Historia de los Lenguajes de Programación.

Con la idea de facilitarnos las tareas que debemos de desempeñar los humanos, hemos venido inventado diversas herramientas a lo largo de nuestra historia, que nos permiten tener una mejor calidad de vida.

Los ordenadores son uno más de los inventos del hombre, aunque debemos decir que las tecnologías para su fabricación y explotación han tenido un desarrollo sorprendente a partir de la segunda mitad del siglo XX. Esta herramienta por sí sola no es capaz de efectuar ninguna tarea, es tan sólo un conjunto de cables y circuitos que necesitan recibir instrucción por parte de los humanos para desempeñar alguna tarea. El problema entonces, se puede fijar en ¿cómo vamos a poder hacer que un conjunto de circuitos desempeñen una determinada tarea y nos entreguen los resultados que nosotros esperamos?, es decir, ¿de qué manera se puede lograr la comunicación entre el hombre y el ordenador?.

Así pues, tratando de dar una solución al problema planteado, surgieron los lenguajes de programación, que son como un lenguaje cualquiera, pero simplificado y con ciertas normas, para poder trasmitir nuestros deseos al ordenador.

Por otro lado, como se sabe, un conjunto de circuitos no entendería ningún lenguaje que nosotros conozcamos, por más sencillo que éste parezca. Los circuitos en todo caso, sólo reconocen presencia o ausencia de energía, es decir que debemos hablarle a la máquina en su propio lenguaje (presencia y ausencia de energía, 0 y 1), o nuestro lenguaje deberá de ser traducido a un lenguaje binario cuyo alfabeto es el 0 y el 1, mediante las herramientas desarrolladas para llevar a cabo esta tarea, las cuales reciben el nombre de traductores, y como veremos más adelante, los hay de muchos tipos, dependiendo de características más específicas del lenguaje a traducir y de la manera de llevar a cabo su traducción.

Como ya habréis entendido, para crear un lenguaje de programación, deberemos crear la herramienta que lo traduce, y es justamente de ellas, de las que hablaremos a continuación, para describir como han ido evolucionando en los últimos 50 años [BYTE 95].

- 1946: Konrad Zuse , un ingeniero Alemán mientras trabajaba en los Alpes de Bavaria, desarrolló el lenguaje Plankalkul, el cual, fue aplicado entre otras cosas para jugar al ajedrez.

- 1949: Aparece Short Code, que viene a ser el primer lenguaje que fue usado en un dispositivo de cómputo electrónico, aunque se debe decir que se trata de un lenguaje traducido a mano.

- 1951: Grace Hopper , trabajando para Remington Rand, comenzó el trabajo de diseño del primer compilador conocido ampliamente, el A-0, el cual, al ser liberado por la compañía en 1957, lo hizo con el nombre de MATH-MATIC.

- 1952: Alick E. Glennie, durante su tiempo libre en la Universidad de Manchester, concibe un sistema de programación llamado AUTOCODE, que viene a ser un compilador muy rudimentario.

- 1957: aparece FORTRAN (FORmula TRANslating) sistema traductor de fórmulas matemáticas. Fue desarrollado por un equipo, al frente del cual se encontraba John Backus quien después vendría a contribuir en el desarrollo del compilador para el lenguaje ALGOL y de la notación usada para la especificación sintáctica de los lenguajes, conocida como BNF (Backus Naur Form).

A partir de los años sesenta, empiezan a surgir diferentes lenguajes de programación, atendiendo a diversos enfoques, características y propósitos, que más adelante describiremos. Por lo pronto, puede decirse, que actualmente existen alrededor de 2000 lenguajes de programación [KINNERSLEY 95] y continuamente, están apareciendo otros más nuevos, que

prometen hacer mejor uso de los recursos computacionales y facilitar el trabajo de los programadores.

Tratando de resumir un poco, presentaremos los siguientes cuadros evolutivos, donde aparecen los lenguajes que por su uso y comercialización, han resultado ser los más populares a lo largo de este medio siglo. [LABRA 98] [RUS 01]

