

Conceptos y Paradigmas de Lenguajes de Programación 2023

Práctica Nro 1

Historia, evolución y características de Leng. de Programación

Objetivo: Conocer la evolución de los lenguajes de programación y sus características.

Ejercicio 1: Los lenguajes de programación más representativos son:

1951 - 1955: Lenguajes tipo assembly
1956 - 1960: FORTRAN, ALGOL 58, ALGOL 60, LISP
1961 - 1965: COBOL, ALGOL 60, SNOBOL, JOVIAL
1966 - 1970: APL, FORTRAN 66, BASIC, PL/I, SIMULA 67, ALGOL-W
1971 - 1975: Pascal, C, Scheme, Prolog
1976 - 1980: Smalltalk, Ada, FORTRAN 77, ML
1981 - 1985: Smalltalk 80, Turbo Pascal, Postscript
1986 - 1990: FORTRAN 90, C++, SML
1991 - 1995: TCL, PERL, HTML
1996 - 2000: Java, Javascript, XML

Indique para cada uno de los períodos presentados cuales son las características nuevas que se incorporan y cual de ellos la incorpora.

Ejercicio 2: Escriba brevemente la historia del lenguaje de programación que eligió en la encuesta u otro de su preferencia.

Ejercicio 3: ¿Qué atributos debería tener un buen lenguaje de programación? Por ejemplo, ortogonalidad, expresividad, legibilidad, simplicidad, etc. De al menos un ejemplo de un lenguaje que cumple con las características citadas.

Ejercicio 4: Tome uno o dos lenguajes de los que ud. Conozca y

- Describa los tipos de expresiones que se pueden escribir en el/ellos
- Describa las facilidades provistas para la organización del programa
- Indique cuáles de los atributos del ejercicio anterior posee el/los lenguaje/s elegidos y cuáles no posee, justifique en cada caso.

Lenguajes - ADA

Ejercicio 5: Describa las características más relevantes de Ada, referida a:

- Tipos de datos
- Tipos abstractos de datos – paquetes
- Estructuras de datos
- Manejo de excepciones
- Manejo de concurrencia

Conceptos y Paradigmas de Lenguajes de Programación 2023

Lenguajes - JAVA

Ejercicio 6: Diga para qué fue, básicamente, creado Java. ¿Qué cambios le introdujo a la Web? ¿Java es un lenguaje dependiente de la plataforma en dónde se ejecuta? ¿Porqué?

Ejercicio 7: ¿Sobre qué lenguajes está basado?

Ejercicio 8: ¿Qué son los applets? ¿Qué son los servlets?

Lenguajes - C

Ejercicio 9: ¿Cómo es la estructura de un programa escrito en C? ¿Existe anidamiento de funciones?

Ejercicio 10: Describa el manejo de expresiones que brinda el lenguaje.

Lenguajes - Python - RUBY - PHP

Ejercicio 11: ¿Qué tipo de programas se pueden escribir con cada uno de estos lenguajes? ¿A qué paradigma responde cada uno? ¿Qué características determinan la pertenencia a cada paradigma?

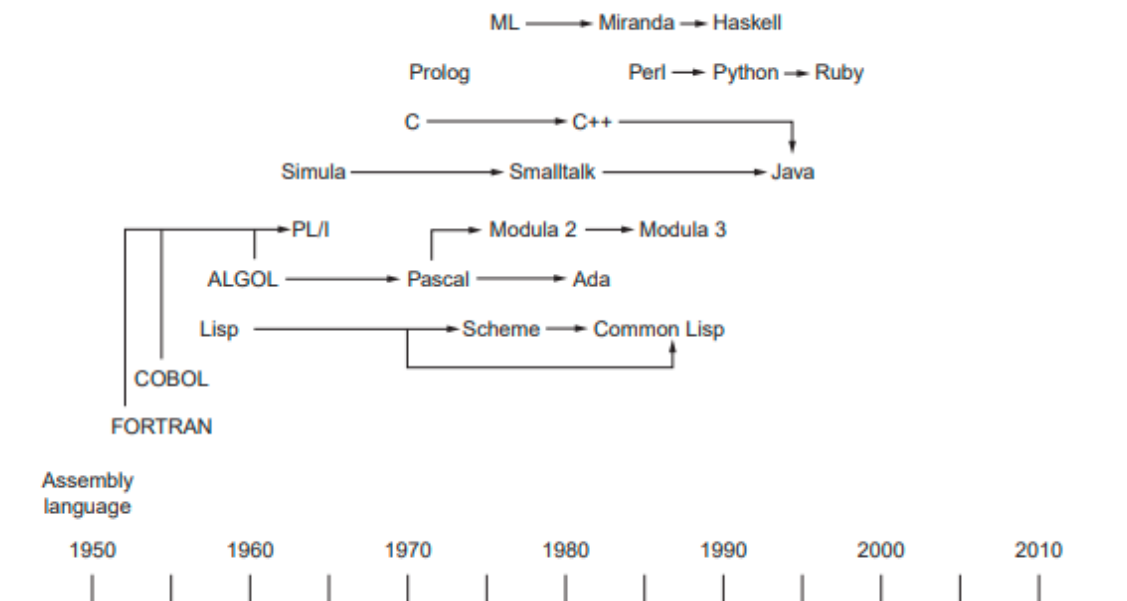
Ejercicio 12: Cite otras características importantes de Python, Ruby, PHP, Golang y Processing. Por ejemplo: tipado de datos, cómo se organizan los programas, etc.

Lenguaje Javascript

Ejercicio 13: ¿A qué tipo de paradigma corresponde este lenguaje? ¿A qué tipo de Lenguaje pertenece?

Ejercicio 14: Cite otras características importantes de javascript. Tipado de datos, excepciones, variables, etc.

