Terv - Síkidomok

Specifikáció

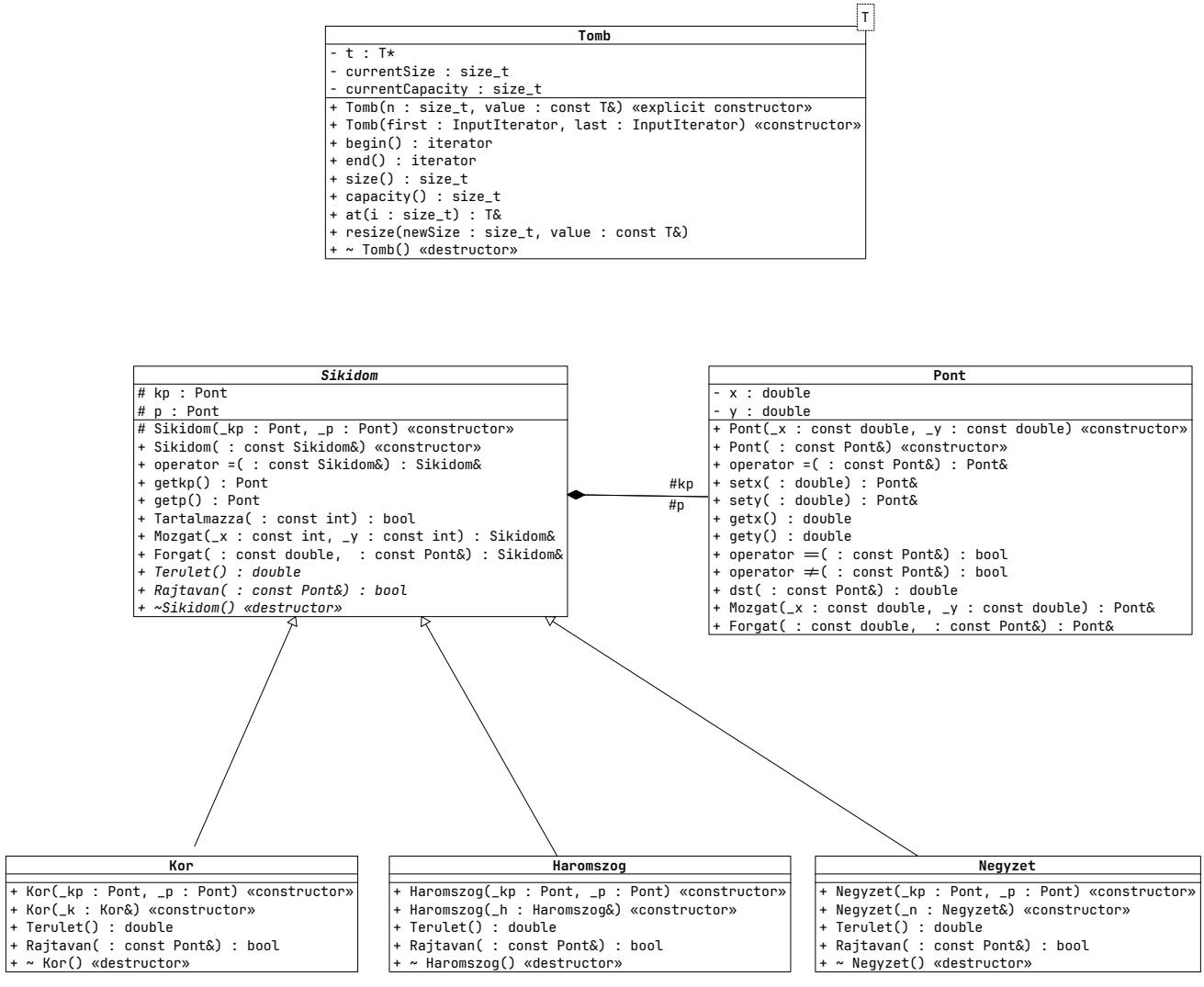
Feladat:

Egy absztrakt síkidom osztály, aminek segítségével szabályos háromszöget, négyzetet és kört valósíthatunk meg! 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 van olyan metódusa, amellyel eldönthető, hogy egy adott pont a síkidom területére esik-e. Van továbbá olyan metódusuk is, ami megadja, hogy tartalmazza-e azokat egy adott sugarú, origó középpontú kör.
A megoldás nem használ STL tárolót.

Működés:

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 olvas be a szabványos bemenetről a fájl végéig, és kiírja az egyes pontokhoz azon eltárolt síkidomok adatait (típus, középpont, csúcs), amelyek az adott pontot tartalmazzák. Azokat a síkidomokat, amikben benne van a pont, amit a felhasználó ad meg, megjelöli, és a program végén kiírja egy másik fájlba.

```
prog2/nhf/sikidom
❱ sikidom beolvasando.txt
"beolvasando.txt" fajl beolvasasa...
{Kor, (0,0), (5,5)}: nincs eltarolva.
{Kor, (5,0), (5,5)}: nincs eltarolva.
{Kor, (0,5), (0,2)}: eltarolva.
{Negyzet, (3,3), (4,4)}: eltarolva.
{Haromszog, (-8,-3), (-8,-4)}: eltarolva.
{Negyzet, (-1,1), (3,-2)}: nincs eltarolva.
fajl vege.

Adjon meg koordinatakat! (formatum: "x,y")
0 23

Formatum hiba!
Adjon meg koordinatakat! (formatum: "x,y")
0,23

A (0,23) pontot nem tartalmazza egyetlen eltarolt sikidom sem.

Adjon meg koordinatakat! (formatum: "x,y")
3,3

A (3,3) pontot tartalmazza:
Negyzet, (3,3), (4,4)
Adjon meg koordinatakat! (formatum: "x,y")
^D
```



Fontosabb függvények, algoritmusok:

- a Síkidomok területét megadó tagfüggvény a matek órán tanult triviális képletekkel dolgozik.

- A Síikidomok forgatása forgatási mátrixszal történik.

$$R_\alpha = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}$$

Egy pont rajta van-e egy adott síkidomon?
- Kör:
A vizsgált pontnak a kör középpontjától vett távolságát vetjük össze a kör egy pontjáéval. Ha kisebb (vagy egyenlő), rajta van, ha nagyobb, nincs.

- Négyzet:
A négyzet és a pont egy-egy másolatát elforgatjuk a négyzet középpontja körül úgy, hogy a négyzet „egyenes legyen”, azaz oldalai párhuzamosak legyenek vagy az x, vagy az y tengellyel. Ezután még 3 forgatással megkaphatjuk a többi csúcsot is. Ezek x valamint y koordinátái közé kell hogy essen a vizsgált pont koordinátája, ekkor rajta van a négyzeten.

- Háromszög:
Van egy nagyon jó és gyors módszer arra, hogy ezt meghatározzuk: https://en.wikipedia.org/wiki/Barycentric_coordinate_system

Azon alapul, hogy a sík pontjait a háromszög csúcsaihoz relatív koordinátákkal írjuk fel. Egy pontnak 3 ilyen koordinátája van. Ha ezek közül mindhárom nemnegatív, akkor a pont a háromszögön belül van. A függvény pszeudokódja:

```
t = terulet
a = |PB x PA| / 2t
b = |PC x PA| / 2t
c = |PA x PB| / 2t
return ( a >= 0 && b >= 0 && c >= 0)
```

Itt fontos megjegyezni, hogy a vektorok vektoriális szorzatát jóval lassabb kiszámolni, mint annak az abszolútértékét. Tehát előjellel ne számoljuk ki, nincs rá szükség.

sikidom

1.0.0

Készítette Doxygen 1.10.0

1. fejezet

Hierarchikus mutató

1.1. Osztályhierarchia

Majdnem (de nem teljesen) betűrendbe szedett leszármazási lista:

2. fejezet

Osztálymutató

2.1. Osztálylista

Az összes osztály, struktúra, unió és interfész listája rövid leírásokkal:

3. fejezet

Fájlmutató

3.1. Fájllista

Az összes fájl listája rövid leírásokkal:

4. fejezet

Osztályok dokumentációja

4.1. Tomb< T >::const_iterator osztályreferencia

const_iterator osztály. Az const_iterator osztály a Tomb osztály const_iteratora.

