Feladat

A feladatokban olyan szabályos síkidomokkal kell dolgozni, mint kör, szabályos háromszög, négyzet, szabályos hatszög. Ezek reprezentálhatóak a középponttal és az oldalhosszal, illetve a sugárral, ha feltesszük, hogy a sokszögek esetében az egyik oldal párhuzamos a koordináta rendszer vízszintes tengelyével, és a többi csúcs ezen oldalra fektetett egyenes felett helyezkedik el. A síkidomokat szövegfájlból töltsük be! A fájl első sorában szerepeljen a síkidomok száma, majd ez egyes síkidomok. Az első jel azonosítja a síkidom fajtáját, amit követnek a középpont koordinátái és a szükséges hosszúság. A feladatokban a beolvasáson kívül a síkidomokat egységesen kezeljük!

Adjuk meg melyik síkidom befoglaló téglalapja a legnagyobb területű! Egy síkidom befoglaló téglalapja lefedi a síkidomot, oldalai párhuzamosak a tengelyekkel.

Síkidomok

Típusspecifikáció

A síkidom típus lesz a feladatban résztvevő kör és szabályos sokszög típusok őse. A feladathoz egyetlen típusműveletre van szükségünk: egy olyan függvényre, amely megmondja egy adott síkidom befoglaló téglalapjának területét.

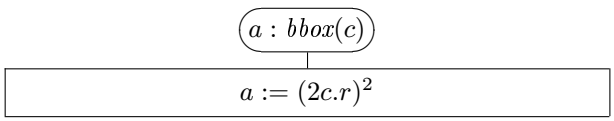
A kör illetve a szabályos sokszög típusok új típusműveletet nem vezetnek be.

Reprezentáció

A síkidom típust közvetlenül nem használjuk, típusérték-halmaza az üres halmaz, reprezentációja nincs. A kört a sugarával, a szabályos sokszöget a csúcsok számával és az élhosszal reprezentáljuk.

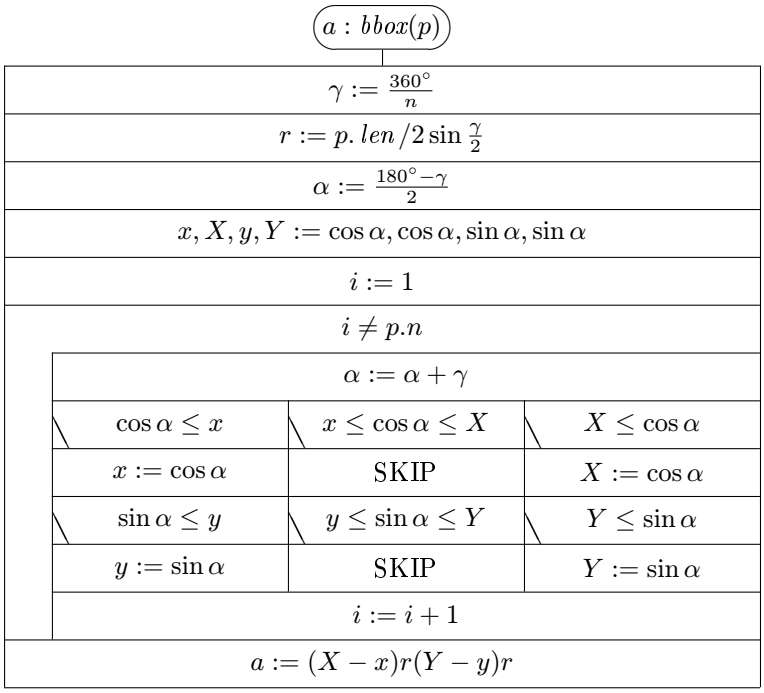
Absztrakt implementáció

Befoglaló téglalap területének kiszámítása körre



Befoglaló téglalap területének kiszámítása szabályos sokszögre

A befoglaló téglalapot egy minimum- és maximumkereséssel találjuk meg: a poligon csúcsainak X- és Y-koordinátáinak szélsőértékeit keressük. A csúcsok koordinátáit $\frac{360^\circ}{n}$ léptékű forgással kapjuk, a körbeírt kör sugarának segítségével.