1. Línea de Tiempo:



● Periodo de 1951 a 1955:

- Assembly: Para el período de tiempo de 1951 a 1955 el lenguaje Assembly permitió incorporar símbolos mnemotécnicos para reemplazar los códigos de instrucción binarios y las localidades de memoria del modelo de Von Neumann.

● Periodo de 1956 a 1965:

- FORTTRAN: Introdujo la modularidad mediante subprogramas desarrollados y compilados de forma separada, y la posibilidad de compartir datos entre módulos a través de un entorno global (COMMON).
- Familia ALGOL: Los objetivos a cumplir de ALGOL eran mantener un lenguaje de programación de alto nivel de abstracción para escribir instrucciones de código concisas y comprensibles. ALGOL proporcionó en primer lugar una notación estándar para que los científicos de la computación publicaran algoritmos en revistas. Del mismo modo, estaban disponibles notaciones elegantes para expresar datos de diferentes tipos numéricos (entero y flotante), así como la estructura de datos de matriz. Por último, se proporcionaba soporte para procedimientos, incluidos los procedimientos recursivos, también logró la independencia de la máquina para la ejecución de programas en computadoras al requerir que cada tipo de

hardware proporcionara un compilador de ALGOL. Este programa traducía programas ALGOL estándar al código de máquina de una máquina particular. A su vez, ALGOL fue uno de los primeros lenguajes en recibir una especificación o definición formal.

- COBOL introdujo archivos y descripciones de datos, así como una noción muy preliminar de programación en un lenguaje casi natural.
- LISP: La definición del lenguaje se basaba en la teoría de funciones recursivas y el cálculo lambda, y sentó las bases para una nueva clase de lenguajes llamados funcionales (o aplicativos). El LISP puro está libre de los conceptos de variables modificables de Von Neumann, declaraciones de asignación, instrucciones goto, y así sucesivamente. Los programas LISP son exactamente como las estructuras de datos generales de LISP, y por lo tanto, el intérprete de LISP puede ser especificado en LISP de una manera bastante simple.
- SNOBOL4: SNOBOL4 proporcionó un lenguaje con facilidades a la hora de manipular strings y utilizar coincidencia de patrones (pattern matching).

- **Periodo de 1966 a 1970:**

- APL es otro lenguaje que admite un estilo de programación funcional. Su conjunto muy amplio de operadores, especialmente en matrices, libera al programador de tener que utilizar manipulaciones de matriz a nivel inferior y elemento por elemento.
- BASIC: Si bien el lenguaje no introduce nuevos conceptos lingüísticos, es de las primeras herramientas con una programación interpretativa altamente interactiva.
- PL/I: PL/I fue diseñado a mediados de la década de 1960 con un objetivo ambicioso: integrar los conceptos más fructíferos y originales de lenguajes anteriores en un lenguaje de programación verdaderamente general y universal. Tomando conceptos de FORTRAN, ALGOL 60, COBOL y LISP e incorporando características como puede ser el manejo de excepciones y algunas facilidades de multitarea primitivas.
- SIMULA 67: Se diseñó para resolver problemas de simulación discreta. Además de construcciones ad hoc para simulación y corrutinas que proporcionan una forma primitiva de ejecución paralela. Este lenguaje introdujo el concepto de clase, un mecanismo de modularización que

puede agrupar un conjunto de rutinas relacionadas y una estructura de datos.

- **Periodo de 1971 a 1975:**

- **PASCAL**: Principalmente concebido como un vehículo para enseñar programación estructurada, hubo una rápida expansión del interés en Pascal con la llegada de computadoras personales de bajo costo. El principal atractivo del lenguaje es la simplicidad y el soporte para la programación disciplinada.
- **C**: Para la época era necesario tratar conceptos como pueden ser: tipos de datos abstractos y control de visibilidad a módulos, tipado fuerte y verificación estática del programa, relación entre las construcciones del lenguaje y demostraciones formales de corrección, módulos genéricos, manejo de excepciones, concurrencia, y comunicación y sincronización entre procesos. C fue un lenguaje que sobrevivió a esta etapa de desarrollo experimental y además, tuvo mucho éxito por su potencia como por la disponibilidad de implementaciones eficientes en una amplia variedad de máquinas.
- **SCHEME**: Proporcionó un dialecto de LISP con el objetivo de simplificar las instrucciones del lenguaje LISP.
- **PROLOG**: fue el punto de partida de una nueva familia de lenguajes: los lenguajes de programación lógica.

- **Periodo de 1976 a 1980:**

- **Smalltalk**: Smalltalk fue el primer lenguaje puramente orientado a objetos, diseñado para una máquina especializada y enfocado en el desarrollo de aplicaciones en un entorno altamente interactivo de estación de trabajo personal para un único usuario. Este lenguaje dinámico asigna el tipo de los objetos en tiempo de ejecución y su sintaxis refleja su enfoque en la orientación a objetos. Smalltalk establece una jerarquía de clases donde las subclases heredan y pueden modificar los atributos y métodos de las superclases. La resolución dinámica de métodos permite que las llamadas a los objetos se vinculen dinámicamente durante la ejecución del programa. Este artículo explora las características fundamentales de Smalltalk y cómo su enfoque en la orientación a objetos influye en el desarrollo de software.
- **Ada**: El deseo de unificar los lenguajes de programación utilizados en aplicaciones informáticas integradas y la necesidad de un software más confiable y mantenible llevaron al Departamento de Defensa de los Estados Unidos

a patrocinar el diseño de un nuevo lenguaje. El resultado de este proceso es el lenguaje de programación Ada, que se puede ver como la síntesis de conceptos de vanguardia de los lenguajes de programación convencionales.

- ML: La importante contribución conceptual de ML fue demostrar que los lenguajes de programación pueden ser muy potentes computacionalmente, y aun así pueden preservar la capacidad de demostrar la ausencia de ciertos tipos de errores sin ejecutar programas.

