Conceptos de programación

Paradigmas de programación:

Un paradigma de programación es un conjunto de principios, enfoques y reglas que guían la forma en que se abordan y resuelven problemas de programación. Es similar a un estilo o marco conceptual que define cómo se debe organizar y escribir el código para lograr un resultado deseado. Los paradigmas de programación son como enfoques reconocidos y establecidos que influyen en cómo se estructuran los datos, cómo se controla el flujo de ejecución y cómo se manipulan las instrucciones. Aunque cada programador puede tener su estilo personal de codificación, los paradigmas proporcionan un enfoque más amplio y generalizado para crear software eficiente y coherente. Algunos lenguajes de programación están diseñados en torno a uno o más paradigmas, y mientras que algunos lenguajes son más flexibles y permiten la combinación de enfoques, los paradigmas siguen siendo un componente fundamental en el diseño y la utilización de lenguajes de programación.

- *Relación entre matemáticas y los paradigmas de programación:*
  las matemáticas y los paradigmas están estrechamente ligados. tanto en las matemáticas como en los paradigmas, encontramos que cada uno tiene una lógica, reglas, estructuración y resolución de problemas similar, podríamos decir que la programación fue creada con las matemáticas en mente, usando los conceptos de las matemáticas a los diferentes paradigmas
- *Números binarios y sistemas de codificación:*
  - Los números binarios son como una serie de "unos" y "ceros". Ayudan a las computadoras a entender las cosas de manera rápida y eficiente.
  - En los sistemas de codificación, cada letra, número o símbolo tiene su propio "número especial" de "unos" y "ceros". Esto ayuda a las computadoras a entender diferentes cosas.
  - Aunque los sistemas de codificación pueden ser diferentes, todos ayudan a las computadoras a comprender y trabajar con información.
  - Los sistemas de codificación hacen que las computadoras puedan leer archivos y cosas escritas de manera más fácil y organizada.
- *Programación en lenguaje máquina, programación de bajo y alto nivel:*
  - Programación en lenguaje de máquina: Usa números binarios que las computadoras entienden directamente, pero es muy técnica y compleja para los humanos.
  - Programación de bajo nivel: Es más comprensible para los humanos, pero aún cerca de los detalles técnicos. Puede ser propensa a errores, pero se ajusta mejor a las máquinas.
  - Programación de alto nivel: Más fácil para los humanos, se parece más al lenguaje que hablamos. Es eficiente y productiva, ya que se concentra en las soluciones y no en los detalles técnicos.

- *Programación de computadores:*
  la programación para computadores es un conjunto de órdenes y reglas que establecemos nosotros las personas para que la maquina consiga el resultado esperado
- *Cuadro de conceptos:*

Paradigma	Descripción Breve	Principal Enfoque	Ejemplo
Programación Imperativa	Instrucciones paso a paso para cambiar cosas	Decirle a la computadora qué hacer	C, Python
Programación Orientada a Objetos	Organizar datos y acciones en objetos	Agrupar datos y acciones en objetos	Java, C++
Programación Declarativa	Decir qué quieres hacer en lugar de cómo hacerlo	Describir resultados en lugar de pasos	SQL, HTML
Programación Funcional	Usar funciones para construir soluciones	Construir con funciones y composición	Haskell, JavaScript
Programación Lógica	Definir relaciones y encontrar respuestas	Plantear preguntas y resolverlas	Prolog, Mercury
Programación por Restricciones	Establecer condiciones para resolver problemas	Plantear restricciones y resolver	CHR, ECLiPSe
Programación Concurrente	Hacer muchas cosas al mismo tiempo	Hacer tareas simultáneamente	Java (hilos), Erlang
Programación para la Web	Crear aplicaciones y sitios web	Construir para navegadores y servidores	JavaScript, PHP

- *Estas son algunas de las actuales tendencias:*
  - Inteligencia Artificial (IA) y Aprendizaje Automático (Machine Learning): La IA y el aprendizaje automático siguen siendo tendencias dominantes. Los programadores están desarrollando algoritmos y sistemas para realizar tareas como reconocimiento de voz, procesamiento de imágenes, análisis de datos y toma de decisiones.
  - Desarrollo Web y Aplicaciones en la Nube: La creación de aplicaciones web y la migración hacia servicios en la nube continúan en auge. Los lenguajes como JavaScript y frameworks como React y Angular son populares para el desarrollo front-end, mientras que tecnologías como Node.js permiten el desarrollo del lado del servidor.
  - Computación Cuántica: Aunque aún está en sus primeras etapas, la computación cuántica está ganando interés. Se espera que esta tecnología transforme la forma en que se resuelven problemas complejos y realicen cálculos a gran escala.
  - Programación en Dispositivos Móviles: Con la proliferación de dispositivos móviles, la programación de aplicaciones para iOS y Android sigue siendo una

tendencia importante. Lenguajes como Swift (iOS) y Kotlin (Android) están en aumento.

