Documentación Proyecto *DSL*

Integrantes: Isleño Gabriel, Mellano Franco y Muntané Héctor.

Descripción del proyecto

Éste proyecto consiste en crear un *DSL* cuya principal función es la generación de figuras geométricas en un plano, gráficos de torta y lineales. Ésta implementación nos permitirá una fácil creación de estos dibujos y su exportación a *PDF.*

Como utilizar DSL

Dependencias:

- Compilador de *Haskell*. *Glasgow Haskell Compiler*, versión 8.0.2.

```
$sudo apt install ghci
```

- Librería de *LaTeX* para *Haskell*. "*Text.LaTeX*".

```
$cabal install HaTeX
```

- Librería utilizada para el parseo de números flotantes. "*Text.Parsec.Number*"

```
$cabal install parsec3-numbers
```

- Paquete TeX Live para permitir convertir archivos TeX a PDF

```
$sudo apt install texlive-latex-recommended
```

Aclaración: En el caso de que al compilar el código del *DSL* muestre un error *"LaTeX Error: File `tikz.sty` not found."* ejecutar además el siguiente comando:

```
$sudo apt-get install -y tikzit
```

- Módulos con código fuente del *DSL (AST.hs, Parser.hs, Eval.hs y Main.hs).*

Aclaraciones para el código:

- Los comandos de nuestro código deberán esta separados por “ **;** ”. Este punto y coma no deberá ser incluido luego de la ultima sentencia del código.
- Como nuestro lenguaje utiliza números flotantes, todos los valores numéricos deberán incluir un numero después del punto a pesar de que sean números enteros.
- El color por defecto de todas las figuras es negro.

Listado de comandos:

- **Punto**:

```
Punto(x,y)
```

Utilizado para determinar la ubicación de las figuras geométricas en el plano.

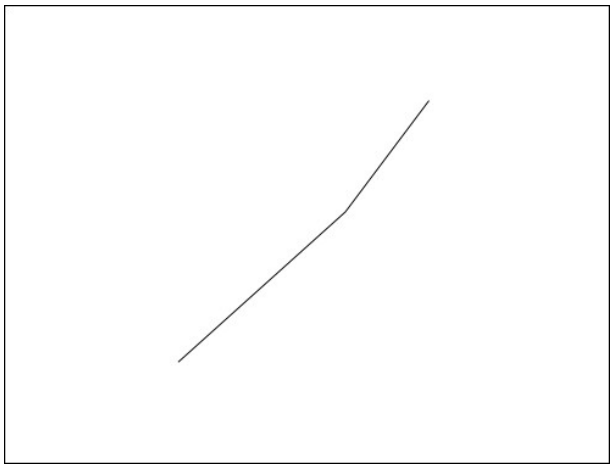
Ejemplo: `Punto(0.0,3.0)`

- **Linea**:

```
Linea[Punto(x,y),Punto(x,y), … n Punto]
```

Dibuja una linea contigua. Se debe establecer un listado de 2 a *n* puntos.

Ejemplo: `Linea[Punto(0.0,1.3), Punto(3.0,4.0), Punto(4.5,6.0)]`

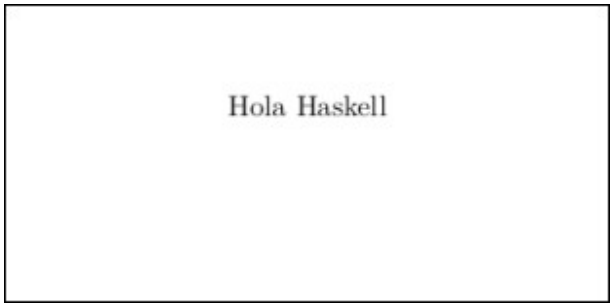


- **Texto**:

```
Texto(Punto(x,y), "texto")
```

Dibuja un texto. Se debe establecer la posición y el texto.

Ejemplo: `Texto(Punto(0.0, 0.0),"Hola Haskell")`

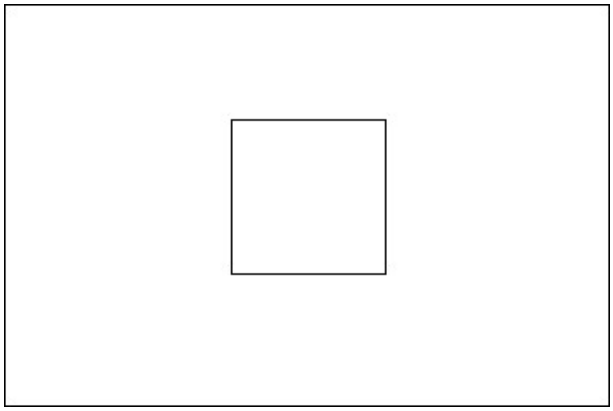


- **Cuadrado**:

```
Cuadrado(Punto(x,y), ladoCuadrado)
```

Dibuja un cuadrado. Se debe establecer la posición y tamaño del los lados del cuadrado.

