Práctica 1: Historia, evolución y características de Leng. de Programación

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 cuál de ellos la incorpora.

- 1951-1955: Uso de mnemotécnicos (ADD, STORE) para traducir a código máquina, específico por arquitectura. Assembly.
- 1956-1960: Lenguajes de alto nivel para cálculos numéricos y científicos. FORTRAN.
- 1961-1965: Portabilidad entre computadoras y orientación a negocios. COBOL.
- 1966-1970: Simplicidad y bajo uso de memoria en sistemas de tiempo compartido. BASIC.
- 1971-1975: Programación estructurada para aprendizaje (Pascal) y declarativa basada en reglas (Prolog). Pascal y Prolog.
- 1976-1980: Orientación a objetos y tipado fuerte para seguridad y fiabilidad. Ada.
- 1981-1985: Entorno de desarrollo integrado (IDE) con compilador. Turbo Pascal.
- 1986-1990: Extensión de C con orientación a objetos y abstracción. C++.
- 1991-1995: Manipulación de texto (Perl) y estructura para páginas web (HTML). Perl y HTML.
- 1996-2000: Portabilidad, concurrencia y orientación a objetos con mínimas dependencias. Java.

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

El lenguaje Java fue desarrollado en sus inicios por James Gosling, en el año 1991. Inicialmente Java era conocido como Oak o Green. La primera versión del lenguaje Java es publicada por Sun Microsystems en 1995. Y es en la versión del lenguaje JDK 1.0.2, cuando pasa a llamarse Java, corría el año 1996.

En las primeras versiones de Java 1.1, 1.2 y 1.3 es en la que el lenguaje va tomando forma, con la inclusión de tecnologías como JavaBeans, JDBC para el acceso a base de datos, RMI para las invocaciones en remoto, Collections para la gestión de múltiples estructuras de datos o AWT para el desarrollo gráfico, entre otros.

Hoy en día, Java sigue siendo un pilar en la programación moderna, especialmente tras la adquisición de Sun por Oracle en 2010. Su Máquina Virtual Java (JVM) garantiza portabilidad en sistemas diversos, desde servidores empresariales hasta dispositivos móviles (como Android). Con versiones recientes como Java 21 (2023), incorpora mejoras en rendimiento, concurrencia y sintaxis.

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.

SIMPLICIDAD Y LEGIBILIDAD: Python, Ruby y JavaScript

- Poder producir programas fáciles de escribir y de leer.

- Resultar fáciles a la hora de aprenderlo o enseñarlo.
- La estructura subyacente del algoritmo y los datos que el programa representa deben quedan en manifiesto al inspeccionar el texto del programa.

CLARIDAD EN LOS BINDINGS: Rush y Python

- Definición del lenguaje.
- Implementación del lenguaje.
- En escritura del programa.
- Compilación.
- Cargado del programa.
- En ejecución.

CONFIABILIDAD: Ada o Rust

- Chequeo de tipos
    - Cuanto antes se encuentren errores menos costosos resulta realizar los arreglos que se requieran.
- Manejo de excepciones
    - La habilidad para interceptar errores en tiempo de ejecución, tomar medidas correctivas y continuar.

SOPORTE: Javascript, Java, C#

- Debería ser accesible para cualquiera que quiera usarlo o instalarlo.
    - Lo ideal sería 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: tutoriales, cursos textos, etc.

ABSTRACCIÓN: Java, Python

- Capacidad de definir y usar estructuras u operaciones complicadas de manera que sea posible ignorar muchos de los detalles.
- Concepto clave para manejar la complejidad, abstracción de procesos y de datos.

ORTOGONALIDAD: Python

- Significa que un conjunto pequeño de constructores primitivos puede ser combinado en número relativamente pequeño a la hora de construir estructuras de control y datos. Cada combinación es legal y con sentido.

