**Universidad Nacional Autónoma de México**

**Facultad de Ingeniería**

**División de Ingeniería Eléctrica**

**Nombre:** Flores Saavedra José Gerardo

**No. de Cuenta:** 318146276

**Correo:** [jgfs1209@gmail.com](mailto:jgfs1209@gmail.com)

**Materia:** Fundamentos de Programación

**Maestro:** M. I. Marco Antonio Martínez Quintana

**Actividad Asíncrona #1:** Historia de la programación

**Grupo:** 3

**Semestre:** 2021-1

**Fecha:** 2/octubre/2020

**Historia de la programación**

**Historia del cómputo**

*El ábaco*

Empecemos con el [ábaco](http://www.syssrc.com/html/museum/html/abacus.html) que es posiblemente el primer dispositivo mecánico de contabilidad de la historia. Tiene unos 5.000 años de antigüedad, y su efectividad ha soportado la prueba del tiempo siendo un dispositivo sencillo: una serie de cuentas ensartadas en varillas que a su vez están montadas en un marco rectangular. Al desplazar las cuentas sobre varillas, sus posiciones representan valores almacenados. A pesar de su capacidad para representar y almacenar datos, a este dispositivo no se le puede llamar computadora, puesto que –entre otras cosas– carece del elemento fundamental llamado programa.

*Calculadoras mecánicas*

Leonardo Da Vinci trazó alrededor de 1500 varios apuntes para una sumadora mecánica. Más de un siglo después, hacia 1623, el alemán Wilhelm Schickard construyó la primera máquina de calcular, sin embargo, la Historia ha reservado el puesto de creador del primer ingenio mecánico calculador a Pascal.

En 1642, el filósofo y matemático francés Blaise Pascal construyó la primera sumadora mecánica, que se llamó Pascalina, y que funcionaba con un complicado mecanismo de engranes y ruedas: la rotación completa de una de las ruedas dentadas hacía girar un paso a la rueda siguiente. La Pascalina sólo realizaba sumas y restas.

Por su parte, el alemán Gottfried Wilhelm von Leibniz diseñó en 1671 otra sumadora mecánica, que concluyó definitivamente en 1694, conocida como la Calculadora Universal o Rueda Escalada de Leibniz, capaz de realizar sumas, restas, divisiones y raíces cuadradas. En estas calculadoras mecánicas, los datos, representados mediante las posiciones de los engranajes, se introducían manualmente, estableciendo dichas posiciones finales de las ruedas de manera similar a como leemos los números en el cuentakilómetros de un automóvil.

*Tarjetas perforadas*

El fabricante de tejidos francés Joseph-Marie Jacquard ideó en 1801 un telar, todavía utilizado en la actualidad, que podía reproducir automáticamente patrones de tejidos leyendo la información codificada en patrones de agujeros perforados en tarjetas de papel rígido. Las tarjetas se perforaban estratégicamente y se acomodaban en cierta secuencia para indicar un diseño de tejido en particular. Para cambiar de diseño, basta con preparar una nueva colección de tarjetas.

El telar consta de una serie de varillas, sobre las que pasan las tarjetas, y de las que están prendidos hilos de distintos colores. Las perforaciones de las tarjetas determinan de manera mecánica qué varillas –y por tanto qué hilos– intervienen en la formación del tejido y en la disposición de los dibujos.

El principio de las tarjetas perforadas de Jacquard es el mismo que rige el funcionamiento de ciertos [aparatos musicales](http://paginaspersonales.deusto.es/airibar/Ed_digital/INF/Intro/pianola.html). Más tarde, las tarjetas perforadas tendrán una gran influencia en los trabajos de [Charles Babbage](http://paginaspersonales.deusto.es/airibar/Ed_digital/INF/Intro/Historia.html#14).

El telar de Jacquard supone una triple aportación teórica para el futuro desarrollo computacional:

* Proporciona un modelo de automatización de los procesos de producción diversificada (opuesta a la de un solo propósito o específica).
* Por primera vez se realiza una codificación de la información. Las tarjetas son la información suministrada (*input*) y el tejido es el resultado (*output*).
* Por primera vez se realiza la programación de las instrucciones. La cadena de tarjetas perforadas prefigura la organización de los procesos mediante técnicas de programación.

