Programa Profesional en Inteligencia Artificial y Data Science

Módulo 2. Lenguajes y desarrollo de soluciones IA

Índice

3.7. JavaScript

3.8. Prolog

3.9. Otros lenguajes y tecnologías

Tema 4. Lenguajes de marcado. Información de sus etiquetas

4.1. Introducción y objetivos

4.2. Los lenguajes de marcado

4.3. Tipos de lenguajes de marcado

4.4. HTML

4.5. XML

4.6. XHTML

4.7. LaTeX

4.8. MarkDown

4.9. Otras tecnologías

Tema 1. Programas informáticos y lenguajes de programación

1.1. Introducción y objetivos

Para estudiar este tema lee las Ideas Clave que se presentan a continuación y revisa los contenidos adicionales para mejorar la comprensión y aumentar los conocimientos sobre las materias tratadas.

Este tema supone la primera introducción a las bases de la programación, los algoritmos, los programas y los lenguajes de programación. Objetivos más concretos son:

- Conocer qué son los algoritmos y el papel que juegan en la construcción de aplicaciones software.
- Tener un acercamiento a ejemplos reales de pseudocódigo.
- Conocer cómo se representan visualmente los algoritmos por medio de diagramas.
- Aplicar las bases del pensamiento computacional.
- Categorizar los distintos tipos de programas software.
- Conocer algunos de los términos principales en el ámbito.

Tema 1. Programas informáticos y lenguajes de programación

1.2. Algoritmos

La palabra algoritmo, uno de los términos tecnológicos de moda, frecuentemente, es usado con cierto desconocimiento cuando hablamos de la **inteligencia artificial** o de conceptos cercanos.

Un **algoritmo** es, sencillamente, una secuencia ordenada y finita de operaciones sencillas a través de la cual podemos determinar la solución a un problema.

Los algoritmos nos permiten ejecutar una **acción** o **resolver un problema** mediante una serie de instrucciones definidas, ordenadas y finitas.

Basándonos en esta definición, tanto una receta de cocina como el cálculo de una raíz cuadrada de un número o un protocolo sanitario concreto son algoritmos.

Un algoritmo es una secuencia de instrucciones finitas que llevan a cabo un conjunto de procesos de forma lógica, sistemática y ordenada para dar respuesta a un determinado problema o propósito, con lo que un algoritmo resuelve un problema concreto a través de una serie finita de una o más instrucciones definidas, concisas y ordenadas.

Características de los algoritmos

Las **características** básicas de un algoritmo son:

- **Serie finita:** tiene inicio y fin, todo algoritmo comienza en un estado inicial con una serie de datos específicos, y culmina con una solución o salida.
- **Secuencialidad:** un algoritmo está compuesto por una serie de pasos ordenados.
- **Sin ambigüedad:** las secuencias son concretas, cada paso es claro y no deja lugar a ambigüedad.

- **Validez.** Un algoritmo es válido si carece de errores. Un algoritmo puede resolver el problema para el que se planteó y sin embargo no ser válido debido a que posee errores.
- **Eficiencia.** Un algoritmo es eficiente si obtiene la solución al problema en poco tiempo. No lo es si es lento en obtener el resultado.
- **Optimización.** Un algoritmo es óptimo si es el más eficiente posible y no contiene errores. La búsqueda de este algoritmo es el objetivo prioritario del programador. No siempre podemos garantizar que el algoritmo hallado es el óptimo, a veces sí.
- Ser **independiente** de la computadora. Los algoritmos deben escribirse para poder ser utilizados por cualquier máquina.
- Ser **preciso.** Los resultados de los cálculos deben de ser exactos, no solo aproximarse a la solución.
- Debe poder **repetirse.** Los algoritmos deben permitir su ejecución tantas veces como sea necesario, además de resolver el problema cambiando los datos de entrada.

Condiciones que deben cumplir los algoritmos:

- **Finitud** del número de acciones elementales.
- **Definibilidad**. Cada paso debe efectuarse de forma precisa. Precisión en el leguaje: pseudocódigo.
- **Entrada**. Deben estar especificados los datos a los que se puede aplicar.
- **Salida**. Debe especificarse el conjunto de todas las salidas que pueden producirse.
- **Efectividad**. Todos los procesos deben ejecutarse en un tiempo finito.

