Zergi Máté
BKZ85U

**Programozás alapjai 2.**

**Nagyházi**

# Feladat:
# Síkidomok

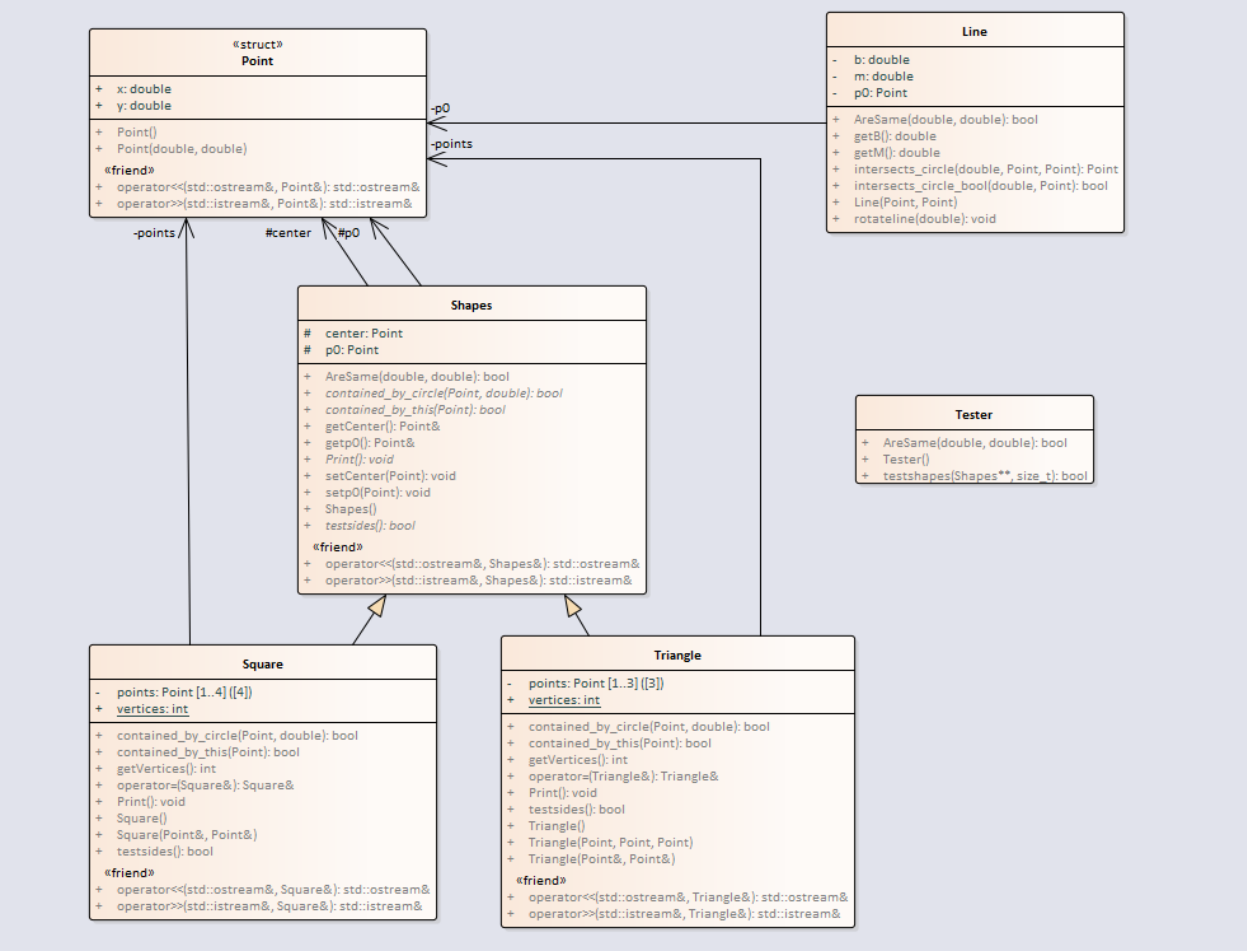
Készítsen absztrakt síkidom-osztályt, és valósítson meg segítségével szabályos háromszöget, négyzetet és kört! Ezen síkidomokat középpontjuk és egy csúcsuk (kör esetén a körvonal egy pontja) határozza meg, amelyek kétdimenziós koordinátákként olvashatóak be egy istream típusú objektumról. A síkidomoknak legyen olyan metódusa, amellyel eldönthető, hogy egy adott pont a síkidom területére esik-e! Legyen továbbá olyan metódusuk is, ami megadja, hogy tartalmazza-e azokat egy adott sugarú, origó középpontú kör!
Írjon főprogramot, amely egy fájlból {típus, középpont, csúcs} tartalmú sorokat olvas be (az istream >> síkidom operátor felhasználásával)! A beolvasott síkidomok közül azokat tárolja el (heterogén kollekció), amelyek teljes terjedelmükben az origó középpontú egységkörön kívül esnek. Ezután koordinátákat olvasson be a szabványos bemenetről a fájl végéig, és írja ki az egyes pontokhoz azon eltárolt síkidomok adatait (név, középpont, csúcs), amelyek az adott pontot tartalmazzák. A megoldáshoz **ne** használjon STL tárolót!