*Máquina analítica de Babbage*

Charles Babbage, visionario científico y matemático inglés, fue el más claro precursor del hardware computacional, hasta el punto de que se le considera el padre histórico de la computación. Preocupado desde su juventud por los frecuentes errores cometidos en el cálculo de las tablas numéricas (y en la posterior impresión de sus resultados), ideó la [Máquina de Diferencias](http://paginaspersonales.deusto.es/airibar/Ed_digital/INF/Intro/M_diferencias.html) (Difference Engine), cuyo modelo definitivo es de 1823, capaz de calcular –e imprimir– tablas matemáticas de hasta veinte cifras con ocho decimales y polinomios de sexto grado. Babbage, subvencionado por el gobierno británico desde 1823, trabajó durante años en el perfeccionamiento de su Máquina de Diferencias, pero finalmente tuvo que abandonar el proyecto, al agotarse la ayuda económica antes de haber podido construir una máquina con el refinamiento técnico que exigía su diseño.

Babbage reaccionó ante el aparente fracaso de su invento con un proyecto aún más ambicioso, y en 1834 concibió su revolucionaria [Máquina Analítica](http://paginaspersonales.deusto.es/airibar/Ed_digital/INF/Intro/M_analitica.html) (Analytical Engine), que puede considerarse un auténtico prototipo decimonónico de ordenador. En esencia, la Máquina Analítica era una calculadora polivalente con capacidad para operar de forma distinta según el problema que se le planteara, es decir, algo muy cercano a una computadora de propósito general. En la máquina de Babbage aparecen ya los elementos básicos de los modernos ordenadores: dispositivos de entrada y de salida, unidad de control, unidad lógico-aritmética y memoria. La programación se debía realizar mediante fichas perforadas. A pesar de su extraordinaria brillantez, el ambicioso proyecto no pudo realizarse por razones económicas e industriales, puesto que la tecnología de la época no bastaba para hacer realidad el proyecto: el diseño requería miles de engranes y mecanismos de gran precisión que cubrirían el área de un campo de futbol y necesitarían accionarse por una locomotora.

Charles Babbage trabajó hasta su muerte en su Máquina Analítica (a la que se le bautizó como La locura de Babbage), de la que sólo pudo construir algún fragmento. Sin embargo, sus notas describían asombrosamente casi todas las características incorporadas hoy en la moderna computadora electrónica. Si Babbage hubiera vivido en la era de la tecnología electrónica y los componentes de precisión, seguramente habría adelantado varias décadas el nacimiento de la computadora electrónica. Sin embargo, su obra cayó en un olvido tan completo que algunos pioneros del desarrollo de la computadora electrónica ignoraron por completo sus ideas sobre memoria, impresoras, tarjetas perforadas y control de programa-secuencia. En 1843, [Lady Ada Augusta Lovelace](http://www.dma.eui.upm.es/historia_informatica/Doc/Personajes/AdaByron.htm), estrecha colaboradora de Babbage, sugirió la idea de que las tarjetas perforadas pudieran adaptarse de manera que propiciaran que el motor de Babbage repitiera ciertas operaciones. Debido a esta sugerencia algunas personas consideran a Lady Lovelace la primera programadora.

*La Tabuladora de Hollerith*

A finales del siglo XIX, la oficina de censos estadounidense se enfrentaba a un grave problema: había tardado ocho años en finalizar el censo de 1880, y había llegado a la conclusión de que el censo de cada diez años tardaría en terminarse más que esos mismos diez años.

Para intentar solucionar el problema, la oficina de censos encargó al estadístico Herman Hollerith que aplicara su experiencia en tarjetas perforadas y llevara a cabo el censo de 1890. Hollerith desarrolló una máquina tabuladora, que se componía de un lector de tarjetas, un contador, un clasificador y un aparato de tabular. Con el procesamiento de las tarjetas perforadas y el tabulador de tarjetas perforadas de Hollerit, el censo se terminó en sólo 3 años y la oficina se ahorró alrededor de 5 millones de dólares. Así comenzó el procesamiento automatizado de datos.

Hollerit no tomó la idea de las tarjetas perforadas del telar de Jacquard, sino de la fotografía de perforación: algunas líneas ferroviarias de la época expedían boletos con descripciones físicas del pasajero; los conductores hacían orificios en los boletos que describían el color de cabello, de ojos y la forma de nariz del pasajero. Eso le dio a Hollerith la idea para hacer la fotografía perforada de cada persona que se iba a tabular.

*Máquinas electromecánicas*

Los resultados de las máquinas tabuladoras tenían que llevarse al corriente por medios manuales, hasta que en 1919 la Computing-Tabulating-Recording-Company anunció la aparición de la impresora/listadora. Esta innovación revolucionó la manera en que las empresas efectuaban sus operaciones. Desde los años cincuenta, la tecnología de las tarjetas perforadas se perfeccionó con la implantación de más dispositivos con capacidades más complejas. Dado que cada tarjeta contenía en general un registro (un nombre, una dirección, etc.), el procesamiento de la tarjeta perforada se conoció también como procesamiento de registro unitario.