`#include <tomb.hpp>`

Publikus tagfüggvények

- const_iterator (const T *p=nullptr, size_t idx=0)
  const_iterator konstruktora.
- const T & operator* () const
  operator függvény.*
- const_iterator & operator++ ()
  operator++ függvény.
- bool operator!= (const const_iterator &other) const
  operator!= függvény.

Privát attribútumok

- const T * p
  Az const_iterator aktuális elemre mutató pointere.
- size_t idx
  Az const_iterator aktuális indexe.

4.1.1. Részletes leírás

template<class T>
class Tomb< T >::const_iterator

const_iterator osztály. Az const_iterator osztály a Tomb osztály const_iteratora.

4.1.2. Konstruktorok és destruktorok dokumentációja

4.1.2.1. const_iterator()

```
template<class T >
Tomb< T >::const_iterator::const_iterator (
            const T * p = nullptr,
            size_t idx = 0 )  [inline]
```

const_iterator konstruktora.

Paraméterek

p	Az const_iterator aktuális elemre mutató pointere.
idx	Az const_iterator aktuális indexe.

4.1.3. Tagfüggvények dokumentációja

4.1.3.1. operator"!=()

```
template<class T >
bool Tomb< T >::const_iterator::operator!= (
            const const_iterator & other ) const  [inline]
```

operator!= függvény.

Paraméterek

other	A másik const_iterator.

Visszatérési érték

Igaz, ha a ket const_iterator nem egyezik meg, egyébként hamis.

4.1.3.2. operator∗()

```
template<class T >
const T & Tomb< T >::const_iterator::operator* ( ) const  [inline]
```

operator∗ függvény.

Visszatérési érték

Az const_iterator aktuális elemre mutató referencia.

4.1.3.3. operator++()

```
template<class T >
const_iterator & Tomb< T >::const_iterator::operator++ ( )  [inline]
```

operator++ függvény.

Visszatérési érték

Az const_iterator az aktuális elem után mutató referencia.

4.1.4. Adattagok dokumentációja

4.1.4.1. idx

```
template<class T >
size_t Tomb< T >::const_iterator::idx  [private]
```

Az const_iterator aktuális indexe.

4.1.4.2. p

```
template<class T >
const T* Tomb< T >::const_iterator::p  [private]
```

Az const_iterator aktuális elemre mutató pointere.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

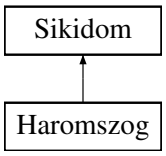
- src/tomb.hpp

4.2. Haromszog osztályreferencia

Haromszog osztály. A Haromszog osztály a Sikidom leszármazottja, mely egy háromszög síkidomot reprezentál, középpontjával és egy csúcsával.

```
#include <sikidom.h>
```

A Haromszog osztály származási diagramja:



Publikus tagfüggvények

- Haromszog (Pont _kp=Pont(0, 0), Pont _p=Pont(0, 0))
- Haromszog (const Haromszog &_h)
- double Terulet () const override

  *Haromszog területét számoló függvény. A kiszamitas modja: R^2 * 3 * sqrt(3) / 4.*
- bool Rajtavan (const Pont &) const override

  Haromszog Rajtavan függvény. A függvény eldönti, hogy egy pont rajta van-e a háromszögön.
- void Write (std::ostream &os) const override

  Haromszog kiíró függvény. A függvény kiírja a háromszög adatait a megadott output streambe.
- void Read (std::istream &is) override

  Haromszog beolvasó függvény. A függvény beolvassa a háromszög adatait a megadott input streamből.
- bool Kivul (const std::size_t r) const override

  Haromszog Kivul függvény. Eldönti, hogy a háromszög kívül van-e egy adott sugarú, origo középpontú körön.
- ~Haromszog () override

Publikus tagfüggvények a(z) Sikidom osztályból származnak

- Sikidom (const Sikidom &)
- Sikidom & operator= (const Sikidom &)

  Sikidom osztaly egyenloseg operatora.
- Pont getkp () const
- Pont getp () const
- bool Tartalmazza (const int) const
- Sikidom & Mozgat (const int _x, const int _y)

  Egy sokszög mozgatását teszi lehetővé.
- Sikidom & Forgat (const double, const Pont &)

  Egy sokszög forgatását teszi lehetővé.
- virtual ~Sikidom ()=0

  Sikidom osztály destruktora. nincs dinamikusan foglalt adattag, tehát nem kell semmit felszabadítani.

Barátok

- std::ostream & operator<< (std::ostream &, const Haromszog &)

További örökölt tagok

Statikus publikus tagfüggvények a(z) Sikidom osztályból származnak

- static Sikidom * createSikidom (const std::string &type)

  Sikidom osztály statikus factory függvénye. A függvény létrehoz egy új síkidomot a megadott típus alapján.

Védett tagfüggvények a(z) Sikidom osztályból származnak

- Sikidom (Pont _kp=Pont(0, 0), Pont _p=Pont(0, 0))

  Sikidom konstruktor.

Védett attribútumok a(z) Sikidom osztályból származnak

- Pont kp

  Középpont.
- Pont p

  A síkidom egy csúcsa.

4.2.1. Részletes leírás

Haromszog osztály. A Haromszog osztály a Sikidom leszármazottja, mely egy háromszög síkidomot reprezentál, középpontjával és egy csúcsával.

4.2.2. Konstruktorok és destruktorok dokumentációja

4.2.2.1. Haromszog() [1/2]

```
Haromszog::Haromszog (
            Pont _kp = Pont(0, 0),
            Pont _p = Pont(0, 0) )  [inline]
```

4.2.2.2. Haromszog() [2/2]

```
Haromszog::Haromszog (
            const Haromszog & _h )  [inline]
```

4.2.2.3. ∼Haromszog()

```
Haromszog::~Haromszog ( )  [inline], [override]
```

4.2.3. Tagfüggvények dokumentációja

4.2.3.1. Kivul()

```
bool Haromszog::Kivul (
            const std::size_t r = 1 ) const  [override], [virtual]
```

Haromszog Kivul függvény. Eldönti, hogy a háromszög kívül van-e egy adott sugarú, origo középpontú körön.

Paraméterek

r	a kör sugara.

Visszatérési érték

true, ha a háromszög kívül van a körön, egyébként false.

Megvalósítja a következőket: Sikidom.

4.2.3.2. Rajtavan()

```
bool Haromszog::Rajtavan (
            const Pont & P ) const  [override], [virtual]
```

Haromszog Rajtavan függvény. A függvény eldönti, hogy egy pont rajta van-e a háromszögön.

Paraméterek

P	a vizsgált pont.

Visszatérési érték

true, ha a pont rajta van a háromszögön, egyébként false.

Megvalósítja a következőket: Sikidom.

4.2.3.3. Read()

```
void Haromszog::Read (
            std::istream & is )  [override], [virtual]
```

Haromszog beolvasó függvény. A függvény beolvassa a háromszög adatait a megadott input streamből.

Paraméterek

is	a bemeneti stream.

Megvalósítja a következőket: Sikidom.

4.2.3.4. Terulet()

```
double Haromszog::Terulet ( ) const  [override], [virtual]
```

Haromszog területét számoló függvény. A kiszamitas modja: R^2 * 3 * sqrt(3) / 4.

Visszatérési érték

A háromszög területe.

Megvalósítja a következőket: Sikidom.

4.2.3.5. Write()

```
void Haromszog::Write (
            std::ostream & os ) const  [override], [virtual]
```

Haromszog kiíró függvény. A függvény kiírja a háromszög adatait a megadott output streambe.

Paraméterek

os	a kimeneti stream.

Megvalósítja a következőket: Sikidom.

