1. INTRODUCCIÓN

En este documento se expondrá el concepto de lenguaje de programación, así como los distintos tipos y sus principales características.

2. CONCEPTO DE LENGUAJE DE PROGRAMACIÓN

Podemos definir un Lenguaje de Programación como un idioma creado de forma artificial, formado por un conjunto de símbolos y normas que se aplican sobre un alfabeto para obtener un código, que el hardware de la computadora pueda entender y ejecutar. Un lenguaje de programación viene definido por:

- **Léxico**: conjunto de símbolos que se pueden utilizar en el lenguaje (identificadores, constantes, operadores…).
- **Sintaxis**: conjunto de reglas que gobiernan la construcción de instrucciones válidas con el léxico anterior.
- **Semántica**: conjunto de reglas que definen el significado de una instrucción.

Los elementos básicos de un lenguaje de programación son los siguientes:

- **Datos**: variables y constantes que llevan asociado un **tipo de dato,** que determina los valores que pueden tomar y operaciones que se puede realizar con ellos.

  Pueden ser simples (enteros, reales, caracteres…) o estructurados (arrays, listas, ficheros…).

- **Expresiones**: formadas por varios operandos (variables y constantes) unidos entre sí mediante operadores (aritméticos, relacionales, lógicos y paréntesis) que realizan una acción sobre ellos.
- **Instrucciones**: pueden ser simples (declarativas, primitivas, de control y comentarios) o compuestas (unión de varias simples).

3. CARACTERÍSTICAS DE UN LENGUAJE DE PROGRAMACIÓN

Estas son una serie de características que pueden tener los lenguajes de programación, de manera que el cumplimento o no de las mismas determine la validez de un lenguaje a la hora de dar solución a un determinado tipo de problema:

- **Portabilidad**: facilidad para que un programa pueda ser transferido de una máquina a otra.
- **Detallabilidad** número de pasos o instrucciones que es necesario definir en un programa para dar solución a un problema.
- **Usabilidad**: facilidad para ser aprendido o utilizado.
- **Generalidad**: capacidad de un lenguaje para dar solución a distintos tipos de problemas.

4. CLASIFICACIÓN DE LOS LENGUAJES DE PROGRAMACIÓN

4.1. Clasificación en función de la cercanía a la máquina

4.1.1. Lenguajes de bajo nivel

Ejercen un control directo sobre el hardware, son muy cercanos a la máquina y presentan un nivel de abstracción muy bajo.

A este grupo pertenecen el lenguaje máquina y el lenguaje ensamblador:

- **Lenguaje máquina:**

  Es el único lenguaje de programación que entiende directamente la máquina y utiliza un alfabeto binario de 0s y 1s. Fue el primer lenguaje de programación utilizado, aunque su complejidad y la aparición de otros lenguajes más cercanos al lenguaje natural, propiciaron su abandono.

- **Lenguaje ensamblador:**

  Surge como el primer intento de acercar el lenguaje de programación al lenguaje natural. Ya no utiliza códigos de operaciones ni direcciones de memoria binarias, sino que se sustituyen por códigos de operaciones y direcciones simbólicas.

  Existe una correspondencia entre este lenguaje y el lenguaje máquina: a cada instrucción en lenguaje ensamblador le corresponde una instrucción en lenguaje máquina.

Sus principales ventajas son:

- **Velocidad de ejecución**: ya que no requieren de una traducción previa a su ejecución o esta es muy simple en comparación con los lenguajes de alto nivel.
- **Máximo aprovechamiento de los recursos de la máquina***:* ya que acceden directamente a estos recursos.

Sus principales inconvenientes son:

- **Requieren conocer la arquitectura de la máquina**.
- **Portabilidad baja**: los programas no son portables de una máquina a otra.
- **Detallabilidad alta**: requiere un número grande de instrucciones.
- **Usabilidad baja**: es muy difícil de aprender y utilizar.

4.1.2. Lenguajes de alto nivel

Se consigue una programación más independiente de la máquina, su nivel de abstracción es alto y son más cercanos al lenguaje natural.

Sus principales ventajas son:

- **Portabilidad alta**: mayor independencia de la arquitectura de la máquina.
- **Detallabilidad baja**: repertorio grande de instrucciones mucho más potentes.
- **Usabilidad alta**: fáciles de aprender, verificar y mantener.

Sus principales inconvenientes son:

- **Requieren un proceso de traducción** a lenguaje máquina.
- **Mayor consumo de recursos**.

A este grupo pertenecen: Cobol, C, C++, Java, Lisp, Fortran…

4.2. Clasificación en función de la evolución del hardware

- **Primera generación** (años 40-50):
  Lenguajes dependientes de la máquina con instrucciones binarias de 0s y 1s.

  Ejemplos: lenguaje máquina.

- **Segunda generación** (finales años 50):
  Ya no utiliza códigos de operaciones ni direcciones de memoria binarias, sino que se sustituyen por códigos de operaciones y direcciones simbólicas.