EFICIENCIA: C, C++, Rust, Go

- Tiempo y Espacio: Capacidad de un programa para realizar una tarea de manera correcta y consumiendo pocos recursos, de memoria, espacio en disco, tiempo de ejecución, tráfico de red, etc.
- Esfuerzo humano: Que mejore el rendimiento del programador.
- Optimizable: Capacidad del lenguaje de programación de venir optimizado para tareas específicas.

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

- Describa los tipos de expresiones que se pueden escribir en él/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.

Java

Tipos de expresiones:

- Expresiones aritméticas: estas expresiones implican operaciones aritméticas como suma, resta, multiplicación y división. Por ejemplo, 2+3 es una expresión aritmética que se evalúa como 5.
- Expresiones relacionales: estas expresiones implican comparar dos valores utilizando operadores relacionales como "mayor que", "menor que", "igual a" y "diferente a". Por ejemplo, 4<5 es una expresión relacional que se evalúa como verdadera.
- Expresiones lógicas: estas expresiones implican operaciones lógicas como "Y", "O" y "NOT". Por ejemplo, (2<3)&&(3<4) es una expresión lógica que se evalúa como verdadera.
- Expresiones condicionales: estas expresiones implican el uso del operador ternario "?:" para asignar un valor según una condición. Por ejemplo, int x = (5<6)?2:3; asigna el valor 2 a x ya que 5 es menor que 6.
- Expresiones de asignación: estas expresiones implican asignar un valor a una variable. Por ejemplo, int x = 5; asigna el valor 5 a la variable x.

Facilidades provistas para la organización del programa:

- Paquetes (Packages): Permiten agrupar clases relacionadas en un espacio de nombres, evitando conflictos y facilitando la reutilización.
- Clases y Objetos: Como lenguaje orientado a objetos, Java organiza el código en clases que encapsulan datos y comportamiento. Se promueve el principio de "ocultar información" mediante modificadores como “private” o “public”.
- Herencia: Permite reutilizar código mediante la extensión de clases (usando “extends”). Una clase hija hereda atributos y métodos de una clase padre.
- Interfaces: Definen contratos que las clases deben cumplir (con `implements`), promoviendo un diseño modular y flexible.
- Colecciones (Collections Framework): Incluidas desde Java 1.2, ofrecen estructuras de datos predefinidas como “List”, “Set” y “Map” (en java.util).
- Métodos: Los métodos organizan la lógica en bloques reutilizables.
- Excepciones: Con “try”, “catch” y “throw”, Java organiza el manejo de errores de forma estructurada, separando la lógica normal de las situaciones excepcionales.

¿Cuáles de los atributos del ejercicio anterior posee y cuáles no posee?

- Simplicidad y legibilidad: Sí posee. Java fue diseñado para ser más simple que lenguajes como C++ (eliminando punteros manuales y gestión explícita de memoria). Su sintaxis es clara y estructurada, lo que facilita la lectura, especialmente con conceptos como clases y paquetes.
- Claridad en los bindings: Sí posee. En Java, los enlaces (bindings) entre nombres y entidades (variables, métodos) son estáticos y claros, definidos en tiempo de compilación gracias a su tipado fuerte.
- Confiabilidad: Sí posee. Java es altamente confiable debido a su manejo automático de memoria (recolector de basura), chequeo de tipos en compilación y ejecución, y un sistema de excepciones que previene fallos inesperados.
- Soporte: Sí posee. Java cuenta con una comunidad activa, documentación extensa de Oracle, y una gran cantidad de bibliotecas y frameworks (como Spring o Hibernate), además de actualizaciones regulares.
- Abstracción: Sí posee. Java soporta abstracción mediante clases abstractas e interfaces, permitiendo ocultar detalles de implementación y enfocarse en funcionalidades esenciales.
- Ortogonalidad: No posee completamente. Java tiene ciertas dependencias y reglas (como herencia única de clases o la necesidad de clases para todo), lo que limita esta propiedad.