Tema 1. Programas informáticos y lenguajes de programación

Algoritmia

La **algoritmia** es el tratamiento sistemático de técnicas para el diseño y análisis de algoritmos eficientes, que faciliten el desarrollo de programas de ordenador. Comprende:

- Estudio conceptual
- Criterios de evaluación

Estudio conceptual apoyándose en:

- La modelización de los datos (árboles, grafos, ...)
- Los modelos de máquinas (Turing, Von Neumann).
- Los métodos (divide y vencerás, recursión, ...)
- Estudio algoritmos clásicos en modelos discretos (palabras, grafos, ...)

Conocimiento **criterios de evaluación:**

- Las prestaciones de los algoritmos (complejidad).
- Su corrección (verificación).
- Su expresión en un lenguaje adecuado (Java, C, Python...)

Visualización de algoritmos

Los **diagramas de flujo** son representaciones gráficas que muestran los pasos de un proceso. Permiten representar procesos computacionales con el fin de comunicar y debatir sobre los mismos en las etapas previas al desarrollo de un código.

Tema 1. Programas informáticos y lenguajes de programación

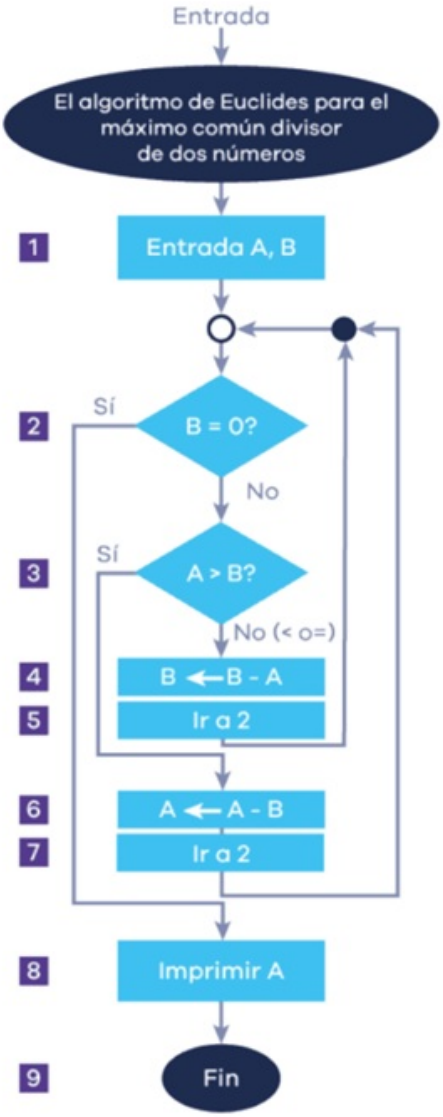

Ilustración 1. Diagrama de flujo ejemplo de la secuencia de pasos de un algoritmo

Los diagramas de flujo siguen unas **normas,** como el uso de rectángulos para las instrucciones y diamantes para las decisiones, utilizando flechas para conectar unos pasos con otros, etc.

Uno de los objetivos de los diagramas de flujo es ayudar a definir la entrada, el proceso y la salida de los algoritmos.

Tema 1. Programas informáticos y lenguajes de programación

Los diagramas de flujo ayudan a los programadores a **visualizar los pasos** de un algoritmo. Al igual que hacer un esquema antes de escribir un ensayo, los diagramas de flujo ayudan a organizar las ideas y usan símbolos específicos para representar diferentes partes de un algoritmo.

Los diagramas de flujo ayudan a **organizar las ideas** definiendo la entrada, el proceso y la salida de los algoritmos, a la vez que siguen unas normas, para las que usan símbolos específicos que representan las diferentes partes del algoritmo. El significado de los símbolos es el siguiente:

- Óvalo: comienzo o final del programa.
- Paralelogramo: operación de entrada.
- Rectángulo: instrucción o proceso a realizar.
- Diamante: decisión a tomar (el programa debe continuar por una de las dos rutas).
- Híbrido: operación de salida.
- Flechas: conectan unos pasos con otros e indican la dirección del flujo.

