**LENGUAJES DE PROGRAMACIÓN**

Lenguajes de programación hay en gran cantidad, algunos han evolucionado a lo largo del tiempo y siguen vigentes en el transcurso de muchos años, mientras que otros han sido operativos durante un período más o menos largo y actualmente no se usan. Dada esta gran variedad de lenguajes, no se pretende dar una visión de todos, sino una clasificación en diversos tipos y concretar algunos de ellos. En general un lenguaje es un método conveniente y sencillo de describir las estructuras de información y las secuencias de acciones necesarias para ejecutar una tarea concreta. Los lenguajes de programación utilizan juegos de caracteres "alfabeto” para comunicarse con las computadoras. Las primeras computadoras sólo utilizaban informaciones numéricas digitales mediante el código o alfabeto digital, y los primeros programas se escribieron en ese tipo de código, denominado código máquina basado en dos dígitos 0 y 1, por ser entendible directamente por la máquina (computadora). La enojosa tarea de programar en código máquina hizo que el alfabeto evolucionase y los lenguajes de programación comenzaran a utilizar códigos o juegos de caracteres similares al utilizado en los lenguajes humanos. A sí, hoy día la mayoría de las computadoras trabajan con diferentes tipos de juegos de caracteres de los que se destacan el código ASCII y el EBCDIC. De este modo, una computadora a través de los diferentes lenguajes de programación utiliza un juego o código de caracteres que serán fácilmente interpretados por la computadora y que pueden ser programados por el usuario.

**Lenguajes naturales**

Los lenguajes naturales son aquellos con los cuales hablamos y escribimos en nuestras actividades cotidianas. Entre sus ventajas podemos citar que es fácil de usar y de entender: no debemos aprendernos ningún lenguaje nuevo y cualquiera puede leer la especificación y comentar la o criticarla. Entre los inconvenientes están la imprecisión y la ambigüedad. Aunque el análisis de requisitos se haya realizado correctamente, una especificación en lenguaje natural puede dar lugar a que la implementación final no cumpla estos requisitos.

Podemos resumir que los lenguajes naturales se distinguen por las siguientes propiedades:  Desarrollados por enriquecimiento progresivo antes de cualquier intento de formación de una teoría.

 La importancia de su carácter expresivo debido grandemente a la riqueza del componente semántico (poli semántica).

 Dificultad o imposibilidad de una formalización completa.

**Lenguajes de programación**

Un lenguaje de programación consiste en un conjunto de órdenes o comandos que describen el proceso deseado. Cada lenguaje tiene sus instrucciones y enunciados verbales propios, que se combinan para formar los programas de cómputo. Los lenguajes de programación no son aplicaciones, sino herramientas que permiten construir y adecuar aplicaciones. Existen muchos lenguajes de programación con características y aptitudes muy diferenciadas. Todo ello se encuentra en dos grandes grupos:

 Los lenguajes máquina.

 Los lenguajes simbólicos. Lenguaje de programación en el que las instrucciones de los diferentes programas se codifican utilizando los caracteres de las lenguas naturales. La ejecución de un programa.

A los lenguajes máquina les sucedieron, los lenguajes simbólicos los cuales utilizan caracteres naturales para escribir las instrucciones de los programas. Los lenguajes simbólicos se dividen a su vez en:

 Lenguajes simbólicos de bajo nivel o ensambladores.

 Lenguajes simbólicos de alto nivel.

A los lenguajes máquina se les conoce como lenguajes de primera generación. Los ensambladores son los lenguajes de segunda generación. Los simbólicos de alto nivel de tipo procedural se les denomina de tercera generación y a los relacionales se considera que son de cuarta generación. A la tercera generación pertenecen lenguajes conocidos como el FORTRAN, COBOL, RPG, BASIC, PL1, SIMULA, ALGOL, PASCAL, ADA, C, LISP, PROGOL, etc. Estos lenguajes de tercera generación son de tipo universal. Entre los lenguajes de cuarta generación tenemos el CSP de IBM asociado al sistema de gestión de base de datos DB2; el NATURAL de SOFTWARE AG asociado al sistema de datos ADABAS; el SQL que es un lenguaje convertido en estándar mundial como lenguaje de cuarta generación para la consulta de bases relacionales; etc.

Existen cuatro modelos básicos de computación que describen casi todos los lenguajes de programación actuales: el imperativo, el aplicativo, el lenguaje con base en reglas y el orientado a objetos. Se describe en forma breve cada uno de estos modelos.

