

Metapost: Figuras y Gráficos

Juan Gutiérrez Aguado

Departament d'Informàtica
Universitat de València

Marzo 2007

Índice

1 Metapost

- *Herramientas básicas*
- *Tipos de variables*
- *Texto*
- *Transformaciones y figuras predefinidas*
- *Repetición*
- *Intersecciones*
- *Variables de tipo picture*
- *Macros*

2 Un listado de otras herramientas

Índice

- 1 Metapost
- 2 Un listado de otras herramientas

Índice

1 Metapost

- *Herramientas básicas*
- Tipos de variables
- Texto
- Transformaciones y figuras predefinidas
- Repetición
- Intersecciones
- Variables de tipo picture
- Macros

Metapost es un lenguaje de programación para realizar figuras y gráficos.

La estructura de un documento metapost (`nomfich.mp`) es:

```
% Preámbulo  
  
beginfig(1);  
% Instrucciones para  
% generar la figura 1  
endfig;  
  
...  
  
beginfig(n);  
% Instrucciones para  
% generar la figura n  
endfig;  
end
```

Una vez se tiene este fichero se procesa mediante la utilidad `mpost`:

```
mpost nomfich.mp  
nomfich.1,..., nomfich.n
```

Estos ficheros se pueden incluir en latex utilizando `includegraphics`:

```
\includegraphics{nomfich.i}
```

O se pueden introducir en `pdflatex` cambiando su extensión a `mps`

```
\includegraphics{nomfich_i.mps}
```

Metapost es un lenguaje de programación para realizar figuras y gráficos.

La estructura de un documento metapost (`nomfich.mp`) es:

```
% Preámbulo  
  
beginfig(1);  
% Instrucciones para  
% generar la figura 1  
endfig;  
  
...  
  
beginfig(n);  
% Instrucciones para  
% generar la figura n  
endfig;  
end
```

Una vez se tiene este fichero se procesa mediante la utilidad `mpost`:

```
mpost nomfich.mp  
nomfich.1,..., nomfich.n
```

Estos ficheros se pueden incluir en latex utilizando `includegraphics`:

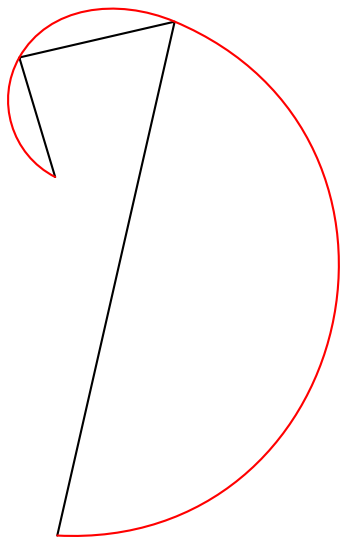
```
\includegraphics{nomfich.i}
```

O se pueden introducir en `pdflatex` cambiando su extensión a `mps`

```
\includegraphics{nomfich_i.mps}
```

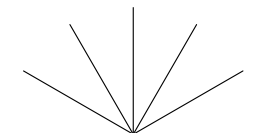
Líneas a lo largo de un camino definido por puntos

```
beginfig(1);  
  
draw (0,0) — (-3mm,1cm) —  
      (1cm,1.3cm) — (0.5,-3cm);  
  
draw (0,0) .. (-3mm,1cm) ..  
      (1cm,1.3cm) .. (0.5,-3cm)  
      withcolor red;  
  
endfig;  
  
end
```



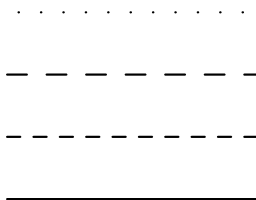
Líneas a lo largo de una dirección

```
beginfig(1);  
  
  draw (0,0) — 2cm*dir 0;  
  draw (0,0) — 2cm*dir 30;  
  draw (0,0) — 2cm*dir 60;  
  draw (0,0) — 2cm*dir 90;  
  draw (0,0) — 2cm*dir 120;  
  draw (0,0) — 2cm*dir 150;  
  draw (0,0) — 2cm*dir 180;  
  
endfig;  
  
end
```



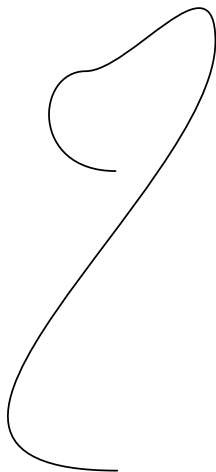
Diferentes tipos de trazos

```
beginfig(1);  
draw (0,0) — (2cm, 0);  
draw (0,0.5cm) — (2cm, 0.5cm)  
    dashed evenly;  
draw (0,1cm) — (2cm, 1cm)  
    dashed evenly scaled 1.5;  
draw (0,1.5cm) — (2cm, 1.5cm)  
    dashed withdots;  
endfig;  
end
```



Líneas curvas a lo largo de un camino definido por puntos especificando las tangentes

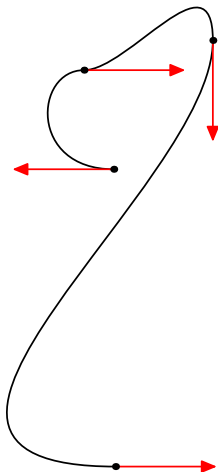
```
beginfig(1);  
draw (0,0){dir 180} ..  
      (-3mm,1cm){1,0} ..  
      (1cm,1.3cm){down} ..  
      (0.5,-3cm){right};  
endfig;  
end
```



Curva resultante

Líneas curvas a lo largo de un camino definido por puntos especificando las tangentes

```
beginfig(1);  
draw (0,0){dir 180} ..  
      (-3mm,1cm){1,0} ..  
      (1cm,1.3cm){down} ..  
      (0.5,-3cm){right};  
endfig;  
end
```



Estas son las tangentes
que se han especificado

Índice

1 Metapost

- Herramientas básicas
- *Tipos de variables*
- Texto
- Transformaciones y figuras predefinidas
- Repetición
- Intersecciones
- Variables de tipo picture
- Macros

