Capítulo 1

PRINCIPIOS BÁSICOS DE LA PROGRAMACIÓN

En este capítulo veremos en qué consiste la tarea de programar, qué herramientas se necesitan para llevarla a cabo, qué herramientas hay disponibles y cuáles conviene elegir según el tipo de programación que se elija. Además, construiremos un pequeño programa para contar las palabras de una frase, el cual nos servirá para tener nuestra primera experiencia con el lenguaje Pascal.

LENGUAJES DE PROGRAMACIÓN: ¿POR QUÉ HAY TANTOS?

(PARADIGMAS, LENGUAJES Y FORMAS DE PROGRAMAR)

En párrafos anteriores vimos que los programas se escriben en lenguajes que pueden traducirse a una forma que la computadora pueda entender (como veremos más adelante, a esta forma se la llama **lenguaje de máquina**). Esto hizo que, a lo largo de la historia de la programación, se fueran creando distintos lenguajes para distintas necesidades, cada uno con su correspondiente "traductor" a lenguaje de máquina. Por lo general, los lenguajes de programación surgieron de centros de investigación en universidades o empresas, cada uno con el objetivo de cubrir alguna necesidad en particular.

Así nacieron, por ejemplo, el **FORTRAN** (por *Formula Translator*, o traductor de fórmulas), pensado especialmente para que los científicos de distintas disciplinas pudieran escribir programas para hacer cálculos de gran complejidad, y el **COBOL** (por *Common Business Oriented Language* o lenguaje común orientado a negocios), que fue pensado con el objetivo de escribir programas para administrar empresas.

LOS DISTINTOS PARADIGMAS Y SUS LENGUAJES

Con el tiempo, los investigadores en ciencias de la computación observaron que, más allá del propósito para el que fueron creados, los lenguajes podían diferenciarse por la forma de trabajo que presentan al programador, ofreciendo diversas formas de "ver" y "pensar" un programa antes de escribirlo.

Así comenzaron a surgir distintos **paradigmas de programación**, cada uno representado por una familia de lenguajes (**Figura 1**). Por ejemplo, a la forma tradicional de programar –que consiste en detallar la secuencia de pasos que debe seguir la computadora para realizar una tarea– se la denominó **programación imperativa**, porque el programa da órdenes que la computadora debe cumplir obedientemente.

LOS LENGUAJES .NET

El entorno .NET de Microsoft presenta un marco de trabajo (llamado Framework) en el que se pueden utilizar lenguajes muy diversos. Todos ellos tienen la particularidad de ser orientados a objetos y de compartir una misma interfaz mediante la cual interactúan con el sistema operativo.

