# 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.

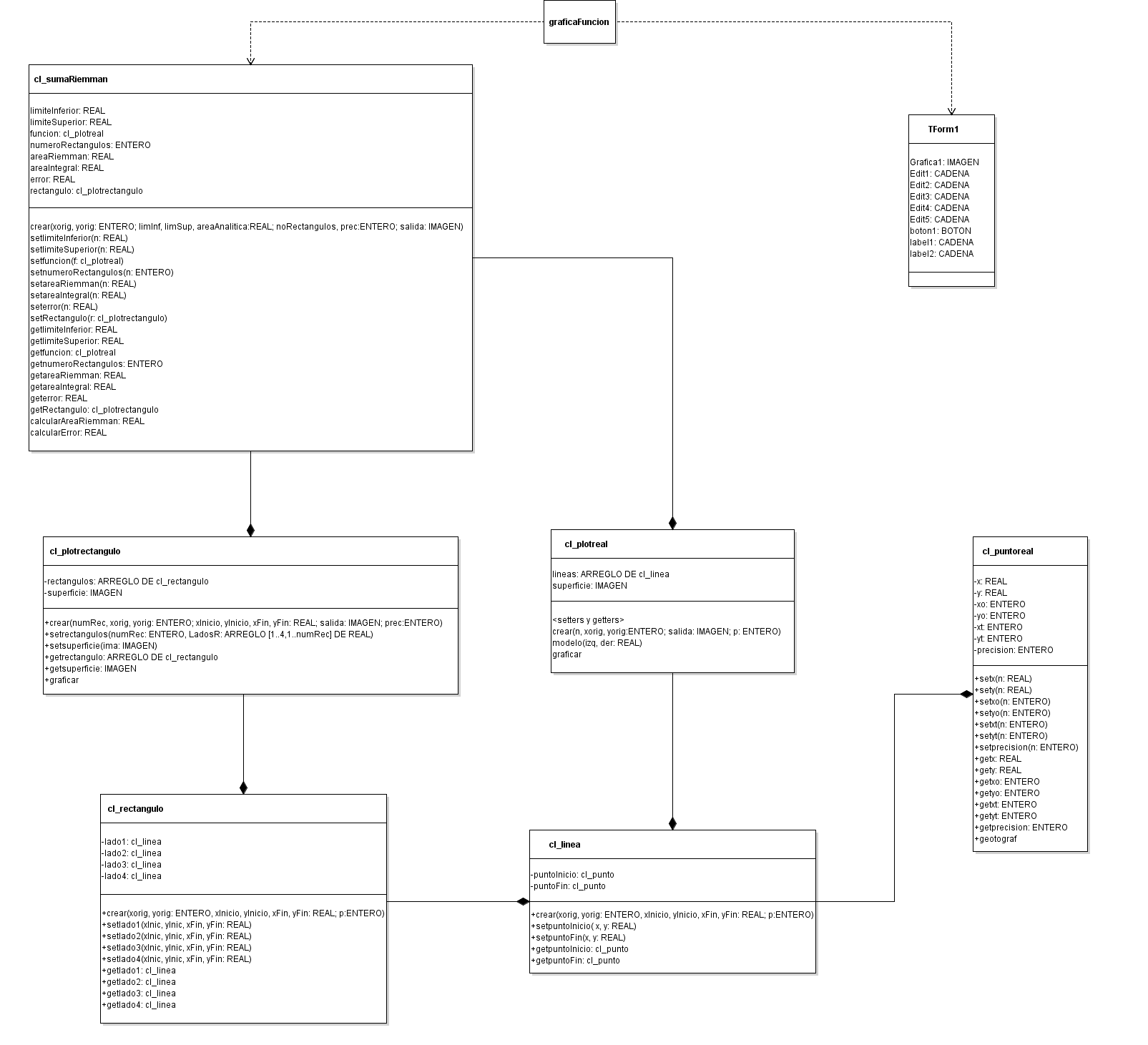
SOLUCIÓN

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.