4.2.4. Barát és kapcsolódó függvények dokumentációja

4.2.4.1. operator<<

```
std::ostream & operator<< (
            std::ostream & ,
            const Haromszog &  )  [friend]
```

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- src/sikidom.h
- src/sikidom.cpp

4.3. Tomb< T >::iterator osztályreferencia

iterator osztály. Az iterator osztály a Tomb osztály iteratora.

`#include <tomb.hpp>`

Publikus tagfüggvények

- iterator (T *p=nullptr, size_t idx=0)
  iterator konstruktora.
- T & operator* ()
  operator függvény.*
- iterator & operator++ ()
  operator++ függvény.
- bool operator!= (const iterator &other) const
  operator!= függvény.

Privát attribútumok

- T * p
  Az iterator aktuális elemre mutató pointere.
- size_t idx
  Az iterator aktuális indexe.

4.3.1. Részletes leírás

template<class T>
class Tomb< T >::iterator

iterator osztály. Az iterator osztály a Tomb osztály iteratora.

4.3.2. Konstruktorok és destruktorok dokumentációja

4.3.2.1. iterator()

```
template<class T >
Tomb< T >::iterator::iterator (
            T * p = nullptr,
            size_t idx = 0 )  [inline]
```

iterator konstruktora.

Paraméterek

p	Az iterator aktuális elemre mutató pointere.
idx	Az iterator aktuális indexe.

4.3.3. Tagfüggvények dokumentációja

4.3.3.1. operator"!=()

```
template<class T >
bool Tomb< T >::iterator::operator!= (
            const iterator & other ) const  [inline]
```

operator!= függvény.

Paraméterek

other	A másik iterator.

Visszatérési érték

Igaz, ha a két iterator nem egyezik meg, egyébként hamis.

4.3.3.2. operator∗()

```
template<class T >
T & Tomb< T >::iterator::operator* ( )  [inline]
```

operator∗ függvény.

Visszatérési érték

Az iterator aktuális elemre mutató referencia.

4.3.3.3. operator++()

```
template<class T >
iterator & Tomb< T >::iterator::operator++ ( )  [inline]
```

operator++ függvény.

Visszatérési érték

Az iterator az aktuális elem után mutató referencia.

4.3.4. Adattagok dokumentációja

4.3.4.1. idx

```
template<class T >
size_t Tomb< T >::iterator::idx  [private]
```

Az iterator aktuális indexe.

4.3.4.2. p

```
template<class T >
T* Tomb< T >::iterator::p  [private]
```

Az iterator aktuális elemre mutató pointere.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

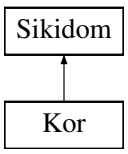
- src/tomb.hpp

4.4. Kor osztályreferencia

Kor osztály. A Kor osztály a Sikidom leszármazottja, mely egy kör síkidomot reprezentál, középpontjával és egy csúcsával.

```
#include <sikidom.h>
```

A Kor osztály származási diagramja:



Publikus tagfüggvények

- Kor (Pont _kp=Pont(0, 0), Pont _p=Pont(0, 0))
- Kor (const Kor &_k)
- double Terulet () const override

  *Kor területét számoló függvény. A kiszamitas modja: r^2 * pi.*
- bool Rajtavan (const Pont &) const override

  Kor Rajtavan függvény. A függvény eldönti, hogy egy pont rajta van-e a körön.
- void Write (std::ostream &os) const override

  Kor kiíró függvény. A függvény kiírja a kör adatait a megadott output streambe.
- void Read (std::istream &is) override

  Kor beolvasó függvény. A függvény beolvassa a kör adatait a megadott input streamből.
- bool Kivul (const std::size_t r) const override

  Kor kivül függvény. Eldönti, hogy a kör kívül van-e egy adott sugarú, origo középpontú körön.
- ~Kor () override

Publikus tagfüggvények a(z) Sikidom osztályból származnak

- Sikidom (const Sikidom &)
- Sikidom & operator= (const Sikidom &)

  Sikidom osztaly egyenloseg operatora.
- Pont getkp () const
- Pont getp () const
- bool Tartalmazza (const int) const
- Sikidom & Mozgat (const int _x, const int _y)

  Egy sokszög mozgatását teszi lehetővé.
- Sikidom & Forgat (const double, const Pont &)

  Egy sokszög forgatását teszi lehetővé.
- virtual ~Sikidom ()=0

  Sikidom osztály destruktora. nincs dinamikusan foglalt adattag, tehát nem kell semmit felszabadítani.

Barátok

- std::ostream & operator<< (std::ostream &, const Kor &)

További örökölt tagok

Statikus publikus tagfüggvények a(z) Sikidom osztályból származnak

- static Sikidom * createSikidom (const std::string &type)

  Sikidom osztály statikus factory függvénye. A függvény létrehoz egy új síkidomot a megadott típus alapján.

Védett tagfüggvények a(z) Sikidom osztályból származnak

- Sikidom (Pont _kp=Pont(0, 0), Pont _p=Pont(0, 0))

  Sikidom konstruktor.

Védett attribútumok a(z) Sikidom osztályból származnak

- Pont kp

  Középpont.
- Pont p

  A síkidom egy csúcsa.

4.4.1. Részletes leírás

Kor osztály. A Kor osztály a Sikidom leszármazottja, mely egy kör síkidomot reprezentál, középpontjával és egy csúcsával.

4.4.2. Konstruktorok és destruktorok dokumentációja

4.4.2.1. Kor() [1/2]

```
Kor::Kor (
            Pont _kp = Pont(0, 0),
            Pont _p = Pont(0, 0) )  [inline]
```

4.4.2.2. Kor() [2/2]

```
Kor::Kor (
            const Kor & _k )  [inline]
```

4.4.2.3. ∼Kor()

```
Kor::∼Kor ( )  [inline], [override]
```

4.4.3. Tagfüggvények dokumentációja

4.4.3.1. Kivul()

```
bool Kor::Kivul (
            const std::size_t r ) const  [override], [virtual]
```

Kor kivül függvény. Eldönti, hogy a kör kívül van-e egy adott sugarú, origo középpontú körön.

Paraméterek

r	a kör sugara.

Megvalósítja a következőket: Sikidom.

4.4.3.2. Rajtavan()

```
bool Kor::Rajtavan (
            const Pont & _p ) const  [override], [virtual]
```

Kor Rajtavan függvény. A függvény eldönti, hogy egy pont rajta van-e a körön.

Paraméterek

↩ _↩ *p*	a vizsgált pont.

Visszatérési érték

true, ha a pont rajta van a körön, egyébként false.

Megvalósítja a következőket: Sikidom.

4.4.3.3. Read()

```
void Kor::Read (
            std::istream & is )  [override], [virtual]
```

Kor beolvasó függvény. A függvény beolvassa a kör adatait a megadott input streamből.

Paraméterek

is	a bemeneti stream.

Megvalósítja a következőket: Sikidom.

4.4.3.4. Terulet()

```
double Kor::Terulet ( ) const  [override], [virtual]
```

Kor területét számoló függvény. A kiszamitas modja: r^2 ∗ pi.

Visszatérési érték

A kör területe.

Megvalósítja a következőket: Sikidom.

4.4.3.5. Write()

```
void Kor::Write (
            std::ostream & os ) const  [override], [virtual]
```

Kor kiíró függvény. A függvény kiírja a kör adatait a megadott output streambe.

Paraméterek

os	a kimeneti stream.

Megvalósítja a következőket: Sikidom.

