

# Síkidomok

Berghammer Dávid  
EB2DYD

# Feladat

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 egységgörön 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!

(Forrás: [InfoC++](#))

## Specifikáció

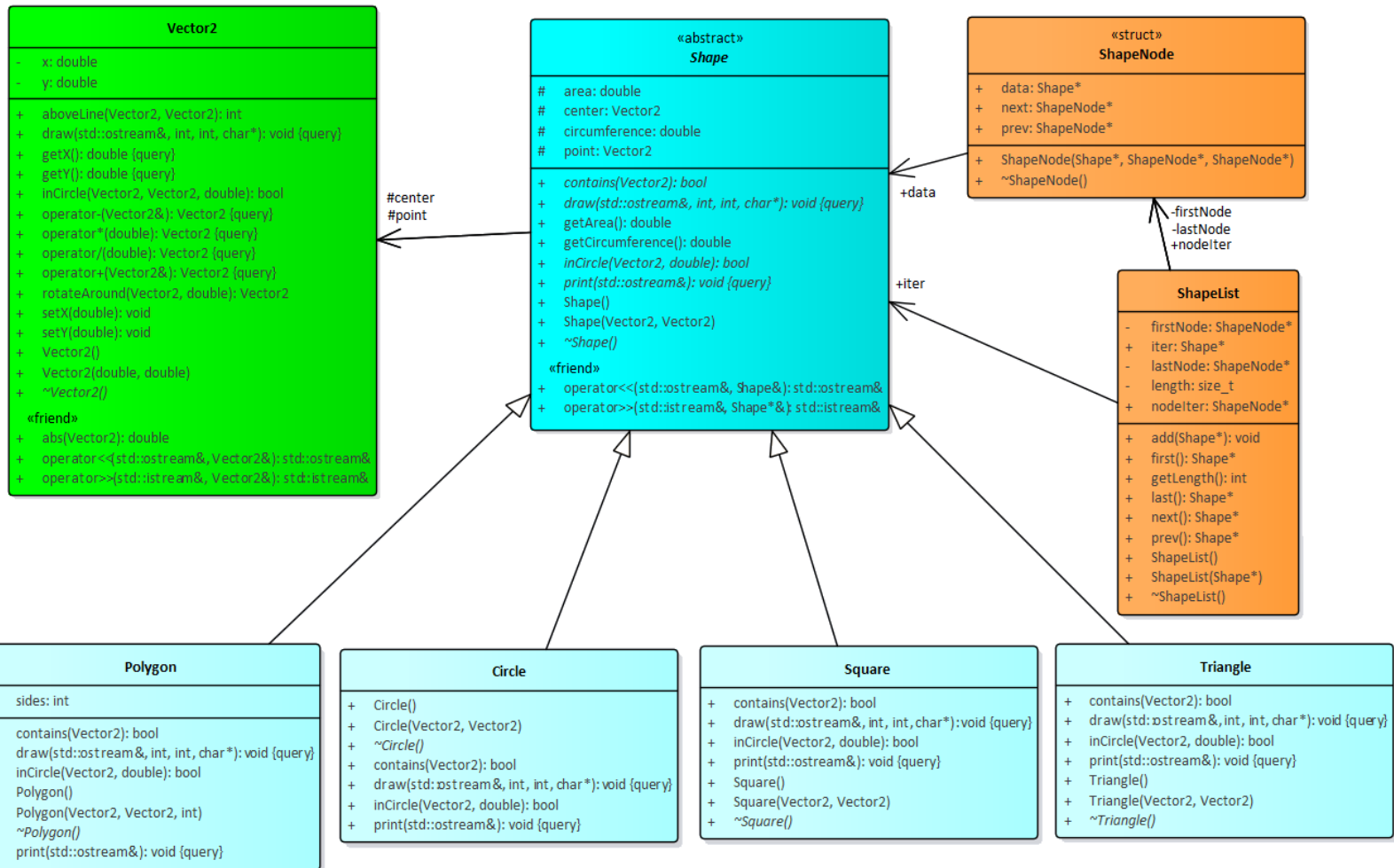
### Az osztály:

- Absztrakt síkidom osztály, aminek a segítségével tetszőleges oldalú szabályos sokszöget valósítok meg.
- A síkidom beolvasható istream típusú objektumról
- A síkidom kiírható ostream típusú objektumra
- A síkidomról eldönthető, hogy egy adott pont a területére esik-e
- A síkidomról eldönthető, hogy egy origó középpontú adott sugarú kör tartalmazza-e
- A síkidomnak lekérdezhető a területe és a kerülete

### A főprogram:

- Egy fájlból {típus, középpont, csúcs} tartalmú sorokat olvas be a fájl végéig
  - Pl.: `square (0,0) (5,5)`  
az origó középpontú négyzet, aminek az egyik csúcsa az (5,5) pontban van
  - Pl.: `polygon (0,0) (5,5) 6`  
az origó középpontú hatszög, aminek az egyik csúcsa az (5,5) pontban van
- A beolvasott síkidomok közül eltárolja azokat, amelyek teljes egészükben az origó középpontú egységgörön kívül esnek
- Ezután a szabványos bemenetről koordinátákat olvas be a fájl végéig, {(x,y)} formátumban
  - Pl.: `(3,4)`
- A program minden beolvasott ponthoz kiírja azon síkidom adatait, amelyek az adott pontot tartalmazzák

# Terv



Az osztálydiagram Enterprise Architect-tel lett generálva forráskódból

# Algoritmusok

## A sokszög tartalmaz-e egy adott pontot:

- A sokszög minden oldalára ellenőrzöm, hogy a kérdéses pont és a sokszög középpontja az oldal által meghatározott egyenes azonos oldalán vannak-e, amennyiben nem a sokszög nem tartalmazza a pontot, ha igen, akkor tartalmazza.

## Egy kör tartalmaz egy sokszöget:

- Egy kör tartalmaz egy szakaszt:
  - Ha a szakasz, valamelyik végpontja a körön belül van, akkor a szakasz is a körön belül van. A kör középpontjából merőlegest állítok a szakasz által meghatározott egyenesre, ha ez az egyenes nem metszi a szakaszt, akkor a szakasz a körön kívül van. Ha az egyenes metszi a szakaszt, akkor, ha a metszéspont a körön belül van, akkor a szakasz is a körön belül van, egyébként a szakasz a körön kívül van.
- Akkor tartalmaz a kör egy sokszöget, ha van közös pontjuk (nem feltétlenül kell, hogy az egész sokszög a körön belül legyen)
- Az előző algoritmussal ellenőrzöm, hogy a kör középpontja a sokszögön belül van-e, ha igen, akkor a sokszög is a körön belül van. Ha nem, akkor a sokszög összes oldalára ellenőrzöm, hogy a körön belül vannak-e, ha egy oldal sincs a körön belül, akkor a sokszög is a körön kívül van, egyébként a sokszög a körön belül van.

# Hierarchikus mutató

## Osztályhierarchia

Majdnem (de nem teljesen) betűrendbe szedett leszármazási lista:

Shape .....	8
Circle .....	6
Polygon.....	7
Square .....	12
Triangle.....	13
 ShapeList .....	 10
Vector2 .....	14

# Osztályok dokumentációja

## Circle osztályreferencia

```
#include <Circle.h>
```

### Publikus tagfüggvények

- **Circle** (**Vector2** center, **Vector2** point)
- bool **inCircle** (**Vector2** c, double r)
- bool **contains** (**Vector2** p)
- void **draw** (std::ostream &os, int shift, int s, const char \*color) const
- void **print** (std::ostream &os) const

### Részletes leírás

Circle osztály Kört tárol

---

## Tagfüggvények dokumentációja

**bool Circle::contains (Vector2 p)[virtual]**

Kör tartalmaz-e egy pontot

#### Paraméterek:

<i>p</i>	- a kérdéses pont
----------	-------------------

#### Visszatérési érték:

tartalmazza-e a pontot

Megvalósítja a következőket: **Shape** (0.8).