	Óvalo	Comienzo o final del programa
	Paralelogramo	Operación de entrada
	Rectángulo	Instrucción o proceso a realizar
	Diamante	Decisión a tomar (el programa debe continuar por una de las dos rutas)
	Híbrido	Operación de salida
	Flechas	Conectan unos pasos con otros e indican la dirección del flujo

Notación algorítmica

Debe recoger las reglas de descripción de los objetos manipulados (informaciones de léxico), las operaciones puestas en juego (acciones del léxico) y el modo de organizar estas acciones en el tiempo (primitivas de control).

Acciones posibles:

- Acción elemental
- Composición secuencial
- Composición condicional.
- Composición alternativa.
- Composición selectiva.
- Composición iterativa.
- Acciones con nombres

Notación para cada posible acción

Acción elemental

acción elemental

Composición secuencial

acción primera

acción segunda

...

acción n-ésima

Composición condicional

si condición entonces

acción si

fin si

Composición alternativa

si condición entonces

acción si

si no

acción no

fin si

Composición selectiva

según indicador hacer

valor primero: acción primera

valor segundo: acción segunda

...

valor último: acción última

fin según

Composición iterativa

1. repetir ... hasta

repetir

acción

hasta condición.

2. mientras ... hacer

mientras condición hacer

acción

fin mientras

3. para ...

para índice = valor inicial hasta índice = valor final hacer

acción

fin para

Acciones con nombre

acción nombre es

acciónprimera

acciónsegunda

...

acciónúltima

finacciónnombre

Partes de un algoritmo

La entrada de un algoritmo

Son los datos con los que el algoritmo opera. Ejemplos:

- Si se trata de una receta, la entrada serían los ingredientes.
- Si se trata de un algoritmo informático, la entrada sería, por ejemplo, un conjunto de datos.

El proceso del algoritmo

Son los pasos y operaciones que sigue el algoritmo utilizando o no los datos de entrada:

- Si se trata de una receta, los diferentes pasos definidos y en orden que se han establecido
- Si se trata de un algoritmo informático, el proceso es la combinación ordenada de operaciones matemáticas, algebraicas, etc. que se han establecido por parte de los autores: sumas, restas, comparaciones, otros algoritmos, etc.

La salida del algoritmo

Es el resultado que entrega el algoritmo:

- Si se trata de una receta, la salida es el plato establecido por parte del autor.
- Si se trata de un algoritmo informático, la salida es el dato o conjunto de datos calculados esperados o un error

Los algoritmos recogen **operaciones** tan sencillas que pueden ser realizadas con éxito por cualquiera, incluso por máquinas: una máquina, ordenador, dispositivo, etc. puede ejecutar de forma automática y en tiempos ínfimos cualquier algoritmo sencillo (sumar, ordenar, repetir condiciones sencillas, etc.), y, por tanto, puede ejecutar cualquier algoritmo complejo siempre que pueda definirse como una combinación ordenada y concisa de algoritmos sencillos (divide y vencerás).

De esta manera las máquinas, ordenadores, etc., disponiendo de una mínima capacidad, pueden realizar casi cualquier operación por compleja que sea si nosotros la podemos **definir en términos** de otras **más simples.**

Tema 1. Programas informáticos y lenguajes de programación

El trabajo de los programadores informáticos, el cual consiste en traducir determinados problemas del mundo, entorno, ciencia o empresa a un lenguaje o lenguajes que una máquina pueda entender, es decir, en algoritmos que una máquina pueda ejecutar: para ello hay que describir una realidad en pequeños problemas simples en sucesión y poner al ordenador/dispositivo a la tarea

En tanto en cuanto se disponga del talento para la programación de algoritmos, así como la capacidad de procesamiento y la capacidad para guardar y disponer de datos, las posibilidades para resolver problemas se multiplican exponencialmente.