La familia de las máquinas electromecánicas de contabilidad (EAM: Eloctromechanical Accounting Machine) de dispositivos de tarjeta perforada comprende: la perforadora de tarjetas, el verificador, el reproductor, la perforación sumaria, el intérprete, el clasificador, el otejador, el calculador y la máquina de contabilidad.

El operador de un cuarto de máquinas en una instalación de tarjetas perforadas realizaba un trabajo que demandaba mucho esfuerzo físico. Algunos cuartos de máquinas asemejaban la actividad de una fábrica: las tarjetas perforadas y las salidas impresas se cambiaban de un dispositivo a otro en carros manuales; el ruido que producía eran tan intenso como el de una planta ensambladora de automóviles.

*Computadora ABC*

Una antigua patente de un dispositivo que muchos creyeron que era la primera computadora digital electrónica se invalidó en 1973 por orden de un tribunal federal, y oficialmente se le dió el credito a John V. Atanasoff como el inventor de la computadora digital electrónica.

El Dr. Atanasoff, catedrático de la Universidad Estatal de Iowa, desarrolló la primera computadora digital electrónica entre los años de 1937 a 1942 llamando a su invento la computadora Atanasoff-Berry, o [ABC](http://www.dma.eui.upm.es/historia_informatica/Doc/Maquinas/ABC.htm) (Atanasoff Berry Computer). Clifford Berry, un estudiante graduado, fue una útil ayuda en la construcción de la computadora ABC.

Algunos autores consideran que no hay una sola persona a la que se le pueda atribuir el haber inventado la computadora, sino que fue el esfuezo de muchas personas. Sin embargo en el antiguo edificio de Física de la Universidad de Iowa aparece una placa con la siguiente leyenda: La primera computadora digital electrónica de operación automática del mundo, fue construida en este edificio en 1939 por John Vincent Atanasoff, matemático y físico de la Facultad de la Universidad, quien concibió la idea, y por Clifford Edward Berry, estudiante graduado de física.

*Mark-I*

En 1944 se presentó la [Mark-I](http://www.dma.eui.upm.es/historia_informatica/Doc/Maquinas/HarvardMarkI.htm) (o IBM ASCC), diseñada por un equipo encabezado por [Howard H. Aiken](http://www.dma.eui.upm.es/historia_informatica/Doc/Personajes/HowardAiken.htm), de la Universidad de Harvard, con la colaboración de IBM. Era una máquina automática eléctrica, aunque tenía componentes electromecánicos; podía realizar 5 operaciones aritméticas: suma, resta, multiplicación, división y referencia a resultados anteriores.

Computadora ENIAC

[John W. Mauchly](http://www.dma.eui.upm.es/historia_informatica/Doc/Personajes/JohnMauchly.htm) y [John Presper Eckert](http://www.dma.eui.upm.es/historia_informatica/Doc/Personajes/JohnEckert.htm), después de estudiar prolijamente la computadora ABC, desarrollaron en la Universidad de Pennsylvania una máquina capaz de calcular tablas de trayectoria para el ejército estadounidense. El producto final, una computadora electrónica completamente operacional a gran escala, se terminó en 1946 y se llamó [ENIAC](http://www.dma.eui.upm.es/historia_informatica/Doc/Maquinas/ENIAC.htm) (Electronic Numerical Integrator And Computer: integrador numérico y calculador electrónico). La ENIAC, construida para aplicaciones de la Segunda Guerra mundial, se terminó en treinta meses por un equipo de científicos que trabajaban contra reloj.

La ENIAC supuso un hito en la tecnología de la computación. Era mil veces más veloz que sus predecesoras electromecánicas y poseía la capacidad de realizar 5.000 operaciones aritméticas en un segundo. No obstante, la máquina pesaba 30 toneladas, ocupaba un espacio de 450 m2 (llenaba todo un sótano de la universidad), contenía 18.000 bulbos (tubos de vacío), requería su completo sistema de aire acondicionado y tenía que programarse manualmente conectándola a 3 tableros que contenían más de 6.000 interruptores. Además, introducir un nuevo programa era un proceso muy tedioso que requería días o incluso semanas.

A diferencia de las computadoras actuales, que operan con un sistema binario, la ENIAC operaba con el sistema decimal. La ENIAC requería una gran cantidad de electricidad (se dice que bajaba la intensidad de las luces de Filadelfia siempre que se activaba). La imponente escala y las numerosas aplicaciones generales de la ENIAC señalaron el comienzo de la primera generación de computadoras.

Computadora EDVAC

En 1945, el ingeniero y matemático húngaro [John von Neumann](http://www.dma.eui.upm.es/historia_informatica/Doc/Personajes/JohnvonNeumann.htm), que había trabajado con Eckert y Mauchly en la Universidad de Pennsylvania, publicó un artículo acerca del almacenamiento de programas. El concepto de programa almacenado permitía la lectura de un programa dentro de la memoria de la computadora y la posterior ejecución de las instrucciones del mismo sin tener que volverlas a escribir.