Tipos de variables

- **numeric**: valor numérico
- **pair**: par de valores (para representar un punto)
- **path**: para almacenar un camino
- **color**: para almacenar un color (R,G,B)
- **string**: para almacenar cadenas
- **boolean**: para almacenar booleanos (true o false)
- **picture**: para almacenar una figura
- **pen**: para almacenar el estilo para trazar caminos
- **transform**: para almacenar una transformación

Declaración: `tipo nombreVariable;`

Asignación de un nuevo valor: `nombreVariable := nuevoValor;`

Tipos de variables

- `numeric`: valor numérico
- `pair`: par de valores (para representar un punto)
- `path`: para almacenar un camino
- `color`: para almacenar un color (R,G,B)
- `string`: para almacenar cadenas
- `boolean`: para almacenar booleanos (true o false)
- `picture`: para almacenar una figura
- `pen`: para almacenar el estilo para trazar caminos
- `transform`: para almacenar una transformación

Declaración: `tipo nombreVariable;`

Asignación de un nuevo valor: `nombreVariable := nuevoValor;`

Tipos de variables

- `numeric`: valor numérico
- `pair`: par de valores (para representar un punto)
- `path`: para almacenar un camino
- `color`: para almacenar un color (R,G,B)
- `string`: para almacenar cadenas
- `boolean`: para almacenar booleanos (true o false)
- `picture`: para almacenar una figura
- `pen`: para almacenar el estilo para trazar caminos
- `transform`: para almacenar una transformación

Declaración: `tipo nombreVariable;`

Asignación de un nuevo valor: `nombreVariable := nuevoValor;`

Tipos de variables

- `numeric`: valor numérico
- `pair`: par de valores (para representar un punto)
- `path`: para almacenar un camino
- `color`: para almacenar un color (R,G,B)
- `string`: para almacenar cadenas
- `boolean`: para almacenar booleanos (true o false)
- `picture`: para almacenar una figura
- `pen`: para almacenar el estilo para trazar caminos
- `transform`: para almacenar una transformación

Declaración: `tipo nombreVariable;`

Asignación de un nuevo valor: `nombreVariable := nuevoValor;`

Tipos de variables

- `numeric`: valor numérico
- `pair`: par de valores (para representar un punto)
- `path`: para almacenar un camino
- `color`: para almacenar un color (R,G,B)
- `string`: para almacenar cadenas
- `boolean`: para almacenar booleanos (true o false)
- `picture`: para almacenar una figura
- `pen`: para almacenar el estilo para trazar caminos
- `transform`: para almacenar una transformación

Declaración: `tipo nombreVariable;`

Asignación de un nuevo valor: `nombreVariable := nuevoValor;`

Tipos de variables

- `numeric`: valor numérico
- `pair`: par de valores (para representar un punto)
- `path`: para almacenar un camino
- `color`: para almacenar un color (R,G,B)
- `string`: para almacenar cadenas
- `boolean`: para almacenar booleanos (true o false)
- `picture`: para almacenar una figura
- `pen`: para almacenar el estilo para trazar caminos
- `transform`: para almacenar una transformación

Declaración: `tipo nombreVariable;`

Asignación de un nuevo valor: `nombreVariable := nuevoValor;`

Tipos de variables

- `numeric`: valor numérico
- `pair`: par de valores (para representar un punto)
- `path`: para almacenar un camino
- `color`: para almacenar un color (R,G,B)
- `string`: para almacenar cadenas
- `boolean`: para almacenar booleanos (`true` o `false`)
- `picture`: para almacenar una figura
- `pen`: para almacenar el estilo para trazar caminos
- `transform`: para almacenar una transformación

Declaración: `tipo nombreVariable;`

Asignación de un nuevo valor: `nombreVariable := nuevoValor;`

Tipos de variables

- `numeric`: valor numérico
- `pair`: par de valores (para representar un punto)
- `path`: para almacenar un camino
- `color`: para almacenar un color (R,G,B)
- `string`: para almacenar cadenas
- `boolean`: para almacenar booleanos (`true` o `false`)
- `picture`: para almacenar una figura
- `pen`: para almacenar el estilo para trazar caminos
- `transform`: para almacenar una transformación

Declaración: `tipo nombreVariable;`

Asignación de un nuevo valor: `nombreVariable := nuevoValor;`

Tipos de variables

- `numeric`: valor numérico
- `pair`: par de valores (para representar un punto)
- `path`: para almacenar un camino
- `color`: para almacenar un color (R,G,B)
- `string`: para almacenar cadenas
- `boolean`: para almacenar booleanos (`true` o `false`)
- `picture`: para almacenar una figura
- `pen`: para almacenar el estilo para trazar caminos
- `transform`: para almacenar una transformación

Declaración: `tipo nombreVariable;`

Asignación de un nuevo valor: `nombreVariable := nuevoValor;`

Tipos de variables

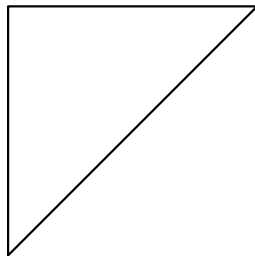
- `numeric`: valor numérico
- `pair`: par de valores (para representar un punto)
- `path`: para almacenar un camino
- `color`: para almacenar un color (R,G,B)
- `string`: para almacenar cadenas
- `boolean`: para almacenar booleanos (`true` o `false`)
- `picture`: para almacenar una figura
- `pen`: para almacenar el estilo para trazar caminos
- `transform`: para almacenar una transformación

Declaración: `tipo nombreVariable;`

Asignación de un nuevo valor: `nombreVariable := nuevoValor;`

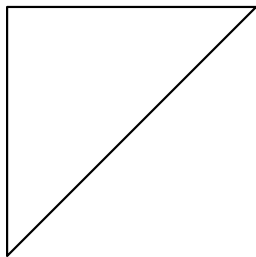
Ejemplos (I): pair

```
beginfig(1);  
pair A,B,C;  
A := (0,0cm);  
B := (0,2cm);  
C := (2cm,20mm);  
  
draw A -- B -- C -- cycle;  
  
endfig;
```



