



PROGRAMA PARA CALCULAR EL ÁREA BAJO LA CURVA DE UNA FUNCIÓN

Documentación técnica

I.C. Miguel Ángel Mejía Ballina

PROBLEMA

Se requiere crear una gráfica de una función, y calcular el área bajo la curva de la misma.

SOLUCIÓN

Se propone desarrollar un programa que grafique la función y que calcule el área bajo la curva per medio de la suma de Riemman.

ANÁLISIS

Determinación de objetos.

Clase *cl_sumaRiemman* en la Figura1.

cl_sumaRiemman
<div>limiteInferior: REAL limiteSuperior: REAL funcion: cl_plotreal numeroRectangulos: ENTERO areaRiemman: REAL areaIntegral: REAL error: REAL rectangulo: cl_plotrectangulo</div>
<div>crear(xorig, yorig: ENTERO; limInf, limSup, areaAnalitica:REAL; noRectangulos, prec:ENTERO; salida: IMAGEN) setlimiteInferior(n: REAL) setlimiteSuperior(n: REAL) setfuncion(f: cl_plotreal) setnumeroRectangulos(n: ENTERO) setareaRiemman(n: REAL) setareaIntegral(n: REAL) seterror(n: REAL) setRectangulo(r: cl_plotrectangulo) getlimiteInferior: REAL getlimiteSuperior: REAL getfuncion: cl_plotreal getnumeroRectangulos: ENTERO getareaRiemman: REAL getareaIntegral: REAL geterror: REAL getRectangulo: cl_plotrectangulo calcularAreaRiemman: REAL calcularError: REAL</div>

Figura 1



El cálculo del área bajo la curva será realizado por medio de suma de Riemman, representada de la siguiente manera:

$$\sum_{i=inicio}^{i=fin} (x_i - x_{i-1}) * f(x_i)$$

DISEÑO DE INTERFAZ

ÁREA BAJO LA CURVA

Grafica1

600 x 600

LÍMITES X

Inferior

Superior

Precision

SUMA DE RIEMMAN

Número de rectángulos

Resultado analítico

CREAR GRÁFICA

Área: <<area>>

Error: <<error>>



DIAGRAMA DE CLASES

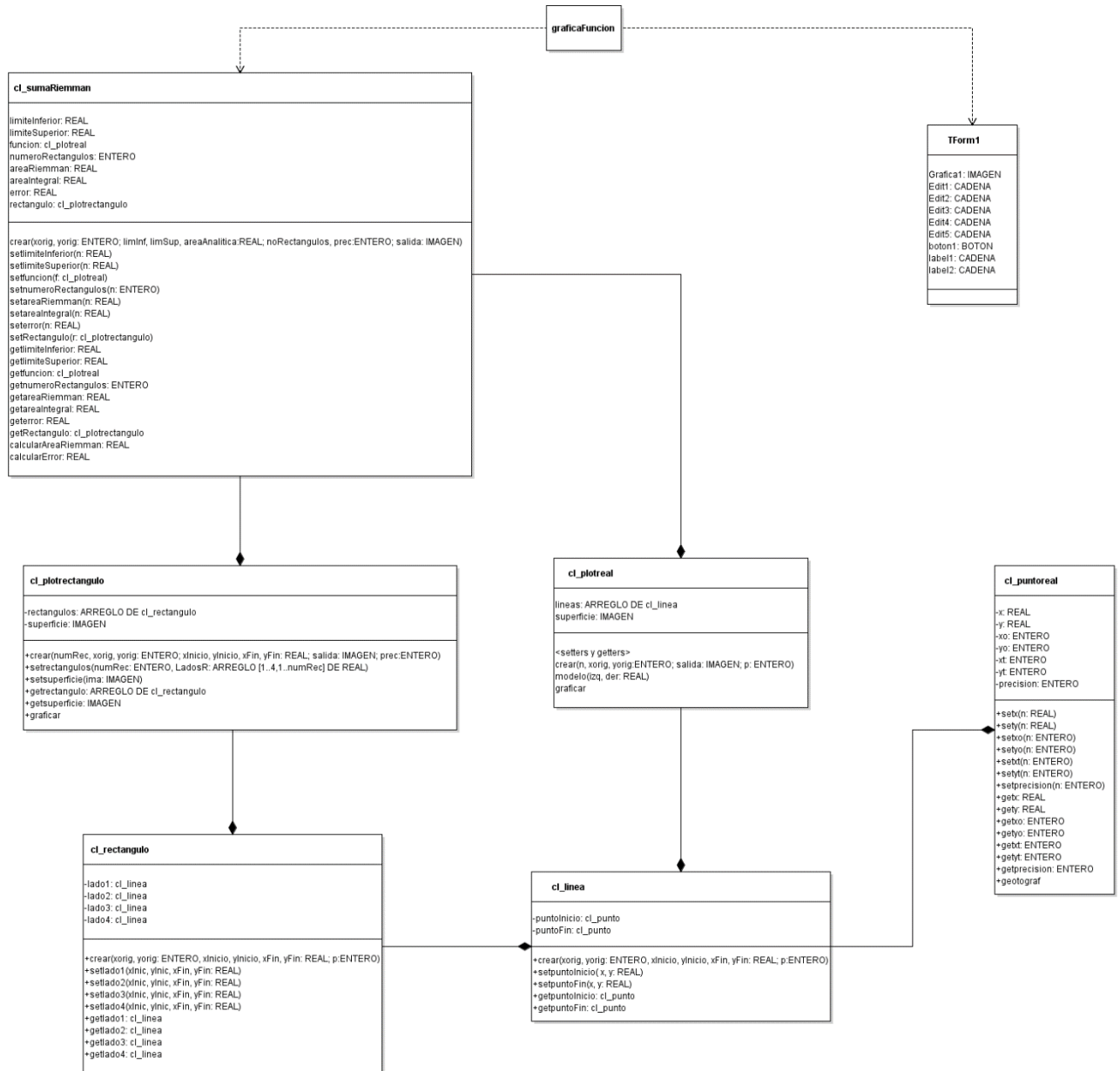
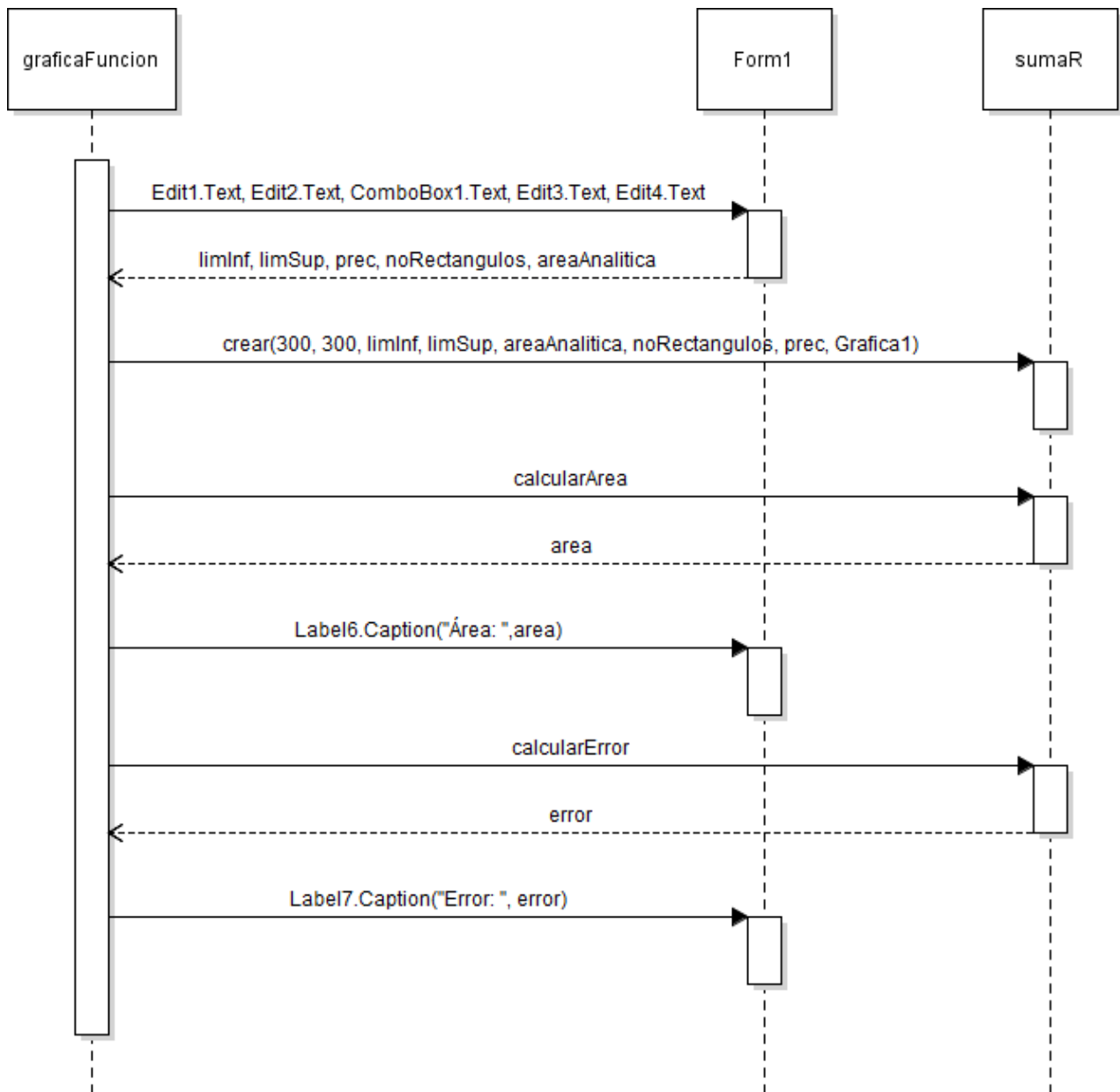


