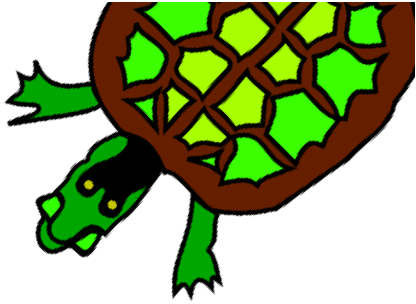


Capítulo 10

Listas

Ya hablamos antes de listas. `[ 53 gato [7 28] 4.9 ]` es una lista en xLogo; su primer elemento es `53`, el segundo es `gato`, el tercero es `[7 28]` y el último es `4.9`. Para almacenarlo en la variable `ejemplo` hacemos:

```
haz "ejemplo [ 53 gato [7 28] 4.9 ]
```

Debemos entender que las listas permiten guardar la información ordenada, y que son un elemento muy importante para simplificar muchos programas.

10.1. Primitivas

Disponemos de las siguiente primitivas para trabajar con listas y con palabras:

Para manejarlas

Primitiva	Forma larga	Forma corta
Devolver el primer elemento	`primero :ejemplo`	`pr :ejemplo`
Devolver el último elemento	`ultimo :ejemplo`	
Devolver el elemento `n`–simo	`elemento n :ejemplo`	
Investigar `algo` en `variable`	`miembro "algo "lista`	
Contar el número de elementos	`cuenta :ejemplo`	
Devolver un elemento al azar	`elige :lista`	

Ejemplo con una lista:

```
    haz "lista1 [Esto es una lista en xLogo]
#
    escribe primero :lista1            ---> Esto
```

```
escribe ultimo :lista1            ---> xLogo
escribe elemento 3 :lista1        ---> una
escribe miembro "es :lista1       ---> es una lista en xLogo
escribe cuenta :lista1            ---> 6
escribe elige :lista1             ---> en
```

Ejemplo con una palabra:

```
escribe primero "Calidociclos     ---> C
escribe ultimo "Calidociclos      ---> s
escribe elemento 3 "Calidociclos  ---> l
escribe cuenta "Calidociclos      ---> 12
escribe miembro "es "Calidociclos ---> falso
escribe elige "Calidociclos       ---> i
```

obviamente, el resultado de `elige` variará de una ejecución a otra.
Para la primitiva `miembro`:

- Si `variable` es una lista, investiga dentro de esta lista; hay dos casos posibles:
  - Si `algo` está incluido en `variable`, devuelve la sub—lista generada a partir de la primera aparición de `algo` en `variable`.
  - Si `algo` no está incluido en `variable`, devuelve la palabra `falso`.
- Si `variable` es una palabra, investiga los caracteres `algo` dentro de `variable`. Dos casos son posibles:
  - Si `algo` está incluido en `variable`, devuelve el resto de la palabra a partir de `algo`.
  - Si no, devuelve la palabra `falso`.

Para modificarlas

En este caso, sólo las dos primeras pueden usarse con palabras:

Primitiva	Forma larga	Forma corta
Quitar el primer elemento/letra	`menosprimero :ejemplo`	`mp`
Quitar el último elemento/letra	`menosultimo :ejemplo`	`mu`
Quitar el elemento `gato`	`quita "gato :ejemplo`	
Añadir `algo` el primero	`ponprimero "algo :ejemplo`	`pp`
Añadir `algo` el último	`ponultimo "algo :ejemplo`	`pu`
Intercalar `algo` en el lugar `n`	`agrega Lista n "algo`	
`Reemplaz`ar el elemento `n`	`reemplaza Lista n "algo`	
Invertir la lista	`invierte :lista`	

Ejemplo con una lista (`lista1` es la misma de antes):

```
escribe menosprimero :lista1      ---> es una lista en xLogo
escribe menosultimo :lista1       ---> Esto es una lista en
escribe quita "es :lista1         ---> Esto una lista en xLogo
escribe ponprimero "Super :lista1 ---> Super Esto es una lista en xLogo
escribe ponultimo "2007 :lista1   ---> Esto es una lista en xLogo 2007
escribe agrega :lista1 4 "super   ---> Esto es una super lista en xLogo
escribe reemplaza :lista1 3 "ye   ---> Esto ye una lista en xLogo
escribe invierte :lista1          ---> xLogo en lista una es Esto
```

Ejemplo con una palabra:

```
escribe menosprimero "Calidociclos  ---> alidociclos
escribe menosultimo "Calidociclos   ---> Calidociclo
```

Para combinar

Primitiva	Forma larga	Forma corta
Combinar en una sola lista	`frase :ejemplo :algo`	`fr`
Concatenar en una sola palabra	`palabra :ejemplo :algo`	
Combinar en una lista de sublistas	`lista :ejemplo :algo`	

