TEMA 24

LENGUAJES DE PROGRAMACIÓN. TIPOS Y CARACTERÍSTICAS

INDICE:

Realizado por Cayetano Borja Carrillo

Tiempo de escritura: 1 hora y 45 minutos

1. INTRODUCCIÓN

Hoy en día vivimos en una sociedad donde el uso de los equipos informáticos (ordenadores, teléfonos inteligentes, etc.) es algo habitual tanto en el trabajo como en la vida privada. Todos estos equipos requieren que, de alguna forma, se les indique qué deben hacer y cómo deben de hacerlo. Aquí es donde los lenguajes de programación cobran importancia, ya que son estos los que permiten crear los programas (*software*) que permiten a la máquina funcionar.

En este tema se desarrollan los aspectos fundamentales de los lenguajes de programación. Se trata de un tema de gran importancia en la informática ya que los programas son imprescindibles para que un sistema informático funcione.

2. LENGUAJES DE PROGRAMACIÓN

Un lenguaje de programación es un lenguaje formal conformado por un conjunto predefinido de palabras y símbolos (léxico) que se utilizan siguiendo unas reglas prefijadas (sintaxis), para representar y definir los datos y algoritmos (semántica) que constituyen un programa.

A través de los lenguajes de programación se define todo lo que un programa debe de hacer, desde realizar una suma de dos números enteros hasta mover un personaje dentro de un videojuego.

2.1. Léxico

El léxico es el vocabulario o conjunto de símbolos válidos que se pueden usar en un lenguaje de programación. Estos símbolos o elementos son los siguientes:

- Identificadores: Nombres simbólicos que se da a ciertos elementos de programación, por ejemplo, nombres de variables, tipos, módulos, etc. Ejemplos en lenguaje C: main, std, cout, numero1, resultado, etc.

- Constantes: Datos que no cambiarán su valor durante la ejecución del programa. Ejemplo: Un valor numérico, una cadena de texto, etc.

- Operadores: Símbolos que representan operaciones entre variables y constantes. Ejemplos: +, -, /, >, >=, !=, &&, etc.

- Palabras reservadas o clave: Símbolos especiales que representan instrucciones de procesamiento y definiciones de elementos de programación. Cada lenguaje tiene sus propias palabras clave. Ejemplos en C: bool, case, break, do, default, continue, class, etc.

- Comentarios: Se usan para crear anotaciones dentro del código y documentar algo sobre el programa. Son ignorados por los compiladores e intérpretes.

- Delimitadores o separadores: Símbolos utilizados para señalar el principio o el final de una unidad sintáctica, como un enunciado o una expresión. Ejemplos: , ; . { } ( ) [ ], etc.

A continuación, se muestra un ejemplo de una sentencia en lenguaje C:

res = x + 3; #Asigna el resultado de la suma de “x + 3” a la variable “res”.

Y su léxico es el siguiente:

- “res” y “x” → Identificadores de variables (espacio reservado en memoria)
- “=” y “+” → Operadores
- “3” → Constante
- “;” → Delimitador que indica fin de sentencia
- “#Almacena la suma de la variable…” → Comentario

2.2. Sintaxis

Son el conjunto de reglas que indican la forma de realizar las construcciones del lenguaje y determinan si una frase está bien escrita o no.

Existen diversas formas de especificar la sintaxis, pero sólo vamos a ver la notación BNF (*Backus-Naur Form*) ya que fue de las primeras notaciones que se empezó a utilizar para especificar lenguajes de programación. Los metasímbolos de BNF son:

Metasímbolo	Tipo	Significado
::=	De definición	El esquema de la derecha define el elemento de la izquierda. Equivale a decir “se define como”
|	De alternativa	Se puede elegir sólo uno de los elementos.
{ }	De repetición	Los elementos se pueden repetir 0 o más veces
[ ]	De opción	Los elementos pueden utilizarse o no
( )	De agrupación	Para agrupar los elementos

Ejemplo: Tenemos la siguiente descripción sintáctica en BNF.