# Megvalósítás:

A megvalósításban a program egy „shapes.txt” nevű fájlból olvas be, fájl végéig, majd kiírja azokat a síkidomokat, amelyek az egységsugarú, origó középpontú körön kívül esnek. Majd kiírja a fájlban található pontok közül azokat, amelyeket tartalmaz bármely, a fentieknek megfelelő síkidom, és kilistázza a síkidomokat, amelyek tartalmazzák a pontot. Ha a fájl végére ér a program, akkor lefuttat egy tesztet, amely az összes, a fájlban szereplő síkidomra leteszteli, hogy valóban szabályos síkidomot generált-e. A fájlformátum a következő: „tri”,”sqr”,”pnt”-al megadható a típus amely a következő sortól következik (háromszög, négyzet, vagy pont) és a következő 2 sorban háromszog és négyzet esetén az első sorban a centerpont koordinátái „x y” formátumban, majd új sorban az adott csúcs koordinátái „x y” formátumban, pont esetén pedig új sorban „x y”, ahol x és y a pontok x és y koordinátája.

A program tudja kezelni az y-al párhuzamos egyeneseket, azt, ha egy síkidomot nem teljes terjedelmében tartalmaz egy kör, hanem csak egy adott területrészletét, valamint minden megadható center-csúcs párosításra képes kiszámolni az adott szabályos síkidomot.

A program a síkidomok csúcsait a köré írható kör segítségével számítja ki. Felveszi a centerpont és az adott csúcs közti egyenest, majd megfelelő elforgatásokkal kiszámítja a kőré írható kör és a megfelelő egyenesek metszéspontjait, így megkapva a csúcsokat. Az ehhez szükséges osztályok és függvények az alábbi UML diagramon láthatók (**1.1 ábra)**.




1.1 ábra

## Osztályok és tagfüggvények rövid leírása

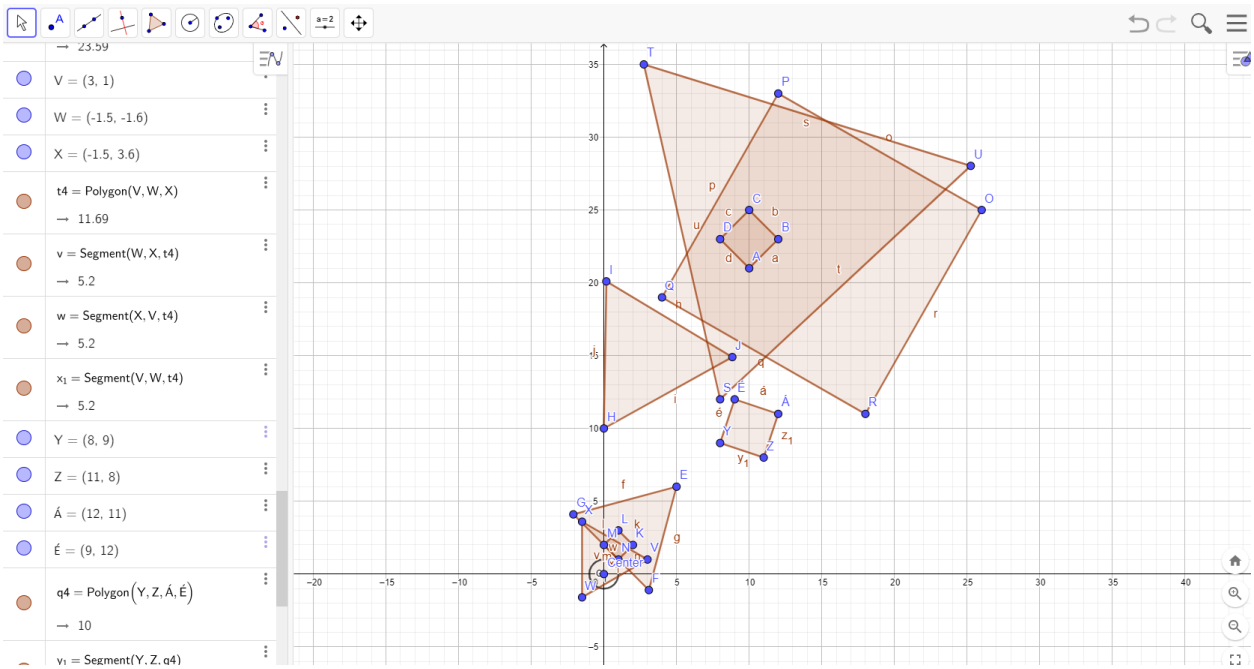
- **class Line**: egyenes osztály, ennek segítségével lehet meghatározni a sokszögeket
  - **Adattagok**
    - double m: magasság
    - double b: mx+b-ből b
    - Point p0: egy adott pont, kell a számításokhoz
  - **Függvények**
    - Line(Point x1, Point x2): *A megadott pontok által meghatározott egyenest hozza létre*
    - void rotateline(double deg): *egyenes elforgatása, szögben kell megadni, ezt átváltja radiánba számolásnál*
    - Point intersects_circle(double r,Point cp,Point excluded): *Az egyenes és egy adott kör metszéspontjának meghatározása*
    - bool intersects_circle_bool(double r,Point cp): *Megállapítja, hogy az egyenes metszi-e az adott kört, vagy sem*
    - double getM(): *m lekérése*
    - double getB(): b lekérése
    - bool AreSame(double a,double b): *segédfüggvény double-ök összehasonlításához*