Ejemplos (II): pair y path

```
beginfig(1);  
  
path p;  
  
pair A,B,C;  
  
A := (0,0cm);  
B := (0,2cm);  
C := (2cm,20mm);  
  
p := A — B — C — cycle;  
  
draw p;  
  
endfig;
```



Ejemplos (III): pair, path y numeric

```

beginfig(1);
numeric a,b,c;
path p;
pair A,B,C,M;

a := 0.25;
b := 0.5;
c := 0.75;

A := (0,0cm);
B := (0,2cm);
C := (2cm,20mm);

p := A — B — C — cycle;
draw p;

M := a[A,B];
draw M withpen pencircle scaled 4bp;
M := b[B,C];
draw M withpen pencircle scaled 4bp;
M := c[C,A];
draw M withpen pencircle scaled 4bp;
endfig;

```

En metapost:

$$M = \text{lambda}[A,B];$$

Matemáticamente:

Recta en forma
paramétrica

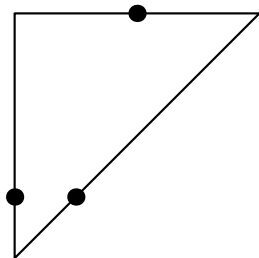
$$M_{\lambda} = A + \lambda(B - A);$$

$$\lambda = 0 \longrightarrow M_{\lambda=0} = A;$$

$$\lambda = 1 \longrightarrow M_{\lambda=1} = B;$$

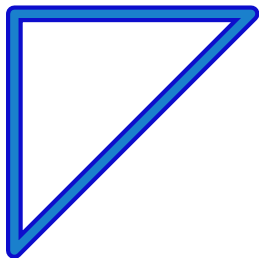
Ejemplos (III): pair, path y numeric

```
beginfig(1);  
numeric a,b,c;  
path p;  
pair A,B,C,M;  
  
a := 0.25;  
b := 0.5;  
c := 0.75;  
  
A := (0,0cm);  
B := (0,2cm);  
C := (2cm,20mm);  
  
p := A — B — C — cycle;  
draw p;  
  
M := a[A,B];  
draw M withpen pencircle scaled 4bp;  
M := b[B,C];  
draw M withpen pencircle scaled 4bp;  
M := c[C,A];  
draw M withpen pencircle scaled 4bp;  
endfig;
```



Ejemplos (IV): pair, path, color y pen

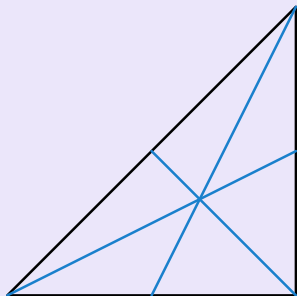
```
beginfig(1);  
pair A,B,C;  
path p;  
color col;  
pen trazo;  
  
col := (0.05,0.05,0.8);  
trazo:= pencircle scaled 4bp;  
  
A := (0,0cm);  
B := (0,2cm);  
C := (2cm,20mm);  
  
p := A — B — C — cycle;  
  
draw p withcolor col withpen trazo;  
  
col := (0.1,0.5,0.8);  
trazo:= pencircle scaled 2bp;  
  
draw p withcolor col withpen trazo;  
endfig;
```



Ejercicio

Tarea 1

Realizar las figureas en metapost como las que se muestran a continuación:



Incluye las figuras en un documento latex del tipo artículo.

Índice

1 Metapost

- Herramientas básicas
- Tipos de variables
- *Texto*
- Transformaciones y figuras predefinidas
- Repetición
- Intersecciones
- Variables de tipo picture
- Macros

Incluir texto en la figura: dotlabel

```

beginfig(1);
  pair A,B;

  A:=(0,0);
  B:=(1.6cm,0);

  dotlabel.lft("L", A);
  dotlabel.rt("R", A);
  dotlabel.top("T", A);
  dotlabel.bot("B", A);

  dotlabel.llft("LL", B);
  dotlabel.ulft("UL", B);
  dotlabel.lrt("LR", B);
  dotlabel.urt("UR", B);

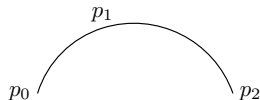
endfig;

end

```

Incluir texto en la figura: label

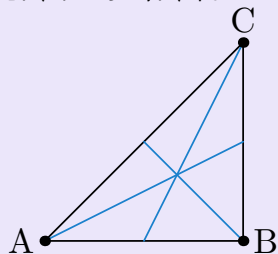
```
beginfig(1);  
  
  pair A,B,C;  
  path P;  
  
  A:=(0,0);  
  B:=(1cm,1cm);  
  C:=(3cm,0);  
  
  draw A .. B .. C;  
  
  label.lft(btex $p_0$ etex,A);  
  label.top(btex $p_1$ etex,B);  
  label.rt(btex $p_2$ etex,C);  
endfig;  
  
end
```



Ejercicio

Tarea 2

Modifica el código del ejemplo anterior para que muestre la siguiente figura:



Incluye la figura en el documento latex anterior.

Índice

1 Metapost

- Herramientas básicas
- Tipos de variables
- Texto
- *Transformaciones y figuras predefinidas*
- Repetición
- Intersecciones
- Variables de tipo picture
- Macros

Transformaciones

Algunas transformaciones predefinidas:

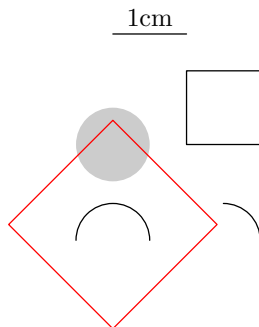
- `shifted pair`: desplaza al punto especificado
- `rotated numeric`: rota el ángulo especificado
- `scaled numeric`: escala en ambas direcciones
- `xscaled numeric`: escala en la dirección x
- `yscaled numeric`: escala en la dirección y

Figuras predefinidas

- `fullcircle`: una circunferencia de diámetro `1pb` y centro en $(0,0)$
- `halfcircle`: La mitad superior de de lo anterior
- `quartercircle`: La mitad derecha de lo anterior
- `unitsquare`: Cuadrado de vértices $(0,0)$, $(1,0)$, $(1,1)$, $(0,1)$

