Proyecto Logo

Fase 1

1. Introducción

Logo es un lenguaje de programación que pertenece a la familia de Lisp (programación funcional) y es particularmente conocido por sus gráficos de tortuga. Aunque es un lenguaje de amplio espectro, nos vamos a centrar precisamente en este aspecto gráfico.

Al ser un lenguaje de muy fácil aprendizaje, suele ser el lenguaje de programación preferido para trabajar con niños. Precisamente por ese motivo es de los pocos lenguajes de programación para los que existen versiones en diferentes idiomas.

Fue creado en 1967 por Daniel G. Bobrow, Wally Faurzeig y Seymour Papert.

2. Programación en Logo (gráficos de tortuga)

Al comenzar el programa aparece una tortuga en mitad de la pantalla que se puede ir desplazando al ejecutar determinados comandos. La tortuga dispone de un lápiz que, dependiendo de que esté en posición *baja* o *alta* va dejando o no un rastro por la pantalla.

En la siguiente tabla se muestran los comandos más frecuentes que puede interpretar la tortuga.

Comando	Abreviatura	Función
avanza N	av N	Mueve la tortuga hacia adelante la cantidad de píxeles especificada en N
retrocede N	re N	Mueve la tortuga hacia atrás la cantidad de píxeles especificada en N
giraizquierda N	gi N	Gira la tortuga hacia la izquierda los grados especificados en N
giraderecha N	gd N	Gira la tortuga hacia la izquierda los grados especificados en N
subelapiz	sl	La tortuga no dibuja mientras se desplaza
bajalapiz	bl	La tortuga dibuja mientras se desplaza
muestratortuga	mt	Muestra la tortuga en la pantalla
ocultatortuga	ot	Oculta la tortuga

En el aula virtual podéis encontrar un sencillo intérprete de Logo. Para ejecutarlo hay que teclear en la consola:

```
java -jar xlogo-new
```

3. La librería Entorno

Para que resulte sencillo crear un programa en C++ que realice gráficos como los que elabora la tortuga, disponéis de una librería llamada "Entorno" que, a su vez, utiliza una conocida librería de C++ llamada "allegro". A continuación se explica el funcionamiento de los módulos de la librería "Entorno".

`void inicio();`

Inicia todas las variables necesarias para que el resto de los módulos puedan ejecutarse correctamente. Crea una ventana de trabajo de 800*600. Debe ser el primer módulo en ser llamado. Es importante tener en cuenta las dimensiones de la ventana y que la esquina superior izquierda de la misma tiene las coordenadas (0,0). Por ejemplo, el punto central de la ventana (que es la posición inicial de la tortuga) tiene las coordenadas (400,300).

`void fin();`

Finaliza el entorno allegro

`void linea(int x1,int y1,int x2,int y2);`

Dibuja una línea desde un punto de la pantalla con coordenadas (x1,y1) hasta otro con coordenadas (x2,y2)

`void pon_tortuga(int x1,int y1,int orientacion);`

Coloca en la imagen de la tortuga en la posición de la pantalla (x1,y1). La orientación de la imagen se define a partir del tercer parámetro (*orientacion*), según los siguientes valores

0: Norte
1: Este
2: Sur
3: Oeste

`void borra_tortuga(int x1,int y1);`

Borra la imagen de la tortuga situada en el punto (x1, x2)

4. Proyecto Logo. Fase 1.

La Fase 1 de este proyecto consistirá en construir un compilador que traduzca un programa escrito en Logo a un programa escrito en C++.

En esta primera fase, sólo tendremos en cuenta un número reducido de instrucciones de Logo, precisamente las que se han definido en el apartado 2.

Para que el trabajo de traducción sea más sencillo se recomienda utilizar la librería "Entorno" comentada en el apartado 3.

Esta primera fase del proyecto deberá entregarse a través del aula virtual antes del día **15 de abril** y se corregirá durante la semana del **16 de abril**.

Ejemplo de utilización.

Si suponemos que el fichero **prueba.lgo** tiene el siguiente código:

```
avanza 100
gd 90
av 100
gd 90
```

```
av 100
giraderecha 90
Av 100
Giraderecha 90
```

Al ejecutarse el compilador (supongamos que se llama “logo”) de la siguiente forma:

./logo prueba.lgo

Debe generarse un fichero llamado **prueba.cpp** con el siguiente contenido:

```
#include "Entorno.h"

int main(){
  inicio();
  pon_tortuga(400,300,0);
  readkey();

  borra_tortuga(400,300);
  linea(400,300,400,200);
  pon_tortuga(400,200,0);
  readkey();

  borra_tortuga(400,200);
  pon_tortuga(400,200,1);
  readkey();

  borra_tortuga(400,200);
  linea(400,200,500,200);
  pon_tortuga(500,200,1);
  readkey();

  borra_tortuga(500,200);
  pon_tortuga(500,200,2);
  readkey();

  borra_tortuga(500,200);
  linea(500,200,500,300);
  pon_tortuga(500,300,2);
  readkey();

  borra_tortuga(500,300);
  pon_tortuga(500,300,3);
  readkey();

  borra_tortuga(500,300);
  linea(500,300,400,300);
  pon_tortuga(400,300,3);
  readkey();

  borra_tortuga(400,300);
  pon_tortuga(400,300,0);
  readkey();

  fin();
  return 0;
}
```

Al ejecutarse este fichero se dibujaría en la pantalla un cuadrado de lado 100.

5. Cuestiones de interés

- Para poder utilizar correctamente la librería “Entorno”, el compilador debe conocer en cada momento la posición de la tortuga, que cambiará debido a los diferentes movimientos que realice. Se recomienda construir una librería auxiliar para definir y gestionar toda la información necesaria para trabajar con la tortuga.
- En Logo está permitido escribir varias instrucciones en la misma línea, pero una instrucción no puede partirse en varias líneas.
- En el código c que se va a generar se intercalará, después de la traducción de cada instrucción, una llamada a readkey() que permitirá comprobar la evolución del dibujo con más facilidad.
- Las palabras reservadas se pueden escribir en mayúscula o minúscula, cualquier combinación es válida.
- El lenguaje admite comentarios que comienzan con el símbolo '#' y acaban con la línea. Pueden aparecer sólos en una línea o detrás de las instrucciones.
- Para poder compilar y ejecutar el programa c generado es necesario tener instalada la librería allegro e incluir las correspondientes librerías en el proceso de compilación. Ejemplo de makefile necesario para crear el ejecutable a partir de *prueba.cpp* que es el fichero previamente generado por el compilador de logo.

```
prueba : prueba.o Entorno.o
```

```
g++ -oprueba Entorno.o prueba.o -lalleg -lX11 -lXpm -lXext -lXcursor -lpthread -lXxf86vm

Entorno.o : Entorno.cpp
   g++ -c Entorno.cpp

prueba.o : prueba.cpp
   g++ -c prueba.cpp
```

- Se recomienda la utilización de dos ficheros *makefile*. Uno para la creación del compilador a partir de los analizadores léxico y sintáctico, y otro para la creación del programa ejecutable a partir del programa “logo”. Si a uno de estos ficheros makefile se le da un nombre diferente a “makefile” o “Makefile”, para que se ejecute es necesario utilizar la opción -f de esta manera:

```
make -f nombre_fich_makefile
```