¿Qué significa programar?

¿De qué nos ocuparemos?
Una introducción a la terminología de la programación, un poco de historia y una introducción a la estructura de un programa de computadora.

Vayamos a lo BASICo

La programación de computadoras es el arte de hacer que una computadora haga lo que nosotros querramos.

En el nivel más simple consiste en ingresar en la computadora una secuencia de órdenes para lograr un cierto objetivo. En el entorno de MS DOS los usuarios solían crear archivos de texto con comandos denominados "archivos por lotes" (.BAT). Estos simplemente ejecutaban la secuencia de órdenes en lotes, de allí su nombre. Bajo Windows es posible producir estos archivos, aunque en la práctica no es lo más común.

Por ejemplo, podrías producir un documento (como este tutorial) compuesto por varios archivos separados. Tu procesador de texto puede crear backups de cada uno de estos archivos cuando guardás una nueve versión de los documentos. Al final del día, querés colocar la versión actual del documento, es decir los últimos archivos, en una carpeta de respaldo. Finalmente, para poner un poco de orden, borrás las versiones previas. Un sencillo archivo BAT para hacer esto sería:

```
COPY *.HTM BACKUP
DEL *.BAK
```

Si el archivo se llamara SAVE.BAT, al final del día simplemente deberías tipear SAVE en la ventana de DOS y los archivos serían guardados y se borrarían los backups. Esto es un programa.

Nota: Los usuarios de Linux y otros sistemas operativos tiene su propia versión de estos archivos, usualmente conocidos como *shell scripts*. Los shell scripts de Linux son mucho más poderosos que los BATs de DOS, y además son compatibles con la mayor parte de las técnicas de programación que veremos en este tutorial.

Voy a decirlo otro vez

Si estás un poco impresionado por lo que acabamos de ver, no te preocupes. Un programa de computación es simplemente un conjunto de instrucciones que le dicen a la computadora cómo realizar una tarea en particular. Es parecido a una receta: un grupo de instrucciones que le dicen al cocinero cómo preparar un determinado plato. Describe los ingredientes (los datos) y la secuencia de pasos (el proceso) necesarios para convertir los ingredientes en una rica torta. Un programa tiene un concepto muy similar.

Un poco de historia

Tal como vos le hablás a un amigo en un idioma, para "hablarle" a una computadora es necesario utilizar un lenguaje en particular. El único lenguaje que una computadora entiende se denomina *binario* y tiene muchos dialectos. Esto demuestra porqué un programa escrito para una iMac no funciona en una PC y viceversa. Desafortunadamente el lenguaje binario es muy difícil de leer y escribir para un humano, por lo cual debemos utilizar un lenguaje intermedio que después será traducido a binario. Esto es como ver una reunión cumbre entre Clinton y Yeltsin. Clinton habla, luego un intérprete repite en ruso lo que se ha dicho. Luego Yeltsin contesta y un intérprete repite lo mismo en inglés.

Lo que traduce nuestro lenguaje intermediario a binario también se denomina intérprete. De la misma manera que es necesario disponer de un intérprete distinto para traducir del inglés al ruso que para hacerlo del árabe al ruso, será necesario disponer de un intérprete distinto para traducir las órdenes de Python a binario y otra para traducir las de BASIC.

Los primeros programadores tenían que ingresar los códigos binarios, lo cual se conoce como programación en *código máquina* y es increíblemente compleja y difícil. El paso siguiente fue crear un traductor que simplemente convertía palabras en inglés equivalentes a los códigos binarios en los propios códigos binarios. De esta manera en vez de tener que recordar que el código `001273 05 04` significaba `sumar 5 + 4` , los programadores podían escribir entonces `ADD 5 4`. Esta simple mejora hizo que la vida fuera más sencilla y estos sistemas de codificación fueron los primeros lenguajes de programación, habiendo distintas versiones para cada tipo de computadora. Se los conocía como lenguajes *assembler ("ensamblador")*. La *programación en Assembler* se utiliza todavía para algunas tareas de programación muy específicas.

Incluso este sistema era muy primitivo, pues le decía a la computadora lo que tenía que hacer en el nivel de hardware -mover bytes de una celda de memoria a otra, sumar este byte a este otro, etc. Lograr un objetivo sencillo era todavía bastante difícil e implicaba un gran esfuerzo de programación.

Gradualmente los expertos en computación desarrollaron lenguajes de alto nivel para facilitar el trabajo de los programadores. Esto fue también el resultado de una demanda por parte de los usuarios que reclamaban tareas más complejas y procesos más potentes para sus computadoras. La competencia entre los ingenieros y los usuarios continúa aun hoy, y nuevos lenguajes son desarrollados y potenciados. Esto vuelve muy interesante a la programación pero también implica que como programador debés comprender bien no solo los conceptos de la programación en general, sino también la práctica de la programación en un lenguaje particular.