C++ implementáció

A síkidom, kör, és szabályos sokszög típusokat osztályokként, a köztük lévő relációt leszármazással implementáljuk. A síkidom absztrakt osztály kódja:

```
class Shape
{
public:
    virtual ~Shape() {};
    virtual double bounding_box_area () const = 0;
    virtual std::string str () const = 0;
};
```

A kör és a sokszög osztály

Ez a két osztály értelemszerűen a síkidom osztályból származik le, és implementálja a műveletet:

```
class Circle: public Shape
{
    double r;

public:
    Circle (double r);

    double bounding_box_area () const;
    std::string str () const;
};

double Circle::bounding_box_area () const
{
    return 2 * r * 2 * r;
}

std::string Circle::str () const
{
    std::stringstream buf;
    buf << "Kor (" << r << ")";
    return buf.str ();
}

class Polygon: public Shape
{
    double n;
    double len;

public:
    Polygon (double n, double len);

    double bounding_box_area () const;
    std::string str () const;
};

double Polygon::bounding_box_area () const
{
    double alpha = 2 * M_PI / n;
    double radius = len / (2 * std::sin (alpha / 2));

    double alpha_i = 0.5 * (alpha - M_PI);
    double min_x = std::cos (alpha_i);
```

```
    double max_x = min_x;
    double min_y = std::sin (alpha_i);
    double max_y = min_y;

    for (unsigned int i = 1; i != n; ++i)
    {
        alpha_i += alpha;

        double x = std::cos (alpha_i);
        double y = std::sin (alpha_i);

        min_x = std::min (min_x, x);
        max_x = std::max (max_x, x);

        min_y = std::min (min_y, y);
        max_y = std::max (max_y, y);
    }

    return ((max_x - min_x) * radius) * ((max_y - min_y) * radius);
}

std::string Polygon::str () const
{
    std::stringstream buf;
    buf << n << "-szog (oldalhossz: " << len << ")";
    return buf.str ();
}
```

A főprogram

A főprogram a maximumkeresés tétele segítségével megkeresi az absztrakt file-ból beolvasott síkidomok közül a legnagyobb területűt:

```
class ShapeFile
{
    std::istream &str;

public:
    ShapeFile (std::istream &str_): str (str_) {}

    IOState Read (std::auto_ptr<Shape> &dx);
};

IOState ShapeFile::Read (std::auto_ptr<Shape> &dx)
{
    int shape_type;
    str >> shape_type;
    if (str.eof ()) return ABNORM;

    double shape_size;
    str >> shape_size;
    if (str.eof ()) return ABNORM;

    if (shape_type == 0)
        dx = std::auto_ptr<Shape> (new Circle (shape_size));
    else
        dx = std::auto_ptr<Shape> (new Polygon (shape_type, shape_size));
```

```
    return NORM;
}

std::auto_ptr<Shape> max_bounding_box_area (ShapeFile &x)
{
    std::auto_ptr<Shape> max;
    double  max_val = 0;

    std::auto_ptr<Shape> dx;
    IOState sx = x.Read (dx);

    while (sx == NORM)
    {
        double val = dx->bounding_box_area ();
        if (val >= max_val)
        {
            max = dx;
            max_val = val;
        }

        sx = x.Read (dx);
    }

    return max;
}

int main (int argc, char **argv)
{
    ShapeFile x (std::cin);
    std::auto_ptr<Shape> s = max_bounding_box_area (x);

    if (s.get ())
    {
        std::cout << "A legnagyobb befoglaloteglalapu sikidom: " << s->str () << std::endl
                  << "Ennek befoglalo-teglalapja: " << s->bounding_box_area () << std::endl;
    } else {
        std::cerr << "HIBA: Ures sikidom-lista!" << std::endl;
    }
}
```