**void Circle::draw (std::ostream & os, int shift, int s, const char \* color = "black") const[virtual]**

Kör rajzolása SVG formátumban

#### Paraméterek:

<i>os</i>	- ostream objektum, amire rajzolunk
<i>shift</i>	- eltolás
<i>s</i>	- nagyítás
<i>color</i>	- szín

Megvalósítja a következőket: **Shape** (0.8).

**bool Circle::inCircle (Vector2 c, double r)[virtual]**

Kör benne van-e egy körben

#### Paraméterek:

<i>c</i>	- a kör középpontja
<i>r</i>	- a kör sugara

#### Visszatérési érték:

benne van-e a körben

Megvalósítja a következőket: **Shape** (0.9).

**void Circle::print (std::ostream & os) const[virtual]**

Kör kiírása

#### Paraméterek:

<i>os</i>	- ostream objektum, amire kiírnunk
-----------	------------------------------------

Megvalósítja a következőket: **Shape** (0.9).

# Polygon osztályreferencia

```
#include <Polygon.h>
```

## Publikus tagfüggvények

- **Polygon** (**Vector2** center, **Vector2** point, int sides)
- bool **inCircle** (**Vector2** c, double r)
- bool **contains** (**Vector2** p)
- void **draw** (std::ostream &os, int shift, int s, const char \*color) const
- void **print** (std::ostream &os) const

## Részletes leírás

Sokszög osztály Szabályos sokszöget tárol

---

## Tagfüggvények dokumentációja

**bool Polygon::contains (Vector2 p)[virtual]**

Sokszög tartalmaz-e egy pontot

### Paraméterek:

<i>p</i>	- a kérdéses pont
----------	-------------------

### Visszatérési érték:

tartalmazza-e a pontot

Megvalósítja a következőket: **Shape** (o.8).

**void Polygon::draw (std::ostream & os, int shift, int s, const char \* color = "black") const[virtual]**

Sokszög rajzolása SVG formátumban

### Paraméterek:

<i>os</i>	- ostream objektum, amire rajzolunk
<i>shift</i>	- eltolás
<i>s</i>	- nagyítás
<i>color</i>	- szín

Megvalósítja a következőket: **Shape** (o.8).

**bool Polygon::inCircle (Vector2 c, double r)[virtual]**

Szabályos sokszög benne van-e egy körben

### Paraméterek:

<i>c</i>	- a kör középpontja
<i>r</i>	- a kör sugara

### Visszatérési érték:

benne van-e a körben

Megvalósítja a következőket: **Shape** (o.9).

**void Polygon::print (std::ostream & os) const[virtual]**

Sokszög kiírása

### Paraméterek:

<i>os</i>	- ostream objektum, amire kiírnunk
-----------	------------------------------------

Megvalósítja a következőket: **Shape** (o.9).

# Shape osztályreferencia

```
#include <Shape.h>
```

## Publikus tagfüggvények

- **Shape** (**Vector2** c, **Vector2** point)
- double **getArea** ()
- double **getCircumference** ()
- virtual bool **inCircle** (**Vector2** center, double r)=0
- virtual bool **contains** (**Vector2** p)=0
- virtual void **print** (std::ostream &os) const =0
- virtual void **draw** (std::ostream &os, int shift, int s, const char \*color="black") const =0

## Védett attribútumok

- **Vector2** center  
*Síkidom középpontja.*
- **Vector2** point  
*Síkidom egy csúcsa.*
- double **area**  
*Síkidom területe.*
- double **circumference**  
*Síkidom kerülete.*

## Barátok

- std::ostream & **operator<<** (std::ostream &os, const **Shape** &s)
- std::istream & **operator>>** (std::istream &is, **Shape** \*&s)

---

## Részletes leírás

**Shape** osztály Szabályos síkidomot tárol

---

## Tagfüggvények dokumentációja

**virtual bool Shape::contains (Vector2 p)[pure virtual]**

Síkidom tartalmaz-e egy pontot

**Paraméterek:**

<i>p</i>	- a kérdéses pont
----------	-------------------

**Visszatérési érték:**

tartalmazza-e a pontot

Megvalósítják a következők: **Triangle** (o.13), **Circle** (o.6), **Square** (o.12) és **Polygon** (o.7).