Más adelante discutiremos en profundidad algunos de estos conceptos.

Características comunes a todos los programas

Hace tiempo Edsgar Dijkstra desarrolló el concepto de la *programación estructurada*. Esto significa que todos los programas pueden estructurarse de las siguientes cuatro formas:

- Secuencias de instrucciones
- Bucles
- Bifurcaciones
- Módulos

Además de estas estructuras los programas necesitan otras características que los hacen útiles:

- Datos
- Operaciones (sumar, restar, comparar, etc.)
- Capacidad de Entrada/Salida (para mostrar resultados)

Una vez que se comprende cómo un lenguaje particular implementa estos conceptos, uno está preparado para escribir un programa en ese lenguaje.

Aclarando la terminología

Ya hemos dicho que la programación es el arte de hacer que una computadora haga lo que uno quiera, pero ¿qué es *un programa*?

De hecho hay dos conceptos distintos de lo que es un programa. El primero es el percibido por el usuario: un archivo ejecutable que se instala en la máquina y puede ser ejecutado repetidas veces para realizar una tarea determinada. Por ejemplo, los usarios utilizan el programa Word para escribir textos. El otro concepto se refiere a un programa visto desde la óptica de un programador: un archivo de texto con instrucciones a la computadora escritas en un determinado lenguaje de programación, que luego podrá convertirse en un ejecutable. Por esta razón, es necesario aclarar a qué nos referimos cuando hablamos de un programa.

Básicamente un programador escribe un programa en un lenguaje de alto nivel que es interpretado y traducido a bytes que la computadora pueda comprender. En la jerga técnica se dice que el programador genera el *código fuente* y el intérprete genera el *código objeto*. A veces, el código objeto también se denomina *P-Code, código binario* o *código máquina*.

El intérprete tiene varios nombres: uno es el de *intérprete* y el otro es el de *compilador*. Estos términos en realidad se refieren a dos técnicas diferentes de generación del código objeto a partir del código fuente. Antes los compiladores generaban el código objeto que podía ejecutarse como tal (un *archivo ejecutable* - otro término) mientras que un intérprete debía estar presente para poder ejecutar el código. Actualmente, la diferencia entre estos términos se ha vuelto difusa ya que algunos compiladores requieren que el intérprete esté presente para realizar a conversión final y algunos intérpretes compilan el código fuente en un objeto temporal y luego lo ejecutan.

Desde nuestra perspectiva no hay una diferencia real, ya que escribiremos código y usaremos una programa para que la computadora pueda leerlo y ejecutarlo.

La estructura de un programa

La estructura exacta de un programa depende del lenguaje que utilicemos y el entorno en el cual lo creemos. Sin embargo, hay algunos principios generales:

- Un cargador - todo programa necesita ser cargado en la memoria por el sistema operativo. De esto se encarga el intérprete.
- Definición de los datos - la mayoría de los programas operan con datos y por lo tanto en el código fuente debemos definir que tipo de datos vamos a utilizar en el programa. Esto se realiza de manera diferente en los distintos lenguajes. Todos los lenguajes que usaremos tienen la posibilidad de crear una nueva definición de datos simplemente al utilizar los datos. Veremos esto en la próxima sección.
- Instrucciones - son la parte central del programa. Las instrucciones manipulan los datos que hemos definido, realizan cálculos, muestran los resultados, etc.

La mayoría de los programas siguen una de dos estructuras:

Programas de lotes

Estos se ejecutan típicamente desde una línea de comando o automáticamente desde otra aplicación (tipo scheduler) y tienden al siguiente patrón:

- Inicialización interna de los datos
- Lectura de los datos ingresados
- Procesamiento de los datos
- Visualización o ejecución de los resultados

Programas controlados por eventos

La mayor parte de las interfaces gráficas (y los sistemas de control presentes en un horno a microondas o una cámara por ejemplo) responden a eventos. Esto significa que el Sistema Operativo envía un evento al programa y este los responde tan pronto como estos le llegan. Los eventos incluyen acciones del usuario como apretar una tecla, mover el mouse, etc, y operaciones propias del sistema operativo tales como la actualización del reloj, el refresco de la pantalla, etc.

Los programas controlados por eventos son generalmente así:

- Inicialización interna de los datos
- Espera de los eventos
- Identificación de los eventos y actuación en consecuencia

Para recordar
• Los programas controlan a la computadora • Los lenguajes de programación nos permiten "hablar" con la computadora a un nivel más cercano al pensamiento humano que al de la computadora • Los programas *operan* con *datos* • Los programas pueden ser *por Lotes* o *controlados por eventos*

Anterior Próxima Contenido

Si tenés preguntas o sugerencias, enviáme un e-mail a alan.gauld@btinternet.com