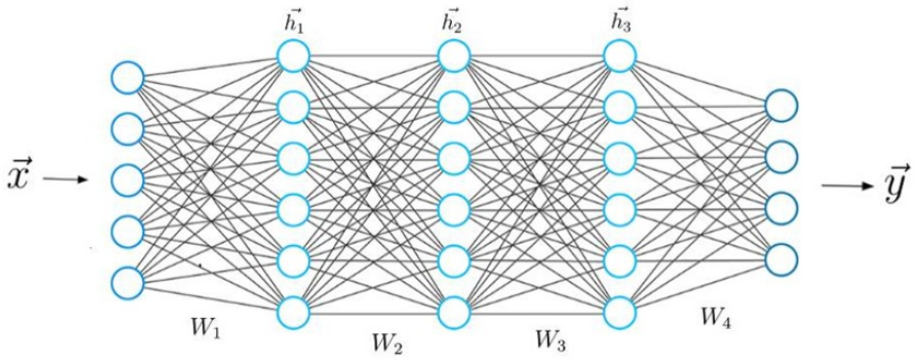

Ilustración 2. Ejemplo de red neuronal como algoritmo que combina cientos, miles o millones de neuronas, que recibiendo una entrada (datos en cualquier de sus formas) generan una salida.

Tal es la capacidad que la sociedad está alcanzando en las últimas décadas, tanto de desarrollar algoritmos complejos como de lograr capacidades asombrosas de procesamiento y almacenamiento de datos (arquitecturas de ordenadores en la nube, grandes capacidades de almacenamiento en bases de datos cada vez más veloces y un largo etc.), que realmente se consiguen resolver problemas verdaderamente retadores y ambiciosos.

Tema 1. Programas informáticos y lenguajes de programación

Problemas tan retadores y complejos que a veces es enormemente complicado poder describir lo que un algoritmo hace y cómo lo hace (interpretabilidad, explicabilidad) y si tienen un comportamiento ético.

Ejemplos de ello no dejan de surgir en los últimos tiempos en los ámbitos de las redes sociales y, en general, en cualquier ámbito donde implicamos evoluciones tecnológicas, entre ellos:

- Transporte (coches autónomos)
- Salud (genética, prevención, etc.)
- finanzas
- ciencia en general
- periodismo

Todo ello también genera el gran reto de evolucionar el marco regulatorio o legal para dar alcance al uso de los algoritmos para dar equilibrio a la utilidad que ofrecen y al coste o impacto que puede suponer su uso, privacidad, sesgos, etc.

Existen diferentes tipos de problemas: ordenar datos, buscar patrones, encontrar la ruta óptima, etc. A su vez, existen múltiples tipos de algoritmos para resolver problemas en diferentes campos de estudio: procesamiento de imágenes, criptografía, inteligencia artificial.

Cuando una solución se lleva a cabo mediante un algoritmo, la solución se vuelve **reutilizable**. Uno de los objetivos de la programación es el desarrollo de software reusable, que pueda ser reutilizado en multitud de proyectos reduciendo así los costes operativos.

1.3. Pseudocódigo

El **pseudocódigo** es una versión simplificada de un código de programación en lenguaje plano utilizando frases cortas para escribir un código para un programa antes de ser implementado en un lenguaje de programación específico.

El pseudocódigo permite expresar **procesos computacionales.** Este tipo de código es cercano al ser humano (human-friendly) y no entendible por una máquina. Existen determinadas pautas sobre cómo escribir un pseudocódigo, pero está destinado a ser leído por humanos, no por una computadora.

El **pseudocódigo es formateado** para que cada línea represente una línea de código real en su programa final. El pseudocódigo no tiene por qué ser perfecto; solo debería dar una idea de cómo se verá el programa final.

Existen multitud de herramientas para asistir en la programación de pseudocódigo con fines docentes:

- PseInt
- PseudoEditor

PSeInt

PSeInt es una herramienta (libre, gratuita y multiplataforma) que permite escribir algoritmos de forma intuitiva mediante pseudolenguaje en español, cuyo objetivo es el aprendizaje de conceptos fundamentales, pero no el de los detalles del lenguaje de programación. Así, permite ejecutar y modificar el algoritmo (o parte del mismo) para observar su funcionamiento y verificar los resultados.