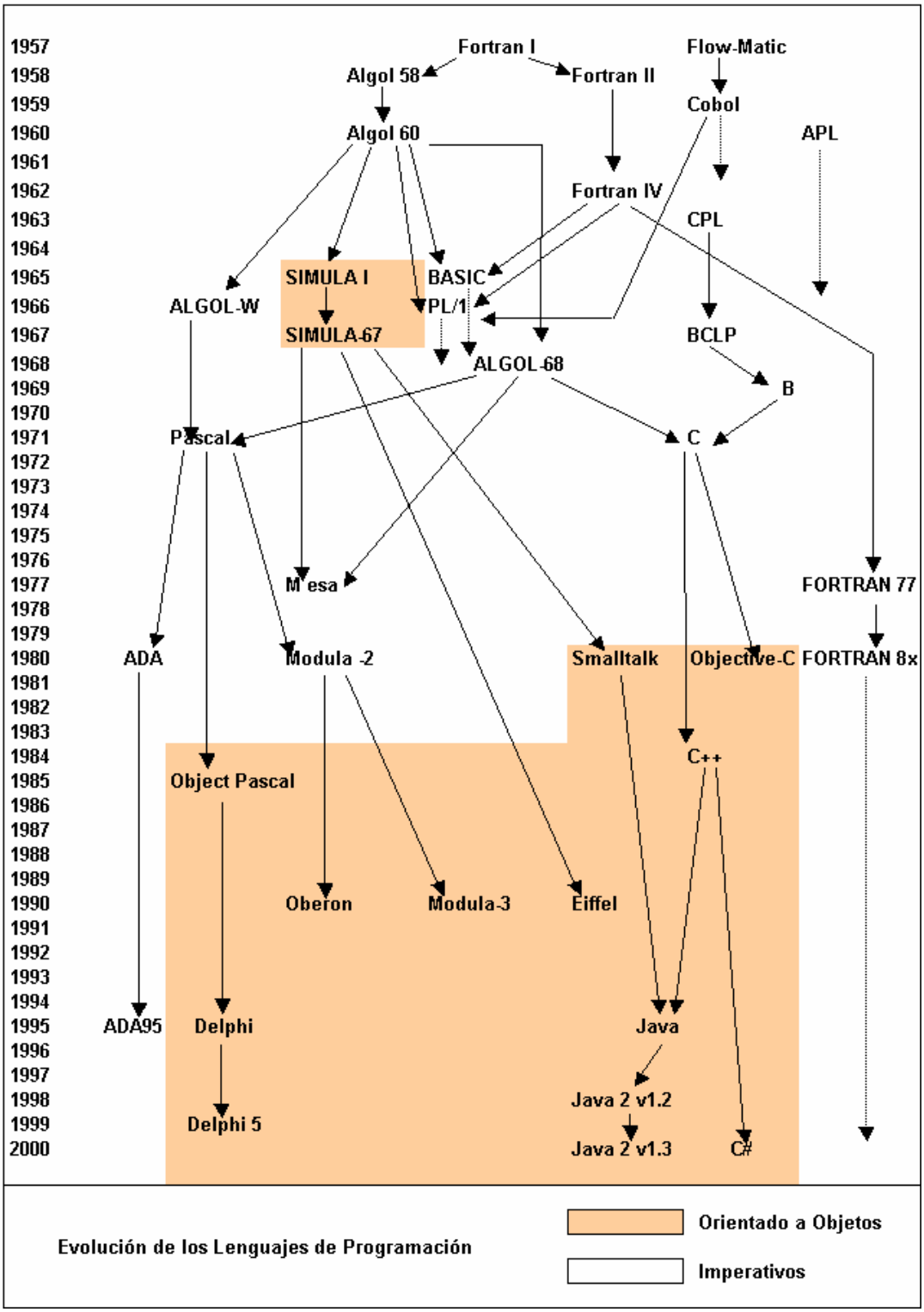

Figura a. Evolución de los Lenguajes Imperativos y Orientados a Objetos

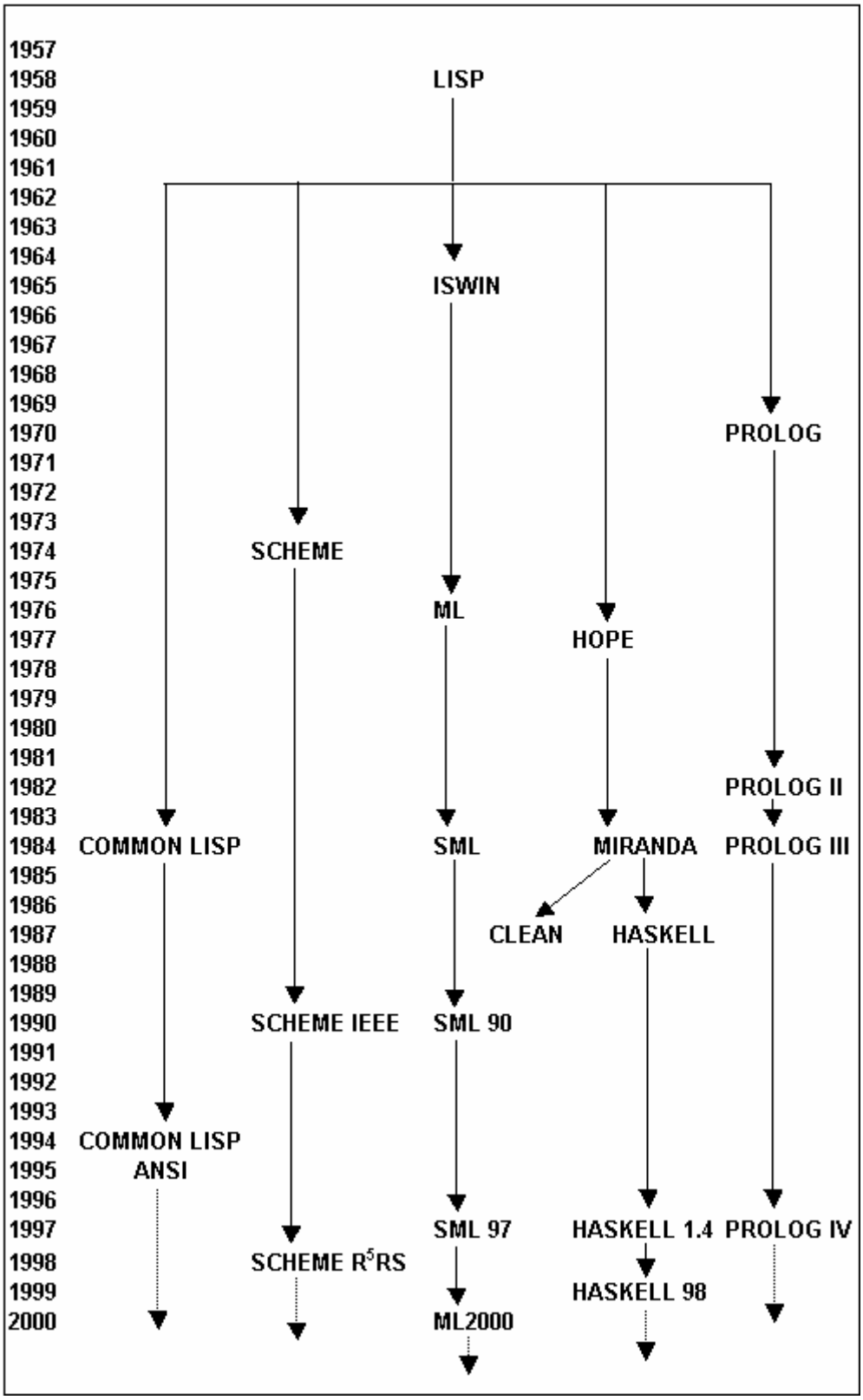

Figura b. Evolución de los lenguajes declarativos

Como ya lo citamos anteriormente y como se puede observar en las figuras a y b, la existencia de tantos lenguajes obedece a que cada uno de ellos está encaminado a resolver ciertas tareas, dentro de la amplia problemática de la explotación de la información, o bien, a que su arquitectura, o su forma de llevar a cabo la programación, tiene un enfoque particular.

De acuerdo con el estilo de programación, podemos clasificar los lenguajes en las siguientes categorías:

- **Imperativos**: Son aquellos lenguajes, que basan su funcionamiento en un conjunto de instrucciones secuenciales, las cuales, al ejecutarse, van alterando las regiones de memoria donde residen todos los valores de las variables involucradas en el problema que se plantea resolver. Es decir, se cambia progresivamente el estado del sistema, hasta alcanzar la solución del problema [CONTRERAS 01].
  Como un ejemplo ilustrativo vamos a escribir un programa en un lenguaje de este tipo para calcular el factorial de un número positivo x.

```
READ(x);
fac := 1 ;
for i = 1 to x
{
  fac := fac * i ;
}
WRITELN(fac);
```

- **Declarativos**: En este paradigma, más que el ¿cómo? desarrollar paso a paso un proceso, nos interesa el ¿qué? deseamos obtener a través del programa. Quizás el lenguaje declarativo que nos sea más familiar, es SQL, el cual es utilizado para interactuar con la información de bases de datos, concentrándose (como se podrá observar en el siguiente ejemplo), sólo en los resultados que van a ser obtenidos, dejándole al traductor la tarea de cómo llegar a ellos y presentárnoslos.[SANDERS-PRICE 02]

  SELECT * FROM alumnos WHERE sexo = "M" ORDER BY edad

  Dentro de este paradigma, se encuentran dos estilos distintos de programación, cada uno de los cuales posee su propia lógica [SANFÉLIX 00].