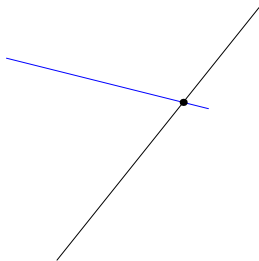
Ejemplos figuras predefinidas y transformaciones

```
beginfig(1);  
  
  draw (0,1.5cm) — (1cm,1.5cm);  
  label.top("1cm", (0.5cm,1.5cm));  
  
  fill fullcircle scaled 1cm withcolor  
    0.8white;  
  draw unitsquare scaled 1cm  
    shifted (1cm,0);  
  
  draw halfcircle scaled 1cm  
    shifted (0,-1.3cm);  
  
  draw quartercircle scaled 1cm  
    shifted (1.5cm,-1.3cm);  
  
  draw unitsquare scaled 2cm rotated 45  
    shifted (0,-2.5cm) withcolor red;  
  
endfig;  
end
```



Intersecciones entre rectas: whatever

```
beginfig(1);  
  numeric u;  
  pair A,B,C,D,M;  
  
  u:=1cm;  
  
  A := (-2u,-2u);  
  B := (2u,3u);  
  
  C := (u,u);  
  D := (-3u,2u);  
  
  %Un punto entre A y B  
  M = whatever[A,B];  
  %Que además debe estar entre C y D  
  M = whatever[C,D];  
  
  draw A — B;  
  draw C — D withcolor blue;  
  draw M withpen pencircle scaled 4bp;  
endfig;  
end
```



Intersecciones entre rectas: whatever I

```
beginfig(1);  
  
pair A,B,C,AB;  
pair N;  
  
% Vertices del triángulo  
  
A := (0,0cm);  
B := (2cm,0.3cm);  
C := (1.7cm,2.5cm);  
  
draw A — B — C — cycle;  
  
% Punto donde se cortan las bisectrices  
N = A + whatever*dir( (angle(B - A) + angle(C - A))/2);  
N = C + whatever*dir( (angle(A - C) + angle(B - C))/2);
```

Intersecciones entre rectas: whatever II

```
% Punto de la recta A-B más cercano a N
AB = whatever[A,B];
AB = N + whatever*(B-A) rotated 90;

draw N withpen pencircle scaled 2bp;

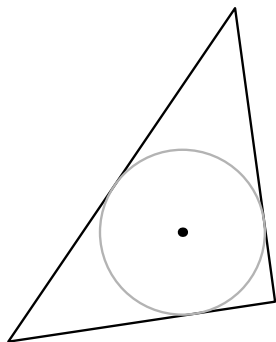
% abs(pair) es el módulo del vector
draw fullcircle scaled (2*(abs(AB-N))) shifted N
  withcolor 0.7white;

endfig;

end
```

Intersecciones entre rectas: whatever III

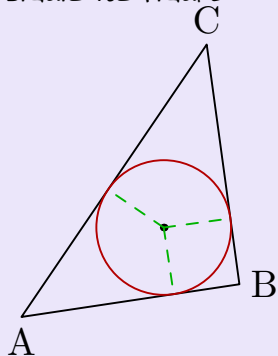
La figura que se genera es:



Ejercicio

Tarea 3

Modifica el código del ejemplo anterior para que muestre la siguiente figura:



Incluye la figura en el documento latex anterior.

Índice

1 Metapost

- Herramientas básicas
- Tipos de variables
- Texto
- Transformaciones y figuras predefinidas
- *Repetición*
- Intersecciones
- Variables de tipo picture
- Macros

Repetición: bucles for

Los bucles `for` sirven para repetir un conjunto de instrucciones un número determinado de veces.

Hay diferentes posibilidades:

```
for variable = inicio upto fin:  
  instrucciones;  
endfor
```

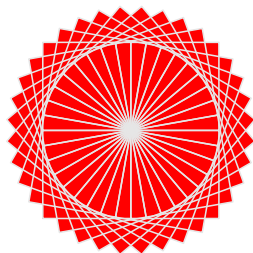
```
for variable = inicio downto fin:  
  instrucciones;  
endfor
```

```
for variable = inicio step paso until fin:  
  instrucciones;  
endfor
```

```
for variable = valor1 , valor2 , ... , valorN:  
  instrucciones;  
endfor
```

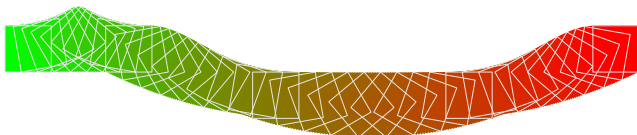
Ejemplos bucles for

```
beginfig(1);  
  numeric angulo;  
  for angulo = 0 step 10 until 350:  
    fill unitsquare rotated angulo  
      scaled 1cm withcolor red;  
  endfor  
  for angulo = 0 step 10 until 350:  
    draw unitsquare rotated angulo  
      scaled 1cm withcolor 0.9white;  
  endfor  
endfig;  
end
```



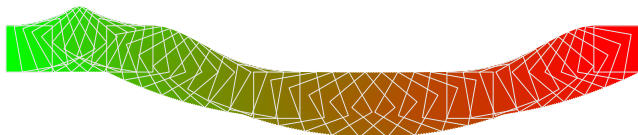
Ejemplos bucles for

```
beginfig(1);  
  numeric x;  
  
  for x=0 upto 360:  
    fill unitsquare scaled 1cm rotated x shifted (x,0)  
      withcolor (x/360)*red + (1-(x/360))*green;  
  endfor;  
  
  for x=0 step 10 until 360:  
    draw unitsquare scaled 1cm rotated x shifted (x,0)  
      withcolor 0.9white;  
  endfor;  
  
endfig;  
end
```



Ejemplos bucles for

```
beginfig(1);  
  numeric x;  
  
  for x=0 upto 360:  
    fill unitsquare scaled 1cm rotated x shifted (x,0)  
      withcolor (x/360)*red + (1-(x/360))*green;  
  endfor;  
  
  for x=0 step 10 until 360:  
    draw unitsquare scaled 1cm rotated x shifted (x,0)  
      withcolor 0.9white;  
  endfor;  
  
endfig;  
end
```