- **Periodo de 1981 a 1985:**

- Turbo Pascal: Es una implementación moderna de Pascal que proporciona instalaciones de compilación separada más seguras basadas en la noción de un módulo que encapsula un conjunto de constantes, procedimientos y tipos.
- Postscript: Lenguaje basado en pilas que utiliza exclusivamente la notación postfija para programar.

- **Periodo de 1986 a 1990:**

- FORTRAN 90: Este lenguaje se instauró como el estándar actual de FORTRAN, compatible con las versiones anteriores que presentó limitadas formas de sobrecargas.
- C++: Apareció en escena justo cuando el interés en técnicas orientadas a objetos estaba en auge. La semántica de C++ no conllevaba ninguna penalización de rendimiento. Estas cualidades permitieron que C++ introdujera características orientadas a objetos en la corriente principal de la informática. La popularidad de C++ también se vio mejorada por su flexibilidad, su naturaleza híbrida y la disposición de su diseñador para ampliar su características basándose en la experiencia práctica.
- SML: Luego de una revisión del lenguaje ML durante esta época, se combinaron características del lenguaje anterior con el lenguaje HOPE, generando el lenguaje Standard ML o SML.

- **Periodo de 1991 a 1995:**

- TCL: Lenguaje de script que especifica patrones de activación para fragmentos de herramientas visuales existentes, comportándose como un soporte para el desarrollo rápido de aplicaciones, lo cual resultó útil para el desarrollo de prototipos.
- PERL: PERL incorporó un sistema robusto de módulos que facilita la reutilización de código y la distribución de software. Además introdujo el CPAN (Comprehensive Perl

Archive Network) que es un repositorio de módulos PERL que contiene miles de módulos listos para usar.

- **HTML:** HTML (HyperText Markup Language) es el lenguaje estándar para crear y diseñar páginas web. Durante esta época este lenguaje introdujo formularios para la entrada de datos del usuario, la adición de elementos como , <map>, y <textarea>, como también la definición de un conjunto más amplio de elementos de estructura como <h1> a <h6>, , , <dl>, entre otros.

- **Periodo de 1996 a 2000:**

- **Java:** Los objetivos que se plantearon a la hora de crear Java eran implementar una máquina virtual y un lenguaje con una estructura y sintaxis que fuera similar a C++, de tal forma que se proporcionó un lenguaje independiente a la plataforma donde se use y un entorno de ejecución (JVM) ligero, gratuito y seguro. A su vez, Java introdujo colecciones, mejoras en la gestión de eventos y GUI (Graphical User Interface) con AWT y Swing y, clases y métodos para manipular hilos de ejecución.
- **Javascript:** Durante este periodo este lenguaje experimentó un crecimiento y desarrollo significativo, ya que se convirtió en un elemento fundamental en el desarrollo web. Incorporó la especificación del DOM Level 1 (Document Object Model), que proporcionaba una interfaz estándar para acceder y manipular documentos HTML Y XML, introdujo nuevas funciones y métodos para mejorar su funcionalidad y utilidad (manejo de arreglos y manejo de cadenas de texto), mejoró su soporte para la manipulación de eventos en la web, introdujo la exploración de nuevas técnicas y prácticas de programación como pueden ser la POO, el uso de design patterns, entre otros. Y por último logró ser adoptado generalmente en una amplia variedad de aplicaciones web.
- **XML:** Durante el período de 1996 a 2000, XML (Extensible Markup Language) experimentó un rápido crecimiento y adopción, lo que llevó a la implementación de varias características nuevas y significativas, es un lenguaje de marcado diseñado para almacenar y transportar datos de forma legible tanto para humanos como para máquinas.

2. Historia de Python:

- Python es un lenguaje de programación de alto nivel creado por Guido van Rossum a finales de los 80 y principios de los 90. Van Rossum comenzó a trabajar en Python en diciembre de 1989, mientras trabajaba en el Centro de Matemáticas y Ciencias de la

Computación (CWI) en los Países Bajos. El nombre "Python" proviene de la afición de Van Rossum por el grupo de comedia británico "Monty Python".

La primera versión pública de Python, la versión 0.9.0, fue lanzada en febrero de 1991. Desde entonces, Python ha experimentado un crecimiento constante en popularidad debido a su sintaxis simple y legible, así como a su versatilidad y potencia. Python se ha utilizado en una amplia gama de aplicaciones, desde desarrollo web y científico hasta automatización de tareas y aprendizaje automático.

Python ha introducido varias innovaciones significativas en el mundo de la programación. Por un lado, su sintaxis clara y legible, diseñada para mejorar la legibilidad del código, ha hecho que Python sea especialmente popular entre los principiantes en programación y ha contribuido a su adopción en una amplia variedad de campos. Además, Python es un lenguaje interpretado y de alto nivel, lo que facilita el desarrollo rápido y la depuración de código, así como la abstracción de detalles de bajo nivel.

Python también es un lenguaje de programación orientado a objetos, lo que permite a los programadores modelar sus programas utilizando objetos y clases para organizar y reutilizar el código de manera eficiente. Además, Python viene con una amplia biblioteca estándar que proporciona una variedad de módulos y funciones para realizar tareas comunes, lo que hace que sea muy versátil y apto para una variedad de aplicaciones. Finalmente, Python cuenta con una comunidad activa y acogedora que contribuye con bibliotecas y marcos de trabajo de código abierto, lo que ha llevado a un ecosistema próspero y diverso de herramientas y recursos que amplían las capacidades del lenguaje.

En resumen, Python no solo ha sido una herramienta poderosa en el ámbito de la programación, sino que también ha introducido innovaciones importantes que han influido en la forma en que se desarrollan y mantienen los programas en la actualidad. Su legibilidad, simplicidad, orientación a objetos y ecosistema de desarrollo activo han contribuido a su posición como uno de los lenguajes de programación más populares y versátiles disponibles actualmente.