Cabe destacar que detecta errores y ofrece sugerencias para corregirlos. Además, PSeInt permite trabajar con diagramas de flujo, convirtiendo automáticamente los algoritmos entre una y otra representación, así como editarlos en cualquiera de los formatos.

https://sourceforge.net/projects/pseint/

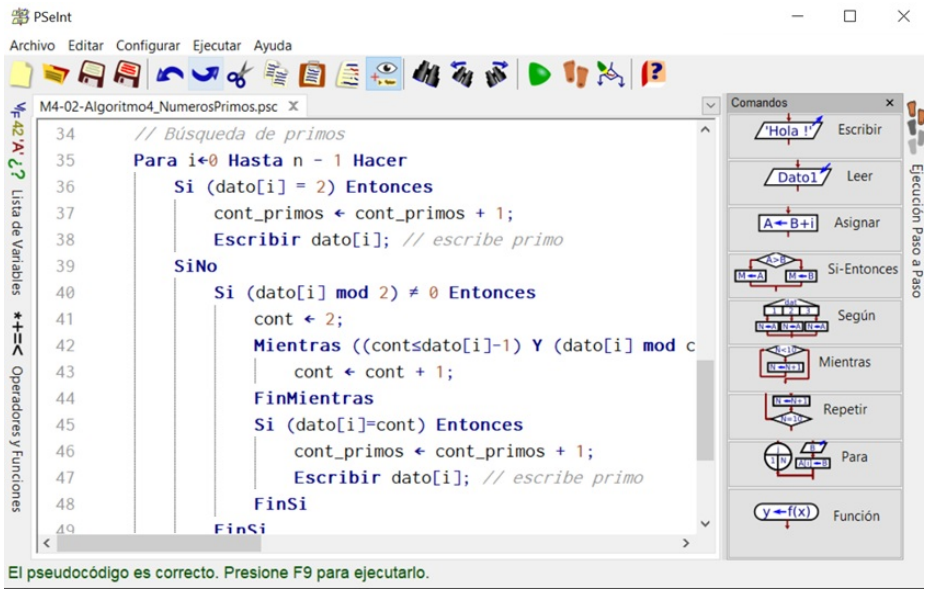

Ilustración 3. Ejemplo de algoritmo para la búsqueda de números primos realizado con pseudocódigo sobre la herramienta PSeInt.

PseudoEditor

PseudoEditor es un editor de pseudocódigo online en inglés y gratuito. Entre sus características se encuentra el resaltado de sintaxis dinámica para palabras clave, funciones, tipos de datos, etc., lo que ayuda a escribir y depurar el pseudocódigo de forma rápida.

Adicionalmente, permite el almacenamiento en la nube y destaca los errores, entre otras funciones.

Tema 1. Programas informáticos y lenguajes de programación

1.4. Programas informáticos

Un **programa** es el código escrito para resolver un problema siguiendo las reglas sintácticas de un lenguaje de programación.

Un **programa de ordenador** es un conjunto de sentencias escrito en un lenguaje de programación que acaba convertido en decenas, cientos o miles de sencillas operaciones que se realizan en base a señales eléctricas manejadas por los circuitos electrónicos de los dispositivos (procesadores y otros).

Los programas dicen a las máquinas lo que deben hacer. Un programa puede ser desde lo más sencillo, por ejemplo, un programa que imprima el mensaje "Bienvenidos" hasta lo más complejo, un navegador, un sistema operativo, etc.

El conjunto de programas informáticos y los datos utilizados por los mismos se conoce como **software.**

Mientras que los algoritmos pueden ser aplicados en el entorno físico en la vida real (una receta de cocina) los **programas informáticos** precisan de una **máquina** para ser ejecutados. Un programa puede contener o no algoritmos y llevar a cabo más tareas, como por ejemplo comunicarse con otro programa, reaccionar ante eventos, recibir información del usuario, etc.

Los programas informáticos consumen **recursos computacionales** del sistema informático en el que se ejecutan: memoria RAM, procesamiento, espacio en disco, etc. Es por tanto que una máquina puede ejecutar un número limitado de programas al mismo tiempo.