**Lenguajes imperativos.** Los lenguajes imperativos o de procedimiento son lenguajes controlados por mandatos u orientados a enunciados (instrucciones). Un programa se compone de una serie de enunciados, y la ejecución de cada enunciado hace que el intérprete cambie el valor de una localidad o más en su memoria, es decir, que pase a un nuevo estado. El desarrollo de programas consiste en construir los estados de máquina sucesivos que se necesitan para llegar a la solución. Ésta suele ser la primera imagen, que se tiene de la programación, y muchos lenguajes de uso amplio (por ejemplo, C, C++, FORTRAN, ALGOL, PL /I, Pascal, Ada, Smalltalk, COBOL) manejan este modelo.

**Lenguajes aplicativos**. Un punto de vista alternativo de la computación representado por un lenguaje de programación consiste en examinar la función que el programa representa y no sólo los cambios de estado conforme el programa se ejecuta, enunciado por enunciado. Esto se puede conseguir observando el resultado deseado en vez de los datos disponibles. En otras palabras, en vez de examinar la serie de estados a través de los cuales debe pasar la máquina para obtener una respuesta, la pregunta que se debe formular es: ¿Cuál es la función que se debe aplicar al estado de máquina inicial accediendo al conjunto inicial de variables y combinándolas en formas específicas para obtener una respuesta? Los lenguajes que hacen énfasis en este punto de vista se conocen como lenguajes aplicativos o funcionales.

**Lenguajes base en reglas**. Los lenguajes con base en reglas se ejecutan verificando la presencia de una cierta condición habilitadora y, cuando se satisface, ejecutan una acción apropiada. El lenguaje más común con base en reglas es Prolog, que también se conoce como de programación lógico, puesto que las condiciones habilitadoras básicas son ciertas clases de expresiones lógicas de predicados. La ejecución de un lenguaje reglas es similar a la de un lenguaje imperativo, excepto que los enunciados no secuenciales.

**Programación orientada a objetos**. En este tipo de lenguaje, se construyen objetos complejos de datos y luego designa un conjunto limitado de funciones para que operen con esos datos. Los objetos complejos se designan como extensiones de objetos más simples y heredan propiedades del objeto más sencillo. Al construir objetos a concretos de datos, un programa orientado a objetos gana la eficiencia de los lenguajes imperativos, y al construir clases de funciones que utilizan un conjunto restringido de objetos de datos, se construye la flexibilidad y confiabilidad del modelo aplicativo.

Algunas de las principales influencias en la evolución de diseños de lenguajes se listan a continuación:

**1.** **Capacidades de las computadoras.** Las computadoras han evolucionado de las máquinas pequeñas lentas y costosas máquinas de tubos de vacío de los años 50 a las supercomputadoras y microcomputadora s de hoy.

**2.** **Aplicaciones.** El uso de la computadora se ha difundido rápidamente, de la concentración original de aplicaciones militares críticas, científicas, negocios industriales de los años 50 donde el costo podría estar justificado en los juegos en computadora, en computadoras personales y aplicaciones en casi todas las áreas de la actividad humana de hoy.

**3.** **Métodos de programación.** Los diseños de lenguajes han evolucionado para re flejar, nuestra cambiante comprensión de los buenos métodos para escribir programas largos y complejos y para reflejar los cambios en el entorno en el cual se efectúa la programación.

**4. Métodos de implementación.** El desarrollo de mejores métodos de implementación ha influido en la selección de las características que se habrán de incluir en los nuevos diseños.

**5. Estudios teóricos**. La investigación de las bases conceptuales del diseño e implementación de lenguajes, a través del uso de métodos de matemáticos formal es, ha profundizado nuestro entendimiento de las fortalezas y debilidad es de las características de los lenguajes y, por tanto, ha influido en la inclusión de estas características en los nuevos diseños de lenguaje. **6. Estandarización.** La necesidad de lenguajes estándar que se puedan implementar con facilidad en una variedad de computadoras y que permita que los programas sean transportados de una computadora a otra ejerce una fuerte influencia conservadora sobre la evolución de los diseños de lenguajes.

La evolución de los lenguajes de programación ha estado guiada por la evolución de:

 Las computadoras y sus sistemas operativos.

 Las aplicaciones.

 Los métodos de programación.

 Los fundamentos teóricos.

 La importancia dada a la estandarización.