**virtual void Shape::draw (std::ostream & os, int shift, int s, const char \* color = "black") const [pure virtual]**

Síkidom rajzolása SVG formátumban

**Paraméterek:**

<i>os</i>	- ostream objektum, amire rajzolunk
<i>shift</i>	- eltolás
<i>s</i>	- nagyítás
<i>color</i>	- szín

Megvalósítják a következők: **Triangle** (o.13), **Circle** (o.6), **Square** (o.12) és **Polygon** (o.7).



## **double Shape::getArea ()**

Area getter

### **Visszatérési érték:**

a síkidom területe

## **double Shape::getCircumference ()**

Circumference getter

### **Visszatérési érték:**

a síkidom kerülete

## **virtual bool Shape::inCircle (Vector2 center, double r)[pure virtual]**

Síkidom benne van-e egy körben

### **Paraméterek:**

<i>c</i>	- a kör középpontja
<i>r</i>	- a kör sugara

### **Visszatérési érték:**

benne van-e a körben

Megvalósítják a következők: **Triangle** (*o.13*), **Circle** (*o.6*), **Square** (*o.12*) és **Polygon** (*o.7*).

## **virtual void Shape::print (std::ostream & os) const[pure virtual]**

Síkidom kiírása

### **Paraméterek:**

<i>os</i>	- ostream objektum, amire kiírunk
-----------	-----------------------------------

Megvalósítják a következők: **Triangle** (*o.13*), **Circle** (*o.6*), **Square** (*o.12*) és **Polygon** (*o.7*).

---

## **Barát és kapcsolódó függvények dokumentációja**

### **std::ostream& operator<< (std::ostream & os, const Shape & s)[friend]**

<< operator

### **Paraméterek:**

<i>os</i>	- ostream objektum, amire kiírunk
<i>s</i>	- amit kiírunk

### **std::istream& operator>> (std::istream & is, Shape \*& s)[friend]**

>> operator

### **Paraméterek:**

<i>is</i>	- istream objektum, amiről beolvasunk
<i>s</i>	- ahova beolvasunk

## ShapeList osztályreferencia

```
#include <Shape.h>
```

### Publikus tagfüggvények

- **ShapeList** (**Shape** \*s)
- **Shape** \* **first** ()
- **Shape** \* **last** ()
- **Shape** \* **next** ()
- **Shape** \* **prev** ()
- **Shape** \* **get** (size\_t idx)
- void **add** (**Shape** \*s)
- int **getLength** ()

### Publikus attribútumok

- **ShapeNode** \* **nodeIter**  
*Lista elemére mutató iterátor.*
- **Shape** \* **iter**  
*Lista adattagjára mutató iterátor.*