DIAGRAMA DE SECUENCIA



PSEUDOCÓDIGO

CLASE **cl_plotreal**

INICIO

ATRIBUTOS

lineas: ARREGLO DE **cl_linea**



Privada 16 de Septiembre No. 101.

Zinacantepec, Estado de México.

C.P. 51350



www.mediacod.com



contacto@mediacod.com



7226741918



facebook.com/mediacod



[@mediacod](https://twitter.com/mediacod)

superficie: IMAGEN

MÉTODOS

```
MÉTODO cl_plotreal(n, xorig, yorig:ENTERO; salida:IMAGEN;
p:ENTERO)
    VARIABLES
        i: ENTERO
    INICIO
        superficie ← salida
        PARA i ← 1 HASTA n HACER
            lineas ← cl_linea.crear(xorig, yorig, 0, 0, 0, 0, p)
        FIN PARA
    FIN MÉTODO cl_plotreal

MÉTODO setlineas(n:ENTERO; ListaP: ARREGLO[1..4, 1..n] DE
REAL)
    VARIABLES
        i: ENTERO
    INICIO
        PARA i←1 HASTA n HACER
            lineas[i].setpuntoInicio(ListaP[1,i], ListaP[2,i])
            lineas[i].setpuntoFin(ListaP[3,i], ListaP[4,i])
        FIN PARA
    FIN MÉTODO setlineas

MÉTODO setsuperficie(ima: IMAGEN)
    INICIO
        superficie←ima
    FIN MÉTODO setsuperficie

MÉTODO getlineas: ARREGLO DE cl_linea
    INICIO
        Regresa lineas
    FIN MÉTODO getlineas

MÉTODO getsuperficie: IMAGEN
    INICIO
        Regresa superficie
    FIN MÉTODO getsuperficie
```



```
MÉTODO modelo(izq, der: REAL)
  FUNCION f(n: REAL): REAL
  INICIO
    REGRESA <<expresion matemática>>
  FIN FUNCION f

VARIABLES
  i, c, p:ENTERO
  vx: REAL
INICIO
  c ← 0
  p ← lineas[1].getpuntoInicio.getprecision

  PARA i ← 1 HASTA 1+(der-izq)*p HACER
    vx ← izq +(i-1)/prec
    lineas[c].setpuntoInicio(vx, f(vx))
    lineas[c].setpuntoFin((izq+(i)/p), f(izq+(i)/p))

    c ← c+1
  FIN PARA
FIN MÉTODO modelo

MÉTODO graficar
VARIABLES
  i, n, alto, ancho, xCero, yCero: ENTERO
  cx, cy, cx1, cy1: ENTERO
INICIO
  alto ← superficie.alto
  ancho ← superficie.ancho

  xCero ← lineas[1].getpuntoInicio.getxo
  yCero ← lineas[1].getpuntoInicio.getyo

  DibujaLinea(0, yCero, ancho, yCero)
  DibujaLinea(xCero, 0, xCero, alto)
  n ← Tamaño(puntos)
  PARA i ← 1 HASTA n HACER
    cx ← lineas[i].getpuntoInicio.getxt
```



```
cy ← lineas[i].getpuntoInicio.getyt  
cx1 ← lineas[i].getpuntoFin.getxt  
cy1 ← lineas[i].getpuntoFin.getyt  
DibujaLinea(cx,cy, cx1, cy1)
```

