UNIVERSIDAD AUTONOMA DE COAHUILA
FACULTAD DE CIENCIAS FÍSICO MATEMÁTICAS

Lenguajes de programación

Luis Eduardo Sánchez González

24 de enero de 2021

1. ¿QUÉ ES LA PROGRAMACIÓN?

Una de las partes que compone la física computacional es la computación. Hasta ahora sabemos que las computadoras pueden realizar cálculos y tomar decisiones lógicas con una rapidez increíblemente mayor que los humanos, por esta razón en física las computadora son de gran ayuda. Las computadoras procesan datos bajo el control de conjuntos de instrucciones conocidas como **programas de computadora** y a la elaboración de estos programas de computadora se le conoce como **programación**. Estos programas guían a la computadora a tráves de conjuntos ordenados de acciones especificadas por gente conocidad como programadores de computadoras. A los programas que se ejecutan en una computadora se les denomina **software**.

En este punto se debe hacer enfasís que no se debe confundir la programación con la computación. La diferencia que existe entre estos dos términos es tal vez la misma que existe entre saber la fórmula para resolver una ecuación de segundo grado y conocer la teoría de ecuaciones. [1]

2. LOS LENGUAJES DE PROGRAMACIÓN

Para crear un programa, y que la computadora lo interprete y ejecute, las instrucciones deben escribirse en un lenguaje de programación.

En sus inicios las computadoras interpretaban solo instrucciones en un lenguaje específico, del más bajo nivel, conocido como **lenguaje máquina**. El lenguaje máquina consiste en cadenas de números (0s y 1s). Dichos lenguajes son difíciles de comprender para los humanos, por esa razón para facilitar el trabajo de programación se empezaron a utilizar abreviaturas del inglés para representar operaciones elementales. Estas abreviaturas formaron la base de los **lenguajes ensambladores**. Se desarrollaron programas traductores conocidos como ensambladores para convertir los programas en lenguaje ensamblador a lenguaje máquina, debido que este es el lenguaje que pueden entender las computadoras.

A medida que la complejidad de las tareas que realizaban las computadoras aumentaba, se hizo necesario disponer de un método más sencillo para programar. Se crearon los lenguajes de alto nivel, en donde podía escribirse instrucciones individuales para realizar tareas importantes. Los lenguajes de alto nivel nos permiten escribir instrucciones que son muy similares al inglés y contienen expresiones matemáticas de uso común. Los programas traductores llamados compiladores convierten los programas que se encuentran en lenguaje de alto nivel a programas en lenguaje máquina.

Toda la historia de los lenguajes de programación se ha desarrollado en base a una sola idea conductora: hacer que la tarea de realizar programas para ordenadores sea cada vez lo más simple, flexible y portable posible. [2]

2.1. Lenguajes de programación de alto nivel

Como hemos visto, los lenguajes de alto nivel se caracteriza por expresar los algoritmos de una manera adecuada a la capacidad cognitiva humana, en lugar de la capacidad con que los ejecutan las máquinas. El primer lenguaje de uso generalizado orientado a esto fue **FORTRAN** (*Formula Translator*), alrededor de 1956. Pocos años después se desarrolló un lenguaje para usos comerciales, donde lo que se deseaba es poder manejar datos a distintos niveles de agregación. Este lenguaje se llamaba **COBOL** (*COmon Bussiness Oriented Language)*. Ambos lenguajes, o versiones modernizadas, sobreviven hasta nuestros días. Hacia finales de los años 50 se diseño un lenguaje, **ALGOL** (*ALGorithmic Oriented Language*) que resultó ser el modelo de desarrollo de prácticamente todos los lenguajes orientados a algoritmos de hoy en día, como **Pascal, C, C++, Java** y muchos más.

Apesar que estos últimos lenguajes pueden ser más utilizados que los primeros, además que los primeros pueden ser considerados lenguajes ”viejos”, no se consideran obsoletos. En particular FORTRAN sigue siendo muy utilizado en la física, debido que está especialmente adaptado al cálculo numérico y a la computación científica.

El proceso de compilación de un programa escrito en lenguaje de alto nivel a un lenguaje máquina puede tardar un tiempo considerable en la computadora. Los **programas intérpretes** se desarrollaron para ejecutar programas en lenguaje de alto nivel de manera directa, esta modalidad de trabajo es equivalente pero se realiza instrucción por instrucción, es decir, se traduce a medida que es ejecutado el programa, esto ocasiona que la ejecución del programa sea más lenta en comparación a los programas compilados. Los lenguajes de secuencias de comandos, como los populares lenguajes **Python, JavaScript** y **PHP**, son procesados por intérpretes.

El lenguaje de programación Python es el más llamativo para nuestros propositos en computación científica. Según el índice de la comundidad de programación *TIOBE*, este lenguaje de programación se encuentra entre los primeros tres más usados, los otros puesto los ocupa Java (2°) y C (1°) [1].

2.2. Características de algunos lenguajes de alto nivel

Dependiendo del tipo de problema que queramos resolver (numérico, administrativo, de propósito general, inteligencia artificial, lógico) tenemos distintos lenguajes de programación que permiten representar de mejor manera los formatos de los datos y los recursos que requerimos para resolver el problema. Así, para procesos numéricos se requiere de bibliotecas muy extensas con funciones

[1] El índice TIOBE puede consultarse en: https://www.tiobe.com/tiobe-index/

matemáticas, un manejo sencillo de matrices y, en general, de espacios de varias dimensiones, etc. A continuación se presenta una tabla comparativa con los lenguajes de programación que esten más orientados a la computación científica.