Ejemplos bucles for I

```
beginfig(1);  
  
  pair A,B,C,M,L;  
  path P;  
  
  A:=(0,0);  
  B:=(1cm,1cm);  
  C:=(4cm,0);  
  
  P := A .. B{dir 90} .. C;  
  
  draw P;  
  
  draw point 0 of P withpen pencircle scaled 4bp;  
  draw point 1 of P withpen pencircle scaled 4bp;
```

Ejemplos bucles for II

```
draw point 2 of P withpen pencircle scaled 4bp;

for i=0 step 0.5 until 2:
  M := point i of P;
  L := 0.5cm * unitvector(direction i of P) shifted M;
  drawarrow M — L withcolor red;

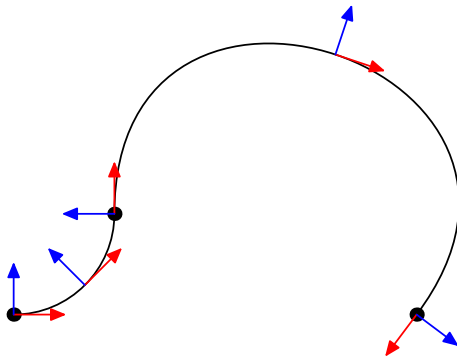
  L := 0.5cm * unitvector(direction i of P)
    rotated 90 shifted M;
  drawarrow M — L withcolor blue;
endfor

endfig;

end
```


Ejemplos bucles for III

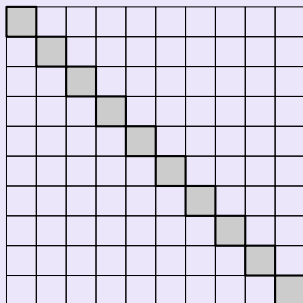
Se genera la siguiente figura:



Ejercicio: bucles for

Tarea 4

Realizar la siguiente figura:



Incluye la figura en el documento latex

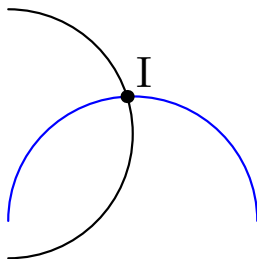
Índice

1 Metapost

- Herramientas básicas
- Tipos de variables
- Texto
- Transformaciones y figuras predefinidas
- Repetición
- *Intersecciones*
- Variables de tipo picture
- Macros

Punto de intersección entre dos curvas

```
beginfig(1);  
path P[];  
P1 := (-1cm,-1cm) .. (0,0) .. (-1cm,1cm);  
P2 := P1 rotated 90 shifted (0,3mm);  
  
draw P1;  
draw P2 withcolor blue;  
  
M := P1 intersectionpoint P2;  
  
dotlabel.urt("I",M);  
  
endfig;  
  
end
```



Camino cerrado entre dos curvas

```
beginfig(1)

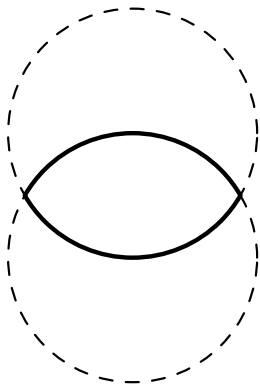
numeric u;
u:=2cm;

% Vector de caminos
path c[];
c[1] := fullcircle scaled u;
c[2] := c[1] shifted (0,.5u);

draw c[1] dashed evenly;
draw c[2] dashed evenly;

draw buildcycle(c[1],c[2]) withpen
    pencircle scaled 1bp;

endfig;
end
```



Ejercicio

Tarea 5

Realizar la siguiente figura:



Incluye la figura en el documento latex

Índice

1 Metapost

- Herramientas básicas
- Tipos de variables
- Texto
- Transformaciones y figuras predefinidas
- Repetición
- Intersecciones
- *Variables de tipo picture*
- Macros

Variables de tipo picture

Estas variables pueden guardar una figura.

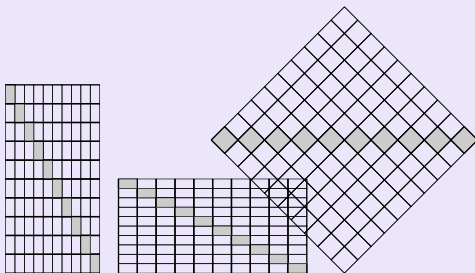
¿Qué valores se le pueden asignar a esta variable?

```
picture pic;  
  
% Instrucciones de dibujo  
  
% Asigna a pic lo que hay dibujado  
pic := currentpicture;  
  
% Dibuja lo que haya en pic  
draw pic;  
  
% Borra lo que haya en pic  
pic := nullpicture;
```


Variables de tipo picture

Tarea 6

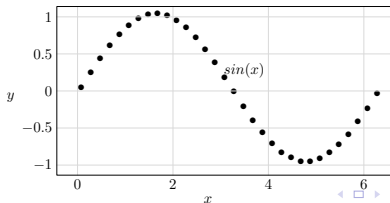
Realizar la siguiente figura:



Incluye la figura en el documento latex.

Gráficas: graph.mp (I)

```
input graph;  
  
beginfig(1);  
  
draw begingraph(8cm,4cm);  
draw begingraph(8cm,4cm);  
  glabel.lft( btex \vbox{\hbox{$y$}} etex , OUT);  
  glabel.bot( btex \vbox{\hbox{$x$}} etex , OUT);  
  gdraw "datos.dat" plot btex $\bullet$ etex;  
  glabel.urrt(btex $\sin(x)$ etex ,15);  
  autogrid(grid.bot,grid.lft) withcolor .85white;  
endgraph;  
endfig;  
end
```



Índice

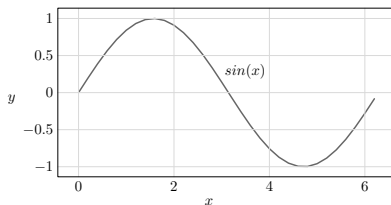
1 Metapost

- Herramientas básicas
- Tipos de variables
- Texto
- Transformaciones y figuras predefinidas
- Repetición
- Intersecciones
- Variables de tipo picture
- *Macros*