Hacia 1948, von Neumann se integra en el equipo de Eckert y Mauchly auspiciado por el Departamento de Defensa de los Estados Unidos. Sus ideas resultaron tan fundamentales para su desarrollo posterior, que es considerado el padre de las computadoras.

Este nuevo equipo crea la computadora [EDVAC](http://www.dma.eui.upm.es/historia_informatica/Doc/Maquinas/EDVAC.htm) (Electronic Discrete Variable Automatic Computer), la primera en aplicar la idea básica de von Neumann: permitir que en la memoria coexistan datos e instrucciones, para que entonces la computadora pueda ser programada en un lenguaje, y no por medio de alambres que eléctricamente conecten varias secciones de control, como en la ENIAC. La EDVAC tenía aproximadamente cuatro mil bulbos y usaba un tipo de memoria basado en tubos llenos de mercurio por donde circulaban señales eléctricas sujetas a retardos.

Los programas almacenados dieron a las computadoras mucha más flexibilidad y confiabilidad, haciéndolas más rápidas y menos sujetas a errores que los programas mecánicos. Una computadora con capacidad de programa almacenado podría ser utilizada para varias aplicaciones, cargando y ejecutando el programa apropiado.

*Programas intérpretes*

Hasta este punto, los programas y los datos podían ser introducidos en la computadora sólo con la notación binaria, que es el único código que las computadoras entienden. El siguiente desarrollo importante en el diseño de las computadoras fueron los programas intérpretes, que permitían a las personas comunicarse con las computadoras utilizando medios distintos a los números binarios.

En 1952, Grace Murray Hoper, una oficial de la Marina de los Estados Unidos, desarrolló el primer compilador, un programa que puede traducir enunciados parecidos al inglés en un código binario comprensible para la máquina, llamado [COBOL](http://www.dma.eui.upm.es/historia_informatica/Doc/Lenguajes/Cobol.htm) (Common Business-Oriented Languaje).

*Primera generación de Computadoras (1951-1958)*

Las computadoras de la primera generación emplearon bulbos para procesar información. Los operadores introducían los datos y los programas en un código especial por medio de tarjetas perforadas. El almacenamiento interno se lograba con un tambor que giraba rápidamente, sobre el cual un dispositivo de lectura/escritura colocaba marcas magnéticas. Esas computadoras de bulbos eran mucho más grandes y generaban más calor que los modelos contemporáneos.

Eckert y Mauchly contribuyeron al desarrollo de computadoras de la primera generación formando una compañía privada y construyendo [UNIVAC](http://www.dma.eui.upm.es/historia_informatica/Doc/Maquinas/UNIVAC.htm), que el Comité del Censo estadounidense utilizó para evaluar el de 1950.

Por su parte, la IBM tenía el monopolio de los equipos de procesamiento de datos a base de tarjetas perforadas y estaba teniendo un gran auge en productos como rebanadores de carne, básculas para comestibles, relojes y otros artículos; sin embargo no había logrado el contrato para el Censo de 1950. Comenzó entonces a construir computadoras electrónicas y su primera entrada fue con la IBM 701 en 1953, un producto que tardó algún tiempo es ser comercialmente viable.

En 1954 fue introducido el modelo IBM 650, que es en gran medida la razón por la que IBM disfruta hoy de una gran parte del mercado de las computadoras. La administración de la IBM asumió un gran riesgo y estimó una venta de 50 computadoras, número mayor que la cantidad de computadoras instaladas en esa época en los Estados Unidos. Sin embargo, la IBM instaló 1000 computadoras. Aunque caras y de uso limitado, las computadoras fueron aceptadas rápidamente por las Compañías privadas y de Gobierno. A la mitad de los años 50, IBM y Remington Rand se consolidaban como los líderes en la fabricación de computadoras.

*Segunda generación de Computadoras (1959-1964)*

El invento del transistor hizo posible una nueva generación de computadoras, más rápidas, más pequeñas y con menores necesidades de ventilación. Sin embargo el costo seguía siendo una porción significativa del presupuesto de una Compañia.

Las computadoras de la segunda generación utilizaban redes de nucleos magnéticos en lugar de tambores giratorios para el almacenamiento primario. Estos núcleos contenían pequeños anillos de material magnético, enlazados entre sí, en los cuales se almacenaban datos e instrucciones.