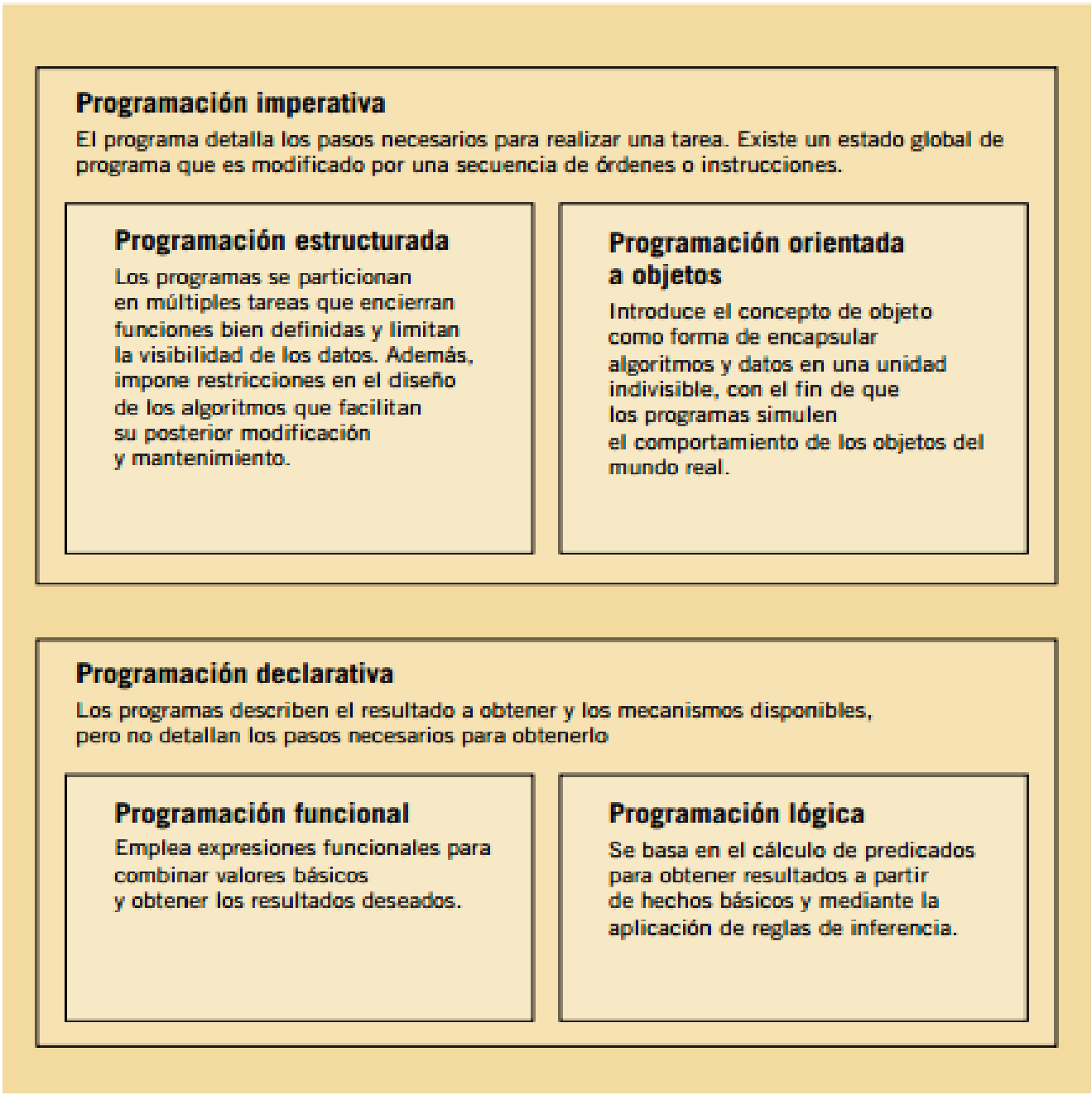

Figura 1. *Los lenguajes más comunes –C, Pascal, Basic, FORTRAN, etc.– pertenecen al paradigma de programación imperativa; los dos primeros enfatizan el concepto de programación estructurada.*

Una rama dentro de la programación imperativa es la **programación estructurada**, cuyo propósito fundamental es construir programas claros, fáciles de entender y de mantener. Para ello se basa en el uso de módulos independientes (funciones y procedimientos) que separan claramente las diferentes tareas que

realiza un programa y en estructuras que conducen claramente el flujo de ejecución (más sobre esto en el **Capítulo 4**). Algunos lenguajes que enfatizan el concepto de programación estructurada son el C y el Pascal.

En contraposición a la programación imperativa se encuentra el paradigma de **programación declarativa**, que en lugar de dar una secuencia detallada de órdenes para que la computadora lleve a cabo, simplemente enuncia el resultado que se desea obtener y se deja que la computadora elija la forma más conveniente de obtenerlo. Este paradigma no es tan común como el de programación imperativa y se usa para aplicaciones que involucran inteligencia artificial, simulaciones científicas, etc. Dentro del paradigma de programación declarativa se encuentran los lenguajes de **programación lógica**, como el ProLog y el Lisp, los lenguajes de **programación funcional**, como Haskell, y otros lenguajes más específicos, como el SQL (*Structured Query Language*), cuyo propósito es el armado de consultas de bases de datos.

Otro paradigma importante derivado de la programación imperativa es el de la **programación orientada a objetos**. De gran popularidad en nuestros días, este paradigma permite pensar una aplicación como un conjunto de objetos autónomos que interactúan entre sí enviándose mensajes, con los cuales un objeto le "pide" a otro que realice una determinada tarea o le solicita determinados datos (**Figura 2**). La ventaja de esta forma de programación es que el desarrollo de las funciones está **encapsulado** dentro de cada objeto. En otras palabras, a un objeto no le preocupa cómo hace otro objeto para cumplir su función; le basta saber que, cuando se lo pida, hará su trabajo.

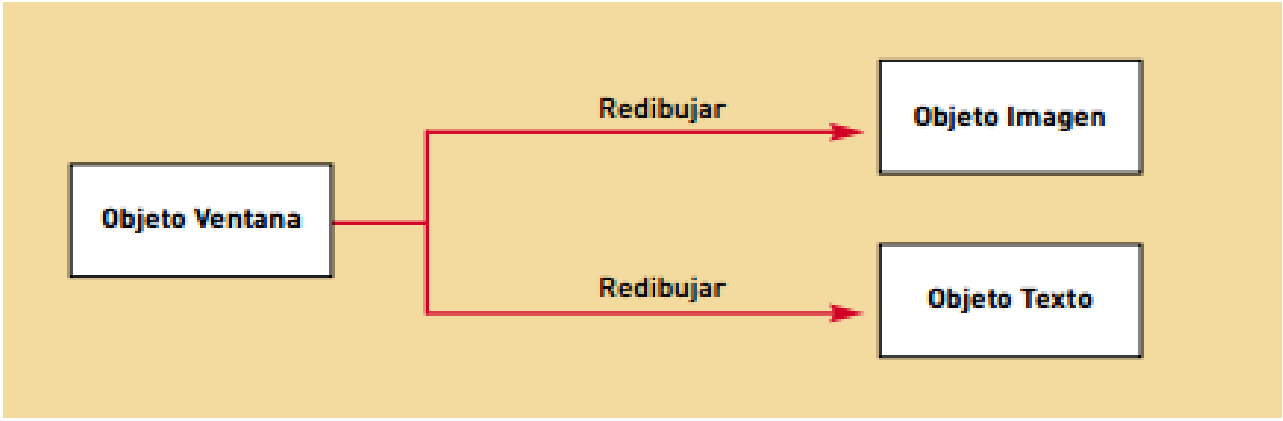

***Figura 2**. En la programación orientada a objetos se facilita la modificación y el mantenimiento de los programas, ya que cualquier modificación interna a un objeto no afectará a los otros.*

¿QUÉ LENGUAJE CONVIENE ELEGIR?