---

## Részletes leírás

**ShapeList** osztály Duplán láncolt lista, amiben síkidomokat lehet tárolni

---

### Tagfüggvények dokumentációja

**void ShapeList::add (Shape \* s)**

Új elem hozzáadása a listához

**Paraméterek:**

<i>s</i>	- hozzáadandó elem
----------	--------------------

**Shape \* ShapeList::first ()**

Iterátor mozgatása az első elemre

**Visszatérési érték:**

mutató a lista első elemére

**Shape \* ShapeList::get (size\_t idx)**

A lista tetszőleges indexű elemének lekérése

**Paraméterek:**

<i>idx</i>	- az elem indexe
------------	------------------

**Visszatérési érték:**

mutató a lista megfelelő elemére

**int ShapeList::getLength ()**

Length getter

**Visszatérési érték:**

a lista hossza

**Shape \* ShapeList::last ()**

Iterátor mozgatása az utolsó elemre

**Visszatérési érték:**

mutató a lista utolsó elemére

**Shape \* ShapeList::next ()**

Iterátor mozgatása a következő elemre

**Visszatérési érték:**

mutató a lista következő elemére

**Shape \* ShapeList::prev ()**

Iterátor mozgatása az előző elemre

**Visszatérési érték:**

mutató a lista előző elemére

# Square osztályreferencia

```
#include <Square.h>
```

## Publikus tagfüggvények

- **Square** (**Vector2** center, **Vector2** point)
- bool **inCircle** (**Vector2** c, double r)
- bool **contains** (**Vector2** p)
- void **draw** (std::ostream &os, int shift, int s, const char \*color) const
- void **print** (std::ostream &os) const

## Részletes leírás

Square osztály Négyzetet tárol

---

## Tagfüggvények dokumentációja

**bool Square::contains (Vector2 p)[virtual]**

Négyzet tartalmaz-e egy pontot

### Paraméterek:

<i>p</i>	- a kérdéses pont
----------	-------------------

### Visszatérési érték:

tartalmazza-e a pontot

Megvalósítja a következőket: **Shape** (o.8).

**void Square::draw (std::ostream & os, int shift, int s, const char \* color = "black") const[virtual]**

Négyzet rajzolása SVG formátumban

### Paraméterek:

<i>os</i>	- ostream objektum, amire rajzolunk
<i>shift</i>	- eltolás
<i>s</i>	- nagyítás
<i>color</i>	- szín

Megvalósítja a következőket: **Shape** (o.8).

**bool Square::inCircle (Vector2 c, double r)[virtual]**

Négyzet benne van-e egy körben

### Paraméterek:

<i>c</i>	- a kör középpontja
<i>r</i>	- a kör sugara

### Visszatérési érték:

benne van-e a körben

Megvalósítja a következőket: **Shape** (o.9).

**void Square::print (std::ostream & os) const[virtual]**

Négyzet kiírása

### Paraméterek:

<i>os</i>	- ostream objektum, amire kiírnunk
-----------	------------------------------------

Megvalósítja a következőket: **Shape** (o.9).

# Triangle osztályreferencia

```
#include <Triangle.h>
```

## Publikus tagfüggvények

- **Triangle** (**Vector2** center, **Vector2** point)
- **bool inCircle** (**Vector2** c, double r)
- **bool contains** (**Vector2** p)
- **void draw** (std::ostream &os, int shift, int s, const char \*color) const
- **void print** (std::ostream &os) const

## Részletes leírás

**Triangle** osztály Szabályos háromszöget tárol

---

## Tagfüggvények dokumentációja

**bool Triangle::contains (Vector2 p)[virtual]**

Háromszög tartalmaz-e egy pontot

### Paraméterek:

<i>p</i>	- a kérdéses pont
----------	-------------------

### Visszatérési érték:

tartalmazza-e a pontot

Megvalósítja a következőket: **Shape** (o.8).

**void Triangle::draw (std::ostream & os, int shift, int s, const char \* color = "black") const[virtual]**

Háromszög rajzolása SVG formátumban

### Paraméterek:

<i>os</i>	- ostream objektum, amire rajzolunk
<i>shift</i>	- eltolás
<i>s</i>	- nagyítás
<i>color</i>	- szín

Megvalósítja a következőket: **Shape** (o.8).

**bool Triangle::inCircle (Vector2 c, double r)[virtual]**

Háromszög benne van-e egy körben

### Paraméterek:

<i>c</i>	- a kör középpontja
<i>r</i>	- a kör sugara

### Visszatérési érték:

benne van-e a körben

Megvalósítja a következőket: **Shape** (o.9).

**void Triangle::print (std::ostream & os) const[virtual]**

Háromszög kiírása

### Paraméterek:

<i>os</i>	- ostream objektum, amire kiírnunk
-----------	------------------------------------

Megvalósítja a következőket: **Shape** (o.9).