Los programas de computadoras también mejoraron. El COBOL, desarrollado durante la generación anterior, estaba ya disponible comercialmente. Los programas escritos para una computadora podían transferirse a otra con un mínimo esfuerzo. El escribir un programa ya no requería entender plenamente el hardware de la computación. Las computadoras de la segunda generación eran sustancialmente más pequeñas y rápidas que las de bulbos, y se usaban para nuevas aplicaciones, como en los sistemas para reserva en líneas aéreas, control de tráfico aéreo y simulaciones para uso general.

Las empresas comenzaron a aplicar las computadoras a tareas de almacenamiento de registros, como manejo de inventarios, nómina y contabilidad. La marina de los Estados Unidos utilizó las computadoras de la segunda generación para crear el primer simulador de vuelo (el Whirlwind I). HoneyWell se colocó como el primer competidor durante la segunda generación de computadoras. Burroughs, Univac, NCR, CDC, HoneyWell, los más grandes competidores de IBM durante los años sesenta, se conocieron como el grupo BUNCH.

*Tercera generación de Computadoras (1964-1971)*

Las computadoras de la tercera generación nacieron con el desarrollo de los circuitos integrados (pastillas de silicio), en los cuales se colocan miles de componentes electrónicos, en una integración en miniatura. Las computadoras se hicieron nuevamente más pequeñas, más rápidas, desprendían menos calor y eran energéticamente más eficientes. Antes del advenimiento de los circuitos integrados, las computadoras estaban diseñadas para aplicaciones matemáticas o de negocios, pero no para las dos cosas. Los circuitos integrados permitieron a los fabricantes de computadoras incrementar la flexibilidad de los programas y estandarizar sus modelos.

La [IBM 360](http://www.dma.eui.upm.es/historia_informatica/Doc/Maquinas/IBM360.htm), una de las primeras computadoras comerciales que usó circuitos integrados, podía realizar tanto análisis numéricos como administración o procesamiento de archivos. Los clientes podían escalar sus sistemas 360 a modelos IBM de mayor tamaño y podían todavía ejecutar sus programas actuales. Las computadoras trabajaban a tal velocidad que proporcionaban la capacidad de ejecutar más de un programa de manera simultánea (multiprogramación). Por ejemplo, la computadora podía estar calculando la nómina y aceptando pedidos al mismo tiempo.

Con la introducción del modelo 360, IBM acaparó el 70% del mercado. Para evitar competir directamente con IBM, la empresa Digital Equipment Corporation (DEC) redirigió sus esfuerzos hacia computadoras pequeñas. Mucho menos costosas de comprar y de operar que las computadoras grandes, las minicomputadoras se desarrollaron durante la segunda generación pero alcanzaron su mayor auge entre 1960 y 1970.

*Cuarta generación de Computadoras (1971-???)*

Dos mejoras en la tecnología de las computadoras marcan el inicio de la cuarta generación: el reemplazo de las memorias con núcleos magnéticos por las de chips de silicio y la colocación de muchos más componentes en un chip, producto de la microminiaturización de los circuitos electrónicos. El tamaño reducido del microprocesador de chips hizo posible la creación de las computadoras personales (PC). El [primer PC](http://www.dma.eui.upm.es/historia_informatica/Doc/Maquinas/IBMPC.htm), de IBM, es de 1981. Antes, ya se habían presentado otros dos ordenadores personales: el [Apple-II](http://www.dma.eui.upm.es/historia_informatica/Doc/Maquinas/Apple.htm) y el [Altair 8800](http://www.dma.eui.upm.es/historia_informatica/Doc/Maquinas/Altair8800.htm). Hoy en día, las tecnologías LSI (Integración a gran escala) y VLSI (integración a muy gran escala) permiten que cientos de miles de componentes electrónicos se almacenen en un chip. Usando VLSI, un fabricante puede hacer que una computadora pequeña rivalice con una computadora de la primera generación que ocupara un cuarto completo.

**Historia de la programación**

La historia de la Programación está relacionada directamente con la aparición de los computadores, que ya desde el siglo XV tuvo sus inicios con la construcción de una máquina que realizaba operaciones básicas y raíces cuadradas (Gottfried Wilheml von Leibniz); aunque en realidad la primera gran influencia hacia la creación de los computadores fue la  máquina diferencial para el cálculo de polinomios, proyecto no concluido de Charles Babbage con el apoyo de Lady Ada Countess of Lovelace, primera persona que incursionó en la programación y de quien proviene el nombre del lenguaje de programación ADA creado por el DoD (Departamento de defensa de Estados Unidos) en la década de 1970.