Tabla 2.1: Tabla comparativa de lenguajes de alto nivel

Lenguaje	Características	Ventajas	Desventajas
FORTRAN	Compilado POO	Rápido Sintaxis simple Tipado fuerte Sencillo de aprender	Uso único en campos científicos Pocas librerias Lenguaje primitivo
C++	Compilado POO	Ejecución rápida Tipado fuerte	Sintaxis compleja Compilación lenta Dificil de aprender
Python	Interpretado Multiparadigma	Sintaxis simple Muchas librerias Sencillo de aprender	Lento Tipado dinámico

Actualmente se encuentran proyectos en desarrollo como **Cython**, un lenguaje de programación que su objetivo es tomar lo mejor de Python y C++ para combinarlo en un mismo lenguaje de programación, es tener Python a la velocidad de C. También el proyecto **F2PY** que puede se utilizado para implementar subrutinas de FORTRAN en Python.

3. PARADIGMAS DE PROGRAMACIÓN

Un paradigma de programación consiste en la forma en la que deben estructurarse y organizarse las tareas que debe realizar un programa. Representa un enfoque particular o filosofía para diseñar soluciones. Los paradigmas difieren unos de otros, en los conceptos y la forma de abstraer los elementos involucrados en un problema, así como en los pasos que integran su solución del problema. El paradigma de programación que actualmente es más utilizado es la ”orientación a objetos” (POO).

3.1. Programación orientada a objetos (POO)

La POO son un conjunto de técnicas que nos permiten incrementar enormemente nuestro proceso de producción de software. Usando estas técnicas, nos aseguramos la reusabilidad de nuestro código, es decir, los objetos que hoy escribimos, si están bien escritos, nos servirán para "siempre".

El núcleo central de este paradigma es la unión de datos y procesamiento en una entidad llamada ”objeto”. Los objetos son entidades que tienen un determinado ”estado”, ”comportamiento” e ”identidad”.

3.2. Características de la POO

Existe un acuerdo acerca de qué características contempla la ”orientación a objetos”. Las características siguientes son las más importantes:

Abstracción: Denota las características esenciales de un objeto, donde se capturan sus comportamientos. Cada objeto en el sistema sirve como modelo de un ”agente” abstracto que puede realizar

trabajo, informar y cambiar su estado, y ”comunicarse” con otros objetos en el sistema sin revelar ”cómo” se implementan estas características.

Encapsulación: Significa reunir todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. En pocas palabras, el encapsulamiento es restringir ”acceso” las propiedades y métodos de un objeto.

Herencia: Esta es la cualidad más importante de un la POO, la que nos dará mayor potencia y productividad, permitiéndonos ahorrar horas y horas de codificación y de depuración de errores. Se basa en lo que conocemos como herencia biológica, es decir que una clase puede heredar de otra métodos y atributos.

Polimorfismo: Por polimorfismo entendemos aquella cualidad que poseen los objetos para responder de distinto modo ante el mismo mensaje.

4. FÍSICA EN POO

El presente apartado es parte de una investigación que sigue en desarrollo y que estoy realizando por mi cuenta. Consta de una propuesta para la enseñanza de la programación orientada a objetos mediante física computacional. La teoría acerca de programación orientada a objetos puede ser bastante abstracta y dificil de comprender cuando se les presenta a los alumnos por primera vez. Es por esto que se buscan métodos para su compresión y utilización, de esta manera poder potencializar una de las características más importantes: *la reutilización de código*.

La programación orientada a objetos como hemos observado consiste en ”traer” los objetos reales de la vida cotidiana a la programación. En este contexto y utilizando la física podemos ver a los objetos físicos (resorte, péndulo, proyectil) como objetos de programación, que pertenecen a una clase (*sistema mecánico*). Pero además de eso, no solamente existe una única clase, sino multiples: sistema eléctrico, sistema óptico, sistema termodinámico, sistema cuántico, etc. De tal manera que estos objetos de distintas clases cuenta con ciertos atributos y métodos, que pueden estar encapsulados o no.

Por ejemplo, un objeto que pertenece a la clase *sistema mecánico*, como un péndulo, cuenta con los atributos de posición, velocidad, masa, etc. este último puede ser un atributo encapsulado, debido que es una propiedad intrínseca y que no se debe de cambiar fuera de la clase. Además cuenta con un método el cual es la ley física que describe su evolución en el tiempo. Quizás esta ley física necesitará resolverse por ciertos algortimos, esto implica que la clase *sistema mecánico* debe heredar de otra clase los métodos necesarios para resolver la ley física. Tomando el ejemplo del péndulo se puede considerar que la clase que debe heredar es la clase *Solve EDO's* y de igual manera esta clase puede contar con atributos (condiciones iniciales) y con métodos (Euler, Runge Kutta o Verlet).

Otro ejemplo a considerarse es un circuito eléctrico, este objeto pertenece a la clase *sistema eléctrico*, además dependiendo del tipo de circuito puede contar con atributos como resistencia, capacitancia, inductancia, etc. Al igual que un sistemas mecánico debería heredar la clase *Solve EDO's*.

Como mencione anteriormente, este planteamiento teórico y la implementación en distintos lenguajes de programación como Python, C++, Cython, Java y FORTRAN, aun sigue en desarrollo.

REFERENCIAS

[1] Elisa Viso G. Canek Peláez V. Introducción a ciencias de la computación I. *Facultad de Ciencias, UNAM*

[2] Francisco Morero (1999). Introducción a la OOP. *Grupo EIDOS, ejemplar gratuito.*

[3] Deitel, Paul y Harvey Deitel (2014.) Cómo programar en C++, 9na Ed. *Pearson educación*