- **struct Point**: pont struktúra
  - **Adattagok**
    - double x
    - double y
  - **Függvények**
    - Point(): x(0), y(0)
    - Point(double x, double y): x(x), y(y)
    - friend std::istream& operator>>(std::istream& in, Point& p)
    - friend std::ostream& operator<<(std::ostream& out, Point& p)
- **class Shapes**: Alaposztály síkidomoknak
  - **Adattagok**
    - Point center: *középpont*
    - Point p0: *a pont, ami rajta van az alakzaton*
  - **Függvények**
    - Shapes():center(0,0),p0(0,0): *default konstruktor*
    - friend std::istream& operator>>(std::istream& in, Shapes& shp);
    - friend std::ostream& operator<<(std::ostream& out, Shapes& shp);
    - Point& getCenter(): *centerpont lekérése*
    - void setCenter(Point p): *centerpont beállítása*
    - Point& getp0(): *adott pont lekérése*
    - void setp0(Point p): *az adott pont beállítása*
    - virtual bool contained_by_this (Point p0)=0: *az adott pontot tartalmazza-e az alakzat*
    - virtual bool contained_by_circle(Point p0, double r)=0*: az adott kör tartalmazza-e az alakzatot*
    - virtual void Print()=0: *hogy a kollekcióban mindig a megfelelő kiíró függvény hívódjon meg*
    - virtual bool testsides()=0: *tester-nek szükséges, oldalak egyenlőségét vizsgálja*
    - bool AreSame(double a,double b): *segédfüggvény double-ök összehasonlításához*
- **class Square:** Négyzetek osztálya
  - **Adattagok**
    - Point points[4];
    - static int vertices: *teszteléshez szükséges, mivel heterogén kollekción tesztelünk*
  - **Függvények**
    - Square():Shapes(): *default konstruktor*
    - Square(Point& c, Point& p): *centerpont és csúcs által megadott négyzet létrehozása*
    - friend std::ostream& operator<<(std::ostream& out, Square& sqr);
    - friend std::istream& operator>>(std::istream& in, Square& sqr);
    - Square& operator=(const Square& sqr);
    - bool contained_by_this (Point p): *az adott pontot tartalmazza-e a négyzet*
    - bool contained_by_circle(Point p, double r): *az adott kör tartalmazza-e a négyzetet*
    - int getVertices(): *teszteléshez szükséges, oldalak számának lekérése*
    - void Print()

        - bool testsides(): *teszteléshez szükséges, leellenőrzi, hogy az alakzat szabályos-e*
- **class Triangle:** Háromszögek osztálya
    - **Adattagok**
    - **Függvények**
        - Triangle():Shapes(): *default konstruktor*
        - Triangle(Point& c, Point& p): *centerpont és csúcs által megadott háromszög létrehozása*
        - friend std::ostream& operator<<(std::ostream& out, Square& tri);
        - friend std::istream& operator>>(std::istream& in, Square& tri);
        - Triangle& operator=(const Triangle& tri);
        - bool contained_by_this (Point p): *az adott pontot tartalmazza-e a háromszög*
        - bool contained_by_circle(Point p, double r): *az adott kör tartalmazza-e a háromszöget*
        - int getVertices(): *teszteléshez szükséges, oldalak számának lekérése*
        - void Print()
        - bool testsides(): *teszteléshez szükséges, leellenőrzi, hogy az alakzat szabályos-e*
- **class Tester:** tesztelő osztály
    - **Adattagok**
        - **nincs**
    - **Függvények**
        - Tester(): *default konstruktor*
        - bool testshapes(Shapes** shp, size_t n): *Shapes heterogén kollekción végigmegy és eldönti, hogy a benne található összes alakzat szabályos-e*
        - bool AreSame(double a,double b): *segédfüggvény double-ök összehasonlításához*