Tema 1. Programas informáticos y lenguajes de programación

Un programa informático puede ser ejecutado en diferentes **tipos de dispositivos,** algunos de ellos:

- Un ordenador personal
- Una estación de trabajo
- Un servidor
- Un teléfono inteligente
- Un dispositivo IoT (Internet de las cosas)

Tema 1. Programas informáticos y lenguajes de programación

1.5. Lenguajes de programación

Los **lenguajes de programación** expresan cálculos de una forma comprensible tanto para personas como para máquinas, permiten dar instrucciones a éstas en un lenguaje que entienden, además de procesar de forma rápida y eficiente grandes y complejas cantidades de información.

Todos los **lenguajes de programación** que se pueden utilizar para comunicarse con las máquinas son reducidos al lenguaje más básico de todos, el único que las máquinas entienden: el lenguaje "**binario**" o “código máquina", el cual es la reducción de cualquier lenguaje a secuencias de ceros y unos.

Este **código binario** o **máquina** recibe su nombre por cómo las máquinas procesan la información, dado que lo hacen mediante circuitos eléctricos que pueden encenderse y apagarse. Se usan los dígitos 1 y 0 para representar encendido y apagado, respectivamente.

La **CPU** de un ordenador solo puede leer instrucciones escritas en binario, por lo que cuando una persona introduce un programa en una computadora, otro programa lo convierte en código binario. A este proceso de traducción de un programa a código máquina se le llama “**compilación**”.

Existen cientos de lenguajes de programación de uso generalizado, cada uno con sus propias complejidades e idiosincrasias.

Dependiendo de los fines e intereses del proyecto de software, un equipo tecnológico elegirá uno u otro, dependiendo de si va a desarrollar una página web, un videojuego, una aplicación, etc.

Tema 1. Programas informáticos y lenguajes de programación

Todos los años el **IEEE** (Institute of Electrical and Electronics Engineers) elabora un ranking de **lenguajes de programación de mayor relevancia** en la industria del software.

En el ranking, cada lenguaje de programación recibe una puntuación de 0 a 100 según la demanda y el número de consultas de búsqueda y menciones de este, siendo el lenguaje de programación con la puntuación más alta el que ocupa el primer lugar. En el año 2021 el ranking ha sido el siguiente:

Rank	Language	Type	Score
1	Python		100.0
2	Java		95.4
3	C		94.7
4	C++		92.4
5	JavaScript		88.1
6	C#		82.4
7	R		81.7
8	Go		77.7
9	HTML		75.4
10	Swift		70.4

Ilustración 4. Ranking de lenguajes de programación según IEEE. https://spectrum.ieee.org/top-programming-languages/#toggle-gdpr

De la misma manera, el índice TIOBE muestra un listado actualizado mensualmente de los **lenguajes de programación más populares** (aquellos para los cuales se han escrito más líneas de código). A continuación, una lista de los cinco más populares en octubre de 2021:

Oct 2021	Oct 2020	Change	Programming Language	Ratings	Change
1	3	▲	Python	11.27%	-0.00%
2	1	▼	C	11.16%	-5.79%
3	2	▼	Java	10.46%	-2.11%
4	4		C++	7.50%	+0.57%
5	5		C#	5.26%	+1.10%

Ilustración 5. Ranking de lenguajes de programación según TIOBE. https://www.tiobe.com/tiobe-index/

Por otra parte, la encuesta anual de JetBrains 'El estado de ecosistema de desarrollo' muestra los **lenguajes de programación más utilizados,** así como frameworks y herramientas relacionadas con cada uno. También comparten conclusiones clave derivadas del análisis de los lenguajes y la evolución de los mismos.