4.4.4. Barát és kapcsolódó függvények dokumentációja

4.4.4.1. operator<<

```
std::ostream & operator<< (
            std::ostream & ,
            const Kor &  )  [friend]
```

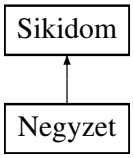
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- src/sikidom.h
- src/sikidom.cpp

4.5. Negyzet osztályreferencia

`#include <sikidom.h>`

A Negyzet osztály származási diagramja:



Publikus tagfüggvények

- Negyzet (Pont _kp=Pont(0, 0), Pont _p=Pont(0, 0))
- Negyzet (const Negyzet &_n)
- double Terulet () const override

  *Negyzet területét számoló függvény. A kiszamitas modja: R^2 * 2.*
- bool Rajtavan (const Pont &) const override

  Negyzet Rajtavan függvény. A függvény eldönti, hogy egy pont rajta van-e a négyzeten.
- void Write (std::ostream &os) const override

  Negyzet kiíró függvény. A függvény kiírja a négyzet adatait a megadott output streambe.
- void Read (std::istream &is) override

  Negyzet beolvasó függvény. A függvény beolvassa a négyzet adatait a megadott input streamből.
- bool Kivul (const std::size_t r) const override

  Negyzet Kivul függvény. Eldönti, hogy a négyzet kívül van-e egy adott sugarú, origo középpontú körön.
- ~Negyzet () override

Publikus tagfüggvények a(z) Sikidom osztályból származnak

- Sikidom (const Sikidom &)
- Sikidom & operator= (const Sikidom &)

  Sikidom osztaly egyenloseg operatora.
- Pont getkp () const
- Pont getp () const
- bool Tartalmazza (const int) const
- Sikidom & Mozgat (const int _x, const int _y)

  Egy sokszög mozgatását teszi lehetővé.
- Sikidom & Forgat (const double, const Pont &)

  Egy sokszög forgatását teszi lehetővé.
- virtual ~Sikidom ()=0

  Sikidom osztály destruktora. nincs dinamikusan foglalt adattag, tehát nem kell semmit felszabadítani.

Barátok

- std::ostream & operator<< (std::ostream &, const Negyzet &)

További örökölt tagok

Statikus publikus tagfüggvények a(z) Sikidom osztályból származnak

- static Sikidom * createSikidom (const std::string &type)

  Sikidom osztály statikus factory függvénye. A függvény létrehoz egy új síkidomot a megadott típus alapján.

Védett tagfüggvények a(z) Sikidom osztályból származnak

- Sikidom (Pont _kp=Pont(0, 0), Pont _p=Pont(0, 0))

  Sikidom konstruktor.

Védett attribútumok a(z) Sikidom osztályból származnak

- Pont kp

  Középpont.
- Pont p

  A síkidom egy csúcsa.

4.5.1. Részletes leírás

Paraméterek

Negyzet	osztály. A Negyzet osztály a Sikidom leszármazottja, mely egy négyzet síkidomot reprezentál, középpontjával és egy csúcsával.

4.5.2. Konstruktorok és destruktorok dokumentációja

4.5.2.1. Negyzet() [1/2]

```
Negyzet::Negyzet (
            Pont _kp = Pont(0, 0),
            Pont _p = Pont(0, 0) )  [inline]
```

4.5.2.2. Negyzet() [2/2]

```
Negyzet::Negyzet (
            const Negyzet & _n )  [inline]
```

4.5.2.3. ~Negyzet()

```
Negyzet::~Negyzet ( )  [inline], [override]
```

4.5.3. Tagfüggvények dokumentációja

4.5.3.1. Kivul()

```
bool Negyzet::Kivul (
            const std::size_t r ) const  [override], [virtual]
```

Negyzet Kivul függvény. Eldönti, hogy a négyzet kívül van-e egy adott sugarú, origo középpontú körön.

Paraméterek

r	a kör sugara.

Visszatérési érték

true, ha a négyzet kívül van a körön, egyébként false.

Megvalósítja a következőket: Sikidom.

4.5.3.2. Rajtavan()

```
bool Negyzet::Rajtavan (
            const Pont & P ) const  [override], [virtual]
```

Negyzet Rajtavan függvény. A függvény eldönti, hogy egy pont rajta van-e a négyzeten.

Paraméterek

P	a vizsgált pont.

Visszatérési érték

true, ha a pont rajta van a négyzeten, egyébként false.

Megvalósítja a következőket: Sikidom.

4.5.3.3. Read()

```
void Negyzet::Read (
            std::istream & is )  [override], [virtual]
```

Negyzet beolvasó függvény. A függvény beolvassa a négyzet adatait a megadott input streamből.

Paraméterek

is	a bemeneti stream.

Megvalósítja a következőket: Sikidom.

4.5.3.4. Terulet()

```
double Negyzet::Terulet ( ) const  [override], [virtual]
```

Negyzet területét számoló függvény. A kiszamitas modja: R^2 * 2.

Visszatérési érték

A négyzet területe.

Megvalósítja a következőket: Sikidom.

4.5.3.5. Write()

```
void Negyzet::Write (
            std::ostream & os ) const  [override], [virtual]
```

Negyzet kiíró függvény. A függvény kiírja a négyzet adatait a megadott output streambe.

Paraméterek

os	a kimeneti stream.

Megvalósítja a következőket: Sikidom.

4.5.4. Barát és kapcsolódó függvények dokumentációja

4.5.4.1. operator<<

```
std::ostream & operator<< (
            std::ostream & ,
            const Negyzet &  )  [friend]
```

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- src/sikidom.h
- src/sikidom.cpp

4.6. Pont osztályreferencia

Pont osztály A pontokat a síkon tárolja, x és y koordinátákkal.

```
#include <pont.h>
```

Publikus tagfüggvények

- Pont (const double _x=0, const double _y=0)
- Pont (const Pont &)

  Pont osztály másoló konstruktora.
- Pont & operator= (const Pont &)

  Pont osztály értékadó operátora.
- Pont & setx (double)

  Pont osztály x koordinátájának beállítása.
- Pont & sety (double)

  Pont osztály y koordinátájának beállítása.
- double getx () const

  Pont osztály x koordinátájának lekérdezése.
- double gety () const

  Pont osztály y koordinátájának lekérdezése.
- Pont operator+ (const Pont &) const

  Pont osztály összeadó operátora.
- Pont operator- (const Pont &) const

  Pont osztály kivonó operátora.
- bool operator== (const Pont &) const

  Pont osztály egyenlőség operátora.
- bool operator!= (const Pont &) const

  Pont osztály egyenlőtlenség operátora.
- double dst (const Pont &) const

  Pont osztály távolság számító metódusa.
- Pont & Mozgat (const double _x=0, const double _y=0)

  Pont osztály mozgató metódusa.
- Pont & Forgat (const double, const Pont &)

  Pont osztály forgató metódusa.

Privát attribútumok

- double x

  Pont x koordinátája.
- double y

  Pont y koordinátája.

Barátok

- std::ostream & operator<< (std::ostream &, const Pont &)

  Pont osztály kiíró operátora.
- std::istream & operator>> (std::istream &, Pont &)

  Pont osztály beolvasó operátora. formátumok: "(x,y)" vagy "x,y".

4.6.1. Részletes leírás

Pont osztály A pontokat a síkon tárolja, x és y koordinátákkal.

4.6.2. Konstruktorok és destruktorok dokumentációja

4.6.2.1. Pont() [1/2]

```
Pont::Pont (
            const double _x = 0,
            const double _y = 0 )  [inline]
```

4.6.2.2. Pont() [2/2]

```
Pont::Pont (
            const Pont & p )
```

Pont osztály másoló konstruktora.

Paraméterek

p	másolandó pont

4.6.3. Tagfüggvények dokumentációja

4.6.3.1. dst()

```
double Pont::dst (
            const Pont & p ) const
```

Pont osztály távolság számító metódusa.

Paraméterek

p	a másik pont

Visszatérési érték

a két pont távolsága

4.6.3.2. Forgat()

```
Pont & Pont::Forgat (
            const double rad,
            const Pont & center = Pont(0, 0) )
```

Pont osztály forgató metódusa.

Paraméterek

rad	forgatás szöge radiánban
center	a forgatás középpontja

Visszatérési érték

∗this

4.6.3.3. getx()

```
double Pont::getx ( ) const
```

Pont osztály x koordinátájának lekérdezése.

Visszatérési érték

x koordináta értéke

4.6.3.4. gety()

```
double Pont::gety ( ) const
```

Pont osztály y koordinátájának lekérdezése.

Visszatérési érték

y koordináta értéke

4.6.3.5. Mozgat()

```
Pont & Pont::Mozgat (
            const double _x = 0,
            const double _y = 0 )
```

Pont osztály mozgató metódusa.