3. Atributos que debería tener un buen lenguaje de Programación:

- **Simplicidad y legibilidad:** Python, Ruby y Javascript
 - Poder producir programas fáciles de escribir y leer.
 - Resultar fácil a la hora de aprenderlo y enseñarlo.
 - Cuestiones que atentan contra este atributo:

- Muchas componentes elementales.
 - Conocer subconjuntos de componentes.
 - El mismo concepto semántico -> distinta sintaxis.
 - Distintos conceptos semánticos -> misma notación sintáctica.
 - Abuso de operadores sobrecargados.
- Claridad en los Bindings: Rust y Python
 - Los elementos de los lenguajes de programación pueden ligarse a sus atributos o propiedades en diferentes momentos:
 - Definición del lenguaje.
 - Implementación del lenguaje.
 - En escritura del programa.
 - Compilación.
 - Cargado del programa.
 - En ejecución.
 - La ligadura en cualquier caso debe ser clara.
- Confiabilidad: Ada o Rust
 - La confiabilidad está relacionada con la seguridad.
 - Chequeo de tipos
 - Cuanto antes se encuentren errores, menos costoso resulta realizar los arreglos que se requieran.
 - Manejo de excepciones
 - La habilidad para interpretar errores en tiempo de ejecución, tomar medidas correctivas y continuar.
- Soporte: Javascript, Java, C#
 - Debería ser accesible para cualquiera que quiera usarlo o instalarlo. Como ideal, que su compilador o intérprete sea de dominio público.
 - Debería poder ser implementado en diferentes plataformas.
 - Deberían existir diferentes medios para poder familiarizarse con el lenguaje.
- Abstracción: Java, Python, Haskell
 - Capacidad de definir y usar estructuras u operaciones complicadas de manera que sea posible ignorar muchos de los detalles, generando una abstracción de procesos y de datos.
- Ortogonalidad: Python
 - Significa que un conjunto limitado de constructores primitivos puede combinarse de manera flexible y coherente para construir una variedad de estructuras de control y datos. Además, cada combinación de estos constructores

es legal y tiene sentido desde el punto de vista de la sintaxis y la semántica del lenguaje.

- **Eficiencia: C, C++, Rust, Go**
 - Esta se mide en tiempo y espacio, esfuerzo humano y en la optimización.

4. Python

- **Tipos de expresiones:**
 - **Expresiones Aritméticas:** Se utilizan para realizar operaciones matemáticas básicas como suma, resta, multiplicación, división, etc.
 - **Expresiones de Comparación:** Se utilizan para comparar valores y devolver un resultado booleano (True o False).
 - **Expresiones Lógicas:** Se utilizan para combinar expresiones de comparación utilizando los operadores lógicos AND, OR y NOT.
 - **Expresiones de Asignación:** Se utilizan para asignar valores a variables. Python permite asignaciones múltiples y asignaciones aumentadas (+, -=, *=, /=, etc.).
 - **Expresiones de Cadena:** Se utilizan para trabajar con cadenas de texto. Python admite operaciones como concatenación, indexación, slicing, entre otras.
 - **Expresiones de Listas, Tuplas y Diccionarios:** Se utilizan para trabajar con estructuras de datos en Python. Esto incluye acceso a elementos, añadir o eliminar elementos, entre otras operaciones.
 - **Expresiones de Funciones:** Se utilizan para llamar a funciones y pasar argumentos. Python permite definir funciones propias y también utiliza muchas funciones incorporadas en su biblioteca estándar.
- **Facilidades provistas para la organización del programa:**
 - **Indentación:** Python utiliza la indentación para definir bloques de código en lugar de llaves u otras estructuras de delimitación. Esto promueve una escritura de código limpia y legible, ya que obliga a mantener una estructura visual coherente.
 - **Funciones y Modularidad:** Python permite definir funciones para dividir el código en partes reutilizables y lógicamente coherentes. La modularidad facilita la organización del código y su mantenimiento. Además, Python admite la importación de módulos externos para reutilizar funcionalidades proporcionadas por la comunidad o desarrolladas internamente.

- **Clases y Programación Orientada a Objetos:** Python es un lenguaje de programación orientado a objetos, lo que significa que permite la definición de clases y objetos. Esto facilita la organización del código al encapsular datos y comportamientos relacionados en objetos, lo que promueve la reutilización y la modularidad del código.
- **Comentarios:** Python permite agregar comentarios en el código para documentar su funcionamiento. Los comentarios son útiles para explicar partes complicadas del código, proporcionar contexto o recordatorios, y facilitar la comprensión del código para otros desarrolladores.
- **Convenciones de Nombres y Estilo de Código:** Python tiene convenciones de nombres y estilos de código bien establecidos (como PEP 8) que ayudan a mantener la consistencia y la legibilidad del código.
- **Atributos previamente mencionados:**
 - **Simplicidad y Legibilidad:** Python se destaca por su sintaxis simple y legible. Su diseño hace hincapié en la legibilidad del código, lo que facilita la comprensión y el mantenimiento del mismo. La simplicidad se refleja en la filosofía del lenguaje, como se describe en el Zen de Python (PEP 20), que enfatiza la claridad y la simplicidad.
 - **Claridad en los Bindings:** Python proporciona una forma clara de vincular nombres a objetos mediante la asignación de variables. Los nombres de variables y funciones pueden ser descriptivos, lo que mejora la claridad y la comprensión del código.
 - **Confiabilidad:** Python es conocido por su estabilidad y confiabilidad. Ha sido utilizado en una amplia gama de aplicaciones durante muchos años y cuenta con una gran comunidad de desarrolladores que contribuyen a su mejora continua. Además, Python ofrece herramientas de prueba integradas y frameworks de pruebas de unidad que ayudan a garantizar la calidad del código.
 - **Soporte:** Python cuenta con una amplia base de usuarios y una comunidad activa que proporciona soporte a través de foros en línea, listas de correo, sitios web y otras plataformas. Además, la documentación oficial de Python es exhaustiva y de alta calidad, lo que facilita el aprendizaje y la resolución de problemas.
 - **Abstracción:** Python ofrece múltiples niveles de abstracción que permiten a los desarrolladores concentrarse en la lógica del programa sin preocuparse por los detalles de implementación. Por ejemplo, las estructuras de datos integradas y las funciones de la biblioteca estándar

proporcionan abstracciones útiles que simplifican el desarrollo de software.