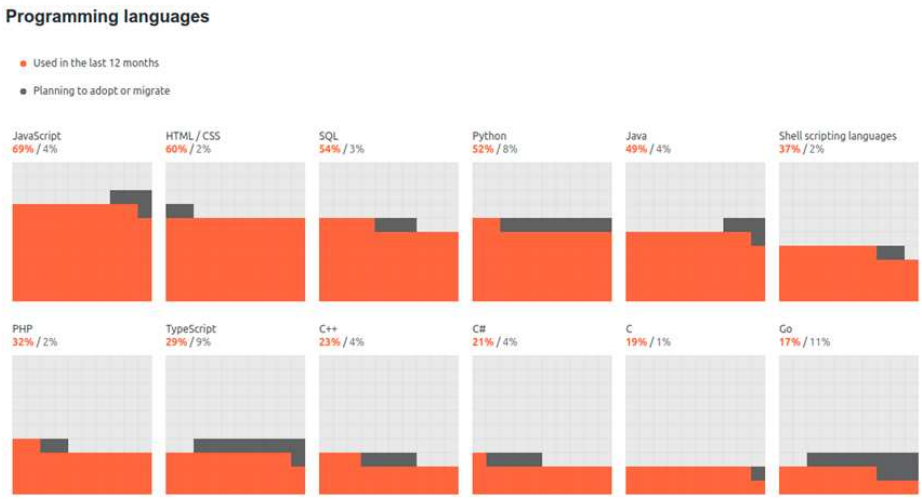

Ilustración 6. Ranking de lenguajes de programación según JetBrains.

https://www.jetbrains.com/lp/devecosystem-2021/

Tema 1. Programas informáticos y lenguajes de programación

Pensamiento computacional

El **pensamiento computacional** es el proceso de pensamiento para crear soluciones que pueden ser llevadas a cabo por una computadora.

Un ejemplo podría ser desarrollar un programa de inteligencia artificial (IA) que pueda comprender el lenguaje natural (la forma en que los humanos hablan entre ellos), analizar un libro y escribir un ensayo sobre el mismo.

En computación, la inteligencia artificial (IA) es una forma eficaz de programar máquinas para que puedan aprender de grandes cantidades de datos para mejorar la forma en que funcionan. Este tipo de computación permite que los programas puedan hacer predicciones y tomar decisiones más allá de aquellas para las que están programados directamente.

El pensamiento computacional se compone de diferentes elementos (descomposición, reconocimiento de patrones, abstracción, diseño de algoritmos, etc.) que se detallan a continuación.

Descomposición

La **descomposición** consiste en dividir un problema en partes más sencillas hace que los problemas sean más manejables y fáciles de resolver.

Para poder encontrar una solución satisfactoria a un problema, en primer lugar, se deben comprender todas las partes del mismo.

Una vez se comprenda cada parte, se podrá dividir el problema en tareas más simples (descomposición).

Reconocimiento de patrones

Identificar elementos comunes, es decir, lo que tienen en común distintos problemas.

Los patrones son eventos que se repiten. Reconocer dónde se han creado soluciones para problemas similares antes será útil a la hora de crear soluciones que puedan usarse para completar diferentes tareas.

Abstracción

Separar los detalles que importan de los que no.

La abstracción se centra en las ideas importantes de un problema e ignora los detalles que no son de ayuda a la hora de encontrar una solución.

De esta forma la abstracción reducir la complejidad por medio de ocultar detalles completos en diferentes niveles de entendimiento.

Diseño de algoritmos

Crear una solución con pasos simples que cualquiera pueda seguir.

El **diseño de algoritmos** cosiste en escribir los pasos que se deben seguir para poder obtener la misma solución en todo momento.

Cuando una solución se lleva a cabo mediante un algoritmo, la solución se vuelve reutilizable.

Tipos de programas

Los **programas informáticos** pueden ser clasificados en diferentes categorías en función del objetivo para el que son desarrollados:

- **Software de sistema:** programas que permiten la interacción entre el usuario y el hardware. Constituyen la plataforma o entorno sobre el que los demás tipos de software pueden ejecutarse.

- **Software de aplicación:** programas de usuario final, utilizados en el entorno empresarial, educativo, recreativo, etc. Constituyen las aplicaciones software utilizadas hoy en día en ordenadores, teléfonos inteligentes, etc.
- **Software de programación:** programas para la asistencia en el desarrollo de software. Comúnmente conocidos como entornos de desarrollo integrado o por sus siglas en inglés IDEs.