Paraméterek

↩ _↩ *x*	x irányú elmozdulás
↩ _↩ *y*	y irányú elmozdulás

Visszatérési érték

∗this

4.6.3.6. operator"!=()

```
bool Pont::operator!= (
            const Pont & p ) const
```

Pont osztály egyenlőtlenség operátora.

Paraméterek

p	a másik pont

Visszatérési érték

igaz, ha a két pont koordinátái nem egyenlők, egyébként hamis.

4.6.3.7. operator+()

```
Pont Pont::operator+ (
            const Pont & b ) const
```

Pont osztály összeadó operátora.

Paraméterek

b	a másik pont

Visszatérési érték

az összeg

4.6.3.8. operator-()

```
Pont Pont::operator- (
            const Pont & b ) const
```

Pont osztály kivonó operátora.

Paraméterek

b	a másik pont

Visszatérési érték

a különbség

4.6.3.9. operator=()

```
Pont & Pont::operator= (
            const Pont & p )
```

Pont osztály értékadó operátora.

Paraméterek

p	másolandó pont

Visszatérési érték

∗this

4.6.3.10. operator==()

```
bool Pont::operator== (
            const Pont & p ) const
```

Pont osztály egyenlőség operátora.

Paraméterek

p	a másik pont

Visszatérési érték

igaz, ha a két pont koordinátái egyenlők, egyébként hamis.

4.6.3.11. setx()

```
Pont & Pont::setx (
            double _x )
```

Pont osztály x koordinátájának beállítása.

Paraméterek

↩ _↩ *x*	beállítandó x koordináta

Visszatérési érték

∗this

4.6.3.12. sety()

```
Pont & Pont::sety (
            double _y )
```

Pont osztály y koordinátájának beállítása.

Paraméterek

↩ _↩ *y*	beállítandó y koordináta

Visszatérési érték

∗this

4.6.4. Barát és kapcsolódó függvények dokumentációja

4.6.4.1. operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const Pont & p )  [friend]
```

Pont osztály kiíró operátora.

Paraméterek

os	output stream referencia
p	a kiírandó pont

Visszatérési érték

output stream referencia

4.6.4.2. operator>>

```
std::istream & operator>> (
            std::istream & is,
            Pont & p )  [friend]
```

Pont osztály beolvasó operátora. formátumok: "(x,y)" vagy "x,y".

Paraméterek

is	input stream referencia
p	a beolvasandó pont

Visszatérési érték

input stream referencia

4.6.5. Adattagok dokumentációja

4.6.5.1. x

```
double Pont::x  [private]
```

Pont x koordinátája.

4.6.5.2. y

```
double Pont::y  [private]
```

Pont y koordinátája.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

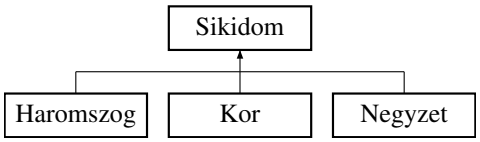
- src/pont.h
- src/pont.cpp

4.7. Sikidom osztályreferencia

Sikidom osztály. A sikidom osztály egy absztrakt osztály, melynek leszármazottai a különböző (szabályos) síkidomokat reprezentálják, középpontjukkal és egy csúcsukkal.

`#include <sikidom.h>`

A Sikidom osztály származási diagramja:



Publikus tagfüggvények

- Sikidom (const Sikidom &)
- Sikidom & operator= (const Sikidom &)

  Sikidom osztaly egyenloseg operatora.
- Pont getkp () const
- Pont getp () const
- bool Tartalmazza (const int) const
- Sikidom & Mozgat (const int _x, const int _y)

  Egy sokszög mozgatását teszi lehetővé.
- Sikidom & Forgat (const double, const Pont &)

  Egy sokszög forgatását teszi lehetővé.
- virtual double Terulet () const =0

  A sokszög területét adja vissza.
- virtual bool Rajtavan (const Pont &) const =0
- virtual void Write (std::ostream &os) const =0
- virtual void Read (std::istream &is)=0
- virtual bool Kivul (const std::size_t r) const =0
- virtual ~Sikidom ()=0

  Sikidom osztály destruktora. nincs dinamikusan foglalt adattag, tehát nem kell semmit felszabadítani.

Statikus publikus tagfüggvények

- static Sikidom ∗ createSikidom (const std::string &type)

  Sikidom osztály statikus factory függvénye. A függvény létrehoz egy új síkidomot a megadott típus alapján.

Védett tagfüggvények

- Sikidom (Pont _kp=Pont(0, 0), Pont _p=Pont(0, 0))

  Sikidom konstruktor.

Védett attribútumok

- Pont kp

  Középpont.
- Pont p

  A síkidom egy csúcsa.

Barátok

- std::ostream & operator<< (std::ostream &, const Sikidom ∗const)

  Sikidom osztály kiíró operátora. A függvény kiírja a síkidomot a megadott output streambe.
- std::istream & operator>> (std::istream &is, Sikidom ∗∗sikidom)

  Sikidom osztály beolvasó operátora. A függvény beolvassa a síkidomot a megadott input streamből.

4.7.1. Részletes leírás

Sikidom osztály. A sikidom osztály egy absztrakt osztály, melynek leszármazottai a különböző (szabályos) síkidomokat reprezentálják, középpontjukkal és egy csúcsukkal.

4.7.2. Konstruktorok és destruktorok dokumentációja

4.7.2.1. Sikidom() [1/2]

```
Sikidom::Sikidom (
            Pont _kp = Pont(0, 0),
            Pont _p = Pont(0, 0) )  [inline], [protected]
```

Sikidom konstruktor.

Paraméterek

_kp	középpont.
_p	egy csúcs.

4.7.2.2. Sikidom() [2/2]

```
Sikidom::Sikidom (
            const Sikidom & s )
```

Sikidom osztaly Copy konstruktora.

Paraméterek

s	Referencia a lemasolando Sikidomra

4.7.2.3. ~Sikidom()

```
Sikidom::~Sikidom ( )  [pure virtual]
```

Sikidom osztály destruktora. nincs dinamikusan foglalt adattag, tehát nem kell semmit felszabadítani.

4.7.3. Tagfüggvények dokumentációja

4.7.3.1. createSikidom()

```
Sikidom * Sikidom::createSikidom (
            const std::string & s )  [static]
```

Sikidom osztály statikus factory függvénye. A függvény létrehoz egy új síkidomot a megadott típus alapján.

Paraméterek

s	a létrehozandó síkidom típusa szöveg formátumban.

Visszatérési érték

A létrehozott síkidomra mutató pointer.

4.7.3.2. Forgat()

```
Sikidom & Sikidom::Forgat (
            const double rad,
            const Pont & center = Pont(0, 0) )
```

Egy sokszög forgatását teszi lehetővé.

Sikidom osztály forgató függvénye. A függvény elforgatja a síkidomot <rad> radiánnal.

Paraméterek

deg	forgatás mértéke, fokban.
rad	forgatás mértéke, radiánban.
center	a forgatás középpontja.

Visszatérési érték

Referencia a forgatott síkidomra.

4.7.3.3. getkp()

```
Pont Sikidom::getkp ( ) const  [inline]
```

4.7.3.4. getp()

```
Pont Sikidom::getp ( ) const  [inline]
```

4.7.3.5. Kivul()

```
virtual bool Sikidom::Kivul (
            const std::size_t r ) const  [pure virtual]
```

Megvalósítják a következők: Kor, Haromszog és Negyzet.

4.7.3.6. Mozgat()

```
Sikidom & Sikidom::Mozgat (
            const int _x,
            const int _y )
```

Egy sokszög mozgatását teszi lehetővé.

Sikidom osztály mozgató függvénye. A függvény elmozgatja a síkidomot az x és y értékekkel.

Paraméterek

↩ _↩ *x*	x tengely irányú mozgatás mértéke.
↩ _↩ *y*	y tengely irányú mozgatás mértéke.
↩ _↩ *x*	x tengely irányú mozgatás mértéke.
↩ _↩ *y*	y tengely irányú mozgatás mértéke.

Visszatérési érték

Referencia a mozgatott síkidomra.

4.7.3.7. operator=()