FIN PARA

FIN MÉTODO graficar

FIN CLASE cl_plotreal

CLASE cl_puntoreal

INICIO

ATRIBUTOS

x: REAL

y: REAL

xo: ENTERO

yo: ENTERO

xt: ENTERO

yt: ENTERO

precision: ENTERO

MÉTODOS

MÉTODO setx(n:REAL)

INICIO

x ← n

FIN MÉTODO setx

MÉTODO sety(n:REAL)

INICIO

y ← n

FIN MÉTODO sety

MÉTODO setxo(n:ENTERO)

INICIO

xo ← n

FIN MÉTODO setxo

MÉTODO setyo(n:ENTERO)

INICIO

yo ← n

FIN MÉTODO setyo





MÉTODO setxt(n:ENTERO)
INICIO

xt ← n

FIN MÉTODO setxt

MÉTODO setyt(n:ENTERO)
INICIO

yt ← n

FIN MÉTODO setyt

MÉTODO setprecision(n:ENTERO)
INICIO

precision ← n

FIN MÉTODO setprecision

MÉTODO getx: REAL
INICIO

Regresa x

FIN MÉTODO getx

MÉTODO gety: REAL
INICIO

Regresa y

FIN MÉTODO gety

MÉTODO getxo: ENTERO
INICIO

Regresa xo

FIN MÉTODO getxo

MÉTODO getyo: ENTERO
INICIO

Regresa yo

FIN MÉTODO getyo

MÉTODO getxt: ENTERO
INICIO

Regresa xt

FIN MÉTODO getxt



Privada 16 de Septiembre No. 101.

Zinacantepec, Estado de México.

C.P. 51350



www.mediacod.com



contacto@mediacod.com



7226741918



facebook.com/mediacod



[@mediacod](https://twitter.com/mediacod)

MÉTODO getyt: ENTERO

INICIO

Regresa yt

FIN MÉTODO getyt

MÉTODO getprecision: ENTERO

INICIO

Regresa precision

FIN MÉTODO getprecision

MÉTODO geotograf

INICIO

$xt \leftarrow xo + (x * precision)$

$yt \leftarrow yo - (y * precision)$

FIN MÉTODO geotograf

MÉTODO cl_puntoreal(cx,cy: REAL; crx,cry,p:ENTERO)

INICIO

$x \leftarrow cx$

$y \leftarrow cy$

$xo \leftarrow crx$

$yo \leftarrow cry$

$precision \leftarrow p$

FIN MÉTODO cl_puntoreal

FIN CLASE cl_puntoreal

CLASE cl_linea

INICIO

ATRIBUTOS

puntoInicio: cl_puntoreal

puntoFin: cl_puntoreal

MÉTODOS


MÉTODO cl_linea(xorig, yorig:ENTERO; xInicio, yInicio, xFin,
yFin:REAL; p:ENTERO)

INICIO

puntoInicio \leftarrow cl_puntoreal.crear(xInicio, yInicio, xorig, yorig,
p);

 www.mediacod.com

 contacto@mediacod.com

 7226741918

 facebook.com/mediacod

 @mediacod



Privada 16 de Septiembre No. 101.

Zinacantepec, Estado de México.

C.P. 51350

```
puntoFin ← cl_puntoreal.crear(xFin, yFin, xorig, yorig, p);  
FIN MÉTODO cl_linea
```

```
MÉTODO setpuntoInicio(x, y:REAL)  
INICIO  
    puntoInicio.setx(x)  
    puntoInicio.sety(y)  
    puntoInicio.geotograf  
FIN MÉTODO setpuntoInicio
```

```
MÉTODO setpuntoFin(x, y:REAL)  
INICIO  
    puntoFin.setx(x)  
    puntoFin.sety(y)  
    puntoFin.geotograf  
FIN MÉTODO setpuntoFin
```

```
MÉTODO getpuntoInicio: cl_puntoreal  
INICIO  
    Regresa puntoInicio  
FIN MÉTODO getpuntoInicio
```

```
MÉTODO getpuntoFin: cl_puntoreal  
INICIO  
    Regresa puntoFin  
FIN MÉTODO getpuntoFin
```

FIN CLASE cl_linea

CLASE cl_rectangulo

```
INICIO  
    ATRIBUTOS  
        lado1: cl_linea  
        lado2: cl_linea  
        lado3: cl_linea  
        lado4: cl_linea
```

MÉTODOS

```
MÉTODO cl_rectangulo(xorig, yorig:ENTERO; xInicio,  
yInicio, xFin, yFin: REAL; p: ENTERO)
```



```
INICIO
    lado1 ← cl_linea.crear(xorig, yorig, xInicio, yInicio, xFin,
yInicio, p)
    lado2 ← cl_linea.crear(xorig, yorig, xFin, yInicio, xFin, yFin, p)
    lado3 ← cl_linea.crear(xorig, yorig, xFin, yFin, xInicio, yFin, p)
    lado4 ← cl_linea.crear(xorig, yorig, xInicio, yFin, xInicio,
yInicio, p)
```

FIN MÉTODO cl_rectangulo

MÉTODO setlado1(xInic, yInic, xFin, yFin: REAL)

INICIO

lado1.setpuntoInicio(xInic, yInic)

lado1.setpuntoFin(xFin, yInic)

FIN MÉTODO setlado1

MÉTODO setlado2(xInic, yInic, xFin, yFin: REAL)

INICIO

lado2.setpuntoInicio(xInic, yInic)

lado2.setpuntoFin(xInic, yInic)

FIN MÉTODO setlado2

MÉTODO setlado3(xInic, yInic, xFin, yFin: REAL)

INICIO

lado3.setpuntoInicio(xFin, yFin)

lado3.setpuntoFin(xInic, yFin)

FIN MÉTODO setlado3

MÉTODO setlado4(xInic, yInic, xFin, yFin: REAL)

INICIO

lado4.setpuntoInicio(xInic, yFin)

lado4.setpuntoFin(xInic, yInic)

FIN MÉTODO setlado4

MÉTODO getlado1: cl_linea

INICIO

Regresa lado1

FIN MÉTODO getlado1





MÉTODO getlado2: cl_linea
INICIO

Regresa lado2

FIN MÉTODO getlado2

MÉTODO getlado3: cl_linea
INICIO

Regresa lado3

FIN MÉTODO getlado3

MÉTODO getlado4: cl_linea
INICIO

Regresa lado4

FIN MÉTODO getlado4

FIN CLASE cl_rectangulo

CLASE cl_plotrectangulo

INICIO

ATRIBUTOS

rectangulos: ARREGLO DE cl_rectangulo

superficie: IMAGEN

MÉTODOS

MÉTODO cl_plotrectangulo(numRec, xorig, yorig:ENTERO; xInicio, yInicio, xFin, yFin:REAL; salida:IMAGEN; p:ENTERO)

INICIO

superficie ← salida

PARA i ← 1 HASTA numRec HACER

rectangulos[i] ← cl_rectangulo.crear(xorig, yorig, xInicio, yInicio, xFin, yFin, p)

FIN PARA

FIN MÉTODO cl_plotRectangulo

MÉTODO setrectangulos(n, LadosR: ARREGLO [1..4,1..n] DE REAL)

VARIABLES

i: ENTERO

INICIO

PARA i <- 1 HASTA n HACER



Privada 16 de Septiembre No. 101.

