Tutorial de programación usando Loro

Equipo Loro
http://loro.sf.net
(En elaboración)

16 de mayo de 2003

Este documento busca servirte como guía de introducción a la programación usando el lenguaje Loro y el Entorno Integrado LoroEDI. No se espera que cuentes con mayores conocimientos de programación pero tampoco se pretende que esta guía, por sí sola, satisfaga todas tus dudas o inquietudes. Más bien, se sugiere utilizar este documento como complemento de otras fuentes de información disponibles. La versión más reciente de este tutorial puede obtenerse a través del sitio oficial del proyecto Loro. Todo comentario para mejorar la exposición en este documento es bienvenido.

Antes de empezar...

La mejor manera de aprender a programar es "programando." Así que asegúrate de tener abierto el Entorno de Desarrollo Integrado de Loro al tiempo que lees lo que sigue. Desde la página del proyecto en http://loro.sf.net puedes descargar el sistema completo y obtener la información necesaria sobre su forma de instalación y ejecución en diversos sistemas operativos, además del Manual de Usuario que te explica cada comando disponible en el entorno LoroEDI.

Para empezar...

Si has observado algunos listados de programas por ahí (no importa mucho en qué lenguaje particular), quizá tengas la impresión de que hay muchos elementos presentes y muchas normas que seguir; por lo tanto, es posible que creas que no es fácil aprender a programar. Las buenas noticias son que –afortunadamente para nosotros y también para el computador– los lenguajes están pensados para facilitar, aunque no parezca a primera vista, la forma en que le comunicamos órdenes a la máquina. Las malas noticias (pero no lo tomes muy en serio) son que "programar" no solo es cuestión de aprender un lenguaje de programación.

Como simple comparación, no basta con saber el idioma español para que podamos escribir una obra de valor literario.

Para facilitar la introducción de conceptos, utilizaremos del entorno LoroEDI en donde hay un elemento que nos va a ser de gran ayuda; se trata de "alguien" que nos servirá de intérprete para comunicarnos con la máquina. Su nombre es Intérprete Interactivo, pero, en algunas ocasiones lo llamaremos "II" para abreviar.

Este II (el Intérprete Interactivo) no entiende nuestro idioma natural, pero sí el lenguaje Loro. Así que nos entenderemos con él intercambiando mensajes escritos en el lenguaje Loro. ¿Complicado? Quizá por ahora, pero en todo caso ¡mucho mejor que sea en Loro y no en el lenguaje de unos y ceros (o de interruptores electrónicos) de la máquina! Como el objetivo de los lenguajes de programación es facilitarnos el trabajo, dejaremos que el II se las arregle para traducir nuestras instrucciones en órdenes en unos y ceros para que así, las ejecute la máquina.

El comando *Alt-I*, en el entorno LoroEDI, hará aparecer una ventana conteniendo algo como:

```
$
```

Se trata del símbolo que nos muestra el II para indicarnos que está esperando a que le digamos algo a la máquina en el lenguaje convenido. Su misión es comunicarse con la máquina (y no nos importa ahora cómo) para que se lleve a cabo la acción que queremos. El II se encargará también de traducir el resultado dado por la máquina para mostrarnos un mensaje de respuesta, que vamos a entender porque estará escrito también en el lenguaje convenido. Bueno, para ser más exactos, habrá ocasiones en que el texto de respuesta no estará en Loro, lo que sucederá cuando se trata de mensajes de error o cuando la orden que hemos dado tiene el efecto lateral de escribir cosas en la pantalla.

Cuando se trata de un mensaje directo dado por la máquina como resultado del comando, éste se mostrará a continuación del símbolo de igualdad '='. Cuando se trata de una mensaje de error, éste aparecerá en color rojo y cada línea comenzará con el símbolo !. Como tercer caso se tiene el texto escrito por pantalla por el comando ingresado (a través de ciertos mecanismos que veremos más adelante); éste aparecerá en color azul.

Primeros balbuceos

Ensayemos algunos "comandos" un poco disparatados sólo con el fin de ver cómo reacciona el II, así empezamos a ganar familiaridad.

```
$ hola!
! Error sintáctico. Encontrado "!"
! Se espera uno de:      . . . (resto de línea omitida)
```

¡Este intérprete no entiende ni un simple saludo! `:-(` Como parece que no le gustó la exclamación, intentemos simplemente:

```
$ hola
! Variable 'hola' no declarada
```

;-(¿Y ahora qué error es éste? ¿Qué es eso de "variable"? Pues bien, aunque el concepto de variable es fundamental, todavía no es momento de abordarlo. Por ahora baste decir que éste no es un error de tipo sintáctico (el mensaje no dice que lo sea), es decir, la orden está bien formada (sigue la sintaxis correcta) pero falla por alguna otra razón.

Intentemos algo más simple:

```
$ 2
= 2
```

:-) ¡Ah!, por fin hemos escrito algo correcto completamente, o bueno, por lo menos no se generó ningún error. El símbolo de igualdad = es utilizado por el II para indicarnos la respuesta dada por la máquina. Antes de continuar, quizá quieras hacer un poco más de exploración por tu cuenta. No debes tomar los mensajes de error como alguna especie de castigo; por el contrario, ellos son una tremenda ayuda para aprender.