- **Ortogonalidad:** Python tiende a ser ortogonal en su diseño, lo que significa que las características del lenguaje no están sobrecargadas ni interconectadas de manera innecesaria. Esto contribuye a la consistencia y coherencia del lenguaje, lo que facilita su aprendizaje y uso.
- **Eficiencia:** Aunque Python no es tan eficiente en términos de velocidad de ejecución como algunos otros lenguajes de programación de bajo nivel, su eficiencia en términos de desarrollo y mantenimiento del código es notable. Además, Python ofrece la capacidad de integrar módulos escritos en lenguajes de bajo nivel como C y C++ para mejorar el rendimiento en áreas críticas de la aplicación.

5. ADA

- **Tipos de datos:**

- **Tipado estático fuerte:** En ADA, cada variable debe ser declarada con un tipo específico y este tipo no puede cambiar durante la ejecución del programa. Esto ayuda a prevenir errores de tipo en tiempo de ejecución.
- **Tipos escalares:** ADA soporta tipos de datos escalares como enteros, reales, caracteres y booleanos. Los enteros pueden ser de tamaño fijo (por ejemplo, Integer) o de tamaño variable (por ejemplo, Integer_16, Integer_32, Integer_64). Los reales pueden ser de precisión simple (Float) o doble precisión (Long_Float).
- **Enumeraciones:** ADA permite definir tipos enumerados, donde el programador puede especificar un conjunto finito de valores posibles para una variable.
- **Tipos derivados:** Los tipos derivados permiten al programador crear nuevos tipos a partir de otros tipos existentes.
- **Arrays:** ADA soporta arreglos multidimensionales de tamaño fijo y tamaño variable.
- **Registros (Records):** Los registros en ADA son similares a las estructuras en otros lenguajes de programación.
- **Punteros:** ADA permite el uso de punteros mediante el tipo Access, que se utiliza para referenciar datos dinámicamente asignados en el almacenamiento.

- **Tipos abstractos de datos - paquetes:**

- **Encapsulación:** Los paquetes en ADA permiten encapsular datos y procedimientos relacionados en un único módulo. Esto promueve la modularidad y el ocultamiento de la

información, lo que facilita el mantenimiento y la reutilización del código.

- **Declaración y especificación:** Un paquete en ADA consta de dos partes principales: la declaración del paquete en la especificación (archivo .ads) y la implementación del paquete en el cuerpo (archivo .adb). La especificación define la interfaz pública del paquete, incluyendo tipos, variables y procedimientos accesibles desde fuera del paquete.
- **Tipos privados:** ADA permite la definición de tipos privados dentro de un paquete, lo que significa que los detalles internos de la implementación pueden ocultarse al usuario.
- **Procedimientos y funciones:** Los paquetes en ADA pueden contener procedimientos y funciones que operan en los tipos definidos dentro del paquete. Estas subrutinas pueden ser públicas o privadas, según se especifique en la interfaz del paquete.
- **Variables privadas y constantes:** Además de tipos, procedimientos y funciones, los paquetes pueden contener variables privadas y constantes que son accesibles sólo dentro del paquete.
- **Uso de renombramiento:** ADA permite el uso de renombramiento dentro de los paquetes para proporcionar nombres alternativos para tipos, variables, procedimientos y funciones.
- **Especificación de la dependencia:** Los paquetes en ADA pueden depender de otros paquetes, lo que significa que pueden utilizar tipos y subrutinas definidos en otros lugares.
- **Estructuras de datos:**
 - **Arrays:** ADA admite la creación de arreglos unidimensionales y multidimensionales con tamaño fijo o variable. Estos arreglos pueden contener elementos de cualquier tipo de datos válido en ADA, incluyendo tipos primitivos como enteros y caracteres, así como tipos compuestos como registros y otros arreglos.
 - **Registros (Records):** Los registros en ADA permiten agrupar diferentes tipos de datos relacionados bajo una única entidad. Cada campo del registro puede tener un tipo de datos distinto. Esto facilita la organización de datos complejos en una estructura coherente.
 - **Listas enlazadas (Linked Lists):** Aunque ADA no proporciona listas enlazadas como parte de su biblioteca estándar, es posible implementarlas utilizando punteros y registros. Las listas enlazadas son útiles para estructuras de datos

dinámicas donde el tamaño puede cambiar durante la ejecución del programa.

- **Manejo de excepciones:**

- **Declaración de excepciones:** En ADA, las excepciones se declaran en la especificación del paquete utilizando la cláusula "exception".
- **Razón y mensaje de excepción:** Además de declarar excepciones, los programadores pueden proporcionar información adicional sobre la excepción, como una descripción o una razón para su ocurrencia. Esto facilita la depuración y el manejo de errores.
- **Rutinas de excepción:** ADA proporciona la cláusula "exception" dentro de los bloques "declare" para manejar excepciones específicas que puedan ocurrir dentro de ese bloque. Las rutinas de excepción pueden incluir lógica para manejar el error, registrar información relevante y, opcionalmente, propagar la excepción.
- **Bloques de excepción anidados:** Los bloques begin pueden estar anidados, lo que permite un manejo de excepciones más granular. Esto significa que se pueden manejar diferentes excepciones en diferentes niveles de anidamiento.
- **Propagación de excepciones:** En ADA, las excepciones no manejadas dentro de un bloque son propagadas automáticamente al bloque superior para su manejo. Esto significa que las excepciones pueden ser manejadas en el nivel más apropiado de la jerarquía del programa.
- **Manejo global de excepciones:** ADA permite definir un bloque exception global al final del programa para manejar excepciones no capturadas en ningún otro lugar.