Gráficas: graph.mp (II)

```
input graph;
beginfig(1);

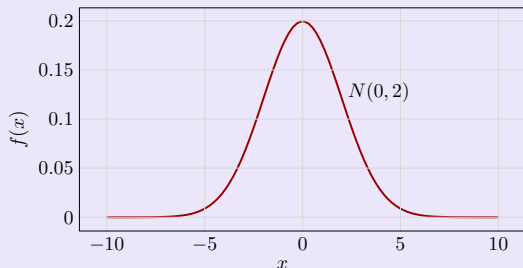
draw begingraph(8cm,4cm);
  glabel.lft( btex \vbox{\hbox{$y$}} etex , OUT);
  glabel.bot( btex \vbox{\hbox{$x$}} etex , OUT);
  gdraw "datos.dat" withpen pencircle scaled 1pt withcolor 0.4
    white;
  glabel.urrt(btex $sin(x)$ etex ,15);
  autogrid(grid.bot,grid.lft) withcolor .85white;
endgraph;
endfig;
end
```



Ejercicio

Tarea 7

Muestra los datos que se encuentran en el fichero exp.dat en una gráfica que sea similar a la que se muestra a continuación:



Incluye la figura en el documento latex.

Cajas: boxes.mp (I)

```
input boxes.mp;

beginfig(1);

% Un nodo identificado como "nodoA" cuya frontera es un retángulo
boxit.nodoA(contenido);

% Posición del centro del nodo
nodoA.c = pair;

% Un nodo identificado como "nodoB" cuya frontera es una elipse
circleit.nombrenodo(contenido);

% Posición del centro del nodo
nodoB.c = pair;

% Quizá algún nodo se deba mostrar con un color de relleno
fill bpath.nodoA withcolor blue;

% Se muestran los nodos
drawboxed(nodoA, nodoB, ..., nodoN)

% Quizá se muestren conexiones entre los nodos...
drawarrow nodoA — nodoB;
```

Cajas: boxes.mp (II) I

Mostrar los puntos cardinales que definen la frontera y los espaciados:

```
input boxes.mp;

beginfig(1) ;

  picture pic;
  pic:=nullpicture;

  fill unitsquare xscaled 2cm yscaled 1cm withcolor (0.7,0.7,0.9)
    ;
  pic := currentpicture;
  currentpicture := nullpicture;

  %Separación entre el contenido y el borde
  defaultdx := 50pt ; defaultdy := 30pt ;

  % Define el nodo "a"
```

Cajas: boxes.mp (II) II

```
boxit.a(pic) ;

% Centro del nodo
a.c = (0,0) ;

% Muestra el nodo "a" dentro de una caja
drawboxed(a)

% Para mostrar algunos atributos de "a"
dotlabel.top ("a.n", a.n) ;
dotlabel.ulft ("a.nw", a.nw) ;
dotlabel.lft ("a.w", a.w) ;
dotlabel.llft ("a.sw", a.sw) ;
dotlabel.bot ("a.s", a.s) ;
dotlabel.lrt ("a.se", a.se) ;
dotlabel.rt ("a.e", a.e) ;
dotlabel.urt ("a.ne", a.ne) ;
dotlabel.top ("a.c", a.c) ;
```

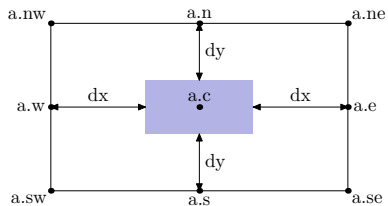

Cajas: boxes.mp (II) III

```
drawdblarrow a.w — a.w shifted( a.dx,0) ;
drawdblarrow a.e shifted(-a.dx,0) — a.e ;
drawdblarrow a.s — a.s shifted(0, a.dy) ;
drawdblarrow a.n shifted(0,-a.dy) — a.n ;
```

```
label.top("dx",a.w shifted ( a.dx/2,0)) ;
label.top("dx",a.e shifted (-a.dx/2,0)) ;
label.rt ("dy",a.s shifted (0, a.dy/2)) ;
label.rt ("dy",a.n shifted (0,-a.dy/2)) ;
```

```
endfig ;
end
```

Cajas: boxes.mp (II) IV



Cajas: boxes.mp (III) I

Nodos unidos:

```
input boxes.mp;

beginfig(1);

  % Separación entre el contenido y el borde
  defaultdx := 6pt ; defaultdy := 6pt ;

  % Defino que las cajas van a estar unidas ¿como?
  boxjoin(a.ne = b.nw; a.se = b.sw);

  % Define el nodo "a"
  boxit.a("0") ;

  % Define el nodo "b"
  boxit.b("1") ;
```

Cajas: boxes.mp (III) II

```
% Define el nodo "c"  
boxit.c("2") ;
```

```
% Define el nodo "d"  
boxit.d("3") ;
```

```
% Centro del nodo  
a.c = (0,0) ;
```

```
drawboxed(a,b,c,d) ;
```

```
endfig ;
```

```
end
```

Cajas: boxes.mp (III) III

0	1	2	3
---	---	---	---

Cajas: boxes.mp (III) I

Diferentes tipos de nodos y uso de comandos latex en metapost:

```
verbatimtex
%&latex
\documentclass{article}

\usepackage{xspace, colortbl}
\usepackage{color}
\usepackage{hyperref}
\usepackage[latin1]{inputenc}
\usepackage[spanish]{babel}

\begin{document}
etex

beginfig(1) ;

    % Separación entre el contenido y el borde
```

Cajas: boxes.mp (III) II

```
defaultdx := 6pt ; defaultdy := 6pt ;

% Defino que las cajas van a estar unidas ¿como?
boxjoin(a.ne = b.nw; a.se = b.sw);

% Define el nodo "a"
boxit.a("s[0]");
% Define el nodo "b"
boxit.b("s[1]");
% Define el nodo "c"
boxit.c("s[2]");
% Define el nodo "d"
boxit.d("s[3]");

% Centro del nodo
a.c = (0,0) ;

drawboxed(a,b,c,d);
```