Software de sistema

Algunos ejemplos de **software de sistema** pueden ser:

- Sistemas operativos
- Drivers de dispositivos
- Firmware
- Traductores de lenguajes de programación
- Utilidades

Software de aplicación

Algunos ejemplos de **software de aplicación** pueden ser:

- Procesadores de texto
- Hojas de cálculo
- Navegadores web
- Software multimedia

- Software de bases de datos
- Aplicación móvil
- CRM

Software de programación

Algunos ejemplos de **software de programación** pueden ser:

- Herramientas para programar **(IDEs):** Eclipse, herramientas de JetBrains, Visual Studio, etc.
- Herramientas de acceso a b**ases de datos:** DataGrip, DBeaver, pgAdmin, MySQL Workbench, etc.
- Herramientas de **testing:** Katalon Studio, Postman.

Tema 2. Características de los lenguajes de programación para IA

2.1. Introducción y objetivos

Para estudiar este tema lee las Ideas Clave que se presentan a continuación y revisa los contenidos adicionales para mejorar la comprensión y aumentar los conocimientos sobre las materias tratadas.

Este tema supone un recorrido por los fundamentos de los lenguajes de programación, los diferentes paradigmas y elementos que los caracterizan. Los objetivos más concretos son:

- Describir la **evolución de los lenguajes de programación** desde el lenguaje máquina a los lenguajes de alto nivel.
- Conocer los **componentes comunes** a todos los lenguajes de programación.
- Entender los distintos **paradigmas de programación** y sus diferencias.
- Saber **elegir** un **paradigma de programación** en función de las necesidades.
- Distinguir las características que hacen que un lenguaje de programación sea idóneo para el desarrollo de **inteligencia artificial.**

Tema 2. Características de los lenguajes de programación para IA

2.2. Evolución de los lenguajes de programación

Para escribir programas informáticos es necesario el uso de lenguajes de programación. Los lenguajes de programación son un conjunto predefinido de palabras que se combinan de acuerdo con una serie de reglas predefinidas (sintaxis).

Los primeros lenguajes de programación son conocidos como **lenguajes máquina,** compuestos por **código binario.** Este tipo de lenguajes es el único que entienden las máquinas y supone mucha dificultad para los programadores desarrollar tareas complejas.

Es por esta razón que los lenguajes de programación han ido evolucionando década tras década para acercarse más al lenguaje natural de las personas, con el objetivo de facilitar la programación de tareas complejas a los programadores.

De esta forma, los lenguajes de programación han evolucionado de la siguiente manera:

- Lenguajes máquina
- Lenguajes ensambladores
- Lenguajes de alto nivel

Lenguaje máquina

La primera generación de lenguajes de programación son los lenguajes máquina. El **lenguaje máquina** recibe su nombre por ser el único que entiende el hardware de las máquinas.

El hardware de las máquinas se compone de circuitos formados por interruptores eléctricos con dos estados: **encendido** (equivale a 1) y **apagado** (equivale a 0).

Debido a que solo hay dos estados posibles, apagado o encendido (0 o 1), estos lenguajes reciben el nombre de **código binario.**

El principal problema del lenguaje máquina es su **dependencia con el hardware,** lo que quiere decir que el lenguaje máquina utilizado para un hardware concreto tendrá que ser distinto para una máquina que utilice un hardware diferente.

Este inconveniente sumado a la **dificultad de programar y encontrar errores** en este tipo de lenguajes hace que no sean idóneos para el día a día de la programación de aplicaciones informáticas.

Lenguaje ensamblador

Para mitigar los inconvenientes del lenguaje máquina, surge una **segunda generación** de lenguajes de programación conocidos como lenguajes ensambladores.

La idea principal de los **lenguajes ensambladores** es la de reemplazar el código binario en la fase de programación con el fin de facilitar su desarrollo.

El código ensamblador está compuesto por símbolos, mnemónicos que simbolizan instrucciones, secciones de datos y directivas del ensamblador.

Como el hardware únicamente entiende el código máquina es necesario el uso de una herramienta **(ensamblador)** que traduzca el código ensamblador al código máquina.

El principal inconveniente es que los lenguajes ensambladores siguen siendo específicos de hardware sobre el que se ejecutan, es decir, cada arquitectura de procesador tiene su propio lenguaje ensamblador específico.