## Vector2 osztályreferencia

```
#include <Vector2.h>
```

### Publikus tagfüggvények

- **Vector2** (double x, double y)
- double **getX** () const
- double **getY** () const
- void **setX** (double x)
- void **setY** (double y)
- int **aboveLine** (Vector2 p1, Vector2 p2)
- bool **inCircle** (Vector2 p1, Vector2 o, double r)
- **Vector2 rotateAround** (Vector2 p, double angle)
- **Vector2 operator-** (const Vector2 &v) const
- **Vector2 operator+** (const Vector2 &v) const
- **Vector2 operator/** (double d) const
- **Vector2 operator\*** (double d) const
- bool **operator==** (Vector2 &v) const
- void **draw** (std::ostream &os, int shift, int s, const char \*color) const

### Barátok

- double **abs** (Vector2 p)
- std::istream & **operator>>** (std::istream &is, Vector2 &s)
- std::ostream & **operator<<** (std::ostream &os, const Vector2 &s)

---

## Részletes leírás

**Vector2** osztály Kétdimenziós vektort tárol

---

### Tagfüggvények dokumentációja

**int Vector2::aboveLine (Vector2 p1, Vector2 p2)**

A pont egy tetszőleges egyenes felett van-e

**Paraméterek:**

<i>p1</i>	- az egyenes 1. pontja
<i>p2</i>	- az egyenes 2. pontja

**Visszatérési érték:**

1 ha a pont az egyenes felett van, -1 ha az egyenes alatt, 0 ha az egyenesen

**void Vector2::draw (std::ostream & os, int shift, int s, const char \* color) const**

Vektor rajzolása SVG formátumban

**Paraméterek:**

<i>os</i>	- ostream objektum, amire rajzolunk
<i>shift</i>	- eltolás
<i>s</i>	- nagyítás
<i>color</i>	- szín

**double Vector2::getX () const**

X getter

**Visszatérési érték:**

a vektor X koordinátája

**double Vector2::getY () const**

Y getter

**Visszatérési érték:**

a vektor Y koordinátája

**bool Vector2::inCircle (Vector2 p1, Vector2 o, double r)**

Egy szakasz egy tetszőleges körön belül van-e

**Paraméterek:**

<i>p1</i>	- a szakasz végpontja
<i>o</i>	- a kör középpontja
<i>r</i>	- a kör sugara

**Visszatérési érték:**

A szakasz a körön belül van-e

**Vector2 Vector2::operator\* (double d) const**

- operator

• **Paraméterek:**

<i>d</i>	- valós szorzó
----------	----------------

• **Visszatérési érték:**

A vektor d-szerese

•

**Vector2 Vector2::operator+ (const Vector2 & v) const**

- operator

• **Paraméterek:**

<i>v</i>	- vektor
----------	----------

• **Visszatérési érték:**

A két vektor összege

•

**Vector2 Vector2::operator- (const Vector2 & v) const**

- operator

• **Paraméterek:**

<i>v</i>	- vektor
----------	----------

• **Visszatérési érték:**

A két vektor különbsége

**Vector2 Vector2::operator/ (double d) const**

/ operator

**Paraméterek:**

<i>d</i>	- valós szorzó
----------	----------------

**Visszatérési érték:**

A vektor 1/d-szerese

**bool Vector2::operator==(Vector2 & v) const**

== operator

**Paraméterek:**

<i>v</i>	- összehasonlítandó vektor
----------	----------------------------

**Visszatérési érték:**

A két vektor azonos-e

**Vector2 Vector2::rotateAround(Vector2 p, double angle)**

A pont elforgatása egy másik pont körül

**Paraméterek:**

<i>p</i>	- e körül a pont körül forgatunk
<i>angle</i>	- az elforgatás szöge radiánban