Cajas: boxes.mp (III) III

```
boxjoin ();

boxit .m("s");
m.c = (-1.5cm, 0);
drawunboxed(m);
drawarrow m.e — a.w;

circleit .r("Cad1");
r.c = (-2cm,-2cm);

circleit .s("Cad2");
s.c = (0cm,-2cm);

circleit .t("Cad3");
t.c = (2cm,-2cm);

circleit .u("Cad4");
u.c = (4cm,-2cm);
```


Cajas: boxes.mp (III) IV

```
fill bpath.r withcolor (0.7,0.7,0.9);
fill bpath.s withcolor (0.7,0.7,0.9);
fill bpath.t withcolor (0.7,0.7,0.9);
fill bpath.u withcolor (0.7,0.7,0.9);

drawboxed(r,s,t,u);

drawarrow a.s{down} .. r.n{down};
drawarrow b.s{down} .. s.n{down};
drawarrow c.s{down} .. t.n{down};
drawarrow d.s{down} .. u.n{down};

picture p;

p:=thelabel(btex
    \begin{minipage}[t]{4cm}
    \sffamily \bfseries \small
    Esta es una ilustración
```

Cajas: boxes.mp (III) V

```
        de un array de referencias  
        del tipo String  
        \end{minipage}  
        etex ,(2cm,2cm));
```

```
fill bbox p withcolor .6green+0.2blue+0.9red;
```

```
draw p;
```

```
pickup pencircle scaled .5;
```

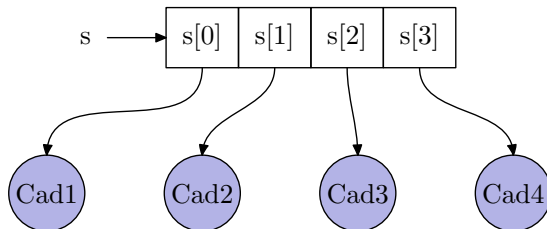
```
draw bbox p;
```

```
endfig ;
```

```
end
```

Cajas: boxes.mp (III) VI

Esta es una ilustración de un array de referencias del tipo String



Inclusión de imágenes: metafun.mp

```
input metafun.mp;

beginfig(1);

  externalfigure "imagen.ext" xscaled numeric yscaled numeric
    shifted pair;

  %Otras instrucciones

endfig;

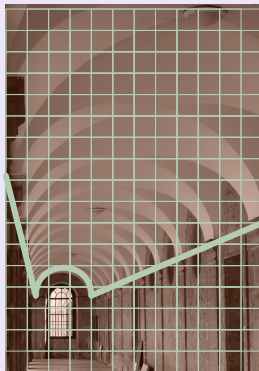
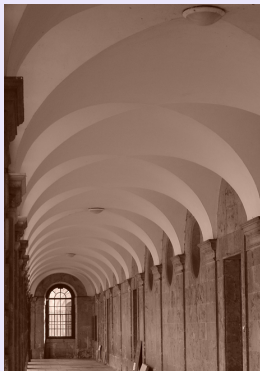
end
```

Este fichero se procesa con `mpost` y el resultado se procesa con `mptopdf` para generar un fichero pdf que se puede incluir en el documento.

Ejercicio

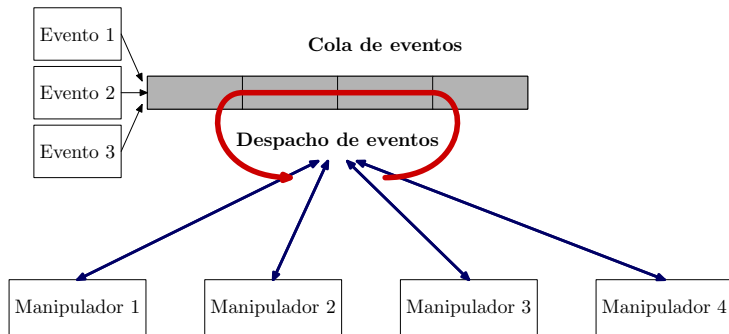
Tarea 8

Haz una figura similar a la que se muestra a continuación

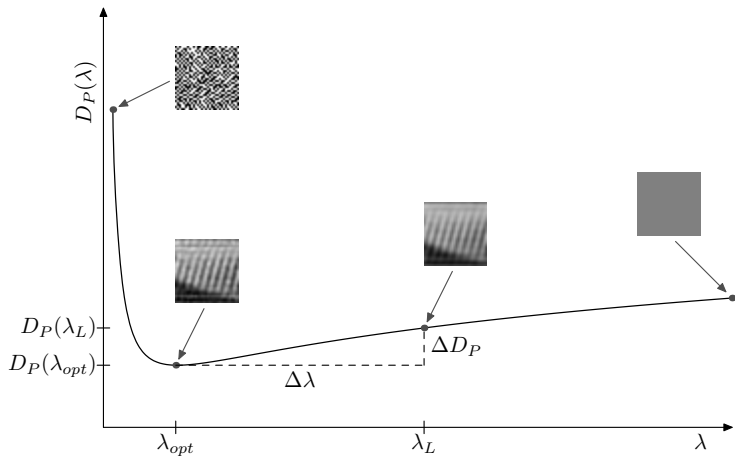


Incluye la figura en el documento latex.

