Historia y evolución de los lenguajes de programación

Moisés García Villanueva

22 de Agosto de 2012

1 Línea de tiempo y el desarrollo de los lenguajes de programación

Primero daré estas definiciones de lo que son los lenguajes de programación:

- Los lenguajes de programación son notaciones, un sistema de símbolos para un uso especializado, diferente al de la escritura ordinaria. Son utilizados para especificar, organizar y razonar en relación a procesos de computo [2].
- Un lenguaje de programación es un idioma artificial diseñado para expresar procesos que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana [1].

2 Familias de lenguajes de programación

Las principales familias de lenguajes de programación son:

- Imperativa, procedural
- Orientada a objetos
- Funcional
- Lógica

En adición a los lenguajes de alto nivel, debemos considerar también el lenguaje de máquina y el ensamblador.

3 Niveles de abstracción

La abstracción consiste en aislar un elemento de su contexto o del resto de los elementos que lo acompañan. En programación, el término se refiere al énfasis en el "¿qué hace?" más que en el "¿cómo lo hace?" (característica de caja negra). El común denominador en la evolución de los lenguajes de programación, desde los clásicos o imperativos hasta los orientados a objetos, ha sido el nivel de abstracción del que cada uno de ellos hace uso.

Los lenguajes de programación son las herramientas mediante las cuales los diseñadores de lenguajes pueden implementar los modelos abstractos. La abstracción ofrecida por los lenguajes de programación se puede dividir en dos categorías: abstracción de datos (pertenecientes a los datos) y abstracción de control (perteneciente a las estructuras de control).

Los diferentes paradigmas de programación han aumentado su nivel de abstracción, comenzando desde los lenguajes de máquina, lo más próximo al ordenador y más lejano a la comprensión humana; pasando por los lenguajes de comandos, los imperativos, la orientación a objetos (POO), la Programación Orientada a Aspectos (POA); u otros paradigmas como la programación declarativa, etc.