<expresión> ::= <número> | (<expresión>) | <expresión><operador><expresión>
<operador> ::= + | - | * | /
<número> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0

Ahora comprobamos si la expresión matemática “4*(3+1)” está bien escrita:

La expresión “4*(3+1)” se define como <expresión><operador><expresión>. La primera expresión se define como <número>, que a su vez se define como “4”. La parte <operador> se define como “*”. La tercera parte “(3+1)” se define como (<expresión>), que a su vez se define como <expresión><operador><expresión>. De ahí, las 2 expresiones se definen como <número>, que a su vez se definen como “3” y “1” respectivamente y la segunda parte se define como “+”.

Como se ha comprobado, la expresión matemática está bien escrita porque no incumple ninguna norma.

Hay que tener en cuenta que cada lenguaje de programación tiene su propia sintaxis, por lo que el código que resuelve un mismo algoritmo puede diferir entre lenguajes.

2.3. Semántica

Define el significado de las construcciones sintácticas del lenguaje y de las expresiones y tipos de datos utilizadas.

Por ejemplo, en la construcción algorítmica <if a == 1 then S1 else S2>, el significado corresponde a <si el valor de “a” es “1” haz “S1”, en caso contrario haz “S2””.

2.4. Criterios para la elección de un lenguaje de programación

A la hora de definir o elegir un lenguaje de programación hay varios criterios que determinarán las características de este lenguaje:

- Facilidad de lectura y comprensión: Los programas son modificados y corregidos con frecuencia, por lo tanto, elegir un lenguaje que facilita esta tarea es fundamental.
- Facilidad de codificación**:** Esto reducirá el tiempo de desarrollo.
- Fiabilidad: Podemos considerar fiable a un lenguaje si funciona bien en cualquier condición. Esto dependerá de la comprobación de tipos de datos, del control de excepciones y del manejo de la memoria.
- Entorno de programación**:** Un entorno de programación adecuado puede facilitar mucho el uso del lenguaje de programación, permitiendo un desarrollo más rápido.
- Portabilidad de programas: Un mismo programa, dependiendo del lenguaje usado, puede compilarse para funcionar en distintos sistemas.
- Coste: Relacionado con la facilidad de aprendizaje, codificación, las necesidades de ejecución, así como la fiabilidad y el mantenimiento.

3. TIPOS Y CARACTERÍSTICAS

Existen multitud de formas de clasificar los lenguajes de programación, la siguiente clasificación se ha hecho según una serie de criterios como pueden ser:

3.1. Según el nivel de abstracción

Dependiendo del nivel de abstracción existen los siguientes tipos:

- Lenguaje máquina: Consiste en un conjunto de instrucciones codificadas en binario y directamente entendible por la máquina, por lo que existe una total dependencia con la arquitectura física de la misma.

  Ejemplo de instrucción: 01101000.

- Lenguajes ensambladores: Permiten escribir las instrucciones con una notación simbólica que se traducirá de forma muy directa a lenguaje máquina. Esta notación facilita un poco la tarea al programador, pero sigue existiendo una fuerte dependencia con la arquitectura física de la máquina.

  Ejemplo de instrucción: ADD AX, 3 (Aumenta el valor del registro AX en 3).

- Lenguajes de nivel medio: Aportan una mayor independencia de la máquina y son más cercanos al lenguaje humano, permitiendo al programador centrarse más en el problema a resolver y menos en el conocimiento de la arquitectura de la máquina. Generalmente suelen estar orientados a procedimientos. Algunos ejemplos son C, Basic o Cobol.

  Ejemplo de instrucción en C: x=3+5; (Suma 3 y 5 y ponlo en la variable “x”).

- Lenguajes de alto nivel: Permiten una expresión casi oral entre la escritura del programa y su posterior compilación. Generalmente suelen estar orientados a objetos, eventos o funciones, asimismo, pueden ser compilados o interpretados. Algunos ejemplos son Java, C#, PHP, Python o C++.

  Ejemplo de instrucción en PHP: echo “Tengo $edad años.”;

3.2. Según su paradigma de programación

Se puede definir paradigma de programación como el enfoque que se utiliza para obtener una solución a un problema, el cual afecta al proceso completo del desarrollo del software. A continuación, se describen algunos paradigmas de programación.