Por eso, pondremos una tabla resumiendo lo más importante sobre la historia de la programación

|  |  |  |
| --- | --- | --- |
| Personaje | Aporte | Año |
| Gottfried Wilheml von Leibniz | Máquinas de operaciones básicas | XV |
| Charles Babbage | Máquina diferencial para el cálculo de polinomios | XVII |
| Lady Ada Countess of Lovelace | Almacenamiento en tarjetas perforadas | XVII |
| Herman Hollerit | Creador de un sistema para automatizar la pesada tarea del censo | 1890 |
| Alan Mathison Turing | Máquina de Turing - una máquina capaz de resolver problemas - Aportes de Lógica Matemática - Computadora con tubos de vacío | 1943 |
| George Boole | Contribuyó al algebra binaria y a los sistemas de circuitos de computadora (álgebra booleana). | 1854 |
| John Vincent Atanasoff | Primera computadora digital electrónica patentada - Atanasoff Berry Computer (ABC) | 1942 |
| Howard Aiken | En colaboración con IBM desarrolló el Mark 1, una computadora electromecánica de 16 metros de largo y más de dos de alto que podía realizar las cuatro operaciones básicas y trabajar con información almacenada en forma de tablas. | 1944 |
| John W. Mauchly | Junto a John Presper Eckert desarrolló una computadora electrónica completamente operacional a gran escala llamada Electronic Numerical Integrator And Computer (ENIAC). | 1946 |
| Johann Ludwig Von Neumann | Propuso guardar en memoria no solo la información, sino también los programas, acelerando los procesos. | 1945 |

**Lenguajes de programación**

Es un lenguaje formal que, mediante una serie de instrucciones, le permite a un programador escribir un conjunto de órdenes, acciones consecutivas, datos y [algoritmos](https://rockcontent.com/es/blog/tipos-de-algoritmos/) para, de esa forma, crear programas que controlen el comportamiento físico y lógico de una máquina.

Mediante este lenguaje se comunican el programador y la máquina, permitiendo especificar, de forma precisa, aspectos como:

* Cuáles datos debe operar un software específico;
* Cómo deben ser almacenados o transmitidos esos datos;
* Las acciones que debe tomar el software dependiendo de las circunstancias variables.

Para explicarlo mejor (en otras y con menos palabras), el lenguaje de programación es un sistema estructurado de comunicación, el cual está conformado por conjuntos de símbolos, [palabras claves](https://rockcontent.com/es/blog/palabras-clave/), reglas semánticas y sintácticas que permiten el entendimiento entre un programador y una máquina.

Es importante recalcar que existe el error común de usar como sinónimos el lenguaje de programación y el lenguaje informático, pero ¿por qué no debemos confundirlos?

Pues, es debido a que el lenguaje de programación obedece a un conjunto de reglas que permiten expresar las instrucciones que serán interpretadas por el [programador](https://rockcontent.com/es/blog/desarrollador-web/). Y el lenguaje informático comprende otros lenguajes que dan formato a un texto pero no son programación en sí mismos.

Los leguajes de programación son:

* PHP: De **código abierto**, enfocado para el **desarrollo**[web](https://www.muyinteresante.es/tecnologia/fotos/mitos-de-internet). Entre las muchas cosas que pueden realizarse con PHP destacan la creación de web dinámicas al poderse incrustar su código dentro de HTML y su uso como scripts**del lado del servidor**.
* Java: O**rientado a objetos** que destaca por ser **multiplataforma**. El código de Java tiene la gran ventaja de que puede ser ejecutado en cualquier dispositivo. Java es uno de los lenguajes más utilizados actualmente gracias al uso que se hace de distintas plataformas (Smartphone, ordenadores y tabletas, principalmente).
* Python: M**uy versátil** de código abierto, muy utilizado actualmente para **inteligencia artificial** y el Big Data. El lenguaje es sencillo, fácilmente legible y elegante, disponiendo de una curva de aprendizaje muy corta en comparación con otros tipos de lenguaje. Python además se puede utilizar en prácticamente cualquier sistema operativo actual.
* C/C++: Lo eligen los informáticos cuando se necesita una **programación potente**que incluso necesite bajar a programar algunas tareas en **ensamblador**. A pesar de tener una curva de aprendizaje alta, sigue siendo muy solicitado en la programación en general y en la de videojuegos en particular.
* JavaScript: Muy utilizado en la **programación web** del lado del cliente. Además de ser muy práctico, es capaz de ser ejecutado en cualquier **navegador**, por lo que es el [lenguaje](https://www.muyinteresante.es/cultura/arte-cultura/articulo/el-lenguaje-de-internet) más utilizado en la web. Javascript es multiplataforma, está orientado a objetos y eventos, y es interpretado; es decir, no necesita compilarse.
* C# y Visual Basic: Orientados a objetos de **Microsoft**que utilizan su **framework.net**. Son muy utilizados para crear una gran variedad de aplicaciones.
* Objective-C: B**asado en C** y destinado al desarrollo de aplicaciones para **iOS y macOS**. Se trata de un lenguaje orientado a objetos con el que se desarrollan la mayoría de [*apps*](https://www.muyinteresante.es/tecnologia/fotos/las-mejores-apps-para-hacer-videollamadas-de-grupo) para iPhone o iPad.
* Ruby: Interpretado donde todo es un objeto
* Swift: Para iOS y macOS
* SQL: Para obtener información de una base de datos
* Delphi: Object Pascal
* Perl: Para aplicaciones web CGI