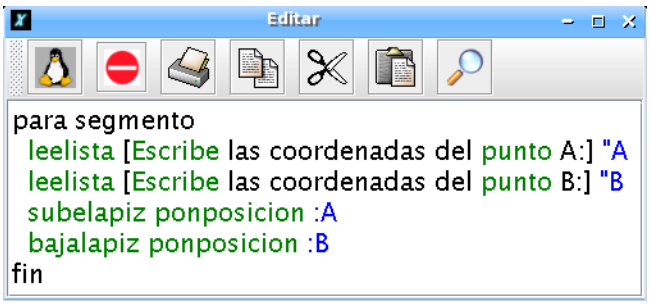
Ejemplos con una lista:

```
escribe frase :lista1 [y es genial]  ---> Esto es una lista en xLogo y es genial
escribe lista :lista1 [y es genial]
       `---> [Esto es una lista en xLogo] [y es genial]
```

Ejemplo con una palabra:

```
escribe frase "Calidociclos "de\ Escher  ---> Calidociclos de Escher
escribe lista "Calidociclos "de\ Escher  ---> Calidociclos de Escher
escribe palabra "Super "Calidociclos     ---> SuperCalidociclos
```

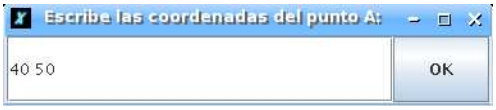
Existe una primitiva que permite que el *usuario* introduzca valores en xLogo, `leelista`:

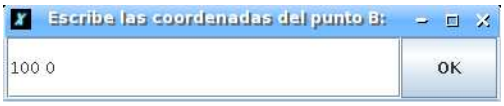


que se ejecuta tecleando:

```
segmento
```

xLogo abrirá una ventana pidiendo las coordenadas de `A`. Contestamos, por ejemplo, `40 50` y pulsamos `Intro`; nos pide luego las coordenadas de `B`, por ejemplo `100 0` e `Intro`.





La tortuga dibujará el segmento cuyos extremos son los puntos cuyas coordenadas hemos introducido.

Observación: `A` y `B` son **listas**, no números. En ejemplo anterior `A` contiene `[40 50]`, y podemos *convertir* sus elementos en números usando `primero`, `ultimo` ó `ultimo`, y usarlos con el resto de primitivas haciendo:

```
avanza primero :A
```

A la inversa, para convertir dos números en una lista tenemos la primitiva `lista`:

```
lista :lado :altura
```

Si se tratara de más de tres números disponemos de `frase` y `lista`:

```
haz "a 50 haz "b 60 haz "c 70
escribe lista :a frase :b :c
escribe lista :a lista :b :c
```

proporcionan: `50 [60 70]` pero sus *formas generales*:

```
escribe (lista :a :b :c)          escribe (frase :a :b :c)
```

proporcionan: `50 60 70` esto es, una única frase. La diferencia con `palabra` es que esta concatena las variables. En el ejemplo anterior:

```
escribe palabra :a :b
escribe (palabra :a :b :c)
```

proporcionan:

```
5060
506070
```

esto es, una única palabra (en este caso, un número).

10.2. Ejemplo: Conjugación

Vamos a construir un programa que conjugue el futuro simple de un verbo (sólo para verbos regulares) de distintas formas, y veremos cómo el uso de listas simplifica el manejo de la información.

10.2.1. Primera versión

La primera posibilidad que se nos puede ocurrir es usar una línea para cada persona:

```
para futuro :verbo
  es frase "yo palabra :verbo "é
  es frase "tú palabra :verbo "ás
  es frase "él palabra :verbo "á
  es frase "nosotros palabra :verbo "emos
  es frase "vosotros palabra :verbo "éis
  es frase "ellos palabra :verbo "án
fin
```

lo que consigue fácilmente nuestro objetivo:

Ejemplo: `futuro "amar`

```
yo amaré
tú amarás
él amará
nosotros amaremos
vosotros amaréis
ellos amarán
```

10.2.2. Segunda versión

Podemos ser un poco más elegantes, con la primitiva `repite` o `repitepara` (sección 11.1.2) combinada con listas:

```
para futuro :verbo
  haz "pronombres [yo tú él nosotros vosotros ellos]
  haz "terminaciones [é ás á emos éis án]
  repitepara [i 1 6]
   [ escribe frase elemento :i :pronombres
          palabra :verbo elemento :i :terminaciones ]
fin
```

ya que, realmente, estamos repitiendo seis veces la misma estrategia: combinar el pronombre con el verbo y la terminación. El resultado, el mismo de antes.