- **Manejo de concurrencia:**

- **Procesos concurrentes:** En ADA, los procesos concurrentes se definen mediante la creación de tareas (tasks) o procesos. Las tareas son unidades de ejecución independientes que pueden ejecutarse simultáneamente con otras tareas.
- **Declaración de tareas:** Las tareas se declaran utilizando la palabra clave "task".
- **Comunicación entre tareas:** ADA proporciona mecanismos seguros para la comunicación entre tareas, como el uso de variables compartidas protegidas (protected objects) y colas de mensajes (message queues).
- **Sincronización de tareas:** ADA permite la sincronización entre tareas mediante el uso de primitivas como semáforos (semaphores), exclusión mutua (mutexes) y barreras (barriers).

- **Planificación de tareas:** ADA proporciona un planificador de tareas incorporado que se encarga de asignar tiempo de CPU a las diferentes tareas en ejecución.
- **Gestión de recursos compartidos:** ADA facilita la gestión de recursos compartidos entre tareas mediante el uso de objetos protegidos.

6. JAVA

- Java fue creado principalmente con la intención de ser un lenguaje de programación versátil, robusto, y portátil, que pudiera funcionar de manera confiable en una variedad de plataformas. Fue desarrollado por Sun Microsystems a principios de la década de 1990, liderado por James Gosling y su equipo, con el objetivo de abordar los desafíos de la programación en el entorno de la World Wide Web emergente en ese momento. Otros de los objetivos claves de diseño para JAVA fueron la Portabilidad, Seguridad y Simplicidad y familiaridad.
- En cuanto a los cambios que introdujo Java en la web, uno de los más significativos fue la capacidad de ejecutar applets Java en los navegadores web. Los applets eran pequeñas aplicaciones que podían integrarse en páginas web para proporcionar interactividad y funcionalidad adicional. Esto permitió una experiencia de usuario más rica y dinámica en comparación con las páginas web estáticas tradicionales.
Sin embargo, a medida que evolucionaron las tecnologías web, como HTML5 y JavaScript, el uso de applets Java ha disminuido significativamente, ya que estas tecnologías ofrecen capacidades similares de manera más eficiente y segura.
- En cuanto a si JAVA es dependiente de la plataforma en donde se ejecuta, la respuesta es sí y no. Java se diseñó para ser "write once, run anywhere" (escribir una vez, ejecutar en cualquier lugar), lo que significa que el código Java compilado debería poder ejecutarse en cualquier dispositivo con una JVM compatible, independientemente de la plataforma subyacente. Sin embargo, la JVM en sí misma es específica de la plataforma, lo que significa que se necesitará una implementación de JVM para cada sistema operativo y arquitectura de hardware en la que se desee ejecutar código Java. Esto hace que Java sea portable en el sentido de que el mismo código fuente puede ejecutarse en diferentes plataformas, pero dependiente de la JVM específica en la que se ejecute.

7. JAVA

- JAVA se basa en una serie de lenguajes, estos son: C, C++, Smalltalk, Objective-C, Ada, Scheme y Lisp.

8. JAVA (De las applets hable un poco en la anterior pero profundizamos más)

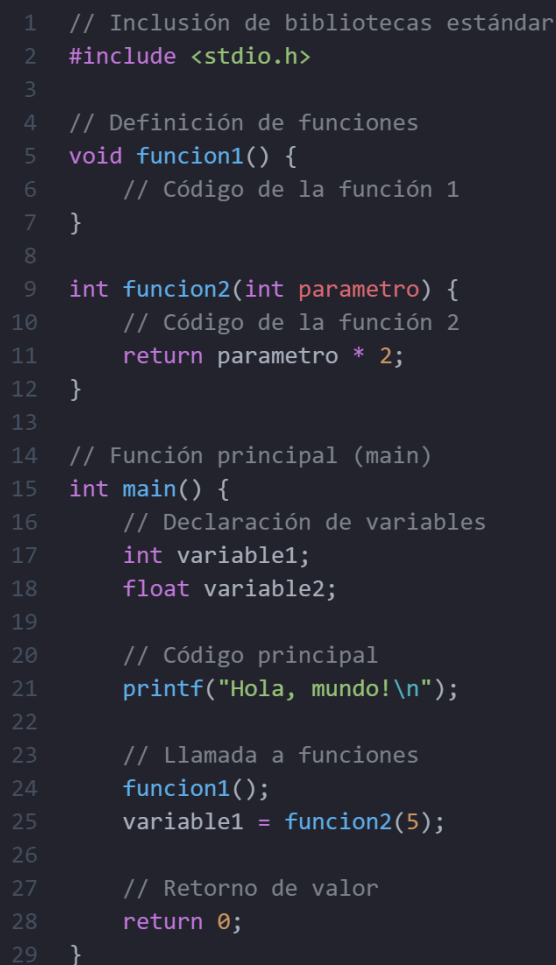
- **Applets:** Son pequeñas aplicaciones diseñadas para ser ejecutadas dentro de un navegador web. Fueron una característica clave en el desarrollo web en la década de 1990 y principios de los 2000, ya que permitían agregar interactividad y funcionalidades avanzadas a las páginas web. Aquí hay algunas características importantes de las applets:
 - **Escritas en Java:** Las applets están escritas en Java y se compilan en bytecode Java, lo que las hace independientes de la plataforma y portables.
 - **Ejecución en un entorno seguro:** Las applets se ejecutan dentro de un entorno controlado llamado sandbox. Esto significa que tienen acceso restringido al sistema y solo pueden realizar operaciones que no comprometan la seguridad del usuario y del sistema.
 - **Interfaz gráfica de usuario (GUI):** Las applets pueden contener elementos de interfaz de usuario como botones, campos de texto, gráficos, etc. Esto les permite ofrecer una experiencia de usuario interactiva similar a las aplicaciones de escritorio.
 - **Integración con HTML:** Las applets se pueden integrar directamente en páginas HTML utilizando la etiqueta <applet>. Esto permite incrustar la applet en una página web y ejecutarla dentro del navegador del usuario.
- **Servlets:** Son programas Java que se ejecutan en el servidor web y generan dinámicamente contenido web. A diferencia de las applets, que se ejecutan en el cliente (navegador web), los servlets se ejecutan en el servidor y procesan las solicitudes web enviadas por los clientes. Aquí hay algunos puntos importantes sobre los servlets:
 - **Diseñados para la lógica del servidor:** Los servlets se utilizan principalmente para implementar la lógica del servidor en una aplicación web.
 - **Basados en Java EE (Enterprise Edition):** Los servlets son una parte fundamental de la plataforma Java EE, que proporciona un conjunto de especificaciones y APIs para el desarrollo de aplicaciones empresariales en Java.
 - **Independientes del cliente:** A diferencia de las applets, que dependen del navegador del cliente, los servlets se ejecutan en el servidor y son independientes del cliente. Esto los