Clasificación de los lenguajes de programación

* De Bajo Nivel: Son lenguajes totalmente orientados a la máquina. Este lenguaje sirve de interfaz y crea un vínculo inseparable entre el [hardware y el software](https://rockcontent.com/es/blog/hardware-y-software/). Además, ejerce un control directo sobre el equipo y su estructura física. Para aplicarlo adecuadamente es necesario que el programador conozca sólidamente el hardware. Éste se subdivide en dos tipos:
  + Lenguaje Máquina: Es el más primitivo de los lenguajes y es una colección de dígitos binarios o bits (0 y 1) que la computadora lee e interpreta y son los únicos idiomas que las computadoras entienden.
  + Lenguaje Ensamblador: es el primer intento de sustitución del lenguaje de máquina por uno más cercano al utilizado por los humanos. Un programa escrito en éste lenguaje es almacenado como texto (tal como programas de alto nivel) y consiste en una serie de instrucciones que corresponden al flujo de órdenes ejecutables por un microprocesador. Sin embargo, dichas máquinas no comprenden el lenguaje ensamblador, por lo que se debe convertir a lenguaje máquina mediante un programa llamado Ensamblador. Este genera códigos compactos, rápidos y eficientes creados por el programador que tiene el control total de la máquina.
* De Alto Nivel: Tienen como objetivo facilitar el trabajo del programador, ya que utilizan unas instrucciones más fáciles de entender. Además, **el lenguaje de alto nivel permite escribir códigos mediante idiomas que conocemos** (español, inglés, etc.) y luego, para ser ejecutados, se traduce al lenguaje de máquina mediante traductores o compiladores.
* Traductor: Traducen programas escritos en un lenguaje de programación al lenguaje máquina de la computadora y a medida que va siendo traducida, se ejecuta.
* Compilador: Permite traducir todo un programa de una sola vez, haciendo una ejecución más rápida y puede almacenarse para usarse luego sin volver a hacer la traducción.

Paradigmas de la programación

Un paradigma de programación es una manera o estilo de programación de software. Existen diferentes formas de diseñar un lenguaje de programación y varios modos de trabajar para obtener los resultados que necesitan los programadores.  Se trata de un conjunto de métodos sistemáticos aplicables en todos los niveles del diseño de programas para resolver problemas computacionales.

Los lenguajes de programación adoptan uno o varios paradigmas en función del tipo de órdenes que permiten implementar como, por ejemplo, **Python o**[JavaScript](https://profile.es/blog/librerias-javascript/), que son multiparadigmas.

*Paradigma imperativo*

Los programas consisten en una sucesión de instrucciones o conjunto de sentencias, como si el programador diera órdenes concretas. El [desarrollador](https://profile.es/blog/desarrollador-senior/) describe en el código paso por paso todo lo que hará su programa. Algunos lenguajes: **Pascal, COBOL, FORTRAN, C, C++**, etc.

Otros e**nfoques subordinados al paradigma de programación imperativa** son:

–          Programación estructurada: La programación estructurada es un tipo de programación imperativa donde el flujo de control se define mediante bucles anidados, condicionales y subrutinas, en lugar de a través de GOTO.

–          Programación procedimental: Este paradigma de programación consiste en basarse en un número muy bajo de expresiones repetidas, englobarlas todas en un procedimiento o función y llamarlo cada vez que tenga que ejecutarse.

–          Programación modular: consiste en dividir un programa en módulos o subprogramas con el fin de hacerlo más manejable y legible. Se trata de una evolución de la programación estructurada para resolver problemas de programación más complejos.

*Paradigma declarativo*

Este paradigma no necesita definir algoritmos, puesto que, describe el problema en lugar de encontrar una solución al mismo. Este paradigma utiliza el principio del razonamiento lógico para responder a las preguntas o cuestiones consultadas.

Este paradigma a su vez se divide en dos:

–          Programación Lógica: **Prolog**

–          Programación funcional: **Lisp, Scala, Java,**[Kotlin](https://profile.es/blog/descubriendo-kotlin-lenguaje-programacion/).

*Programación orientada a objetos*

Se construyen modelos de objetos que representan elementos (objetos) del problema a resolver, que tienen características y funciones. Permite separar los diferentes componentes de un programa, simplificando así su creación, depuración y posteriores mejoras. La programación orientada a objetos disminuye los errores y promociona la reutilización del código. Es una manera especial de programar, que se acerca de alguna manera a cómo expresaríamos las cosas en la vida real.