10.2.3. Tercera versión

En esta versión usaremos la recursividad, una característica muy útil en el uso de listas (ver sección 11.5)

Analiza qué hace este programa y cómo lo hace. Usa un papel para seguir la secuencia de pasos que da, y razona qué puede ser eso que llamamos *recursividad* (o *recurrencia*)

```
para futuro :verbo
  haz "pronombres [yo tú él nosotros vosotros ellos]
  haz "terminaciones [é ás á emos éis án]
  conjugar :verbo :pronombres :terminaciones
fin


para conjugar :verbo :pronombres :terminaciones
  si vacio? :pronombres [alto]
  escribe frase primero :pronombres palabra :verbo primero :terminaciones
  conjugar :verbo menosprimero :pronombres menosprimero :terminaciones
fin
```

10.3. Ejercicios

1. Plantea un programa sobre conjugación de verbos que:

   a) Pida un verbo con una ventana emergente

   b) Determine a qué conjugación pertenece

   c) Contenga dos listas:

      - `"pronombres` con los pronombres personales
      - `"morfemas` con los morfemas para conjugar el presente, siendo distinta en función de la conjugación del verbo

   d) Combine la raiz del verbo con las terminaciones del presente en una única palabra

   e) Combine en una frase los pronombres con la palabra generada, y las muestre en el Histórico de Comandos con `escribe`

2. ¿Cómo puede extraerse el `22` de la lista de listas `[[22 3] [4 5] [8 35]]`?

3. Plantea el procedimiento `prime`, con una entrada, `listado`, que devuelva su primer elemento, sin usar `primero`

4. Plantea el procedimiento `ulti`, con una entrada, `listado`, que devuelva su último elemento, sin usar `ultimo`

5. Diseña el procedimiento `triangulo`, que **pida** las coordenadas de los vértices de un triángulo uno a uno y lo dibuje

6. Escribe un procedimiento que pida la medida del lado de un cuadrado y devuelva la medida de su diagonal

7. Plantea el procedimiento `mengua` que pida escribir una serie de números y escriba las listas que se obtienen al ir eliminando un elemento de cada vez (por ejemplo el último) hasta quedar vacía.

8. Diseña un procedimiento `inversa` que reciba una lista y la devuelva con los elementos dispuestos en orden inverso al inicial

9. Plantea el procedimiento `maximo`, que pida una serie de números y devuelva el mayor de todos ellos

10. Diseña un procedimiento `suprime`, con dos entradas: `n` y `listado`, que devuelva la lista que se obtiene al suprimir el elemento `n`–simo, sin usar `quita`

11. Plantea el procedimiento `adjunta`, con tres entradas; `n`, `listado_1` y `listado_2`, que añada `listado_1` en la posición `n` de `listado_2` (sin usar `agrega`)

12. Escribe un procedimiento que determine si una palabra es un palíndromo, es decir, que se lee igual en la forma habitual que de derecha a izquierda

    `palíndromo? reconocer ---> cierto`

    `palíndromo? anilina ---> cierto`

    `palíndromo? animal ---> falso`

13. ¿Cómo ampliarías el procedimiento anterior para que trabajara también con frases?

    `palíndromo? Sé verlas al revés ---> cierto`

    `palíndromo? Átale o me delata ---> falso`

    Puedes encontrar palíndromos en:

    `http://es.wikipedia.org/wiki/Palíndromo`

    `http://es.wikiquote.org/wiki/Palíndromos`

    `http://www.juegosdepalabras.com/palindromo.htm`

    `http://www.carbajo.net/varios/pal.html`

10.4. Listas de Propiedades

Desde la versión 0.9.92, pueden definirse Listas de Propiedades con xLogo. Cada lista tiene un nombre específico y contiene una pareja de "valores–clave".

Para manejar estas listas, podemos utilizar las siguientes primitivas, para las que, por ejemplo, podemos considerar una lista llamada "coche", que debe contener la clave "`color`" asociado al valor "`rojo`", otra clave "`tipo`" con el valor "`4x4`" y una tercera clave "`vendedor`" asociada al valor "`Citröen`",

Descripción	Primitiva	Ejemplo (con forma corta)
Añadir una propiedad a la lista	`ponpropiedad`	`ponprop "Coche "Color "Rojo`
Devolver el valor asociado a una `clave` de la lista	`leepropiedad`	`leeprop "Coche "Color`
Borrar el par `clave-valor` de una lista	`borrapropiedad`	`boprop "Coche "Color`
Mostrar todos los elementos de una lista	`listapropiedades`	`listaprop "Coche`
Mostrar todas las listas creadas	`listaspropiedades`	`listasprop "Coche`

Juguemos con los elementos de la lista de propiedades llamada "coche".

```
# Llenado de la Lista de Propiedades
ponpropiedad "Coche "Color "Rojo
ponpropiedad "Coche "Tipo "4x4
ponpropiedad "Coche "Vendedor "Citroen

# Mostrar un valor
escribe leepropiedad "Coche "Color    ---> Rojo

# Mostrar todos los elementos
escribe listapropiedades "Coche       ---> Color Roja Tipo 4x4 Vendedor Citroen
```