```
Sikidom & Sikidom::operator= (
            const Sikidom & s )
```

Sikidom osztaly egyenloseg operatora.

Paraméterek

s	

Visszatérési érték

Az egymásután fűzhető műveletek miatt Sikidom referencia.

4.7.3.8. Rajtavan()

```
virtual bool Sikidom::Rajtavan (
            const Pont &  ) const  [pure virtual]
```

Megvalósítják a következők: Kor, Haromszog és Negyzet.

4.7.3.9. Read()

```
virtual void Sikidom::Read (
            std::istream & is )  [pure virtual]
```

Megvalósítják a következők: Kor, Haromszog és Negyzet.

4.7.3.10. Tartalmazza()

```
bool Sikidom::Tartalmazza (
            const int  ) const
```

4.7.3.11. Terulet()

```
virtual double Sikidom::Terulet ( ) const  [pure virtual]
```

A sokszög területét adja vissza.

Megvalósítják a következők: Kor, Haromszog és Negyzet.

4.7.3.12. Write()

```
virtual void Sikidom::Write (
            std::ostream & os ) const  [pure virtual]
```

Megvalósítják a következők: Kor, Haromszog és Negyzet.

4.7.4. Barát és kapcsolódó függvények dokumentációja

4.7.4.1. operator<<

```
std::ostream & operator<< (
            std::ostream & os,
            const Sikidom * const sikidom )  [friend]
```

Sikidom osztály kiíró operátora. A függvény kiírja a síkidomot a megadott output streambe.

Paraméterek

os	a kimeneti stream.
sikidom	a kiírandó síkidomra mutató pointer.

Visszatérési érték

A kimeneti stream referenciája.

4.7.4.2. operator>>

```
std::istream & operator>> (
            std::istream & is,
            Sikidom ** sikidom )  [friend]
```

Sikidom osztály beolvasó operátora. A függvény beolvassa a síkidomot a megadott input streamből.

Paraméterek

is	a bemeneti stream.
sikidom	a beolvasott síkidomra mutató pointer.

Visszatérési érték

A bemeneti stream referenciája.

4.7.5. Adattagok dokumentációja

4.7.5.1. kp

```
Pont Sikidom::kp  [protected]
```

Középpont.

4.7.5.2. p

```
Pont Sikidom::p  [protected]
```

A síkidom egy csúcsa.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- src/sikidom.h
- src/sikidom.cpp

4.8. Tomb< T > osztálysablon-referencia

Generikus Dinamikus tömb osztály. A Tomb osztály egy dinamikus tömböt reprezentál, melynek mérete a felhasználó által nem korlátozott.

```
#include <tomb.hpp>
```

Osztályok

- class const_iterator

  const_iterator osztály. Az const_iterator osztály a Tomb osztály const_iteratora.
- class iterator

  iterator osztály. Az iterator osztály a Tomb osztály iteratora.

Publikus tagfüggvények

- Tomb ()

  Tomb osztály konstruktora. A konstruktor létrehoz egy üres tömböt, 1 kapacitással.
- iterator begin ()

  begin iterator.
- iterator end ()

  end iterator.
- const_iterator begin () const

  begin const_iterator.
- const_iterator end () const

  end const_iterator.
- size_t size () const

  Tomb méretét visszaado függvény.
- size_t capacity () const

  Tomb kapacitásat visszaado függvény.
- T & at (const size_t i)

  Tomb at függvénye.
- T & operator[ ] (const size_t i)

  Tomb operator[] függvénye.
- const T & at (const size_t i) const

  Tomb const at függvénye.
- const T & operator[ ] (const size_t i) const

  Tomb const operator[] függvénye.
- void resize (size_t newSize)

  Tomb resize függvénye. A függvény a tömb méretét növeli a megadott értékkel.
- void push_back (T value)

  Tomb push_back függvénye. A függvény a tömb végére fűz egy elemet.
- int find (const T &value) const

  Tomb find függvénye. A függvény megkeresi az első olyan elemet, mely megegyezik az értékkel.
- virtual ~Tomb ()

  Tomb destruktora. A destruktor felszabadítja a tömbot.

Privát attribútumok

- T * t

  A dinamikusan foglalt tömb elejére mutató pointer.
- size_t currentSize

  A tömb aktuális mérete.
- size_t currentCapacity

  A tömb aktuális kapacitása.

4.8.1. Részletes leírás

template<class T>
class Tomb< T >

Generikus Dinamikus tömb osztály. A Tomb osztály egy dinamikus tömböt reprezentál, melynek mérete a felhasználó által nem korlátozott.

4.8.2. Konstruktorok és destruktorok dokumentációja

4.8.2.1. Tomb()

```
template<class T >
Tomb< T >::Tomb ( )  [inline]
```

Tomb osztály konstruktora. A konstruktor létrehoz egy üres tömböt, 1 kapacitással.

4.8.2.2. ~Tomb()

```
template<class T >
virtual Tomb< T >::~Tomb ( )  [inline], [virtual]
```

Tomb destruktora. A destruktor felszabadítja a tömbot.

4.8.3. Tagfüggvények dokumentációja

4.8.3.1. at() [1/2]

```
template<class T >
T & Tomb< T >::at (
            const size_t i )  [inline]
```

Tomb at függvénye.

Paraméterek

i	Az index, ahol lévő az elemet szeretnénk elérni.

Visszatérési érték

Az i-edik elem.

4.8.3.2. at() [2/2]

```
template<class T >
const T & Tomb< T >::at (
            const size_t i ) const  [inline]
```

Tomb const at függvénye.

Paraméterek

i	Az index, ahol lévő elemet szeretnénk elérni.

Visszatérési érték

const referencia az i-edik elemre.

4.8.3.3. begin() [1/2]

```
template<class T >
iterator Tomb< T >::begin ( )  [inline]
```

begin iterator.

Visszatérési érték

Az első elemre mutató iterator.

4.8.3.4. begin() [2/2]

```
template<class T >
const_iterator Tomb< T >::begin ( ) const  [inline]
```

begin const_iterator.

Visszatérési érték

Az első elemre mutató const_iterator.

4.8.3.5. capacity()

```
template<class T >
size_t Tomb< T >::capacity ( ) const  [inline]
```

Tomb kapacitását visszaado függvény.

Visszatérési érték

A tömb kapacitása.

4.8.3.6. end() [1/2]

```
template<class T >
iterator Tomb< T >::end ( )  [inline]
```

end iterator.

Visszatérési érték

Az utolsó elem után mutató iterator.

4.8.3.7. end() [2/2]

```
template<class T >
const_iterator Tomb< T >::end ( ) const  [inline]
```

end const_iterator.

Visszatérési érték

Az utolsó elem után mutató const_iterator.

4.8.3.8. find()

```
template<class T >
int Tomb< T >::find (
            const T & value ) const  [inline]
```

Tomb find függvénye. A függvény megkeresi az első olyan elemet, mely megegyezik az értékkel.

Paraméterek

value	Az érték, melyet keresünk.

Visszatérési érték

Az első olyan elem indexe, mely megegyezik az értékkel, ha nincs ilyen, akkor -1.

4.8.3.9. operator[]() [1/2]

```
template<class T >
T & Tomb< T >::operator[] (
            const size_t i )  [inline]
```

Tomb operator[] függvénye.

Paraméterek

i	Az index, ahol lévő elemet szeretnénk elérni.

Visszatérési érték

Az i-edik elem.

4.8.3.10. operator[]() [2/2]

```
template<class T >
const T & Tomb< T >::operator[] (
            const size_t i ) const  [inline]
```

Tomb const operator[] függvénye.

Paraméterek

i	Az index, ahol lévő elemet szeretnénk elérni.

Visszatérési érték

const referencia az i-edik elemre.

4.8.3.11. push_back()