**Visszatérési érték:**

Az elforgatott pont

**void Vector2::setX(double x)**

X setter

**Paraméterek:**

<i>x</i>	- a vektor új X koordinátája
----------	------------------------------

**void Vector2::setY(double y)**

Y setter

**Paraméterek:**

<i>y</i>	- a vektor új Y koordinátája
----------	------------------------------

---

## Barát és kapcsolódó függvények dokumentációja

**double abs(Vector2 p)[friend]**

Vektor hosszának kiszámítása

**Paraméterek:**

<i>p</i>	- a vektor
----------	------------

**Visszatérési érték:**

A vektor hossza

**std::ostream& operator<<(std::ostream & os, const Vector2 & s)[friend]**

<< operator

**Paraméterek:**

<i>os</i>	- ostream objektum, amire kiírnunk
<i>s</i>	- amit kiírnunk

**std::istream& operator>>(std::istream & is, Vector2 & s)[friend]**

>> operator

**Paraméterek:**

<i>is</i>	- istream objektum, amiről beolvasunk
<i>s</i>	- ahova beolvasunk



# Tesztelés

## Lista.Elemszam

- Lista elemszámának, és túlindexelésének tesztelése, a listához 4 elemet adok, majd megpróbálom lekérni a 10. elemet.
- Várt érték: 4, out\_of\_range kivétel

## Negyzet.Terulet-Kerulet

- Az előző tesztnél létrehozott lista 2. elemének ((1,1) középpontú 2 oldalú négyzet) a területét és kerületét kérdezi le
- Várt érték: Terület: 4, Kerület: 8

## Kor.Terulet-Kerulet

- Létrehoz egy új (1,1) középpontú 1 sugarú kört, lekérdezi a területét és kerületét
- Várt érték: Terület: pi, Kerület: 2\*pi

## Haromszog.Terulet-Kerulet

- Létrehoz egy új (0,0) középpontú (0,2) csúcsú háromszöget, lekérdezi a területét és kerületét
- Várt érték: Terület:  $\frac{1}{4}\sqrt{3}$ , Kerület:  $6\sqrt{3}$

## Sokszog.Terulet-Kerulet

- Létrehoz egy új (0,0) középpontú (0,2) csúcsú szabályos hatszöget, lekérdezi a területét és kerületét
- Várt érték: Terület:  $6\sqrt{3}$ , Kerület: 4

## Alakzatok.Beolvas-kiir

- stringstream objektumról olvas be különböző síkidomokat, majd kiírja azokat
- Várt kimenet:  
Triangle (-1,4) (2,3,1)  
Square (2,4,2) (3,1,-1.1)  
Polygon (4,2) (-2,1.5) 7

## Vektor.X-Y-abs

- A vektor objektum gettereit és abs függvényét teszteli
- Várt érték: X: 2, Y: 3, abs: sqrt(13)

## Vektor.Operatorok

- A vektor +, -, \* és / operátorait teszteli az (1,2) vektorral
- Bemenet: +: (2,2), -: (1,2), \*: 2, /: 2
- Várt érték: +: (3,4), -: (0,0), \*: (2,4), /: (0.5, 1)

## Vektor.aboveLine

- Teszteli, hogy a v vektor az adott egyenesek felett van-e
- Bemenet / Várt érték: (0,0), (5,0) / 1  
(0,5), (5,5) / -1  
(0,0), (4,6) / 0

## Vektor.beolvasas

- stringstream objektumról olvas be vektort
- Várt érték: (3,4)

Az összes teszt eset hiba nélkül, és a várt eredményekkel lefutott.

# Memóriakezelés tesztje

A memóriakezelés ellenőrzését Valgrind segítségével végeztem, nem tapasztaltam memóriaszivárgást a futtatás során.

A Valgrind kimenete:

```
==89== LEAK SUMMARY:
==89==    definitely lost: 0 bytes in 0 blocks
==89==    indirectly lost: 0 bytes in 0 blocks
==89==    possibly lost: 0 bytes in 0 blocks
==89==    still reachable: 72,704 bytes in 1 blocks
==89==    suppressed: 0 bytes in 0 blocks
==89==
==89== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
==89== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```