Zinacantepec, Estado de México.

C.P. 51350



www.mediacod.com



contacto@mediacod.com



7226741918



facebook.com/mediacod



[@mediacod](https://twitter.com/mediacod)

```
        rectangulos[i].setlado1(LadosR[i,1], LadosR[i,2],
LadosR[i,3], LadosR[i,4])
        rectangulos[i].setlado2(LadosR[i,1], LadosR[i,2],
LadosR[i,3], LadosR[i,4])
        rectangulos[i].setlado3(LadosR[i,1], LadosR[i,2],
LadosR[i,3], LadosR[i,4])
        rectangulos[i].setlado4(LadosR[i,1], LadosR[i,2],
LadosR[i,3], LadosR[i,4])
```

```
        FIN PARA
    FIN MÉTODO setrectangulos
```

```
MÉTODO setsuperficie(ima: IMAGEN)
```

```
INICIO
```

```
    superficie ← ima
```

```
FIN MÉTODO setsuperficie
```

```
MÉTODO getrectangulos: ARREGLO DE cl_rectangulo
```

```
INICIO
```

```
    Regresa rectangulos
```

```
FIN
```

```
MÉTODO getsuperficie: IMAGEN
```

```
INICIO
```

```
    Regresa superficie
```

```
FIN MÉTODO getsuperficie
```

```
MÉTODO graficar
```

```
VARIABLES
```

```
    i: ENTERO
```

```
    n: ENTERO
```

```
    cy, cy, cx1, cy1: ENTERO
```

```
    alto, ancho, xCero, yCero: ENTERO
```

```
INICIO
```

```
    alto ← superficie.alto
```

```
    ancho ← superficie.ancho
```

```
    xCero ← rectangulo[1].getlado1.getpuntoInicio.getxo
```

```
    yCero ← rectangulo[1].getlado1.getpuntoInicio.getyo
```

```
    DibujaLinea(0, yCero, ancho, yCero)
```



DibujaLinea(xCero, 0, xCero, alto)

$n \leftarrow \text{Tamaño}(\text{rectangulos})$

PARA $i \leftarrow 1$ HASTA n HACER

$cx \leftarrow \text{rectángulo}[i].\text{getlado1}.\text{getpuntoInicio}.\text{getxt}$

$cy \leftarrow \text{rectángulo}[i].\text{getlado1}.\text{getpuntoInicio}.\text{getyt}$

$cx1 \leftarrow \text{rectángulo}[i].\text{getlado2}.\text{getpuntoFin}.\text{getxt}$

$cy1 \leftarrow \text{rectángulo}[i].\text{getlado2}.\text{getpuntoFin}.\text{getyt}$

DibujaRectangulo(cx,cy,cxq,cy1)

FIN PARA

FIN MÉTODO graficar

FIN CLASE cl_plotrectangulo

CLASE cl_sumaRiemman

INICIO

ATRIBUTOS

limiteInferior: REAL

limiteSuperior: REAL

funcion: cl_plotreal

numeroRectangulos: ENTERO

areaRiemman: REAL

areaIntegral: REAL

error: REAL

rectangulo: cl_plotrectangulo

MÉTODOS

MÉTODO cl_sumaRiemman(xorig, yorig:ENTERO; limInf, limSup,
areaAnalitica:REAL; noRectangulos, prec:ENTERO; salida:IMAGEN)

VARIABLES

numeroObjetos: ENTERO

INICIO

limiteInferior \leftarrow limInf

limiteSuperior \leftarrow limSup

numeroRectangulos \leftarrow noRectangulos

areaIntegral \leftarrow areaAnalitica

numeroObjetos \leftarrow (limiteSuperior-limiteInferior)*prec + 1

funcion \leftarrow cl_plotreal.crear(numeroObjetos, xorig, yorig, salida,
prec)



```
funcion.modelo(limitesInferior, limitesSuperior)
rectangulo ← cl_plotrectangulo.crear(numeroRectangulos,
xorig, yorig, 0, 0, 0, 0, salida, prec)
FIN MÉTODO cl_sumaRiemman
```

```
MÉTODO setLimitesInferior(n: REAL)
INICIO
    limitesInferior ← n
FIN MÉTODO setLimitesInferior
```

```
MÉTODO setLimitesSuperior(n: REAL)
INICIO
    limitesSuperior ← n
FIN MÉTODO setLimitesSuperior
```

```
MÉTODO setFuncion(f: cl_plotREAL)
INICIO
    funcion ← f
FIN MÉTODO setFuncion
```

```
MÉTODO setNumeroRectangulos(n: ENTERO)
INICIO
    numeroRectangulos ← n
FIN MÉTODO setNumeroRectangulos
```

```
MÉTODO setAreaRiemman(n: REAL)
INICIO
    areaRiemman ← n
FIN MÉTODO setAreaRiemman
```

```
MÉTODO setAreaIntegral(n: REAL)
INICIO
    areaIntegral ← n
FIN MÉTODO setAreaIntegral
```

```
MÉTODO setError(n: REAL)
INICIO
    error ← n
FIN MÉTODO setError
```


MÉTODO setRectangulo(r: cl_plotrectangulo)