```
template<class T >
void Tomb< T >::push_back (
            T value )  [inline]
```

Tomb push_back függvénye. A függvény a tömb végére fűz egy elemet.

Paraméterek

value	Az érték, melyet a tömb végére fűzünk.

4.8.3.12. resize()

```
template<class T >
void Tomb< T >::resize (
            size_t newSize )  [inline]
```

Tomb resize függvénye. A függvény a tömb méretét növeli a megadott értékkel.

Paraméterek

newSize	A novelendo meret.

4.8.3.13. size()

```
template<class T >
size_t Tomb< T >::size ( ) const  [inline]
```

Tomb méretét visszaado függvény.

Visszatérési érték

A tömb mérete.

4.8.4. Adattagok dokumentációja

4.8.4.1. currentCapacity

```
template<class T >
size_t Tomb< T >::currentCapacity  [private]
```

A tömb aktuális kapacitása.

4.8.4.2. currentSize

```
template<class T >
size_t Tomb< T >::currentSize  [private]
```

A tömb aktuális mérete.

4.8.4.3. t

```
template<class T >
T* Tomb< T >::t  [private]
```

A dinamikusan foglalt tömb elejére mutató pointer.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- src/tomb.hpp

5. fejezet

Fájlok dokumentációja

5.1. src/pont.cpp fájlreferencia

```
#include "pont.h"
#include <cmath>
#include <stdexcept>
```

Függvények

- std::ostream & operator<< (std::ostream &os, const Pont &p)

  Pont osztály kiíró operátora.
- std::istream & operator>> (std::istream &is, Pont &p)

  Pont osztály beolvasó operátora. formátumok: "(x,y)" vagy "x,y".
- double dst (const Pont &a, const Pont &b)

  két pont közötti távolság számító függvény.

5.1.1. Függvények dokumentációja

5.1.1.1. dst()

```
double dst (
            const Pont & a,
            const Pont & b )
```

két pont közötti távolság számító függvény.

Paraméterek

a	egyik pont
b	másik pont

Visszatérési érték

a két pont távolsága

5.1.1.2. operator<<()

```
std::ostream & operator<< (
            std::ostream & os,
            const Pont & p )
```

Pont osztály kiíró operátora.

Paraméterek

os	output stream referencia
p	a kiírandó pont

Visszatérési érték

output stream referencia

5.1.1.3. operator>>()

```
std::istream & operator>> (
            std::istream & is,
            Pont & p )
```

Pont osztály beolvasó operátora. formátumok: "(x,y)" vagy "x,y".

Paraméterek

is	input stream referencia
p	a beolvasandó pont

Visszatérési érték

input stream referencia

5.2. src/pont.h fájlreferencia

```
#include <iostream>
```

Osztályok

- class Pont

  Pont osztály A pontokat a síkon tárolja, x és y koordinátákkal.

Függvények

- double dst (const Pont &a, const Pont &b)

  két pont közötti távolság számító függvény.

5.2.1. Függvények dokumentációja

5.2.1.1. dst()

```
double dst (
            const Pont & a,
            const Pont & b )
```

két pont közötti távolság számító függvény.

Paraméterek

a	egyik pont
b	másik pont

Visszatérési érték

a két pont távolsága

5.3. pont.h

Ugrás a fájl dokumentációjához.

```
00001 #include <iostream>
00002
00003 #ifndef SIKIDOM_PONT_H
00004 #define SIKIDOM_PONT_H
00005
00010 class Pont{
00011 private:
00015   double x;
00016
00020   double y;
00021 public:
00022   Pont(const double _x = 0, const double _y = 0): x(_x), y(_y){}
00023
00024   Pont(const Pont&);
00025
00026   Pont& operator=(const Pont&);
00027
00028   Pont& setx(double);
00029
00030   Pont& sety(double);
00031
00032   double getx() const;
00033
00034   double gety() const;
00035
00036   Pont operator+(const Pont&) const;
00037
00038   Pont operator-(const Pont&) const;
00039
00040   bool operator==(const Pont&) const;
00041
00042   bool operator!=(const Pont&) const;
00043
00044   double dst(const Pont&) const;
00045
00046   Pont& Mozgat(const double _x = 0, const double _y = 0);
```

```
00047
00048   Pont& Forgat(const double, const Pont&);
00049
00050   friend std::ostream& operator«(std::ostream&, const Pont&);
00051
00052   friend std::istream& operator»(std::istream&, Pont&);
00053 };
00054
00055 double dst(const Pont& a, const Pont& b);
00056
00057 #endif //SIKIDOM_PONT_H
```

5.4. src/sikidom.cpp fájlreferencia

```
#include <cmath>
#include <istream>
#include <algorithm>
#include "pont.h"
#include "sikidom.h"
```

Függvények

- std::istream & operator>> (std::istream &is, Sikidom **sikidom)

  Sikidom osztály beolvasó operátora. A függvény beolvassa a síkidomot a megadott input streamből.
- std::ostream & operator<< (std::ostream &os, const Sikidom *const sikidom)

  Sikidom osztály kiíró operátora. A függvény kiírja a síkidomot a megadott output streambe.
- bool IsOnTriangle (const Pont &P, const Pont &A, const Pont &B, const Pont &C)

  IsOnTriangle függvény. A függvény eldönti, hogy egy pont rajta van-e egy háromszögön.

5.4.1. Függvények dokumentációja

5.4.1.1. IsOnTriangle()

```
bool IsOnTriangle (
            const Pont & P,
            const Pont & A,
            const Pont & B,
            const Pont & C )
```

IsOnTriangle függvény. A függvény eldönti, hogy egy pont rajta van-e egy háromszögön.

Paraméterek

P	a vizsgált pont.
A	háromszög egyik csúcsa.
B	háromszög másik csúcsa.
C	háromszög harmadik csúcsa.

Visszatérési érték

true, ha a pont rajta van a háromszögön, egyébként false.

A kiszamitas modja: https://en.wikipedia.org/wiki/Barycentric_coordinate_system#Determining_location_with_respect_to_a_triangle

5.4.1.2. operator<<()

```
std::ostream & operator<< (
            std::ostream & os,
            const Sikidom *const sikidom )
```

Sikidom osztály kiíró operátora. A függvény kiírja a síkidomot a megadott output streambe.

Paraméterek

os	a kimeneti stream.
sikidom	a kiírandó síkidomra mutató pointer.

Visszatérési érték

A kimeneti stream referenciája.

5.4.1.3. operator>>()

```
std::istream & operator>> (
            std::istream & is,
            Sikidom ** sikidom )
```

Sikidom osztály beolvasó operátora. A függvény beolvassa a síkidomot a megadott input streamből.

Paraméterek

is	a bemeneti stream.
sikidom	a beolvasott síkidomra mutató pointer.

Visszatérési érték

A bemeneti stream referenciája.

5.5. src/sikidom.h fájlreferencia

```
#include "pont.h"
#include <cstddef>
#include <iostream>
#include <memory>
```

Osztályok

- class Sikidom

  Sikidom osztály. A sikidom osztály egy absztrakt osztály, melynek leszármazottai a különböző (szabályos) síkidomokat reprezentálják, középpontjukkal és egy csúcsukkal.
- class Kor

  Kor osztály. A Kor osztály a Sikidom leszármazottja, mely egy kör síkidomot reprezentál, középpontjával és egy csúcsával.
- class Haromszog

  Haromszog osztály. A Haromszog osztály a Sikidom leszármazottja, mely egy háromszög síkidomot reprezentál, középpontjával és egy csúcsával.
- class Negyzet

Függvények

- bool IsOnTriangle (const Pont &, const Pont &, const Pont &, const Pont &)

  IsOnTriangle függvény. A függvény eldönti, hogy egy pont rajta van-e egy háromszögön.

5.5.1. Függvények dokumentációja

5.5.1.1. IsOnTriangle()

```
bool IsOnTriangle (
            const Pont & P,
            const Pont & A,
            const Pont & B,
            const Pont & C )
```

IsOnTriangle függvény. A függvény eldönti, hogy egy pont rajta van-e egy háromszögön.

Paraméterek

P	a vizsgált pont.
A	háromszög egyik csúcsa.
B	háromszög másik csúcsa.
C	háromszög harmadik csúcsa.

Visszatérési érték

true, ha a pont rajta van a háromszögön, egyébként false.

A kiszamitas modja: https://en.wikipedia.org/wiki/Barycentric_coordinate_system#Determining_location_with_respect_to_a_triangle

5.6. sikidom.h

Ugrás a fájl dokumentációjához.