Ejemplo: `Cuadrado(Punto(0.0,0.0), 2.0)`

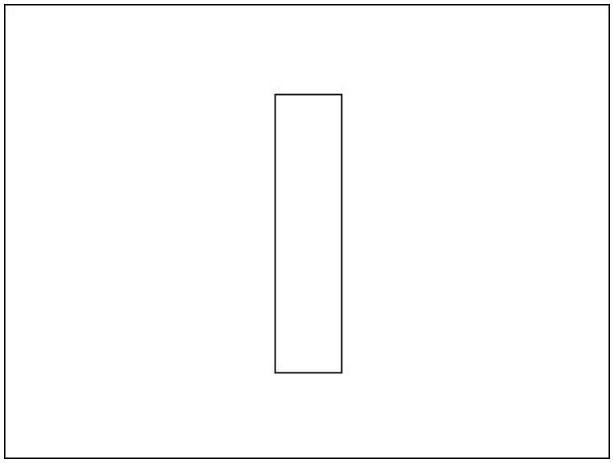


- **<u>Rectángulo</u>**:

```
Rectangulo(Punto(x,y), baseRectangulo, alturaRectangulo)
```

Dibuja un rectángulo. Se debe establecer la posición y tamaño de la base y altura del rectángulo.

Ejemplo: `Rectangulo(Punto(0.0,0.0), 1.0, 4.2)`

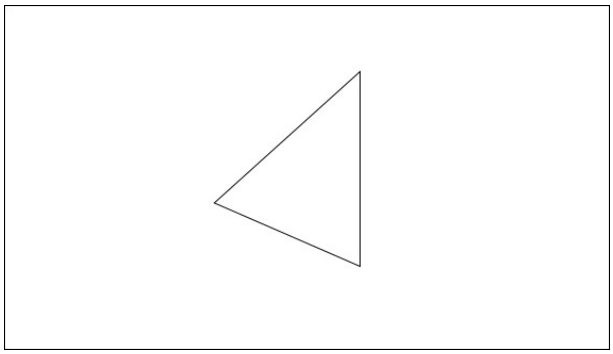


- **<u>Polígono</u>**:

```
Poligono[Punto(x,y),Punto(x,y), … n Punto]
```

Dibuja un polígono. Se debe establecer un listado de 2 a *n* puntos. Si en el ultimo punto del listado no se vuelve al origen se conectaran el primer y ultimo punto automáticamente.

Ejemplo: `Poligono[Punto(0.0,1.3), Punto(3.0,4.0), Punto(3.0,0.0)]`

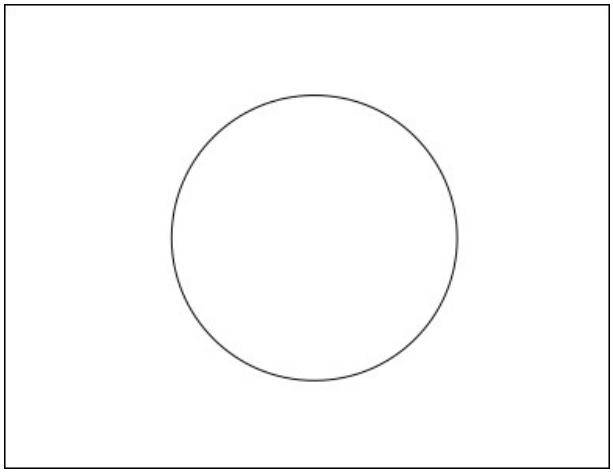


- **Circulo**:

```
Circulo(Punto(x,y), radioCirculo)
```

Dibuja un circulo. Se debe establecer la posición y el radio del circulo.

Ejemplo: `Circulo(Punto(0.0,0.0), 2.0)`

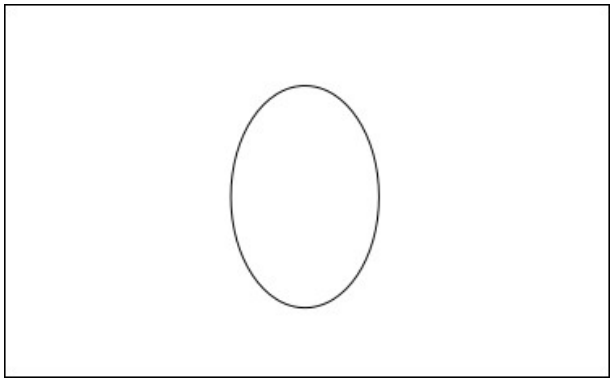


- **Elipse**:

```
Elipse(Punto(x,y), anchoElipse, alturaElipse)
```

Dibuja un elipse. Se debe establecer la posición y tamaño del ancho y altura del elipse.

Ejemplo: `Elipse(Punto(0.0,0.0), 2.0, 3.0)`



- **Dato**:

```
Dato(porcentaje, "descripcion")
```

Utilizado para listar los elementos que compondrán un grafico de torta.