*Figura 1*

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

DISEÑO DE INTERFAZ

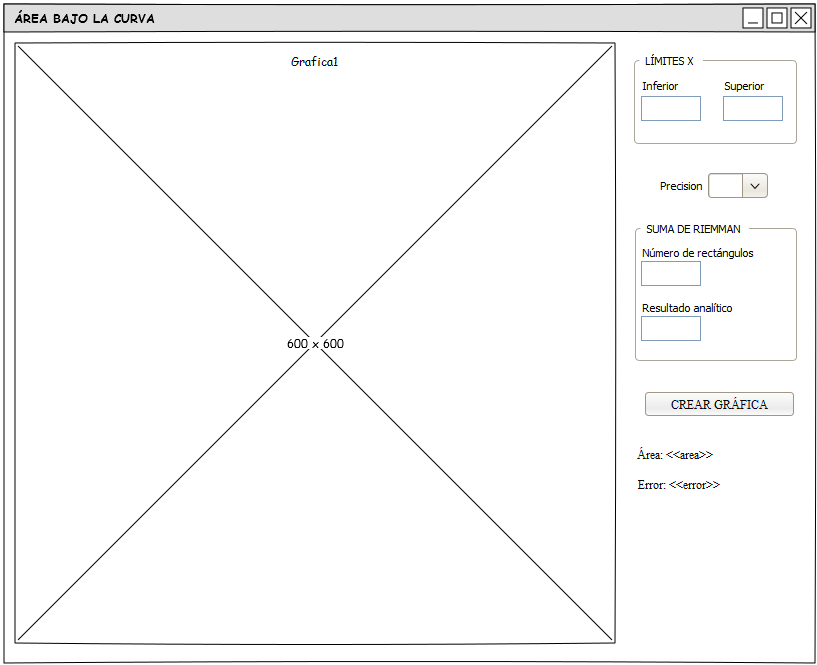


DIAGRAMA DE CLASES

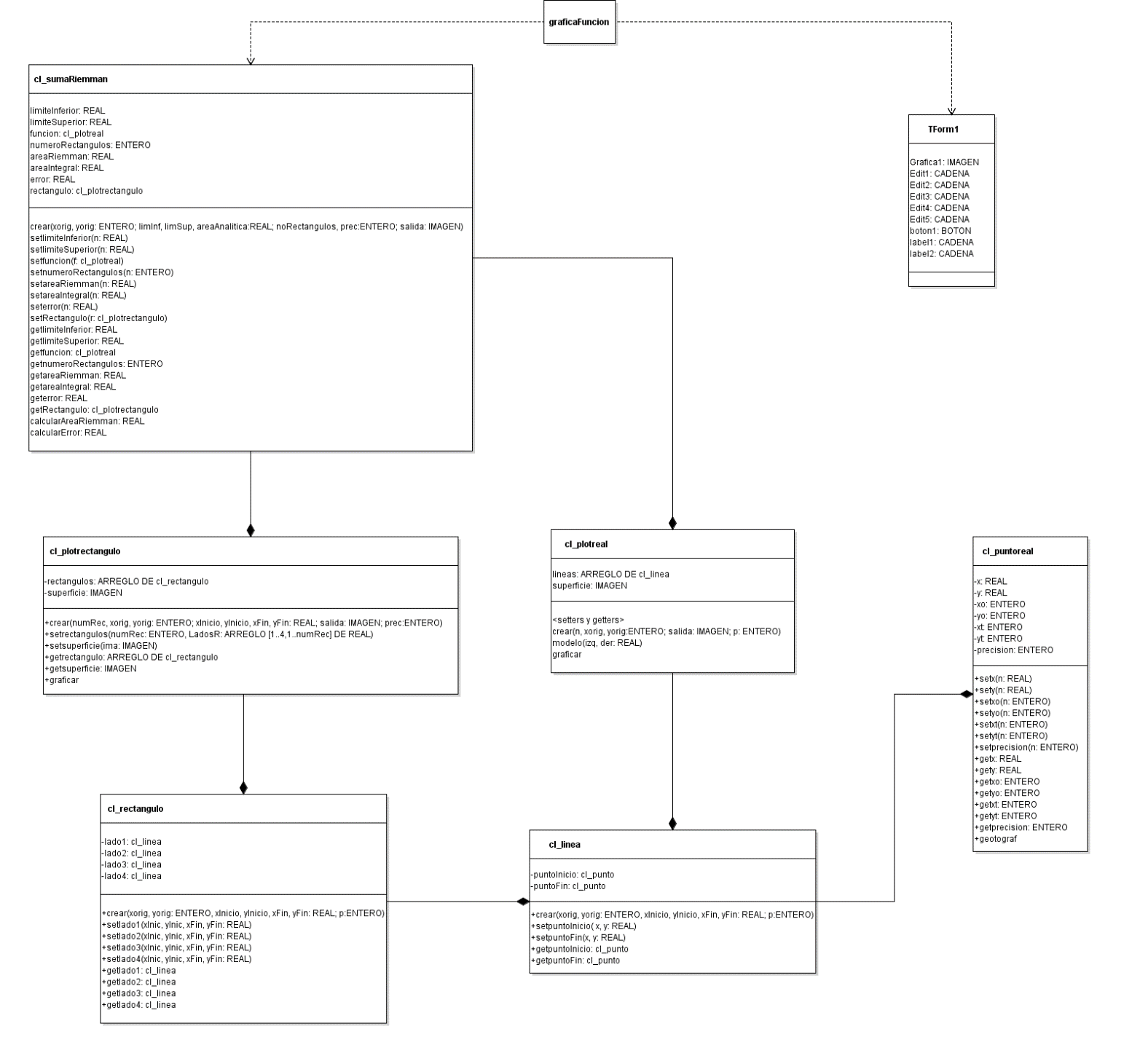
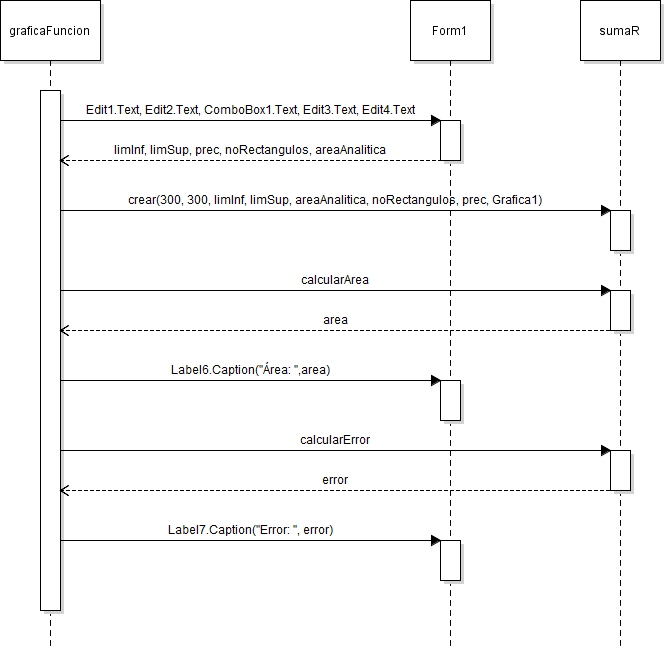