Algo de aritmética

Los computadores parecen ser tan "hábiles" que las operaciones aritméticas como sumar, restar, etc., deben resultarles triviales. Y cualquiera que sea el lenguaje de comunicación con la máquina, por lo menos debe permitirnos hacer lo que hacemos con una simple calculadora de bolsillo. Y claro, eso es fácil en Loro. Digamos algo simple como:

```
$ 2 + 2
```

En lenguaje natural le estamos diciendo a la máquina: "por favor maquinita, dame el resultado de sumar dos más dos." Completamos el mensaje presionando la tecla *Intro* y a continuación el II nos avisa que la máquina ha obedecido la orden y que el resultado de la expresión es:

```
= 4
```

Nada sorprendente, desde luego, pero lo importante es que aparecen más elementos básicos del lenguaje en escena. En primero término, están las operaciones aritméticas. El lenguaje nos debe permitir comunicarle a la máquina la realización de operaciones aritméticas. Y por operación se está queriendo decir un tipo especial de comando, es decir, una petición a la máquina para que obtenga el resultado correspondiente. Una operación aritmética es un caso particular de expresión. Luego estaremos mostrando otros tipos de expresiones. Ahora querrás intentar tus propias expresiones (ensaya agrupamientos con paréntesis, por ejemplo).

Algo menos aritmético

La aritmética parece fácil hasta ahora. Pero no sólo números se ven en las pantallas de los computadores. De hecho, muy probablemente tu estás ahora mismo leyendo este texto en la pantalla. Así que el lenguaje nos debe permitir darle comandos a la máquina para que despliegue textos además de simples números. En Loro se utiliza el término *cadena* para hacer referencia a una secuencia de caracteres particular. Una forma de indicar una cadena mediante el lenguaje es simplemente encerrando sus letras entre comillas dobles:

```
$ "hola mundo"
= "hola mundo"
```

El II nos responde exactamente con la misma cadena. Muchas veces necesitaremos hacer operaciones y mostrar mensajes de manera combinada. Por ejemplo, un mensaje más completo sería indicado a la máquina como:

```
$ "El resultado de 9 + 5 es 15"
= "El resultado de 9 + 5 es 15"
```

¡Vaya! `:-(` acabo de escribir esta cadena muy deprisa y el mensaje me ha salido incorrecto (en realidad, 9 + 5 = 14). Sería mejor que el resultado fuera calculado por la misma máquina (que es muy precisa) y no arriesgarnos a mostrar mensajes erróneos por un descuido de escritura. Necesitamos entonces medios de composición de textos; en el ejemplo, obtener un mensaje final compuesto de la cadena "El resultado de 9 + 5 es" y el resultado de la operación aritmética 9 + 5. Para ello, aprovecharemos dos características de Loro en este sentido:

- El resultado de la evaluación de toda expresión puede manejarse como una cadena.
- Dos cadenas pueden concatenarse para obtener la cadena completa.

La primera característica significa que todo valor computado por la máquina puede promoverse a su representación como cadena de caracteres. Por ejemplo, el valor numérico 2003 no sólo puede ponerse en el contexto de una operación de suma, resta y otras de índole aritmética, sino también que puede obtenerse fácilmente su representación como cadena "2003" para fines de composición de textos. Una forma explícita de obtener la representación cadena de un valor es dando la orden `como cadena` a continuación del valor:

```
$ 2003 como cadena
= "2003"
```

En cuanto a la segunda característica, la concatenación entre dos cadenas se define como la cadena con todos los caracteres de la primera cadena seguidos de los de la segunda cadena. Por ejemplo, si se concatenan las cadenas "concate" y "nación" se obtiene la cadena "concatenación". El operador de concatenación de cadenas es el mismo símbolo utilizado para sumar, '+'. La máquina no sabe esto,

pero el II se las arregla para entender cuando queremos sumar dos números o cuando queremos concatenar dos cadenas. Los dos ejemplos siguientes ilustran esto muy claramente:

```
$ 2 + 1
= 3
$ "2" + "1"
= "21"
```

En efecto, el II procede del siguiente modo: si por lo menos uno de los dos operandos a los lados del símbolo + es una cadena, entonces el II entiende que queremos una concatenación de cadenas. En caso contrario, entenderá que deseamos una suma aritmética. Cuando se trata de concatenación y uno de los operandos no es una cadena, entonces el II se encarga automáticamente de obtener su versión cadena, de tal manera que la concatenación tenga sentido. Esto nos ahorra un poco de escritura en la composición de textos. Un ejemplo:

```
$ "este es el año " + 2003
= "este es el año 2003"
```

¡Qué bien! :-) Volvamos entonces a nuestro objetivo inicial y escribamos sin más demora:

```
$ "el resultado de 9 + 5 es " + 9 + 5
= "el resultado de 9 + 5 es 95"
```

¿Y ahora qué está pasando aquí?... Veamos. Lo que sucede es que el operador + se aplica de izquierda a derecha. Examinando por partes, el primer + se aplica a los dos operandos "el resultado de 9 + 5 es " y 9. Para probarlo directamente:

```
$ "el resultado de 9 + 5 es " + 9
= "el resultado de 9 + 5 es 9"
```

y es este resultado el que se utiliza finalmente en la segunda aplicación de + junto con el numero 5:

```
$ "el resultado de 9 + 5 es 9" + 5
= "el resultado de 9 + 5 es 95"
```

O sea que ambas operaciones vienen a ser de concatenación. Esa es la explicación.

Un cambio de orden nos permite ilustrar esta situación un poco más:

```
$ 9 + 5 + " es el resultado de 9 + 5"
= "l4 es el resultado de 9 + 5"
```

¡Bien! En este caso el primer + es de suma y el segundo + es de concatenación.

Pero aún queremos el orden que buscábamos originalmente; ¿qué hacemos? La solución es sencilla: simplemente utilicemos paréntesis para hacer explícito el orden en que queremos que se evalúen las expresiones. En nuestro caso queremos que se haga primero la suma 9 + 5 y luego la concatenación; pues bien, ponemos la suma aritmética entre paréntesis:

```
$ "el resultado de 9 + 5 es " + (9 + 5)
= "el resultado de 9 + 5 es 14"
```

¡Por fin!

[*continuará ...]*