hace adecuados para desarrollar aplicaciones web escalables y seguras.

- **Manejo de solicitudes HTTP:** Los servlets pueden manejar diferentes tipos de solicitudes HTTP, como GET, POST, PUT, DELETE, etc. Esto les permite interactuar con clientes web y procesar datos enviados desde formularios web.

9. C

- Estructura básica de un programa en C:

A screenshot of a code editor with a dark background and light-colored text. The code is a C program structure. It starts with a comment and an include statement for <stdio.h>. Then there are two function definitions: 'funcion1' which is empty, and 'funcion2' which takes an integer parameter and returns its double. Finally, there is a 'main' function which declares two variables, prints 'Hola, mundo!\n', calls 'funcion1' and 'funcion2', and returns 0. The code is numbered from 1 to 29.

```
1 // Inclusión de bibliotecas estándar
2 #include <stdio.h>
3
4 // Definición de funciones
5 void funcion1() {
6     // Código de la función 1
7 }
8
9 int funcion2(int parametro) {
10     // Código de la función 2
11     return parametro * 2;
12 }
13
14 // Función principal (main)
15 int main() {
16     // Declaración de variables
17     int variable1;
18     float variable2;
19
20     // Código principal
21     printf("Hola, mundo!\n");
22
23     // Llamada a funciones
24     funcion1();
25     variable1 = funcion2(5);
26
27     // Retorno de valor
28     return 0;
29 }
```

En cuanto al anidamiento de funciones, técnicamente no es posible en C. Las funciones en C no pueden definirse dentro de otras funciones. Cada función debe ser definida fuera de otras funciones, y solo pueden ser llamadas dentro de la función principal (main) u otras funciones después de ser declaradas.

Sin embargo, puedes tener funciones dentro de estructuras en C, conocidas como "funciones miembro". Esto se utiliza en el contexto de programación orientada a objetos en C, donde una estructura puede contener tanto datos como funciones que actúan sobre esos datos. Pero esta no es una forma típica de anidamiento de funciones en el sentido tradicional.

10.C

- Manejo de expresiones que brinda el lenguaje:
 - **Operadores aritméticos:** C admite operadores aritméticos estándar como suma (+), resta (-), multiplicación (*), división (/) y módulo (%).
 - **Operadores de asignación:** C utiliza el operador de asignación (=) para asignar valores a variables.
 - **Operadores de comparación:** C proporciona operadores de comparación como igualdad (==), desigualdad (!=), mayor que (>), menor que (<), mayor o igual que (>=) y menor o igual que (<=). Estos operadores se utilizan para comparar valores y producir resultados booleanos (verdadero o falso).
 - **Operadores lógicos:** C incluye operadores lógicos como AND (&&), OR (||) y NOT (!).
 - **Operadores de incremento y decremento:** C ofrece los operadores de incremento (++) y decremento (--), que se utilizan para aumentar o disminuir el valor de una variable en una unidad. Estos operadores pueden ser pre o post incremento/decremento.

11. Python, Ruby Y PHP

- Python: Python es un lenguaje de programación multiparadigma, lo que significa que soporta diferentes estilos de programación, pero su paradigma principal es el de programación orientada a objetos (POO). Sin embargo, también es muy utilizado en programación imperativa y funcional. Aquí hay algunos tipos de programas comunes que se pueden escribir con Python:
 - **Aplicaciones web:** Python es muy popular en el desarrollo web gracias a frameworks como Django y Flask.
 - **Análisis de datos y ciencia de datos:** Python es ampliamente utilizado en el ámbito de la ciencia de datos y el análisis de datos. Bibliotecas como NumPy, Pandas, y Matplotlib son fundamentales para este tipo de aplicaciones.
 - **Automatización y scripting:** Python es excelente para escribir scripts y automatizar tareas.