- Imperativo o por procedimientos: Este tipo de programación es el más antiguo y el más usado. Consiste en dar instrucciones al ordenador de cómo hacer las cosas en lugar de describir el problema o la solución. Las recetas de cocina tienen un concepto similar a la programación imperativa ya que se dan los pasos para llegar a una solución. Algunos ejemplos son C, Basic y Pascal.

- Declarativa: Consiste en describir el problema declarando las propiedades y reglas que deben de cumplirse en lugar de instrucciones. Algunos ejemplos son Lisp y Prolog.

- Orientada a objetos: Basada en el imperativo, este tipo de programación se caracteriza por la forma de manejar la información. En ella, todos los elementos se modelan como objetos con estado que pueden interaccionar entre sí. En este paradigma se manejan tres conceptos claves:

- Clases: Definen las propiedades y comportamiento de un tipo de objeto.
- Objetos: Instancias concretas de una clase. Representan objetos reales del mundo que nos rodea u objetos internos del sistema. Posee un estado (datos) y su comportamiento está definido en su clase.
- Herencia: Mecanismo que permite definir subclases (o clases hijas) que heredan las características de sus padres y se le pueden añadir las que les son propias.

Algunos ejemplos son C++, C#, Java y Python.

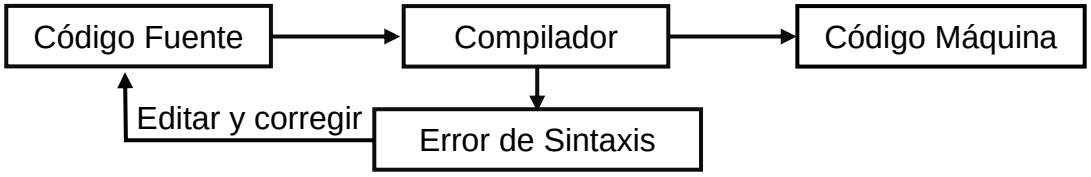
- Dirigida por eventos: Permite interactuar con el usuario y reaccionar a cualquier evento mientras se ejecuta el programa. En cierta manera, se puede decir que en la programación clásica el programador es quien decide el orden de ejecución de los procesos mientras que en ese paradigma es el usuario el que decide y controla el flujo. Un ejemplo es Swif.

- Multiparadigma: Recogemos aquí los lenguajes que no están restringidos a un único paradigma de programación, sino que permiten emplear o incluso mezclar varios paradigmas. Ejemplo: C++, Delphi, PHP, D y Visual Basic combinan el paradigma imperativo y el orientado a objetos.

3.3. Según su traducción al código máquina

Dependiendo de si el código se ha de transformar a código máquina o no, se pueden clasificar en:

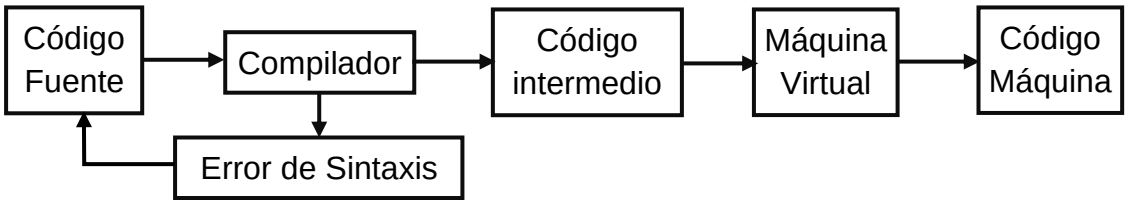
- Compilados: Son aquellos en los que el programa debe ser convertido a código máquina (ceros y unos) para ser ejecutado. Este proceso se conoce como compilado y es realizado por un programa compilador que, además, realiza tareas adicionales como, por ejemplo, comprobar la sintaxis del programa para verificar que sea correcta antes de traducir a código máquina. El programa resultante depende tanto de la arquitectura como del sistema operativo de destino, por lo que para poder ejecutar el programa en otro sistema operativo debería de ser recompilado. Algunos lenguajes dentro de esta categoría son Pascal, Fortran, Basic, C y C++.