**Origen. (Autocódigos).**

El primer lenguaje de programación nació de las ideas de Charles Babage a mediados de 1800. Lo que él llamó Motor Analítico, pero por razones técnicas no se creó hasta mediados de 1900. Debido a que la máquina aún no estaba construida, el proyecto de Ada No estaba hecho, pero fue un punto de partida para el proyecto. Esta combinación permite probar la proposición de que una máquina de Turing no puede decidir si una oración es un sistema de pensamiento formal; pero también podemos demostrar que si la estadística se puede explicar en lenguaje natural con ayuda de la notación matemática, será así: ya que la información se almacena. Una vez que se logre la eficiencia, todavía es posible escribir una computadora digital capaz de realizar cálculos. Suficiente. Babbage preparó sus máquinas, pero durante la Segunda Guerra Mundial se produjeron importantes avances en áreas como las computadoras. El lenguaje ensamblador es solo una representación estándar de un lenguaje de máquina que es similar y reduce la tediosa programación de los lenguajes tradicionales. Sin embargo, programar eficazmente en cualquiera de estos niveles de lenguaje requiere una comprensión de la arquitectura subyacente de la máquina.

**FORTRAN**

En los años 50 se realizaron varios compiladores primitivos y fue en 1957 cuando apareció el primer compilador de FORTRAN. El compilador de FORTRAN (FORmula TRANslator) estaba diseñado para traducir a lenguaje máquina expresiones y operaciones matemáticas, e incluso permitía la manipulación de matrices. La aparición del FORTRAN fue un gran cambio para los programadores que no todos aceptaron de buen grado. No les gustaba que sus programas fueran tratados por la computadora como meros datos, y argumentaban que el código máquina generado por el compilador nunca podría ser tan eficiente como el escrito por ellos directamente. Esto no era generalmente así, puesto que el FORTRAN no fue diseñado pensando en crear un lenguaje bien estructurado sino pensando en crear un traductor de expresiones aritméticas a código máquina muy eficiente. Por ello, el diseño lógico de la computadora IBM 704 para el que fue creado casi puede deducirse del lenguaje FORTRAN. En diferentes versiones, cada vez más estructuradas, el lenguaje FORTRAN se ha utilizado extensivamente des de que apareció hasta hoy en día, y puede considerarse el lenguaje estándar del cálculo científico.

**COBOL**

Cobol evolucionó a través de varios cambios de diseño, comenzando con su primera versión en 1960 y terminando con su última revisión en 1974 (COBOL 1974). La implementación y el uso generalizado de COBOL llevaron a la estandarización de las definiciones del lenguaje. Esta configuración es el resultado de dos objetivos principales de diseño: separar los elementos del programa que dependen de la máquina y separar las descripciones algorítmicas para que cada elemento pueda modificarse sin afectar a los demás. La representación de datos COBOL es un concepto empresarial simple pero muy flexible. Los números y las cadenas son tipos de datos primitivos básicos. El idioma es conocido por su gramática similar al inglés, lo que hace que el contenido sea más fácil de leer. La mayoría de los primeros compiladores COBOL eran muy lentos, pero las mejoras recientes en la tecnología de compilación han dado como resultado compiladores COBOL que producen código rápido, preciso y ejecutable.

**ALGOL60**

Algoritmie Languge (lenguaje algortmico) Lenguaje de programación creado en los años 60 ́s que se usaba principalmente para el diseño de aplicaciones de cálculo, la versión más conocida data del año 1968 que se conoció con el nombre de Proporciona recursos para estructurar datos, similares a los que se pueden encontrar en lenguajes derivados de ALGOL 60 como Pascal, C o Modula -2. Permite sobrecarga de operadores y colocar una declaración en cualquier lugar donde pueda aparecer un enunciado (como en C++).

**LISP**

Lo más impresionante es la aquí valencia de formas entre los programas y los datos en el lenguaje en el cual se permite que las estructuras de datos se ejecuten como programas y que los programas se ejecuten como datos. LISP ofrece una amplia variedad de primarias para la creación, destrucción y modificación de listas (incluyendo las listas de propiedad). Las estructuras de control en LISP son relativamente simples. Las referencias de LISP se basan principalmente en la regla de asociación más reciente para referencias foráneas, que con frecuencia se implementan usando una simple lista enlazada de asociaciones actuales. LISP se implementa más fácilmente con un intérprete de software y simulación de software para todas las primarias la mayoría de las implementaciones también proporcionan un compilador que puede usarse para compilar las definiciones de la función seleccionada en código de máquina.