La elección de un lenguaje de programación no es algo trivial. Antes que nada, debemos determinar qué clase de programas pensamos hacer. Si, por ejemplo, apuntamos a hacer programas que manejen bases de datos, nos convendrá dominar el lenguaje SQL. Pero el SQL no es muy adecuado para crear interfaces de usuario, por lo cual deberemos complementarlo con algún lenguaje que sí lo sea; por ejemplo, algún lenguaje que ofrezca un **entorno visual**, como Visual Basic o Delphi.

Los lenguajes puramente declarativos –el Haskell, el ProLog o el Lisp– son usados principalmente por investigadores y programadores muy vanguardistas en áreas como la inteligencia artificial.

Para tareas de programación más cotidianas –como podría ser escribir un programa para ordenar un conjunto de datos–, la mayoría de los programadores emplea lenguajes de propósito general propios de la programación estructurada, entre los cuales los más representativos son el Pascal y el C. Estos dos lenguajes son similares en su capacidad expresiva, pudiéndose utilizar cualquiera de los dos indistintamente para implementar cualquier programa estructurado. En particular, el Pascal es más fácil de aprender debido a que la mayoría de sus instrucciones son vocablos del idioma inglés (lamentablemente, no hay un Pascal en español), mientras que el C es más sintético, pues utiliza menos palabras y más símbolos para representar las instrucciones Tanto el Pascal como el C tienen versiones orientadas a objetos (en el caso del C se denomina C++), lo cual les permite mantenerse vigentes ante las "nuevas modas" en materia de programación, a diferencia de otros lenguajes, como por ejemplo, el FORTRAN o el COBOL, que han caído en desuso.

PSEUDOCÓDIGO

Existe un lenguaje de programación que, curiosamente, no se utiliza para crear programas ejecutables. Sirve solamente para que el programador exprese sus ideas de una forma clara y que estas ideas puedan comunicarse a otros programadores para que ellos las implementen en el lenguaje que prefieran.

INTERFAZ

Interfaz es todo aquello que media entre dos mundos diferentes creando la posibilidad de comunicación; en este caso, entre el hombre y la máquina. En la actualidad las interfaces están compuestas por ventanas, botones y barras, pero en el comienzo de la programación todo se realizaba por medio de comandos (o sea, escribiendo).

Este lenguaje del que hablamos es el **pseudocódigo**. A pesar de que no sirve para crear en forma directa programas ejecutables, es una herramienta de gran utilidad para programar "en papel"; es decir, para esbozar cómo un programa debe llevar a cabo su tarea.

Por ejemplo, el siguiente programa en pseudocódigo se utiliza para encontrar el mayor elemento dentro de una lista de datos:

```
Función BuscarMaximo(lista)
    Mayor = lista(1)
    Contador = 2
    Mientras Contador ≤ longitud(lista) hacer
        Si lista(Contador) > Mayor entonces
            Mayor = lista(Contador)
        Fin Si
        Contador = Contador + 1
    Fin Mientras
    Devolver Mayor
Fin Función
```

Dado que no se trata de un lenguaje "formal", nos da la libertad de escribirlo en el idioma de nuestra preferencia (en este caso, en español).

Otra gran ventaja del pseudocódigo es que nos permite hacer **refinamientos sucesivos** de un programa; esto quiere decir que nuestro programa, al principio, puede expresar las instrucciones de una forma muy general (por ejemplo, en vez de detallar todas las instrucciones para recorrer una lista de datos, simplemente escribimos: **Recorrer Lista**), y luego de varios refinamientos del programa, especificamos cada vez más las instrucciones, hasta llegar a un punto en el que prácticamente pueden traducirse a un verdadero lenguaje de programación.

EJECUTABLES FICTICIOS

Algunos entornos de desarrollo –como el Visual Basic– dan como resultado archivos ejecutables (**.EXE**) que a simple vista parecen programas autónomos o stand-alone. Sin embargo, contienen instrucciones que debe interpretar un componente denominado run-time, que debe estar instalado en la computadora de destino para que el programa se ejecute.

TOMADO DEL LIBRO “INTRODUCCIÓN A LA PROGRAMACIÓN”
DISPONIBLE EN http://img.redusers.com/imagenes/libros/ldrme014/capitulogratis.pdf
CONTACTO drmax@mpediciones.com