- Automatización y DevOps: La automatización de procesos y la integración continua/entrega continua (CI/CD) son fundamentales para acelerar el desarrollo de software y mejorar la calidad.
- Ciberseguridad: Con la creciente dependencia de la tecnología, la seguridad cibernética sigue siendo una preocupación crítica. Los expertos en seguridad informática son cada vez más importantes para proteger sistemas y datos.
- Realidad Virtual (VR) y Realidad Aumentada (AR): La programación para experiencias inmersivas y aplicaciones en el ámbito de la VR y AR está creciendo en campos como el entretenimiento, la educación y la industria.
- Internet de las Cosas (IoT): La programación para dispositivos IoT, que abarcan desde electrodomésticos inteligentes hasta sensores industriales, está ganando importancia a medida que más dispositivos se conectan a Internet.
- Desarrollo Sostenible y Ético: Los programadores están prestando más atención a la sostenibilidad y la ética en el desarrollo de tecnologías para garantizar que los productos sean responsables y respeten valores importantes.
- Lenguajes y Herramientas Nuevas: Nuevos lenguajes y herramientas emergen constantemente, abordando desafíos específicos y mejorando la productividad del programador.

- *Programación funcional:*

En la programación funcional, defines un conjunto de instrucciones en forma de funciones. Estas funciones son como pequeñas recetas que dices a la computadora. Luego, cuando le das ciertos datos o números, la computadora sigue esas recetas y realiza las operaciones que les habías dicho que hiciera. Las funciones siempre hacen lo mismo para los mismos datos y no cambian nada en el proceso. Es un enfoque predecible y organizado que se basa en usar funciones como herramientas para resolver problemas.

- Características:
  1. Funciones Puras: Funciones predecibles, sin efectos secundarios.
  2. Inmutabilidad: Datos que no cambian, creando nuevos en su lugar.
  3. Recursión: Repetir tareas mediante llamadas a funciones.
  4. Expresiones y Evaluación: Usar ecuaciones evaluables en lugar de comandos.
  5. Transparencia Referencial: Reemplazar una función por su resultado sin cambios.
  6. Composición de Funciones: Crear soluciones combinando funciones pequeñas.
  7. Lazy Evaluation: Evaluar operaciones solo cuando sea necesario.
  8. Enfoque Declarativo: Decir qué hacer, no cómo hacerlo.

- *LIPS:*

LISP es un antiguo y versátil lenguaje de programación. Destaca por su notación de paréntesis y su capacidad para manipular símbolos y listas. Es dinámico, permite la recursión y es utilizado en inteligencia artificial y programación simbólica. Su diseño flexible lo hace útil para experimentación y extensibilidad.

- *Racket y Scheme:*

  Dr. Scheme fue un entorno de desarrollo y sistema educativo que permitía aprender programación mediante Scheme, un lenguaje de programación de la familia Lisp. Scheme es conocido por su simplicidad y enfoque en la programación funcional. Presenta una sintaxis basada en paréntesis y se ha utilizado para enseñar conceptos de programación en entornos educativos.
  Con el tiempo, Dr. Scheme evolucionó en Dr. Racket, un entorno más avanzado que permite aprender múltiples lenguajes, incluido Scheme. Dr. Racket ofrece una amplia gama de herramientas para programar y experimentar con diferentes paradigmas. Racket, que antes se llamaba "PLT Scheme", es un lenguaje relacionado con Scheme. Sin embargo, Racket ha crecido en un lenguaje más completo con características propias y un enfoque en la experimentación con lenguajes y programación educativa.
  En resumen, Dr. Scheme y Dr. Racket son herramientas educativas basadas en Scheme, un lenguaje de programación funcional. Racket, un lenguaje relacionado con Scheme, ha evolucionado para convertirse en un lenguaje versátil y un entorno para la experimentación con lenguajes y la enseñanza de programación. Estas herramientas y lenguajes han sido utilizados en entornos educativos y en la comunidad de investigación en ciencias de la computación.
- *Software:*

  El software es la parte no física de los sistemas informáticos. Está compuesto por programas, datos y procesos que permiten que las computadoras realicen tareas específicas. Estas categorías organizan el software en diferentes roles y propósitos, desde controlar dispositivos industriales hasta crear aplicaciones para usuarios finales, como yo. Así como soy una instancia de software interactuando contigo en este momento, el software está en el corazón de muchas de las cosas que hacemos en el mundo digital.
- *Hardware:*

  El hardware se refiere a las partes físicas de las computadoras y tecnologías. Está dividido en diferentes categorías que tienen funciones específicas:

  - Entrada: Dispositivos para ingresar información.
  - Salida: Cosas que muestran información.
  - Procesamiento: El "cerebro" que hace cálculos y ejecuta programas.
  - Almacenamiento: Lugares para guardar datos a largo plazo.
  - Comunicación: Cosas para conectarse a otros dispositivos.
  - Conexión: Puertos y cables para unir cosas.
  - Auxiliar: Partes extra que ayudan.
  - Embebido: Componentes dentro de otros productos.

  El procesador es una parte del hardware que realiza acciones siguiendo instrucciones del software. Juntos, el hardware y el software hacen que las computadoras y tecnologías funcionen para realizar tareas.
- *Tipos de almacenamiento:*

  En resumen, en una computadora, el almacenamiento de información se realiza mediante la memoria RAM como almacenamiento principal, que es rápida pero

volátil, y el almacenamiento secundario, que incluye dispositivos como discos duros y SSD, que son más lentos, pero no volátiles y retienen datos incluso después de apagar la computadora. Ambos tipos de almacenamiento son fundamentales para el funcionamiento y el almacenamiento de datos a largo plazo en una computadora.