**APL**

Sus siglas significan (A Programming Language). Un lenguaje de programación. Este programa fue desarrollado por Kenneth Inverson a mediados de la década de 1960 para resolver problemas matemáticos. Este lenguaje se caracteriza por su brevedad y por su capacidad de generación de matrices y se utiliza en el desarrollo de modelos matemáticos. APL tiene dos características que lo distinguen y lo apartan de los otros lenguajes:

 Lenguaje interactivo. APL es el único lenguaje que está diseñado expresamente para ser interactivo.

 Proceso directo de estructuras completas de datos. En APL las operaciones primarias aceptan arreglos completos como argumentos y producen arreglos completos como resultados. Por lo tanto, la unidad básica de datos tiende a ser un arreglo completo.

**BASIC**

En general, y al contrario de lo que suele suceder con los más importantes lenguajes simbólicos de programación, en BASIC se suele trabajar en modo intérprete. Mientras se está en fase de desarrollo es muy conveniente trabajar en intérprete porque la inmediata ejecución del programa en cada momento permite detectar los errores con gran rapidez, lo que es muy positivo en un lenguaje destinado a principiantes que lógicamente cometen un gran número de errores al desarrollar un programa. El BASIC es un lenguaje poco estandarizado y hay diferencias bastantes apreciables del que se utiliza en unos y otros sistemas. Aunque es un lenguaje poco utilizado en la informática profesional, se ha empleado mucho en los microcomputadores y mientras en los primeros años de los 80 ́s fue el lenguaje en el que se solían desarrollar las aplicaciones en micros. Hay una versión avanzada del BASIC, denominada QUICK BASIC que además de dar mayores prestaciones presenta las características de un lenguaje de programación algo estructurado, evitando así el desarrollo de los típicos programas espagueti.

**PL1**

Abreviatura de Portfolio Language One. Es un lenguaje de programación desarrollado por Atari. Admite gráficos en movimiento, fotogramas, líneas, conjuntos de caracteres, archivos y cadenas, matemáticas dinámicas y programación dinámica. Lenguaje fácil de usar; las palabras son simples y las reglas también. Estos archivos son fáciles de usar. El lenguaje también incluye herramientas sofisticadas para admitir el uso de archivos externos para entrada y salida. La programación PL/l es difícil debido a la complejidad del lenguaje y la necesidad de compilar el código correcto.

**SIMULA67.**

Esta versión surgió en 1967 es una extensión de Algol 60 bastante diferente a su predecesor (Simula I), que era básicamente un lenguaje de simulación de procesos. Simula 67 es un lenguaje de programación de propósito general que surge de los modelos de simulación. Parte del éxito de este lenguaje se debe a que se realizaron implementaciones para ordenadores IBM, DEC, Control Data y UNIVAC. SIMULA 67 fue el primer lenguaje de programación que incorporó el concepto de clase. La clase de Simula se puede considerar como una generalización muy flexible del concepto de bloque de ALGOL. Su mecanismo de definición de subclases permite establecer jerarquías de clases. Influyó directamente en las clases de C++ y SMALLTALK e indirectamente, a través de MESA, en los módulos de Modula -2. Su concepto de clase también de adaptó en lenguajes como CLU o ALPHARD.

**ALGOL 68**

Se basa en la idea de que la dificultad del lenguaje es la razón de sus limitaciones. Niklaus Wirth, sin embargo, prefiere suavizar el tono para tener un mejor control. Wirth creía que era un error pensar que una característica importante del lenguaje de alto nivel era la capacidad de expresar contenidos lo más rápido posible. ALGOL 68 rara vez se utiliza porque el informe original usaba palabras nuevas y era difícil de leer. ALGOL 68 utiliza la notación VWF (forma Van Wijngaarden) para definir su gramática y el inglés formal para definir su vocabulario. La notación utilizada es más completa que la BNF y se adapta mejor a la definición del lenguaje de programación.

**PASCAL.**

El lenguaje de programación Pascal fue desarrollado por Niklaus Wirth, miembro de la Federación Internacional para el Procesamiento de la Información (IFIP), quien desarrolló Pascal para proporcionar funciones que no estaban disponibles en otros lenguajes en ese momento. Su principal objetivo para Pascal era que el lenguaje fuera funcional y manejable, que permitiera estructurar y programar y que sirviera como medio para enseñar conceptos informáticos importantes. Un programa Pascal contiene un bloque de programa básico que contiene las definiciones de las subrutinas utilizadas. Pascal permite a los programadores definir tipos de datos de forma única. Pascal se puede implementar con éxito en computadoras modernas.