```
00001 #include "pont.h"
```

```
00002 #include <cstddef>
00003 #include <iostream>
00004 #include <memory>
00005
00006 #ifndef SIKIDOM_SIKIDOM_H
00007 #define SIKIDOM_SIKIDOM_H
00008
00009
00015 class Sikidom {
00016 protected:
00019   Pont kp;
00020
00023   Pont p;
00024
00029   Sikidom(Pont _kp = Pont(0, 0), Pont _p = Pont(0, 0)) : kp(_kp), p(_p) {}
00030
00031 public:
00032   Sikidom(const Sikidom &);
00033
00034   Sikidom &operator=(const Sikidom &);
00035
00036   Pont getkp() const { return kp; }
00037
00038   Pont getp() const { return p; }
00039
00040   bool Tartalmazza(const int) const;
00041
00046   Sikidom &Mozgat(const int _x, const int _y);
00047
00051   Sikidom &Forgat(const double, const Pont &);
00052
00055   virtual double Terulet() const = 0;
00056
00057   virtual bool Rajtavan(const Pont &) const = 0;
00058
00059   virtual void Write(std::ostream& os) const = 0;
00060
00061   virtual void Read(std::istream& is) = 0;
00062
00063   static Sikidom* createSikidom(const std::string& type);
00064
00065   virtual bool Kivul(const std::size_t r) const = 0;
00066
00067   friend std::ostream &operator«(std::ostream&, const Sikidom * const);
00068
00069   friend std::istream& operator»(std::istream& is, Sikidom** sikidom);
00070
00071   virtual ~Sikidom() = 0;
00072 };
00073
00079 class Kor : public Sikidom {
00080 public:
00081   Kor(Pont _kp = Pont(0, 0), Pont _p = Pont(0, 0)) : Sikidom(_kp, _p) {}
00082
00083   Kor(const Kor &_k) : Sikidom(_k) {}
00084
00085   double Terulet() const override;
00086
00087   bool Rajtavan(const Pont &) const override;
00088
00089   void Write(std::ostream& os) const override;
00090
00091   void Read(std::istream& is) override;
00092
00093   bool Kivul(const std::size_t r) const override;
00094
00095   friend std::ostream &operator«(std::ostream &, const Kor &);
00096
00097   ~Kor() override {}
00098 };
00099
00105 class Haromszog : public Sikidom {
00106 public:
00107   Haromszog(Pont _kp = Pont(0, 0), Pont _p = Pont(0, 0)) : Sikidom(_kp, _p) {}
00108
00109   Haromszog(const Haromszog &_h) : Sikidom(_h) {}
00110
00111   double Terulet() const override;
00112
00113   bool Rajtavan(const Pont &) const override;
00114
00115   void Write(std::ostream& os) const override;
00116
00117   void Read(std::istream& is) override;
00118
00119   bool Kivul(const std::size_t r) const override;
00120
```

```
00121   friend std::ostream &operator«(std::ostream &, const Haromszog &);
00122
00123   ~Haromszog() override {}
00124 };
00125
00131 class Negyzet : public Sikidom {
00132 public:
00133   Negyzet(Pont _kp = Pont(0, 0), Pont _p = Pont(0, 0)) : Sikidom(_kp, _p) {}
00134
00135   Negyzet(const Negyzet &_n) : Sikidom(_n) {}
00136
00137   double Terulet() const override;
00138
00139   bool Rajtavan(const Pont &) const override;
00140
00141   void Write(std::ostream& os) const override;
00142
00143   void Read(std::istream& is) override;
00144
00145   bool Kivul(const std::size_t r) const override;
00146
00147   friend std::ostream &operator«(std::ostream &, const Negyzet &);
00148
00149   ~Negyzet() override {}
00150 };
00151
00152 bool IsOnTriangle(const Pont&, const Pont&, const Pont&, const Pont&);
00153
00154 #endif // !SIKIDOM_SIKIDOM_H
```

5.7. src/tomb.hpp fájlreferencia

```
#include <stdexcept>
```

Osztályok

- class Tomb< T >

  Generikus Dinamikus tömb osztály. A Tomb osztály egy dinamikus tömböt reprezentál, melynek mérete a felhasználó által nem korlátozott.
- class Tomb< T >::iterator

  iterator osztály. Az iterator osztály a Tomb osztály iteratora.
- class Tomb< T >::const_iterator

  const_iterator osztály. Az const_iterator osztály a Tomb osztály const_iteratora.

5.8. tomb.hpp

Ugrás a fájl dokumentációjához.

```
00001 #ifndef MYTOMB
00002 #define MYTOMB
00003
00004 #include <stdexcept>
00005
00010 template <class T> class Tomb {
00013   T *t;
00016   size_t currentSize;
00019   size_t currentCapacity;
00020
00021 public:
00022   class iterator;
00023   class const_iterator;
00024
00028   Tomb() : t(new T[1]), currentSize(0), currentCapacity(1) {}
00029
00034   iterator begin() { return iterator(t, 0); }
00035
00040   iterator end() { return iterator(t, currentSize); }
```

```
00041
00046   const_iterator begin() const { return const_iterator(t, 0); }
00047
00052   const_iterator end() const { return const_iterator(t, currentSize); }
00053
00057   size_t size() const { return currentSize; }
00058
00062   size_t capacity() const { return currentCapacity; }
00063
00068   T &at(const size_t i) {
00069     if (i >= currentSize)
00070       throw std::out_of_range("Array.at(): invalid index");
00071     return t[i];
00072   }
00073
00078   T &operator[](const size_t i) { return at(i); }
00079
00084   const T &at(const size_t i) const {
00085     if (i >= currentSize)
00086       throw std::out_of_range("Array.at(): invalid index");
00087     return t[i];
00088   }
00089
00094   const T &operator[](const size_t i) const { return at(i); }
00095
00100   void resize(size_t newSize) {
00101     if (newSize > currentCapacity) {
00102       T* newT = new T[newSize * 2];
00103       for (size_t i = 0; i < currentSize; ++i) {
00104         newT[i] = t[i];
00105       }
00106       delete[] t;
00107       t = newT;
00108       currentCapacity = newSize * 2;
00109     }
00110   }
00111
00116   void push_back(T value) {
00117     if (currentSize == currentCapacity) {
00118       resize(currentSize * 2);
00119     }
00120     t[currentSize++] = value;
00121   }
00122
00128   int find(const T& value) const {
00129     for (size_t i = 0; i < currentSize; ++i) {
00130       if (t[i] == value) {
00131         return i;
00132       }
00133     }
00134     return -1;
00135   }
00136
00141   class iterator {
00145     T *p;
00146
00150     size_t idx;
00151
00152   public:
00157     iterator(T *p = nullptr, size_t idx = 0) : p(p), idx(idx) {}
00158
00162     T &operator*() { return p[idx]; }
00163
00167     iterator &operator++() {
00168       ++idx;
00169       return *this;
00170     }
00171
00176     bool operator!=(const iterator &other) const { return idx != other.idx; }
00177   };
00178
00183   class const_iterator {
00187     const T *p;
00188
00192     size_t idx;
00193
00194   public:
00199     const_iterator(const T *p = nullptr, size_t idx = 0) : p(p), idx(idx) {}
00200
00204     const T &operator*() const { return p[idx]; }
00205
00209     const_iterator &operator++() {
00210       ++idx;
00211       return *this;
00212     }
00213
00218     bool operator!=(const const_iterator &other) const { return idx != other.idx; }
```

```
00219   };
00220
00224   virtual ~Tomb() {
00225     delete[] t;
00226   }
00227 };
00228
00229 #endif
```

Tárgymutató