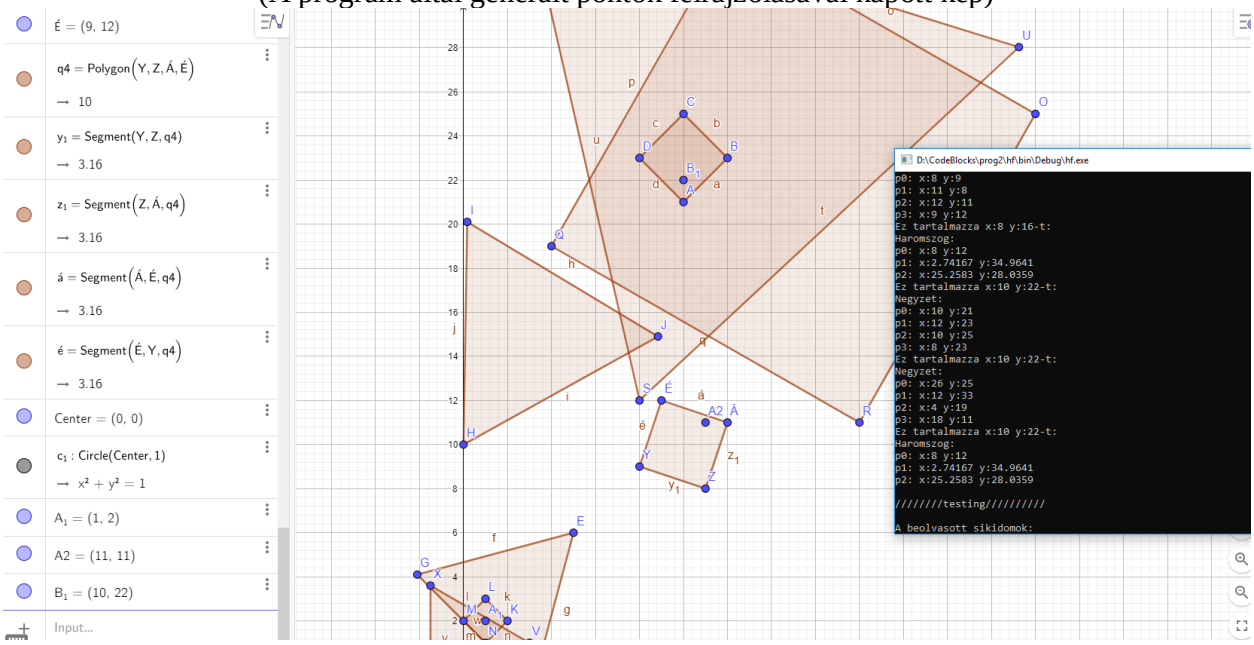
## Tesztelési dokumentáció

A teszteléshez a https://www.geogebra.org/classic?lang=en oldalt használtam, felrajzoltam a fájlban szereplő síkidomokat, majd összevetettem azzal, amit a program számolt ki. Minden esetben egyenlő oldalú, szabályos sokszögeket kaptam. Majd felvettem néhány pontot azok közül, amelyek szerepelnek a fájlban, és megnéztem, hogy mely síkidomok tartalmazzák ezeket a pontokat, és itt is 100%-os egyezést tapasztaltam a program által kiírtakkal. A képeken látható az origó sugarú, egységsugarú kör is, amelyet csak a (3,1),(-1.5,-1.6),(-1.5,3.6) háromszög metsz, és tesztelés során látható, hogy ezt a háromszöget hagyja egyedül ki a program amikor listázza a körön kívül eső síkidomokat. Ezek után megírtam a „testshapes” függvényt, amely ellenőrzi, hogy a beolvasott síkidomok összes pontja ugyanolyan távolságra van-e a középponttól, és ez a teszt is mindig sikeresen fut le. Az alábbi két képen felrajzoltam az összes, fájlban szereplő síkidomot, valamint futtattam a programot ellenőrízve azt, hogy ha több síkidom tartalmazza a pontot, akkor is kilistázza az összeset. (**2.1 és 2.2 ábra**)




**2.1 ábra**

(A program által generált pontok felrajzolásával kapott kép)




**2.2 ábra**

(látható, hogy a (10,22) pontot 2 négyzet és 1 háromszög tartalmazza)