Podemos definir un objeto como una estructura abstracta que, de manera más fiable, describe un posible objeto del mundo real y su relación con el resto del mundo que lo rodea a través de interfaces.  Ejemplos de lenguajes de programación orientados a objetos serían **Java, Python o C#**.

La **programación orientada a objetos se sirve de diferentes conceptos** como:

–          Abstracción de datos

–          Encapsulación

–          Eventos

–          Modularidad

–          Herencia

–          Polimorfismo

*Programación reactiva*

Se basa en escuchar lo que emite un evento o cambios en el flujo de datos, en donde los objetos reaccionan a los valores que reciben de dicho cambio. Las librerías más conocidas son Project Reactor, y RxJava. React/Angular usan RxJs para hacer uso de la [programación reactiva](https://profile.es/blog/que-es-la-programacion-reactiva-una-introduccion/).

*Programación estructurada*

Constituye una **forma simplificada** de programación imperativa. La principal modificación del principio básico radica en que, en lugar de instrucciones de salto absolutas (instrucciones que provocan que el procesamiento no continúe con la siguiente instrucción, sino en otro lugar) este paradigma de programación de software prevé el **uso de bucles y estructuras de control**.

*Programación procedimental*

Amplía el enfoque imperativo con la posibilidad de desglosar **algoritmos en porciones manejables**. Estos se denominan como procedimientos, dependiendo del lenguaje de programación, o también como subprogramas, rutinas o funciones. El sentido y el propósito de esta distribución es hacer que el **código de programa sea más claro** y evitar las **repeticiones innecesarias de código**. Mediante la abstracción de los algoritmos, el paradigma de software procedimental representa un paso decisivo desde los lenguajes ensambladores simples hasta los lenguajes estándar complejos.

*Programación modular*

En principio, es muy similar al enfoque procedimental, o más bien lo adapta a los requerimientos de proyectos de software mayores y más amplios. En este sentido, el código fuente se divide específicamente en bloques parciales lógicos independientes los unos de los otros para proporcionar más **transparencia** y facilitar el **proceso de debugging** (resolución de errores). Los bloques parciales individuales, denominados **módulos**, se pueden probar por separado antes de vincularlos posteriormente a una aplicación conjunta.

*Programación funcional*

Las funciones existen en cualquier lenguaje de programación estándar. Sin embargo, los enfoques funcionales del desarrollo de software cuentan con funciones de una clase especial:

Un programa de programación funcional consta de llamadas de función concatenadas en las que cada parte del programa se interpreta como una función. En este sentido, las funciones dentro de la programación funcional pueden adoptar distintas “estructuras”

*Programación lógica*

Denominado también como programación predicativa, se basa en la lógica matemática. En lugar de una sucesión de instrucciones, un software programado según este principio contiene un **conjunto de principios** que se pueden entender como una recopilación de **hechos y suposiciones**. Todas las solicitudes al programa se procesan de forma que el intérprete recurre a estos principios y les aplica **reglas definidas previamente**para alcanzar el resultado deseado.

**Referencias**

* Cervantes, N. y Pineda, C. (s/f). Fundamentos de Programación. Recuperado el día 3 de octubre de 2020, en: <http://www.utn.edu.ec/reduca/programacion/fundamentos/un_poco_de_historia.html#:~:text=La%20historia%20de%20la%20Programaci%C3%B3n,primera%20gran%20influencia%20hacia%20la>
* Digital Guide IONOS. (2020). Paradigmas de la programación: principios básicos de la programación. Recuperado el día 3 de octubre de 2020, en: <https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/paradigmas-de-programacion/>
* Iribar, A. (s/f). *Historia de la computación*. Recuperado el día 3 de octubre del 2020, en: <http://paginaspersonales.deusto.es/airibar/Ed_digital/INF/Intro/Historia.html>
* Martínez, M. (2020). ¿Qué son los paradigmas de programación? Recuperado el día 3 de octubre de 2020, en: <https://profile.es/blog/que-son-los-paradigmas-de-programacion/>
* Muy Interesante. (2020) ¿Cuántos tipos de lenguaje de programación existen? Recuperado el día 3 de octubre de 2020, en: <https://www.muyinteresante.es/tecnologia/articulo/cuanto-tipos-de-lenguaje-de-programacion-existen-511584540297>
* Redactor Rock Content. (2019). ¿Qué es un lenguaje de programación y qué tipos existen? Recuperado el día 3 de octubre de 2020, en: <https://rockcontent.com/es/blog/que-es-un-lenguaje-de-programacion/>