MD5\_1

Dainis Tillers

dt08050

class

ROOT

inherit

ARGUMENTS

create

make

feature {NONE} -- Initialization

make

local

q: QUAD;

t: TRIANGLE;

c: CIRCLE;

l: FIGURELIST;

p: POINT

do

create l.make;

create q.make\_xy(0, 0, 0, 2, 4, 2, 4, 0);

p := q.weight\_center();

l.put\_front(q);

create t.make\_xy(0, 2, 0, 5, 4, 2);

l.put\_front(t);

create c.make\_xy(0, 0, 3);

l.put\_front(c);

print(l.area());

print("%N");

p := l.weight\_center();

print("x: ");

print(p.get\_x);

print(" y: ")

print(p.get\_y());

print("%N");

end

end

deferred class FIGURE

inherit DOUBLE\_MATH

feature

weight\_center: POINT is deferred end

area: REAL\_64 is deferred end

end

class

FIGURELIST

inherit

FIGURE undefine is\_equal, copy end

LINKED\_LIST[FIGURE]

create

make

feature

weight\_center: POINT --aprekina sistemas smaguma centru

local

p:POINT

do

create p.make\_xy(0, 0);

from start

until exhausted

loop

p.adjust\_x(item.weight\_center().get\_x() \* item.area());

p.adjust\_y(item.weight\_center().get\_y() \* item.area());

forth;

end

p.scale(1 / area());

Result := p;

end

area: REAL\_64 --aprekina saraksta kopejo laukumu, saskaitot kopa visu elementu laukumus

local

a: REAL\_64

do

from start

until exhausted

loop

a := a + item.area;

forth;

end

Result := a;

end

end

class POINT

inherit DOUBLE\_MATH

create make\_xy

feature

get\_x: REAL\_64

do

Result := x

end --atgriez x vertibu

get\_y: REAL\_64

do

Result := y

end --atgriez y vertibu

adjust\_x(adjustment: REAL\_64)

do

x := x + adjustment; --izmaina x vertibu

end

adjust\_y(adjustment: REAL\_64)

do

y := y + adjustment; --izmaina y vertibu

end

distance(p:POINT): REAL\_64

do

Result := sqrt((x-p.get\_x())^2+(y-p.get\_y())^2) --aprekina attalumu starp punktiem

end

scale(ratio: REAL\_64) --izmaina x un y vertibu par noteiktu attiecibu

do

x := x \* ratio;

y := y \* ratio;

end

make\_xy(xx:REAL\_64; yy: REAL\_64) --konstruktors

do

x:=xx;

y:=yy;

end

feature {NONE}

x,y: REAL\_64 -- x un y ir privati

invariant

invariant\_clause: True

end

class

TRIANGLE

inherit

FIGURE

create make\_points, make\_xy

feature

make\_points(vv1, vv2, vv3 : POINT) --izviedo trijsturi izmantojot POINT objektus ka ta virsotnes

do

v1 := vv1;

v2 := vv2;

v3 := vv3;

end

make\_xy(xv1, yv1, xv2, yv2, xv3, yv3: REAL\_64) --izveido trijsturi pec ta virsotnu koordinatas

do

create v1.make\_xy(xv1, yv1);

create v2.make\_xy(xv2, yv2);

create v3.make\_xy(xv3, yv3);

end

area: REAL\_64 --aprekina trijstura laukumu

local

a, b, c, p: REAL\_64

do

--malu garumi

a := v1.distance(v2);

b := v2.distance(v3);

c := v3.distance(v1);

p := (a + b + c) / 2; --herona formula

Result := sqrt(p\*(p-a)\*(p-b)\*(p-c));

end

weight\_center: POINT --aprekina smaguma centru

local

p: POINT

do

create p.make\_xy ((v1.get\_x() + v2.get\_x() + v3.get\_x()) / 3, (v1.get\_y() + v2.get\_y() + v3.get\_y()) / 3); --smaguma centrs atrodas medianu krustpunkta

Result := p;

end

feature {NONE}

v1, v2, v3: POINT;

end

class

QUAD

inherit

FIGURE

create make\_points, make\_xy

feature

make\_points(vv1, vv2, vv3, vv4: POINT) --konstruktors, kas izveido cetrsturi izmantojot padotos POINT objektus, virsotnes ir japadod sakartotas pulkstena raditaja virziena

do

v1 := vv1;

v2 := vv2;

v3 := vv3;

v4 := vv4;

end

make\_xy(xv1, yv1, xv2, yv2, xv3, yv3, xv4, yv4: REAL\_64) --konstruktors kas izveido cetrsturi no padotajam koordinatem, virsotnes ir japadod sakartotas pulkstena raditaja virziena

do

create v1.make\_xy(xv1, yv1);

create v2.make\_xy(xv2, yv2);

create v3.make\_xy(xv3, yv3);

create v4.make\_xy(xv4, yv4);

end

area: REAL\_64 --aprekina cetrstura laukumu, laukums tiek aprekinats sadalot cetrsturi divos trijsturos, siem trijsturiem aprekinot laukumu un tos saskaitot kopa iegust cetrstrura laukumu

local

t1, t2: TRIANGLE

do

create t1.make\_points(v1, v2, v3);

create t2.make\_points(v1, v3, v4);

Result := t1.area() + t2.area();

end

weight\_center: POINT --smaguma centra aprekinasana

local

p, p1, p2, p3, p4: POINT;

t1, t2, t3, t4: TRIANGLE;

k1, k2, b1, b2, x, y: REAL\_64

do

create t1.make\_points (v1, v2, v3); --tiek izveidoti 4 trijsturi, kurus iegust cetrsturi sadalot pa diognali

create t2.make\_points (v1, v3, v4);

create t3.make\_points (v1, v2, v4);

create t4.make\_points (v2, v3, v4);

p1 := t1.weight\_center(); --ar diognali sadalitajiem trijsturu pariem atrod smaguma centrus

p2 := t2.weight\_center();

p3 := t3.weight\_center();

p4 := t4.weight\_center();

--pirmas diognales trijsturu para apstrade

k1 := (p2.get\_y() - p1.get\_y()) / (p2.get\_x() - p1.get\_x()); --atrod taisnes vienadojuma koeficentu

b1 := -p1.get\_x() \* k1 + p1.get\_y(); --atrod taisnes vienadojuma parametru

--otras diognales trijsturu paru apstrade

k2 := (p4.get\_y() - p3.get\_y()) / (p4.get\_x() - p3.get\_x());

b2 := -p3.get\_x() \* k2 + p3.get\_y();

--atrod krustpunktu, diognales pretejas puses atrodosos trijsturu ar otras diognales tada pasa veida izveidoto trijsturu smagumu centru savienojosajam taisnem

x := (b2 - b1) / (k1 - k2);

y := k1 \* x + b1;

create p.make\_xy (x, y);

Result := p;

end

feature {NONE}

v1, v2, v3, v4: POINT;

end

class

CIRCLE

inherit

FIGURE

create make\_point, make\_xy

feature

make\_point(cc: POINT; rr: REAL\_64) --izveido rinki centra koordinatas izmantojot ka POINT objektu

do

c := cc;

r := rr;

end

make\_xy(xx, yy, rr : REAL\_64) --izveido rinki izmantojot centra x un y koordinatas

local

p: POINT

do

create p.make\_xy (xx, yy);

r := rr;

c := p;

end

area: REAL\_64 --aprekina laukumu

do

Result := PI \* r \* r /2;

end

weight\_center: POINT --rinka smaguma centrs atrodas ta centra

do

Result := c;

end

feature {NONE}

c : POINT;

r : REAL\_64;

end