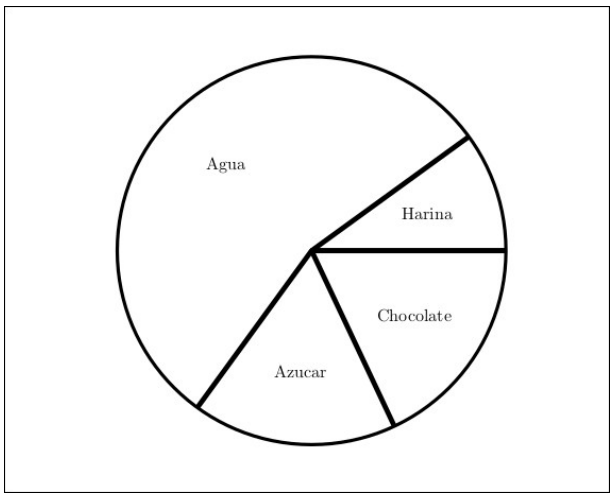
Ejemplo: `Dato(25.0, "Harina")`

- **GraficoTorta**:

```
GraficoTorta[Dato(porcentaje, "descripcion"),… n Dato]
```

Dibuja un grafico de torta. Se debe establecer un listado de 1 a *n* datos. El porcentaje total de todos los datos listados debe sumar cien (100.0).

Ejemplo: `GraficoTorta[Dato(10.0,"Harina"), Dato(55.0,"Agua"), Dato(17.0,"Azucar"), Dato(18.0,"Chocolate")]`

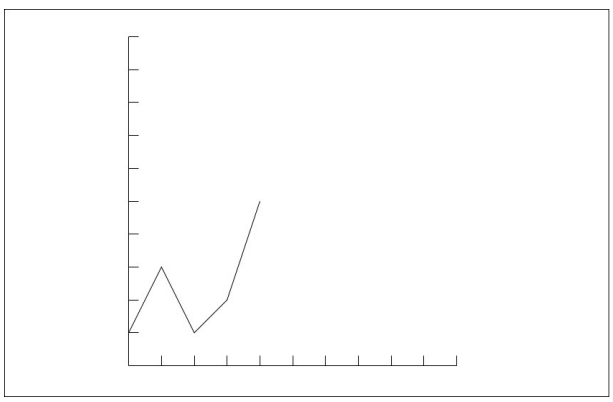


- **GraficoLinea**:

```
GraficoLinea([a,b,c,d, ...])
```

Dibuja un grafico de linea. Se debe establecer un listado de 2 a *n* números.

Ejemplo: `GraficoLinea([1.0,3.0,1.0,2.0,5.0])`



- **Pintado**:

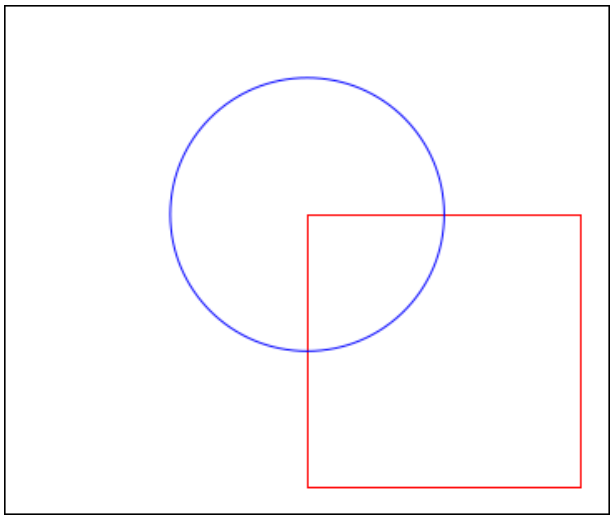
```
Pintado “color” forma
```

Modifica el color de una forma geométrica o gráfico. Los colores disponibles son:

Rojo	Azul	Amarillo	Verde	Cian	Fucsia	Negro	Blanco

Ejemplo:

```
Pintado "Rojo" Cuadrado (Punto (0.0, 0.0), 3.0);
Pintado "Azul" Circulo (Punto (0.0, 0.0), 1.5)
```



Instrucciones de uso:

1. Guardar código en archivo de texto plano (Ej: codigodsl).
2. Compilar código fuente del *DSL* mediante el comando:

```
$ghc -o exe Main.hs
```

3. Utilizar el ejecutable para compilar nuestro código del *DSL*:

```
$./exe codigodsl
```

4. Se generará el archivo PDF del dibujo con el mismo nombre que el archivo con el código. (Ej: codigodsl.pdf)

Organización de los archivos

```
/HaskellDSL
        ├── AST.hs
        ├── Eval.hs
        ├── Main.hs
        └── Parser.hs
```

Bibliografia

Latex	https://www.latex-project.org
Tikz	https://www.overleaf.com/learn/latex/TikZ_package
Haskell	https://www.haskell.org
Parsec	https://hackage.haskell.org/package/parsec
Hatex	https://hackage.haskell.org/package/HaTeX
Text.Parsec.Number	https://hackage.haskell.org/package/parsec3-numbers-0.1.0/docs/Text-Parsec-Number.html
Intro to parsing	http://jakewheat.github.io/intro_to_parsing
Scheme in 48hs	https://en.wikibooks.org/wiki/Write_Yourself_a_Scheme_in_48_Hours