- Eficiencia: Sí posee, pero es peor que otros lenguajes. Java es eficiente gracias a la JVM, que optimiza el código en tiempo de ejecución (JIT compilation), pero no alcanza la velocidad de lenguajes de bajo nivel como C debido a la capa de abstracción y el recolector de basura.

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

Tipos de datos: En ADA, cada variable debe ser declarada con un tipo específico y este tipo no puede cambiar durante la ejecución del programa. Algunos de los tipos de datos disponibles en ADA son:

- Enteros (integer)
- Caracteres (character)
- Booleanos (boolean)
- Reales (real)
- Punteros (access)
- Arreglos (array)
- Registros (record)
- Variantes (variant)
- Enumeraciones
- Tipos derivados

Tipos abstractos de datos - paquetes: Los tipos de datos abstractos en ADA se definen por el usuario y se crean mediante la combinación de tipos de datos existentes. Los paquetes en ADA permiten encapsular datos y procedimientos relacionados en un único módulo.

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). Pueden contener variables privadas y constantes que son accesibles sólo dentro del paquete.

Estructuras de datos:

- Arrays: Pueden tener índices dinámicos o estáticos, con chequeo de límites en tiempo de ejecución.
- Registros: Similares a estructuras en C, con campos y posibles discriminantes para variantes.
- Punteros (Access Types): Gestionados de forma segura, evitando referencias nulas no deseadas.
- Colas (queues)
- Árboles (trees)
- Conjuntos (sets)
- Mapas (maps)

Manejo de excepciones: ADA maneja excepciones mediante bloques try-catch. Cuando se lanza una excepción, el programa busca el bloque catch correspondiente que maneje la excepción. Si se encuentra un bloque catch correspondiente, el código dentro del bloque se ejecutará. Si no se encuentra un bloque catch correspondiente, el programa se cerrará con un mensaje de error. Los bloques try-catch pueden anidarse para manejar excepciones específicas en diferentes niveles de la jerarquía del programa.

En ADA, las excepciones se declaran en la especificación del paquete utilizando la cláusula “exception". Además de declarar excepciones, los programadores pueden proporcionar información adicional sobre la excepción, como una descripción o una razón para su ocurrencia.

Manejo de concurrencia:

- 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.
- Se proporcionan 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.

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? ¿Por qué?

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. 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.

Java permite "escribir una vez, ejecutar en cualquier lugar" gracias a la JVM, que ejecuta código compilado en cualquier plataforma compatible. Sin embargo, la JVM es específica de cada sistema operativo y hardware, haciendo que Java sea portátil a nivel de código fuente, pero dependiente de una implementación JVM adecuada para cada entorno.

Ejercicio 7: ¿Sobre qué lenguajes está basado?

Java está basado C/C++, Smalltalk y Objective-C.

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

Los applets son pequeñas aplicaciones escritas en el lenguaje de programación Java que se ejecutan para incluirse en un documento web HTML. Estos programas permiten a los desarrolladores web crear aplicaciones interactivas y dinámicas en las páginas web sin la necesidad de instalar software adicional en el equipo del usuario.

Son programas Java que se ejecutan en el servidor web utilizando la interfaz de programación de aplicaciones (API) de servlet Java,y generan dinámicamente contenido web. A diferencia de los 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.

Lenguajes - C

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

Estructura de un programa escrito en C:

1. Directivas de preprocesador: Incluyen archivos de cabecera (#include <stdio.h>) y definiciones de macros (#define), procesadas antes de la compilación.
2. Declaraciones globales: Variables o funciones definidas fuera de cualquier función, accesibles en todo el programa.
3. Función principal (main): Punto de entrada del programa, donde comienza la ejecución.
4. Definiciones de funciones: Funciones adicionales definidas por el programador, que se llaman desde main u otras funciones.