INICIO

 rectangulo \leftarrow r

FIN MÉTODO setRectangulo

MÉTODO getLimiteInferior: REAL

INICIO

 Regresa limiteInferior

FIN MÉTODO getLimiteInferior

MÉTODO getLimiteSuperior: REAL

INICIO

 Regresa limiteSuperior

FIN MÉTODO getLimiteSuperior

MÉTODO getFuncion: cl_plotREAL

INICIO

 Regresa funcion

FIN MÉTODO getFuncion

MÉTODO getNumeroRectangulos: ENTERO

INICIO

 Regresa numeroRectangulos

FIN MÉTODO getNumeroRectangulos

MÉTODO getAreaRiemman: REAL

INICIO

 Regresa areaRiemman

FIN MÉTODO getAreaRiemman

MÉTODO getAreaIntegral: REAL

INICIO

 Regresa areaIntegral

FIN MÉTODO getAreaIntegral

MÉTODO getError: REAL

INICIO

 Regresa error



FIN MÉTODO getError

MÉTODO getRectangulo: cl_plotrectangulo

INICIO

Regresa rectangulo

FIN MÉTODO getRectangulo

MÉTODO calcularArea: REAL

VARIABLES

i, p, recFull, decimales: ENTERO

base, fx, xFin: REAL

linea: ARREGLO DE cl_linea

ladosRectangulo: ARREGLO[1..4, 1..numeroRectangulos] DE

REAL

INICIO

p ←

rectangulo.getrectangulos.getlado1.getpuntoInicio.getprecision

EN CASO DE (p)

INICIO

Caso p = 1:

INICIO

decimales ← 0

FIN

Caso p = 10:

INICIO

decimales ← 1

FIN

Caso p = 100:

INICIO

decimales ← 2

FIN

Caso P = 1000:

INICIO

decimales ← 3

FIN



FIN CASO

base \leftarrow (limiteSuperior-limiteInferior)/numeroRectangulos

recFull \leftarrow 1

linea \leftarrow funcion.getlineas

PARA i \leftarrow 1 Hasta 1+(limiteSuperior-limiteInferior)*p HACER

 xFin \leftarrow linea[i].getpuntoInicio.getx

 SI (REDONDEAR(xFin, decimales) =
 REDONDEAR(limiteInferior+(base*recFull),decimales)) ENTONCES

 fx \leftarrow linea[i].getpuntoInicio.gety

 ladosRectangulo[1,recFull] \leftarrow xFin-base

 ladosRectangulo[2,recFull] \leftarrow 0

 ladosRectangulo[3,recFull] \leftarrow xFin

 ladosRectangulo[4,recFull] \leftarrow fx

 areaRiemman \leftarrow areaRiemman + (fx*base)

 recFull \leftarrow recFull + 1

 FIN SI

 FIN PARA

 rectangulo.setrectangulos(numeroRectangulos,
ladosRectangulo)

 rectangulo.graficar

 funcion.graficar

 Regresa areaRiemman

FIN MÉTODO calcularArea

MÉTODO calcularError: REAL

INICIO

 error \leftarrow |areaIntegral - areaRiemman|

 Regresa error

FIN MÉTODO calcularError

FIN CLASE cl_sumaRiemman



Clase graficaFuncion

VARIABLES

area,error, limInf, limSup, areaAnalitica: REAL

prec, noRectangulos: ENTERO

sumaR: cl_sumaRiemman

INICIO

limInf ← Edit1.Text

limSup ← Edit2.Text

prec ← ComboBox1.Text

noRectangulos ← Edit3.Text

areaAnalitica ← Edit4.Text

sumaR ← cl_sumaRiemman.crear(300, 300, limInf, limSup, areaAnalitica,
noRectangulos, prec, Grafica1)

area ← sumaR.calcularArea

Label6.Caption("Área: ", area)

error ← sumaR.calcularError

Label7.Caption("Error: ", error)

FIN Clase graficaFuncion

IMPLEMENTACIÓN EN FREEPASCAL CON IDE LAZARUS

Nombre del proyecto: areaRiemman.lpi

Programa principal (Unit): areaRiemman_principal.pas

Número de formularios: 1

Programa para definición de objetos: Objetos.pas

```
{*****  
***** DEFINICIÓN DE LA CLASE cl_puntoreal *****}
```

type

cl_puntoreal = class

private

x: real;

y: real;

xo: integer;

yo: integer;

xt: integer;



Privada 16 de Septiembre No. 101.

Zinacantepec, Estado de México.

C.P. 51350



www.mediacod.com



contacto@mediacod.com



7226741918



facebook.com/mediacod



@mediacod



```
yt: integer;
precision: integer;
public
  procedure setx(n:real);
  procedure sety(n:real);
  procedure setxo(n:integer);
  procedure setyo(n:integer);
  procedure setxt(n:integer);
  procedure setyt(n:integer);
  procedure setprecision(n:integer);
  function getx: real;
  function gety: real;
  function getxo: integer;
  function getyo: integer;
  function getxt: integer;
  function getyt: integer;
  function getprecision: integer;
  procedure geotograf;
  constructor create(cx,cy: real; crx,cry,p:integer);
end;
```

```
{*****
***** DEFINICIÓN DE LA CLASE cl_linea *****}
type
cl_linea = class
  private
    puntolnicio: cl_puntoreal;
    puntoFin: cl_puntoreal;

  public
    procedure setpuntolnicio(x, y: real);
    procedure setpuntoFin(x, y: real);
    function getpuntolnicio: cl_puntoreal;
    function getpuntoFin: cl_puntoreal;
    constructor create(xorig, yorig:integer; xlnicio, ylnicio, xFin, yFin:real; p: integer);
end;
{*****
***** DEFINICIÓN DE LA CLASE cl_rectangulo
*****}
```



Privada 16 de Septiembre No. 101.
Zinacantepec, Estado de México.
C.P. 51350



www.mediacod.com



contacto@mediacod.com



7226741918



facebook.com/mediacod



[@mediacod](https://twitter.com/mediacod)



```
type
cl_rectangulo = class
private
    lado1: cl_linea;
    lado2: cl_linea;
    lado3: cl_linea;
    lado4: cl_linea;

public
    procedure setlado1(xInic, yInic, xFin, yFin:real);
    procedure setlado2(xInic, yInic, xFin, yFin:real);
    procedure setlado3(xInic, yInic, xFin, yFin:real);
    procedure setlado4(xInic, yInic, xFin, yFin:real);
    function getlado1: cl_linea;
    function getlado2: cl_linea;
    function getlado3: cl_linea;
    function getlado4: cl_linea;
    constructor create(xorig, yorig:integer; xInicio, yInicio, xFin, yFin:real;
p:integer);
end;

{*****
***** DEFINICIÓN DE LA CLASE cl_plotreal *****}

type
PuntosL = array[1..4, 1..100] of real;
cl_plotreal = class
private
    lineas: TObjectlist;
    superficie: TImage;

public
    procedure setlineas(n:integer; ListaP:PuntosL);
        procedure setsuperficie(ima: TImage);
        function getlineas: TObjectlist;
        function getsuperficie: TImage;
    procedure modelo(izq, der: real);
    procedure graficar;
    constructor create(n, xorig, yorig:integer; salida:TImage; p:integer);
end;
```