- **Ruby:** Ruby es conocido por su elegancia y facilidad de lectura, y se enfoca principalmente en la programación orientada a objetos. Aquí hay algunos tipos de programas comunes que se pueden escribir con Ruby:
 - **Desarrollo web:** Ruby on Rails es un framework web muy popular que utiliza Ruby como lenguaje principal. Se puede usar para desarrollar aplicaciones web de cualquier tipo.
 - **Automatización y scripting:** Al igual que Python, Ruby es excelente para escribir scripts y automatizar tareas.
 - **Desarrollo de juegos:** Aunque no es tan común como en otros lenguajes, Ruby también se puede utilizar para el desarrollo de juegos, especialmente juegos indie o juegos web simples.
- **PHP:** PHP es un lenguaje de programación ampliamente utilizado en el desarrollo web del lado del servidor. Su paradigma principal es la programación orientada a objetos, pero también es compatible con la programación imperativa y funcional. Aquí hay algunos tipos de programas comunes que se pueden escribir con PHP:
 - **Desarrollo web:** PHP es quizás más conocido por su uso en el desarrollo web. Se utiliza para crear aplicaciones web dinámicas.
 - **Aplicaciones empresariales:** PHP también se utiliza en el desarrollo de aplicaciones empresariales, como sistemas de gestión de recursos empresariales (ERP), sistemas de gestión de relaciones con el cliente (CRM), etc.
 - **Automatización de tareas:** PHP se puede utilizar para escribir scripts que automatizan tareas del lado del servidor, como procesamiento de formularios, generación de informes, envío de correos electrónicos, etc.
- **Características que determinan la pertenencia a cada paradigma:**
 - **Programación Orientada a Objetos:** La programación orientada a objetos se basa en la creación de objetos que contienen datos y métodos. Las características principales de la POO son:
 - **Abstracción:** La capacidad de modelar entidades del mundo real como objetos que tienen atributos y comportamientos.
 - **Encapsulamiento:** La ocultación de los detalles internos de un objeto y la exposición solo de las operaciones relevantes a otros objetos.
 - **Herencia:** La capacidad de una clase (objeto) de heredar atributos y métodos de otra clase.

- **Polimorfismo:** La capacidad de diferentes objetos de responder a la misma invocación de método de manera diferente.
- **Programación Imperativa:** La programación imperativa se centra en describir detalladamente cómo se deben realizar las operaciones. Las características principales son:
 - **Secuencia:** Las instrucciones se ejecutan en secuencia, una tras otra.
 - **Estructuras de control:** Uso de estructuras como bucles y condicionales para controlar el flujo de ejecución.
 - **Variables:** Las variables se utilizan para almacenar y manipular datos.
- **Programación Funcional:** La programación funcional se basa en funciones puras y evita los estados compartidos y los datos mutables. Las características principales son:
 - **Funciones con importancia:** Las funciones se pueden pasar como argumentos, asignar a variables y devolver como valores.
 - **Inmutabilidad:** Los datos son inmutables y no cambian una vez creados.
 - **Programación declarativa:** Se enfoca en qué se debe hacer en lugar de cómo hacerlo.

12. Python, Ruby, PHP, Gobstone y Processing

- **Tipado de datos de cada uno:**
 - **Python:** Tipado dinámico. Python es un lenguaje de tipado dinámico, lo que significa que el tipo de una variable se determina en tiempo de ejecución, no en tiempo de compilación.
 - **Ruby:** Tipado dinámico. Al igual que Python, Ruby es un lenguaje de tipado dinámico. Las variables pueden contener cualquier tipo de dato y su tipo se determina en tiempo de ejecución.
 - **PHP:** Tipado débil y dinámico. PHP es conocido por su tipado débil y dinámico. Esto significa que las conversiones de tipo son implícitas en muchas operaciones y que el tipo de una variable puede cambiar durante la ejecución del programa.
 - **Gobstone:** Tipado estático. Gobstone es un lenguaje de programación diseñado específicamente para enseñar conceptos de programación a niños. Utiliza un sistema de tipado estático, lo que significa que el tipo de cada variable

se debe declarar explícitamente y no puede cambiar durante la ejecución del programa.

- **Processing:** Tipado estático y dinámico. Processing es un lenguaje de programación que se utiliza principalmente para la creación de imágenes, animaciones y aplicaciones interactivas. Aunque su tipado puede considerarse principalmente dinámico debido a la forma en que se usa comúnmente, también puede ser estático en ciertos contextos.

13. Javascript

- JavaScript es un lenguaje de programación multiparadigma que abarca varios estilos de programación. En cuanto al tipo de lenguaje, JavaScript es un lenguaje de scripting interpretado principalmente ejecutado en el navegador del cliente. Sin embargo, también es utilizado en entornos de servidor a través de plataformas como Node.js. Además, se utiliza para el desarrollo de aplicaciones de escritorio y móviles a través de frameworks como Electron y React Native respectivamente. Por lo tanto, JavaScript es un lenguaje versátil que puede ser utilizado en una amplia gama de aplicaciones y entornos.

14. Javascript

- **Tipado de Datos Dinámico:** JavaScript es un lenguaje de tipado dinámico, lo que significa que las variables no están asociadas a un tipo de dato específico.
- **Manejo de Excepciones:** JavaScript admite el manejo de excepciones mediante la utilización de bloques try, catch y finally. Esto permite a los desarrolladores controlar y manejar errores que puedan ocurrir durante la ejecución del programa, lo que ayuda a mejorar la robustez y la confiabilidad del código.
- **Variables:** En JavaScript, las variables se pueden declarar utilizando las palabras clave var, let y const.
 - **var:** Se utilizaba antes de la introducción de let y const para declarar variables. Sin embargo, tiene alcance de función, lo que significa que su alcance está limitado al contexto de la función en la que se declara.
 - **let:** Introducido en ECMAScript 6, let permite declarar variables con alcance de bloque, lo que significa que su alcance está limitado al bloque en el que se declara.
 - **const:** También introducido en ECMAScript 6, const se utiliza para declarar variables cuyo valor no cambia durante la

ejecución del programa. Las variables declaradas con `const` también tienen alcance de bloque.