DIAGRAMA DE SECUENCIA



PSEUDOCÓDIGO

**CLASE cl\_plotreal**

INICIO

ATRIBUTOS

lineas: ARREGLO DE cl\_linea

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 setprecission(n:ENTERO)

INICIO

precision ← n

FIN MÉTODO setprecission

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

MÉTODO getyt: ENTERO

INICIO

Regresa yt

FIN MÉTODO getyt

MÉTODO getprecision: ENTERO

INICIO

Regresa precision

FIN MÉTODO getprecision

MÉTODO geotograf

INICIO

xt ← xo+(x\*precision)

yt ← yo-(y\*precision)

FIN MÉTODO geotograf

MÉTODOl cl\_puntoreal(cx,cy: REAL; crx,cry,p:ENTERO)

INICIO

x ← cx

y ← cy

xo ← crx

yo ← cry

precision ← 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 ← cl\_puntoreal.crear(xInicio, yInicio, xorig, yorig, p);

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

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 ← Tamaño(rectangulos)

PARA i ← 1 HASTA n HACER

cx ← rectángulo[i].getlado1.getpuntoInicio.getxt

cy ← rectángulo[i].getlado1.getpuntoInicio.getyt

cx1 ← rectángulo[i].getlado2.getpuntoFin.getxt

cy1 ← rectángulo[i].getlado2.getpuntoFin.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 ← limInf

limiteSuperior ← limSup

numeroRectangulos ← noRectangulos

areaIntegral ← areaAnalitica

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

funcion ← cl\_plotreal.crear(numeroObjetos, xorig, yorig, salida, prec)

funcion.modelo(limiteInferior, limiteSuperior)

rectangulo ← cl\_plotrectangulo.crear(numeroRectangulos, xorig, yorig, 0, 0, 0, 0, salida, prec)

FIN MÉTODO cl\_sumaRiemman

MÉTODO setLimiteInferior(n: REAL)

INICIO

limiteInferior ← n

FIN MÉTODO setLimiteInferior

MÉTODO setLimiteSuperior(n: REAL)

INICIO

limiteSuperior ← n

FIN MÉTODO setLimiteSuperior

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 ← 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 ← (limiteSuperior-limiteInferior)/numeroRectangulos

recFull ← 1

linea ← funcion.getlineas

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

xFin ← linea[i].getpuntoInicio.getx

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

fx ← linea[i].getpuntoInicio.gety

ladosRectangulo[1,recFull] ← xFin-base

ladosRectangulo[2,recFull] ← 0

ladosRectangulo[3,recFull] ← xFin

ladosRectangulo[4,recFull] ← fx

areaRiemman ← areaRiemman + (fx\*base)

recFull ← recFull + 1

FIN SI

FIN PARA

rectangulo.setrectangulos(numeroRectangulos, ladosRectangulo)

rectangulo.graficar

funcion.graficar

Regresa areaRiemman

FIN MÉTODO calcularArea

MÉTODO calcularError: REAL

INICIO

error ← |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;

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

puntoInicio: cl\_puntoreal;

puntoFin: cl\_puntoreal;

public

procedure setpuntoInicio(x, y: real);

procedure setpuntoFin(x, y: real);

function getpuntoInicio: cl\_puntoreal;

function getpuntoFin: cl\_puntoreal;

constructor create(xorig, yorig:integer; xInicio, yInicio, xFin, yFin:real; p: integer);

end;

**{\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* DEFINICIÓN DE LA CLASE cl\_rectangulo \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}**

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;

**{\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 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;

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);

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;

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;

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;

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;

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;