Privada 16 de Septiembre No. 101.
Zinacantepec, Estado de México.
C.P. 51350



www.mediacod.com
contacto@mediacod.com
7226741918
facebook.com/mediacod
[@mediacod](https://twitter.com/mediacod)

```
{*****  
***** DEFINICIÓN DE LA CLASE cl_plotrectangulo *****}  
type  
LadosRectangulo = array[1..4,1..600] of real;  
cl_plotrectangulo = class  
  private  
    rectangulos: TObjectlist;  
    superficie: TImage;  
  public  
    procedure setrectangulos(n:integer; LadosR:LadosRectangulo);  
    procedure setsuperficie(ima: TImage);  
    function getrectangulos: TObjectList;  
    function getsuperficie: TImage;  
    procedure graficar;  
    constructor create(numRec, xorig, yorig:integer; xInicio, yInicio, xFin, yFin: real;  
salida: TImage;p:integer);  
  end;
```

```
{*****  
***** DEFINICIÓN DE LA CLASE cl_sumaRiemman *****}  
type  
cl_sumaRiemman = class  
  private  
    limiteInferior: real;  
    limiteSuperior: real;  
    funcion: cl_plotreal;  
    numeroRectangulos: integer;  
    areaRiemman: real;  
    areaIntegral: real;  
    error: real;  
    rectangulo: cl_plotrectangulo;  
  
  public  
    procedure setLimiteInferior(n: real);  
    procedure setLimiteSuperior(n: real);  
    procedure setFuncion(f: cl_plotreal);  
    procedure setNumeroRectangulos(n: integer);  
    procedure setAreaRiemman(n: real);
```



```
procedure setAreaIntegral(n: real);
procedure setError(n: real);
procedure setRectangulo(r: cl_plotrectangulo);
function getLimiteInferior: real;
function getLimiteSuperior: real;
function getFuncion: cl_plotreal;
function getNumeroRectangulos: integer;
function getAreaRiemman: real;
function getAreaIntegral: real;
function getError: real;
function getRectangulo: cl_plotrectangulo;
function calcularArea: real;
function calcularError: real;
constructor create(xorig, yorig:integer;
limInf,limSup,areaAnalitica:real;noRectangulos, prec:integer; salida:TImage);
end;
implementation
{*****
***** MÉTODOS DE LA CLASE cl_puntoreal *****}
procedure cl_puntoreal.setx(n:real);
begin
  x:=n;
end;

procedure cl_puntoreal.sety(n:real);
begin
  y:=n;
end;

procedure cl_puntoreal.setxo(n:integer);
begin
  xo:=n;
end;

procedure cl_puntoreal.setyo(n:integer);
begin
  yo:=n;
end;
```





```
procedure cl_puntoreal.setxt(n:integer);
begin
  xt:=n;
end;

procedure cl_puntoreal.setyt(n:integer);
begin
  yt:=n;
end;

procedure cl_puntoreal.setprecision(n:integer);
begin
  precision:=n;
end;

function cl_puntoreal.getx: real;
begin
  getx:=x;
end;

function cl_puntoreal.gety: real;
begin
  gety:=y;
end;

function cl_puntoreal.getxo: integer;
begin
  getxo:=xo;
end;

function cl_puntoreal.getyo: integer;
begin
  getyo:=yo;
end;

function cl_puntoreal.getxt: integer;
begin
  getxt:=xt;
```



Privada 16 de Septiembre No. 101.
Zinacantepec, Estado de México.
C.P. 51350



www.mediacod.com



contacto@mediacod.com



7226741918



facebook.com/mediacod



[@mediacod](https://twitter.com/mediacod)



```
end;

function cl_puntoreal.getyt: integer;
begin
    getyt:=yt;
end;

function cl_puntoreal.getprecision: integer;
begin
    getprecision:=precision;
end;

procedure cl_puntoreal.geotograf;
begin
    xt:=round(xo+(x*precision));
    yt:=round(yo-(y*precision));
end;

constructor cl_puntoreal.create(cx,cy: real; crx,cry,p:integer);
begin
    x:=cx;
    y:=cy;
    xo:=crx;
    yo:=cry;
    precision:=p;
end;

{*****
***** MÉTODOS DE LA CLASE cl_linea *****}
procedure cl_linea.setpuntoInicio(x, y: real);
begin
    puntoInicio.setx(x);
    puntoInicio.sety(y);
    puntoInicio.geotograf;
end;

procedure cl_linea.setpuntoFin(x, y: real);
begin
    puntoFin.setx(x);
```



Privada 16 de Septiembre No. 101.
Zinacantepec, Estado de México.
C.P. 51350



www.mediacod.com



contacto@mediacod.com



7226741918



facebook.com/mediacod



[@mediacod](https://twitter.com/mediacod)



```
puntoFin.sety(y);
puntoFin.geotograf;
end;

function cl_linea.getpuntoInicio: cl_puntoreal;
begin
    getpuntoInicio:= puntoInicio;
end;

function cl_linea.getpuntoFin: cl_puntoreal;
begin
    getpuntoFin:= puntoFin;
end;

constructor cl_linea.create(xorig, yorig:integer; xInicio, yInicio, xFin, yFin:real; p:
integer);
begin
    puntoInicio := cl_puntoreal.create(xInicio, yInicio, xorig, yorig, p);
    puntoFin := cl_puntoreal.create(xFin, yFin, xorig, yorig, p);
end;

{*****
***** MÉTODOS DE LA CLASE cl_rectangulo *****}

procedure cl_rectangulo.setlado1(xInic, yInic, xFin, yFin:real);
begin
    lado1.setpuntoInicio(xInic,yInic);
    lado1.setpuntoFin(xFin,yFin);
end;

procedure cl_rectangulo.setlado2(xInic, yInic, xFin, yFin:real);
begin
    lado2.setpuntoInicio(xInic,yInic);
    lado2.setpuntoFin(xFin,yFin);
end;

procedure cl_rectangulo.setlado3(xInic, yInic, xFin, yFin:real);
begin
    lado3.setpuntoInicio(xInic,yInic);
    lado3.setpuntoFin(xFin,yFin);
```



```
end;
```

```
procedure cl_rectangulo.setlado4(xInic, yInic, xFin, yFin:real);  
begin  
  lado4.setpuntoInicio(xInic,yInic);  
  lado4.setpuntoFin(xFin,yFin);  
end;
```

```
function cl_rectangulo.getlado1: cl_linea;  
begin  
  Result:=lado1;  
end;
```

```
function cl_rectangulo.getlado2: cl_linea;  
begin  
  Result:=lado2;  
end;
```

```
function cl_rectangulo.getlado3: cl_linea;  
begin  
  Result:=lado3;  
end;
```

```
function cl_rectangulo.getlado4: cl_linea;  
begin  
  Result:=lado4;  
end;
```

```
constructor cl_rectangulo.create(xorig, yorig:integer; xInicio, yInicio, xFin, yFin:real;  
p:integer);  
begin  
  lado1 := cl_linea.create(xorig,yorig,xInicio,yInicio,xFin,yInicio, p);  
  lado2 := cl_linea.create(xorig, yorig, xFin, yInicio, xFin, yFin, p);  
  lado3 := cl_linea.create(xorig, yorig, xFin, yFin, xInicio, yFin, p);  
  lado4 := cl_linea.create(xorig, yorig, xInicio, yFin, xInicio, yInicio, p);  
end;
```

```
{*****}
```



