

AF4, Síntesis.



UANL
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN



FIME
FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

Lenguajes de Programación.

Ing. Karla Patricia Uribe Sierra.

Grupo 005.

M5.

Semestre enero – junio 2025.

Matthew Alejandro Martínez Zambrana.

2223170.

IAS.

San Nicolás de los Garza, N.L, miércoles 19 de febrero de 2025.

Reporte exposiciones equipo #2.

Q) Sintaxis y semántica. (Definición, características, propósitos)

La sintaxis y la semántica son aspectos fundamentales en los lenguajes de programación, ya que determinan tanto la estructura como el significado de las instrucciones escritas. La sintaxis establece las reglas y estructuras gramaticales para la correcta escritura de un programa, asegurando que el código sea comprensible para el compilador o intérprete. Estas reglas incluyen el uso correcto de operadores, delimitadores, palabras clave y estructuras de control.

Por otro lado, la semántica se encarga de definir el significado de las expresiones y sentencias utilizadas dentro de un programa. Alienta que un código cumpla con la sintaxis del lenguaje, si su semántica es incorrecta, el programa no producirá los resultados esperados. Un error semántico ocurre cuando una instrucción es escrita correctamente, pero su significado dentro del contexto del programa es erróneo o no válido.

Ambos conceptos son esenciales para garantizar que los programas sean comprensibles y ejecutables sin errores. Una buena práctica en el desarrollo de software es combinar un código sintácticamente correcto con una lógica bien definida para emitir errores en la ejecución y facilitar la depuración del programa.

Lenguaje de representación (Definición, características y ejemplos).

Existen distintos tipos de lenguajes que cumplen funciones específicas en el ámbito de la informática. Algunos de ellos están diseñados para la representación de información en formatos estructurados, facilitando el almacenamiento, intercambio y visualización de datos en diversos sistemas.

Estos lenguajes utilizan símbolos y reglas específicas que permiten la portabilidad y la compatibilidad entre diferentes plataformas y aplicaciones.

Los lenguajes de representación permiten organizar datos en formatos que pueden ser leídos tanto por humanos como por máquinas. Entre los más utilizados se encuentran XML y JSON. XML (Extensible Markup Language) se emplea para estructurar datos de manera jerárquica mediante etiquetas, lo que facilita su interpretación y modificación. JSON (JavaScript Object Notation), por su parte, ofrece una alternativa más ligera y fácil de manejar, siendo ampliamente usado en el intercambio de datos entre servidores y aplicaciones web.

El uso de estos lenguajes es crucial en entornos donde la interoperabilidad entre sistemas es un requisito, como en bases de datos, servicios web y aplicaciones móviles. Su capacidad de representar información estructurada los hace indispensables en la comunicación de datos entre distintos dispositivos y plataformas.

Lenguaje de consulta. (Definición, características y ejemplo).

Los lenguajes de consulta están orientados a la manipulación y recuperación de datos almacenados en bases de datos. Su propósito principal es facilitar el acceso y la gestión de la información, permitiendo a los usuarios realizar operaciones como selección, filtrado, actualización y eliminación de registros de manera eficiente.

SQL (Structured Query Language) es el lenguaje de consulta más utilizado en bases de datos relacionales. Este lenguaje permite realizar consultas complejas mediante comandos como SELECT, INSERT, UPDATE y DELETE. Su sintaxis estructurada y su capacidad para manejar grandes volúmenes de información lo convierten en una herramienta esencial en el manejo de datos.

Otras lenguajes de consulta incluyen XPath y XQuery, utilizados en bases de datos que almacenan información en formato XML. Estos lenguajes permiten buscar y extraer datos de documentos XML mediante expresiones estructuradas que facilitan la navegación dentro de la jerarquía de elementos.

El conocimiento y uso adecuado de estos lenguajes es fundamental en el ámbito de la informática, ya que permiten optimizar el acceso a la información y garantizar la integridad de los datos almacenados en sistemas de bases de datos.

Lenguaje de alto nivel.

Dentro de la clasificación de los lenguajes de programación, se encuentran aquellos considerados de alto nivel. Estos lenguajes están diseñados para facilitar la escritura de código al proporcionar una sintaxis cercana al lenguaje humano, lo que permite un desarrollo más intuitivo y menos propenso a errores.

Los lenguajes de alto nivel ofrecen abstracción del hardware, permitiendo a los programadores centrarse en la lógica del programa sin preocuparse por los detalles específicos de la arquitectura del sistema. Esto los hace ideales para el desarrollo de aplicaciones en diversos entornos, desde software de escritorio hasta inteligencia artificial.

Ejemplos de lenguajes de alto nivel incluyen Python, Java y C++. Python es ampliamente utilizado debido a su simplicidad y legibilidad, lo que lo convierte en una opción ideal para principiantes y proyectos de desarrollo rápido. Java, por su parte, es popular en aplicaciones empresariales y móviles gracias a su portabilidad y robustez. C++ combina características de alto y bajo nivel, permitiendo un control detallado sobre la memoria y el rendimiento del programa.

