b) ? a : b;`

7. Conversión de tipos:

- Implícita: `float resultado = 5 / 2; // resultado = 2.0 (conversión automática)`
- Explícita (casting): `float resultado = (float) 5 / 2; // resultado = 2.5`

11

¿Qué tipo de programas se pueden escribir con cada lenguaje?

- **Python:** Aplicaciones web (Django, Flask), inteligencia artificial, análisis de datos, scripting, automatización, videojuegos, aplicaciones de escritorio.
- **Ruby:** Desarrollo web (Ruby on Rails), automatización, aplicaciones de escritorio, herramientas de administración de servidores.
- **PHP:** Desarrollo web backend (WordPress, Laravel), gestión de bases de datos, aplicaciones empresariales.

¿A qué paradigma responde cada uno?

- **Python:** Multiparadigma (*orientado a objetos, funcional, imperativo*).
- **Ruby:** Multiparadigma (*principalmente orientado a objetos, admite funcional*).
- **PHP:** Principalmente *imperativo y orientado a objetos*, pero también admite programación funcional.

¿Qué características determinan la pertenencia a cada paradigma?

- **Orientado a objetos:** Uso de clases y objetos, encapsulación, herencia y polimorfismo.
- **Funcional:** Funciones de orden superior, inmutabilidad, uso de `map`, `reduce` y `filter`.
- **Imperativo:** Uso de estructuras como bucles y condicionales para modificar el estado del programa.

12

Características importantes de cada lenguaje

- **Python:**
 - Tipado dinámico y fuerte.
 - Se organiza en módulos y paquetes.
 - Manejo de excepciones con `try-except`.
 - Interactivo e interpretado.
- **Ruby:**
 - Todo es un objeto (incluso números y booleanos).
 - Tipado dinámico y fuerte.
 - Código muy legible y flexible.
 - Soporta metaprogramación.
- **PHP:**
 - Diseñado para el desarrollo web.
 - Tipado dinámico y débil.
 - Puede incrustarse en HTML.
 - Manejo de sesiones y cookies incorporado.
- **Gobstone:**
 - Lenguaje educativo para enseñar programación.
 - Basado en comandos simples para mover fichas en un tablero.
 - No tiene variables, trabaja con estado.
- **Processing:**
 - Usado en arte visual e interacción gráfica.
 - Basado en Java.
 - Ideal para gráficos, animaciones y visualización de datos.

Ejercicio 13

¿A qué paradigma pertenece JavaScript?

JavaScript es **multiparadigma**, ya que soporta:

- **Programación orientada a objetos** (con prototipos y clases).

- **Programación funcional** (funciones de orden superior, inmutabilidad).
- **Programación imperativa** (bucles, estructuras de control).

¿A qué tipo de lenguaje pertenece?

- Es un lenguaje **interpretado** y **dinámico**.
- Se ejecuta en navegadores y servidores (*Node.js*).
- Se usa principalmente para aplicaciones web interactivas.

14

Características importantes de JavaScript

- **Tipado de datos:** Débil y dinámico (las variables pueden cambiar de tipo).
- **Excepciones:** Se manejan con `try-catch`.
- **Variables:**
 - `var` : Variable global o de función (evitar usarla).
 - `let` : Variable con ámbito de bloque.
 - `const` : Variable inmutable.

Otras observaciones:

¿Qué significa que un lenguaje sea tipado?

El **tipado** se refiere a cómo un lenguaje de programación maneja y asigna **tipos de datos** a las variables. Un lenguaje puede ser más o menos estricto en cuanto a la necesidad de definir o convertir los tipos de datos.

1 Tipado estático vs. tipado dinámico

✓ Tipado estático

- El tipo de una variable se **declara en tiempo de compilación** y no puede cambiar.
- Si intentas asignar un tipo incorrecto, obtendrás un error antes de ejecutar el programa.
- Ejemplos de lenguajes con tipado estático: **C, Java, C++, Rust, TypeScript**.

✓ Tipado dinámico

- El tipo de una variable **se asigna en tiempo de ejecución**.
- Puedes cambiar el tipo de una variable sin que el lenguaje lo impida.
- Ejemplos de lenguajes con tipado dinámico: **Python, JavaScript, Ruby, PHP**.

2 Tipado fuerte vs. tipado débil

✓ Tipado fuerte

- No permite conversiones implícitas entre tipos incompatibles.
- Evita errores al mezclar tipos de datos de forma no controlada.
- Ejemplos: **Python, Java, Rust, Ruby**.

✓ Tipado débil

- Permite conversiones implícitas entre tipos diferentes.
- Puede llevar a errores inesperados en la ejecución.
- Ejemplos: **JavaScript, PHP, C (en ciertas situaciones)**.

3 Tipado implícito vs. tipado explícito

✓ Tipado implícito

- El lenguaje **deduce el tipo de dato automáticamente**.
- Ejemplos: **Python, JavaScript, C++ (con `auto`), TypeScript (con inferencia de tipos)**.

✓ Tipado explícito

- El programador **debe declarar el tipo de dato** en cada variable.
- Ejemplos: **C, Java, Rust**.