***** MÉTODOS DE LA CLASE cl_plotreal *****}

```
procedure cl_plotreal.setlineas(n:integer; ListaP:PuntosL);
```

```
var
```

```
  i: integer;
```

```
begin
```

```
  for i:=1 to n do
```

```
    begin
```

```
      with lineas do
```

```
        begin
```

```
          (Items[i-1] as cl_linea).setpuntoInicio(ListaP[1,i], ListaP[2,i]);
```

```
          (Items[i-1] as cl_linea).setpuntoFin(ListaP[3,i], ListaP[4,i]);
```

```
        end;
```

```
      end;
```

```
end;
```

```
procedure cl_plotreal.setsuperficie(ima: TImage);
```

```
begin
```

```
  superficie:=ima;
```

```
end;
```

```
function cl_plotreal.getlineas: TObjectlist;
```

```
begin
```

```
  getlineas:=lineas;
```

```
end;
```

```
function cl_plotreal.getsuperficie: TImage;
```

```
begin
```

```
  getsuperficie:=superficie;
```

```
end;
```

```
procedure cl_plotreal.modelo(izq, der: real);
```

```
var
```

```
  i, c, p: integer;
```

```
  vx: real;
```

```
function f(n: real):real;
```

```
begin
```

```
  f:=1-power(n-1,3)-power(n-1,2); // 0 - 1 = 0.91666
```

```
  //f:=power(n,2); // -1.5 - 1 = 1.4583
```





```
//f:=cos(n);           // -2.5 - 2.5 = 1.1969
//f:=sin(n);           // -pi - pi = 2
//f:=ln(n);            // 1 - e (2.71.82) = 1

end;

begin
  c:=0;
  with lineas do
    begin
      p := (Items[0] as cl_linea).getpuntoInicio.getprecision;
      end;

    for i:=1 to round(1+(der-izq)*p) do
      begin
        vx:=izq+(i-1)/p;
        with lineas do
          begin
            (Items[c] as cl_linea).setpuntoInicio(vx,f(vx));
            (Items[c] as cl_linea).setpuntoFin((izq+(i)/p),f(izq+(i)/p));
          end;
          c:=c+1;
        end;
      end;
    end;
  procedure cl_plotreal.graficar;
  var
    i, n, alto, ancho, xCero, yCero: integer;
    cx,cy,cx1,cy1: integer;
  begin
    alto:=600;
    ancho:=600;
    with lineas do
      begin
        xCero:=(Items[1] as cl_linea).getpuntoInicio.getxo;
        yCero:=(Items[1] as cl_linea).getpuntoInicio.getyo;
      end;
    end;

    superficie.Canvas.Pen.Color:=clGreen;
    superficie.Canvas.Line(0, yCero, ancho, yCero);
```





```
superficie.Canvas.Line(xCero, 0, xCero, alto);
    n:=lineas.count;
superficie.Canvas.Pen.Color:=clBlack;
    for i:=1 to n do
begin
    with lineas do
    begin
        cx := (Items[i-1] as cl_linea).getpuntoInicio.getxt;
        cy := (Items[i-1] as cl_linea).getpuntoInicio.getyt;
        cx1 := (Items[i-1] as cl_linea).getpuntoFin.getxt;
        cy1 := (Items[i-1] as cl_linea).getpuntoFin.getyt;
        end;
        superficie.Canvas.Line(cx,cy,cx1,cy1);
    end;
end;

constructor cl_plotreal.create(n, xorig, yorig:integer; salida: TImage; p:integer);
var
    i: integer;
begin
    superficie:=salida;
    lineas:=TObjectlist.Create();
    for i:=1 to n do
    begin
        lineas.Add(cl_linea.create(xorig,yorig,0,0,0,0,p));
    end;
end;

{*****
***** MÉTODOS DE LA CLASE cl_plotrectangulo *****}
procedure cl_plotrectangulo.setrectangulos(n:integer; LadosR:LadosRectangulo);
var
    i:integer;
begin

    for i:=1 to n do
    begin
        with rectangulos do
            begin
```



```
(Items[i-1] as
cl_rectangulo).setlado1(LadosR[1,i],LadosR[2,i],LadosR[3,i],LadosR[4,i]);
(Items[i-1] as
cl_rectangulo).setlado2(LadosR[1,i],LadosR[2,i],LadosR[3,i],LadosR[4,i]);
(Items[i-1] as
cl_rectangulo).setlado3(LadosR[1,i],LadosR[2,i],LadosR[3,i],LadosR[4,i]);
(Items[i-1] as
cl_rectangulo).setlado4(LadosR[1,i],LadosR[2,i],LadosR[3,i],LadosR[4,i]);
end;
end;
end;
```

```
procedure cl_plotrectangulo.setsuperficie(ima: TImage);
begin
superficie := ima;
end;
```

```
function cl_plotrectangulo.getrectangulos: TObjectlist;
begin
getrectangulos := rectangulos;
end;
```

```
function cl_plotrectangulo.getsuperficie: TImage;
begin
getsuperficie := superficie;
end;
```

```
procedure cl_plotrectangulo.graficar;
var
alto, ancho, xCero, yCero, n: integer;
x, y, x1, y1, i: integer;
begin
alto := 300;
ancho := 300;
```

```
with rectangulos do
begin
xCero := (Items[0] as cl_rectangulo).getlado1.getpuntoInicio.getxo;
yCero := (Items[0] as cl_rectangulo).getlado1.getpuntoInicio.getyo;
```





```
end;
superficie.Canvas.Pen.Color := clGreen;
superficie.Canvas.Line(0, yCero, ancho, yCero);
superficie.Canvas.Line(xCero, 0, xCero, alto);

n:=rectangulos.Count;
superficie.Canvas.Pen.Color := clskyblue;
for i:= 1 to n do
begin
  with rectangulos do
  begin
    x := (Items[i-1] as cl_rectangulo).getlado1.getpuntoInicio.getxt;
    y := (Items[i-1] as cl_rectangulo).getlado1.getpuntoInicio.getyt;
    x1 := (Items[i-1] as cl_rectangulo).getlado2.getpuntoFin.getxt;
    y1 := (Items[i-1] as cl_rectangulo).getlado2.getpuntoFin.getyt;
  end;
  superficie.Canvas.Rectangle(x, y, x1, y1);
end;
end;

constructor cl_plotrectangulo.create(numRec, xorig, yorig:integer; xInicio, yInicio,
xFin, yFin: real; salida: TImage; p:integer);
var
  i: integer;
begin
  superficie := salida;
  rectangulos:=TObjectlist.Create();
  for i:=1 to numRec do
  begin
    rectangulos.Add(cl_rectangulo.create(xorig, yorig, xInicio, yInicio, xFin, yFin, p));
  end;
end;

{*****
***** MÉTODOS DE LA CLASE cl_sumaRiemman *****}
procedure cl_sumaRiemman.setLimiteInferior(n: real);
begin
  limiteInferior:=n;
end;
```





```
procedure cl_sumaRiemman.setLimiteSuperior(n: real);
begin
    limiteSuperior:=n;
end;

procedure cl_sumaRiemman.setFuncion(f: cl_plotreal);
begin
    funcion:=f;
end;

procedure cl_sumaRiemman.setNumeroRectangulos(n: integer);
begin
    numeroRectangulos:=n;
end;

procedure cl_sumaRiemman.setAreaRiemman(n: real);
begin
    areaRiemman:=n;
end;

procedure cl_sumaRiemman.setAreaIntegral(n: real);
begin
    areaIntegral:=n;
end;

procedure cl_sumaRiemman.setError(n: real);
begin
    error:=n;
end;

procedure cl_sumaRiemman.setRectangulo(r: cl_plotrectangulo);
begin
    rectangulo:=r;
end;

function cl_sumaRiemman.getLimiteInferior: real;
begin
    Result:=limiteInferior;
```