**PROLOG (programación lógica)**

Se trata de computadoras que tienen el PROLOG como lenguaje nativo (lenguaje máquina), con capacidad para procesar millones de inferencias lógicas por segundo (LIPS). La programación lógica, representada por lenguajes como Prolog, permite utilizar un subconjunto de la lógica de primer orden, concretamente las cláusulas de Horn, para especificar sistemas. Su base formal permite razonar sobre estas especificaciones, que pueden ser interpretadas directamente mediante resolución, con lo que podemos realizar pruebas de la especificación. Existen lenguajes lógicos concurrentes, como Parlog, que subsanan estas deficiencias. Otro inconveniente que presentan estos lenguajes es que no admiten tipos de datos, ni siquiera definiciones de datos, por lo que son poco adecuados para especificaciones complejas. Por lo tanto, podemos decir que los lenguajes lógicos, aunque a veces se utilizan para la especificación, no son en sí mismos lenguajes de especificación.

**C**

El lenguaje C reúne características de programación intermedia entre los lenguajes ensambladores y los lenguajes de alto nivel; con gran poderío basado en sus operaciones a nivel de bits (propias de ensambladores) y la mayoría de los elementos de la programación estructurada de los lenguajes de alto nivel, por lo que resulta ser el lenguaje preferido para el desarrollo de software de sistemas y aplicaciones profesionales de la programación de computadoras. Su diseño incluyo una sintaxis simplificada, la aritmética de direcciones de memoria (permite al programador manipular bits, bytes y direcciones de memoria) y el concepto de apuntador; además, al ser diseñado para mejorar el software de sistemas, se buscó que generase códigos eficientes y uno portabilidad total, es decir el que pudiese correr en cualquier máquina. Logrados los objetivos anteriores, C se convirtió en el lenguaje preferido de los programadores profesionales. Con este nuevo enfoque surge la nueva metodología que aumenta las posibilidades de la programación bajo nuevos conceptos.

**MODULA -2**

Modula -2 es descendiente directo de Pascal. Modula -2 tiene poderosas extensiones, especialmente su concepto de “modelo” (el cual permite la compilación independiente y la creación de librerías de programas) y el permitir el procesamiento asíncrono. A pesar de su potencia, existen eficientes implementaciones de Modula -2 y junto con C y Ada, debe considerarse un importante candidato para las futuras aplicaciones de programación de sistemas. Modula -2, estuvo influenciado por las necesidades especiales de la programación de sistemas, especialmente la necesidad de la compilación independiente de procedimiento.

**SMALLTACK.**

Smalltalk es un lenguaje de programación confiable orientado a objetos integrado en un entorno de desarrollo de ventanas múltiples. El entorno de desarrollo integrado Smalltalk tiene un elemento "divertido de usar". Anteriormente, no estaba disponible en los idiomas que debían editarse. Sin embargo, no se distribuyó ampliamente entre la comunidad de desarrolladores hasta la creación de una nueva empresa, ParcPlace Systems Inc., dirigida por Adele Goldberg. OTI desarrolló la máquina virtual de 32 bits para productos Digitalk para Apple y participó en muchos proyectos de investigación, desde herramientas cliente-servidor orientadas a objetos, Smalltalk integrado y procesamiento de imágenes de radar para operaciones militares en sistemas cautivos y sistemas virtuales portátiles. . fabricante de máquinas IBM se asoció con Object Technology International Inc. Desarrollar la línea de productos VisualAge para Smalltalk. Hoy en día, ObjectShare (anteriormente ParcPlace -Digitalk) e IBM siguen siendo los principales distribuidores de los entornos de desarrollo Smalltalk.

**ADA.**

Ada es un lenguaje de programación de alto nivel diseñado para desarrollar aplicaciones con precisión, seguridad, confiabilidad y mantenibilidad como objetivos principales. comenzó a desarrollar lenguajes de programación estándar para aplicaciones de sistemas integrados, donde una computadora es uno o más componentes de un sistema grande, como un avión, un barco o un software de computadora. Ada está diseñada para apoyar a los equipos de programación en la creación de grandes proyectos; Los programas en Ada están diseñados como colecciones de "componentes de software". Esta característica única de Ada y la programación de Ada proviene del énfasis en la creación de programas utilizando paquetes. La característica más notable de las capacidades de procesamiento secuencial de Ada es que proporciona tareas que pueden ejecutarse simultáneamente y controlarse mediante relojes u otros mecanismos de programación. Las estructuras de gestión de datos en Ada están organizadas en subrutinas (como Pascal) utilizando estructuras de bloques estáticos para referencias externas.