No existe anidamiento de funciones en C en el sentido estricto. Esto significa que no se pueden definir funciones dentro de otras funciones. Cada función en C debe definirse en el ámbito global o dentro de un archivo, de forma independiente.

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

- Expresiones aritméticas: Combinan operadores como +, -, *, / y % (módulo).
- Expresiones relacionales: Comparan valores con operadores como <, >, <=, >=, ==, !=.
- Expresiones lógicas: Usan && (AND), || (OR) y ! (NOT) para combinar condiciones.
- Expresiones de asignación: Asignan valores con = o operadores compuestos como +=, -=, etc.
- Expresiones de bits: Operan a nivel binario con & (AND), | (OR), ^ (XOR), ~ (NOT), << (desplazamiento izquierda) y >> (desplazamiento derecha).
- Expresiones condicionales: Usan el operador ternario ?:.

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?

Lenguaje	Tipos de programas	Paradigmas principales	Características clave
Python	Web, ciencia de datos, automatización	Imperativa, POO, funcional	Sintaxis clara, clases, lambdas
Ruby	Web (Rails), scripts, protitipos	POO, imperativa, funcional ligero	Todo es objeto, bloques, flexibilidad
PHP	Web del servidor, scripts dinámicos	Imperativa, POO, procedimental	Funciones, clases modernas, foco en web

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

<u>Python</u>

- Tipado: Dinámico y fuerte; no necesitas declarar tipos (ej. x = 5), pero respeta estrictamente las operaciones entre tipos.
- Organización: Programas basados en módulos y paquetes (archivos .py importados con import), con soporte para clases (POO) y funciones.
- Otras: Interpretado, multiplatforma, énfasis en indentación para definir bloques de código (sin llaves ni end).

<u>Ruby</u>

- Tipado: Dinámico y fuerte; las variables no tienen tipo fijo (ej. x = 5; x = "hola"), pero los métodos respetan tipos.
- Organización: Todo es un objeto; usa clases y módulos para estructurar el código, con bloques (do ... end) para lógica flexible.

- Otras: Interpretado, sintaxis expresiva, diseñado para ser intuitivo y "humano".

PHP

- Tipado: Dinámico y débil; permite conversiones automáticas (ej. "5" + 3 = 8), pero desde PHP 7 soporta declaraciones de tipos opcionales.
- Organización: Tradicionalmente procedimental (funciones en scripts), pero moderno con clases y namespaces para proyectos grandes.
- Otras: Interpretado, embebido en HTML para web, ejecución del lado del servidor, gran integración con bases de datos.

Gobstone

- Tipado: Simple y no explícito; enfocado en enseñanza, los tipos (números, colores) son implícitos y manejados por el entorno.
- Organización: Programas lineales con comandos básicos (ej. Poner(Rojo)), estructurados como secuencias o bloques visuales.
- Otras: Interpretado, entorno gráfico interactivo, diseñado para aprender lógica y algoritmos sin complejidad técnica.

Processing

- Tipado: Estático y basado en Java; requiere declarar tipos (ej. int x = 5), heredado de su base en Java.
- Organización: Estructura basada en funciones como setup() y draw(), orientada a bucles gráficos continuos.
- Otras: Compilado a Java, enfocado en visualización, entorno integrado (IDE) para artistas con ejecución en tiempo real.

Lenguaje JavaScript

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

JavaScript es un lenguaje de programación que admite múltiples paradigmas de programación, incluyendo programación orientada a objetos, programación funcional y programación imperativa. JavaScript es utilizado principalmente para el desarrollo de aplicaciones web, tanto en el lado del cliente como en el del servidor.

En su uso más común, JavaScript se utiliza como lenguaje de programación orientado a objetos, utilizando su sistema de prototipos para crear objetos y herencia. Además, JavaScript también admite técnicas de programación funcional, lo que permite a los desarrolladores escribir código en términos de funciones y expresiones en lugar de objetos y métodos.

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

- 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.
- 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, es decir 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.