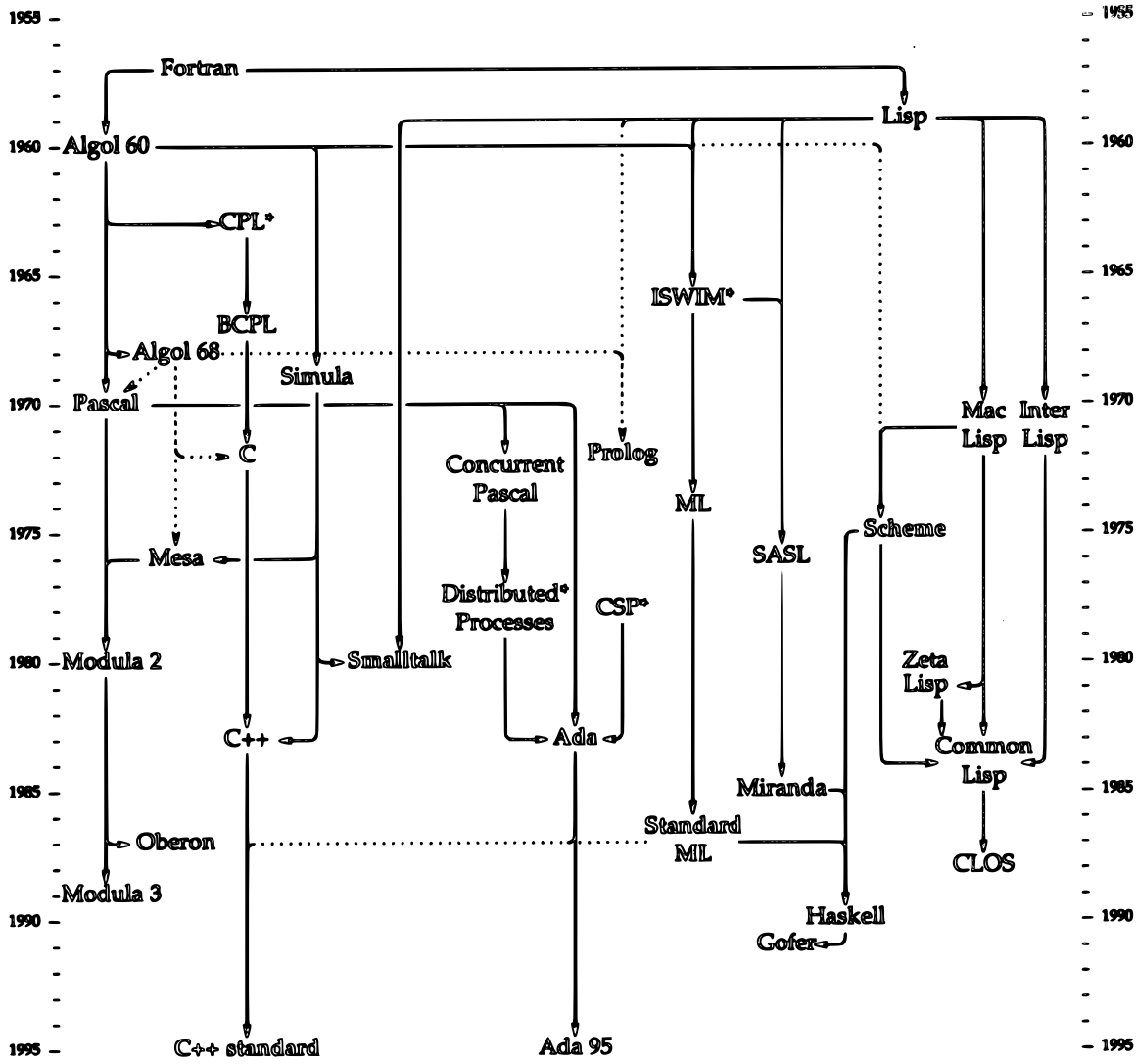

Figure 1: Lenguajes de Programación influenciados por otros lenguajes. Un (*) al lado de un lenguaje indica que nunca fue completamente implementado.

La abstracción encarada desde el punto de vista de la programación orientada a objetos expresa las características esenciales de un objeto, las cuales distinguen al objeto de los demás. Además de distinguir entre los objetos provee límites conceptuales. Entonces se puede decir que la encapsulación separa las características esenciales de las no esenciales dentro de un objeto. Si un objeto tiene más características de las necesarias los mismos resultarán difíciles de usar, modificar, construir y comprender.

La figura 2 nos presenta el nivel de abstracción de algunos lenguajes de programación.

4 Paradigmas de programación

El esfuerzo necesario para introducir un nuevo lenguaje es substancial, no solamente en el diseño e implementación, sino a demás en enseñarlo y proporcionarle el soporte adecuado.

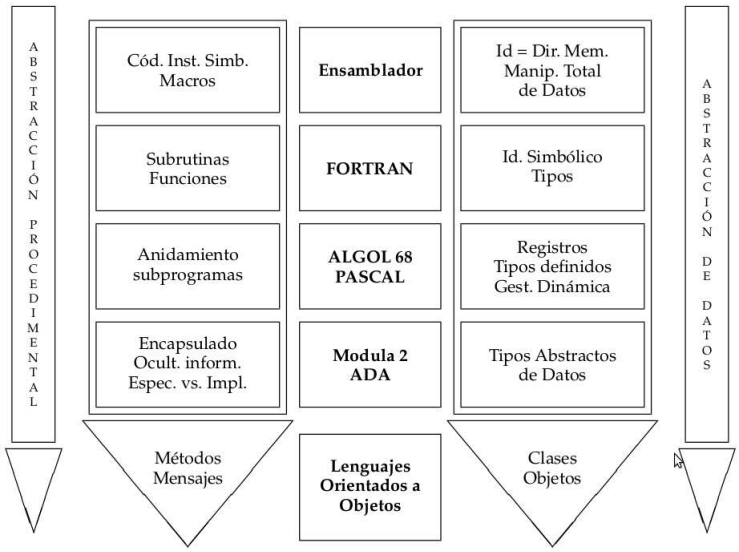

Figure 2: Nivel de abstracción de los Lenguajes de Programación

4.1 Programación imperativa

Los lenguajes imperativos son acciones orientadas, es decir, un proceso de computación es visualizado como una secuencia de acciones. Pascal y C cubren las ideas clave de la programación imperativa. Algunos lenguajes en esta familia son: Fortran, BCPL, C y la familia de Algol: Algol-60, Pascal, Modula-2, Ada .

4.2 Programación funcional

Los conceptos básicos de lenguajes funcionales están originados en Lisp, un lenguaje diseñado en 1958 para aplicaciones en Inteligencia Artificial.

Algunos lenguajes en esta familia son: Miranda, ML, Haskell y Lisp.

4.3 Programación orientada a objetos

C++ y Smalltalk son lenguajes populares en la programación orientada a objetos. Este tipo de programación tiene mucho de sus orígenes en el lenguaje Simula, utilizado en simulación. Este lenguaje fue diseñado como lenguaje de programación y además un descriptor del lenguaje, en el periodo de 1961 y 1967 por Kristen Nygaard y Ole-Johan Dahl. Simula cambio la forma en como pensaba la gente en relación a la programación.

El concepto clave de Simula es lo que se denomina una clase de objetos. La clasificación de objetos en clases y subclases es el tópico central para la programación orientada a objetos. Los lenguajes en esta familia son: Simula, Smalltalk-72, Smalltalk-76, Smalltalk-80, C++, Objective C, Common Lisp Object System (CLOS), Java y C#.