  Ejemplos: lenguaje ensamblador.

- **Tercera generación** (años 60-70):
  Lenguajes con una mayor independencia de la máquina y que utilizan estructuras de datos abstractos y un repertorio de instrucciones más complejo, haciendo más sencilla la programción.

  Son principalmente lenguajes procedimentales.

  Ejemplos: primeros lenguajes de alto nivel como Cobol, Fortran, Basic.

A partir de los años 70:

- Permiten la definición de tipos de datos propios.
- Permiten asignación dinámica de memoria.
- Primeros lenguajes orientados a objetos.

Ejemplos: lenguajes de alto nivel como Pascal, C, C++, Java.

- **Cuarta generación** (años 70):
  Lenguajes de propósito específico, orientados principalmente a las aplicaciones de gestión (Crystal Reports) y al manejo de bases de datos (SQL, Natural).

  Son fundamentalmente lenguajes declarativos que permiten al programador definir cuáles son los resultados finales que se desea obtener sin necesidad de diseñar el algoritmo.

- **Quinta generación** (años 80):
  Lenguajes declarativos que se han usado en el campo de la inteligencia artificial y el procesamiento de lenguajes naturales.

  Ejemplos: LISP o PROLOG.

4.3. Clasificación en función del paradigma de programación.

Un paradigma de programación representa un enfoque particular o filosofía para la construcción de software. Esto no implica que un paradigma sea mejor que otro, sino que dependiendo de las circunstancias, resultará más apropiado uno u otro.

4.3.1. Lenguajes imperativos o procedimentales

Este tipo de paradigma implica que el programador, para resolver un problema, ha de diseñar un algoritmo y concretarlo en un programa que le indique a la máquina paso a paso, qué debe hacer. Estos lenguajes se fundamentan en el uso de **variables** para almacenar datos y de **instrucciones** que indican las operaciones a realizar sobre los datos.

En este grupo se encuentran la mayoría de los lenguajes conocidos: FORTRAN, COBOL, C, PASCAL, etc.

4.3.1.1. Lenguajes orientados a objetos

Se pueden considerar como otra evolución de los lenguajes imperativos ya que:

- Elevan el nivel de abstracción.
- Simplifican el diseño de los programas.

- Aumentan la reutilización del código.

Están basados en los conceptos de **clases y objetos.** Los **objetos** son entidades que combinan estado (datos) y comportamiento (operaciones o métodos), a diferencia de los lenguajes imperativos tradicionales, donde datos y métodos no están relacionados. Mientras que las **clases** son los tipos de pertenencia de los objetos.

A grandes rasgos, la programación con estos lenguajes se basa en definir los objetos adecuados para resolver un problema determinado y en la comunicación de estos objetos a través de mensajes.

Ejemplos: lenguajes de alto nivel como C#, C++, o JAVA.

4.3.2. Lenguajes declarativos

La idea de los lenguajes declarativos es que el programador sólo tenga que **declarar** o especificar el problema, siendo el propio traductor del lenguaje el que se encargue de construir el algoritmo.

Estos programas se construyen mediante la definición de funciones (**lenguajes funcionales**) o mediante la definición de expresiones lógicas (**lenguajes lógicos**).

Ejemplos: SQL, LISP, PROLOG.

4.3.2.1. Lenguajes funcionales

Definen **funciones** matemáticas y carecen de instrucciones. Un programa funcional es una función que se define por composición de funciones más simples.

Características propias de estos lenguajes son:

- No existencia de asignaciones de variables.
- No existencia de estructuras repetitivas (las repeticiones de instrucciones se llevan a cabo por medio de funciones recursivas).

Ejemplo: LISP.

4.3.2.2. Lenguajes lógicos

Mediante **expresiones lógicas** o **predicados**, se definen **objetos** (que cuentan con propiedades y relaciones) y un conjunto de **reglas** (que describen las relaciones entre los objetos). En base a este conocimiento, el intérprete tratará de encontrar la respuesta a las preguntas planteadas por el programador.

Ejemplo: PROLOG.

4.4. Clasificación en función de su utilización

Se pueden clasificar atendiendo al área de trabajo a la que están enfocados:

- **De propósito general**: pueden ser utilizados en diversas áreas (BASIC, C++…).
- **Educativos**: es recomendable que posean una sintaxis clara, que favorezca la programación estructurada o, más recientemente, la programación orientada a objetos (PASCAL, EIFFEL…).
- **De gestión**: orientados a la solución de problemas de tratamiento de datos para la gestión (COBOL, SQL…).
- **De inteligencia artificial**: utilizados para robótica, sistemas expertos, visión artificial, etc. (LISP, PROLOG).
- **Para Internet**: diseñados para desarrollar aplicaciones para Internet, por ejemplo:
    - HTML: lenguaje de marcas para la creación de páginas web.
    - PHP: utilizado para el desarrollo web de contenido dinámico.