Privada 16 de Septiembre No. 101.
Zinacantepec, Estado de México.
C.P. 51350



www.mediacod.com



contacto@mediacod.com



7226741918



facebook.com/mediacod



[@mediacod](https://twitter.com/mediacod)



```
end;

function cl_sumaRiemman.getLimiteSuperior: real;
begin
    Result:=limiteSuperior;
end;

function cl_sumaRiemman.getFuncion: cl_plotreal;
begin
    Result:=funcion;
end;

function cl_sumaRiemman.getNumeroRectangulos: integer;
begin
    Result:=numeroRectangulos;
end;

function cl_sumaRiemman.getAreaRiemman: real;
begin
    Result:=areaRiemman;
end;

function cl_sumaRiemman.getAreaIntegral: real;
begin
    Result:=areaIntegral;
end;

function cl_sumaRiemman.getError: real;
begin
    Result:=error;
end;

function cl_sumaRiemman.getRectangulo: cl_plotrectangulo;
begin
    Result:=rectangulo
end;

function cl_sumaRiemman.calcularArea: real;
var
```





```
i,p,recFull: integer;
base, fx, xFin: real;
recs, linea: TObjectlist;
ladosRectangulo: array[1..4,1..600] of real;
decimales: integer;
begin
  recs := rectangulo.getrectangulos;

  with recs do
    begin
      p := (Items[0] as cl_rectangulo).getlado1.getpuntoInicio.getprecision;
    end;

    case p of
      1: decimales:=0;
      10: decimales:=-1;
      100: decimales:=-2;
      1000: decimales:=-3;
    end;

    base:=(limiteSuperior-limiteInferior)/numeroRectangulos;
    recFull:=1;
    linea:=funcion.getlineas;

    for i:=1 to round(1+(limiteSuperior-limiteInferior)*p) do
      begin
        with linea do
          begin
            xFin := (Items[i-1] as cl_linea).getpuntoInicio.getx;
          end;

          if roundto(xFin,decimales) = roundto(limiteInferior+(base*recFull),decimales)
          then
            begin
              with linea do
                begin
                  fx := (Items[i-1] as cl_linea).getpuntoInicio.gety;
                end;
              end;
            end;
          end;
        end;
      end;
```





```
ladosRectangulo[1,recFull]:= xFin-base;
ladosRectangulo[2,recFull]:= 0;
ladosRectangulo[3,recFull]:= xFin;
ladosRectangulo[4,recFull]:= fx;
areaRiemman:=abs(areaRiemman+(fx*base));
recFull:=recFull+1;
end;
end;
rectangulo.setrectangulos(numeroRectangulos, ladosRectangulo);
rectangulo.graficar;
funcion.graficar;
result:=roundto(areaRiemman,-4);
end;

function cl_sumaRiemman.calcularError: real;
begin
    error:=abs(roundto(areaIntegral,-4)-roundto(areaRiemman,-4));
    result:=roundto(error,-4);
end;

constructor cl_sumaRiemman.create(xorig, yorig:integer;
limInf,limSup,areaAnalitica:real;noRectangulos, prec:integer; salida:TImage);
var
    numeroObjetos: integer;
begin
    limiteInferior:=limInf;
    limiteSuperior:=limSup;
    numeroRectangulos:=noRectangulos;
    areaIntegral:=areaAnalitica;
    numeroObjetos:=round(((limiteSuperior-limiteInferior)*prec)+1);
    funcion := cl_plotreal.create(numeroObjetos,xorig,yorig,salida,prec);
    funcion.modelo(limiteInferior,limiteSuperior);
    rectangulo :=
cl_plotrectangulo.create(numeroRectangulos,xorig,yorig,0,0,0,0,salida,prec);
end;
end.
```



CLASE graficaFuncion

```
procedure TForm1.Button1Click(Sender: TObject);
var
  area,error, limInf, limSup, areaAnalitica: real;
  prec, noRectangulos: integer;
  sumaR: cl_sumaRiemman;
begin
  Grafica1.Canvas.FillRect(1,1,600,600);
  limInf:=strtofloat(Edit1.Text);
  limSup:=strtofloat(Edit2.Text);
  prec:=strtoint(ComboBox1.Text);
  noRectangulos:=strtoint(Edit3.Text);
  areaAnalitica:=strtofloat(Edit4.Text);
  sumaR :=
cl_sumaRiemman.create(300,300,limInf,limSup,areaAnalitica,noRectangulos,prec,
Grafica1);
  area := sumaR.calcularArea;
  Label6.Caption:=Concat('Área: ',floattostr(area),' u2');
  error:=sumaR.calcularError;
  Label7.Caption:=Concat('Error: ',floattostr(error));
  sumaR.Destroy;
end;
```