- Interpretados: Son aquellos que necesitan un intérprete que transforme cada instrucción a código máquina conforme se va ejecutando. Una de las grandes ventajas de estos lenguajes es que el programa se puede ejecutar en cualquier dispositivo que tenga un intérprete del lenguaje, sin importar la

arquitectura y sistema operativo. Algunos lenguajes dentro de esta categoría son Python, JavaScript, PHP y un ejemplo de intérprete es un navegador web.

- Híbridos: En este caso, el código fuente también es compilado, pero no a código máquina, sino a un código intermedio. Una máquina virtual se encarga de transformar el código intermedio al código nativo de la máquina donde se ejecuta. Al igual que los interpretados, el programa se puede ejecutar en cualquier dispositivo que tenga instalado la máquina virtual necesaria. Algunos ejemplos de lenguajes son Java, C# y Kotlin.



3.4. Según su etapa de generación

Otra forma de clasificar los lenguajes es separándolos en generaciones, las cuales han evolucionado de la siguiente forma:

- 1ª generación: Lenguajes máquina.
- 2ª generación: Se inicia con la aparición del lenguaje ensamblador. Hoy en día, este lenguaje sólo se usa para crear pequeños programas como controladores (*drivers)* o *firmwares* de algunos sistemas embebidos.
- 3ª generación: Aparecen los lenguajes de nivel medio y alto.
- 4ª generación: Son lenguajes declarativos cuyas declaraciones son similares al lenguaje humano. Se usan comúnmente en la programación de bases de datos y *scripts*. Algunos ejemplos son SQL, Perl, Python y Ruby.
- 5º generación: Esta generación se ha asociado a los lenguajes que se utilizan en el campo de la inteligencia artificial y las redes neuronales artificiales. Algunos ejemplos son LISP, Mercury y PROLOG.

3.5. Según su ámbito de aplicación

En este caso podemos diferenciar los lenguajes de propósito general que nos permiten crear aplicaciones de cualquier tipo de aquellos lenguajes que tienen un propósito más específico tal como puede ser el cálculo científico (Fortran, Pascal), procesamiento de datos (Cobol, SQL), aplicaciones para inteligencia artificial (Lisp, PROLOG), etc.

3.6. Lenguajes de programación visuales

Los lenguajes visuales utilizan elementos gráficos para representar las construcciones del lenguaje en vez de los símbolos léxicos como ocurre con los lenguajes basados en texto. Esto elimina los errores de sintaxis y hace más atractiva la programación a quienes se inician en ella, hecho que se ve reflejado en el creciente uso el ámbito educativo de lenguajes como *AppInventor* o *Scratch* como introducción a la programación.

4. CONCLUSIÓN

Los lenguajes de programación son una necesidad incuestionable para dar "vida" a computadores y dispositivos electrónicos. A lo largo de su historia han ido alejándose cada vez más del lenguaje máquina y acercándose más al lenguaje humano, consiguiendo niveles de abstracción aún mayor. Todavía hoy en día siguen apareciendo nuevos lenguajes evolucionando los ya consolidados.

5. BIBLIOGRAFÍA

- López Ureña, L. A. et al. (1997). *Fundamentos de Informática (1ª ed.)*. Ra-ma.
- Prieto Espinosa, A. et al. (2006). *Introducción a la informática (4ª ed.)*. McGraw-Hill.
- Brookshear, J. G. (2012). *Introducción a la computación (11ª ed.)*. Pearson Educación.
- Joyanes Aguilar, L. (2020). *Fundamentos de programación. Algoritmos, estructuras de datos y objetos (5ª ed.)*. McGraw-Hill.

6. NORMATIVA

Para el desarrollo de este tema, se ha tenido en cuenta la siguiente normativa, donde se especifican los contenidos, competencias y criterios de evaluación de los Ciclos Formativos y Bachillerato en Andalucía:

- Orden 16 de junio de 2011 (DAW/DAM). La parte correspondiente al módulo "Programación" y "Entornos de desarrollo".
- Instrucción 13/2022 (Bachillerato). La parte correspondiente a la asignatura "Tecnologías de la Información y Comunicación"