Estos lenguajes han sido fundamentales en el avance de la informática, facilitando el desarrollo de software eficiente y accesible para una amplia gama de aplicaciones.

Componentes de un lenguaje de programación.

Los lenguajes de programación están compuestos por diversos elementos interrelacionados que permiten la construcción de programas funcionales. Entre los principales componentes se encuentran los lexemas, que representan las unidades mínimas de significado en el código fuente. Estos incluyen palabras clave, identificadores, operadores y símbolos especiales. Así como también estructuras de control y las bibliotecas estándar.

Las estructuras de control, como los condicionales y los bucles, permiten modificar el flujo de ejecución del programa, haciendo posible la toma de decisiones y la repetición de instrucciones de manera controlada. Los operadores, por su parte, permiten realizar cálculos y comparaciones entre valores, facilitando la manipulación de datos. Las funciones permiten la reutilización del código al encapsular instrucciones en bloques independientes, lo que mejora la modularidad y mantenibilidad del programa.

Por otro lado, las bibliotecas estándar contienen funciones y clases predefinidas que facilitan la programación al proporcionar herramientas listas para su uso, sin necesidad de implementar soluciones desde cero. La relación entre estos componentes es crucial para garantizar que un programa sea funcional, eficiente y fácil de mantener.

Un diseño bien estructurado que aproveche adecuadamente estos elementos puede mejorar significativamente la legibilidad y rendimiento del código.

Definición y características de un compilador e intérprete.

El compilador y el intérprete son herramientas esenciales en la ejecución de programas escritos en distintos lenguajes de programación. Ambos tienen la función de traducir el código fuente, escrito en un lenguaje de alto nivel, a un formato que puede ser ejecutado por la computadora, aunque difieren en la forma en que realizan esta tarea.

Un compilador es un programa que traduce el código fuente completo a código máquina antes de su ejecución. Esto significa que el programa es procesado en su totalidad, generando un archivo ejecutable que puede ser ejecutado de manera independiente del código fuente. Algunas de sus características incluyen una mayor eficiencia en la ejecución, ya que el código ya está traducido, y la capacidad de detectar errores en la fase de compilación, lo que evita que el programa se ejecute con errores sintácticos.

En contraste, un intérprete traduce y ejecuta el código línea por línea, sin generar un archivo ejecutable. Esto permite una ejecución más flexible, ya que el código puede modificarse y ejecutarse de inmediato sin necesidad de recomilar. Sin embargo, esta característica puede hacer que la ejecución sea más lenta en comparación con un programa compilado, ya que la traducción ocurre en tiempo real.

Los compiladores se utilizan comúnmente en lenguajes como C y C++, mientras que los intérpretes son empleados en lenguajes como Python y JavaScript. La elección entre un compilador o un intérprete depende del propósito del programa y de los requerimientos de ejecución, ya que ambos tienen ventajas y desventajas según el contexto en el que se utilicen.

Estructura, operación y ejemplos de un Intérprete.

Un intérprete es un software que se encarga de traducir y ejecutar código fuente directamente, sin generar un archivo ejecutable intermedio. Su estructura consta principalmente de un analizador léxico, un analizador sintáctico, un evaluador semántico y un ejecutor.

El analizador léxico toma el código fuente y lo divide en tokens, que son las unidades mínimas de significado dentro del lenguaje de programación. Luego, el analizador sintáctico verifica que los tokens estén organizados correctamente de acuerdo con la gramática del lenguaje. Posteriormente, el evaluador semántico revisa que las expresiones tengan un significado válido, asegurando que el código sea coherente. Finalmente, el ejecutor interpreta las instrucciones y las ejecuta en el entorno de ejecución.

El proceso de operación de un intérprete ocurre en tiempo real, lo que permite la ejecución inmediata del código sin necesidad de compilación previa. Esta característica es útil en el desarrollo de software interactivo y en la depuración de programas, ya que los errores pueden corregirse de inmediato sin necesidad de recompilar todo el código. Sin embargo, debido a que cada instrucción es traducida y ejecutada en el momento, los programas interpretados suelen ser más lentos que los compilados.

Algunos ejemplos de intérpretes incluyen el intérprete de Python, que permite ejecutar scripts y probar código de manera interactiva en la consola; el motor V8 de JavaScript, utilizado en navegadores web como Google Chrome para ejecutar código JavaScript en páginas web; y Ruby, cuyo intérprete permite

desarrollar aplicaciones web con rapidez y flexibilidad.

El uso de intérpretes es común en entornos donde la rapidez de desarrollo y la flexibilidad son más importantes que la velocidad de ejecución, como en el desarrollo de scripts, la enseñanza de programación y el análisis de datos. La capacidad de ejecutar código de manera interactiva y modificarlo en tiempo real hace que los intérpretes sean herramientas valiosas en muchas áreas de la Informática.