  - **Funcionales**: Son lenguajes basados en funciones, las cuales se representan mediante expresiones, que nos permiten obtener ciertos resultados a partir de una serie de argumentos**[BIBBY 00].** De hecho las expresiones están formadas por un conjunto de términos, que a su vez pueden encapsular otras expresiones, para con la evaluación de todas ellas, llegar a la solución deseada.[GAULD 01]. Para describir la idea, retomaremos el ejemplo del factorial escrito en el lenguaje funcional Haskell.

```
fac :: Integer -> Integer
fac 0 = 1
fac x = x * fac (x-1)
```

  - **Lógicos**: Este tipo de lenguajes se basan en el cálculo de predicados, la cual es una teoría matemática que permite entre otras cosas, lograr que un ordenador basándose en un conjunto de hechos y de reglas lógicas, pueda derivar en soluciones inteligentes. [DIMARE 90]. El mismo ejemplo del factorial, se vería de la siguiente manera, escrito en PROLOG.

```
factorial (0, 1)
factorial (X, Fac) :- Y is X-1,   fac(Y, F2),  Fac is F2 * X .
```

- **Orientados a Objetos**: Este último paradigma, como se puede observar en la figura 1, algunas veces se mezcla con alguno de los otros 2 modelos, sin embargo mantiene características propias, que lo diferencian claramente. Los programas de este tipo, se concentran en los objetos que van a manipular, y no en la lógica requerida para manipularlos [MARBUS 00]. Ejemplos de objetos pueden ser: estudiantes, coches, casas etc, cada uno de los cuales tendrá ciertas funciones (métodos) y ciertos valores que los identifican, teniendo además, la facultad de comunicarse entre ellos a través

del paso de mensajes. Cabe mencionar con más detalle los elementos fundamentales que deben de poseer este tipo de lenguajes [BOOCH 96]:

- **Abstracción**: Determinación de las características de los objetos, que sirven para identificarlos y hacerlos diferentes a los demás.
- **Encapsulamiento**: Es el proceso que agrupa y almacena los elementos que definen la estructura y el comportamiento de una abstracción, en un mismo lugar.
- **Modularidad**: Es la propiedad de agrupar las abstracciones que guardan cierta relación lógica, y a la vez minimizar la interdependencia entre las diversas agrupaciones.
- **Jerarquía**: Consiste en establecer un orden o una clasificación de las abstracciones.

Además de estos elementos fundamentales, también existen otros 3 elementos secundarios , que aunque son deseados, no son indispensables para clasificar un lenguaje dentro de este estilo.

- **Tipificación**: Mecanismo que intenta restringir el intercambio entre abstracciones que poseen diversas características.
- **Persistencia**: Es la propiedad de un objeto a continuar existiendo a través del tiempo y/o del espacio.
- **Concurrencia**: Es la propiedad que distingue a los objetos activos, de los que no lo están.

Ahora bien, si tomamos como referencia las herramientas usadas en el proceso de traducción y ejecución de los programas esbozada en la figura 2, vamos a tener la siguiente clasificación de lenguajes[AHO 77]:

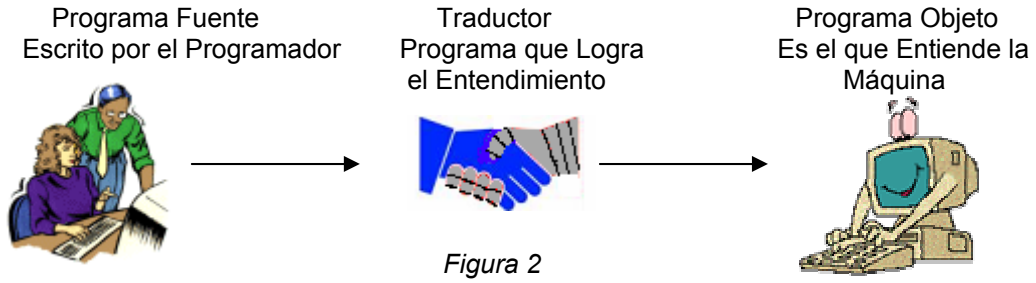

Figura 2

- **Lenguajes Ensamblados**: Se refieren al lenguaje ensamblador, que viene a ser una representación simbólica de las instrucciones correspondientes al lenguaje ensamblafor de alguna arquitectura específica, con lo que, casi siempre, la correspondencia entre las instrucciones de este lenguaje, y las del lenguaje máquina son de 1 a 1, si bien existen algunas excepciones, que dan lugar a lo que se conoce como lenguajes macro-ensambladores [CUEVA 88]
- **Lenguajes Compilados**: Son aquellos, que son traducidos de un lenguaje de alto nivel (como FORTRAN o PASCAL) a lenguaje máquina o bien a lenguaje ensamblador, produciendo un programa objeto permanente.
- **Lenguajes Interpretados**: Estos lenguajes, tienen la particularidad, de que no producen código objeto, sino que cada instrucción es analizada y ejecutada a la vez, lo que ofrece mucha interacción con los usuarios, pero a la vez resultan ineficientes, cuando se desea ejecutar repetitivamente un programa.

- **Lenguajes Preprocesados**: Son lenguajes que son traducidos primeramente a un lenguaje intermedio de más bajo nivel, para posteriormente volverlos a traducir y producir el programa objeto. Este tipo de lenguajes fueron creados, con la idea de proporcionar un lenguaje más potente que el lenguaje intermedio, mediante la implementación de algunas macroinstrucciones. [SANCHIS-GALAN 86].

Finalmente, existen otros conceptos tomados en cuenta para agrupar los lenguajes, que dan origen a diversas clasificaciones, entre los que destacan las siguientes:

- **Lenguajes de cuarta generación 4GL**: Estos lenguajes se distinguen por formar parte de un entorno de desarrollo, que comprende el manejador de una base de datos, y todo lo que de esto se deriva, como la administración de un diccionario de datos, el control de accesos, el manejo de la consistencia de la información y otras características enfocadas a facilitar los programas de acceso y explotación de la información. Como ejemplos podemos citar a los 4 grandes: PROGRESS, SYSBASE, INFORMIX, y ORACLE.

- **Lenguajes Visuales**. Se les llama de esta manera a los lenguajes que forman parte de una aplicación dotada de una Interfase gráfica, la cual por medio de iconos y otras herramientas visuales y simbólicas, pretenden facilitar las tareas rutinarias de los programadores, como son el diseño y desarrollo de formularios e informes. Los ejemplos más comerciales de estos lenguajes son: VISUAL BASIC, VISUAL CAFE, VISUAL FOX, etc.

- **Metalenguajes**: Son lenguajes como XML, SGML y HTML que sirven para definir otros lenguajes, cuyo objetivo es llevar a cabo la estructuración de textos mediante un conjunto de etiquetas, de manera tal, que puedan ser entendidos por los humanos y también procesado por los ordenadores. Estos lenguajes están teniendo un gran auge sobre la plataforma de Internet, en la cual son usados para la creación de documentos, y el intercambio o transferencia de información.

- **Lenguajes de propósito específico:** Son aquellos lenguajes desarrollados con la finalidad de resolver problemas de una naturaleza muy determinada, tal como SPSS para problemas estadísticos, MATLAB para cálculos científicos y de ingeniería, CAD/CAM para el diseño de piezas y programación de máquinas de control numérico, como tornos y fresadoras, GPSS para simulación de sistemas, CORBA para el manejo de interfaces en ambientes cliente-servidor, etc.

- **Lenguajes Script:** Son lenguajes como JAVASCRIPT, VBSCRIPT, PERLSCRIPT, que se utilizan en ambientes clientes servidor, mediante la incrustación de código en las páginas HTML, y así permitir la programación del lado del cliente, buscando fundamentalmente, hacer más atractivos los interfaces gráficos de las páginas [BUITRAGO 00].

Esta gran cantidad de lenguajes, señala de manera clara que existe un esfuerzo continuo en la creación, y mejora de los lenguajes de programación, en aras, de hacer más fácil la tarea del programador y/o hacer un uso más eficiente de los recursos computacionales.

La búsqueda de los objetivos antes mencionados, así como la guerra mercantil de las compañías dedicadas a la producción de herramientas de software, han diversificado las opciones que los programadores pueden elegir. Sin embargo , hasta nuestros días, podemos decir que realmente no existe ningún lenguaje, o grupo de ellos, que destaque en la totalidad de las aplicaciones informáticas que se desarrollan actualmente, ya que cada uno, tiene cualidades que lo hacen más convenientes para algunos propósitos, pero al mismo tiempo, cuentan con inconvenientes para otros.

Lo anterior, puede resumirse en la siguiente pregunta y respuesta, citada en inglés para no alterar su sentido.

- *What is the BEST programming language?*
    - *To my knowledge there is no programming language called BEST*