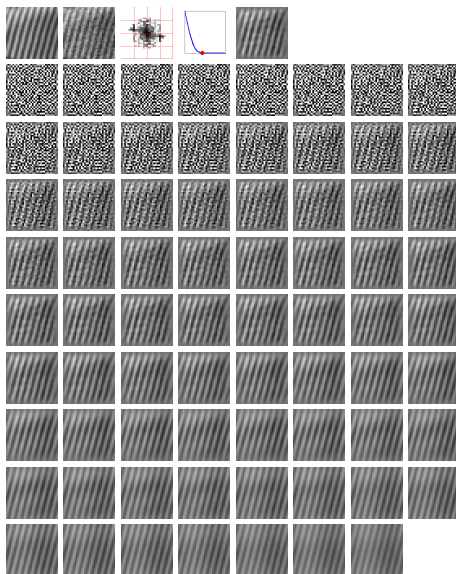
Ejemplos



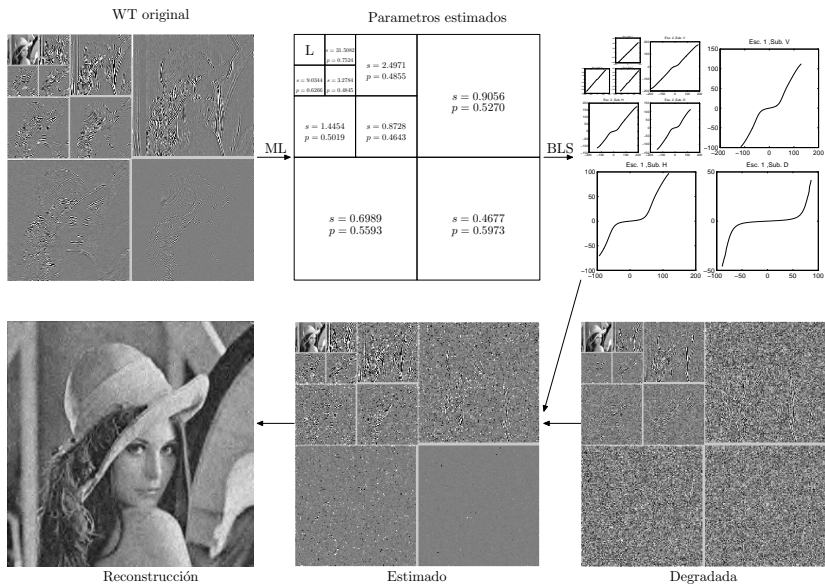
Ejemplos



Ejemplos



Ejemplos



Ejemplos

Índice

- 1 Metapost
- 2 Un listado de otras herramientas

graphviz (para grafos) I

```
digraph arbol {
  label="Un árbol que representa la estructura\n de éste capítulo
  ";
  node [shape=ellipse ];
  Arboles;
  Introduccion;
  Term_Basica [label="Terminologia \n Basica "];
  Arb_Bin [label = "Arboles \n Binarios "];
  Esp_Alg [label = "Especificacion \n Algebraica "];
  Ejemplos [label = "Recorridos "];
  Rep [label = "Representacion \n de Arboles \n Binarios "];
  Med_Ar [label = "Mediante \n Arrays "];
  Med_Est [label = "Mediante \n Estructuras \n Enlazadas "];
  Arb_Bin_Bus [label = "Arboles \n Binarios \n de Busqueda "];
  Heaps;
```

graphviz (para grafos) II

```
Ord_Arb [label="Ordenacion \n con Arboles"];  
  
Arboles -> Introduccion;  
Arboles -> Term_Basica;  
Arboles -> Arb_Bin;  
Arboles -> Arb_Bin_Bus;  
Arb_Bin -> Esp_Alg  
Arb_Bin -> Ejemplos;  
Arb_Bin -> Rep;  
Rep -> Med_Ar;  
Rep -> Med_Est;  
Arboles -> Heaps;  
Arboles -> Ord_Arb;  
}
```

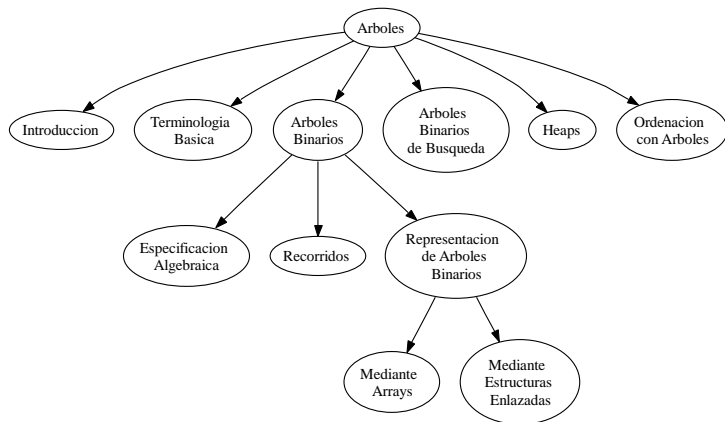
graphviz (para grafos) III

El fichero se procesa del siguiente modo:

```
dot -Tps -o nombre.eps nombre.dot
```

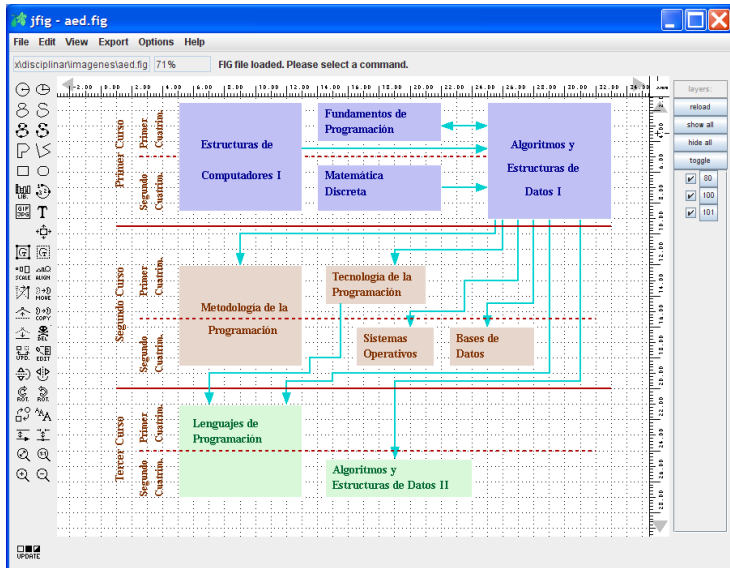
El fichero eps se puede utilizar en un fichero fuente para ser procesado con Latex o se puede transformar a pdf para incluirlo en un fichero para ser procesado con pdflatex.

graphviz (para grafos) IV



Un árbol que representa la estructura de éste capítulo

jfig (shareware)



Para realizar conversiones entre formatos:

epstopdf: para convertir eps a pdf

imagemagick: conjunto de utilidades para procesar imágenes.