4.4 Programación lógica

Prolog fue desarrollado en 1972 para el procesamiento natural del lenguaje en Francia por Alain Colmerauer y Philippe Roussel. El primer programa de Prolog de tamaño considerable fue para la interacción máquina-humano.

Los lenguajes en esta familia son: Prolog, Concurrent Prolog y CLP(R).

5 Lenguajes de programación e ingeniería de software

La investigación en los grupos de lenguajes de programación incluyen trabajos en particular sobre lenguajes de programación, aspectos teóricos, tales como tipos y semántica, análisis de programa y verificación de software,

paradigmas de programación, aspectos de concurrencia, paralelismo y programación distribuida. Existe además un traslape con la ingeniería de software y los métodos formales.

5.1 Ingeniería de software

La ingeniería de software se refiere al problema de como producir software de alta calidad a partir de los aspectos de diseño para la implementación y subsecuente mantenimiento del software. En Ingeniería de Software se denomina "ciclo de vida" a una determinada organización en el tiempo de las actividades de desarrollo de software. Las principales actividades son enumeradas en la figura 3.

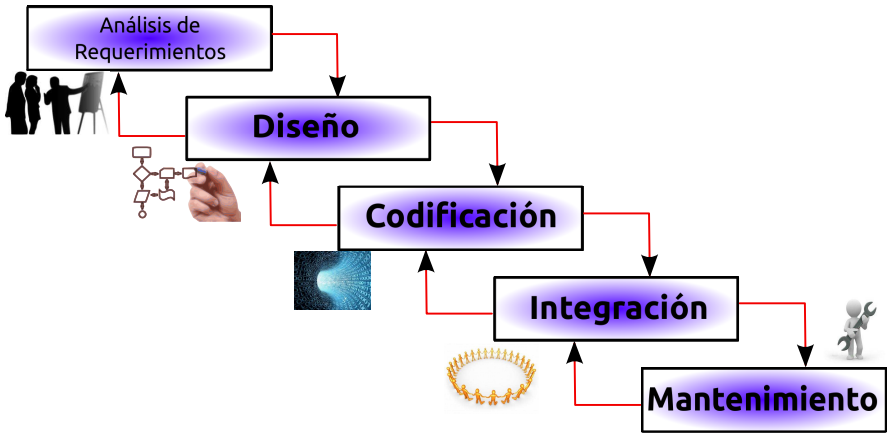

Figure 3: Actividades de desarrollo de software

La figura 3 representa el denominado "ciclo de vida en cascada", donde las flechas indican el orden en que se van realizando las actividades.

- Análisis de requisitos: Se estudian las necesidades de los usuarios, se decide qué debe hacer la aplicación informática para satisfacerlas en todo o en parte, y se genera un Documento de Requisitos.
- Diseño de la arquitectura: Se estudia el Documento de Requisitos y se establece la estructura global de la aplicación, descomponiéndola en partes (módulos, subsistemas) relativamente independientes. Se genera un Documento de Diseño.
- Diseño detallado: En esta segunda parte de la actividad de diseño se fijan las funciones de cada módulo, con el detalle de su interfaz. Se genera el código de declaración (o especificación) de cada módulo.
- Codificación: Se desarrolla el código de cada módulo.
- Integración.

  Pruebas de unidades: Como complemento de la codificación, cada módulo o grupo de módulos se prueba por separado. En las pruebas se comprueba si cada módulo cumple con su especificación de diseño detallado.

  Pruebas de integración: Se hace funcionar la aplicación completa, combinando todos sus módulos. Se realizan ensayos para comprobar que el funcionamiento de conjunto cumple lo establecido en el documento de diseño.

  Pruebas de validación: Como paso final de la integración se realizan nuevas pruebas de la aplicación en su conjunto. En este caso el objetivo es comprobar que el producto desarrollado cumple con lo establecido en el documento de requisitos, y satisface por tanto las necesidades de los usuarios en la medida prevista.

- Fase de mantenimiento No hay actividades diferenciadas de las anteriores. El mantenimiento del producto exige rehacer parte del trabajo inicial, que puede corresponder a cualquiera de las actividades de las etapas anteriores.

6 Ejercicios

1. Hacer un diagrama de flujo para un programa que solicite un conjunto de números enteros y los imprima sin repetición. Ejemplo de entrada: 120 643 201 120 11 201 921; La salida producida será: 120 643 201 11 921

2. Escribir con sus palabras un programa que imprima los números leídos en el ejercicio anterior en forma ascendente.

3. Indica en que familia podemos ubicar los siguientes lenguajes:

   - javascript
   - shell
   - Ocaml

Bibliografía

[1] M. Lutz. *Learning Python: Powerful Object-Oriented Programming.* Learning Python. O'Reilly Media, 2009.

[2] R. Sethi. *Programming Language With Java Package.* Addison Wesley, 2002.